AN ABSTRACT OF THE THESIS OF

Stephen David Louis Snider for the degree of Master of Science in Mechanical Engineering presented on April 18, 2008.

Title: Detection and Analysis of Separated Flow Induced Vortical Structures

Abstract approved: _____

Sourabh V. Apte

Flow separation is an important phenomenon in fluid dynamics because of the effect it has on lift and drag on immersed bodies. Areas of swirl within a separated flow region may have a distinct effect on the surface forces, modifying the lift and drag characteristics. A correlation between the passage of vortical structures and the surface pressure can be used to determine locations on the surface most affected by separation and swirls in the flow. These locations can be used to place sensors to detect any variations in flow patterns that can be used to control lift and drag.

Unsteady separated flow over a square cylinder and a thin airfoil at high angle of attack are investigated using large eddy simulation. A full three-dimensional simulation is performed using high performance parallel computing. The flow Reynolds number is on the order of 10^4 in both cases. At this Reynolds number, both flows contain separation and periodic vortex

shedding over the surface of the object. The effect of these vortical structures in each flow is analyzed using different vortex detection techniques.

Four methods of vortex detection are investigated and compared: (i) the eigenvalue method (λ_2), (ii) eigenvalue of the Hessian of pressure (λ_p), (iii) the Γ function, and (iv) Γ_p, which is the Γ function applied to the rotated pressure gradient. Both λ_2 and λ_p detect vortical structures by locating local pressure minima and use gradient fields. The Γ function is the area averaged circulation around each point in the flow. The Γ_p function shows locations in the flow where the pressure gradient is strongest based on the area integral of the rotated pressure gradient.

The eigenvalue methods tend to detect the vortex cores and small scale features in the flow because these methods are based on derivatives of flow variables, so are most sensitive to changes on the order of the grid size. The features detected with λ_2 are similar in size and location to those detected with λ_p. Both Γ and Γ_p tend to locate large swirls and group small flow features into larger regions of swirl. They are integrated approaches most sensitive to changes on the order of the size of the integral area. However, Γ_p tends to identify more individual features than Γ because it is based on pressure derivatives, so is also sensitive to changes on the order of the grid size. All vortex detection methods tested are used to track flow structures over time.

A two-point covariance between surface pressure and flow swirl is found to be periodic and linked to the oscillatory nature of the flow. In the mean, the correlation is shown to be strongest in regions where the time-averaged Γ

magnitude is the highest. These results can be expanded to other immersed bodies, with the future goal of developing a control scheme for flight.

Detection and Analysis of Separated Flow Induced Vortical Structures

by

Stephen David Louis Snider

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented April 18, 2008
Commencement June 2008

Master of Science thesis of Stephen David Louis Snider presented on April 18, 2008.

APPROVED:

Major Professor, representing Mechanical Engineering

Head of the School of Mechanical, Industrial and Manufacturing Engineering

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Stephen David Louis Snider, Author

ACKNOWLEDGEMENTS

I'd like to acknowledge the support I've received from the other students in our CFD group: Ehsan Shams Sobhani, Mathieu Martin, and Justin Finn. Their help with coding, debugging, and general fluids knowledge has improved my research in more ways than I can count. Thank you to my adviser of the last two years, Dr. Sourabh Apte. Dr. Apte allowed me a great deal of freedom with my research and encouraged me to try many new approaches and methods.

My thanks also go to Dr. Deb Pence, who is the professor who first piqued my interest in fluids way back in ME331, dragged me into CFD in ME567 and first introduced me to Dr. Apte. Without her teaching and support, I would not be allowed to play with supercomputers and Fortran. Thank you to Dr. Jim Liburdy, Dr. Eugene Zhang, Daniel Morse, and Guoning Chen for their help with vortex detection methods and flow analysis techniques.

Thanks to both of my parents for their encouragement, support and proofreading skills. Finally, thank you to my fiancée Carol, who over the last three years has put up with my long hours in front of the computer both at school and at home. She deserves as much credit as I for helping me proofread papers and listening to me try to work out issues with research, coding, or classes.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

DEDICATION

This thesis is dedicated to L.W. "Doc" Lepkowski, a mentor who taught me to think critically, work hard, and never stop doing what you love.

Chapter 1 – Introduction

Separating flow is a common topic in fluid dynamics because of the effect it has on the surface forces of bodies in a flow. Despite extensive research in the field of separating flow, little is known about the exact effect of vortex passage on surface forces. There have been studies to show the gross effects of separating flow on lift and drag, and research focused on controlling the separating flow over bluff bodies [Cheng and Chen, 2007], [Choi et al., 2008], but few studies on how the flow features within the separated region individually affect surface forces.

In this research, a case is selected that features massive flow separation: flow over a square cylinder. The Reynolds number in this flow is low, $Re = 21000$, but because of the sharp edges and bluff profile, the flow separates at the leading edges of the cylinder and remains separated, generating a large wake. This wake is observed to oscillate with a fixed period. This oscillation has a large effect on the size and shape of the separation above (below) the top (bottom) surfaces of the square cylinder. Along the edge of these separated regions there are vortices that are shed from the leading edge and roll up due to shear layer instabilities.

Another favorite subject of study in fluid dynamics is the detection and tracking of vortices in a flow. While there is still much debate on what constitutes a vortex in the flow from a mathematical standpoint, most authors agree that a vortex is characterized by a swirl around a pressure minima in the flow. There are several

methods for vortex detection including the Γ function [Graftieaux et al., 2001] and the critical point and eigenvalue methods [Chong et al., 1990], [Jeong and Hussain, 1995]. The weakness of previous methods stems from a lack of information about the pressure within the flow. While the Γ function looks at the swirl directly, and the eigenvalue method attempts to locate pressure minima based on velocity gradients, neither method is designed to look for pressure minima directly.

Flow over a square cylinder is an ideal case for developing and evaluating vortex detection methods. The massive separation leads to vortex generation along the free shear layer, while the sharp leading edges cause vortex shedding. These vortices are convected along the edge of the separated region and into the wake. As the flow evolves, a large recirculating region will develop in the near wake. Since these vortices are of different scales in space and time, this flow is a robust test case for vortex detection methods.

In the interest of combining separating flow analysis with vortex detection, computational fluid dynamics (CFD) is used exclusively to gather data for this research. The main advantage of using CFD is the ability to directly compute the pressure and its gradient at all points in the field. CFD also has the advantage of being able to compute gradients and other derived data at run-time, reducing the amount of post-processing that must be done for vortex detection. Another advantage of CFD in an academic setting is the relatively low cost of performing many simulations as opposed to the time and materials required for a similar quantity of experimental tests.

With CFD there is a tradeoff between the computational power required to

solve a flow and the accuracy of the end solution. The potentially most accurate solution with CFD is a direct numerical simulation (DNS) in which the Navier-Stokes equations are solved at each grid point. DNS is the most accurate method of solving flow numerically, as it exactly solves the Navier-Stokes equations for each grid cell. However, as Reynolds number increases, so does the number of grid cells required to accurately model the flow physics. While laminar flow can be solved using a relatively low number of grid points, fully turbulent flow even for simple geometry can require a prohibitively large number of grid points when using DNS. Recent advances in turbulence models, such as large eddy simulation (LES) and Reynolds-averaged Navier-Stokes (RANS) models have helped reduce this cost by easing grid resolution requirements.

In LES, a filter is defined with size greater than the grid size. The velocity field is then filtered to separate the large scales of turbulence, which are directly solved, from the small eddies, which must be modeled. Eddies that are smaller than the filter size are most often modeled using an eddy viscosity that dissipates energy in the flow. For a perfect model, the energy dissipated through the artificial eddy viscosity would exactly match the energy that would be dissipated by all swirls smaller than the filter size. In practice, there are several ways of defining the eddy viscosity, from setting a constant eddy viscosity based on *a priori* knowledge of the flow physics, to letting the simulation define a dynamic eddy viscosity based on the physics at each grid cell in the flow. These models have been shown to perform well in turbulent and separating flows, but still require additional computing cost to solve the turbulence models. This added cost to solve the turbulence models is

typically much lower than the cost for a full DNS.

Another type of turbulence model that is widely used is the Reynolds-averaged Navier-Stokes (RANS). The idea behind RANS is that most of the time engineers are interested in the average flow, such as the mean flow rate through a pipe or the mean pressure distribution on a bluff body, and are less interested in the fluctuations about the mean. Simulations using RANS provide a fast solution for the mean variables in a flow by modeling the effect of the transient fluctuations. In a statistically steady flow the mean value of a flow variable will not change with time and the time-average of its fluctuations will be zero. In these types of flow, each variable can be broken down into its mean value plus a fluctuation. By replacing each variable in the Navier-Stokes equation with its mean plus its fluctuation, the result is the RANS equations.

After multiplying terms and simplifying the equations, there is one additional term for which a model is needed. This term is the time-average of the product of the fluctuations of velocity (e.g. $\overline{u'u'}$, $\overline{u'v'}$), also known as the correlation. These terms introduce additional variables into the set of equations without additional equations to solve for them. This is called the closure problem, a problem that is solved with RANS models. One common type of RANS model is known as the k-ϵ model, and uses the kinetic energy, k a length scale, and a dissipation, ϵ, to define an eddy viscosity. This eddy viscosity is then used to calculate the fluctuation terms in the RANS equations. The main weakness of RANS, at least with this particular model, is that it requires a very fine grid near walls, and even with such a fine grid it may not model near-wall effects accurately in separated flow.

Data collected by CFD are more complete than experimental data because of the ability to store the pressure, flow variable gradients, and many derived data at each time step. The availability of the data makes CFD an attractive choice when attempting to define new vortex detection routines, or when trying to correlate vortex location to surface forces. For example, one could modify the eigenvalue method of vortex detection to look directly at the Hessian of the pressure, rather than relying on velocity derivatives to approximate those quantities.

The first goal of the research presented herein is to develop and evaluate new methods of vortex detection in separating turbulent flow. Vortex detection methods will be evaluated on their ability to detect and track vortices on different scales in both space and time. New vortex detection techniques based on the pressure gradient vector field are presented and compared to existing vortex detection schemes. The second goal of this research is to apply a vortex detection scheme to a separating flow, and use the results of the vortex detection and the pressure on the surface of a bluff body to develop a correlation between the passage of vortices and the surface forces.

The motivation for such a correlation is to be able to predict modification of lift and drag based on flow feature detection. A possible application for such a correlation is for the control of a micro air vehicle (MAV). The operator of an MAV would be able to predict the lift of the vehicle based on its angle of attack and air speed. The angle of attack and air speed would determine the amount and type of separation. The correlation would help predict the surface forces based on the separation.

Another application for this correlation would be in separation control. One could measure the surface forces directly and deduce what features the separating flow may contain, then change angle of attack or air speed, or use passive or active suction and/or blowing to change the separation. One specific application for this correlation is determining an ideal location for a pressure sensor on the surface that would give the most information about what is happening in the flow. An ideal location would be where the correlation magnitude is large, on average, as that would indicate a strong effect of the vortex passage and the surface pressure in that segment.

Chapter 2 – Literature Review

The literature review is divided into three different subject areas: cylinders in external flow, numerical simulation, and vortex detection. When discussing the cylinders, two cases will be examined, one of a circular cylinder and one of a square cross-section cylinder. Both cases will be discussed primarily from a numerical standpoint, but will highlight some of the findings of experimental studies as well. In the section covering numerical methods, several models will be discussed, some which are used in the current research while others are presented as groundwork for the modern models. Finally, the vortex detection will cover some of the current techniques in detecting swirling structures in separated flows.

2.1 Cylinder in cross flow

Flow past a circular cylinder is a widely studied and modeled flow in the field of fluid dynamics. It has been studied for well over a century, so it is an ideal starting case for any study that focuses on flow separation. It is well known that flow past a circular cylinder at Reynolds numbers above approximately 180 becomes three dimensional and at Re greater than 300 multiple vortex shedding modes exist. In one study by Thompson et al. (1996), the importance of modeling the flow as three dimensional is discussed, and their numerical predictions are better

when performing full three-dimensional simulation than the two-dimensional cases [Thompson et al., 1996]. Massively separating at high Re, the cylinder case is ideal for studying vortex shedding and detection methods. However, a slightly better case is that of the square cylinder.

A square cylinder is expected to have the same flow topology as a circular cylinder in cross flow, but with different length and velocity scales. Specifically, the wake of a square cylinder has higher mixing and is wider than that of the circular cylinder [Lyn et al., 1995]. Two benchmark studies have been performed that are used to validate and verify other experimental and numerical studies. The first of these cases is that of Durão et al. (1988), who experimentally studied flow over a square cylinder at Re=14,000. Their study is performed in a narrow-span water tunnel to emulate two-dimensional flow conditions, with data taken using laser Doppler velocimetry. Their focus was on understanding the periodic nature of the separation around the square cylinder. By performing a frequency analysis on the velocity signal from the data, Durão and his colleagues report a Strouhal number of 0.13, and report a recirculation length of L/D = 0.83 [Durão et al., 1988]. The Strouhal number, St, is defined as $\frac{fD}{u_\infty}$, where f is the frequency, D is the cylinder side length, and u_∞ is the inlet velocity. The recirculation length L/D is defined with L as the distance from the back face of the cylinder.

The second benchmark experimental study is that of Lyn et al. (1995). Lyn and his colleagues used laser-Doppler velocimetry to collect data for fully three-dimensional flow around a square cylinder at Re = 21,400, comparing their results with several previous studies including the aforementioned Durão case and an

experimental study of a circular cylinder in cross flow. Their reported Strouhal number of 0.13 matches well with previous square cylinder results, as does their recirculation length of L/D of about 0.9 in the cylinder wake. The Strouhal number for both square cylinder cases is lower than the reported Strouhal numbers for circular cylinders, indicating lower frequency vortex shedding. Another difference between the square and circular cylinder cases is the recirculation length in the wake, with the circular cylinder having a shorter region of recirculation [Lyn et al., 1995].

In the workshop paper by Rodi et al. (1997), the authors compile and report on the status of large eddy simulation (LES) based on comparing simulation results with the results of the Lyn et al. (1995) study discussed above. The authors chose to simulate flow over a square cylinder because it is a complex flow with secondary separation, yet the geometry is not complex enough to require complicated grids. Comparing the results of several simulations, they note that the Strouhal number seems to be a poor indicator of the quality of a simulation, as all the simulations report similar values despite somewhat poor replication of the experimental results. Rodi et al. (1997) note that the two quantities that have the largest variance are the mean drag coefficient and the recirculation length, indicating that these are two quantities that should be compared when simulating flow over a square cylinder.[Rodi et al., 1997]

After comparing the results of the simulation, Rodi et al. (1997) suggested some reasons the simulations may not match well with the experimental data. One of the most influential variables in a simulation is the spanwise length and the resulting

grid resolution in the spanwise direction. Many of the simulations reported used coarse resolution and insufficient spanwise length to accurately model the three-dimensionality of the separated flow. The two other reasons cited are insufficient near-wall grid resolution and lack of upstream turbulence. [Rodi et al., 1997]

A number of studies focus on increasing the accuracy of simulations of flow over cylinders. One method is to perform a direct numerical simulation (DNS), which solves the Navier-Stokes equation directly. In the study by Wissink (1997), he performs a direct numerical simulation of flow over a square cylinder at Re =10,000 in two dimensions. The biggest problem with this study is lack of comparison against experimental data, which calls into question the validity of the results of his study. Wissink compares the predicted wake momentum deficit decay with an empirical correlation and found that the simulated results match well [Wissink, 1997]. Wissink also plots vorticity contours at several time steps, pointing out that even in a turbulent wake, there exist identifiable vortical structures.

After the report by Rodi et al. (1997) mentioned above, the research of Sohankar et al. (2000) worked on the same simulation, trying different subgrid scale models with LES to better match the experimental results of Lyn et al. (1995). Their study mentions complicating factors in modeling flow over a sharp-edged object, including the aforementioned massive flow separation, the inherent three-dimensionality of the flow, and the presence of sharp corners on the cylinder. Sohankar et al. (2000) perform several simulations using three subgrid scale models: the Smagorinsky, the standard dynamic model, and a new one-equation dynamic model that they introduce. The predictions using their one-equation model match

the Lyn case better than the other two models, but all three of their cases show better agreement with Lyn than those reported in the workshop paper by Rodi et al. (1997) [Sohankar et al., 2000].

Sohankar et al. (2000) also address some of the weaknesses mentioned in the workshop paper by correcting their prediction for blockage effects in the experiment. They also included more points in the spanwise direction and modeled a larger span width to allow the simulation to develop more fully in all three dimensions. The grid and method used by this study became a starting point for the model and grid used in the research for this thesis.

Another turbulence modeling method used to increase the accuracy of RANS is a technique called detached-eddy simulation (DES). DES combines the methods of RANS and LES, using LES near walls and surfaces where RANS is weak, and using RANS in the far field to describe the mean flow. While this method is not used in the current research, it is worth noting as it does provide results for massively separated flow that match closely with experimental results. The work of Travin et al. (1999) describes the method of DES and compares simulation results with experimental results for a circular cylinder at Re =50,000. Their model is shown to predict very well the Strouhal number and coefficient of lift correlation as well as the pressure coefficient around the cylinder. However, the centerline velocity profile does not match well, which may be caused by increased dissipation in the simulation, as indicated by larger Reynolds stress in the base region [Travin et al., 1999].

2.2 Numerical methods

In this section, the numerical method used in this research will be described and some literature validating this research code will be presented. The groundwork for large eddy simulation and Reynolds-averaged Navier Stokes methods will be presented as well. Since large eddy simulation is used in the current research, validation cases are presented for LES specifically meant to highlight its strengths in modeling separating or highly three dimensional flows. Specific validation cases performed for the current research will not be presented here, but will be explained in detail in a later section of this thesis.

There are many methods used to solve the Navier-Stokes equations for fluid flow, broadly categorized by the form of the equations, their grid requirements, whether the solver is implicit or explicit, and the turbulence model used (if any). For example, the code used for the simulations for the current research is an unsteady, unstructured grid, collocated grid flow solver that uses an explicit scheme to advance the solution in time. It uses an algebraic multigrid to solve the pressure equation at each time step. It is capable of performing large eddy simulation (LES) or direct numerical simulation (DNS) in highly complex geometries. [Mahesh et al., 2004]

Before delving into the details of any numerical solvers, it is important to understand the basis on which they are founded. Most numerical solvers for CFD are formulated by diving a global flow domain into many differential volumes over which the Navier-Stokes and continuity equations are solved. This method is called

finite volume analysis. Finite volume methods solve the conservation equations using an integral approach in which the flow variables are solved directly at the cell centroids by using the fluxes at the cell boundaries. The main advantage of finite volume methods is continuity over the domain is guaranteed as long as continuity is maintained in each differential volume. That is, because the integral continuity equation is solved in each differential volume, the global continuity equation is a summation of each of the differential volumes' continuity equation and is guaranteed to globally conserve mass. [Ferziger and Perić, 2002]

By using finite volume methods with optimized codes, it is possible to model simple flows at low Reynolds (Re) number on a modern personal computer, or small cluster. However, the grid resolution required to get an accurate solution increases as $Re_L^{\frac{3}{4}}$, where Re_L is a Reynolds number based on the magnitude of velocity fluctuations and the integral scale and is usually about 0.01 times the Reynolds number of the flow [Ferziger and Perić, 2002]. In turbulent flows, the direct numerical simulation of flow can require tens to hundreds of millions of grid points, which quickly increases the cost of the simulation. To reduce computational requirements, models are introduced to the system of equations that model the effect of turbulent energy dissipation without requiring that the Navier-Stokes equations be solved down to the viscous scale. One such method is large eddy simulation (LES).

In LES, the most common sub-grid scale (SGS) model is that of Smagorinsky (1963). This technique uses the grid size as a filter for the flow and models the energy dissipation of eddies on a scale about equal to and smaller than the grid

resolution by introducing an eddy viscosity. The basic idea is that energy transport and dissipation on the sub-grid scale are due to the viscosity at the smallest flow scales. As such, it makes sense to model the dissipation as some eddy viscosity multiplied by the filtered Reynolds stresses, where the filter is the grid size. The eddy viscosity can be found to be a function of the density, grid size, the filtered strain rate tensor, and a model parameter that is left free. Setting the model parameter is the focus of many studies, as it can change by orders of magnitude even within simple shear flows. To help solve this problem, a refinement to the Smagorinsky model was developed called the dynamic Smagorinsky model.

The dynamic Smagorinsky model attempts to address the problem of the non-constant model parameter mentioned above. In the work of Germano et al. (1991) a dynamic SGS stress model is developed that attempts to overcome the deficiency in defining an *ad hoc* SGS eddy viscosity for shear flows [Germano et al., 1991]. The model samples the smallest scales of eddy that are fully resolved and, using these scales, defines the sub-grid scale model parameter dynamically. To find the smallest resolved scale, a test filter is defined that is larger than the grid resolution, the flow field is filtered by both this filter and the grid filter. After both filters are applied, the grid filtered field is subtracted from the test filtered field to define the smallest eddies that can be solved numerically. The ratio of the test filter size to the grid filter size becomes the only free variable; this ratio α is defined once before the simulation is run, and has little effect on the solution for values of $\alpha > 2$ [Germano et al., 1991].

Even with the effects of the small eddies modeled instead of resolving all scales

of turbulence, complex flows can become computationally expensive very quickly. To accurately model complex geometries and flows, it is necessary to be able to use unstructured grids that can handled varied sizes, shapes, and skewness of the differential volumes. Unstructured meshes have the additional benefit of reducing grid generation time. The paper of Mahesh et al. (2004) develops an algorithm to apply LES to unstructured grids with arbitrary elements. During the formulation of the method, they discuss the importance of satisfying energy conservation in each grid cell to prevent the scheme from being too dissipative. This energy conservation in each cell is what sets this method apart from previous attempts at applying LES to an unstructured mesh [Mahesh et al., 2004], [Apte et al., 2003].

The algorithm developed in the paper by Mahesh et al. (2004) is applied to several test cases. It is first validated against laminar flow recirculation in a cavity, the Taylor problem, which has an analytical solution, and flow over a cylinder. In each case, the unstructured LES solver is shown to produce accurate results even on coarse grids. By comparing the error at each grid refinement level against the analytical solution for the Taylor problem, they show that their scheme is second order accurate overall. The sample case that is of most relevance to the current research is their test case for separating flow over a circular cylinder. The unstructured LES results are shown to be in good agreement with experimental results at similar Reynolds numbers; the simulation results match well with the global variables such as the Strouhal number, separation point, mean recirculation length, and the mean velocity and mean pressure profiles. The final validation case is flow in a coaxial combustor, which shows very good agreement in both mean

velocity profiles and turbulent kinetic energy with experimental data for the same combustor [Mahesh et al., 2004], [Apte et al., 2003].

Since the current research is focused on turbulent flow over a cylinder, albeit a square cylinder instead of round, it is of interest to evaluate the effectiveness of LES on modeling flow over a cylinder. LES with an unstructured mesh is shown to accurately model flow over a circular cylinder in Mahesh et al. (2004), indicating that it may be a good method for this research. Research by Camarri et al. (2002) examines the effect of altering different parameters in LES for flow over a square cylinder at $Re = 21000$, the same as the current research.

In the research by Camarri et al. (2002), flow over a square cylinder is modeled using a dynamic SGS model with LES. Their objective was to investigate the capabilities of LES on unstructured grids and, to that end, they modeled the flow with several dynamic models with different test filter ratios and compared the results with the experimental results in Lyn et al. (1995). The formulation of their model is slightly different than that of Mahesh et al., and is not as flexible with the unstructured grids that it will use. Specifically, the code used by Camarri et al. (2002) does not handle highly stretched grids very well, so the tests are limited to only moderately stretched grids. Their results show good agreement with the Strouhal number and mean recirculation length in the wake, but tend to underpredict the drag coefficient when compared with experimental results. As with many other LES studies, the wake velocity recovery length is much shorter for simulation than for experimental, as shown in the mean centerline velocity profile. However, the velocity profiles along the surface of the square cylinder match quite

well with experimental results. Their findings show that grid point distribution in the separation regions strongly affects the resulting predictions, while the numerical viscosity has a much smaller effect on the velocity field [Camarri et al., 2002]. This information was useful when generating the grids for the current research, as it helped determine the grid resolution both in the separation region and the wake.

There are several studies that examine the accuracy and methods of the current research code, including the aforementioned work by Mahesh et al. (2004), Ham et al. (2003), and Apte et al. (2003). In summary, it is a "parallel unstructured finite-volume code developed specifically for LES of variable density low Mach-number flows." It is designed for use on unstructured grids, can handle multi-physics flows, and has been parallelized in such a way that it has nearly ideal scaling to multiple processor machines. It uses an algebraic multigrid program called HYPRE for solving the pressure equation at each time step. The results of simulations using this code will be used to develop and test vortex detection methods, as outlined in the following sections.

2.3 Vortex detection

Based solely on the free-stream Reynolds number, the flow for the current research is laminar. However, it is well known that turbulent boundary layer separation occurs due to adverse pressure gradients near the surface of bluff bodies in external flow. In an early numerical simulation, Simpson et al. (1981) the authors modeled the separation by an adverse pressure gradient in a converging-diverging nozzle,

also experimentally obtaining the same data. Their objective was to both validate their numerical model, then use the numerical model to better understand the underlying structure of a separating boundary layer [Simpson et al., 1981]. They show that separation occurs over a region on the surface, not necessarily from a single fixed point, and that the region within the separation is characterized by high levels of turbulence. In the separation region, there is no clear way to characterize the backflow with a universal backflow equation or function, but the authors did identify three regions within the separation region: the viscous wall region, a transition zone, and the outer backflow region [Simpson et al., 1981].

Another article examining the separating boundary layer around a cylinder sought to confirm the presence of a singularity in the boundary layer. The paper by van Dommelin and Shen (1980) describes a numerical solution for a separating boundary layer as it evolves over time. Previous works had performed numerical simulations and had not found a singularity in the separating flow, but the authors explain that the previous solutions had not been allowed to run long enough [van Dommelen and Shen, 1980]. After stepping the solution through $t = 3.0s$, the authors noted the spontaneous generation of a singularity, thus resolving the debate of its existence. Since the singularity is proved to exist, the next step is to determine the form of the singularity. The authors did not conjecture as to what the singularity means in the boundary layer.

To better understand the overall physics inside the separation bubble, Maucher et al. (1999) performed a direct numerical simulation (DNS) of the laminar-turbulent transition in separation bubbles. The mean flow is compared to ex-

perimental results for flow over an airfoil, and is shown to match well until the mean velocity becomes highly negative within the separation region. The authors show that the separating boundary layer develops into a free shear layer, a phenomenon which is predicted very well by their numerical model [Maucher et al., 1999]. However, they are still lacking an understanding of the underlying physics behind the flow separation. The authors show that the transition from a laminar separation bubble to a turbulent bubble is caused by three-dimensional disturbances in the roll-up of the two-dimensional shear layer, even if they do not fully explain or understand the physical mechanism driving the disturbance [Maucher et al., 1999]

In a more recent work, Morse and Liburdy (2007) built on the separation bubble research with an experimental study of massively separating flow over a thin airfoil. Their work focused on describing the flow within and on the edge of the separation zone formed at the leading edge of the airfoil. At the high angles of attack studied, it is unlikely that the flow reattached to the surface, so the separation does not form a bubble. Rather, at the edge of separation, vorticity sheets form due to the Kelvin-Helmholtz instability between the recirculation regime and the free-shear layer. Morse and Liburdy used two vortex detection methods to detect and track vortices in separating flow, showing that a detection method based on flow swirl is feasible for this flow situation [Morse and Liburdy, 2007]. They also showed that vortices undergo up to five stages: creation, convection, stall, collision, and destruction/dissipation. By tracing vortices through time, it is possible to define a vortex convection velocity, and correlations between the vortex passing and the

unsteady velocities both within and outside the separation region.

One common feature in the separating flows modeled and discussed above is the presence of the free shear layer, which is shown to spawn vorticity sheets. To better understand the vortical structures in such a flow field, it is important to develop methods of vortex detection and tracking. One of the most used methods is to examine the eigenvalue of the velocity deformation tensor, a method introduced by Chong and Perry (1990). The authors base their detection on a three-dimensional critical point analysis of the velocity field, generalized to apply to any vector field. The critical point analysis involves calculating the eigenvalues of the rate of deformation tensor. If the eigenvalues are complex, the rate of rotation dominates the tensor and in that region there exists a vortex core [Chong et al., 1990].

The work by Jeong and Hussain (1995) explores many methods of vortex detection, explaining the weaknesses of some standard methods such as vorticity contours and pathline traces, while introducing a new method that has become somewhat standard in the fluids community. The authors point out that vorticity contours are an inadequate method of vortex identification, as areas of high shear near a fixed surface also show contours of high vorticity. However, in free shear layers, vorticity can still be used to good effect. The authors also state that the presence of a pressure minima is neither a necessary nor a sufficient condition for a vortex, as pressure minima can exist in unsteady non-swirling flows. Finally, the authors state that seeking closed pathlines may not detect all of the vortices in the flow, as it is possible that a particle in the flow may not make a complete circuit around a vortex core before being convected away. [Jeong and Hussain, 1995]

The method put forth by Jeong and Hussain (1995) is to start with the criteria for a pressure minimum within the flow and eliminate the two effects that can cause the pressure minimum to fail at vortex detection, namely the fact that viscous effects can eliminate a pressure minimum even in areas of swirl, and that unsteady straining can create pressure minimum where no swirl exist. They start with the gradient of the Navier-Stokes equation and decompose it into symmetric and antisymmetric parts, shown in Equation 2.3.1 below. The antisymmetric part of Eq. 2.3.1 is the vorticity transport equation, so is identically zero. The authors then eliminate the unsteady irrotational term, and the viscous effects of the symmetric part, shown in Equation 2.3.2. The resulting equation sets $S^2 + \Omega^2$ approximately equal to the Hessian of the pressure. The authors note that since two positive eigenvalues of the Hessian of the pressure indicates a low pressure region, two negative eigenvalues of $S^2 + \Omega^2$ should represent an approximation of the pressure minimum location. Since they eliminated transient and viscous effects, $\lambda_2 < 0$ becomes the criteria for detecting a vortex core. It is noted that in planar flow, this criterion becomes identical to the vortex detection criterion of Chong and Perry (1990) [Jeong and Hussain, 1995].

$$\left[\frac{\mathrm{D}S_{ij}}{\mathrm{D}t} + \Omega_{ik}\Omega_{kj} + S_{ik}S_{kj}\right] + \left[\frac{\mathrm{D}\Omega_{ij}}{\mathrm{D}t} + \Omega_{ik}S_{kj} + S_{ik}\Omega_{kj}\right] = -\frac{1}{\rho}p_{,ij} + \nu u_{i,jkk} \quad (2.3.1)$$

$$\frac{\mathrm{D}S_{ij}}{\mathrm{D}t} - \nu S_{ij,kk} + \Omega_{ik}\Omega_{kj} + S_{ik}S_{kj} = -\frac{1}{\rho}p_{,ij} \quad (2.3.2)$$

Another way of locating vortices, presented in Graftieaux et al. (2001), examines the velocity field directly and uses an integral method to look for regions of

swirl in the flow. The Γ function, presented in equation 2.3.3 below, is a direct measure of the tendency of the surrounding flow to swirl about a point [Graftieaux et al., 2001]. Because it uses the velocity field directly, it is a useful approach for experimental velocity data such as particle image velocimetry (PIV) and laser Doppler velocimetry (LDV), but can be applied to any vector field. Two advantages of the Γ function are that it gives some sense of the extent of the swirl around a core, and it can be used to give information about the strength of the swirl. However, since it is an area-based approach, it is slower to compute than the λ_2 method presented by Jeong and Hussain.

$$\Gamma(x,y) = \frac{1}{A_M} \int_{A_M} \frac{(\overline{PM} \times \overline{U_M}) \cdot \widehat{Z} dA}{\|\overline{PM}\| \|\overline{U_M}\|} \tag{2.3.3}$$

One paper that uses vortex detection techniques, and offers suggestions for their use, is the presentation of research by Adrian et al. (2000). The authors present several methods for decomposing complex velocity fields in such a way that it is easier to locate and evaluate the vortices within the flow, as well as methods for locating and visualizing the vortices. Decomposition methods presented include Galilean decomposition, Reynolds decomposition, large and small eddy decomposition, and proper orthogonal decomposition [Adrian et al., 1990]. While all methods have their strengths and weaknesses, filtering by spacial scale is a very useful way of extracting the swirls that are largest and most likely to have an effect on bluff bodies. After decomposing the vector field, a snapshot of fully turbulent pipe flow, the authors use the critical point analysis, λ_{ci}, from Chong

and Perry (1990) to detect the presence of vortex cores.

Large eddy decomposition as proposed in Adrian et al. (2000) is straightforward and has been used to good effect to divide the flow into several scales by Chen et al. (2007). In their research, several of the previously mentioned vortex detection methods are applied to unsteady separated flows in which the velocity fields are filtered using a Gaussian filter before the detection takes place. The authors show results for vortex detection at small and large scales for simulated flow over a square cylinder and experimental flow over an airfoil. By filtering the field using *a priori* knowledge about the expected swirl size, the flow fields are decomposed into large and small scales. When applying the vortex detection methods, it is found that the λ_{ci} criterion is best at finding small vortices with strong cores, while the Γ function seems to better locate larger vortices and their extent [Chen et al., 2008]. Using these results, the authors conclude that λ_{ci} is a better detector for local flow events and Γ is better used for global scale flow structures.

Another application of the vortex detection methods is presented by Snider et al. (2008), in which the authors present detailed detection and analysis methods of vortex detection. In addition to evaluating the use of λ_{ci} and the Γ function as vortex detection methods over time, the authors introduce two new methods of vortex detection, λ_p and Γ_p, which are a critical point analysis of the Hessian of pressure and the Γ function as applied to a rotated pressure gradient, as shown in Equation 2.3.4.

$$\Gamma_p(x,y) = \frac{1}{A_M}\int_{A_M} \frac{(\overline{PM} \times \overline{P'_M}) \cdot \widehat{Z} dA}{(\|\overline{PM}\|\|\overline{P'_M}\|)} \tag{2.3.4}$$

where $\overline{P'_M} = -(\nabla p)^{\perp}$

For experimental data of separated flow over a thin airfoil, no pressure data is available, so the standard critical point analysis and Γ function are used for vortex detection. The results of their analysis show similar results between λ_{ci} and λ_p, which is to be expected as λ_{ci} in two dimensions is meant to approximate the eigenvalue of the Hessian of pressure [Jeong and Hussain, 1995]. When comparing the Γ function with Γ_p, the authors show that Γ_p tends to detect more individual areas of swirl than the Γ function does. This can be explained by the fact that Γ_p is based on derivatives of a flow variable, making it sensitive to changes as small as the order of the grid spacing, and the Γ function, being an integrated approach of a base flow variable, is less sensitive to such small changes [Snider et al., 2008].

Chapter 3 – Numerical methods

In this chapter the specific numerical methods used to solve the simulated flows will be discussed. A brief overview of the equations for a finite volume formulation will be presented with the general form of a predictor-corrector scheme for advancing the solution in time. The formulation for large eddy simulation will also be discussed. After the numerical method is described, two validation cases and their results will be presented: the Taylor vortex case and turbulent flow in a channel. The Taylor case has an analytical solution readily available [Mahesh et al., 2004], and there is DNS data [Kim et al., 1987] for a turbulent channel, so they make good validation cases..

3.1 Finite volume, unsteady solver formulation

While the specific algorithms in the solver used for the current research are beyond the scope of this thesis, the basic formulation behind its time advancement scheme and finite volume approach will be presented here. The finite volume method starts with the Navier-Stokes equations, shown in their incompressible form as Equations 3.1.1 and 3.1.2 below. The conservation equations are solved for each control volume (CV) in a grid by using the surface integral on each face of the control volume, and are applied to the flow volume as a whole. The code uses

a predictor-corrector method with second order Adams Bashforth explicit time advancement, described in greater detail below.

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j} \tag{3.1.1}$$

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{3.1.2}$$

In Eq. 3.1.1 the density has been absorbed into the pressure term, p. The other variables are velocity, u_i, and the kinematic viscosity, ν. From left to right, the terms in the Navier-Stokes equations are the temporal variation, convective term, pressure gradient, and the viscous effects.

The conservation equations are discretized over the domain on each control volume in such a way that the surface fluxes are summed over all the faces of each control volume. The flux on each CV surface is approximated as the flux at the face centroid multiplied by the cell-face area. This flow solver uses a collocated grid, which means that the pressure, cell velocity, and any other transport properties are stored at the CV centroids (centers). Figure 3.1.1 shows a simplified two-dimensional representation of a control volume and variables as they will be used below. One major advantage of using a collocated grid is its ability to handle unstructured elements, even highly skewed elements, of both tetrahedral and hexahedral shapes. With the other common grid type, the staggered grid, the pressure and scalars are stored at the control volume circumcenter, while the velocity is stored at the faces. Staggered grids have the advantage that they discreetly con-

serve energy, but storing variables at the cell circumcenter can cause problems in highly skewed elements, as the circumcenter may lie on a cell edge, or even outside the cell. Both the collocated and staggered grid schemes discretely conserve mass over the domain.

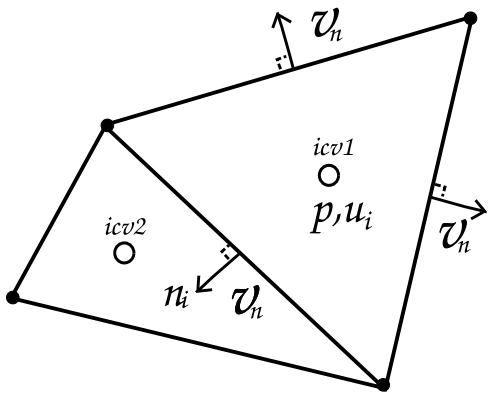


Figure 3.1.1: Two-dimensional unstructured, collocated grid with face normals and cell notation.

In addition to discrete mass conservation, discrete conservation of kinetic energy is an important feature in a discretization scheme because it will help ensure robustness of the solution. A solution that conserves kinetic energy discretely ensures that the only contribution to kinetic energy in the flow comes from the boundary elements, not from numeric error, in the limit of zero viscosity and the absence of body forces. If the scheme is formulated in this manner, the solution can be robust without the addition of numerical dissipation [Ham and Iaccarino,

2004],[Mahesh et al., 2004]. [Mahesh et al., 2006], [Apte et al., 2003]. Numerical dissipation is especially unwanted in large eddy simulation because the magnitude of numerical dissipation can rival, or even exceed the dissipation due to the eddy viscosity introduced by LES. The challenge, then, is to develop a method of solving the Navier-Stokes equations discretely while still satisfying conservation of mass, momentum, and kinetic energy.

To solve this problem, a predictor-corrector method is used that emphasizes conservation of energy for the convective and pressure terms to solve the CV center velocity at each time step. The solution is advanced in time using a second-order Adams-Bashforth scheme, which is explicit. The algorithm is outlined below, starting with the prediction of the CV center velocity based on the previous and current time step values for the convective and viscous terms in Equation 3.1.3,

$$\frac{\widehat{u}_i - u_i^k}{\Delta t} = \frac{1}{2}\left[3\left(\mathrm{NL} + \mathrm{VISC}\right)^k - \left(\mathrm{NL} + \mathrm{VISC}\right)^{k-1}\right], \tag{3.1.3}$$

where k represents the current time step (known values), $k-1$ represents the last time step, and the $\widehat{u}$ indicates the predicted velocity. For brevity, NL represents the non-linear (i.e. convective) term, and VISC represents the viscous term from the Navier-Stokes equations.

Notice that there is no pressure gradient in Eq. 3.1.3. The pressure gradient is used later to correct the predicted velocities. The viscous terms take the general form shown in Equation 3.1.4. The convective terms must be treated somewhat differently because they are nonlinear. It is shown in Mahesh et al. (2004) that a

convective term of the form shown in Equation 3.1.5 will conserve kinetic energy if $u_i|_f$ is approximated as the symmetric mean $\frac{(u_{icv1}+u_{icv2})}{2}$, even on unstructured meshes.

$$\sum_{\text{faces of cv}} \nu \left(\frac{\partial u_i}{\partial x_j}|_f + \frac{\partial u_j}{\partial x_i}|_f \right) \times N_j A_f \tag{3.1.4}$$

In Eq. 3.1.4, f indicates that the quantity is taken at the face. Velocity gradients at the face are taken as the symmetric mean of the velocity gradients at the adjoining CV centers. N_j is the face normal vector between cells, directed from $icv1$ to $icv2$, and A_f is the area of the face between the two control volumes. This notation is extended to Eq. 3.1.5, with one additional term v_n representing the face-normal velocity *out of* the control volume.

$$\sum_{\text{faces of cv}} u_f|_f v_n A_f \tag{3.1.5}$$

After using Eq. 3.1.5 and 3.1.4 to obtain a predicted velocity from Eq. 3.1.3, the predicted velocity is projected to the face centers,

$$\widehat{v} = \frac{\widehat{u}_i^{icv1} + \widehat{u}_i^{icv2}}{2} \tag{3.1.6}$$

to get the predicted face velocities. These face velocities are used to solve for the pressure gradient,

$$\frac{v_n - \widehat{v}_n}{\Delta t} = -\frac{\partial p}{\partial n} \tag{3.1.7}$$

where n indicates that the pressure gradient is in the face-normal direction.

To enforce continuity over the domain, Equation 3.1.7 is modified using the discrete form of the continuity equation to yield the Poisson equation for pressure, shown below as Equation 3.1.8.

$$\sum_{\text{faces of CV}} \frac{\partial p}{\partial N} A_f = \sum_{\text{faces of CV}} \widehat{v}_n A_f \tag{3.1.8}$$

Once this equation is solved, using an iterative method, the final step is to correct the earlier velocity prediction using the pressure. This step is shown as Equation 3.1.9. This seems like a simple matter, but problems occur because the Poisson equation solves the pressure at the faces, and Eq. 3.1.9 requires the CV center pressure.

$$\frac{u_i^{k+1} - \widehat{u}_i}{\Delta t} = -\frac{\partial p}{\partial x_i} \tag{3.1.9}$$

The simplest method of calculating the CV center pressure from the face pressure is shown as Equation 3.1.10. This formulation works well for simple flows and homogeneous turbulence on regular grids, but causes instability for more complex cases. A better method is presented in Mahesh et al. (2004) and is the method used in this solver.

$$\frac{\partial p}{\partial x_i} = \frac{1}{V} \sum_{\text{faces of CV}} p_f A_f N_i \tag{3.1.10}$$

It is shown that the pressure gradient, when projected to the CV centers in Eq. 3.1.10, contributes a non-zero term in the kinetic energy conservation equation over the domain. Mahesh et al. (2004) show that this kinetic energy error is due to the fact that $v_n \neq \left(\frac{u_i^{icv1} + u_i^{icv2}}{2}\right) n_i$ on a collocated grid. This results in an additional

term in the kinetic energy conservation equation,

$$\sum_{\text{volumes}} \sum_{\text{faces of cv}} p_{nbr} v_n A_f, \tag{3.1.11}$$

that must be minimized for stability. To make the solver as energy-conserving as possible, a least-squares approach is used to minimize

$$\sum_{\text{faces of CV}} \left(\frac{\partial p}{\partial x_i}|_{icv} n_i^{face} - \frac{\partial p}{\partial n}|_{face} \right) A_f, \tag{3.1.12}$$

which minimizes the error. This formulation is derived so that it is possible to compute $\partial p/\partial x_i$ using only the neighboring cells.

The last challenge in solving for the pressure is that the Poisson equation for pressure is elliptic. Algebraic multigrid, which uses a fictitious coarse outer mesh over the finer grid in the domain is used to speed convergence. Without multigrid, the required time steps to get a solution would be proportional to the number of cells in the domain, as the solution would normally diffuse over only one grid cell per time step. With a multigrid, the number of iterations on the finest grid becomes independent of the number of grid points, which can lead to massive performance increases. In this code the algebraic multigrid algorithm used is called *BoomerAMG*, part of the *hypre* software library [Falgout and Yang, 2002].

3.2 Large eddy simulation

Direct numerical simulation (DNS), which solves the Navier-Stokes equations exactly for each CV, is prohibitively expensive for high Reynolds number flows because of the large number of grid cells required. Large eddy simulation (LES) helps to alleviate these costs by modeling the small scale turbulence in the flow as an additional viscosity. However, setting a constant subgrid-scale (SGS) viscosity model has several weaknesses: the model must be changed for each flow, the model does not behave correctly near walls, it does not disappear for laminar flow, and it does not account for energy backscatter from the small scales to the large scales [Germano et al., 1991]. A dynamic SGS model is used to address all of these concerns.

The key idea behind the dynamic SGS model is that information from the large scale field is used to define the model for the subgrid-scale eddy energy dissipation. Before defining the dynamic model, it is necessary to explain LES in general. The derivation below can be applied to the continuity and energy equations, but herein it is limited to the non-dimensionalized, incompressible momentum equation:

$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial x_j}\left(u_i u_j\right) = -\frac{\partial p}{\partial x_i} + \frac{1}{\mathrm{Re}}\frac{\partial^2 u_i}{\partial x_j x_j}, \tag{3.2.1}$$

where u_i is the velocity in the ith direction, p is the pressure, and Re is the Reynolds number.

These equations are filtered using a top-hat filter spacial filter to divide the large scale field from the small scales that will be modeled. The large scale (filtered)

quantities are denoted with an overbar,

$$\overline{f} = \int_D G(x - x')f(x')dx', \tag{3.2.2}$$

where G is the subgrid scale spacial filter. After applying the filter, G, the momentum equation becomes

$$\frac{\partial}{\partial t}\overline{u_i} + \frac{\partial}{\partial x_j}\left(\overline{u_i u_j}\right) = -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \frac{1}{\mathrm{Re}}\frac{\partial^2 \overline{u_i}}{\partial x_j x_j}, \tag{3.2.3}$$

where $\tau_{ij} = \overline{u_i u_j} - \overline{u_i}\,\overline{u_j}$ is the subgrid scale stress, and must be modeled. This term represents the effects of the small scales in the flow that are not directly solved using the filtered Navier-Stokes equations.

From this point, it is necessary to model the SGS stress tensor τ_{ij}. The dynamic model uses information about the large scale flow to define τ_{ij}. This is done by introducing a "test filter," which is a spacial filter that has a larger filter width than the subgrid filter. This test field generates a second field with larger features than the subgrid filtered field. Denoting test filtered quantities with a hat and defining a test filtered stress leads to

$$\mathcal{T}_{ij} = \widehat{\overline{u_i u_j}} - \widehat{\overline{u}}_i \widehat{\overline{u}}_j. \tag{3.2.4}$$

The difference between the test filtered field and the subgrid filtered field,

$$\mathcal{L}_{ij} = \mathcal{T}_{ij} - \tau_{ij} = \widehat{\overline{u_i}\,\overline{u_j}} - \widehat{\overline{u}}_i \widehat{\overline{u}}_j, \tag{3.2.5}$$

is the resolved turbulent stress. These stresses are representative of the contribution to the Reynolds stresses from the scales of turbulence in the flow of size between the test filter and the subgrid scale filter widths. In other words, they are the smallest resolved scales of turbulence.

The Smagorinsky model is then used to parametrize both τ_{ij} and $\mathcal{T}_{ij}$. The anisotropic parts of $\mathcal{T}_{ij}$ and τ_{ij} are modeled with M_{ij} and m_{ij}, respectively:

$$\tau_{ij} - (\delta_{ij}/3)\tau_{kk} \simeq m_{ij} = -2C\bar{\Delta}^2|\overline{S}|\overline{S}_{ij}, \tag{3.2.6}$$

$$\mathcal{T}_{ij} - (\delta_{ij}/3)\mathcal{T}_{kk} \simeq M_{ij} = -2\hat{\bar{\Delta}}^2|\widehat{\overline{S}}|\widehat{\overline{S}}_{ij}, \tag{3.2.7}$$

where

$$\widehat{\overline{S}}_{ij} = \frac{1}{2}\left(\frac{\partial\widehat{\overline{u}}_i}{\partial x_j} + \frac{\partial\widehat{\overline{u}}_j}{\partial x_i}\right), |\widehat{\overline{S}}| = \sqrt{2\widehat{\overline{S}}_{mn}\widehat{\overline{S}}_{mn}},$$

$\overline{\Delta}$ is the characteristic subgrid scale filter width, and $\widehat{\overline{\Delta}}$ is the characteristic test filter width. By substituting Eq. 3.2.7 into Eq. 3.2.5, and contracting the result with $\overline{S}_{ij}$, it is possible to obtain an equation that can be solved for C:

$$\mathcal{L}_{ij}\overline{S}_{ij} = -2C(\hat{\bar{\Delta}}^2|\widehat{\overline{S}}|\widehat{\overline{S}}_{ij}\overline{S}_{ij} - \bar{\Delta}^2|\widehat{\overline{S}}|\widehat{\overline{S}}_{ij}\overline{S}_{ij}). \tag{3.2.8}$$

Although Eq. 3.2.8 can be solved for C directly in principle, the terms in the parenthesis on the right hand side can become zero, which would cause the equation for C to become ill-conditioned. To avoid this, the equation is spatially averaged, denoted by $\langle\rangle$, as shown in Equation 3.2.9. For example, for flow through a channel, the spacial average is taken over planes parallel to the wall, since the

flow is assumed to be a function of only the distance from the wall and time.

$$< C >= -\frac{1}{2}\frac{\left\langle \mathcal{L}_{kl}\overline{S}_{kl}\right\rangle}{\widehat{\overline{\Delta}}^2\left\langle |\widehat{\overline{S}}|\widehat{\overline{S}}_{mn}\overline{S}_{mn}\right\rangle - \overline{\Delta}^2\left\langle |\widehat{\overline{S}}|\widehat{\overline{S}}_{pq}\overline{S}_{pq}\right\rangle} \tag{3.2.9}$$

Finally, the spatially averaged equation for C is used with Eq. 3.2.7 to define the dynamic eddy viscosity subgrid-scale stress model:

$$m_{ij} = \frac{\left\langle \mathcal{L}_{kl}\overline{S}_{kl}\right\rangle}{\alpha^2\left\langle |\widehat{\overline{S}}|\widehat{\overline{S}}_{mn}\overline{S}_{mn}\right\rangle - \left\langle |\widehat{\overline{S}}|\widehat{\overline{S}}_{pq}\overline{S}_{pq}\right\rangle}|\overline{S}|\overline{S}_{ij}, \tag{3.2.10}$$

where $\alpha = \widehat{\overline{\Delta}}/\overline{\Delta} \geq 1$ is the ratio of the test filter width to the subgrid-scale filter width and is the only free parameter for the dynamic LES model. All the other variables are solved directly using the resolved scales in the large-scale flow.

It has been shown that choosing $\alpha = 2$ leads to good results [Germano et al., 1991], [Moin et al., 1991], and that above $\alpha = 2$, the quality of results tends to be independent of the choice of α [Germano et al., 1991]. This LES model has been validated for a wide range of cases, from turbulent flow through a channel [Moin et al., 1991], to flow around a cylinder [Mahesh et al., 2004] and flow through a coaxial jet combustor [Apte et al., 2003]. The solver using this LES model will be used to model turbulent flow through a channel, flow over a square cylinder and flow over an airfoil in the current research.

3.3 Validation

Before using any CFD code, commercial or research, it is vital to perform validation testing on the code to make sure it gives accurate solutions. Validation requires that one either knows an analytical solution for a flow field, or has experimental data to which the simulation will be compared. In the current research, three validation cases were run: a Taylor vortex in a periodic domain, flow through a three-dimensional channel, and flow over an airfoil. The Taylor case will be compared to the exact analytical solution to show the order of accuracy of the code. The channel case will be compared to the DNS results from Kim et al. (1987). The final validation case is flow over an airfoil with no camber at a high angle of attack. Numerical results from the airfoil case will be compared to experimental results from PIV in a wind tunnel.

3.3.1 Taylor case

As previously mentioned, the Taylor case has analytical solution that can be used to get the exact error in the simulation at any given time. This exact error can be plotted on a logarithmic plot as a function of grid size to get the order of error for the simulation. This case is run to verify that the order of accuracy in this code is second-order. The domain for the Taylor case is three dimensional, but is periodic in the z-direction to simulate a two-dimensional flow. A sample mesh is shown in Figure 3.3.1 below, with twenty control volumes on each side of the domain. Also shown in Fig. 3.3.1 are the velocity vectors in the domain plotted over contours of

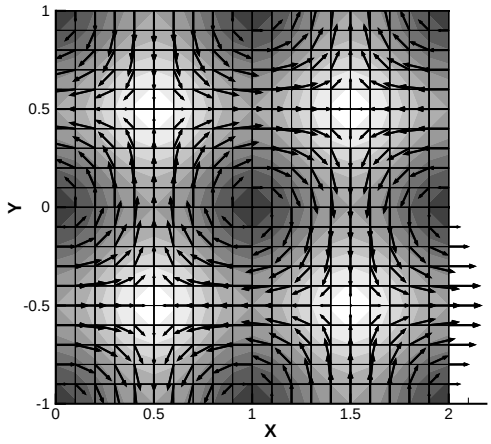


Figure 3.3.1: Sample grid for the Taylor case showing a 20x20x1 mesh, velocity vectors, and pressure contours.

the pressure. Low pressure regions indicate the center of the vortices. The domain extends from 0 to 2 in the x-direction, -1 to 1 in the y-direction, and is 0.1 units thick in the z-direction. Note that all variables are scaled by their reference values; in this case the reference length and velocity are 1m and 1m/s, respectively.

The analytical solution for the Taylor case is shown below in Equation 3.3.1. The initial condition for the simulation is applied by setting $t = 0$ in Eq. 3.3.1 and using the resulting equations to set the velocity everywhere within the domain. The z-component of velocity is set to zero. The boundary conditions are periodic in all three directions, meaning that, for example, flow out of the positive x boundary will go directly into the negative x boundary. The simulation is allowed to run

without any other velocity or pressure input until an end time of $t = 0.20$ seconds.

$$
\begin{aligned}
u(x, y, t) &= -\cos(\pi \mathrm{x})\sin(\pi \mathrm{y})\mathrm{e}^{-2\pi^2 \nu \mathrm{t}} \\
v(x, y, t) &= \sin(\pi \mathrm{x})\cos(\pi \mathrm{x})\mathrm{e}^{-2\pi^2 \nu \mathrm{t}} \\
p(x, y, t) &= -0.25[\cos(2\pi \mathrm{x}) + \cos(2\pi \mathrm{y})]\mathrm{e}^{-4\pi^2 \nu \mathrm{t}}
\end{aligned} \tag{3.3.1}
$$

In Eq. 3.3.1, ν is the dynamic viscosity, set to $\nu = 0.1$ for this simple test case. All other reference variables are set to 1.0. At each time step, the analytical solution and L_∞ error are calculated for the velocity and pressure in the field. The errors after the last time step are plotted as a function of the grid resolution on a log-log plot in Figure 3.3.2. Accompanying the data from this case is a line with a slope of 2 to indicate a perfect second-order plot. From Fig. 3.3.2, it is clear that the research code is second order accurate for Cartesian meshes.

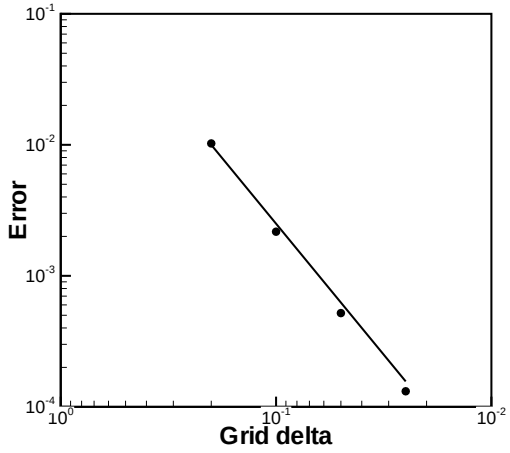


Figure 3.3.2: Error versus grid spacing - Taylor case. Symbols are current data, line is second order.

3.3.2 Channel case

The channel case is fully three-dimensional, and is periodic in the z- and x-directions. Turbulent flow through the channel is simulated with LES and compared to direct numerical simulation (DNS) results from Kim et al. (1987). The initial condition within the domain is set as a parabolic profile, and a constant pressure gradient, $dp/dx = -0.0030$, is applied as a velocity source. The parabolic profile initial condition is set with a centerline velocity such that $Re_c = 3300$ at $t = 0$. Sinusoid disturbances are initially set throughout the domain; no further perturbation is applied after $t = 0$.

The channel extends 6δ in the x-direction, 2δ in the y-direction, and 4δ in the z-direction, where δ is the half-height of the channel, x is the primary flow direction and the y-direction bounds the flow with walls. A computational grid of 60 by 120 by 60 (432,000 total) elements in the x-, y-, and z-directions is used. The grid is highly stretched near the walls of the domain and uses regularly spaced elements in the stream and span directions. The first grid cell from the wall has thickness of $y^+ = 0.5$ to help resolve the wall shear layer. The grid is partitioned into twenty pieces to run on the cluster at OSU.

Based on the channel dimensions and the initial centerline velocity, a channel flow through time is defined as $L/U_c = 6.0s$. After $t \approx 19s$ (three flow-through times) the statistics were reset and the solution was run for an additional 112 seconds, or 18 flow through times. Statistics were collected from the steady flow to compare to results from Kim et al. (1987). For validation, plots of the mean

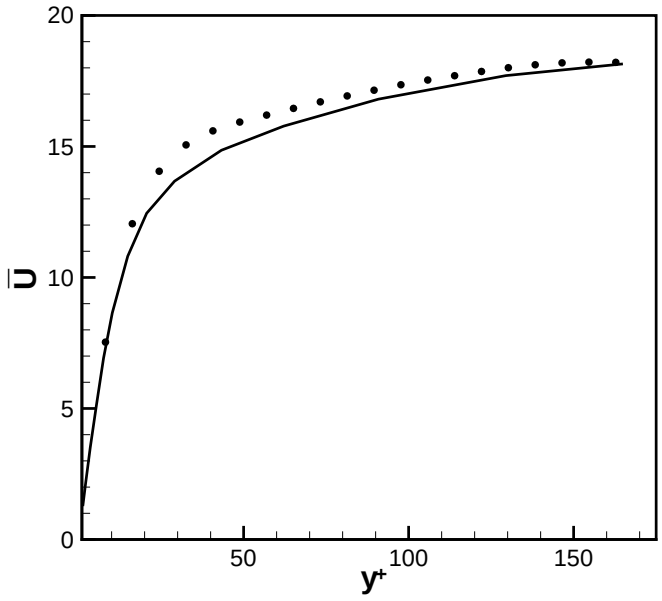


Figure 3.3.3: $\overline{U}/u_\tau$ vs y^+ velocity profile of LES (symbols) and DNS (line) flow in a turulent channel.

velocity profile as well as the RMS fluctuations are compared below. Figure 3.3.3 compares the current LES results with the DNS results for the mean velocity profile, where the mean velocity is normalized by the shear velocity.

The mean velocity profile of the LES results match very well with the DNS mean velocity, with a maximum error of approximately 7%. The Reynolds number based on the shear velocity for the LES case is $\mathrm{Re}_\tau = 169$, which is slightly lower than the reported value of $\mathrm{Re}_\tau \approx 180$ for DNS. Likewise, the Reynolds number based on the mean centerline velocity of $\mathrm{Re}_\mathrm{c} \approx 3000$ is about 10% lower in the LES prediction than the DNS case. This discrepancy is most likely due to a velocity source that is slightly too small. The final comparison is the mean streamwise

velocity fluctuations, $\overline{u'}$, normalized by the shear velocity and plotted in Figure 3.3.4.

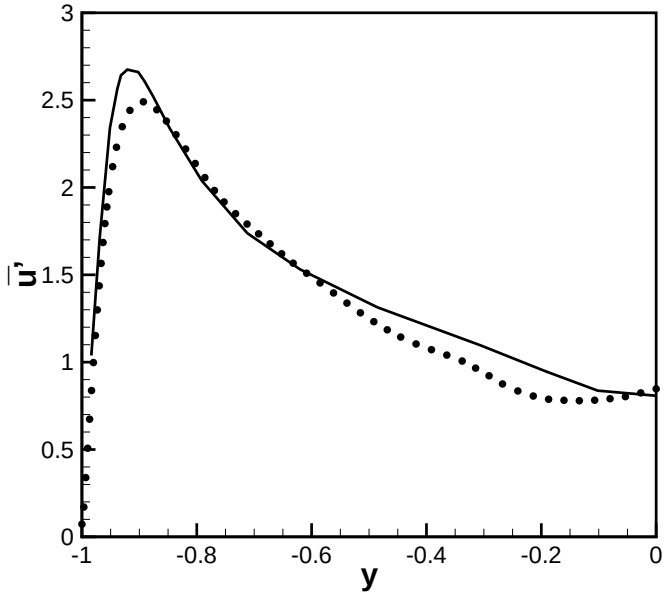


Figure 3.3.4: $\overline{u'}/u_\tau$ vs y velocity fluctuations of LES (symbols) and DNS (line) flow in a turbulent channel.

The LES predicted mean velocity fluctuations match well with the DNS predictions, as shown in Fig. 3.3.4. The magnitude of the fluctuations is slightly underpredicted near the center of the channel and at $y \approx -0.90$, but the variation from the DNS results is less than 10%. Since the LES results match well with the DNS results for the mean velocity and fluctuations as well as the predicted Reynolds numbers, flow through a turbulent channel has been successfully validated. With the LES model validated, the solver can be used to predict other flows, such as flow over an airfoil.

3.3.3 Airfoil

The final code test case is flow over an airfoil using both direct numerical simulation and large eddy simulation. For the airfoil case, the model was generated to match the experiment discussed in Morse and Liburdy (2007) and Chen et al. (2007). The airfoil is a flat plate with a chord length C=20cm, thickness h=4mm, and aspect ratio of 1:2. Both ends are composed of half of an ellipse with a major axis of 10mm and a minor axis of 4mm. Data is collected for flow at $u_\infty = 5m/s$, which gives a Reynolds number $Re \approx 66000$, and angle of attack of $\alpha = 20\,\mathrm{deg}$. The experimental data were taken at the OSU wind tunnel with a cross section of approximately three feet high by four feet wide, large enough that the walls of the airfoil did not affect the flow around the airfoil. PIV data was taken at the symmetry plane of the airfoil. The numerical simulation is compared with this PIV data for validation.

The fluid domain around the airfoil is small to reduce computational costs. The domain extends $0.75C$ upstream from the nose, $1C$ downstream from the trailing edge, and $0.75c$ above and below the surface of the airfoil. In the spanwise direction, the extent of the domain is $0.1C$, much narrower than the experimental airfoil. An illustration of the entire computational domain is shown in Figure 3.3.5.

The simulation mesh, generated using Gambit, is a C-grid around the nose and a rectangular grid for the rest of the domain. A C-grid subdomain was generated around the tail of the airfoil to better handle the elliptic trailing edge. A small region of unstructured elements patches the hexahedral elements from the C-grid

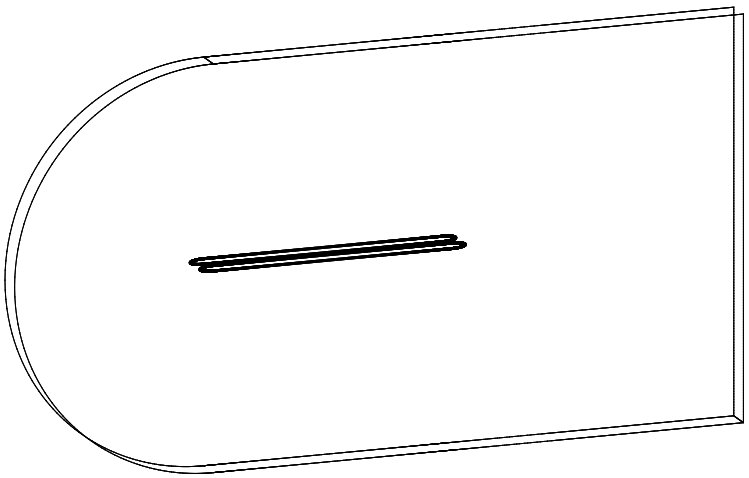
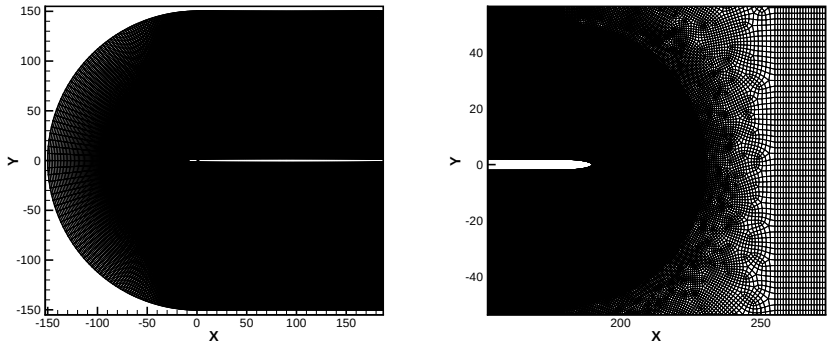


Figure 3.3.5: Airfoil computational domain.

subdomain to the hexahedral elements in the extended fluid domain. To better illustrate these features of the grid, Figure 3.3.6 shows the airfoil grid as viewed down the span of the airfoil for the C-grids near the nose and tail. The grid is generated assuming that the flow is two-dimensional, so is thin in the spanwise direction. The final grid generated contains about five million elements, 550 in the stream direction, 440 elements in the cross-stream direction, and 20 along the span. There are 1300 grid points on the surface of the airfoil in an XY plane.

The inlet velocity is applied to the C-grid and bottom of the fluid domain to simulate flow at the proper angle of attack $\alpha = 20\,\text{deg}$. The top and downstream edges of the fluid domain are set to convective outlets. The boundary condition on the airfoil itself is a no-slip wall condition. No outer wall boundary conditions are applied to the domain because of the assumption that the airfoil is far enough



(a) (b)

Figure 3.3.6: Airfoil computational grid showing the (a) C-grid around the nose; and (b) the C-grid subdomain around the tail.

from any wall to neglect wall effects. The spanwise borders are set to periodic to simulate two-dimensional flow. Because this is a large fluid domain, the grid is partitioned into 96 pieces, then transferred to the Datastar cluster at the San Diego supercomputer (SDSC) facility.

To determine an appropriate run time for this case, a flow-through time based on the domain size and the free-stream velocity was defined as $L/U_\infty = \frac{0.55m}{5.0m/s} =$ 110ms. The simulation was run for approximately five flow through times without statistics collection to allow the starting vortex and start-up effects to be convected out of the domain. After the start-up effects were shed from the domain, the simulation was run for an additional two flow-through times and was observed to have a separated region that was much larger than the separated region seen in the experimental data. In this separated region, there was also a large periodic recirculation that was not present in the experimental data.

After examining the mesh near the surface, it was decided that the mesh may not be fine enough to fully resolve all of the turbulent scales in the flow with DNS, so the dynamic Smagorinsky LES model was activated and data collection resumed. The flow was run for three additional flow-through times, which is likely not long enough to collect stationary state statistics but is long enough to draw some qualitative conclusions about the simulation. Several time-slices of the pressure field near the airfoil are shown in Figure 3.3.7 to show the large separation and vortex shedding. In Fig. 3.3.7, dark regions indicate low pressure, showing the location of centers of swirls.

Since the experimental data is taken using PIV, the simulation can only be compared to experimental in terms of velocity data. Figure 3.3.8 shows the mean velocity profile for the simulation and experimental results taken at various locations along the top surface. The y-axis represents the distance normal to the airfoil surface. For each of the three locations, $x/c = 0$, $x/c = 0.1$, and $x/c = 0.2$, the separation region is larger in the simulated flow than in the experimental results. For the two stations downstream of the leading edge, the magnitude of the negative mean velocity is lower in the simulation than the experimental, but the extent of the recirculation extends further from the airfoil surface into the flow.

Also compared in Fig. 3.3.8 are results from a two-dimensional RANS simulation of flow over the same airfoil performed using the commercial code Star-CCM+. The data is from a steady state solution; the simulation was run until the residual between iterations for mass and velocity was less than 10^{-5}. The velocity profile from the Reynolds-averaged simulation shows much larger separation than the

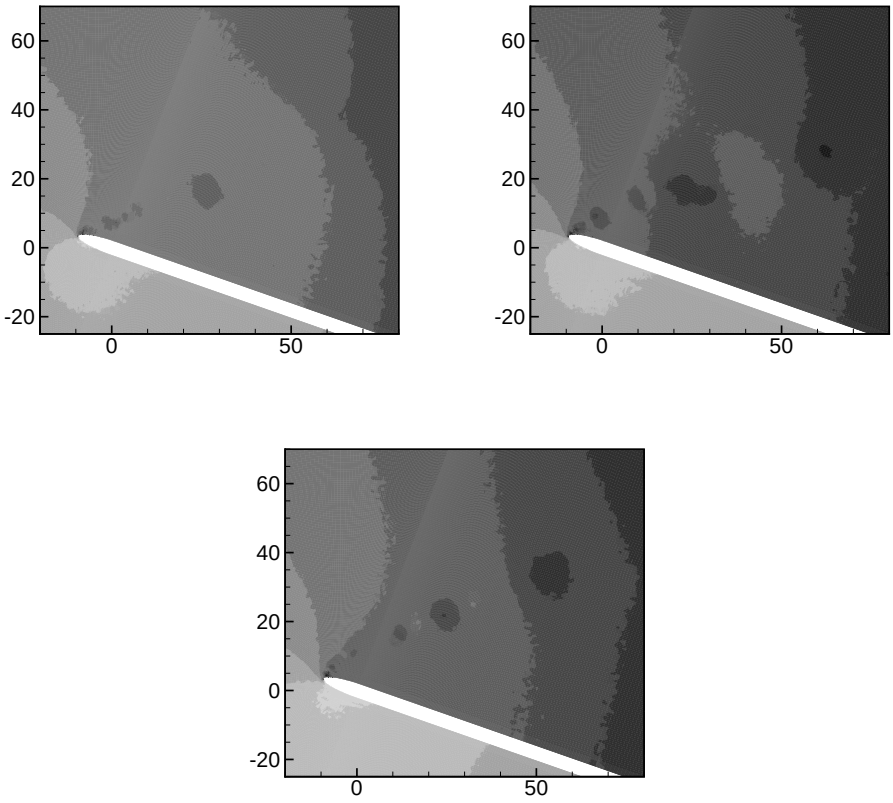


Figure 3.3.7: Contours of pressure showing leading edge vortex shedding and large separation.

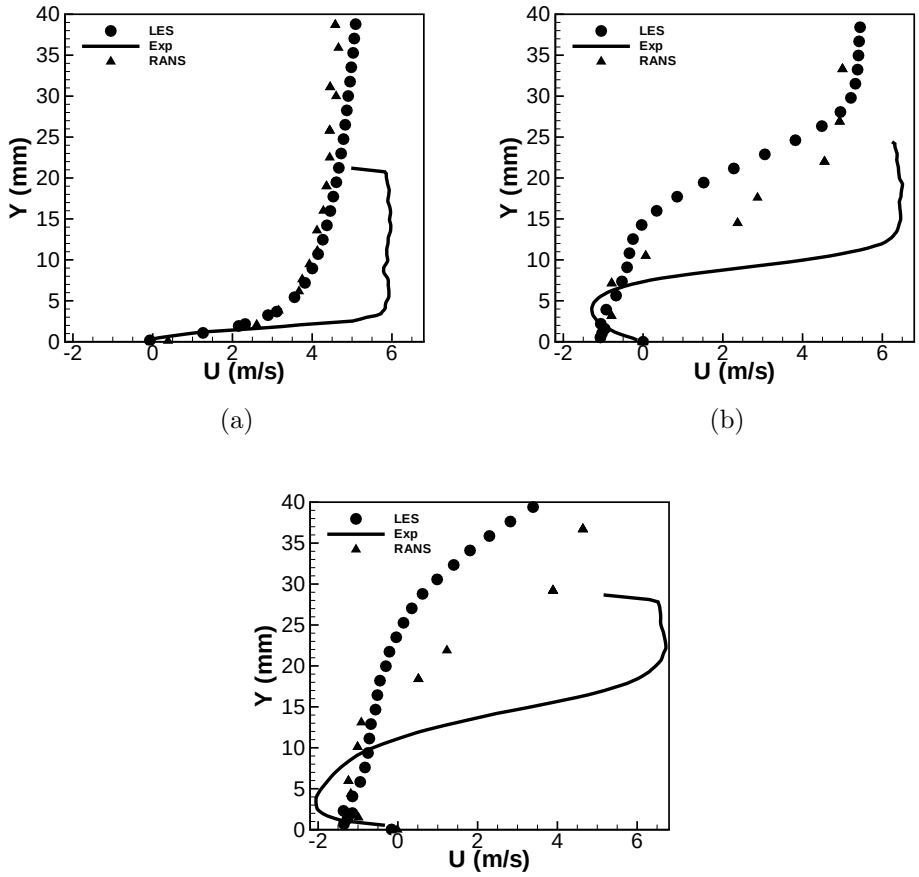


Figure 3.3.8: Comparison of the mean velocity profile at (a)$x/c = 0$, (b)$x/c = 0.1$, (c)$x/c = 0.2$ for LES, RANS, and experimental flow over an airfoil.

experimental results, but a smaller separation than the DNS/LES solution. This is one indication that the LES model was not run for sufficient time to convect starting effects out and collect statistics on the steady flow. The fact that the steady-state RANS solution also overpredicts the separation also reinforces the idea that a simulation with a thin span domain may not be able to accurately predict three-dimensional turbulence.

There are several possible reasons the predicted results do not match the experimental results. As with the turbulent channel case discussed above, the two-dimensional representation of this case limits the scale of turbulence that occurs across the span of the airfoil. This would reduce the viscous dissipation in the flow, allowing the larger recirculation region and the larger separation. The addition of subgrid scale dissipation through LES did reduce the size of separation, but even this added dissipation could not reduce the separation enough to match the experimental results.

Another possibility is that the simulation results may still be influenced by the start-up effects. The simulation was run for about 10 flow-through times, but this may not be enough to fully convect all start-up effects from the domain. For experimental data collection, the wind tunnel is running for many seconds, if not minutes, before data collection is started, ensuring that all start-up effects have been convected away from the airfoil. One other source of difference between the simulation and experimental case is turbulence in the outer flow. Although the wind tunnel incorporates flow straightening devices, the outer flow will still contain some turbulence. In the simulation, the inlet is unperturbed flow. The additional

turbulence in the outer flow would help dissipate energy, reducing the size of the recirculation near the airfoil and shrinking the flow separation.

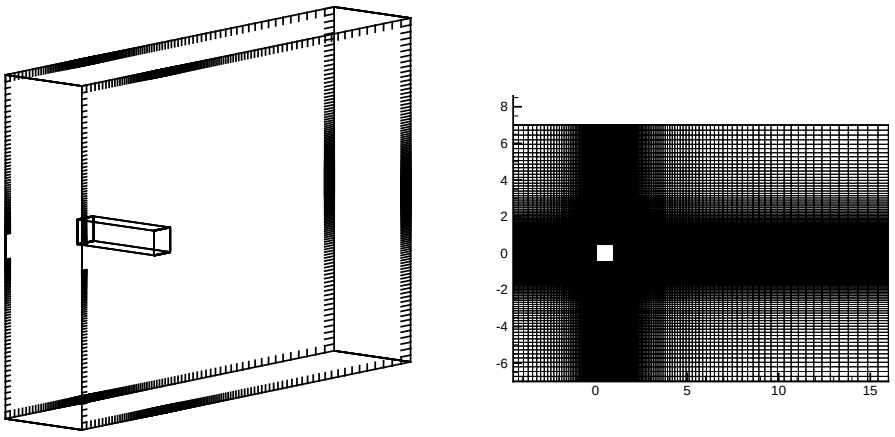
Although the airfoil case does not match the experimental results, it is still a good test case as it pointed out many model flaws that one must be aware of when modeling flow over a bluff body. It is clear that a two-dimensional representation of turbulence is inadequate when attempting to accurately predict flow behavior. Furthermore, although the airfoil was run for quite some time, it is possible that the start-up effects still influence the flow field. For future simulation, such as flow around a square cylinder studied in this research, the flow will be allowed to evolve for a longer time before collecting flow statistics, and all effort will be made to accurately model the three-dimensionality of the flow.

Chapter 4 – Simulation method and validation

The data used when developing vortex detection schemes and the correlation are from a large eddy simulation (LES) flow over a square cylinder. A square cylinder is chosen because there is extensive experimental data with which it can be validated, it has massive separation with vortex shedding, and can be modeled using a Cartesian grid of hexahedral elements. Before using the data for vortex detection, it is compared to the LES workshop results [Rodi et al., 1997] and validated against experimental results [Lyn et al., 1995].

4.1 Model and mesh

The physical model is based on the experimental setup Lyn et al. (1995), modified slightly for simulation. The exact model setup is the model used in the LES workshop organized by Rodi et al. in 1997. Figure 4.1.1 shows a representation of the geometry and mesh of this case. The fluid domain extends from $-4.5 < x < 16$ in the streamwise direction, $-7 < y < 7$ in the cross stream direction, and $-2 < z < 2$ in the spanwise direction, with the origin at the center of the upstream cylinder face. The boundary conditions are slip walls on the cross stream and spanwise walls, velocity inlet, and convective outlet. No-slip boundary conditions are applied on the surface of the square cylinder.



(a) (b)

Figure 4.1.1: Square cylinder (a) Geometry and (b) grid for simulation

.

The mesh used is generated in Gambit by creating two rectangular prisms and then using boolean subtraction to remove the square cylinder from the fluid region. The mesh is stretched using a geometric stretching with first cell thickness of 0.0125m to place more points near the square cylinder. The cylinder edges are meshed with 80 equally spaced grid points each in the streamwise and cross stream directions. The fluid domain has 230 points total in the streamwise and cross-stream directions, and 80 points in the spanwise direction. The total grid contains 4.2 million elements. This resolution is much finer than any of the previous LES studies presented in Rodi et al. (1997), especially in the spanwise direction. A finer resolution is used to help resolve the spanwise components of velocity, hopefully giving better three-dimensional results than the previous LES studies.

The inlet velocity is specified such that $Re_D = 21,000$, using the freestream velocity and square cylinder side length to define the Reynolds number. This

matches the experimental Reynolds number measured by Lyn et al. (1995), and is high enough to ensure separation from the leading edges of the square cylinder. A time step, denoted by s^*, is defined based on the cylinder side length and inlet velocity as $D/U_\infty = 1/0.3318 = 3.014s$. Finally, the flowthrough time is defined as $L_{FD}/U_\infty = 20.5/0.3318 = 61.78s$, and is the time required for flow to travel the entire length of the fluid domain. The flowthrough time is used to verify that the simulation has evolved beyond the start-up period before statistics are collected.

For running on a cluster, the domain is split into 40 sub-domains of about 100,000 elements each. This ensures that the simulation will run reasonably quickly because the number of grid cells on each processor is balanced. After the grid is divided, it is copied to the Datastar supercomputing facility in San Diego (SDSC) for extended runs. Each simulation run is ten hours in length, collecting approximately 4000 time steps of data in each run.

Several simulation runs are performed to collect over 500 seconds of data, with one second in between each frame of data output to disk. After 120 seconds, the statistics are reset to remove the start-up effects from the statistics, then are collected for another 300 seconds to encompass five flow-through times. Data is collected for 100 seconds for use in the vortex detection and correlation. An additional 45 seconds of data is collected to verify that the statistics are not changing with time.

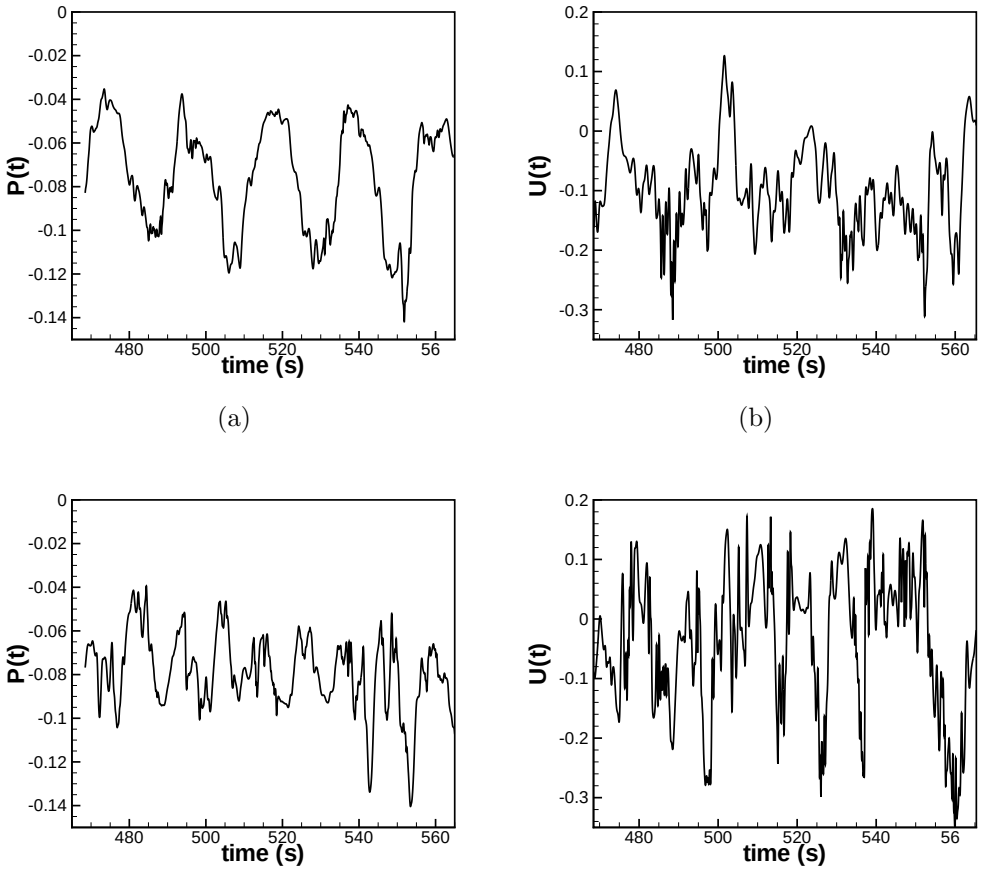
Flow statistics are collected for a total of $34.2s^*$, after which the time-based mean and RMS values were observed to not change significantly between time steps. During this simulation, probes of velocity and pressure are placed in the

flow for data collection at each time step. Frames of data were collected every 100 time steps, which gives one frame per second of simulation time. The simulation time step is fixed at 10ms so a meaningful frequency analysis can be performed on the pressure and velocity probes in the flow. A fixed time step is also useful when determining vortex convection velocity and to have physically relevant time-series analysis on vortex shedding.

4.2 Results and validation

Time series data for the streamwise velocity and pressure are shown in Figure 4.2.1 for probes taken at the midpoint on the top surface of the cylinder and at a station one cube length behind the square cylinder in the wake. A low frequency oscillation is visible in the top surface probe, with the velocity oscillation lagging the pressure oscillation. The velocity signal seems to contain more power in higher frequencies than the pressure signal, which tends to be dominated by the low frequency oscillation. In the wake, both the pressure and velocity probes show higher dominant frequencies and more power in higher frequencies than the probes on the top surface. A spectrum distribution of both signals at both points is presented in Figure 4.2.2.

The frequency plots in Fig. 4.2.2 are presented with the freqency non-dimensionalized by D/U_∞, giving the Strouhal number. The probe on the top surface is used to compare results with the experiments of Lyn et al. (1995), Rodi et al. (1997), and Durão et al. (1998), as that is where the probes are placed for their frequency



(a) (b)

(c) (d)

Figure 4.2.1: Time history plots of probes in the flow for (a) pressure at center of top surface; (b) velocity at center of top surface; (c) pressure at 1D downstream; (d) velocity at 1D downstream.

analysis. The Strouhal number of 0.1288 based on the velocity probe matches quite well with the reported Strouhal number for previous work with square cylinders. The wake probe frequency distribution is more spread out than that of the top surface, indicating fluctuations in pressure and velocity on a broader range of time scales.

Other than the Strouhal number, several other quantities are compared to the experimental results to help validate this model. The mean recirculation length in the wake is found to be equal to $\frac{L}{D} = 0.94$, which compares well to the experimental value of $\frac{L}{D} = 0.90$ [Lyn et al., 1995]. Mean streamwise velocity and velocity fluctuation profiles are compared to the experimental results in Figure 4.2.3 along the centerline $y = z = 0$ of the fluid domain.

The simulation results overpredict the streamwise velocity recovery in the wake, as well as the magnitude of the mean fluctuations on the centerline. However, the centerline mean velocity profile does match the experimental data better than previous LES studies [Rodi et al., 1997]. It is interesting that the mean streamwise velocity fluctuations are predicted well, but the mean cross stream velocity fluctuations are predicted about 50% lower than measured experimentally. The underprediction of these variations is likely related to the overprediction of the mean velocity along the centerline.

Flow profiles are also compared at a station one cylinder length downstream in the wake. Figure 4.2.4 shows mean velocity profiles and the velocity fluctuations across one half of the wake. The simulation is shown to slightly over predict the streamwise velocity across the entire wake, which is expected after looking

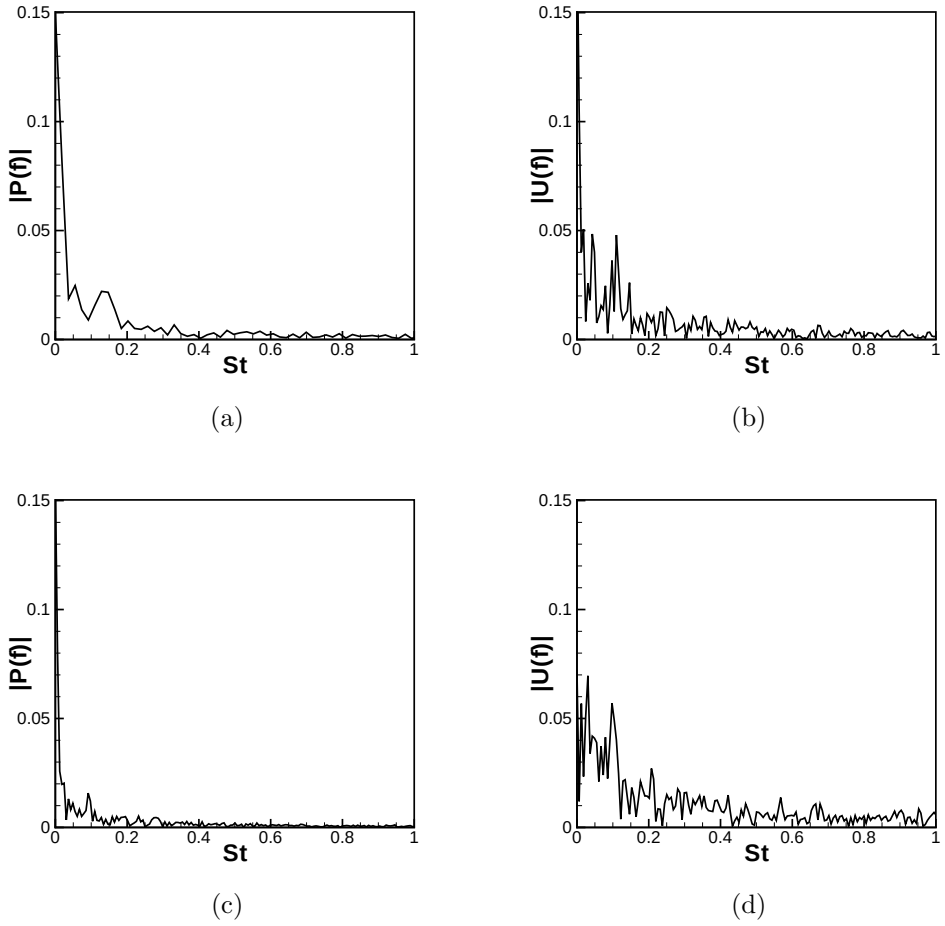
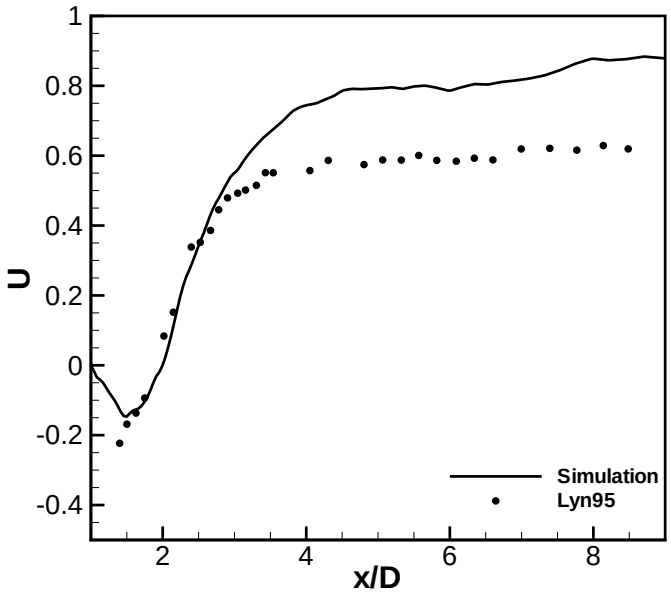
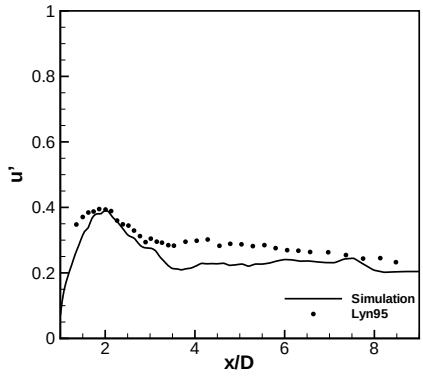


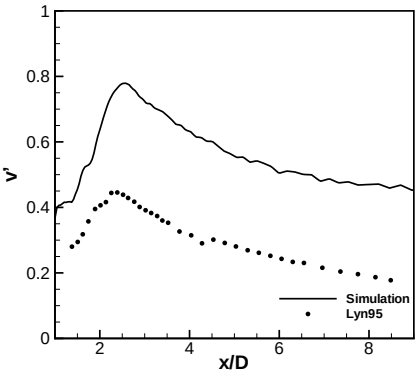
Figure 4.2.2: Frequency distribution plots of probes in the flow for (a) pressure at center of top surface; (b) velocity at center of top surface; (c) pressure at 1D downstream; (d) velocity at 1D downstream.



(a)



(b)



(c)

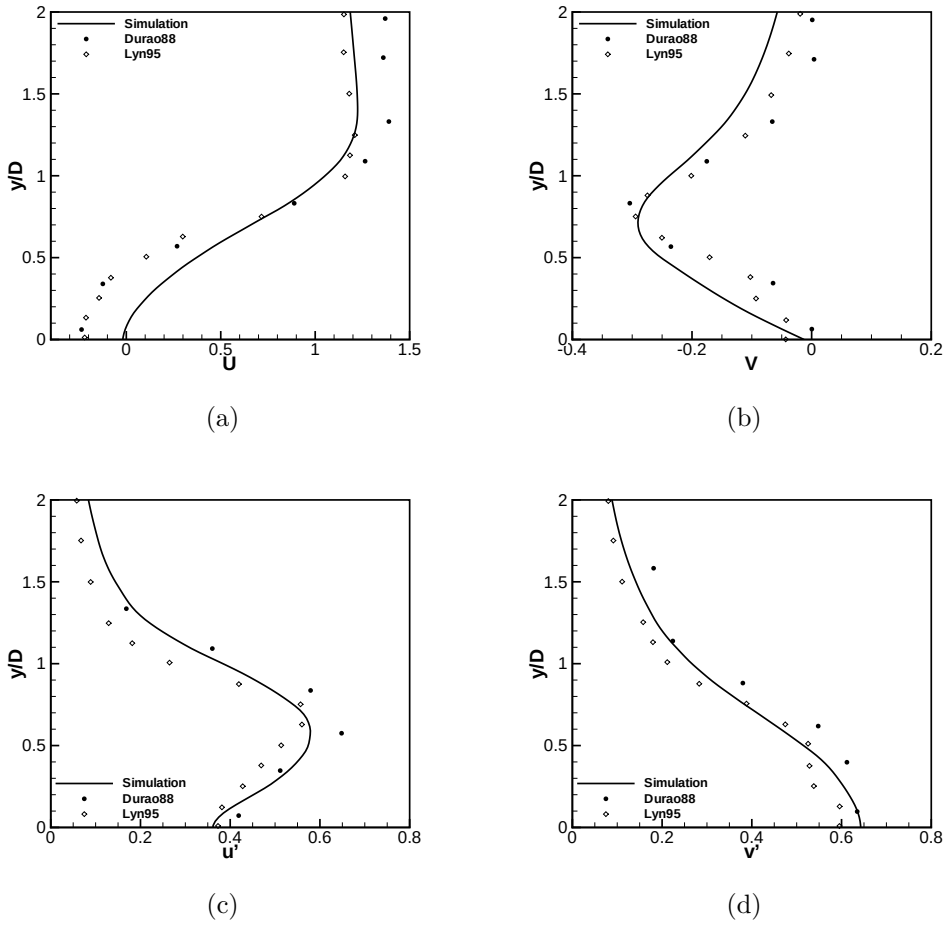
Figure 4.2.3: Velocity profiles on the flow centerline: (a) $\overline{U}$; (b)$\overline{u'}$; (c)$\overline{v'}$.

at the centerline velocity profile in Fig. 4.2.3. The cross-stream velocity profile matches well, though it slightly overpredicts the wake width. Fig. 4.2.4 also shows agreement between the predicted and experimental mean velocity fluctuations in both streamwise and cross stream directions.

There are several explanations for why the simulation overpredicts the fluctuations on the centerline. The simplest explanation is that statistics may not have been collected for long enough time. This theory is tested by running the simulation for an additional flow-through period and comparing the flow statistics before and after the additional simulation time. Figure 4.2.5 shows the centerline distribution of $\overline{v'}$ at two times separated by 45 seconds. While there is a small variation betwen the two curves, the difference is less than 5%, indicating that the flow has indeed reached a stationary state.

The second explanation may be due to insufficient grid resolution near the cylinder in the wake. This could cause an under-prediction of the viscous effects, so the fluctuations would not damp out as quickly as they should. Related to this possibility is the third possible reason the flows do not match exactly: there may be too few grid points in the spanwise direction. While the current research uses a much more resolved grid than any used in the workshop paper by Rodi et al. (1997), it is possible that a finer grid may be necessary to fully resolve the spanwise turbulent effect on the flow, even with the use of LES modeling.

Despite the discrepancies between the simulation and experimental results, this simulation provides a better prediction than many previous LES studies performed for the same case. Because it improves on the previous work in the field, this test



(a) (b)

(c) (d)

Figure 4.2.4: Mean velocity and velocity fluctuation profiles at $\frac{L}{D} = 1.0$: (a)$\overline{U}$; (b)$\overline{V}$; (c)$\overline{u'}$; (d)$\overline{v'}$.

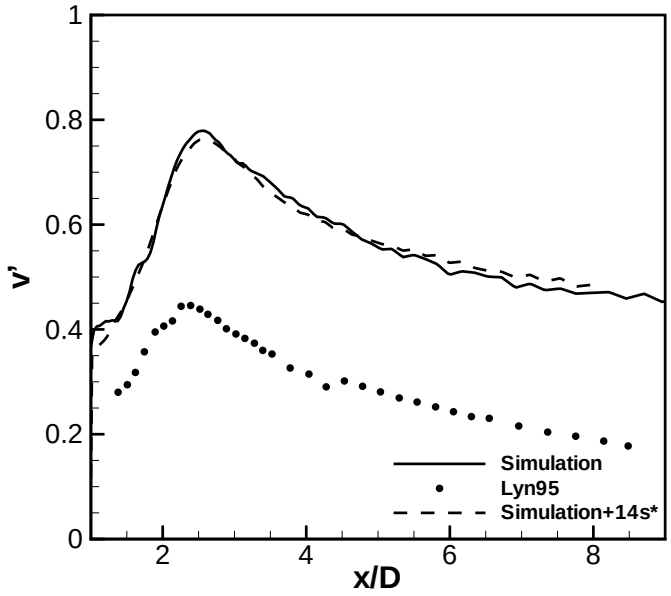


Figure 4.2.5: Cross stream velocity fluctuation profile at two different times: $t = 34.2s^*$ and $t = 48.4s^*$.

case is successful. The data collected is useful to develop and test vortex detection methods. The mean flow matches the experimental well enough to be able to apply what is learned through simulation to the physical cases, so the square cylinder will be used to test new methods of vortex detection as well as in the development of a vortex-pressure correlation on the top surface of the square cylinder.

Chapter 5 – Data analysis and reduction

Data collected from the simulation is written to non-indexed, non-ordered data files set up to be read into the CFD post-processing program Tecplot. This format is used to keep the grid intact during the export, even if that grid is unstructured. However, this file format is not easy to read through other means, so the data is interpolated onto a subgrid using Tecplot before further post-processing.

The interpolation used is inverse distance weighting to maintain the order of accuracy in the data sets. All the subgrids used for interpolation are Cartesian with a constant Δ_x and Δ_y, set such that the grid size is equal to the first cell size away from the square cylinder surface. This results in slices of data, taken along the $z = 0$ plane every second where $-1 \leq x \leq 5$ with 300 grid points and $-2 \leq y \leq 2$ with 200 grid points. At the time of interpolation, the region within the square cylinder is masked off in the data set so no calculations will be performed on points within the body.

Further post-processing takes place using several Fortran codes developed for vortex detection and calculating the pressure-vortex correlation. In the vortex detection code, it is necessary to calculate velocity gradients and the Hessian of the pressure, which is composed of second derivatives of the pressure field. While a simple central differencing scheme would work for taking these derivatives without reducing the overall order of accuracy, a fourth order central differencing scheme,

shown in general form in Equation 5.0.1, was applied to enhance the accuracy of the derivatives.

$$f'_j = \frac{f_{j-2} - 8f_{j-1} + 8f_{j+1} - f_{j+2}}{12h} \tag{5.0.1}$$

The data is filtered after the derivatives are taken using a low-pass Gaussian filter with a fixed width, the equation for which is shown in Equation 5.0.2. A filter is used to isolate the larger, and presumably more powerful, flow features. These features are of more interest, as they are more likely to affect the surface forces on the body in the flow.

$$\tilde{f}(x,y) = \int_A (f \cdot G)\, dA; G(x,y) = \frac{a}{2\pi\sigma^2}\exp(-\frac{\mathrm{x}^2 + \mathrm{y}^2}{2\sigma^2}) \tag{5.0.2}$$

In Eq. 5.0.2 σ is the filter width, and is fixed at $L/D = 0.25$ for this research.

The Gaussian filter is a high pass filter, so a low pass field is obtained by subtracting the high pass data from the raw data. Although the smaller features may not contain as much power as the large features, they may still be of interest. After filtering, the vortex detection routines are applied, as outlined in the following sections. The results will be compared between low-pass, high-pass, and unfiltered data sets.

5.1 Vortex detection

Vortex detection methods can be divided into two broad categories: integral methods and derivative-based methods. These categories may be thought of as global

(integral) or local (derivative) scale detectors. It is expected that the integral methods will detect flow features on the order of their integral area or larger, while the derivative methods are expected to detect flow features on the scale of the grid spacing or slightly larger.

Five methods are discussed below, three unique and two derived based the other methods. The unique methods are the Γ function, the λ_2 (eigenvalue) method, and the vorticity. Two new methods are introduced, the Γ_p method, in which the Γ function is modified to use the pressure gradient (tensor) field, and the λ_p method that uses the Hessian of the pressure in place of the velocity gradient as a detector of pressure minima.

5.1.1 Integral methods

The integral methods, comprising of Γ and Γ_p, are based on an area-average integral of the actual swirl in the flow. The Γ function was introduced by Graftieaux et al. (2001) as a method of flow feature detection for experimental velocity data. The equation for the Γ function is show below in Equation 5.1.1. The accompanying diagram in Figure 5.1.1 will help illustrate each term in the equation.

$$\Gamma(x,y) = \frac{1}{A_m}\int_{A_M} \frac{\left(\overline{PM} \times \overline{U_M}\right)\cdot \widehat{Z}}{\left(\|\overline{PM}\|\|\overline{U_M}\|\right)} dA = \frac{1}{A_M}\int_{A_M} sin(\theta_M) dA \qquad (5.1.1)$$

In Eq. 5.1.1, P is the point around which Γ is being computed, M are the points in the surrounding area denoted by A_M. PM is the vector from point P to each

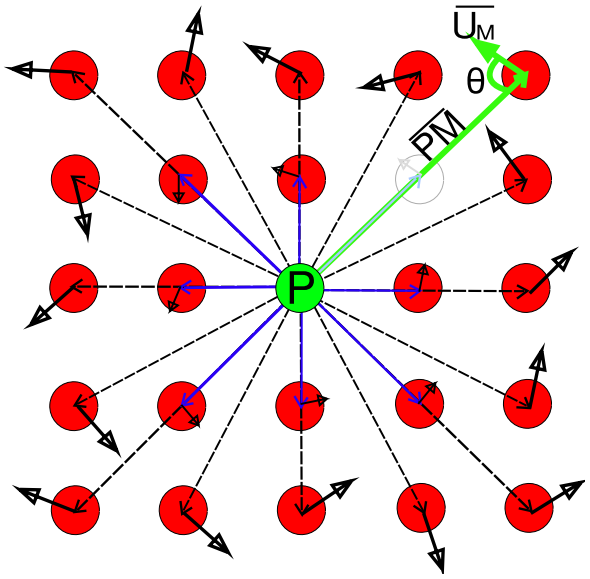


Figure 5.1.1: Illustration showing Γ function.

point M, and U_m is the velocity at point M. The Γ function can also be described as the area-averaged sine of the angles between PM and U_M over the area A_m.

The Γ function ranges from $-1 \leq \Gamma \leq 1$ with magnitudes close to 1 indicating a strong, nearly circular path, and values close to zero representing weak or non-existant swirl in the flow. The sign of Γ is important, too. Negative values of Γ indicate a clockwise swirl, while positive values indicate a counter-clockwise swirl. This indication of strength and direction in a single detection scheme is one of the best features of the Γ function.

In a region of swirl, vectors of the gradient of pressure will point radially away from the vortex core, as shown for a Taylor vortex in Figure 5.1.2. With this in mind, a new vortex detection method is derived from the Γ function using the rotated pressure gradient vector field. With this formulation, regions of swirl will

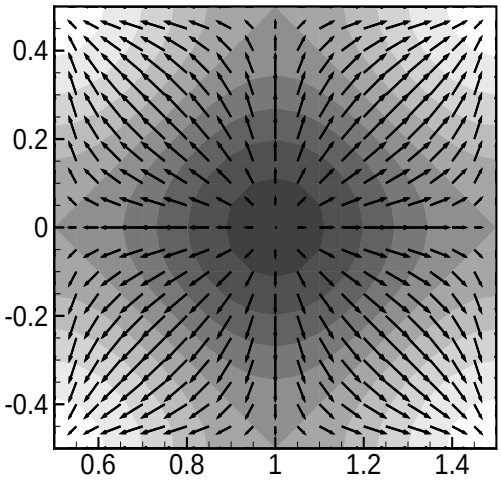


Figure 5.1.2: Vectors of pressure gradient over pressure contours for Taylor vortex.

be found where the non-rotated pressure field has a strong attraction or repulsion from a specific point in the flow. Equation 5.1.2 shows the calculation for Γ_p, where $\overline{P'_m} = -(\nabla p)^{\perp}$, which is the pressure gradient tensor rotated 90° counter-clockwise.

$$\Gamma_p(x,y) = \frac{1}{A_M}\int_{A_M} \frac{\left(\overline{PM} \times \overline{P'_M}\right)\cdot \widehat{Z}}{\left(\|\overline{PM}\|\|\overline{P'_M}\|\right)} dA \tag{5.1.2}$$

The integral methods are area dependent, so some care must be used when selecting a search area. An area too small and they will detect only the cores of strong vortices, while an area that is too large will tend to smear individual features together into indistinct regions. In this research a search area of $L/D = 0.0625$ is used, which is found to be a good balance between the two extremes and corresponds to a radius of five grid points. This search area is smaller than

the filter width, so there may be some features that would be detected using this search area in the raw data that will not show up in the low pass data.

5.1.2 Derivative-based methods

Unlike the area-dependent integral methods which average the swirl over the search area, the derivative-based methods should be most sensitive to changes in the flow on the order of the grid spacing and slightly larger. The first derivative-based approach that is commonly used is the vorticity. However, for vortex detection near the surface of bluff bodies, the vorticity can lead to false detection because regions of high vorticity do not necessarily indicate regions of swirl. For example, in Figure 5.1.3, near the walls of laminar channel flow, the vorticity is high because of the large cross-stream velocity gradient, but there is no swirl at that location.

Vorticity contours and isosurfaces are useful as a vortex visualization techniques for qualitative vortex identification, but inside the separated region, other methods may be better at detecting vortices. Another common approach to vortex detection is to locate local pressure minima accompanied by swirl in the flow.

Information about pressure extrema is contained in the Hessian of the pressure, $p_{,ij}$, which is composed of second derivatives of pressure. However, in experimental data it is not possible to gather information about the pressure at all points in the flow. To apply the eigenvalue method, an approximation for the Hessian of the pressure is derived by first taking the gradient of the Navier-Stokes equations, as shown in Equation 5.1.3. The velocity terms are then divided into symmetric and

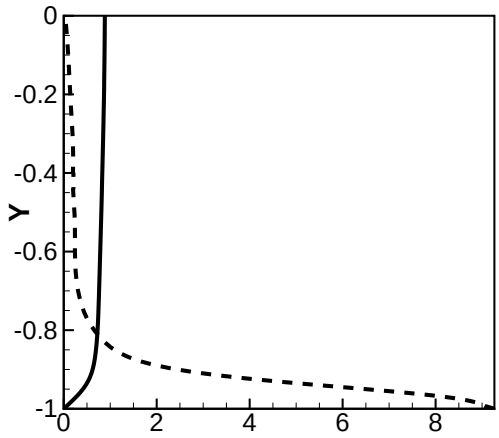


Figure 5.1.3: Velocity profile (solid line) and near-wall vorticity (dashed line) for turbulent flow through a channel.

anti-symmetric parts of the equation for further simplification.

$$\left[\frac{DS_{ij}}{Dt} + S_{ik}S_{kj} + \Omega_{ik}\Omega_{kj}\right] + \left[\frac{D\Omega_{ij}}{Dt} + \Omega_{ik}S_{kj} + S_{ik}\Omega_{kj}\right] = -\frac{1}{\rho}p_{,ij} + 2\left(\frac{1}{2}\nu u_{i,jkk}\right) \tag{5.1.3}$$

The first bracketed group of terms on the right hand side of Eq. 5.1.3 represent the symmetric part of the equation; the second bracketed group is the anti-symmetric part. The anti-symmetric terms are grouped with one half of the viscous term to form the vorticity transport equation, which is identically zero for incompressible flow. After rearrangement, the equation for the Hessian of pressure becomes

$$\frac{DS_{ij}}{Dt} + S^2 + \Omega^2 - \frac{1}{2}\nu u_{i,jkk} = -\frac{1}{\rho}p_{,ij}, \tag{5.1.4}$$

where $S^2 = S_{ik}S_{kj}$ and $\Omega^2 = \Omega_{ik}\Omega_{jk}$.

To work with the velocity gradient tensor field, two other terms are eliminated from Eq. 5.1.3: the unsteady strain, represented by $\frac{DS_{ij}}{Dt}$, and the viscous term $\frac{1}{2}\nu u_{i,jkk}$. These two terms can cause the location of a pressure minima and the location of the swirl to differ. The unsteady strain can create pressure minima without swirl or rotation; this is why pressure minima may occur outside the presence of a vortex. Viscous effects act to eliminate pressure minima where swirls do exist. By eliminating these terms, a more accurate method for detecting vortices is obtained: the eigenvalues of $S^2 + \Omega^2$ [Jeong and Hussain, 1995].

Local pressure minima are identified where two of the eigenvalues of the Hessian of pressure are positive. Since the right hand side of Eq. 5.1.4 is negative, this means that two negative eigenvalues of $S^2+\Omega^2$ will identify a local pressure minima using velocity data. That is, if the eigenvalues are ordered $\lambda_1 \leq \lambda_2 \leq \lambda_3$, then $\lambda_2 \leq 0$ is indicative of a local pressure minima.

For planar data, this criterion is identical to the critical point criterion developed by Chong and Perry (1990). The critical point analysis examines the eigenvalues of the rate of deformation tensor in three dimensions. A vortex is defined at points where the eigenvalues are complex with vortex direction defined by the eigenvector and vortex strength defined by the magnitude of the complex term in the eigenvalue. In two dimensions, vortices are detected where the eigenvalues of the rate of deformation tensor are complex conjugates, where the magnitude of the complex term indicates the swirl strength. Although the critical point analysis and eigenvalue approaches give the same results for these two-dimensional cases, they are generally different methods. The eigenvalue approach from Jeong and

Hussain (1995) is used herein, so the results are denoted with λ_2.

In simulation data sets, the pressure value is directly accessible at each point in the grid, so the Hessian of pressure can be computed using these values rather than being approximated by velocity gradients. The eigenvalues of the pressure Hessian are used as a pressure minima detector, called λ_p. This λ_p is used to evaluate the effect of eliminating the temporal and viscous terms in Eq. 5.1.4 when deriving the equation for λ_2, and is compared directly to λ_2 as a vortex detection method.

5.2 Pressure-Γ correlation

To assess the influence of passing vortices to the surface forces on a bluff body, a correlation function is derived that links the magnitude of the passing swirl and the magnitude of the pressure on the surface. By examining how this correlation evolves in time, it should be possible to derive a relationship between a flow event passing at a certain distance to the surface forces. The eventual goal of such a correlation is to be able to predict the surface effect if the location of such a vortex is known. Conversely, if a specific surface force pattern is recognized, one would be able to deduce the structure of the surrounding flow field.

A correlation between surface pressure and the Γ function is a function of time and the distance from the surface. Equation 5.2.1 shows the general form of the correlation. This is a normalized cross-covariance function. When values are negative, it indicates that one variable is less than its expected value while the other is greater than its expected value. This is called anti-covariance. When the

values are positive, both variables are either above or below their expected values, indicating covariance.

$$\left\langle C_{S_j}(\lambda,t)\right\rangle = \frac{\left\langle \mathrm{Cp_j^{'}}\cdot\Gamma^{\mathrm{c}'}_{\mathrm{j},\lambda}\right\rangle}{\left\langle\sqrt{\overline{\mathrm{Cp_j^{'2}}}}\right\rangle\cdot\left\langle\sqrt{\overline{\Gamma^{c'2}_{j,\lambda}}}\right\rangle} \tag{5.2.1}$$

It is not immediately clear what each term in Eq. 5.2.1 signifies, so some explanation is needed. The quantities in angle brackets $\langle\rangle$ indicates segment averaging. Values denoted with a prime are fluctuations from the mean, e.g. $\mathrm{Cp}'_\mathrm{j} = (\mathrm{Cp_j} - \overline{\mathrm{Cp_j}})$. In the equation Γ^c is the measurement of the swirl with a cutoff applied to isolate only the strongest swirls, $\mathrm{Cp} = \frac{\mathrm{P_s}-\mathrm{P}_\infty}{\rho \mathrm{u}_\infty}$, and λ indicates the bin, or distance, from the top surface. The index j is the segment of the surface over which the correlation is computed. $C_{S_j}(\lambda,t)$ indicates that the correlation is for each segment S_j, and is a function of the distance from the surface and time. The denominator of Eq. 5.2.1 is the product of the root means squared values of the pressure coefficient and Γ^c fluctuations.

In the covariance calculation, both terms are nondimensional and are $O(1)$. The two-point covariance is normalized by the RMS values of the two variables, so is normalized to have maximum magnitude of one. Covariance magnitudes close to one indicate the strongest link between the two variables, and covariance magnitudes of 0.1 or higher indicate some relation between the two variables.

The distance from the surface is measured radially from each point on the surface within a segment, then is put into bins, denoted by λ. Each index λ

corresponds to a range of radii away from a given surface point. The algorithm for computing this correlation is listed below for clarity. Figure 5.2.1 shows a sample bin distribution and sample segmentation of the surface of a cylinder.

1. Read vortex detection data.
2. Compute and store the bin value for each point in the data field at each point on the surface
3. Loop over the number of surface segments S_j
   (a) Loop over the number of surface points in each segment N_i
   (b) Loop over each point in the field above the surface and multiply the surface pressure coefficient fluctuation $\mathrm{Cp}'_{\mathrm{i}}$ by the value of $\Gamma^{c'}_{\lambda}$ for each point
   (c) Accumulate the value of $(\mathrm{Cp}'_{\mathrm{i}} \cdot \Gamma^{c'}_{\lambda})$ as functions of S_j, λ, and time
4. Use the accumulated value to calculate $\left\langle C_{S_j}(\lambda, t) \right\rangle$ and store the value of the correlation at each segment-bin point

Thus, for each time step there are curves showing the correlation value as a function of the distance from the surface for each surface segment. This data is analyzed in time to determine if there is a temporal or periodic relation between the passage of a vortex at a specific distance λ and the surface forces on a specific segment. It is also possible that the correlation will show distinct distances from the surface where the passage of a vortex has a very strong influence on the surface

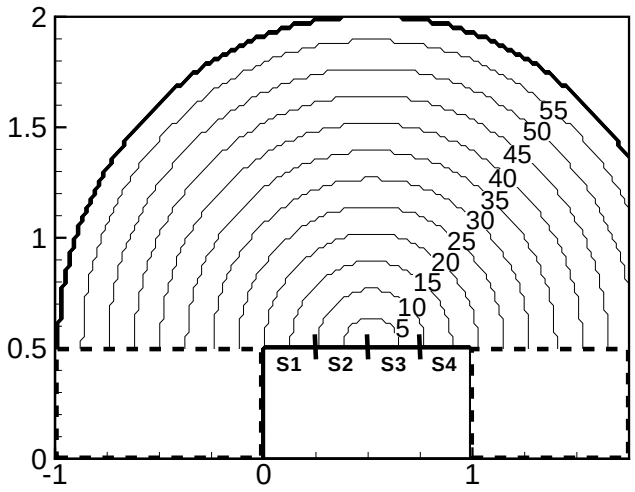


Figure 5.2.1: Indices of (λ) and segments for correlation calculation.

forces over all time steps, or that there are distances at which the passage of a vortex has little or no effect on the surface forces.

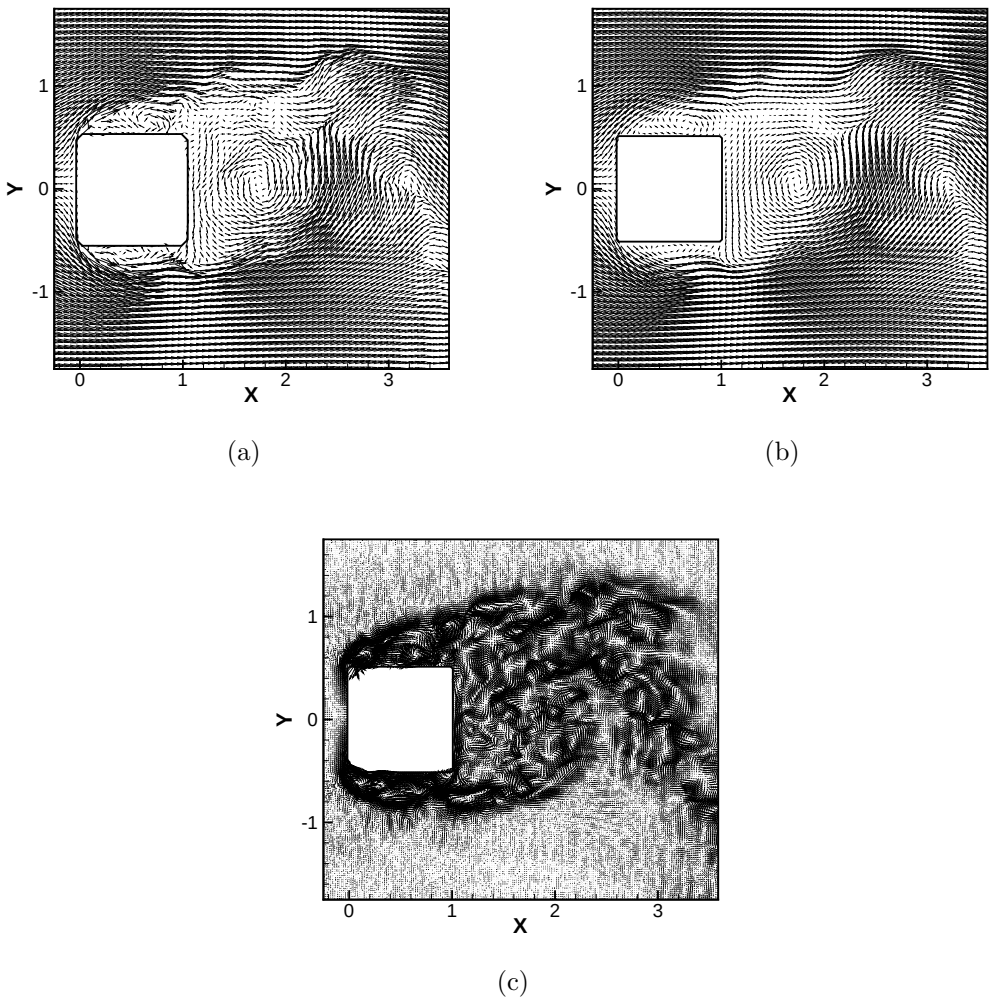
Chapter 6 – Results and discussion

The results are presented in the same order in which they are discussed above. First, the general vortex detection results, then the specific results for the integral and derivative-based results. Finally, the correlation is calculated based on the results of the vortex detection.

6.1 Vortex detection

The flow features can be broadly classified into local and global scales, and are preferentially detected based on which type of filter is used on the flow field. Figure 6.1.1 shows the flow field decomposed into large and small scales, compared to the raw field for a single instant in time. Velocity vectors are shown to highlight the flow features.

There are several discernable differences between the filtered and unfiltered data. In the raw data field, there are several scales of swirl visible, from the small swirls near the leading edge through the large recirculation in the near wake. In the low-pass filtered data (center), large swirls are visible near the trailing edge of the square cylinder, outside of the separation region, as well as in the wake. However, in the high-pass filtered data, the vortices shed from the leading edge are most visible, and the large scale wake recirculation does not show up at all.



(a) (b)

(c)

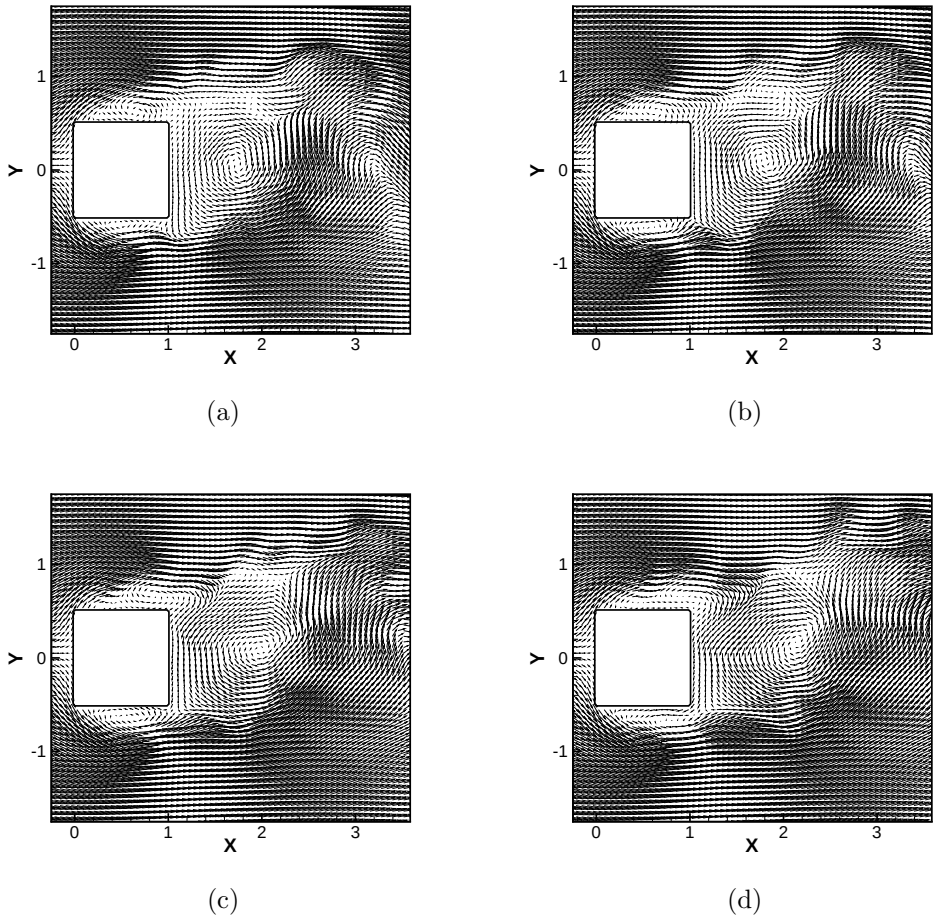
Figure 6.1.1: Velocity field filter comparison: (a) raw data; (b) low pass; (c) high pass.

Also of interest is how the flow field evolves over time, because the temporal evolution of vortices in the flow will affect the surface forces. The low pass data is compared for several successive time slices in Figure 6.1.2. It is possible to track large scale features as they are convected through the domain by comparing successive instants in time. This may help predict where in the flow vortex passing may have the largest effect on the surface by showing where vortices reside for the longest times.

In Fig. 6.1.2, the large scale recirculation in the near wake is visible for each time plotted. It is convected downstream between instants in time. Other features are visible in the separated region and on the edges of the wake that are also convected downstream between the snapshots shown. The structures detected are large enough that tend not to disappear between the frames shown. The ability to track swirls at different instances in time is one reason the low pass filtered flow field is used for vortex detection.

6.1.1 Integral methods

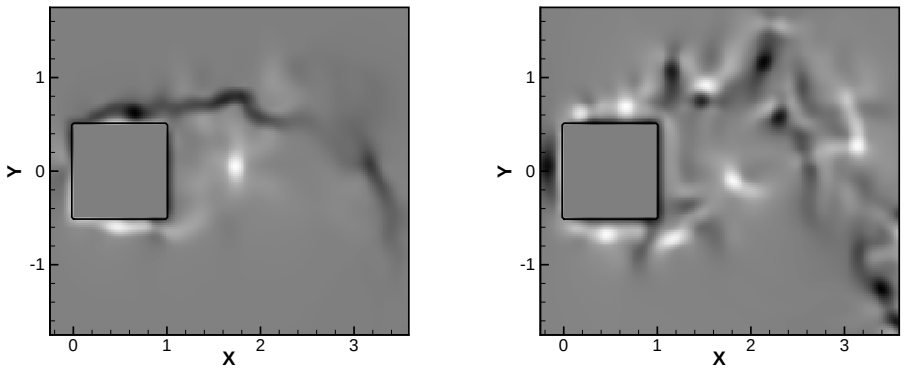
The integral methods are area dependent methods of locating flow swirl. The Γ function represents the area averaged circulation around each point in the flow and Γ_p locates regions where the pressure gradients are strong based on where the rotated pressure gradient has the greatest swirl. There are distinct differences between Γ and Γ_p, both in the scale of features detected and the quantity of features detected. The Γ function locates the large scale features in the flow well,



(a) (b)

(c) (d)

Figure 6.1.2: Velocity vectors time evolution (a) $t = 157.6s^*$; (b) $t = 157.9s^*$; (c) $t = 158.3s^*$; (d) $t = 158.6s^*$.



(a) (b)

Figure 6.1.3: Contours of (a) Γ; (b) Γ_p at $t = 157.6s^*$.

as is shown in Figure 6.1.3. Features detected include the vortices shed at the leading edge and the large scale swirl in the near wake of the cylinder.

Comparing the two integral methods shows that Γ_p detects more features than the original Γ function. The features detected with Γ_p are not as large as those detected by Γ, although both methods detect similar features in each frame. The Γ_p function is more sensitive to smaller features because it is based on derivatives of a flow variable, which makes it sensitive to changes on the order of the grid spacing. From a physical point of view, it makes sense that the Γ_p detector is more sensitive than the Γ function, as the pressure gradient is proportional to the velocity squared, as shown in the Navier-Stokes equations.

The Γ function tends to smear individual features together into regions of swirl, as shown in the separated shear region above the surface of the cylinder. The Γ_p function tends to identify the individual vortices shed from the leading edge,

as well as the counter-rotating vortices that roll up inside the separated region. One main difference between the two techniques is the thickness of the separated region identified by the methods. The Γ function shows a smaller separation region because of the smearing of the swirls into one region. Both identify the large region of swirl in the near wake of the cylinder, although the Γ_p function shows individual features surrounding the large recirculation region as well.

To help identify different scales in the flow, the Γ function is calculated in high pass and raw data sets as well as the low pass data sets. The low pass data is likely to contain the largest and strongest vortices, but the high pass data may show more features in the flow of smaller scales. These smaller scales should give more indication of the overall shape of the separated region as well as the wake region. Figure 6.1.4 shows the Γ function computed on the raw data, low pass, and high pass data sets for a single time.

As expected, the low pass filter extracts the largest features from the raw field while the high pass filtered data contains the smallest features in the flow field. The shape of the wake and the edges of the separated region are clearly visible in the high pass data set, but because the features are so small that they will tend to disappear between frames, so it is not as useful to track features from instant to instant. Similar plots of the Γ_p function are shown in Figure 6.1.5, with similar results. The main difference between the results of Γ and Γ_p is that Γ_p shows more features at each filtering level.

Both methods are successful at detecting flow features in a single frame, but it is important to determine their validity as methods to detect and track vortices

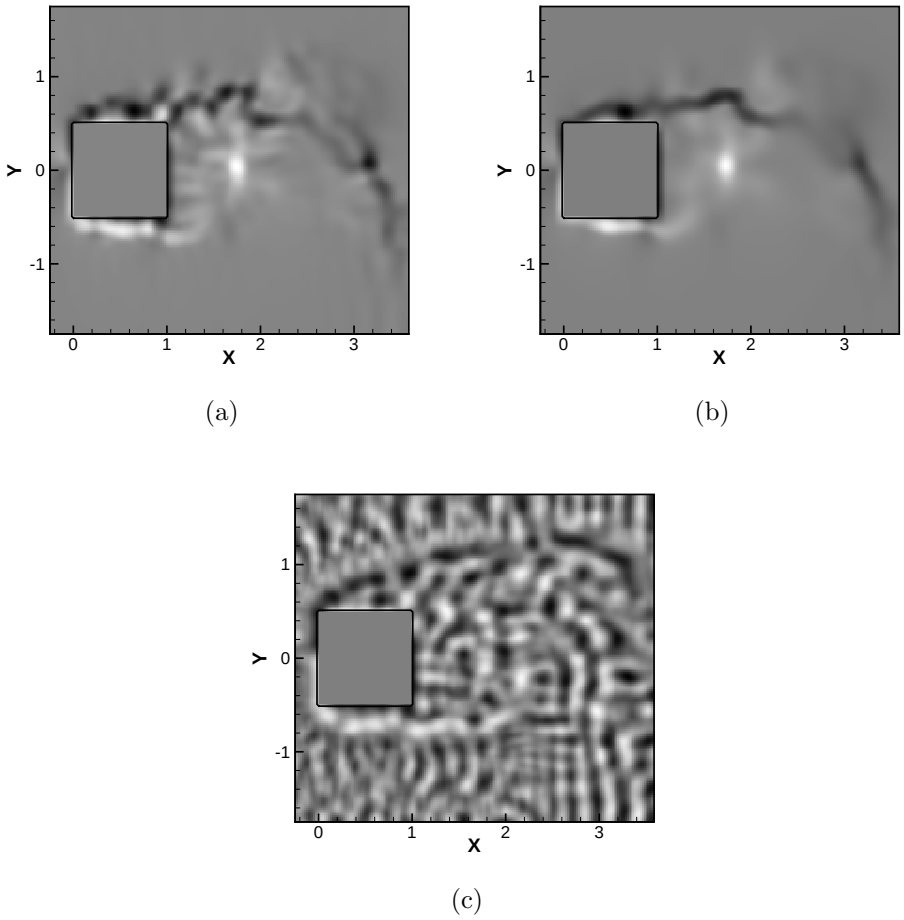


Figure 6.1.4: Γ function of (a) raw; (b) low pass; (c) high pass velocity fields.

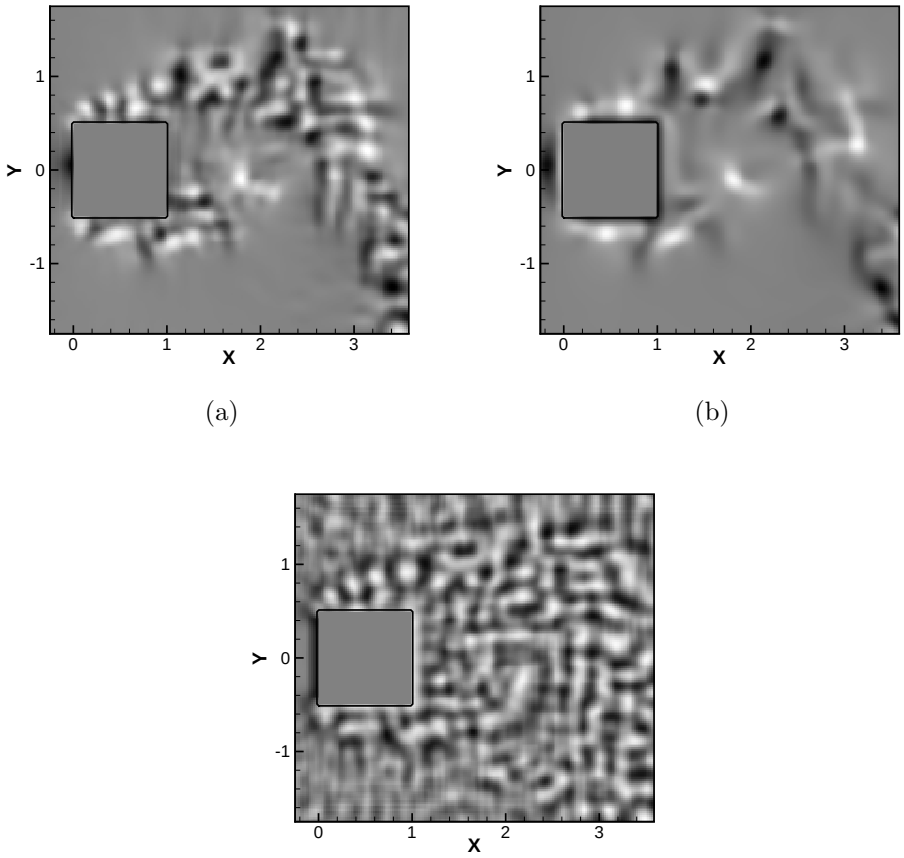


Figure 6.1.5: Γ_p function of (a) raw; (b) low pass; (c) high pass pressure gradient fields.

over time. Looking only at the low pass data to isolate the stronger swirls, Figure 6.1.6 shows the temporal evolution of contours of Γ and Γ_p. The Γ function shows a smooth evolution with consistent convection of the flow features. Because there are more features, and smaller features, in the Γ_p plots, they tend to disappear between frames.

Over time, the shear layer stretches into the wake, shown in the plots of Γ over time. The large area of recirculation in the near wake is convected downstream as it is dissipated, becoming weaker. In the Γ_p plots, the same behavior is visible although harder to track because of the number of features detected. The vortices shed from the leading edge are convected along the edge of the shear layer above the top surface. These features are easily traceable in the Γ_p contour plots over time.

By placing probes in the domain at several point in the field, it is possible to get a time history of Γ and Γ_p to compare the vortex passage frequency with the shedding frequency observed in the pressure signal on the top edge. Figure 6.1.7 shows the three probe locations used for the time history of Γ and Γ_p, one near the leading edge, and two points in the free shear layer. Figure 6.1.8 shows the time history of both Γ and Γ_p at these points.

Time history plots of the Γ and Γ_p functions show periodic content with a wides range of time scales. The time history is converted to the frequency domain to examine the frequency content of the signal at each point, shown in Figure 6.1.9. This is done to verify that the frequency of the detected vortices matches the shedding frequency of the pressure and velocity probes previously discussed.

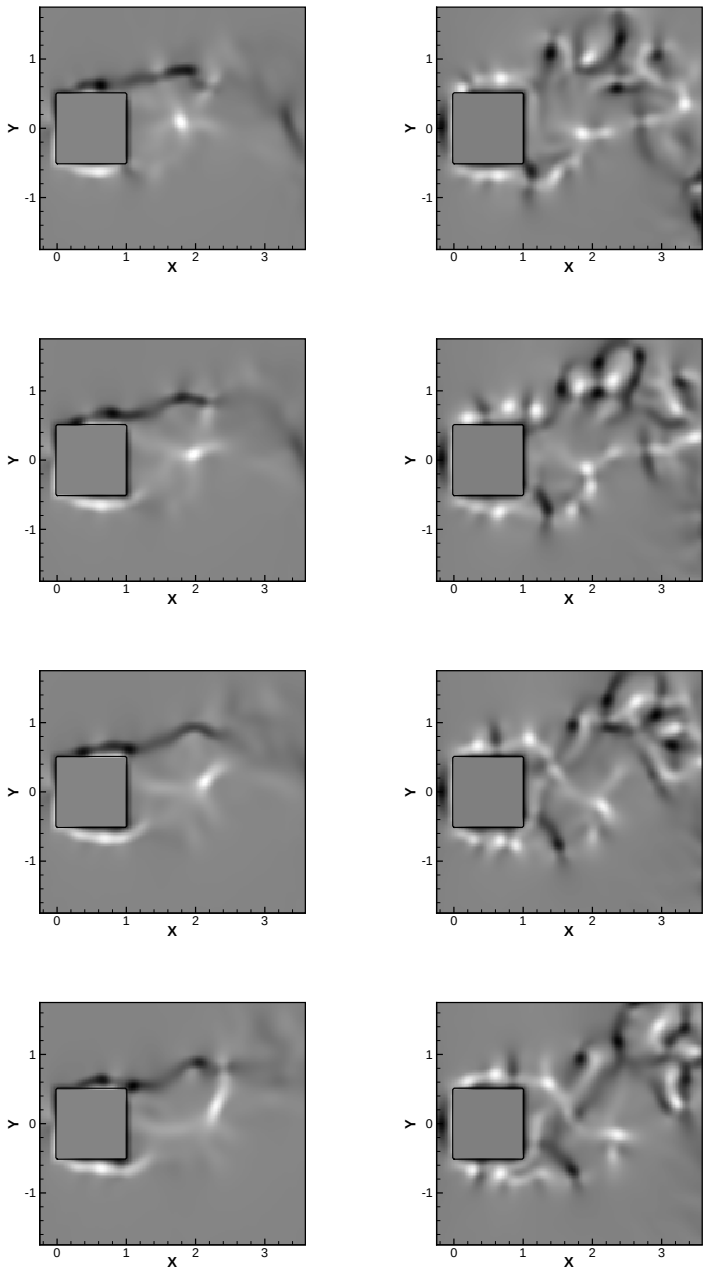


Figure 6.1.6: Time series evolution of Γ (left) and Γ_p for $t = 157.9s^*$, $t = 158.3s^*$, $t = 158.6s^*$, and $t = 160.0s^*$.

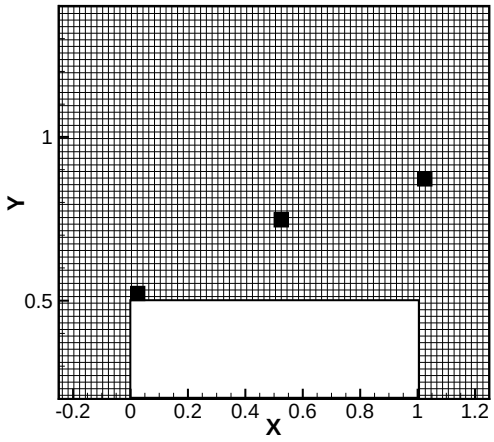


Figure 6.1.7: Probe points in the flow field, placed in or near the separated region.

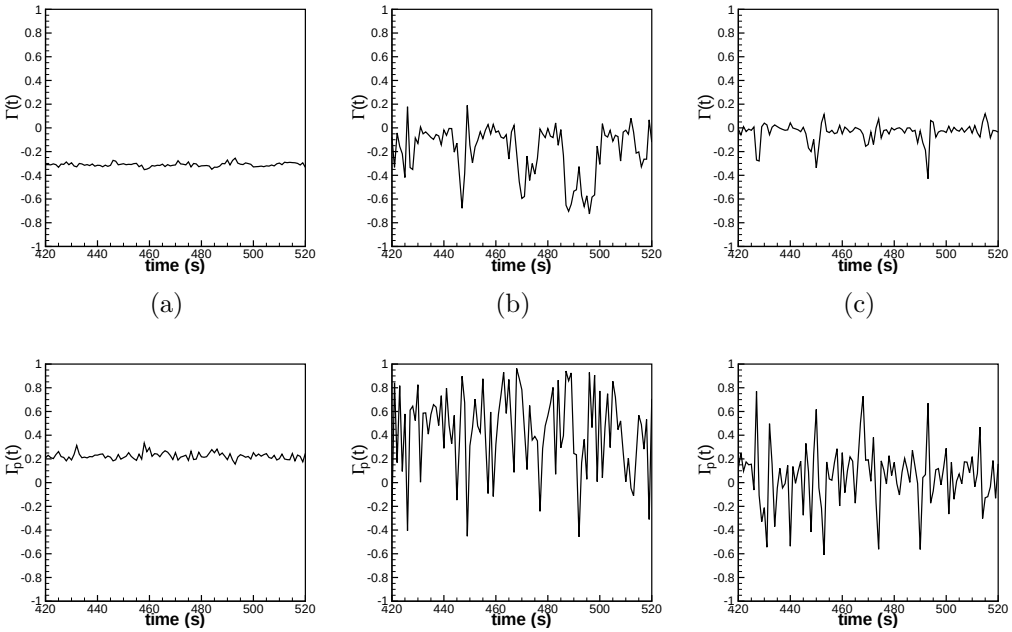


Figure 6.1.8: Time history plot for (a-c) Γ; (d-f) Γ_p. Left is leading edge, next are boundary layer points.

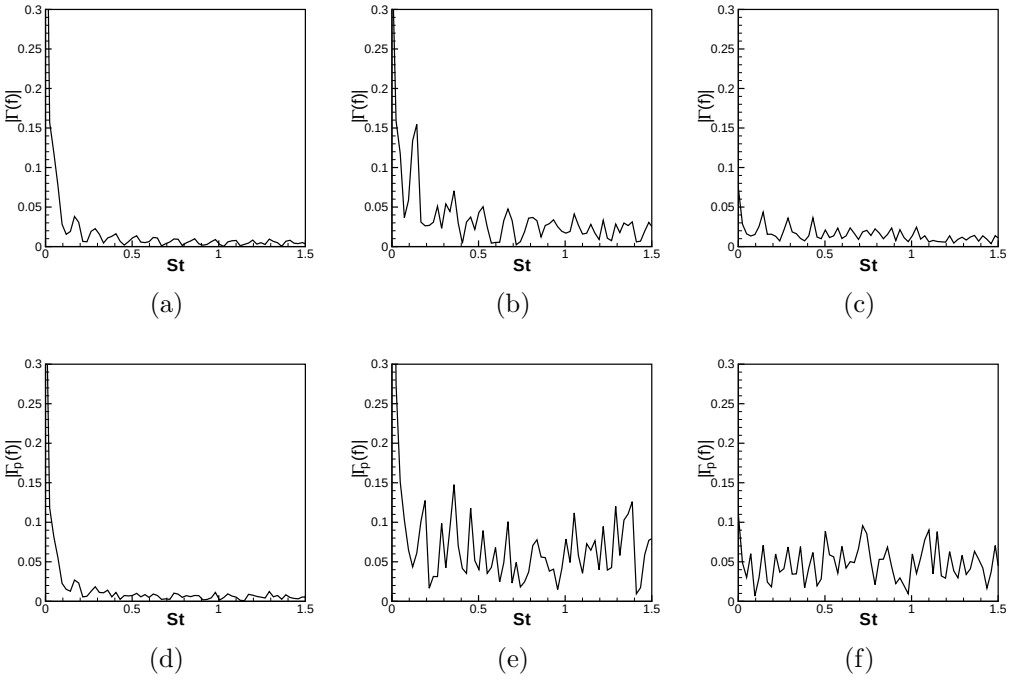


Figure 6.1.9: Frequency plot for (a-c) Γ; (d-f) Γ_p. Left is leading edge, next are boundary layer points.

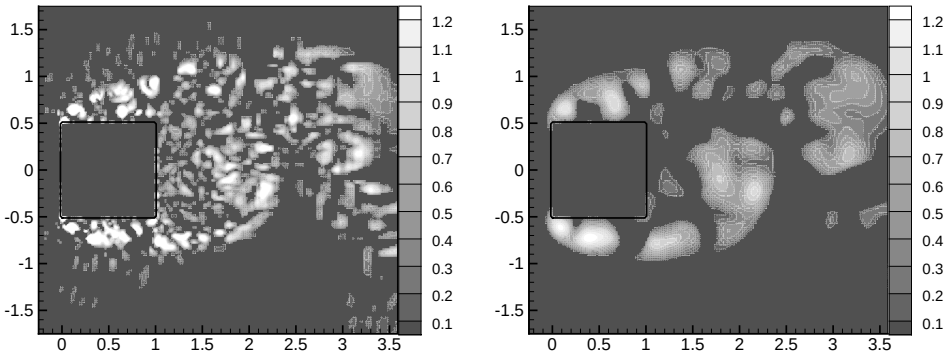
The frequency of detected vortices matches well with the probes of velocity and pressure. There is more power in higher frequencies in the signal of Γ_p than than the Γ function because Γ_p detects more features at each time step. The frequency of the probed signal nearest the wake shows the broadest spectrum in both Γ and Γ_p, while the leading edge probes show a dominant shedding frequency of 0.05Hz, which coresponds with a Strouhal number of 0.13.

6.1.2 Derivative-based methods

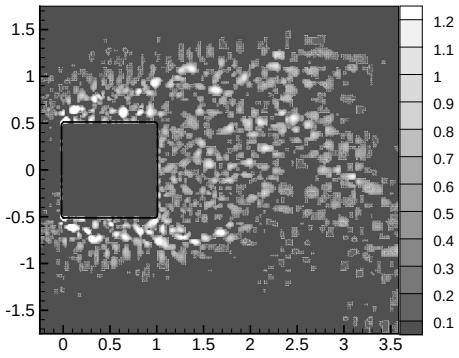
The derivative-based methods detect smaller flow structures than the integral methods because they look at differences in flow variables on the order of the grid spacing. Both eigenvalue methods detect vortices by locating pressure minima within the flow. The eigenvalue λ_2 is calculated using the gradients of the velocity and λ_p is calculated using the Hessian of the pressure. To investigate the structures detected for different filters using the eigenvalue methods, comparison of the results for three filters for λ_2 is shown in Figure 6.1.10.

The low pass filtered data isolates a few larger flow features. The high pass shows noise from the derivatives along with the detected features, so is less useful for identifying individual features. Figure 6.1.11 compares the filtered data for λ_p. The results are similar to the λ_2 results, with the larger vortices isolated using the low pass filter. Both methods are compared side by side in Figure 6.1.12 to identify similarities and differences in the results.

Both the λ_2 and λ_p contours show very similar results in the location and size of

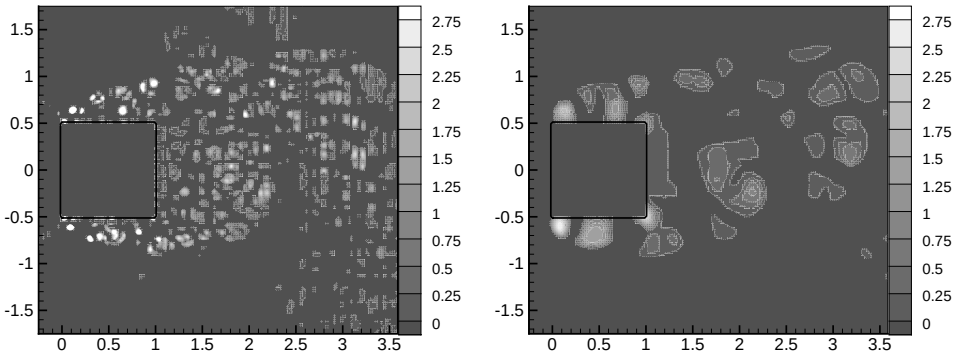


(a) (b)

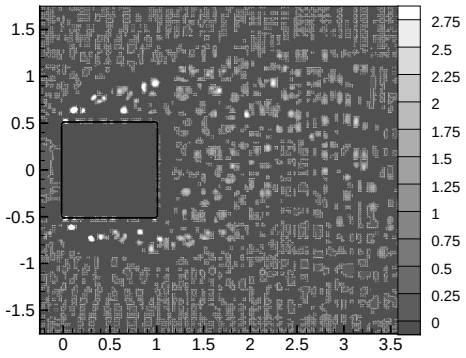


(c)

Figure 6.1.10: λ_2 of (a) raw; (b) low pass; (c) high pass velocity fields at $t = 157.6s^*$.

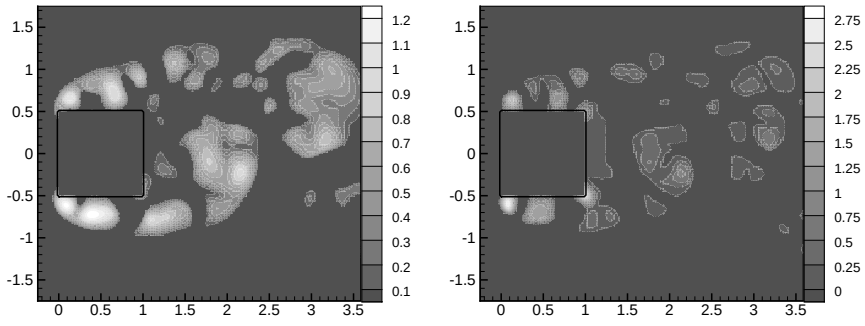


(a) (b)



(c)

Figure 6.1.11: λ_p of (a) raw; (b) low pass; (c) high pass velocity fields at $t = 157.6s^*$.



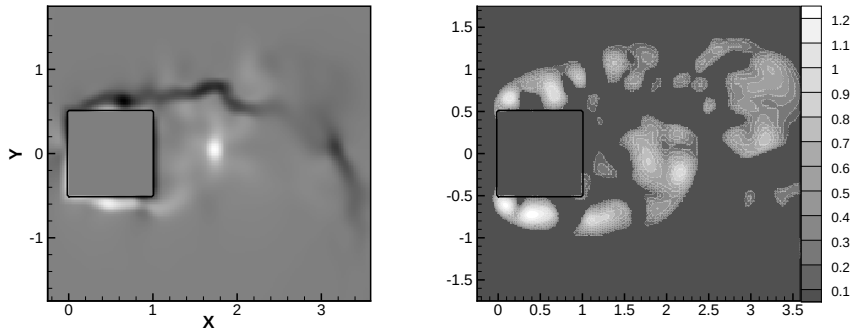
(a) (b)

Figure 6.1.12: Low pass filtered (a) λ_2 and (b) λ_p comparison at $t = 157.6s^*$.

detected flow features. In the contours of λ_p, the regions surrounding the core are somewhat smaller. However, the magnitudes of λ_p are slightly higher than those of λ_2. The differences between the two methods are due to the terms neglected when calculating λ_2 based on $S^2 + \Omega^2$.

The eigenvalue methods are based on the derivatives of flow variables, so they are expected to detect flow features defined by changes in flow variables on the order of the grid size. For large areas of recirculation, such as the vortex in the cylinder near wake, the vortex may be several orders of magnitude larger than the grid spacing, so the derivative-based approaches may miss the core itself and only detect the strong gradients near the edges of the vortex. This is shown by comparing λ_2 and Γ in Figure 6.1.13. Both methods identify the vortices shed by the leading edge, but the Γ function shows the core of the near-wake swirl while λ_2 shows the edges of the same swirl.

Another difference between the Γ function and eigenvalue approach is the lo-



(a) (b)

Figure 6.1.13: Vortex detection comparison of (a) Γ and (b) λ_2.

cation of the center of the feature detected. This difference is noticeable not only in the wake, as previously mentioned, but near the separation region. The locations detected by the Γ function correspond to the regions of highest swirl, which surround the vortex cores. The locations shown in the λ_2 contours are the points with the strongest difference in derivatives from point to point, which will be at the center of a vortex.

While the eigenvalue methods does not detect the largest flow features, it is still a very useful method for detecting small scale flow features, especially those shed from the leading edge of the cylinder. To assess how well these detected features can be tracked over time, Figure 6.1.14 shows several consecutive frames of the eigenvalue contours.

Some of the features detected by the eigenvalue methods are traceable from frame to frame. Once features have been shed from the leading edge and convected about halfway along the top edge, they can be tracked as they are convected

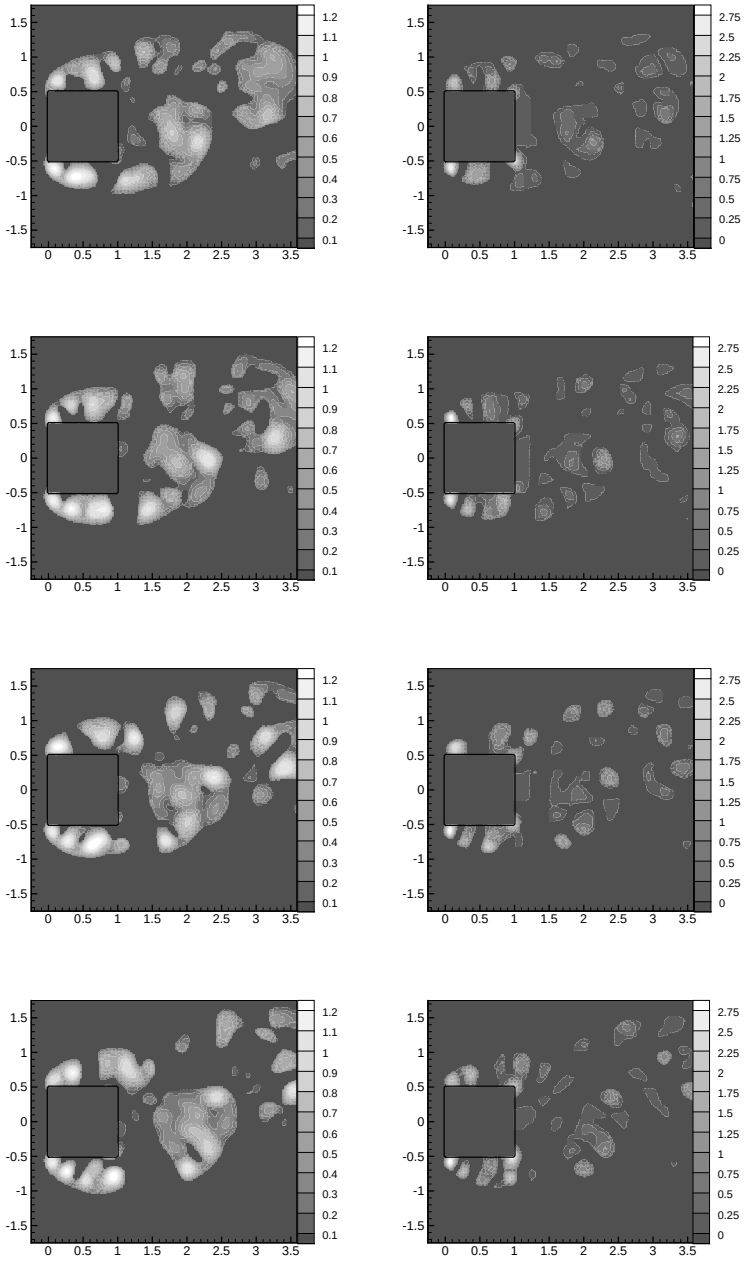


Figure 6.1.14: Time series evolution of λ_2 (left) and λ_p for $t = 157.6s^*$, $t = 157.9s^*$, $t = 158.3s^*$, and $t = 158.6s^*$.

downstream and into the wake. At the leading edge, there is a constant region of swirl of varying intensity, and it is difficult to pinpoint exactly when a vortex is shed. In the wake, the swirls are large and fairly slow moving; they show up in multiple successive frames.

The time series contours of λ_p show similar results to the contours of λ_2. Some of the flow features appear in multiple time slices, especially the larger features in the wake. Many of these features are the same as those shown in the contours of Γ and Γ_p over time in Fig. 6.1.6. As with the single instant shown in Fig. 6.1.13, features near the trailing edge but not yet in the wake are detected well by both integral and derivative-based methods. The large scale recirculation in the wake is detected by the integral methods, while the small scale features at the leading edge are detected well by the derivative methods.

The Γ function is used for developing a correlation between the passage of vortices in the flow and the surface pressure. The main advantage that Γ has for this purpose over the other vortex detection methods is that it is already normalized and provides information about the direction of the swirl as well as its size. It is also useful because the features detected using Γ are large enough to be tracked between instants in time, which will be useful when deriving a correlation as a function of time.

6.2 Pressure-Γ correlation

The pressure-Γ correlation is a function defined at each segment of the top surface of the square cylinder as a function of the distance away from the segment and time. However, the transient effects are difficult to separate from the distance effects if both are considered simultaneously. To better understand the results the time and distance variables are presented as separate independent variables, with the value of the correlation as the dependent variable.

It is instructive to look first at the time series of the pressure signal $Cp' = (Cp - \overline{Cp})$, where $\overline{Cp}$ is the time averaged coefficient of pressure for each segment. These plots will show which segments have the most variation in pressure over time. The segments with the most variation are those most influenced by the passage of vortical structures and other flow events. Figure 6.2.1 shows the time series plot for each segment. Segment one is the segment nearest the leading edge; segment four is the segment nearest the trailing edge.

There is a low frequency oscillation in the time signal for each segment caused by the oscillation of the cylinder wake. No single segment has significantly larger magnitude fluctuations than the other segments. There are higher frequency fluctuations in each segment caused by the shedding and passage of vortices from the leading edge. This higher frequency signal increases in magnitude from the leading edge segment back to the trailing edge. A spectral analysis is performed to determine the dominant frequency of the pressure signals, shown in Figure 6.2.2.

The dominant frequency in each segment is approximately 0.05 Hz. Since this

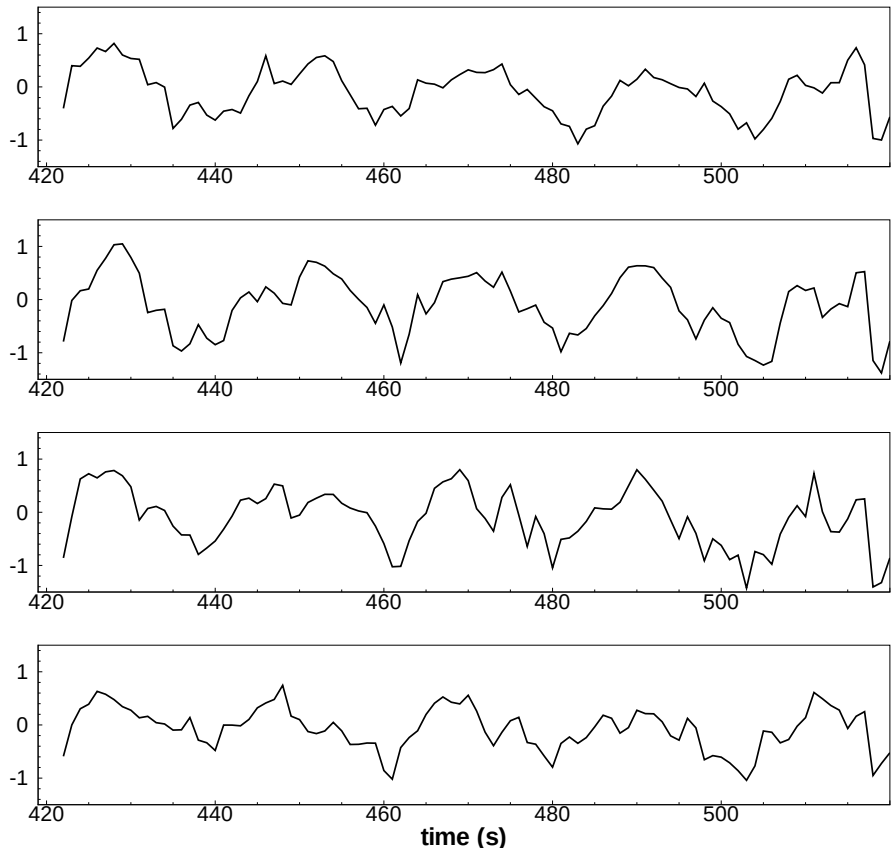


Figure 6.2.1: Time series of Cp' for each segment, ordered 1-4 from top to bottom.

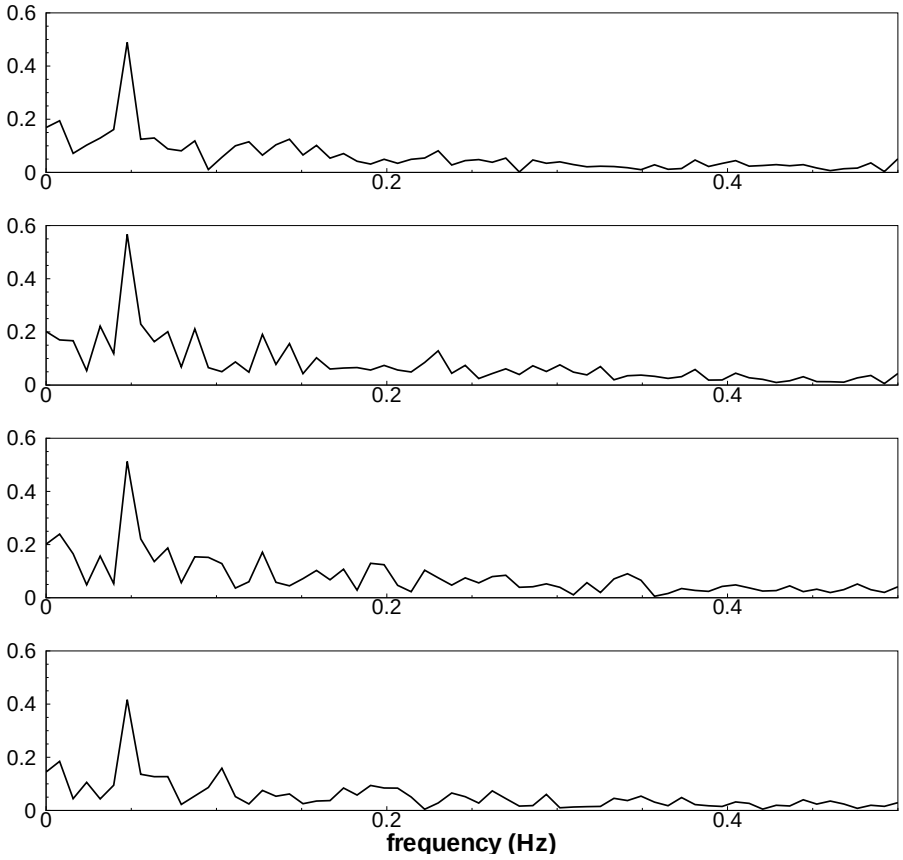


Figure 6.2.2: Spectral analysis of Cp′ for each segment, ordered 1-4 from top to bottom.

frequency analysis is based on the post-processed data with a sampling frequency of 1Hz and 100 samples, the resolution of the plot is 0.01Hz, so the dominant frequency has an error of ± 0.005, which results in a Strouhal number of $St = 0.15 \pm 0.0015$. This range matches the Strouhal number computed for the pressure signal at the midpoint of the square cylinder. The spectral analysis shows that there is some higher frequency content in each segment. Segments two and three have the largest magnitudes of higher frequency due to the combination of the effects of wake oscillation and passage of vortices shed from the leading edge. A similar approach is used to analyze $\Gamma' = (\Gamma - \overline{\Gamma})$ to predict the form of the correlation.

Since there are approximately 200 different segment-λ combinations where Γ is stored, the time series of Γ' will be plotted for a few representative distances from the surface. Similar to $\overline{Cp}$, $\overline{\Gamma}$ represents the time averaged Γ function. The time series fluctuation about the mean is plotted at λ close to the surface, λ near the edge of the separated region, and λ far from the surface for segments one and three. The bins are separated by a constant distance $\lambda/D = 0.03$, so the closest distance plotted is $\lambda/D = 0.15$, next is $\lambda/D = 0.45$ and the outer distance plotted is $\lambda/D = 1.2$. Figures 6.2.3 and 6.2.4 show the time series of Γ' for bins in segments one and three, respectively.

In both segments one and three, the magnitude of fluctuations in Γ decrease as distance from the cylinder surface increases. In the time series for segment one, there is some low frequency oscillation visible, but the fluctuations are predominantly high frequency. In λ far from the surface, both the magnitude and frequency

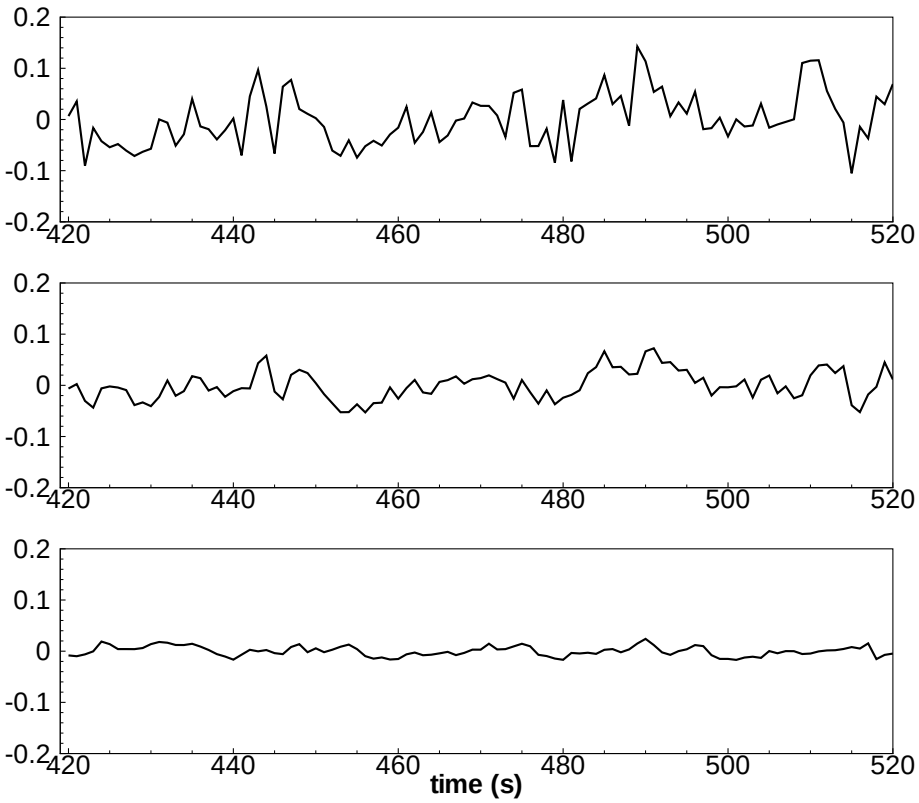


Figure 6.2.3: Time series of Γ' in segment one. From top: $\lambda/D = 0.15$, $\lambda/D = 0.45$, $\lambda/D = 1.2$.

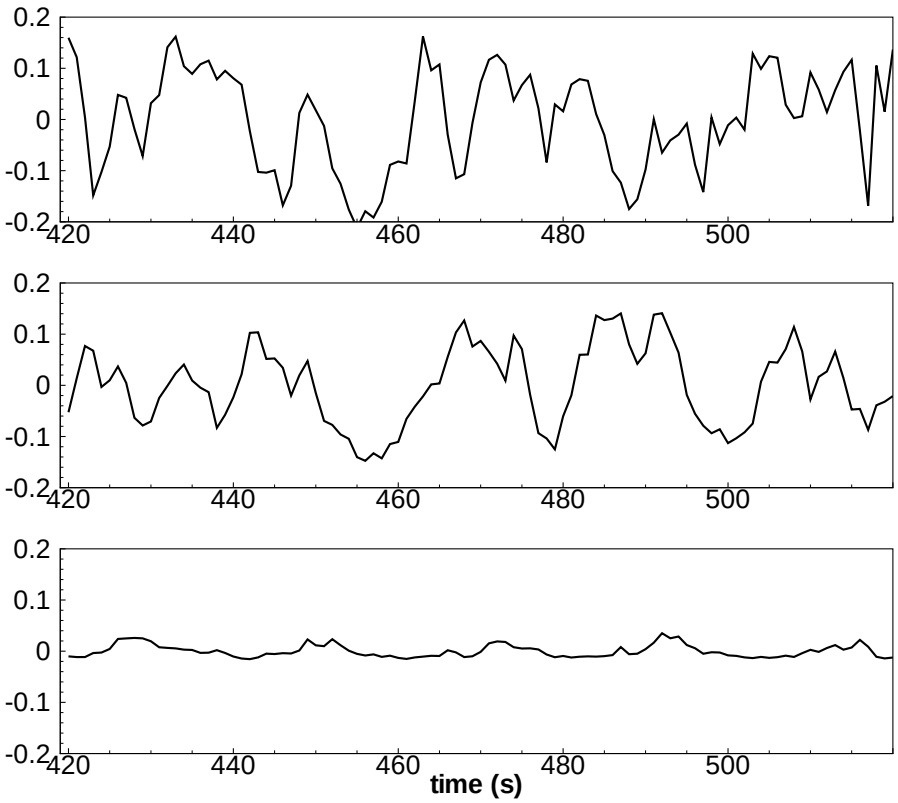


Figure 6.2.4: Time series of Γ' in segment three. From top: $\lambda/D = 0.15$, $\lambda/D = 0.45$, $\lambda/D = 1.2$.

of the fluctuations are low. This decline in magnitude is expected because there are few flow structures outside of the separated region and the wake, so Γ will be near zero for most instants in time.

The time series plots of Γ' relative to segment three, Fig. 6.2.4, reveal a stronger low frequency oscillation than the plots of segment one due to the effect of wake oscillation. For λ close to the surface, some higher frequency fluctuations are present due to swirling structures within the separated region. These high frequency oscillations are lower in magnitude in λ near the edge of the separation. For λ far from the surface, the magnitude of fluctuations is low; the fluctuations observed are low frequency and are caused by the wake oscillation.

The nature of both the surface pressure and the Γ fluctuation time series are oscillatory, so the correlation between the two will be oscillatory. To determine the correlation frequency, a time series for the value of the correlation is plotted for each of the distances plotted in Fig. 6.2.3 and 6.2.4. Figures 6.2.5 and 6.2.6 show the time series plot of the correlation in segments one and three, respectively, for λ near the surface, near the edge of the separated region, and far from the surface.

For segment one, the correlation time series shows high frequency oscillations near the surface and lower frequency variations at λ further from the surface. The magnitude of the correlation is less than 0.5 over time for each λ, indicating a moderate covariance between the surface pressure and the swirl in the flow at those distances. As distance from the surface increases, the correlation oscillation frequency decreases.

The correlation magnitudes for segment three are higher than those of segment

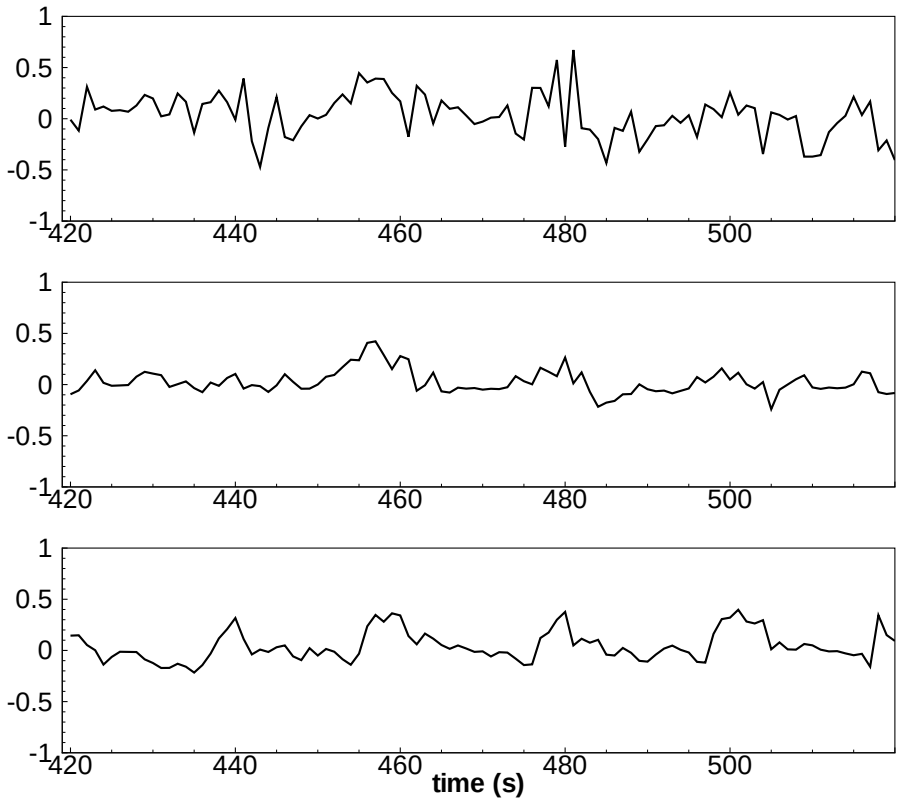


Figure 6.2.5: Time series of C_{S_j} in segment one. From top: $\lambda/D = 0.15$, $\lambda/D = 0.45$, and $\lambda/D = 1.2$.

one. This indicates a stronger covariance between the surface pressure fluctuation and the fluctuations of the swirl in the flow. At each of the three distances plotted from segment three, the correlation signals exhibit dominant low frequency oscillations. In the correlation plots for segment one, there are high frequency fluctuations near the surface that reduce in magnitude as distance from the cylinder surface increases. A spectral analysis, shown for several distances λ in segments one and three in Figures 6.2.7 and 6.2.8, will help determine the dominant frequencies in these signals.

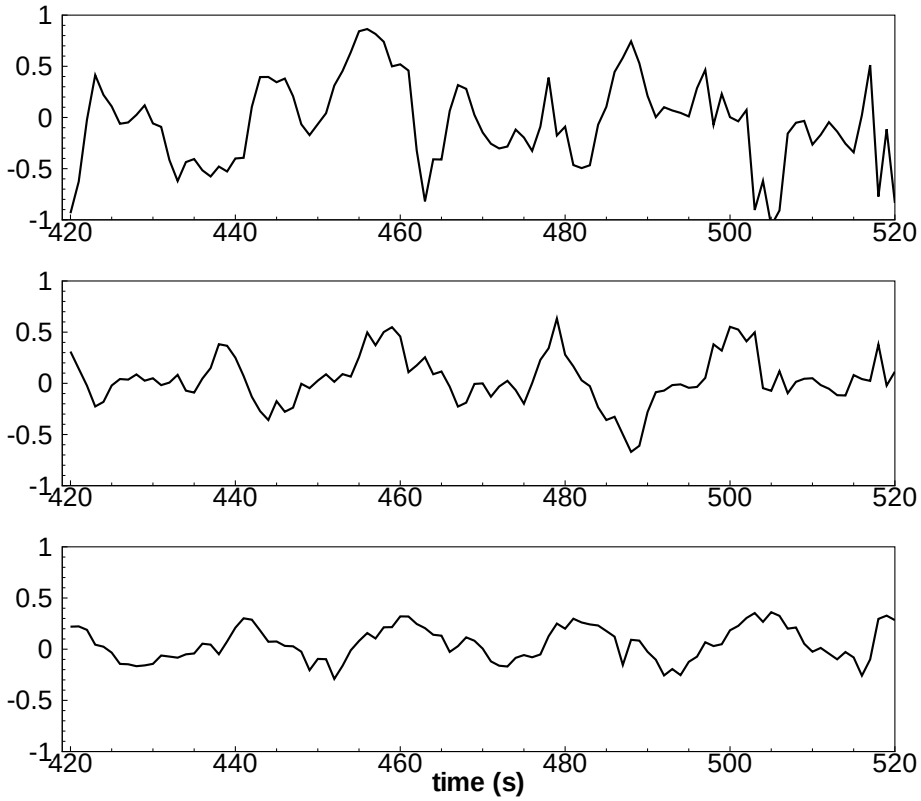


Figure 6.2.6: Time series of C_{S_j} in segment three. From top: $\lambda/D = 0.15$, $\lambda/D = 0.45$, and $\lambda/D = 1.2$.

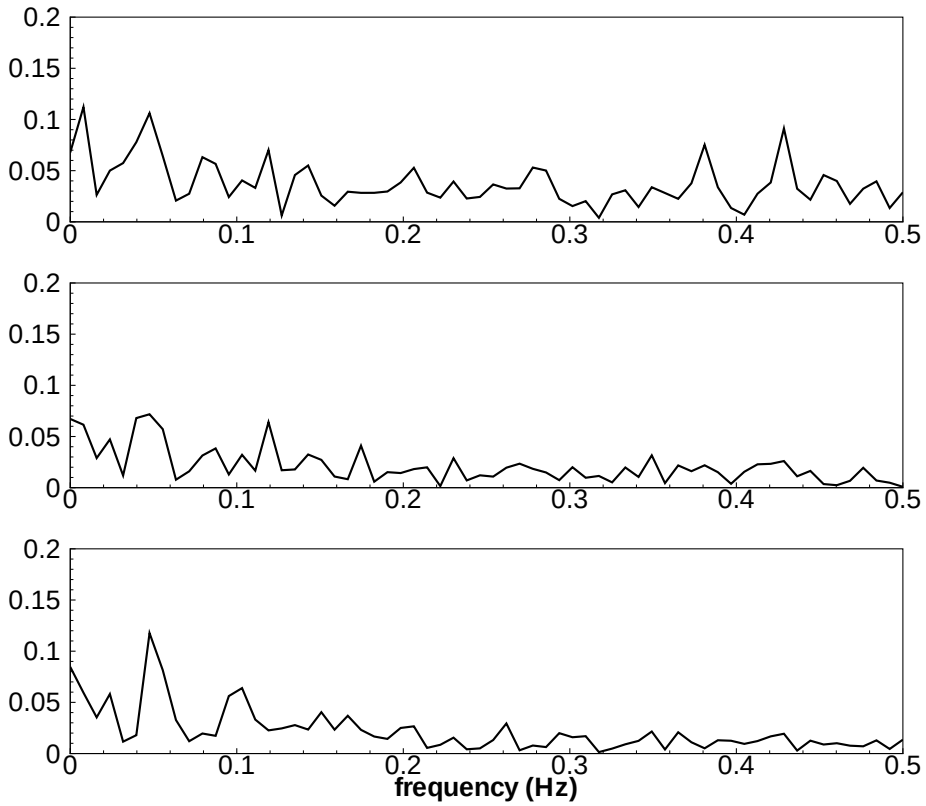


Figure 6.2.7: Spectral analysis of C_{S_j} of segment 1. From top: $\lambda/D = 0.15$, $\lambda/D = 0.45$, and $\lambda/D = 1.2$.

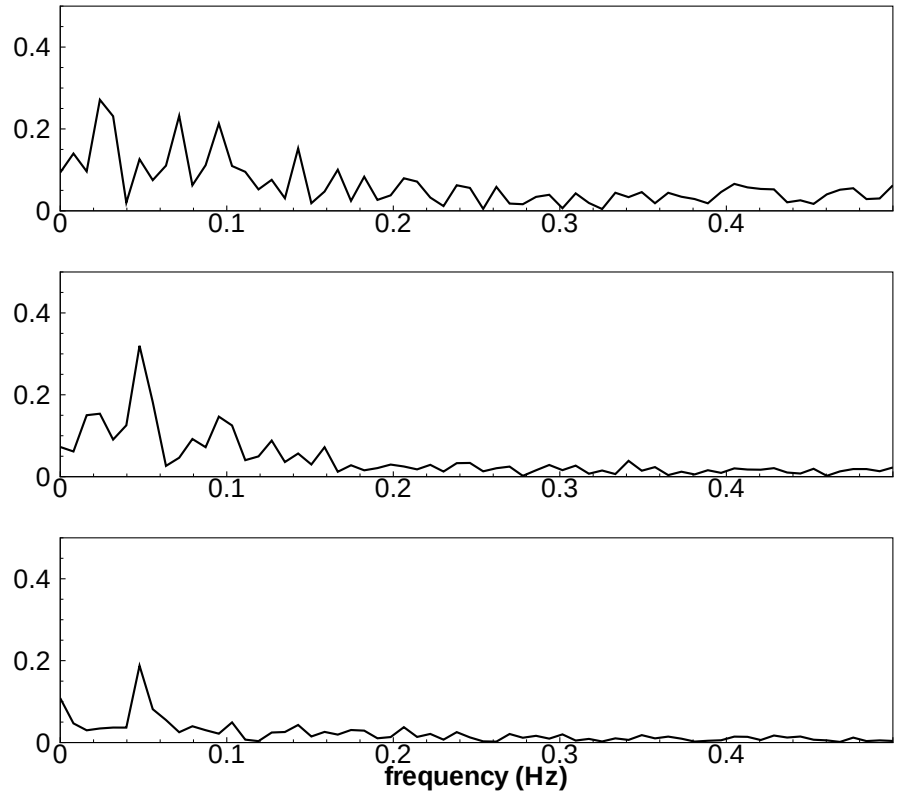


Figure 6.2.8: Spectral analysis of C_{S_j} of segment 3. From top: $\lambda/D = 0.15$, $\lambda/D = 0.45$, and $\lambda/D = 1.2$.

The spectral analysis of the correlation for segment one shows that λ close to the surface has a wide range of frequency content. This result is expected because of the fluctuations seen in the Γ signal for that λ and fluctuations in Cp for segment one. For the two other distances plotted, the spectral analysis reveals a peak frequency of 0.05 ± 0.005Hz, which is the same as the frequency seen in the surface pressure signal. A similar frequency response is shown in Fig. 6.2.8 for each value of λ plotted for segment three. The signal from λ close to the cylinder surface exhibits a wide range of frequencies, whereas the signals from both values of λ further from the surface have one main peak at $f = 0.05 \pm 0.005$Hz.

To further investigate the peaks in the correlation time series, time series of Cp$'$, Γ', and C_{S_j} are plotted together for the same segment and distance λ. This is done to determine which features in the pressure and Γ signals lead to the peaks in the correlation plots. Figure 6.2.9 shows the time series of pressure and Γ fluctuations with the correlation for segment three, $\lambda/D = 0.6$.

It is observed that each of the maxima in the correlation time series corresponds to times when the fluctuations of pressure and Γ are both negative. This is observed to be true for each segment and value of λ analyzed (not shown). Another observation is that the minima in the correlation time series tend to correspond to instants in time when Γ' is near a maxima and Cp$'$ is less than zero. This indicates that the presence of a vortex causes the pressure on the cylinder surface to be lower than its mean value regardless of the direction of swirl. Based on this, and because the purpose of the correlation is to determine the effect of vortex passage on the surface regardless of the direction of swirl, it is useful to look at the magnitude of

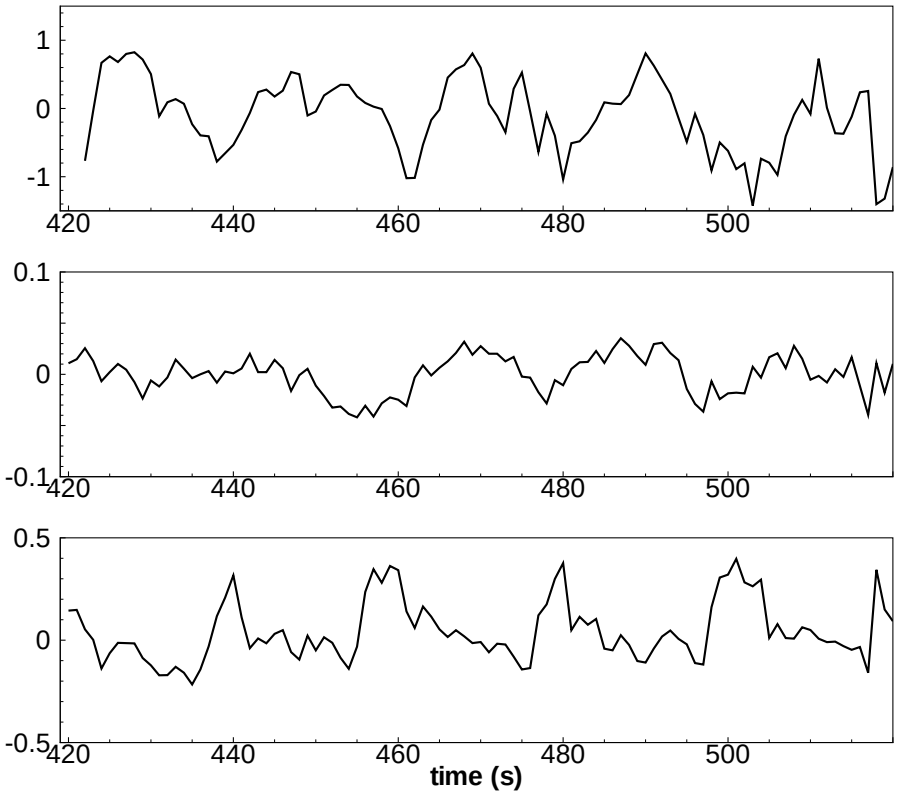


Figure 6.2.9: Segment three time series, from top: Cp′, Γ′ at $\lambda/D = 0.6$, and C_{S_j} at $\lambda/D = 0.6$.

the correlation. At each instant in time, $|C_{S_j}|$ will be non-zero where there is some interaction between the swirl in the flow and the surface pressure.

In addition to how $|C_{S_j}|$ changes in time with the fluctuations in pressure and Γ, it is necessary to investigate how the correlation changes in space for an instant in time. At each instant in time, the surface pressure is constant for each segment, so the only variation in $|C_{S_J}|$ will come from Γ' as a function of the distance from the surface of the cylinder. Two distinct moments in time are examined: (i) Figure 6.2.10 shows an instant time when the wake is oscillating towards the bottom of the cylinder and (ii) Figure 6.2.11 shows an instant in time when the wake is oscillating towards the top surface.

When the wake is oscillating away towards the bottom cylinder surface, the separated region is small and the vortices shed from the leading edge are close to the surface. The correlation curves have a peak near the surface for all four segments because of the proximity of the vortices. Although there are no strong swirls above the surface plane in the wake, there is still some correlation value because there is a difference between Γ at this instant and the time average value for Γ at these distances.

When the wake oscillates towards the top surface, the separated region is large and the vortices remain above the top surface plane as they are convected downstream. This is most visible in the correlation curves for segments three and four, which have peaks at the distance where the cluster of vortices is visible in the contours of Γ. Even though there is vortex activity near the surface of the square cylinder, the correlation values are low. This is because the difference between Γ

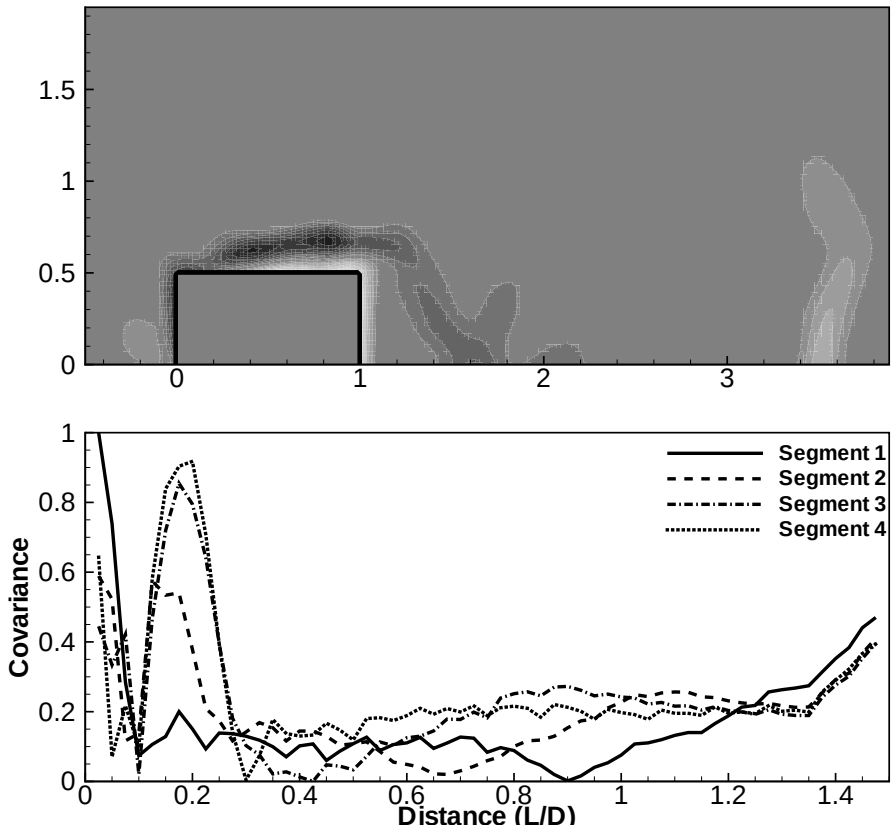


Figure 6.2.10: Magnitude of correlation curves and Γ contours for $t = 486s$ for all surface segments.

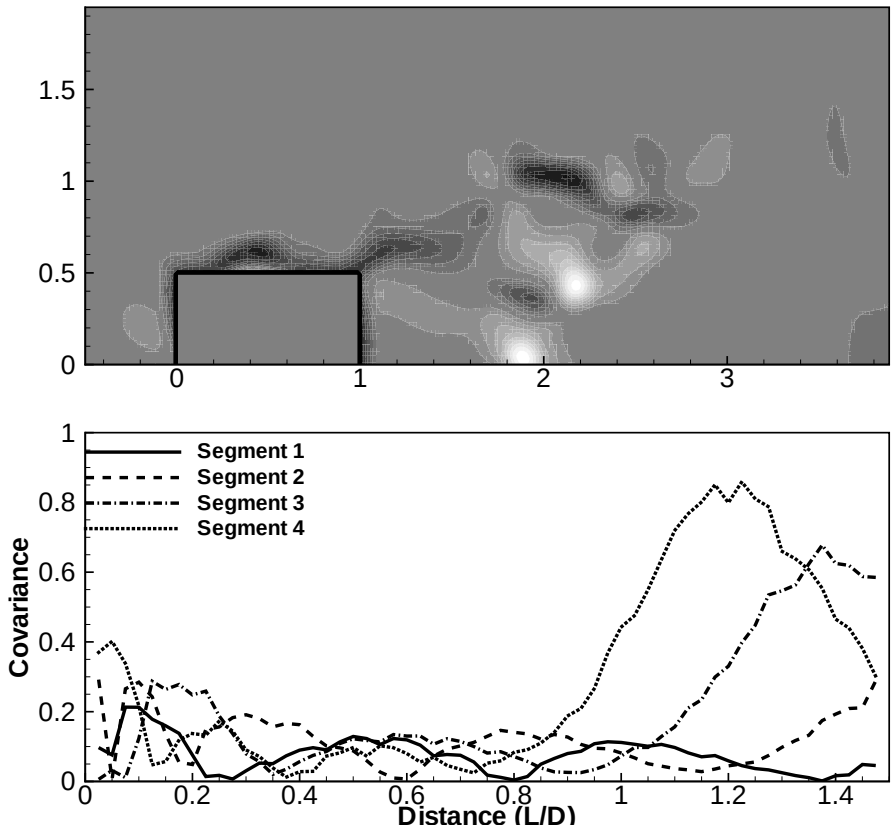


Figure 6.2.11: Magnitude of correlation curves and Γ contours for $t = 499s$ for all surface segments.

and $\overline{\Gamma}$ is small in these regions.

It is clear that the fluctuations in Γ have a strong effect on the correlation curves. One way of quantifying this effect is to examine the mean contours of $|\Gamma|$ and the curves of $|C_{S_j}|$ averaged over time. This will give information about which segments are most affected, on average, by the passage of vortical structures. Figure 6.2.12 shows the mean contours of $|\Gamma|$ with the curves of the mean correlation magnitude.

On the time average, each segment has some magnitude of correlation between the passage of vortices and the surface pressure. The peak correlation is for λ nearest the surface. Segments three and four have a second peak at a distance of $\lambda/D \approx 0.2$, which corresponds with the center of the mean swirl. From this plot, it appears that both segment three and segment four are, on average, more affected by swirls in the flow near the surface than segments one and two. All four segments show low correlation in the mean for distances further than $\lambda/D = 0.5$ from the surface.

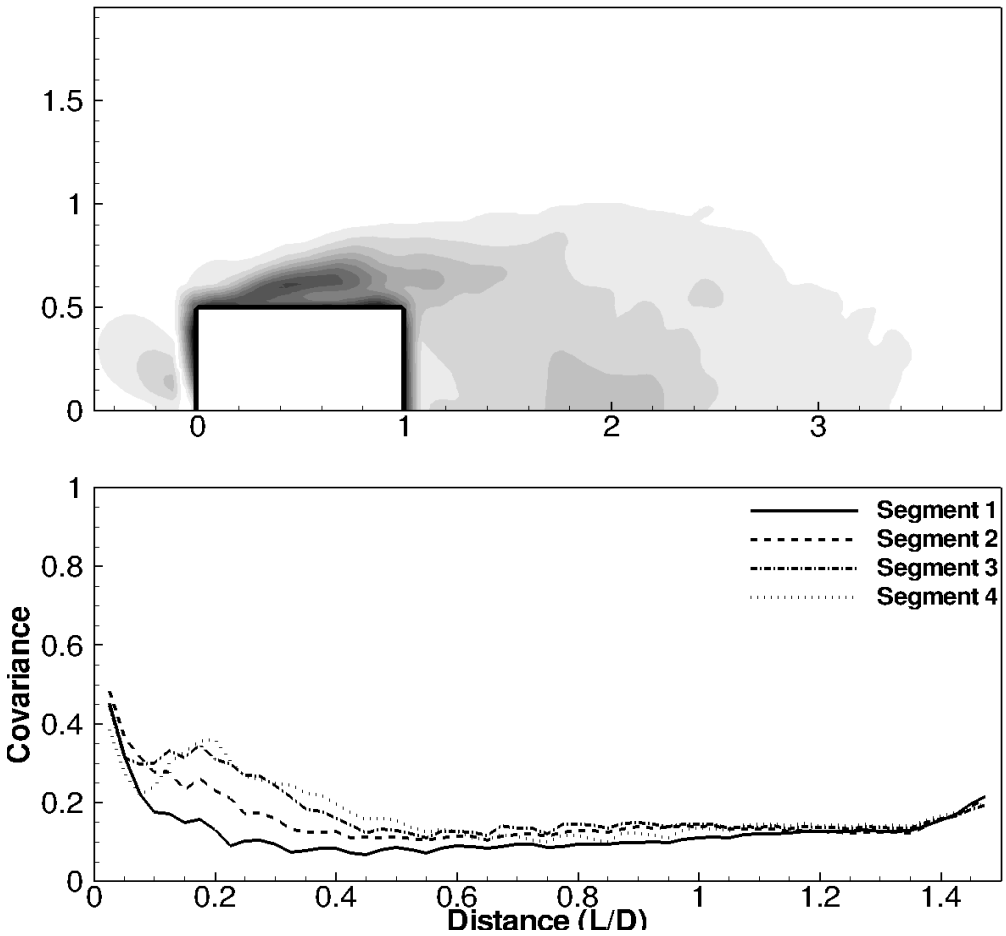


Figure 6.2.12: Mean contours of $|\Gamma|$ and time-averaged curves of $|C_{S_j}|$.

Chapter 7 – Conclusion and recommendations

In this research a computational approach has been used to study the separating flow over a square cylinder. The predicted values of the mean flow are compared to experimental results for flow over a square cylinder and are found to be in good agreement. The discrepancies between the simulation and experiment may be explained by insufficient grid resolution around the cylinder and in the wake and the coarse grid in the span direction. CFD is ideal for generating data for vortex detection because the pressure and pressure gradients are computed with the velocity data in the simulation.

The data are filtered to investigate different scales of flow features in turbulent separated flow. Decomposing the flow into different scales helps isolate the scales of swirl that have the most influence on the surface forces. Larger scale swirls contain more energy than the small scales, and are easier to visually identify. Larger scale features have longer time scales as well, so they are traceable from time step to time step, as shown in time series of all vortex detection methods tested.

Even in the low-pass filtered data sets, there are features of different spacial scale. Different methods of vortex detection preferentially detect different scales of vortex depending on how they are computed. The derivative-based methods are best at detecting the vortices on the order of the grid spacing. Derivative-based methods may not detect large swirls that are more than an order of magnitude

larger than the grid spacing. Also, since the vortices detected by the eigenvalue methods are small, they have short time scales and are more likely to disappear between time frames, making them more difficult to track.

The two derivative-based methods developed are λ_2 and λ_p. Both methods locate pressure minima in the flow; λ_2 is calculated using velocity derivatives and λ_p is based on the Hessian of the pressure. Both detection methods locate small swirls in the flow that correspond to locations where the gradients are strong on the order of the grid spacing. The location and strength of vortices detected with λ_p match well with the results of λ_2. The differences between the two methods are due to unsteady straining in the flow and viscous effects that are neglected when computing λ_2, but are included with the calculation of λ_p.

In contrast to the small scales detected using the derivative-based approaches, the integral-based methods are best at locating swirls of size on the order of the integral area. The two integral based methods are the Γ function and the Γ_p function. The Γ function represents the area-averaged circulation around each point in the flow and is based on the velocity vector field. The Γ_p method is a measure of where the rotated pressure gradients are most circular. These locations are found by applying the Γ function to the rotated pressure gradient field.

The Γ function locates regions of swirl that may include groups of small vortices. It is best at identifying large scale features in the flow, such as the large recirculation in the near wake of the cylinder, and the large scale features on the edges of the wake. The Γ_p approach also detects large scale features well, but because it is based on derivatives, it is more sensitive to changes on the order of

the grid spacing. This is most clear nearest the leading edge where Γ_p detects individual swirls and Γ smears the small vortices together into one large swirl region.

Large scale features are of most interest because they will have the greatest effect on the surface forces. When developing a correlation between the surface pressure and vortex passage in the flow, a vortex detection method that is smooth from time slice to time slice is desired. This means that features must remain visible between instants in time so they are trackable. These criteria lead to using the Γ function with the pressure coefficient on the surface to generate a two-point covariance that shows the correlation between surface pressure and swirl in the flow.

Time signals of the surface pressure and Γ fluctuations in the flow are found to be periodic. The correlation between the two variable is also periodic, and is highly influenced by the oscillation of the wake. It is observed that maxima in the correlation as a function of time are a result of the product of a minima in Γ' and negative values of Cp'. Minima in the correlation tend to correspond to instants in time when Γ' in the bin is at a peak and the pressure signal fluctuations are negative. This means that peaks in the correlation magnitude correspond to the presence of a swirl in the flow of either direction.

Correlation curves are examined at instants in time to investigate the effect of Γ on the correlation as a function of distance away from the cylinder surface. It is observed that locations in the flow that have non-zero fluctuations in the Γ function lead to maxima in the correlation curves. On the time average, the correlation is

strongest near the surface of the cylinder. Segments on the downstream half of the square cylinder have a secondary maxima near the edge of the separated region. Both of the segments on the downstream half of the cylinder are affected about the same by the passage of vortices in the flow, so would be ideal locations for pressure sensors.

The correlation results are somewhat inconclusive, likely due to the symmetric nature of flow over a cylinder. Large scale wake oscillation obscures the effect of the passage of vortices from the leading edge. It is recommended that the next case to which the correlation is applied should be a non-symmetric flow, such as flow over half of a cylinder, or separated flow over an airfoil at high angle of attack. A combination of experimental data and a simulation that is validated for the same case would be a ideal dataset for refinement and testing of the correlation between surface pressure and vortex passage.

Future research should simulate flow over an airfoil for which experimental lift and drag, as well as flow velocity data is collected. By applying the vortex detection methods to flow over an airfoil and linking the times of velocity data with lift and drag data, it should be possible to get a more solid link between the passage of vortices in the flow and their effect on the lift and drag. If the simulation is validated against such experimental data, it would be possible to extend the correlation to the surface pressure at different locations on the airfoil, further refining any control scheme that may come of such research.

APPENDICES

Appendix A – Vortex detection code

Both the vortex detection code and the correlation code are written in fortran 90, which allows for free formatting and much less restrictive syntax requirements than fortran 77. Fortran was used for post-processing because of its speed for mathematical applications and because it is easily parallelized for large data sets. Both of the main post-processing codes are included below as in-line PDF pages to improve the readability of the code and to preserve the exact syntax and formatting of the code for any who wish to use the it in the future.

```
program vortexdetect
use matfun
use vecfun
implicit none

! The purpose of this program is to read grid indexed data for velocity
! from a text file, or several sequential text files, and compute the
! gamma function.  The gamma function is a measure of the circulation
! in the flow and can be used to locate vorticies.  The strength of the
! vorticies in the flow are also found.

integer, parameter  :: xmax=300, ymax=200, tmax=1 ! range of the data set input

real*8, parameter  :: speed = 0.3318, angle=0.0 ! parameters of data set
integer, parameter  :: rad = 5   ! "radius" of gamma - 4 is a 9x9 grid
integer     :: col, row , numpts, subx, suby ! column and row loop counters
real*8, dimension(xmax) :: xpos  ! x and y position in millimeters
real*8, dimension(ymax) :: ypos
real*8, dimension(1,xmax) ::  xpost  ! for the transpose of the xposition. used
for output
real*8, dimension(xmax,ymax) :: xvel, yvel, gam, str, egval_r1, egval_r2,
egval_i, press
real*8, dimension(xmax,ymax) :: dudx,dudy,dvdx,dvdy, dpdx, dpdy, dpdx_p, dpdy_p,
gam_p, str_p
        ! ^arrays for the node position, velocity, gamma, and strength
real*8, dimension(xmax,ymax) :: egval_v1, egval_v2, egval_vi, isvortex, press_xy,
egval_p1,egval_p2, egval_pi
real*8, dimension(xmax,ymax) ::press_dxy, press_dxx, press_dyy, press_dyx
real*8, dimension(xmax,ymax) :: xvel_raw,yvel_raw,xvel_hp,yvel_hp,xvel_lp,yvel_lp
real*8, dimension(xmax,ymax) :: temp_raw1, temp_raw2, temp_raw3, temp_raw4
real*8, dimension(xmax,ymax) ::
dpdx_filt,dpdy_filt,dudx_filt,dudy_filt,dvdx_filt,dvdy_filt
real*8, dimension(xmax,ymax) ::
press_dxx_filt,press_dyy_filt,press_dxy_filt,press_dyx_filt
real*8, dimension(2,2) :: derivarray
real*8, dimension(2) :: pm, um  ! vector from centroid to edge, velocity at edge
of subregion
real*8, dimension(3,3) :: deriv, strainrate, rotation, s_square, o_square,
s_plus_o ! arrays for S and omega
real*8, dimension(2,2) :: hessian_p
integer, dimension(xmax,ymax) :: mask
real*8     :: dummygam, dummystr, tempo1, tempo2 ! intermediate values, time
counters
real*8     :: dummyegval_i, dummyegval_r1, dummyegval_r2, delta_x, delta_y
integer ::      filtersize=20, stdev, headlines, dpressfilt=20, d2pressfilt=20,
filtvelgrad=20
!integer  :: filtersize=1, stdev, headlines, dpressfilt=1, d2pressfilt=1,
filtvelgrad=1
integer     :: icount,jcount,kcount,tcount,tcount2, lcount, mcount, ic, jc, ic2,
jc2
!integer    :: test12
real*8     :: test1, test2, test12, pdum, test13, test14, test15 ! dummy variables
real*8     :: test3, test4, test5, test6, test7, test8, test9, test10, test11
```

```
real*8  :: dummyegval_1, dummyegval_2, dummyegval_i2, dummyegval_p1,dummyegval_p2
 character*50   :: filename, junk, output_file,guoning_file, strength_file,
egval_file, timefile='timehistory.dat'

!-------------------------------------------------------------!
! The first order of business is to open the file and read
!  the relevant data.  For the time being, it is opening the
!  entire data file, but only reading the first time step
! of data.  In the future, it should be simple enough
!  to set up to read the entire data set, sequentially finding
! the gamma values for all time steps.
!-------------------------------------------------------------!

stdev=4
gam = 0.0
str = 0.0
egval_i=0.0
egval_pi = 0.0
egval_vi = 0.0
egval_p1 = 0.0
egval_p2 = 0.0
egval_v1 = 0.0
egval_v2 = 0.0

headlines = 11

 call cpu_time(tempo1)

open(3,file='5slicefiles.dat')
open(4,file='5vortdetfiles.dat')
!open(7,file='guoningfiles.dat') !Guoning's data files
!open(5,file='strfiles.txt')
!open(6,file='egvalfiles.txt')
open(5,file=timefile)
 write(5,*) '"TIME"'
 write(5,*) '"GAMMA_1"'
 write(5,*) '"GAMMA_2"'
 write(5,*) '"GAMMA_3"'
 write(5,*) '"GAMMAP_1"'
 write(5,*) '"GAMMAP_2"'
 write(5,*) '"GAMMAP_3"'


do tcount=1,tmax
 read(3,*) filename
 read(4,*) output_file
! read(7,*) guoning_file !Guoning's data files

print*,'reading data from: ',filename
print*,'output data is: ',output_file

 open(9, file=filename)
 do lcount=1,headlines
```

```
  read(9,*) junk
 end do

 do jcount=1,ymax  ! Populates the data arrays for calculating gamma and the
strength
  do icount=1,xmax ! 'test' variables are just dummy variables that hold a single
value
!   read(9,*)  test1, test2, pdum, test3, test4, test5, test6, test7,test8,
test9,&
!    test10, test11, test12 , test13, test14  ! This is for the full output
   read(9,*) test1, test2, pdum, test3, test4, test14 !!test5, test6, test14
!this is for the small output
   xpos(icount) = test1
   ypos(jcount) = test2
   press(icount,jcount) = pdum
   xvel(icount,jcount) = test3
   yvel(icount,jcount) = test4
   ! The following lines are for the pressure gradient vector and its
perpindicular
   !dpdx(icount,jcount) = test5
   !dpdy(icount,jcount) = test6
   !dpdx_p(icount,jcount) = (test6)
   !dpdy_p(icount,jcount) = (-test5)

   ! This is where the second derivatives of pressure are read in
   !press_dxx(icount,jcount) = test7
   !press_dyy(icount,jcount) = test8
   !press_dxy(icount,jcount) = test9
   !press_dyx(icount,jcount) = test9

   !dudx(icount,jcount) = test10
   !dudy(icount,jcount) = test11
   !dvdx(icount,jcount) = test12
   !dvdy(icount,jcount) = test13
   mask(icount,jcount) = int(test14)
  end do
 end do
 close(9)

delta_x = abs(xpos(xmax)-xpos(1))/xmax
delta_y = abs(ypos(ymax)-ypos(1))/ymax
print*,'space steps: (x,y)',delta_x,delta_y

!!!! Take derivatives of velocity to compare to actual values
!! syntax: centdiff4(array(:,:),stepx,stepy,mask(:,:),derivx(:,:),derivy(:,:))
print*,'Taking derivatives - 4th order accurate'
 call centdiff4(xvel(:,:),delta_x,delta_y,mask(:,:),dudx(:,:),dudy(:,:))
 !call
centdiff4(xvel(icount,:),delta_y,jcount,mask(icount,jcount),dudy(icount,jcount))
 call centdiff4(yvel(:,:),delta_x,delta_y,mask(:,:),dvdx(:,:),dvdy(:,:))
 !call
centdiff4(yvel(icount,:),delta_y,jcount,mask(icount,jcount),dvdy(icount,jcount))
```

```
!derivatives of the pressure... dp/dx and dp/dy should be output from code, but
for AF it isn't
 call centdiff4(press(:,:),delta_x,delta_y,mask(:,:),dpdx(:,:),dpdy(:,:))
 !call
centdiff4(press(icount,:),delta_y,jcount,mask(icount,jcount),dpdy(icount,jcount))
 dpdx_p(:,:) = (-dpdy(:,:))
 dpdy_p(:,:) = dpdx(:,:)

!derivatives of pressure gradient...
 call centdiff4(dpdx(:,:),delta_x,delta_y,mask(:,:),press_dxx(:,:),press_dxy(:,:))
 !call
centdiff4(dpdx(icount,:),delta_y,jcount,mask(icount,jcount),press_dxy(icount,jcoun
t))
 call centdiff4(dpdy(:,:),delta_x,delta_y,mask(:,:),press_dyx(:,:),press_dyy(:,:))
 !call
centdiff4(dpdy(icount,:),delta_y,jcount,mask(icount,jcount),press_dyy(icount,jcoun
t))


!-----------------------------------------------------!
! This calls a gaussian filter using parameters specified
! above.  The filter itself is a high pass filter, but
! can be used to get low pass data by subtracting the hp
! from the raw field data.
!-----------------------------------------------------!
print*,'going to filter data now'
xvel_raw(:,:)=xvel(:,:)
yvel_raw(:,:)=yvel(:,:)
print*,' Velocity ....'
!print*,xvel_raw(20,20),yvel_raw(20,20)

 call gauss2(xvel_raw,xvel_hp,filtersize,stdev,mask(:,:),delta_x,delta_y)
 call gauss2(yvel_raw,yvel_hp,filtersize,stdev,mask(:,:),delta_x,delta_y)
!xvel_lp(:,:)=xvel_raw(:,:)-xvel_hp(:,:)
!yvel_lp(:,:)=yvel_raw(:,:)-yvel_hp(:,:)
xvel(:,:)=xvel_hp(:,:)
yvel(:,:)=yvel_hp(:,:)
! xvel(:,:)=xvel_raw(:,:)-xvel_hp(:,:)
! yvel(:,:)=yvel_raw(:,:)-yvel_hp(:,:)


 print*,' pressure gradient....'
 !Filter the rotated pressure gradient data
 temp_raw1(:,:)=dpdx_p(:,:)
 temp_raw2(:,:)=dpdy_p(:,:)
 call gauss2(temp_raw1,dpdx_filt,dpressfilt,stdev,mask(:,:),delta_x,delta_y)
 call gauss2(temp_raw2,dpdy_filt,dpressfilt,stdev,mask(:,:),delta_x,delta_y)
 dpdx_p(:,:) = dpdx_filt(:,:)
 dpdy_p(:,:) = dpdy_filt(:,:)
 !dpdx_p(:,:) = temp_raw1(:,:) - dpdx_filt(:,:)
 !dpdy_p(:,:) = temp_raw2(:,:) - dpdy_filt(:,:)
```

```
print*,' velocity gradient...'
 temp_raw1 = 0.0;
 temp_raw2 = 0.0;
 temp_raw3 = 0.0;
 temp_raw4 = 0.0;
 !Filter the velocity gradient data
 temp_raw1(:,:)=dudx(:,:)
 temp_raw2(:,:)=dudy(:,:)
 temp_raw3(:,:)=dvdx(:,:)
 temp_raw4(:,:)=dvdy(:,:)
 call gauss2(temp_raw1,dudx_filt,filtvelgrad,stdev,mask(:,:),delta_x,delta_y)
 call gauss2(temp_raw2,dudy_filt,filtvelgrad,stdev,mask(:,:),delta_x,delta_y)
 call gauss2(temp_raw3,dvdx_filt,filtvelgrad,stdev,mask(:,:),delta_x,delta_y)
 call gauss2(temp_raw4,dvdy_filt,filtvelgrad,stdev,mask(:,:),delta_x,delta_y)
 ! High pass:
 dudx(:,:) = dudx_filt(:,:)
 dudy(:,:) = dudy_filt(:,:)
 dvdx(:,:) = dvdx_filt(:,:)
 dvdy(:,:) = dvdy_filt(:,:)
 ! Low pass:
 !dudx(:,:) = temp_raw1 - dudx_filt(:,:)
 !dudy(:,:) = temp_raw2 - dudy_filt(:,:)
 !dvdx(:,:) = temp_raw3 - dvdx_filt(:,:)
 !dvdy(:,:) = temp_raw4 - dvdy_filt(:,:)

 print*,' pressure hessian...'
 temp_raw1 = 0.0;
 temp_raw2 = 0.0;
 temp_raw3 = 0.0;
 temp_raw4 = 0.0;
 !Filter the pressure hessian data
 temp_raw1=press_dxx(:,:)
 temp_raw2=press_dyy(:,:)
 temp_raw3=press_dxy(:,:)
 temp_raw4=press_dyx(:,:)
 call gauss2(temp_raw1,press_dxx_filt,d2pressfilt,stdev,mask(:,:),delta_x,delta_y)
 call gauss2(temp_raw2,press_dyy_filt,d2pressfilt,stdev,mask(:,:),delta_x,delta_y)
 call gauss2(temp_raw3,press_dxy_filt,d2pressfilt,stdev,mask(:,:),delta_x,delta_y)
 call gauss2(temp_raw4,press_dyx_filt,d2pressfilt,stdev,mask(:,:),delta_x,delta_y)
 ! high pass:
 press_dxx(:,:) = press_dxx_filt(:,:)
 press_dyy(:,:) = press_dyy_filt(:,:)
 press_dxy(:,:) = press_dxy_filt(:,:)
 press_dyx(:,:) = press_dyx_filt(:,:)
 ! Low Pass:
 !press_dxx(:,:) = temp_raw1 - press_dxx_filt(:,:)
 !press_dyy(:,:) = temp_raw2 - press_dyy_filt(:,:)
 !press_dxy(:,:) = temp_raw3 - press_dxy_filt(:,:)
 !press_dyx(:,:) = temp_raw4 - press_dyx_filt(:,:)

!------------------------------------------------------------!
! This section of the program is to actually calculate the
! gamma function and the strength.  The data will be written
```

```
! to a file that Tecplot can read for plotting and
!  further post-processing.
!-------------------------------------------------------------!

 call gammadet(xpos,ypos,xvel,yvel,mask,rad,gam,str)
 call gammadet(xpos,ypos,dpdx_p,dpdy_p,mask,rad,gam_p,str_p)

! ---------------------------------------------------------
! This section of the code uses a critical point analysis
! based on the imaginary part of the eigenvalue of the
! derivative matrix (basically a measure of vorticity)
! to detect vorticies.
! ---------------------------------------------------------

 print*, 'Starting velocity eigenvalue loops'
 do row = 2,ymax-1
  do col=2,xmax-1
   ! Calculate the derivative matrix
   !call centdiff(xvel(:,row),delta_x,col,derivarray(1,1))
   !call centdiff(xvel(col,:),delta_y,row,derivarray(1,2))
   !call centdiff(yvel(:,row),delta_x,col,derivarray(2,1))
   !call centdiff(yvel(col,:),delta_y,row,derivarray(2,2))

   derivarray(1,1) = dudx(col,row)
   derivarray(1,2) = dudy(col,row)
   derivarray(2,1) = dvdx(col,row)
   derivarray(2,2) = dvdy(col,row)


   ! Calculate the eigenvalues, store in dummy variables temporarily
   call eigenval2(derivarray,dummyegval_r1,dummyegval_r2,dummyegval_i)

   egval_i(col,row) = dummyegval_i
   egval_r1(col,row) = dummyegval_r1
   egval_r2(col,row) = dummyegval_r2

  end do
 end do

 print*,'Starting Hessian and S**2 + O**2 eigenvalue loops'
 do subx=2,xmax-1
  do suby=2,ymax-1
   !if(mask(subx,suby) /= 0) then
    ! fill in a derivatives matrix of u(x,y)
    deriv(1,1) = dudx(subx,suby)
    deriv(1,2) = dudy(subx,suby)
    deriv(2,1) = dvdx(subx,suby)
    deriv(2,2) = dvdy(subx,suby)
    deriv(:,3) = 0.0
    deriv(3,:) = 0.0

    ! Split into symmetric and antisymmetric parts
    call sym3(deriv,strainrate)
```

```
    call asym3(deriv, rotation)
    ! square the symmetric and antisymmetric parts
    call matsquare3(strainrate,s_square)
    call matsquare3(rotation,o_square)

    s_plus_o(1,1) = s_square(1,1) + o_square(1,1)
    s_plus_o(1,2) = s_square(1,2) + o_square(1,2)
    s_plus_o(2,1) = s_square(2,1) + o_square(2,1)
    s_plus_o(2,2) = s_square(2,2) + o_square(2,2)

    ! Calculate and store the eigenvalues of  S^2 plus Omega^2
    call eigenval2(deriv(1:2,1:2),dummyegval_1,dummyegval_2, dummyegval_i2)
    egval_v1(subx,suby) = dummyegval_1
    egval_v2(subx,suby) = dummyegval_2
    egval_vi(subx,suby) = dummyegval_i2

    ! Calculate and store the eigenvalues of the hessian of the pressure
    hessian_p(1,1)=press_dxx(subx,suby)
    hessian_p(1,2)=press_dxy(subx,suby)
    hessian_p(2,1)=press_dyx(subx,suby)
    hessian_p(2,2)=press_dyy(subx,suby)

   call eigenval2(hessian_p,dummyegval_p1,dummyegval_p2,dummyegval_i2)
   egval_p1(subx,suby) = dummyegval_p1
   egval_p2(subx,suby) = dummyegval_p2
   egval_pi(subx,suby) = dummyegval_i2

  end do
 end do


 print*,'writing output files'

 write(5,*) tcount, gam(50,127), gam(75,136), gam(100,145), gam_p(50,127),
gam_p(75,136), gam_p(100,145)

 open(37, file=output_file)
! open(38, file=guoning_file)
 write(37,*) 'TITLE = "Vortex detection data" '
 write(37,*) 'VARIABLES = "X"'
 write(37,*) '"Y"'
 write(37,*) '"U"'
 write(37,*) '"V"'
 write(37,*) '"PRESS"'
 write(37,*) '"GAMMA"'
 write(37,*) '"GAMMA_P"'
 write(37,*) '"LAMBDA_V"'
 write(37,*) '"LAMBDA_P"'
 write(37,*) '"MASK"'
 write(37,*) 'ZONE T = "Rectangular Zone"'
 write(37,*) 'I = ',xmax,', J = ',ymax,', K = 1, ','ZONETYPE = Ordered'
 write(37,*) 'DATAPACKING=POINT'
 write(37,*) 'DT = (SINGLE SINGLE SINGLE SINGLE SINGLE SINGLE SINGLE SINGLE
```

```
SINGLE SINGLE)'

 do jc=1,ymax
  do ic=1,xmax

   ! This line is to write the filtered velocity & gradient for Guoning's group
!   write(38,*) xpos(ic), ypos(jc), xvel(ic,jc), yvel(ic,jc), &
!    -dpdx(ic,jc), -dpdy(ic,jc), mask(ic,jc)

   write(37,*) xpos(ic), ypos(jc), xvel(ic,jc),yvel(ic,jc),&
    press(ic,jc), gam(ic,jc), gam_p(ic,jc),&
    egval_vi(ic,jc), egval_p2(ic,jc), mask(ic,jc)
  end do
 end do
 close(37)
! close(38)
end do

close(5)

 call cpu_time(tempo2)
print*,'Total program time = ',tempo2-tempo1, " seconds."

contains
 function mag2(vec)
  real*8, dimension(2) :: vec
  real*8     :: mag2
  mag2 = sqrt(vec(1)**2 + vec(2)**2)
 end function


 subroutine centdiff4(array,stepx,stepy,mask,derivx,derivy)
  implicit none
  !! This subroutine approximates a derivative using a 4th order accurate
  ! central differencing scheme
  real*8, dimension(:,:), intent(in) :: array
  real*8, intent(in)   :: stepx, stepy
  integer, dimension(:,:), intent(in)  :: mask
  integer, parameter    :: N = size(array(:,1)), M = size(array(1,:))
  real*8, dimension(N,M), intent(out)  :: derivx, derivy

  integer icount, jcount

  do icount=1,N
   do jcount=1,M
    if(mask(icount,jcount)/=0) then
     if(icount<=2) then
      derivx(icount,jcount) = (array(icount+1,jcount)-array(icount,jcount))/stepx
     elseif(icount>=(N-3)) then
      derivx(icount,jcount)= (array(icount,jcount) - array(icount-1,jcount))/stepx
     else
      derivx(icount,jcount)=1/(12.0*stepx)*(array(icount-2,jcount)- &
       8.0*array(icount-1,jcount)+8.0*array(icount+1,jcount)-array(icount+2,jcount
```

```
))
     end if
    else
     derivx(icount,jcount) = 0
    endif

   end do
  end do

  do jcount=1,M
   do icount=1,N

    if(mask(icount,jcount)/=0) then
     if(jcount<=2) then
      derivy(icount,jcount) = (array(icount,jcount+1)-array(icount,jcount))/stepy
     elseif(jcount>=(M-3)) then
      derivy(icount,jcount)= (array(icount,jcount) - array(icount,jcount-1))/stepy
     else
      derivy(icount,jcount)=1/(12.0*stepy)*(array(icount,jcount-2)-&
       8.0*array(icount,jcount-1)+8.0*array(icount,jcount+1)-array(icount,jcount+2
))
     end if
    else
     derivy(icount,jcount) = 0
    endif

   end do
  end do


 end subroutine

 subroutine centdiff(array,step,idex,deriv)
  implicit none
  ! This subroutine approximates a first derivative using 2nd-order
  ! central differencing
  real*8, dimension(:), intent(in) :: array
  real*8, intent(in)     :: step
  integer,intent(in)     :: idex
  real*8, intent(out)     :: deriv

  if(step<=1) then
   deriv = (array(idex+1)-array(idex))/step
  else
   deriv = 1.0/(2.0*step)*(array(idex-1) - array(idex+1))
  end if

 end subroutine

 subroutine eigenval2(array,egval_r1,egval_r2,egval_i)
  implicit none
  ! This subroutine calculates the eigenvalues of a 2D array
  real*8, dimension(:,:), intent(in) :: array
```

```
 real*8, intent(out)     :: egval_r1, egval_r2, egval_i
 real*8        :: gamma_d, gamma_r, gamma_s

 gamma_d = (array(1,1) + array(2,2))/2.0_8
 gamma_r = (array(2,1) - array(1,2))/2.0_8
 gamma_s = sqrt((array(1,1)-array(2,2))**2 + (array(1,2)+array(2,1))**2)/2.0_8

 if (gamma_s**2 < gamma_r**2) then
  egval_r1 = gamma_d
  egval_r2 = gamma_d
  egval_i = sqrt(gamma_r**2-gamma_s**2)
 else
  egval_r1 = gamma_d + sqrt(gamma_s**2 - gamma_r**2)
  egval_r2 = gamma_d - sqrt(gamma_s**2 - gamma_r**2)
  egval_i = 0.0
 end if
end subroutine


subroutine gauss2(matrix,filtmatrix,filtsize,sd,mask,dx,dy)
implicit none
real*8,dimension(:,:), intent(in) :: matrix
integer, dimension(:,:),intent(in) :: mask
real*8,dimension(:,:), intent(out) :: filtmatrix
integer, intent(in)     :: sd, filtsize
real*8, intent(in)     :: dx, dy
integer, parameter   :: N=size(matrix(:,1)), M=size(matrix(1,:))
integer     :: ic, jc, col, row, ic2, jc2
real*8     :: tempnum, tempdenom, tempfunc
! The intent of this subroutine is to filter the input matrix (NxM) using
! a gaussian filter based on dx, dy, and the standard deviation.
! The kernel is basically an exponential that incorporates the values over
! a subregion to modify the input value.

!print*,matrix(20,20)

do ic=1,N
 do jc=1,M
  tempnum = 0.0_8
  tempdenom = 0.0_8
  tempfunc = 0.0_8

  if (mask(ic,jc) /= 0) then
   do ic2=(-filtsize),(filtsize)
    do jc2=(-filtsize),(filtsize)
     if ((ic+ic2<1) .or. (ic+ic2>N) .or. (jc+jc2<1) .or. (jc+jc2>M)) then
      continue
     else
      tempfunc=exp(-((ic2*dx)**2+(jc2*dy)**2)/(2*(sd*dx)**2))
      tempnum = tempnum + tempfunc*matrix(ic+ic2,jc+jc2)
      tempdenom = tempdenom + tempfunc
     end if
    end do
```

```
   end do
  end if

  if (tempdenom == 0.0_8) then
   filtmatrix(ic,jc)=0.0_8
  else
   filtmatrix(ic,jc) = matrix(ic,jc) - (tempnum/tempdenom)
  end if
 end do
end do


end subroutine

subroutine gammadet(xpos,ypos,xvel,yvel,mask,rad,gam,str)
real*8, dimension(:), intent(in) :: xpos, ypos
real*8, dimension(:,:), intent(in) :: xvel,yvel
integer, dimension(:,:), intent(in) :: mask
integer, intent(in) :: rad
real*8, dimension(:,:), intent(out) :: gam,str
real*8, dimension(2) :: um, pm
real*8  :: dummygam, dummystr
integer :: col, row, subx, suby, numpts

print*,'Starting the gamma detect loops'
do col=(rad+1),(xmax-(rad+1))
 do row=(rad+1),(ymax-(rad+1))
  dummygam = 0.0
  dummystr = 0.0
  numpts = 0
  do subx=(-rad),rad
   do suby=(-rad),rad
    if ((mask(col,row) /= 0) .and. (subx /= 0) .and. (suby /=0)) then
     ! assign values to position and velocity vectors about midpoint of subregion
     pm = [xpos(col+subx)-xpos(col),ypos(row+suby)-ypos(row)]
      um = [xvel(col+subx,row+suby),yvel(col+subx,row+suby)]


      !um = [xvel(col+subx,row+suby)-xvel(col,row), &
      ! yvel(col+subx,row+suby)-yvel(col,row)]

      if (mag2(um) /= 0) then
       ! calculate gamma and strength for each point, summing for each step
       dummygam = dummygam + (pm(1)*um(2)-um(1)*pm(2))/(mag2(pm)*mag2(um))
       dummystr = dummystr +
mag2(um)*((pm(1)*um(2)-um(1)*pm(2))/(mag2(pm)*mag2(um)))
       numpts = numpts + 1
      end if
    else
     continue
    end if
   end do
  end do
```

```
   if (numpts == 0) then
    gam(col,row) = 0
    str(col,row) = 0
   else
    gam(col,row) = dummygam/numpts
    str(col,row) = dummystr/numpts
   end if
  end do
 end do

 end subroutine

end program
```

Appendix B – Covariance calculation code

```
program correlation
implicit none

! The purpose of this program is to read grid indexed data for velocity
! pressure, etc, or the calcualated values of gamma, gamma_p, etc.
! Once the data is read in, a time-probe is performed at some user-defined
! points and output to a single file.

!! Parameters, variables to change to alter the correlation
integer, parameter  :: xmax=300, ymax=200, tmax=5 ! range of the data set input
integer, parameter  :: numsegments=4 ! The number of segments on the surface
integer, parameter  :: numbins=60  ! The number of regions of correlation
integer, parameter  :: pts_per_seg = 19 ! the number of subpoints to look at in
each segment
integer     :: total_surf_pts = numsegments*pts_per_seg
real*8, parameter  :: rho = 1.0, u_inf=0.3318 !density & free stream velocity
integer, dimension(numsegments,numbins) :: num_pressgam, num_gam, tnum_pressgam =
0, tnum_gam = 0
integer, dimension(numsegments) ::num_press, tnum_press = 0

!! The mean values stuff
integer, dimension(numsegments, numbins) :: numgambar = 0.0, numgamrms = 0.0,
numtgamrms = 0.0
integer, dimension(numsegments)    ::  numpressbar = 0, numpressrms = 0,
numtpressrms = 0, nummeancorrelation = 0
real*8, dimension(numsegments,numbins)  ::  gambar = 0.0, gamrms = 0.0, tgamrms =
0.0
real*8, dimension(numsegments)    :: pressbar = 0.0, pressrms = 0.0, tpressrms =
0.0, meancorrelation = 0.0
integer          :: tempnumgam = 0, tempnumpress=0, tempnumrms=0

!! Variables regarding the bins
integer, dimension(xmax, ymax, numsegments, pts_per_seg) :: bin_pts
real*8, dimension(numbins) :: bin_radii
real*8     :: bin_max = 1.5, temp_mag
integer     :: temp_bin, temp_num_gam, num_bin_pts=0
integer     :: loopcount = 0

!! Variables for segments
integer, dimension(numsegments,2) :: segment_pts
integer, dimension(numsegments, pts_per_seg, 2) :: seg_sub_pts
integer     :: seg_size_x, seg_size_y ! size of each surface segment
integer     :: pt_skip ! defines how many surface points to skip
integer, dimension(2,2) :: surface_pts_index
real*8, dimension(2,2) :: surface_pts
real*8, dimension(2) :: P_inf_pt = (/-0.49,0.0/)
integer, dimension(2) :: P_inf_index
 character*50   :: filelist = 'vortdetfiles5.dat', outfilelist =
'correlationfiles5.dat'
 character*50   :: outputfile = 'correlation.dat'

!!Variables that store the correlation, and sub-portions of the correlation
real*8 :: P_inf, P_si, P_diff
```

```
real*8 :: tempppressgam=0.0, temppress=0.0, tempgam=0.0, tempgam2=0.0, temppress2,
tempppressgam2
real*8 :: gamsum=0, presssum=0, pressgamsum=0
real*8  :: pts_gam=0, pts_press=0, pts_pressgam=0
real*8, dimension(numsegments,numbins) :: pressgamcorrelation = 1.0e-10,
tpressgamcorrelation = 1.0e-10
real*8, dimension(numsegments,numbins) ::  seg_pressgam= 1.0e-10, seg_gam =
1.0e-10
real*8, dimension(numsegments,numbins) :: tseg_pressgam = 1.0e-10, tseg_gam =
1.0e-10
real*8, dimension(numsegments)    :: seg_press = 1.0e-10, tseg_press = 1.0e-10

!!Variables for reading the data, counters, other misc
real*8, dimension(xmax) :: xpos  ! x and y position in millimeters
real*8, dimension(ymax) :: ypos
real*8, dimension(xmax,ymax) :: xvel, yvel, press, gam, str, egval_i, egval_p
real*8, dimension(xmax,ymax) :: dpdx, dpdy, gam_p, str_p
         ! ^arrays for the node position, velocity, gamma, and strength
real*8, dimension(xmax,ymax) :: gam_bar=0.0, press_bar=0.0
integer, dimension(xmax,ymax) :: mask
real*8     :: tempo1, tempo2 !time counters
real*8     :: delta_x, delta_y
integer     :: icount,jcount,kcount,tcount
integer, parameter  :: headlines = 15 !13 headlines for slice, 15 for detected
integer     :: subx, suby, subx2, suby2, y_tmp
real*8     :: test1, test2, test12, test13, test14, test15 ! dummy variables
real*8     :: test3, test4, test5, test6, test7, test8, test9, test10, test11
integer, dimension(2) :: point1, point2, point3, point4, point5 !probe points
 character*50   :: filename, junk, output_file, timefile='timehistory_gamma.dat'
real*8  :: maxgam=0.0, maxpress = 0.0, mingam=1.0, minpress=100.0,
maxnormpress=0.0, minnormpress=1.0
real*8  :: tempgambar = 0.0, tempppressbar = 0.0, tempgamrms=0.0, tempppressrms=0.0

!!Define the surface endpoints
surface_pts(1,1) = 0.0_8  !x1
surface_pts(1,2) = 0.5  !y1
surface_pts(2,1) = 1.0  !x2
surface_pts(2,2) = 0.5  !y2

pressbar(:) = 0.0
numpressbar(:) = 0
gambar(:,:) = 0.0
numgambar(:,:) = 0

!------------------------------------------------------------!
! The first order of business is to open the file and read
!  the relevant data.
!------------------------------------------------------------!

 call cpu_time(tempo1)

open(4,file=filelist)
```

```
print*,'Calculating the time average in each seg/bin'
do tcount=1,tmax

 maxgam = -1.0;
 mingam = 1.0;
 minpress = 100.0;
 maxpress = -100.0;
 maxnormpress=-10.0
 minnormpress=10.0
 pressgamcorrelation(:,:) = 0.0

 loopcount = 0

 read(4,*) filename

 print*,'reading data from: ',filename

 open(9, file=filename)
 kcount = 0
 do kcount=1,headlines
  read(9,*) junk
 end do

 do jcount=1,ymax  ! Populates the data arrays for doing the time probe
  do icount=1,xmax ! 'test' variables are just dummy variables that hold a single
value
   read(9,*)  test1, test2, test3, test4, test5, test6, test7, test8, test9,
test10

    ! These are variables output from the vortex detection
    xpos(icount) = test1
    ypos(jcount) = test2
    xvel(icount,jcount) = test3
    yvel(icount,jcount) = test4
    press(icount,jcount) = test5
    gam(icount,jcount) = abs(test6)
    gam_p(icount,jcount) = test7
    egval_i(icount,jcount) = test8
    egval_p(icount,jcount) = test9
    mask(icount,jcount) = int(test10)

    !Test for max gamma in the data set
  if(ypos(jcount) >= surface_pts(2,2)) then
    maxgam = max(maxgam, gam(icount,jcount))
    if(mask(icount,jcount)==1) mingam = min(mingam, gam(icount,jcount))

    maxpress = max(maxpress,press(icount,jcount))
    minpress = min(minpress,press(icount,jcount))

  end if

  end do
 end do
```

```
 close(9)

 if (tcount == 1) then
  !!Find the space steps and use them to find the indicies of relevant points
  delta_x = (xpos(xmax) - xpos(1))/(xmax+1)
  delta_y = (ypos(ymax) - ypos(1))/(ymax+1)

  !Upstream reference pressure point
  P_inf_index(1) = ceiling((P_inf_pt(1)-xpos(1))/delta_x)
  P_inf_index(2) = ceiling((P_inf_pt(2)-ypos(1))/delta_y)
  do while (mask(P_inf_index(1),P_inf_index(2))==0)
   P_inf_index(1) = P_inf_index(1)-1
  end do

  ! Define end points of surface to be tested
  surface_pts_index(:,1) = ceiling((surface_pts(:,1)-xpos(1))/delta_x)
  surface_pts_index(:,2) = ceiling((surface_pts(:,2)-ypos(1))/delta_y)
  pt_skip = (surface_pts_index(2,1)-surface_pts_index(1,1))/total_surf_pts

  ! Make sure we are outside the masked region.  Since dp/dx=0 at surfaces, this
is OK
  if (mask(surface_pts_index(1,1),surface_pts_index(1,2)) == 0 .or. &
   mask(surface_pts_index(2,1),surface_pts_index(2,2)) == 0) then
   surface_pts_index(:,2) = surface_pts_index(:,2) + 1
   print*,'Adjusting surface to be outside mask'
  end if

  ! define the segment size
  seg_size_x = (surface_pts_index(2,1) - surface_pts_index(1,1))/numsegments
  seg_size_y = (surface_pts_index(2,2) - surface_pts_index(1,2))/numsegments
  print*,'Segment size in x = ',seg_size_x,' and y = ',seg_size_y
  !if(seg_size_x<pts_per_seg) print*,'TOO MANY POINTS REQUESTED, INCORRECT
RESULTS IMMENENT!!!'

  !define the coordinates of the segments
  do icount=1,numsegments
   segment_pts(icount,1) = surface_pts_index(1,1) + seg_size_x*icount
   segment_pts(icount,2) = surface_pts_index(1,2) + seg_size_y*icount
!   print*,'Segment',icount,' coordinates: ',segment_pts(icount,:)
  end do

  !define the coordinates of the points within the segments
  do icount=1,numsegments
   do jcount=1,pts_per_seg
    seg_sub_pts(icount,jcount,1) = segment_pts(icount,1) -seg_size_x+ jcount
    seg_sub_pts(icount,jcount,2) = segment_pts(icount,2)
    print*,'Segment',icount,'subpoint',jcount,'coordinates:',seg_sub_pts(icount,jc
ount,:)
   end do
  end do

  do icount=1,numbins
```

```
   bin_radii(icount) = icount*(bin_max/numbins)
  end do
 end if


 P_inf = press(P_inf_index(1),P_inf_index(2))

 do icount = 1,numsegments !! Loop over the number of segments
  do jcount = 1,pts_per_seg !! Loop over the number of points in the segments

   temppress =
(press(seg_sub_pts(icount,jcount,1),seg_sub_pts(icount,jcount,2))-P_inf)/(0.5*rho*
u_inf**2)

   maxnormpress = max(maxnormpress,temppress)
   minnormpress = min(minnormpress,temppress)


   do subx = 1, xmax !! loop over the xpoints
    do suby = 1, ymax  !! loop over the ypoints

     if (tcount==1) then ! For the first time loop, define the bin for each
point/segment

      temp_mag = ((xpos(subx)-(seg_sub_pts(icount,jcount,1)*delta_x+xpos(1)))**2
+&
          (ypos(suby)-(seg_sub_pts(icount,jcount,2)*delta_y+ypos(1)))**2)**0.5

      do kcount=1,numbins

       if(temp_mag>=bin_radii(kcount) .and. temp_mag<=bin_radii(kcount+1)) then
        bin_pts(subx,suby,icount,jcount) = kcount
       end if

      end do

     end if

     !! need to check if the region is correct (above top surface only)
     temp_mag = ((xpos(subx)-(seg_sub_pts(icount,jcount,1)*delta_x+xpos(1)))**2 +&
          (ypos(suby)-(seg_sub_pts(icount,jcount,2)*delta_y+ypos(1)))**2)**0.5

     temp_bin = bin_pts(subx,suby,icount,jcount)
     tempgam = gam(subx,suby)

     if (mask(subx,suby) == 1 .and. ypos(suby) >= surface_pts(1,2) &
      .and. temp_mag<=bin_max) then

      if(tempgam>0.0 .or. tempgam < 0.0) then

       tempgambar = gambar(icount,temp_bin)
        gambar(icount,temp_bin) = tempgambar + tempgam
```

```
       tempnumgam = numgambar(icount,temp_bin)
       numgambar(icount,temp_bin) = tempnumgam + 1

      end if

      temppressbar = pressbar(icount)
      pressbar(icount) = temppressbar + temppress

      tempnumpress = numpressbar(icount)
      numpressbar(icount) = tempnumpress + 1

      end if
    end do
   end do


  end do
 end do

print*,'Testing output of maxpress', maxnormpress,'and min press', minnormpress
print*,"Testing running output of pressbar at each time step",
pressbar(1)/numpressbar(1)

end do

close(4)
! print*,"test",pressbar(:)
! print*,"test2",numpressbar(:)
do jcount = 1,numsegments
 temppressbar = pressbar(jcount)
 pressbar(jcount) = temppressbar / numpressbar(jcount)
 do kcount=1,numbins

  tempgambar = gambar(jcount,kcount)
  gambar(jcount,kcount) = tempgambar/numgambar(jcount,kcount)

 end do
end do

 print*,"Cp bar in Segments 1-4:", pressbar(:)
! print*,"gam bar in segments 1-4, bin 1:", gambar(:,1)

pressbar(1) = -1.59000
pressbar(2) = -1.5277
pressbar(3) = -1.3734
pressbar(4) = -1.4611

open(4,file=filelist)
print*,'Calculating the time rms in each seg/bin'
do tcount=1,tmax
 read(4,*) filename

 print*,'reading data from: ',filename
```

```
 open(9, file=filename)
 kcount = 0
 do kcount=1,headlines
  read(9,*) junk
 end do

 do jcount=1,ymax  ! Populates the data arrays for doing the time probe
  do icount=1,xmax ! 'test' variables are just dummy variables that hold a single
value
   read(9,*)  test1, test2, test3, test4, test5, test6, test7, test8, test9,
test10

    ! These are variables output from the vortex detection
    xpos(icount) = test1
    ypos(jcount) = test2
    xvel(icount,jcount) = test3
    yvel(icount,jcount) = test4
    press(icount,jcount) = test5
    gam(icount,jcount) = abs(test6)
    gam_p(icount,jcount) = test7
    egval_i(icount,jcount) = test8
    egval_p(icount,jcount) = test9
    mask(icount,jcount) = int(test10)

  end do
 end do

 close(9)

 P_inf = press(P_inf_index(1),P_inf_index(2))

 do icount = 1,numsegments !! Loop over the number of segments
  do jcount = 1,pts_per_seg !! Loop over the number of points in the segments

   temppress2 =
(press(seg_sub_pts(icount,jcount,1),seg_sub_pts(icount,jcount,2))-P_inf)/(0.5*rho*
u_inf**2)
   temppress = (temppress2 - pressbar(icount))**2

   maxnormpress = max(maxnormpress,temppress)
   minnormpress = min(minnormpress,temppress)


   do subx = 1, xmax !! loop over the xpoints
    do suby = 1, ymax  !! loop over the ypoints

     !! need to check if the region is correct (above top surface only)
     temp_mag = ((xpos(subx)-(seg_sub_pts(icount,jcount,1)*delta_x+xpos(1)))**2 +&
          (ypos(suby)-(seg_sub_pts(icount,jcount,2)*delta_y+ypos(1)))**2)**0.5

     temp_bin = bin_pts(subx,suby,icount,jcount)
```

```
      tempgam = (gam(subx,suby) - gambar(icount,temp_bin))**2

      if (mask(subx,suby) == 1 .and. ypos(suby) >= surface_pts(1,2) &
       .and. temp_mag<=bin_max) then

       if(tempgam>0.0 .or. tempgam < 0.0) then

        tempgamrms = gamrms(icount,temp_bin)
         gamrms(icount,temp_bin) = tempgamrms + tempgam

        tempnumgam = numgamrms(icount,temp_bin)
        numgamrms(icount,temp_bin) = tempnumgam + 1

       end if

       tempppressrms = pressrms(icount)
       pressrms(icount) = tempppressrms + tempppress

       tempnumpress = numpressrms(icount)
       numpressrms(icount) = tempnumpress + 1

       end if
     end do
    end do


   end do
  end do


end do

close(4)

do jcount = 1,numsegments
 tempppress = pressrms(jcount)
 pressrms(jcount) = sqrt(tempppress / numpressrms(jcount))
 do kcount=1,numbins

   tempgam = gamrms(jcount,kcount)
   gamrms(jcount,kcount) = sqrt(tempgam/numgamrms(jcount,kcount))

 end do
end do

!  print*,"Cp bar in Segments 1-4:", pressbar(:)
!
!  print*,"Cp rms in Segments 1-4:", pressrms(:)
! print*,"gam rms in segments 1-4, bin 1:", gamrms(:,1)

open(4,file=filelist)

open(99,file=outfilelist)
```

```
 open(51,file="seg4gammatime.dat")
 open(52,file="seg1correlationtime.dat")
 open(53,file="seg3correlationtime.dat")
 open(54,file="seg4correlationtime.dat")
 open(55,file="segpresstime.dat")

open(5,file='timehistory.dat')
 write(5,*) '"TIME"'
 write(5,*) '"LAMBDA_1"'
 write(5,*) '"LAMBDA_2"'
 write(5,*) '"LAMBDA_3"'
 write(5,*) '"LAMBDAP_1"'
 write(5,*) '"LAMBDAP_2"'
 write(5,*) '"LAMBDAP_3"'

print*,'Calculating the temporal covariance in each seg/bin'
do tcount=1,tmax

 num_pressgam(:,:) = 1
 num_press(:) = 1
 num_gam(:,:) = 1

 temppress = 0.0
 tempppressgam = 0.0
 tempppressgam2 = 0.0
 tempgam = 0.0
 tempgam2 = 0.0

 maxgam = -1.0;
 mingam = 1.0;
 minpress = 100.0;
 maxpress = -100.0;
 maxnormpress=0.0
 minnormpress=10.0
 pressgamcorrelation(:,:) = 0.0
 seg_pressgam(:,:) = 0.0
 seg_press(:) = 0.0
 seg_gam(:,:) = 0.0

 loopcount = 0

 read(4,*) filename

 print*,'reading data from: ',filename

 open(9, file=filename)
 kcount = 0
 do kcount=1,headlines
  read(9,*) junk
 end do

 do jcount=1,ymax  ! Populates the data arrays for doing the time probe
```

```
  do icount=1,xmax ! 'test' variables are just dummy variables that hold a single
value
   read(9,*)  test1, test2, test3, test4, test5, test6, test7, test8, test9,
test10

    ! These are variables output from the vortex detection
    xpos(icount) = test1
    ypos(jcount) = test2
    xvel(icount,jcount) = test3
    yvel(icount,jcount) = test4
    press(icount,jcount) = test5
    gam(icount,jcount) = abs(test6)
    gam_p(icount,jcount) = test7
    egval_i(icount,jcount) = test8
    egval_p(icount,jcount) = test9
    mask(icount,jcount) = int(test10)

    !Test for max gamma in the data set
  if(ypos(jcount) >= surface_pts(2,2)) then
    maxgam = max(maxgam, abs(gam(icount,jcount)))
    if(mask(icount,jcount)==1) mingam = min(mingam, abs(gam(icount,jcount)))

    maxpress = max(maxpress,press(icount,jcount))
    minpress = min(minpress,press(icount,jcount))

  end if


  end do
 end do

 close(9)

 P_inf = press(P_inf_index(1),P_inf_index(2))
 !print*,'Upstream (fixed) pressure = ',P_inf


 do icount = 1,numsegments !! Loop over the number of segments
  do jcount = 1,pts_per_seg !! Loop over the number of points in the segments
temppress2 =
(press(seg_sub_pts(icount,jcount,1),seg_sub_pts(icount,jcount,2))-P_inf)/(0.5*rho*
u_inf**2)
temppress = temppress2 - pressbar(icount)

    maxnormpress = max(maxnormpress,temppress)
    minnormpress = min(minnormpress,temppress)

  seg_press(icount) = seg_press(icount) + temppress
  num_press(icount) = num_press(icount) + 1

   do subx = 1, xmax !! loop over the xpoints
    do suby = 1, ymax  !! loop over the ypoints
```

```
     if (tcount==1) then ! For the first time loop, define the bin for each
point/segment

      temp_mag = ((xpos(subx)-(seg_sub_pts(icount,jcount,1)*delta_x+xpos(1)))**2
+&
          (ypos(suby)-(seg_sub_pts(icount,jcount,2)*delta_y+ypos(1)))**2)**0.5

      do kcount=1,numbins

       if(temp_mag>=bin_radii(kcount) .and. temp_mag<=bin_radii(kcount+1)) then
        bin_pts(subx,suby,icount,jcount) = kcount
       end if

      end do

     end if

     !! For all steps need to calculate the correlation value
     !! need to check if the region is correct (above top surface only)
     temp_mag = ((xpos(subx)-(seg_sub_pts(icount,jcount,1)*delta_x+xpos(1)))**2 +&
          (ypos(suby)-(seg_sub_pts(icount,jcount,2)*delta_y+ypos(1)))**2)**0.5

     temp_bin = bin_pts(subx,suby,icount,jcount)
     tempgam = gam(subx,suby) - gambar(icount,temp_bin)

   if (mask(subx,suby) == 1 .and. ypos(suby) >= surface_pts(1,2) &
      .and. temp_mag<=bin_max) then

      if(tempgam>0.0 .or. tempgam < 0.0) then
       temppressgam = tempgam*temppress

        seg_gam(icount,temp_bin) = seg_gam(icount,temp_bin) + tempgam
 !       num_gam(icount,temp_bin) = num_gam(icount,temp_bin) + 1
      ! else
     !  temppressgam = 0.0
!       tempgamrms = gamrms(icount,temp_bin)
!       gamrms(icount,temp_bin) = tempgamrms + tempgam**2
!       numgamrms(icount,temp_bin) = numgamrms(icount,temp_bin) + 1
!
!       temppressrms = pressrms(icount)
!       pressrms(icount) = temppressrms + temppress**2
!       numpressrms(icount) = numpressrms(icount) + 1

      temppressgam2 = seg_pressgam(icount,temp_bin)

       seg_pressgam(icount,temp_bin) = temppressgam2 + temppressgam
       num_pressgam(icount,temp_bin) = num_pressgam(icount,temp_bin) + 1
      end if
!
      end if
    end do
   end do
```

```
  end do
 end do

 print*,"Maximum gamma magnitude in data: ", maxgam
 print*,"Minimum gamma magnitude in data: ", mingam

 print*,"Max pressure difference in field: ", maxnormpress
 print*,"Min pressure difference in field: ", minnormpress

 write(5,*) tcount, gam(50,127), gam(75,136), gam(100,145), gam_p(50,127),
gam_p(75,136), gam_p(100,145)

 ! Write the header
 read(99,*) outputfile
 open(3,file=outputfile)

 write(3,*) 'TITLE = "correlation',tcount,'"'
 write(3,*) 'VARIABLES = "bin" '
 do icount=1,numsegments
  write(3,*) ' "Segment ',icount, ' " '
 end do
 do icount=1,numsegments
  write(3,*) ' "gam_seg ',icount, ' " '
 end do
 write(3,*) 'TIMECOUNT'
 write(3,*) 'ZONE T = "ZONE',tcount,'"'
 write(3,*) 'STRANDID=',tcount,', SOLUTIONTIME=',tcount
 write(3,*) 'I=,',numbins-1,' J=1, K=1, ZONETYPE=Ordered'
 write(3,*) 'DATAPACKING=POINT'
 write(3,*) 'DT=(SINGLE SINGLE SINGLE SINGLE SINGLE SINGLE SINGLE SINGLE SINGLE
SINGLE)'

 !make the final correlation calculations at each time step
 do jcount = 1,numbins-1
  do kcount=1,numsegments

   pressgamcorrelation(kcount,jcount) =
seg_pressgam(kcount,jcount)/num_pressgam(kcount,jcount) / &
    (pressrms(kcount)*gamrms(kcount,jcount))
  if(pressgamcorrelation(kcount,jcount) > 1.0) pressgamcorrelation(kcount,jcount)
= 1.0
  end do
  write(3,*) bin_radii(jcount), pressgamcorrelation(:,jcount),
(seg_gam(:,jcount)/num_gam(:,jcount)), tcount
 end do
 close(3)

!  write(50,*) tcount, seg_press(1)/num_press(1), seg_press(2)/num_press(2), &
!       seg_press(3)/num_press(3), seg_press(4)/num_press(4)
!
write(51,*) tcount, &
  seg_gam(4,1) /num_pressgam(4,1),  seg_gam(4,5) /num_pressgam(4,5), &
```

```
  seg_gam(4,10)/num_pressgam(4,10), seg_gam(4,15)/num_pressgam(4,15), &
  seg_gam(4,20)/num_pressgam(4,20), seg_gam(4,25)/num_pressgam(4,25), &
  seg_gam(4,30)/num_pressgam(4,30), seg_gam(4,35)/num_pressgam(4,35), &
  seg_gam(4,40)/num_pressgam(4,40)
! write(51,*) tcount, seg_pressgam(3,1)/num_pressgam(3,1),
seg_pressgam(3,2)/num_pressgam(3,2), &
!   seg_pressgam(3,3)/num_pressgam(3,3), seg_pressgam(3,4)/num_pressgam(3,4), &
!   seg_pressgam(3,5)/num_pressgam(3,5), seg_pressgam(3,6)/num_pressgam(3,6), &
!   seg_pressgam(3,7)/num_pressgam(3,7), seg_pressgam(3,8)/num_pressgam(3,8), &
!   seg_pressgam(3,9)/num_pressgam(3,9)

do kcount=1,numsegments
 do jcount = 1,numbins
  meancorrelation(kcount) = pressgamcorrelation(kcount,jcount) +
meancorrelation(kcount)
  nummeancorrelation(kcount) = 1 + nummeancorrelation(kcount)
 end do
 meancorrelation(kcount) = meancorrelation(kcount) / nummeancorrelation(kcount)
end do

! write(51,*) tcount, pressgamcorrelation(1,1), pressgamcorrelation(1,5), &
!   pressgamcorrelation(1,10), pressgamcorrelation(1,15), &
!   pressgamcorrelation(1,20), pressgamcorrelation(1,25), &
!   pressgamcorrelation(1,30), pressgamcorrelation(1,35), &
!   pressgamcorrelation(1,40)

write(52,*) tcount, pressgamcorrelation(1,1), pressgamcorrelation(1,5), &
  pressgamcorrelation(1,10), pressgamcorrelation(1,15), &
  pressgamcorrelation(1,20), pressgamcorrelation(1,25), &
  pressgamcorrelation(1,30), pressgamcorrelation(1,35), &
  pressgamcorrelation(1,40)

write(53,*) tcount, pressgamcorrelation(3,1), pressgamcorrelation(3,5), &
  pressgamcorrelation(3,10), pressgamcorrelation(3,15), &
  pressgamcorrelation(3,20), pressgamcorrelation(3,25), &
  pressgamcorrelation(3,30), pressgamcorrelation(3,35), &
  pressgamcorrelation(3,40)

write(54,*) tcount, pressgamcorrelation(4,1), pressgamcorrelation(4,5), &
  pressgamcorrelation(4,10), pressgamcorrelation(4,15), &
  pressgamcorrelation(4,20), pressgamcorrelation(4,25), &
  pressgamcorrelation(4,30), pressgamcorrelation(4,35), &
  pressgamcorrelation(4,40)

test1 = (seg_press(1)/num_press(1))
test2 = (seg_press(2)/num_press(2))
test3 = (seg_press(3)/num_press(3))
test4 = (seg_press(4)/num_press(4))
print*,'Test seg press numbers:', num_press(1), num_press(2), num_press(3),
num_press(4)
print*,'Test seg press summation:', seg_press(1), seg_press(2), seg_press(3),
seg_press(4)
print*,'Test seg press output:',test1, test2, test3, test4
```

```
write(55,*) tcount, test1, test2, test3, test4
! write(55,*) tcount, (seg_press(1))/num_press(1), (seg_press(2))/num_press(2), &
!      seg_press(3)/num_press(3), seg_press(4)/num_press(4)


 do kcount = 1,numsegments

  do jcount=1,numbins
   !Add values to the time-averaged, too
   tseg_pressgam(kcount,jcount) = abs(seg_pressgam(kcount,jcount)) +
tseg_pressgam(kcount,jcount)
   tnum_pressgam(kcount,jcount) = num_pressgam(kcount,jcount) +
tnum_pressgam(kcount,jcount)

  end do
 end do

end do

 close(5)
 close(51)
 close(52)
 close(53)
 close(54)
 close(55)
!  close(50)
 close(99)
! close(42)
 close(4)
!
! ! ! Write the header
open(5,file='tcorrelation.dat')
write(5,*) 'TITLE = "Correlation time avg"'
write(5,*) 'VARIABLES = "bin" '
do icount=1,numsegments
 write(5,*) ' "Segment ',icount, ' " '
end do
write(5,*) 'ZONE T = "ZONE',tcount,'"'
write(5,*) 'STRANDID=0, SOLUTIONTIME=',tcount
write(5,*) 'I= ',numbins-1,', J=1, K=1, ZONETYPE=Ordered'
write(5,*) 'DATAPACKING=POINT'
write(5,*) 'DT=(SINGLE SINGLE SINGLE SINGLE SINGLE )'

!make the final correlation calculations
print*,'Calculating the time averaged covariance in each seg/bin'
do jcount = 1,numbins-1
 do kcount=1,numsegments
  if(tnum_pressgam(kcount,jcount) > 1 ) then

  tpressgamcorrelation(kcount,jcount) =
(tseg_pressgam(kcount,jcount)/tnum_pressgam(kcount,jcount)) / &
   (pressrms(kcount)*gamrms(kcount,jcount))
  end if
```

```
  !print*,'bin ',jcount,'segment ',kcount,'correlation:
',pressgamcorrelation(kcount,jcount)
 end do
 write(5,*) bin_radii(jcount), tpressgamcorrelation(:,jcount)
end do
 close(5)

 call cpu_time(tempo2)
print*,'Total program time = ',tempo2-tempo1, " seconds."

end program
```

Bibliography

R.J. Adrian, K.T. Christensen, and Z.-C. Liu. Analysis and interpretation of instantaneous turbulent velocity fields. *Experiments in Fluids*, 29(3):275–290, 1990.

S.V. Apte, K. Mahesh, P. Moin, and J.C. Oefelin. Large-eddy simulation of swirling particle-laden flows in a coaxial-jet combustor. *Int. J. of Multiphase Flow*, 29: 1311–1331, 2003.

S. Camarri, M.V. Salvetti, B. Koobus, and A. Dervieux. Large eddy simulation of a bluff-body flow on unstructured grids. *Int. J. for Numerical Methods in Fluids*, 40:1431–1460, 2002.

Guoning Chen, Zhongzang Lin, Daniel Morse, Stephen Snider, Sourabh Apte, James Liburdy, and Eugene Zhang. Multiscale feature detection in unsteady separated flows. *Accepted for publication in Int. J. of Numerical Analysis and Modeling*, 2008.

Minter Cheng and B.K. Chen. A numerical study on fluid force reduction of a square cylinder by flow control. *Proceedings of Joint ASME/JSME Fluids Engineering Conference, July 30-Aug. 2, 2007, San Diego, CA*, FEDSM2007, 2007. FEDSM2007-37025.

Haecheon Choi, Jeon Woo-Pyung, and Jinsung Kim. Control of flow over a bluff body. *Annu. Rev. Fluid Mech.*, 40:113–139, 2008.

M.S. Chong, A.E. Perry, and B.J. Cantwell. A general classification of three-dimensional flow fields. *Phys. Fluids A*, 2(5):765–777, 1990.

G.S. Constantinescu and K.D. Squires. LES and DES investigations of turbulent flow over a sphere at Re = 10,000. *Flow, Turb. and Comb.*, 70:267–298, 2003.

D.F.G. Durão, M.V. Heitor, and J.C.F. Pereira. Measurements of turbulent and periodic flows around a square cross-section cylinder. *Experiments in Fluids*, 6: 298–304, 1988.

Robert D. Falgout and Ulrike Meier Yang. *hypre*: a library of high performance preconditioners. *Lecture notes in computer science*, 2331:632–641, 2002. Proceedings of the International Conference on Computational Science.

J.H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer-Verlag, third edition, 2002.

Massimo Germano, Ugo Piomelli, Parviz Moin, and William H. Cabot. A dynamic subgrid-scale eddy viscosity model. *Phys. Fluids A*, 3(7):1760–1765, 1991.

L. Graftieaux, M. Michard, and N. Grosjean. Combining PIV, POD, and vortex identification algorithms for the study of unsteady turbulent swirling flows. *Meas. Sci. and Tech.*, 12:1422–1429, 2001.

F. Ham and G. Iaccarino. Energy conservation in collocated discretization schemes on unstructured meshes. *CTR Annual Research Briefs 2004*, pages 3–14, 2004.

Jinhee Jeong and Fazle Hussain. On the identification of a vortex. *J. Fluid Mech.*, 285:69–94, 1995.

John Kim, Parviz Moin, and Robert Moser. Turbulence statistics in fully developed channel flow at low Reynolds number. *J. Fluid Mech.*, 177:133–166, 1987.

Bruno Koobus and Charbel Farhat. A variational multiscale method for the large eddy simulation of compressible turbulent flows on unstructured meshes – application to vortex shedding. *Comp. Meth. Appl. Mech. Engr.*, 193:1367–1383, 2004.

D.A. Lyn, S. Einav, W. Rodi, and J.-H. Park. A laser-Doppler velocimetry study of ensemble-averaged characteristics of the turbulent near wake of a square cylinder. *J. Fluid Mech.*, 304:285–319, 1995.

K. Mahesh, G. Constantinescu, and P. Moin. A numerical method for large-eddy simulation in complex geometries. *J. Comp. Phys.*, 197:215–240, 2004.

K. Mahesh, G. Gonstantinescu, S. Apte, G. Iaccarino, F. Ham, and P. Moin. Large-eddy simulation of reacting turbulent flows in complex geometries. *J. Applied Mechanics*, 73(3):374–381, 2006. Proceedings of the AIAA.

U. Maucher, U. Rist, M. Kloker, and S. Wagner. DNS of laminar-turbulent transition in separation bubbles. In E. Krause and W. Jäger, editors, *High performance*

computing in science and engineering '99, pages 279–294. Springer New York, 1999.

P. Moin, K. Squires, W. Cabot, and S. Lee. A dynamic subgrid-scale model for compressible turbulence and scalar transport. *Phys. Fluids A*, 3(11):2746–2757, 1991.

Daniel R. Morse and James A. Liburdy. Dynamic characteristics of flow separation from a low Reynolds number airfoil. *Proceedings of Joint ASME/JSME Fluids Engineering Conference, July 30-Aug. 2, 2007, San Diego, CA*, FEDSM2007, 2007. FEDSM2007-37083.

Victor Ovchinnikov, Ugo Piomelli, and Meelan M. Choudhari. Numerical simulations of boundary-layer transition induced by a cylinder wake. *J. Fluid Mech.*, 547:413–441, 2006.

W. Rodi, J.H. Ferziger, M. Breuer, and M. Pourquié. Status of large eddy simulation: results of a workshop. *Transactions of the ASME*, 119:248–262, June 1997.

Roger L. Simpson, Y.-T. Chew, and B.G. Shivaprasad. The structure of a separating turbulent boundary layer. part 1. mean flow and Reynolds stresses. *J. Fluid Mech.*, 113:23–51, 1981.

Stephen Snider, Daniel Morse, Sourabh Apte, and James Liburdy. Correlation of surface pressure and vortical flow structures in an unsteady separating flow. *Proceedings of the AIAA*, June 2008.

Ahmad Sohankar, L. Davidson, and C. Norberg. Large eddy simulation of flow past a square cylinder; Comparison of different subgrid scale models. *J. Fluids Eng.*, 122:39–47, 2000.

Mark Thompson, Kerry Hourigan, and John Sheridan. Three-dimensional instabilities in the wake of a circular cylinder. *Exp. Thermal and Fluid Science*, 12: 190–196, 1996.

Andrei Travin, Michael Shur, Michael Strelets, and Philippe Spalart. Detached-eddy simulations pas a circular cylinder. *Flow, Turbulence and Combustion*, 63: 293–313, 1999.

L.L. van Dommelen and S.F. Shen. The spontaneous generation of the singularity in a separating laminar boundary layer. *J. Comp. Phys.*, 38:125–140, 1980.

Jan G. Wissink. DNS of 2D turbulent flow around a square cylinder. *Int. J. for Numerical Methods in Fluids*, 25:51–62, 1997.