Detailed Characterization of Turbulent Separated Flow Dynamics and Boundary Layer Evolution Over a Speed-Bump Geometry

Hariprasad Annamalai

A thesis
submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

University of Washington

2022

Committee:

Owen J. H. Williams

Antonino Ferrante

Program Authorized to Offer Degree:
William E. Boeing Department of Aeronautics and Astronautics

University of Washington

Abstract

Detailed Characterization of Turbulent Separated Flow Dynamics and Boundary Layer Evolution Over a Speed-Bump Geometry

Hariprasad Annamalai

Chair of the Supervisory Committee:
Owen J. H. Williams
Department of Aeronautics & Astronautics

The separated flow over a speed-bump was studied using planar particle image velocimetry (PIV) and high frequency surface pressures to investigate the separation and reattachment dynamics. The source of low frequency unsteadiness has yet to be determined and is of great interest for aerospace structures. The upstream boundary layer dynamics were dominated by the pressure gradient when in the presence of changing surface curvature. Conditional averaging of PIV fields based on reverse flow fraction yielded the mean separation and reattachment locations. For $Re_L = 2.56 * 10^6$, the mean separation point $x/L = 0.105$ varied between $0.089 < x/L < 0.120$ ($\pm 2\sigma$) and the mean reattachment point $x/L = 0.352$ varied between $0.300 < x/L < 0.403$ ($\pm 2\sigma$), yielding a mean separation bubble length $L_{sep} = 0.226m$. Velocity perturbations in the upstream boundary layer were correlated to reverse flow fractions, indicating a relationship with the separation location. Proper orthogonal decomposition of the velocity fluctuations indicated the largest contribution to total energy and turbulent shear stresses came from the low frequency motion of the separation bubble, while other modes showed convecting structures and the shear layer. Surface pressures revealed a spectral peak downstream of the apex at $St = f * L_{sep}/U_\infty \approx 0.05$ and peaks downstream of separation at $St \approx 0.62$, consistent with the presence of low-frequency bubble breathing and shear layer vortex shedding, respectively.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

U: Streamwise velocity

W: Wall-normal velocity

U_∞: Freestream velocity

u: Streamwise velocity fluctuation

w: Wall-normal velocity fluctuation

x, y, z: Streamwise, spanwise and wall-normal directions

Re: Reynolds Number

$-\overline{uw}$: Reynolds shear stress

u^2, w^2: Reynolds normal stresses

$u_{\tau o}$: Upstream friction velocity

α: Reverse flow fraction

C_p: Pressure coefficient

C_f: Skin friction coefficient

δ_ω: Boundary layer thickness

L: Bump width

R: Radius of curvature

St: Strouhal Number

L_{sep}: Separation bubble length

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Dr. Owen Williams for all his support and advice over the last two years, both in and outside the laboratory. His encouragement and patience in what has been the most demanding educational endeavor of my academic career has been so crucial to this project's success and my growth as a researcher.

I am grateful for the support of Matthew Robbins, Kevin Manohar and our collaborators at the University of Calgary, who I worked with closely on this project. I would also like to thank those in my lab, especially Elliot Jennis and Lauren Jones, for their support and the friendship that we built over the last two years. The experiences we have shared over countless cups of coffee, through doing homework together and working on our theses, which would actually just end up as hours of laughter, are moments I will forever cherish.

I am thankful for the Aeronautics and Astronautics department at the University of Washington for financially supporting my graduate studies.

I also owe a debt of gratitude to the Aerospace and Mechanical Engineering departments at the University of Michigan - Ann Arbor, for molding my foundation in engineering and exposing me to the world of innovation. I especially would like to thank my mentors and colleagues from my time at Rolls-Royce, who motivated me to pursue a graduate-level degree.

Last, but certainly not the least, I would like to thank my family and friends for their unconditional love. To my family: you have been with me every step of the way and continuously support my aspirations to become a scientist and an engineer. To my friends: I am incredibly lucky to have you all in my life and blessed to know each and every one of you.

DEDICATION

To Amma, Appa, and Geethu, who push me to be my best self every day.

Chapter 1
INTRODUCTION

Since the development of computational fluid dynamics (CFD) in the 1960s, the tool has enabled the design and analyses of complex structures interacting with a variety of fluid phenomena across many industries. CFD allows engineers to extract fundamental parameters governing the flow of interest such as velocity, pressure and density but also calculated quantities such as skin friction, Reynolds stresses, and pressure coefficients. However, CFD results must be validated against experimental data because at Reynolds numbers relevant to engineering applications, some aspects of turbulence must be modelled. Under these restricted circumstances, very few analytical solutions to full compressible Navier-Stokes equations exist.

Direct Numerical Simulation (DNS) solves the Navier Stoke equations numerically but this technique is computationally expensive to resolve the flow at the smallest Kolmogorov scales at high Reynolds numbers, which is the regime for most aerospace applications. Moreover, in the field of turbulence, the broadband length and time scales make solving the Navier Stokes equations through DNS very impractical. To mitigate this, a number of CFD solvers make various assumptions to close the system to model and resolve flow structures, one being Reynolds Averaged Navier Stokes (RANS) model. It is more widely used in industry to describe turbulent flows through Reynolds decomposition by describing a flow with its time-averaged and fluctuating quantities. Its time-averaged nature makes turbulent separated flows and boundary layer separation difficult to predict and the unsteadiness is not captured. The unsteady RANS (uRANS) model can capture time dependent phenomena but is computationally expensive and suffers from similar issues as RANS. In aeronautics and internal flows, boundary layer separation can cause airfoil stall and compressor surge,

which are undesirable. Therefore, it is important to understand the dynamics of turbulent boundary layer separation to recognize where models perform poorly and improve the design of aerodynamic bodies.

In 2014, NASA published a report titled *CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences* describing the steps necessary to significantly improve CFD capability. One section specifically called for more physics-based accurate predictions of complex turbulent flows including boundary layer separation to enable improved design and analysis [15]. There has been substantial experimental work done in this regime to further our understanding of turbulent boundary layer separation dynamics but it has also resulted in more questions. For example, several separated flow experiments have identified the presence of frequency dependent structures across differing geometries such as a flat plate [16], a forward-facing step [17], [18] and a cylinder elongated by another cylinder of smaller diameter [19]. In particular, there is a low frequency motion associated with the separation bubble breathing but it is unknown what the source of this motion is. Potential sources are thought to be upstream low frequency large scale motions ([17],[18]) and elongated streamwise structures in the separated regions linked to a Gôrter instability [16]. In addition to these studies with canonical geometries, pressure-induced boundary layer separation has also been observed to show low frequency dependent structures [20], whose causes are also yet to be determined. This is difficult to model and predict since most simulations are steady and is of great interest as low frequency motion can lead to large loading oscillations and structural fatigue [21].

A novel test geometry, the Gaussian speed-bump, was designed by Phillipe Spalart and Jeff Slotnick, and further developed at the University of Washington, to investigate low-speed incompressible turbulent boundary layer separation over varying surface curvature and pressure gradients. With a thin incoming boundary layer, the experimental setup is relevant to high lift configurations such an aircraft wing. Recent investigations into the smooth-body separation over the Gaussian speed-bump revealed a large three-dimensional separated region on the leeward side of the bump with two counter-rotating vortices ([4],

[11]) and that RANS did not capture the separated region well [12]. The flow field was found to be independent of Reynolds number for $Re_L > 2.4 * 10^6$ and the separation bubble exhibited a spanwise dependency. The separated region was also found to be larger than that predicted by SARC and SST RANS [4]. This sets the stage for some of these analyses focusing on the dynamics of the separation bubble, utilizing reverse flow fraction calculated from PIV, correlation methods, and the high frequency pressure spectra.

The four overarching goals in this research were to (1) understand the relative strength of pressure gradients and curvature on non-equilibrium boundary layer dynamics, (2) examine the dynamics of the separation region or bubble, (3) examine the frequency content of the separation bubble and evaluate if low frequency unsteadiness exists in this flow and (4) determine if unsteadiness of the separation bubble is correlated with upstream disturbances. The first goal was accomplished by calculating the boundary layer properties over the bump to understand the effects of pressure gradients and surface curvatures. The second goal was met with conditional averaging based on negative streamwise flow fraction and high frequency pressure data to produce probability distribution functions of separation and reattachment point locations.

The presence of low frequency unsteadiness and identification of flow structures was conducted with high frequency pressure acquisition across the bump surface and spectra analysis. This was also supported with proper orthogonal decomposition of the PIV data focused on the separated region to identify flow structures. Lastly, to identify potential sources of the unsteadiness, upstream velocity disturbances were correlated to streamwise negative flow fraction, which is directly linked to a specific separation location.

Chapter 2
BACKGROUND & THEORY

2.1 Challenges of Turbulent Separated Flows

In separated flows over smooth-bodies, separation begins sporadically at a given location meaning that flow reversal at that location occurs only a fraction of the total time. At successive downstream locations, the fraction of the time that flow moves downstream is progressively less. Simpson proposed a set of quantitative definitions on the detachment state near the wall based on the fraction of time that flow velocities point downstream γ [22]. Over a certain timeframe, incipient detachment occurs with instantaneous backflow 1% of the time ($\gamma = 0.99$); intermittent transitory detachment occurs with instantaneous backflow 20% of the time ($\gamma = 0.80$); transitory detachment occurs with instantaneous backflow 50% of the time ($\gamma = 0.50$); and the flow is fully separated when the time-averaged wall shear stress $\tau_w = 0$. These time-averaged definitions allow for the classification of the mean separation points. However, the instantaneous separation point itself has been observed to move upstream and downstream of the mean separation location ([17], [23]). Malm et.al. found that turbulent boundary layers can be instantaneously separated but not separated in the mean [24], meaning the time-averaged analyses may not shed light on the full dynamics. This unsteadiness suggests there is a distribution of separation points characteristic to the geometry and flow field conditions, which has not yet been determined for the speed-bump.

Turbulent boundary separation also leads to higher levels of surface pressure fluctuations and acoustic noise due to the varying velocity field within the attached boundary layer. For detached flows, it is known the largest pressure fluctuations occur in the middle of the shear layer, which strongly influence the near-wall flow [22], especially the separation and reattachment point motion. It has been observed that the surface pressure fluctuations

increase at detachment to a maximum just downstream of reattachment and they are large, almost on the same order of the local mean-pressure difference. The maximum fluctuation occurring near reattachment appears to be influenced by a variety of velocity variations of the fluid impinging on the wall, such as the passage of large-scale structures and unsteadiness associated with the reattachment point fluctuating upstream and downstream. As a result, this is why many separation studies employ high frequency pressure measurements to better capture the pressure fluctuations associated with separation, reattachment and unsteadiness ([18], [20], [25]).

Turbulent boundary layer separation can be induced by a variety of factors. In geometry induced boundary layer separation, major detachment occurs near the sharp edges of the body, such as corners. Downstream of separation, the flow is heavily dependent on the upstream velocity distribution and geometry. The backward-facing step produces a complex separated flowfield, consisting of a free-shear layer that sharply curves downward and impinges on the wall, reattaching to the surface. Beneath the shear layer and upstream of the reattachment point is a region of recirculating flow with large backflow. The flow can be characterized as highly unsteady with large turbulent structures with length scales as large as the step height passing through the reattachment region. It was also found the reattachment point moved up and downstream, by as many as $\pm$ 2 step heights. These fluctuations cause the low frequency flapping motion that correlates with strong flow reversals in the vicinity of reattachment. Similarly, the forward-facing step causes a separated flowfield detaching at two places, upstream of the step face due to the adverse pressure gradient and at the corner of the step, resulting in two re-circulation zones. The flow also exhibits a low-frequency flapping motion due to the shear layer near separation and reattachment. Differing from separation controlled by geometry, pressure and curvature induced separation do not occur at a fixed point, allowing the separation and reattachment points to fluctuate. There are many studies on geometric and pressure-induced turbulent boundary layer separation, but there are very few in which the test article, such as the speed Bump, offers simultaneously changing pressure gradients and surface curvature, without setting the separation point due

to geometric discontinuity.

Another factor that can affect boundary layer dynamics and separation is surface curvature. Without acting external pressure gradients, surface curvature has been found to heavily influence turbulent boundary layer behavior. Experiments have shown that if the flow streamlines near the free stream in the turbulent non-turbulent interface region exhibit convex curvature, creating an adverse pressure gradient, this led to a reduction in flow entrainment downstream. This is because the larger eddies, which engulf high momentum fluid, encounter the adverse pressure gradient, which reduces their motion into the free stream. This leads to reduced mixing and lower Reynolds shear stresses. However, if the flow streamlines exhibit concave curvature, this has a stabilizing effect for the turbulent boundary layer. It creates a favorable pressure gradient, leading to higher entrainment and mixing, thereby higher Reynolds shear stresses ([26], [27]).

In addition to surface curvature, the presence of pressure gradients can also solely affect boundary layer dynamics and separation. When there is a pressure gradient present with varying surface curvature such as with an airfoil, the turbulent boundary layer is influenced by both. For example, with an adverse pressure gradient and a convex surface, the streamlines are not as curved as the surface, so there is less influence of the curvature on entrainment than compared to when there is zero pressure gradient. It was observed that convex curvature through a strong adverse pressure gradient caused separation to occur farther upstream than for a similar pressure gradient flow without any curvature [22]. This is because when a surface diverges from the free-stream, mass-flow continuity requires the velocity near the wall to decrease more rapidly than for a flat plate under the same streamwise pressure gradient, as less momentum and energy are transported to the wall. For this case, the understanding is that curvature effects are significant in the wall-normal momentum equation and relatively negligible in the streamwise momentum equation.

All turbulent separated flows are inherently unsteady as all basic thermodynamics properties are significantly changing in time. Simplifications can be made to describe the flow as quasi-steady flow if the period of organized unsteadiness is relatively long compared with

the turbulence time scales and the phase-averaged flow can be described by the steady free-stream flow structure. However, in many turbulent separated flows, unsteadiness is induced throughout the flow field by the interacting inviscid and turbulent flow region, even if the approaching flow is steady. Low-frequency flapping associated with the shear layer in the backward facing step has been observed by Simpson [22]. Similarly, low and medium frequency motions associated with bubble breathing and vortex shedding, respectively, have been observed by [25]. In particular, low-frequency unsteadiness is of interest to various fields and its source has not been found. This is because the observed frequencies are lower than that of the incoming flow structures and could possibly be due to the large scale turbulent boundary layer motions or some natural instability of the separated region.

2.2 Turbulent Separated Flow Studies

2.2.1 Previous Studies

Many experimental turbulent separated flow studies have been conducted in the last few decades focused on separation as a result of an adverse pressure gradient ([25], [28], [29]), while others focused on separation as a result of surface curvature ([17], [18]). However, smooth-body separation under varying pressure gradients and surface curvature present unique complexities such as three-dimensionality, recirculating and highly turbulent flow. Some of the past studies looked at separation occurring over two-dimensional [30] and three-dimensional geometries that did not examine three-dimensional effects and had geometry which controlled the separation location ([17], [18]).

One of the most elementary geometries in flow separation studies is the forward-facing step, which creates two regions of separated flow: one, upstream of the step face and two, downstream of the step corner. The separated flowfield is a result of the combination of an adverse pressure upstream of the step face and the geometry of the step corner. Pearson et. al (2013) studied the forward-facing step using two-dimensional time resolved PIV and investigated the separated region in front of the step using conditional averaging based

on reverse flow fraction. Distributions of separation bubble area and the separation point locations were calculated and a power spectral density premultiplied by frequency of the streamwise centroid of reverse flow showed a broad peak at 30 Hz [17]. Subsequent analyses found that the separation region grew simultaneously in both the wall-normal and streamwise directions and that transverse movement of fluid along the step face, modulated by low-velocity perturbations in the upstream boundary layer, is dominant in determining the size of the separation bubble. The maximum extent of the upstream position of separation (i.e. largest reverse flow fraction) was found to be limited by the convection of mass over the step corner.

In 2015, Weiss and Taifour investigated the unsteady behavior in incompressible, pressure-induced turbulent separation bubbles (TSB). Results indicated that the low frequency pressure fluctuations observed upstream of the mean separation and downstream of the mean reattachment are consequences of the low frequency contraction and expansion of the separation bubble, which had a Strouhal number $\S_t = 0.01$ [20]. A higher frequency mode with a Strouhal number $\S_t = 0.35$ was associated to the roll-up of vortices in the shear layer above the recirculation bubble and their shedding downstream [20]. These two phenomena are reminiscent of the "flapping" and "shedding" modes observed in geometrically fixed-separation experiments, though their Strouhal numbers are different. The breathing mode was also shown to be similar to the low-frequency unsteadiness observed in shock-induced separated flows at supersonic speeds.

Following the work conducted by Weiss and Taifour [20], Le'Floch et.al. [25] measured the pressure and velocity fluctuations of pressure induced TSBs of various sizes, with emphasis on low and medium energy frequencies. It was found that wall-surface pressure fluctuations were attributed to the low frequency breathing of the TSBs and the effect of the adverse pressure gradient on the turbulent structures in the boundary layer. It was also suggested that the size and structure of the TSB has a strong influence on its low-frequency unsteadiness (i.e. as the size of the bubble increases, the amplitude of the low-frequency breathing or contraction grows as well). Partial orthogonal decomposition (POD) on the flow field in the recirculation

region revealed the time coefficient of the 1st POD mode was well synchronized with the time history of total reverse flow area, suggesting the 1st POD mode was an indicator of the breathing motion. Spectra investigations showed the potential of scaling laws between the pressure fluctuations and local maximum Reynolds shear and wall-normal stresses, but did not indicate any direct relationship between turbulent stresses and the low-frequency breathing motion [25].

Byun et. al. designed one of the earliest experimental smooth-body separation studies of an axisymmetric hill ([31], [32], [33]). The separated region on the leeward side of the hill was characterized using a variety of techniques such as oil flow visualization, laser Doppler velocimetry, and high frequency surface pressure measurements. Several key observations were made in these studies: 1) the separation line near the hill's apex was three-dimensional 2) the reattachment point was preceded by a region of reverse flow, and 3) a small region of slow-moving separated flow near the apex. Large eddy-simulations of this geometry were conducted and indicated poor agreement with the experimental findings within and near the separated region ([34], [35]).

Bell et.al corroborated the findings by Byun et. al. with the FAITH Hill (Fundamental Aeronautics Investigates the Hill) model shown in Figure 2.1. The geometry was a axisymmetric, modified cosine shaped, wall-mounted hill [1]. The flow was investigated through qualitative and quantitative techniques such as water tunnel dye injection, pressure sensitive paint, PIV, fringe-imaging and luminescent oil film skin friction tagging [36]. These studies demonstrated the ability to analyze the flow utilizing a variety of experimental methods to characterize both surface and off-surface flow features, point-wise steady and unsteady 3D velocities, and determine mean surface pressures.

2.2.2 Currently Ongoing Studies

Recently, Lynch et. al. revisited the classic Bachalo-Johnson axisymmetric transonic hump experiment, which is a scaled version of the original geometry [37], consisting of a circular bump on a constant diameter cylinder aligned with the flow, as shown in Figure 2.1. The

geometry was tested in a wind tunnel under transonic freestream conditions and is an experimental characterization of transonic, turbulent, separated flow generated by an axisymmetric model. The flow is turbulent approaching the hump and becomes locally supersonic at the apex, leading to shockwave-boundary layer interactions, a turbulent separation bubble and reattachment some distance downstream. The interaction and separated region are investigated through various experimental techniques. Mean surface pressure, mean skin friction, and mean and fluctuating velocity fields were measured, along with characterization of the tunnel boundary conditions to support an ongoing simulation challenge ([2], [38]).

Since 2020, an ongoing collaboration between Virginia Polytechnic Institute and State University and NASA has developed a CFD validation benchmark geometry named BeVERLI (Benchmark Validation Experiment for RANS and LES Investigations) [3]. Experiments have investigated subsonic turbulent boundary layer flow over the superelliptic-shaped, wall-mounted hill to characterize surface and off-surface turbulent flow structures. The symmetric model can be rotated through two orientations: 0 degree case where the windward face is normal to the oncoming flow, and the 45 degree case where one of the corners is normal to the oncoming flow. The BeVERLI hill experiments feature highly three-dimensional flow with a large boundary layer to hill height ratio of $\delta/h = 1 - 3$. Boundary layer separation under varying curvature and pressure gradients was observed on both the windward and leeward sides at low Reynolds number, but only leeward side separation was observed as the Reynolds number increased [39]. Surface pressure measurements were also taken along the bump centerline where a sudden increase was observed on the leeward side, indicating the presence of a large separation bubble.

Results from recent studies on the BeVERLI hill [3] summarized the flow topologies, Reynolds numbers effects, three-dimensionality and non-equilibrium flow that results from various geometric and inflow definitions. Downstream of the hill peaks, surface flow topologies were similar to the owl-face pattern of the first type with two foci and two saddle points. However, the centerline saddle points near separation and reattachment appeared to bifurcate into a centerline node and two spanwise symmetric saddles when higher local

curvature was present [3]. The BeVERLI hill exhibits strong flow asymmetry in both the 0°and 45°cases. The 0°case asymmetry was observed in unsteady pressure measurements which revealed the asymmetry switched sides randomly with a long timescale. However, the 45°case exhibited asymmetry and symmetric behavior which depended on Reynolds number (symmetric for $Re_h = 250,000$ and $325,000$ and asymmetric for $Re_h = 650,000$). The unsteady surface pressures did not exhibit any switching. The interactions between the surface curvature and varying pressure gradients were compared and found that the skin friction on the non-equilibrium up-slope was dominated by the favorable pressure gradient, even in the convex region, which would normally have an opposite influence.

2.3 Speed Bump Geometry

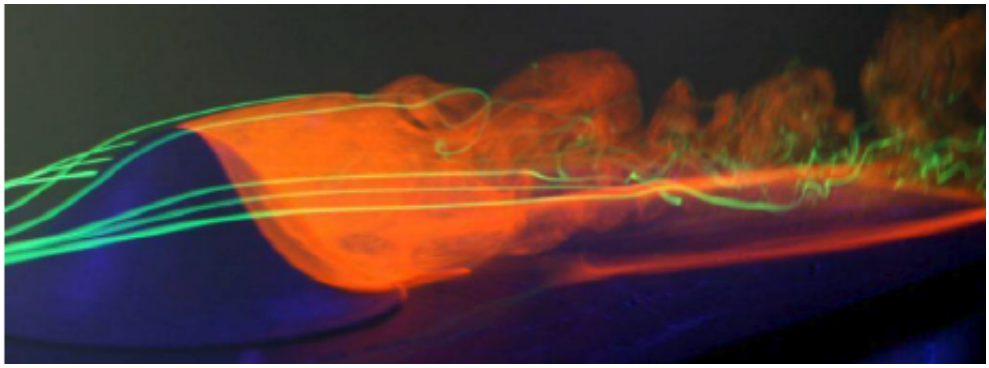
The speed-bump geometry was deliberately designed to induce separated flow to stress the current capabilities of turbulence modelling, while also being simple to manufacture and integrate into the existing wind tunnel facility. Originally designed for validation efforts of collecting benchmark quality data, the bump was outlined to meet requirements set forth by Slotnick and Spalart [40], including a point of separation not fixed by the geometry and a number of control variables to affect the extent of separation (e.g Reynolds number, inflow boundary layer thickness, etc.).

The speed bump is defined by the following function:

$$z(x,y) = \frac{h_o}{2} e^{-\frac{x}{x_o}^2} \left[1 + erf\left(\frac{\frac{L_b}{2} - 2y_o - |y|}{y_o} \right) \right] \tag{2.1}$$

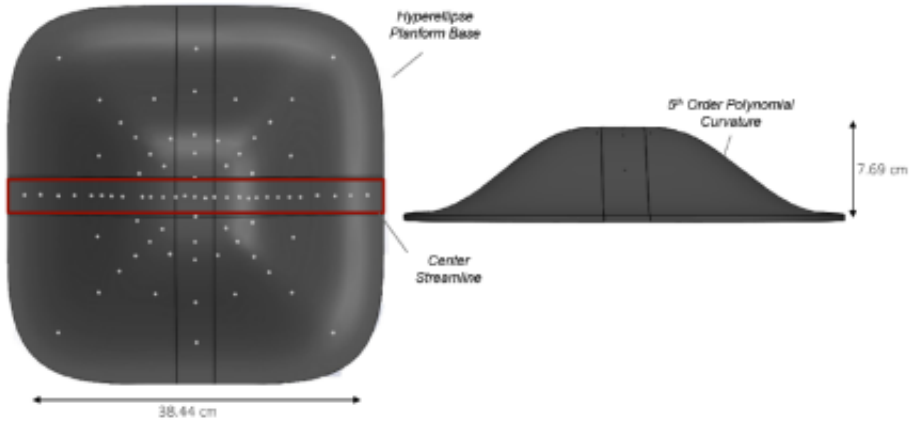
where x, y, and z are the streamwise, spanwise, and wall-normal directions, respectively. The width of the splitter plate, L_b, is 35.5 inches and the parameters used to define the geometry were chosen to be $x_o = 0.195L_b$, $y_o = 0.06L_b$, $h_o = 0.085L_b$, such that the bump height $h = 0.085L_b$. Note that L, the width of the tunnel cross-section, is equal to 36 in, which is bit larger than L_b. The resulting bump geometry from this definition is shown in Figure 3.2.

The streamwise Gaussian profile resembles a wing-like shape that presents a unique challenge to current turbulence models and CFD capabilities by not encouraging flow separation at a constant location. The profile tapers towards the side walls of the wind tunnel test section in the shape of an error function to minimize sidewall interactions and promote quasi two-dimensional behavior along the spanwise centerline and three-dimensionality around the shoulders. The geometry is symmetric on either side of the streamwise centerline. In addition to the physical bump, the tunnel walls and ceiling were decided to be apart of the simulation definition as well. Therefore, the level of the confinement between the geometry and ceiling is a tunable parameter to affect the extent of flow separation, along with the inflow boundary layer thickness and the freestream Reynolds number. The speed-bump geometry is unique



(a)

(b)



(c)

Figure 2.1: Geometries of previous/ongoing validation experiments: (a) FAITH Hill [1], (b) Sandia Axisymmetric Transonic Hump and CFD Challenge [2], and (c) BeVERLI Hill [3]

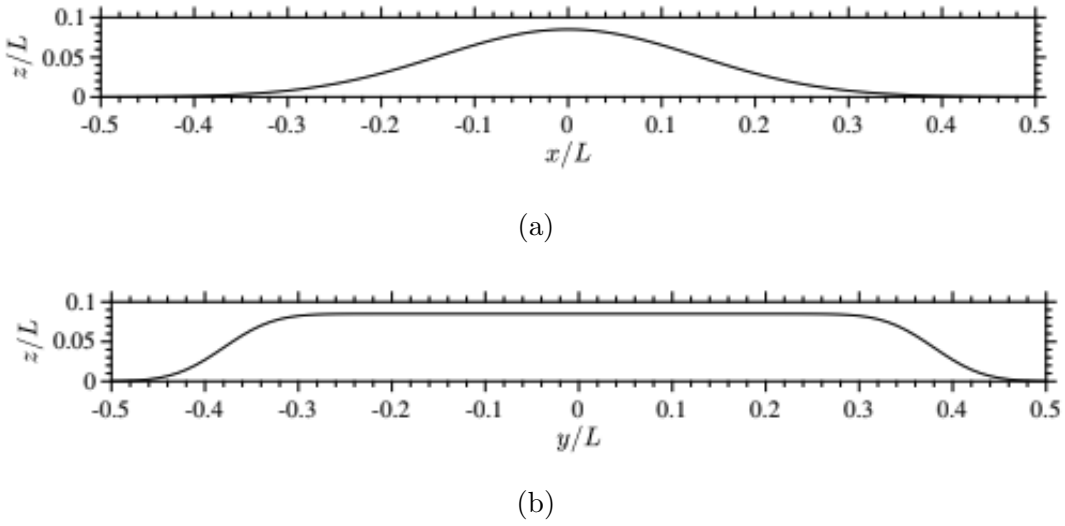


Figure 2.2: Geometry of bump model in (a) streamwise and (b) spanwise directions relative to wind tunnel test section width, L. [4]

compared to past test articles, such as the hill-type models, and the BeVERLI Hill, due to its thinner incoming turbulent boundary layer ($h/\delta = 10 - 12$) [3], and relevance to high-lift configurations.

Sarwas initiated the experimental investigation of this test geometry with studies focused on tripping the incoming boundary layer, conducting a survey on the in-flow freestream turbulence and measuring surface pressure variation over the bump surface [41]. Oil-clay visualizations of the bump surface showed strong flow separation on the leeward side of the bump and two large counter-rotating vortices for all Reynolds numbers that were tested ($1.32 \leq Re_L$ x $10^6 \leq 3.41$).

Samuell expanded upon this work and studied the effect of varying Reynolds number and confinement on surface pressures. The surface pressure profiles indicated a series of favorable and adverse pressure gradients along the streamwise direction of flow with the separation point located downstream of the bump apex in an adverse pressure gradient region. The

surface pressure profile was independent of Reynolds number above Re_L = 2.46 x 10^6 [12]. Samuell also compared the experimental results to a corresponding RANS simulation of the geometry, showing a similar insensitivity to Reynolds number in surface pressure. However, the simulation failed to capture a similar surface profile in the separated region. In addition, the separation dynamics were found to be heavily dependent on the RANS turbulence model and were very different from the experimental data, further exposing the deficiencies of RANS in modelling smooth-body separation.

Robbins continued the experimental investigation of the speed bump towards the development of a turbulence model validation dataset for separated flows. The boundary conditions were characterized, including a survey of the inflow uniformity and side-wall boundary layers at the inflow plane, along with measurement of the as-built test geometry [10]. Statistical measurements of the flowfield were acquired with PIV to study the inflow and separated region along various spanwise planes and it was found that features of separation exhibited a dependency on Reynolds number below Re_L = 2.55 x 10^6. Three-dimensionality of the separated region was observed in PIV results as the size of the separated region decreased as the distance from the spanwise centerline increased. A slight asymmetry in the mean pressure profile along the bump apex was noted and the cause is likely due to imperfect geometry.

Uzun and Malik performed a computation of a turbulent flow past the speed bump geometry in a hybrid DNS and wall-modelled LES (WMLES) 2D simulation at Reynolds numbers Re_L = 1 x 10^6 and Re_L = 2 x 10^6 [13]. It was observed that separation occurred later for the lower Reynolds number case, suggesting potential relaminarization of the boundary layer on the upstream bump slope. The results of both Reynolds number simulations also determined the presence of an internal layer generated in the acceleration region that evolves into a free shear layer that develops in the deceleration region and subsequently, separates. Internal layers were observed to form as a result of a surface curvature discontinuity, triggering significant changes in the flow's mean state and turbulent stresses [42]. The interaction between the varying surface curvature and pressure gradient was suggested to be responsible for the

decrease in surface skin friction coefficient C_f slightly upstream of the apex, indicating the convex curvature taking over as the favorable pressure gradient vanishes and the adverse pressure gradient appears.

Balin, Jansen and Spalart compared the results of DNS and improved delayed detached eddy simulation (IDDES) of a turbulent boundary layer over a two-dimensional Gaussian bump at $Re_\theta = 1000$ [43]. The DNS solution revealed significant affects of the strong pressure gradients on the near-wall turbulence, resulting in atypical C_f, velocity and shear stress profiles. Inspection of the boundary layer in the favorable pressure gradient region indicated the inner layers showed sign of a reverse transitional process and a strong reduction in Reynolds shear stress and turbulent kinetic energy. At the peak of the bump, a rapid increase in the near-wall turbulence was observed as the boundary layer encountered the strong adverse pressure gradient. In comparison, the IDDES model effectively predicted the favorable pressure gradient effects. However, the adverse pressure gradient effects were significantly delayed, especially by IDDES-SST, resulting in poor C_f predictions at the bump peak and downstream. The RANS-LES interface distance from the wall was varied and the predicted C_f profile best matched that of a full RANS-SST simulation when the interface was placed at $\delta/10$. Despite this, the RANS-SST and IDDES-SST were not able to capture the weakening of the near-wall turbulence in the favorable pressure gradient region. As a result, this study highlighted how C_f of the bump flow is determined largely by the near-wall physics and their response to strong pressure gradients, and the failure of IDDES models to predict these effects [43].

This past work sets the precedent for experimentally investigating the simultaneously varying surface curvature and pressure gradients present in the speed bump geometry.

Chapter 3

EXPERIMENTAL APPARATUS & METHODS

3.1 Facility

Experiments were conducted in the Low Speed Wind Tunnel (LSWT) located in the Aerodynamics Laboratory in the Aeronautics and Astronautics Department at the University of Washington-Seattle. Originally constructed in 1917, the 1612 square foot historic Aerodynamics Laboratory contains a large single room situated 8 feet underground with the wind tunnel located at its center. The below-grade walls are concrete and the upper walls are made of wood. The tunnel floor can be accessed through one of two staircases located on opposites side of the building. In addition to the actual tunnel structure, there are multiple workbenches, cabinets, testing artifacts and equipment located on the tunnel floor placed in such a way to minimize the blockage of the return flow to the tunnel inlet.

The temperature within the Aerodynamics Laboratory is controlled by a JencoFan belt drive centrifugal roof exhaust fan capable of flow rates up to 4,984 cubic feet per minute. The roof exhaust is situated in the laboratory ceiling, above the tunnel contraction. On the northern end of the building above the tunnel diffuser, a louvered ceiling inlet allows fresh air to enter laboratory while the roof exhaust is in operation. Prior to commencing testing, the exhaust fan was started and louvers were opened to allow the room to reach a steady-state temperature. However, during all tests, the exhaust fan was turned off and the ceiling louvers were closed to maintain the building pressure and prevent the vaporized fog fluid from leaving the room.

The LSWT is an open-loop circuit type wind tunnel with a velocity range of 15 to 60m/s (34 - 135mph). The tunnel is powered by a constant speed 200HP electric motor with pneumatically actuated variable pitch blades to adjust the freestream velocity. Designed

and manufactured by Engineering Laboratory Design, the LSWT is comprised of a radiused inlet, flow straighteners, a contraction, the test section, a diffuser, the fan assembly with drive motor, fan diffuser, supporting frames and two acoustic dampening wooden baffles. The walls of the tunnel are fabricated from a composite laminate fiberglass reinforced plastic and rigid PVC foam core finished with a polyester gel-coat. Flow enters the wind tunnel from the southern end of the laboratory through a 9 foot square inlet [5] and is conditioned by going through a foam filter media, a perforated plate, an aluminum honeycomb and a high porosity screen. The flow then passes through the 9:1 contraction ratio and enters the 3 feet by 3 feet by 8 feet test section. The test section walls are made of 1 1/4" thick type GM plexiglass to provide optical access and the corners are made of 5 inch radiused fiberglass. The test section is accessed through a plexiglass door on the east side of the tunnel, which is secured into place with quick release clamps during testing. Two square plexiglass cutouts are located in the test section floor and ceiling to provide access for pressure tubing and wiring. Upstream and downstream of the motor sections are wooden baffles for sound dampening. Engineering drawings of the LSWT and further detailed information such as the geometric coordinates of the contraction are provided in Appendix A.

The empty tunnel velocity was calculated by the pressure difference across two static pressure ports 11 inches upstream of the test section and another pair 9 feet upstream. Both pairs were located along the spanwise centerline on the top and bottom tunnel walls. Tubes coming from the static ports were fed into a single tube at a T-junction and then sent to a Omega PX653-10D5V pressure transducer, with properties listed in Table 3.1. Using this information, the pressure difference across the two pairs of static ports was a measure of the bulk velocity passing through the tunnel. The static pressure differential in the tunnel contraction was calibrated against a pitot-static differential measured with the boundary layer traverse with the bump installed. This calibration accounted for the blockage induced by the bump in calculating the bulk velocity. More details about the surface pressure measurement setup can be found in [10].

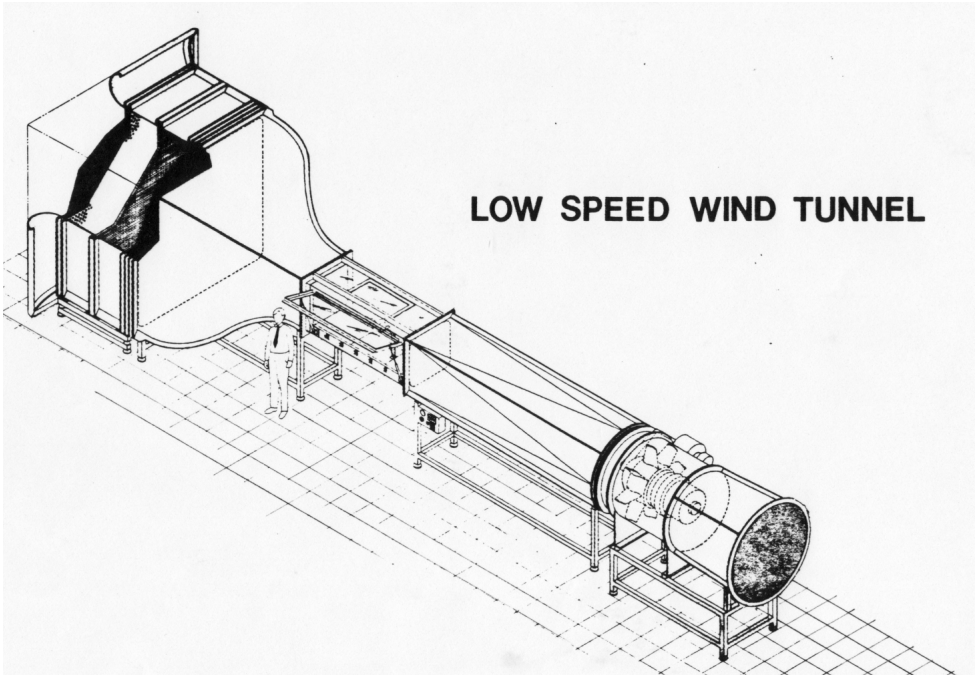


Figure 3.1: Isometric view of Low-Speed Wind Tunnel (LSWT) provided by Engineering Laboratory Design shown without the wooden acoustic dampening sections [5] located in the Aerodynamics Laboratory at the University of Washington.

	1	2	3
Manufacturer	Omega	Baratron	Baratron
Model	PX653-10D5V	226A	226A
Range (kPa)	0-2.49	± 2.67	± 6.67
Accuracy (Pa)	±6.2	±4.0	±8.0
Sensor Type	Capacitance	Capacitance	Capacitance
Thermal Zero (Pa/°C)	0.67	2.67	6.67
Thermal Span (%rdg/°C)	0.02	1.07	2.67

Table 3.1: Selected pressure sensor specifications. The Baratron pressure sensors were used to determine low frequency surface pressures [10]. See Appendix A for full manufacturer data sheets.

The total flow temperature through the test section was measured with a 1/16 inch diameter, Type K theromocouple (accuracy 1.5°C [44] which extended 3 3/4 inches into the flow from the top wall, 6 1/4 inches downstream of the downstream most edge of the test section. Ambient temperature, relative humidity (RH), and pressure were measured prior to each test using a Digi-Sense Traceable Digital Barometer with accuracies of ±0.4°C, ±3% RH, and ±4 mbar respectively [45]. Density was calculated using the ideal gas law, assuming a universal gas constant R of 287 J/(kg-K).

Data acquisition (DAQ) was performed using a National Instruments (NI) CompactDAQ platform. The Omega PX653-10D5V used to measure the pressure differential to calculate bulk tunnel velocity was sampled continuously at 100 Hz through a NI9205 voltage input module, featuring a gain amplifier and a 16 bit analog-to-digital converter (ADC). Similarly, the thermocouple was sampled continuously at 100 Hz using a NI9213 temperature input module, featuring a differential filter and a 24 bit analog-to-digital converter. For the highest temperatures observed in a test run, the maximum measurement error of this setup is 2.4°.

Manufacturer datasheets and calibrations for the equipment previously described are located in Appendix A. Details related to the speed-bump and splitter plate are found in Section 3.2, the PIV equipment in Section 3.3 and the surface pressure instrumentation in Section 3.4. Additional information on instrumentation error and uncertainties can found in Appendix A.

3.2 Speed Bump and Splitter Plate

The speed bump geometry, defined by equation 2.1, was manufactured out of epoxy and alumina trihydrate powder by Steven Seim of Cyber Modelle. It was CNC machined, painted and polished and has a maximum wall thickness of 0.5 inch. Robbins found the root-mean-square of the deviation of the as built-geometry to the mathematical definition was 0.018 inches, with the maximum deviation of 0.057 inches observed at the edges of the model closest to the tunnel walls [10]. The deviations were approximately symmetric about the model centerline. The height of the actual geometry was slightly lower than prescribed on

the streamwise faces, on the order of 1% of the bump height. The bump was fastened to a 3/8 inch thick aluminum backing plate with 6 bolts to increase its rigidity. This structure was mounted to two 3/4 inch thick MIC-6 aluminum plates having surface roughness less than 0.5µm and maximum flatness deviation of 0.005 inches, to ensure proper boundary layer development. This formed the splitter plate which divided the flow in the test section into flow going over and under the bump. The splitter plate was then secured to two longitudinal (streamwise direction) bars via metal straps and the bars were mounted to the tunnel walls via pins. The array of pin holes in the tunnel walls allowed to test for various tunnel confinement levels. The accessible vertical positions were $H/L = 1/4$, 1/3, 5/12, 1/2, 7/12 and 2/3, where $H/L = 1/2$ was considered nominal, which was used throughout the results of this thesis. To ensure sufficient clearance was available during installation of the structure, the splitter plate width L_b was 35.5", which is narrower than the test section width $L =$ 36". The gaps between the splitter plate and tunnel walls were filled with 3/4 inch wide, 1/4 in thick Neoprene foam strips to form an airtight seal and separate the regions above and below the plate.

Figure 3.4 shows a schematic of the bump and splitter plate and CAD rendering of the set up in the test section. The leading edge of the splitter plate featured a 8 inch long, 3D printed, 10:1 modified super-ellipse profile, to prevent transition in the initial presence of the splitter plate, upstream of the artificial trip. A strip of 240-grit sandpaper was placed across the joint between the leading edge and aluminum plate, located $1L$ upstream of the bump peak. Previous studies compared various tripping devices and it was found the boundary layer produced was independent of the trip used [41]. Therefore, the decision was made to use the finest grit sandpaper tested (240-grit). The most downstream splitter plate featured an adjustable 12-inch trailing edge flap via turnbuckle to control the location of the leading edge stagnation point. It was imperative the leading edge stagnation point was at or above the leading edge plate. To determine the stagnation point location, two pairs of 1/32 inch static pressure taps were placed on the upper and lower surfaces situated 0.5" from the leading edge along the test section centerline. Measuring the pressure differential across these taps

and tuning the trailing edge flap via the turnbuckle allowed for adjusting the stagnation point location until it was favorable.

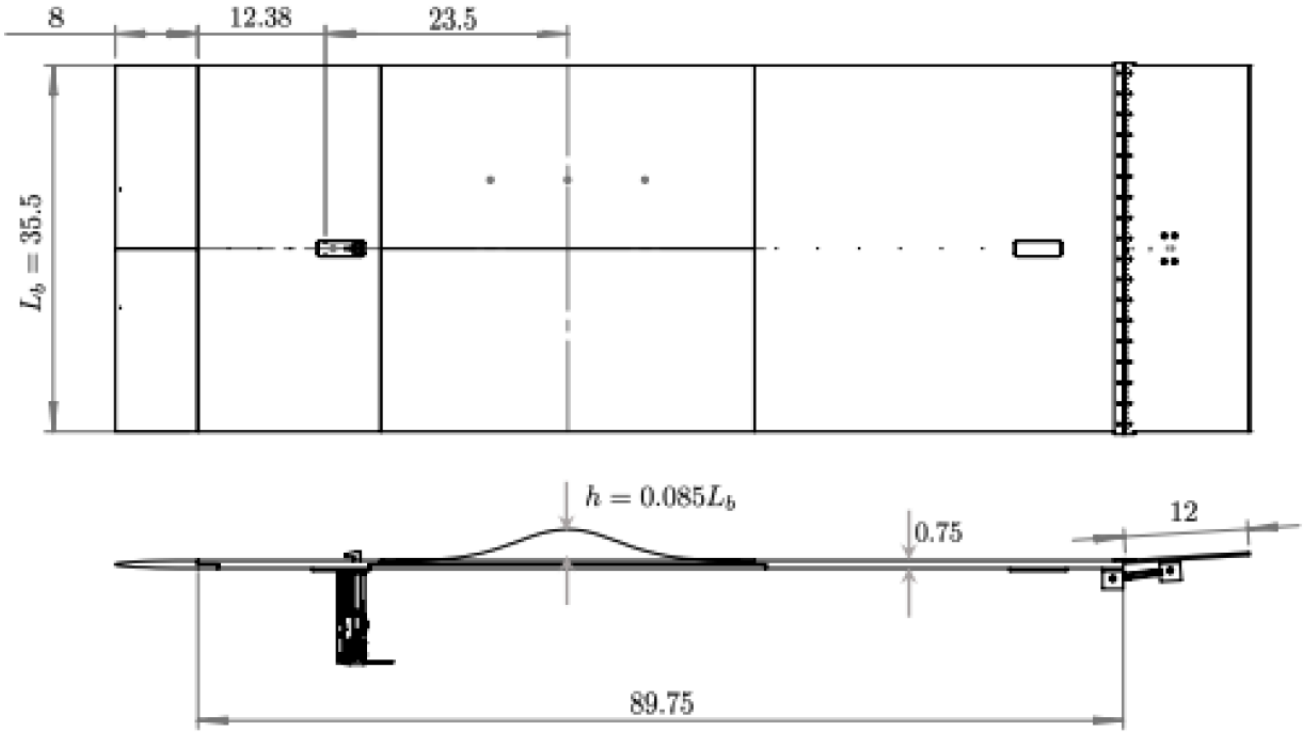
Both the upstream and downstream MIC-6 aluminum plates featured 1.5 in by 4.5 in cutouts where the motorized boundary layer traverse could be inserted for measurement of the streamwise velocity profiles ([11],[41]). The traverse was not used for this study so modular inserts made of the same MIC-6 aluminum were secured in the openings to be flush with the plate surface.

3.3 Particle Image Velocimetry

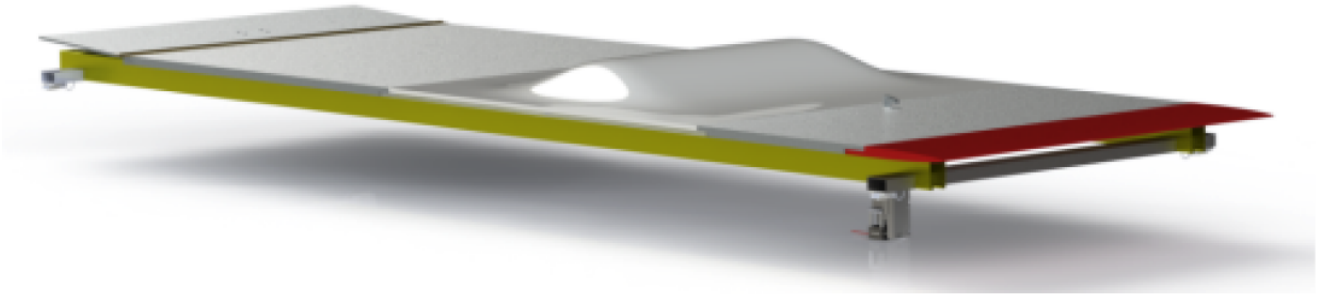
3.3.1 Side View Planar Double Frame PIV Setup

Detailed flowfield measurements were obtained using planar, double frame, double exposure PIV in a series of streamwise/vertical planes. Using up to two cameras, instantaneous and non-time resolved velocity fields up and over the speed-bump were experimentally measured. This method offered no disruption to the flow within the test section because all the PIV equipment was located exterior to the tunnel. Locations of smoke particles illuminated by the pulsing laser were captured in image pairs in quick succession. A cross-correlation of the image pairs was performed and yielded the particles' velocities at that instant. Thousands of image pairs were taken in every test allowing for the calculation of the mean and fluctuating quantities of various important turbulent flow parameters such as Reynolds stresses.

Differing from time-resolved PIV, which would require the laser to pulse at significantly higher frequencies, non-time resolved snapshots are statistically independent [46]. The PIV system was mobile via a traverse in the spanwise and streamwise directions, which enabled taking flowfield measurements at various streamwise and spanwise planes without any loss of calibration. Mean velocity fields, and Reynolds stresses were some of the quantities extracted from PIV to shed light on the flow dynamics and boundary layer separation phenomena. The PIV data was also acquired in conjunction with time-resolved pressure data (Section 3.4) to understand the flowfield from a new lens.



(a)



(b)

Figure 3.2: (a) Schematic and (b) CAD rendering of splitter plate and bump with removable pitot tube traverse installed. All dimensions in inches [4].

To perform PIV in the LSWT, the interior of the Aerodynamics Laboratory was filled with smoke particles using an off-the-shelf commercial Chauvet DJ Hurricane 1800 Flex fog machine, with a maximum output of 25,000 cfm [6]. The seeding material was Black Label Ray Tracer low density fog fluid, a mixture of glycols and de-ionized water, used typically to accentuate effects from lights and lasers. It is made for indoor use with slow dissipative properties, wide dispersion and long hang time, presenting itself as a good choice for PIV seeding. The fog machine was set to produce smoke at 15,000 cfm for six seconds at 2 minute intervals to maintain a constant seeding density throughout the duration of the test. However, the output was adjusted as necessary as the testing progressed.

A combination of one or two 5.5 megapixel Imager scientific complementary metal-oxide semiconductor (sCMOS) cameras were used to capture images of the illuminated flowfield. Specifically designed for laser imaging applications such as PIV, the camera specialized in low-light imaging combined with large signal variations and offered high temporal and spatial resolution over large fields of view illuminated by the laser sheet [7]]. The camera sensor dimensions were 2560x2160 pixels and each pixel was 42.25 m^2. During testing, data was transferred from the sCMOS cameras to a Camera Link high-speed frame grabber (CLHS) via a fiber optic cable. Images were recorded directly to RAM and offloaded to the computer hard disk simultaneously. This enabled the acquisition of a large number of samples (see Section 4.1)without concerns of speed limitations attributed to offloading directly to the hard disk.

A combination of two fields of view were imaged. The 8.76 by 7.39 inch large field of view (LFOV) was captured using a Nikon Micro 60 mm focal length lens to capture the overall flow characteristics, larger scale structures and boundary layer. The 2.62 by 2.21 inch small field of view (SFOV) was captured using a Nikon Micro 200 mm focal length lens to image the upstream slope and the developing boundary layer in greater detail. To achieve optimal illumination of the particles in the image, the lens apertures were set to an f-stop of f/8 and f/11, yielding average particle diameters of 4 and 2 pixels for the LFOV and SFOV, respectively. The pixel diameters were measured via visual inspection of the individual PIV

frames.

The seeded particles were illuminated by a Quantell EverGreen 200 dual pulsed neodymium doped yttrium aluminum (Nd: YAG) laser. It was rated for 200 mJ at a wavelength of 532 nm and a repetition rate up to 15 Hz [8] Specifically designed for PIV applications, the laser head was connected to an integrated cooling and electronic (ICE) unit during operation, which was controlled from outside the test section. For the streamwise PIV planes that were studied, the laser was directed out of a single aperture in the laser head, through a series of adjustable optics and lastly, through a cylindrical lens to create a thin laser sheet from the initial beam. The laser optics were tuned to ensure the light sheet was finely focused on the splitter plate and as streamwise and wall-normal as possible. The speed-bump and splitter plate were spray-painted black to minimize reflections off the surface.

The cameras were focused in the spanwise direction using calibration targets placed coincident with laser sheet at a known position. The LFOV setup was calibrated with a two-level double-sided LaVision Type 204-15 calibration target, with dimensions 204 mm by 204 mm. The SFOV setup was calibrated with a smaller two-level double-sided stereo-PIV LaVision Type 058-5 calibration target, with dimensions 58 mm by 58 mm. The targets contain a grid of circles, triangles and squares, as shown in Figure , and the layout is known by the DaVis software. Using automated mark detection, DaVis locates the marks and recognizes the orientation of the calibration target. The resulting scale factors were 11.13 and 38.44 pixels/inch for the LFOV and SFOV respectively.

The LaVision Programmable Timing Unit (PTU) X controlled the laser and camera timing and synchronization within 10 ns [9] via BNC cables. Specifically, the PTU linked to the flash lamp and Q-switch trigger inputs of each laser cavity and the timing port of the cameras. The PTU interfaced with the PIV acquisition computer via USB.

The laser head was mounted onto a Vere optical breadboard located above the test section. The cameras were mounted onto a right angle frame as shown in Figure 3.5. With this setup, the cameras were viewing into the test section from the port side of the tunnel (when looking downstream).

Figure 3.3: Equipment used for PIV measurements of the speed-bump. (a) Chauvet Hurricane 1800 Flex fog machine [6], (b) Imager sCMOS camera [7], (c) EverGreen 200 ICE and laser head [8] and (d) LaVision PTU X timing unit [9].

The two-axis traverse featured four motorized Velmex BiSlides moving in two orthogonal directions: streamwise and spanwise. It was possible to have the camera traverse in the wall-normal direction as well, but the slide was removed and the camera was fixed to one of the legs of the right-angle frame. The optic table was fastened to two parallel 40 inch lead-screw driven slides, powered by a Vecta Type 34T1 double shaft stepper motor. This sub-assembly created motion in the spanwise direction. The lead-screw slides were mounted to two parallel 100 inch belt drive slides, powered by a Vexta Type 34T1 10:1 gear ratio stepper motor. This allowed for movement in the streamwise direction. Ultimately, this assembly was fastened to the top of the wind tunnel, as shown in Figure . The motors were powered and controlled by a VXM-2 stepper motor controller and interfaced to a computer via an RS-232 cable. A LabView script controlled the position of the center of the camera's field of view. The length of the slides and the wind tunnel geometry bounded the PIV domain to $-0.74 \leq x/L \leq 1.18$ and $-0.24 \leq y/L \leq 0.24$, with limit switches on each slide to prevent movement outside these ranges. Information on the equipment that is apart of the setup, including data sheets and camera calibrations, can be found in Appendix A.

3.3.2 Top Down Planar Double Frame Particle Image Velocimetry Setup

A different setup was employed to conduct PIV measurements in spanwise plane parallel to the splitter plate at various heights. The laser was again directed out of a single aperture in the laser head, through three 90 degree turning mirrors on adjustable kinematic mounts and lastly, through a cylindrical lens to create a spanwise PIV plane, as shown in Figure . An optical rail was fastened to the right-angle frame above the laser head. The camera was centered on the optical rail and mounted in place, pointed downwards into the test section. Only the 60 mm lens was used in the top-down PIV so the two-level double-sided LaVision Type 204-15 calibration target, with dimensions 204 mm by 204 mm, was used to perform the calibration. This yielded a scale factor of 265.7 pixels/inch.

In addition to the 2-axis traverse used in the setup previously, a 15 inch lead-screw driven slide, powered by a Vexta Type 23T1 single shaft stepper motor was secured to the legs of the

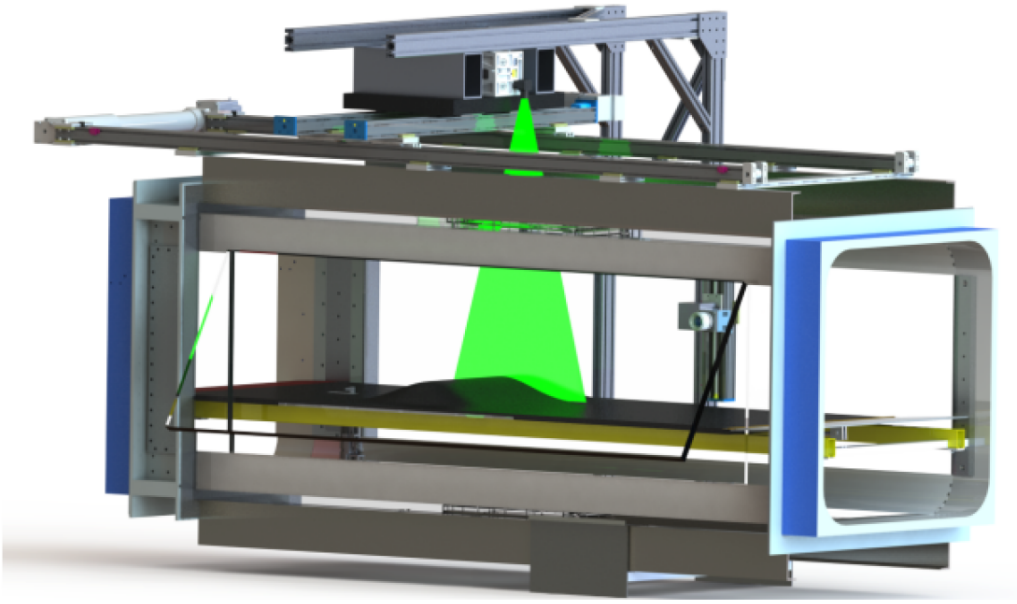


Figure 3.4: Isometric CAD rendering of the traversing PIV apparatus installed on top of the Low-Speed Wind Tunnel [10].

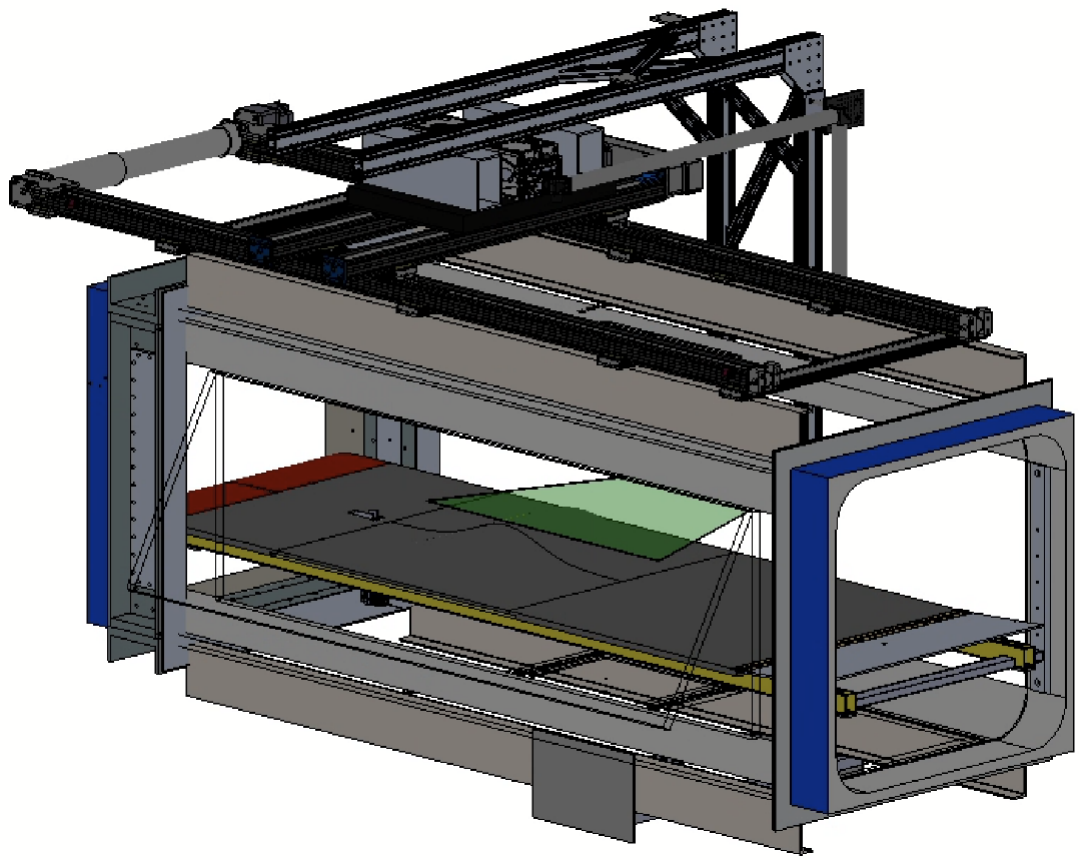


Figure 3.5: Isometric CAD rendering of the traversing PIV apparatus installed on top of the Low-Speed Wind Tunnel.

right-angle frame. The final kinematic mount was mounted to the slide to move in the wall-normal direction. An agent VXM-1 controller was connected to the master VXM-2 stepper motor controller and interfaced to a computer via the same RS-232 protocol. The same LabView script controlled the position of the kinematic mount to move the laser sheet to different heights, with limit switches on either end of the slide to prevent erroneous movement. Information on the equipment that is apart of this new setup, including manufacturer data sheets and camera calibrations, can be found in Appendix A.

3.3.3 PIV Data Acquisition

The first data sets were acquired along a series of streamwise locations over the bump at three spanwise planes closely corresponding to the surface pressure taps locations as shown in Figures 4.2a and b. The images were acquired at a constant vertical position. The streamwise overlap between adjacent fields of view was 30% to create statistical flowfields over the speed bump geometry. The second data set comprised of utilizing both SFOV and LFOV measurements simultaneously, with the SFOV focused on the upstream slope and LFOV focused on the separation region, along the bump centerline. The motorized slide was not used for this setup as well, so the cameras were fixed to the right-angle frame. The streamwise overlap between the two fields of view was also 30% based off the SFOV streamwise width to create statistical flowfields in this region of interest. Lastly, the data sets for the top down PIV were acquired at five streamwise locations and two vertical planes. The third motorized slide was used for this setup so data was captured at different vertical heights to investigate the flowfield near separation, reattachment and within the separation bubble. Certain regions of interest were larger than the LFOV dimensions and required two adjacent frames to fully capture the flow. Therefore, the streamwise overlap between the two fields of view was 30% to create statistical flowfields.

The PIV image acquisition was fully performed through DaVis 10 by Lavision. Another program, run in LabView on a separate computer, monitored and recorded tunnel conditions including freestream velocity, and controlled the position of the traverse moving the cameras and laser head. Prior to the start of each test, the traverse was brought to the most upstream limit switch, in each of its axes, to zero its position. Coordinates were provided to the program through a .csv file and the LabView program parsed through each line to move the traverse to the desired position. Once the traverse arrived at each coordinate, image acquisition was started by the user through DaVis 10. After acquisition at the station was complete, the tunnel was turned off to reduce the interior temperature and improve the seeding density. This was repeated for all stations specified by the user. In this way, the

details of the image acquisition process such as temperature control and maintaining seeding density could be monitored.

For the spanwise planes, a total of ten thousand image pairs were acquired at each streamwise location. Therefore, an entire spanwise plane was defined by thirty thousand PIV image pairs. This was done to ensure adequate statistical convergence of the mean flow properties. Ten thousand image pairs per streamwise location were selected because a previous study found that one thousand image pairs did not provide enough information to statistically converge the data [10]. Images were acquired at the maximum laser frequency of 15 Hz. The time between pulses, dt, was determined such that the maximum particle displacement was limited to 12 pixels. For data sets taken using both the SFOV and LFOV simultaneously focused on the upstream and separated region, a compromise was made in setting the dt as only one input time can be used in DaVis. As a result, the mean dt between the SFOV and LFOV individual dt's was used.

Table 3.2: Time between image pairs, $dt(\mu s)$, for maximum particle displacement of 12 pixels, for double-frame PIV for nominal freestream velocities tested.

Nominal Velocity (m/s)	LFOV (μs)	SFOV, LFOV (μs)
40	25.4	17.35
60	17.95	–

The times between laser pulses for the nominal velocities tested in both the LFOV and SFOV are listed in Table 3.2. Other pertinent PIV parameters are summarized in Table 3.3.

3.3.4 Data Processing and Reduction

In addition to performing the image acquisition, the PIV processing was also conducted in DaVis 10. The raw images were pre-processed and initial data reduction was performed through using a variety of available functions. After the acquisition process, the images were

subject to a stabilization technique to attenuate the effects from the vibrations due to the wind tunnel. The images were shifted relative to the wall reflection on the curved portion of the bump, taken from the first image of the data set. Next, an average background subtraction was performed on the images to remove the majority of the steady-state background artifacts. A filter length of 49 images was found to work well for this application. Then, the last pre-processing step consisted of manually masking the image below the bump surface and splitter plate. This was done by identifying the surface reflection of the laser located below the line of pixels with intensities approximately equal to the well-depth of the camera sensor.

The images were then analyzed using a multi-pass, cross-correlation method with iterative image deformation. Two two initial passes employed 128 x 128 pixel correlations windows with 50% overlap. The final passes were done with a 32 x 32 pixel interrogation window with Gaussian weighting and 75% overlap. Vector validation was performed using a correlation level filter ($r < 3$) and three passes of a median filter utilizing the universal outlier detection scheme with 5x5 vector windows [46]. The post-processed instantaneous vector fields were then exported to MATLAB for further processing and generating a single composite flowfield encompassing all of the streamwise stations along a single spanwise plane. This was possible because of the overlap between adjacent streamwise field of views.

In MATLAB, the edges of the vector fields were trimmed by 10 vectors in order to compensate for poor correlations. The coordinate system of the individual vector fields were transformed into the global wind tunnel coordinate system to locate each station in space to ultimately generate the single composite flowfield. This was accomplished using the location of the bump apex obtained from the raw images of the relevant data set. The position of the bump apex is known in the global coordinate system ($\frac{x}{L} = 0$) so the absolute position of the frame with the apex was determined with the pixel location of the apex in the raw image and the scale factor from the camera calibration. All other coordinates were prescribed relative to the bump apex using the .csv file sent to the traverse and the frames were moved in the streamwise direction accordingly. The statistical quantities of the vector fields were

Table 3.3: Summary of PIV parameters for LFOV, the SFOV and LFOV combination, and the top-down LFOV.

	LFOV	SFOV, LFOV	Top-Down LFOV
FOV Dimensions (mm.)	(222.5x187.7)	(66.5 x 56.1), (222.5x187.7)	(222.5x187.7)
Image Pairs per Station		10000	
Acquisition Rate (Hz)		15	
Calibration (px/mm)	11.13	38.44, 11.13	10.46
Lens Aperture	f/8	f/11, f/8	f/11
Particle Image Diameter (px)	≈ 2	$\approx 4, \approx 2$	≈ 2
Particle Size (mm)	≈ 0.18	$\approx 0.1, \approx 0.18$	≈ 0.2
Max. Particle Displacement (px)	12	9	12

computed at each station and added to a scatter interpolant object, which was used to generate a stitched composite flowfields for the flow property of interest. The scripts used to perform operations can be found in Appendix B. An example of the workflow is illustrated in Figure 3.6.

3.4 Pressure Instrumentation

3.4.1 Low Frequency Pressure Data Acquisition Setup

Robbins acquired low frequency surface pressure profiles over the bump surface and splitter plate with 49 total pressure taps [10]. 3 taps were located on the upstream splitter plate and 6 on the downstream splitter plate, all along the spanwise centerline. 40 taps were located on the bump surface. Taps were concentrated on the leeward (downstream) side of the bump to increase resolution in the separated region. They were also located along many symmetric planes to understand the spanwise pressure variance and the bias present. Details of the pressure transducers, diagram of the tap locations on the bump and splitter plate,

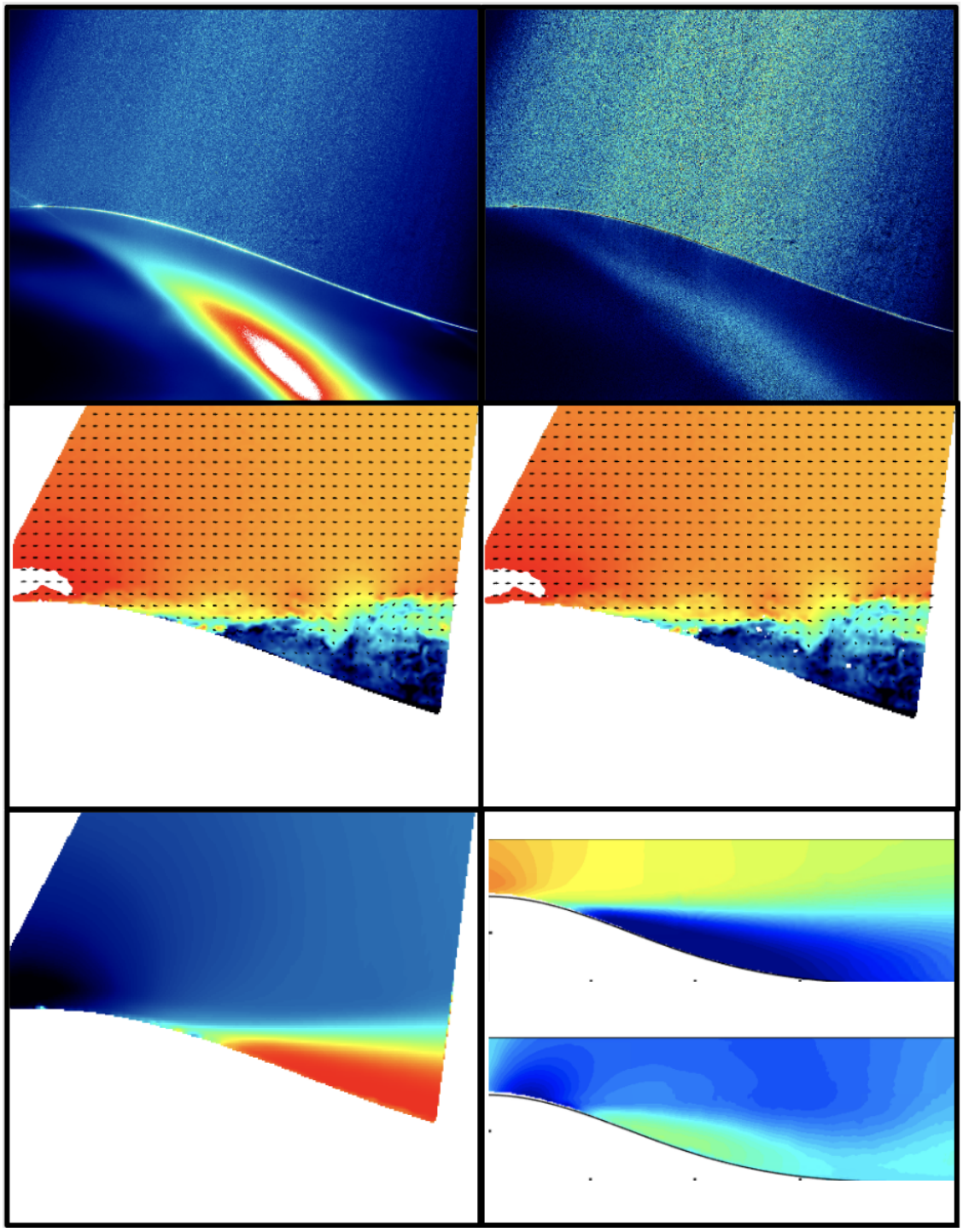


Figure 3.6: PIV reduction workflow example ($Re_L = 2.56 * 10^6$ along centerline). From top left to bottom right: raw image, background subtraction, instantaneous vector field with wall masking, filtered vector field, averaged scalar field, composite flowfield.

and acquisition setup can be found in Appendix A.

3.4.2 High Frequency Pressure Acquisition Set up Synchronized with PIV Setup

This data was acquired in collaboration with Kevin Manohar from the University of Calgary and his advisors' support has been invaluable towards this project. High frequency surface pressures over the bump surface and splitter plate were acquired across 31 pressure taps used in the low frequency acquisition. 31 Miniature Amplified Output pressure sensors were acquired and connected to the pressure taps via short 1/32 diameter polyurethane tubing and routed to a custom printed circuit board (PCB) to interface the sensors with the Advantech PCIE-1805 DAQ card. Tap 1 was selected to be the reference tap as tubing was routed via a manifold to the reference port of all pressure sensors. Leak sealant was applied at the overlap between the tubing, the tap, and the pressure sensor to ensure minimal to no leakage. The tubes were bundled and taped to the tunnel walls with aluminum speed tape and routed out of the tunnel test section through a gap in the tunnel floor. The gap was sealed with speed tape.

The DAQ card contains a single multiplexed analog/digital converter with 32 single ended input/outputs. All channels were sampled at 1 MHz total, so each channel was sampled at 31.25 kHz. The final channel, which was not connected to a pressure sensor, was connected to the laser's q-switch line to be read as an analog voltage to be used to synchronize the PIV data to the pressure readings.

Chapter 4

PIV RESULTS AND DISCUSSIONS

4.1 Basic Features of Separated Bump Flow

4.1.1 Bump Geometric Characteristics

The evolution of the non-equilibrium turbulent boundary layer in the presence of changing pressure gradients and surface curvature is investigated and the bump geometric characteristics are first extracted. The speed bump defined by equation 2.1, exhibits varying geometric characteristics in the streamwise direction that have an effect on the flow topology, boundary layer evolution, and separation properties. Figure 4.1 shows the progression of the local angle of the wall and the local radius of curvature, in the streamwise direction. Upstream of the bump apex, the surface quickly undergoes changes in these properties at streamwise locations that are in close proximity to each other. The wall angle gradually increases from 0° to a maximum of 20.5° at $x/L = -0.135$, as seen in Figure 4.1a. As the bump apex is approached, the wall angle begins to rapidly decrease to 0°. Due to symmetry, downstream of the bump apex, the wall angle rapidly decreases to a minimum of -20.5° at $x/L = 0.135$ before gradually increasing to 0°.

Figure 4.1b shows the changing surface curvature R/L in the streamwise direction. The initial upstream bump surface transitions to concave curvature, where R/L decreases to a local minimum of 0.534 at $x/L = -0.2441$. R/L then rapidly increases as the inflection point on the bump surface is approached at $x/L = -0.135$, which is also the location of maximum wall angle and magnitude of wall-normal slope. The bump surface quickly transitions from concave to convex curvature at this inflection point and R/L rapidly reduces to the global minimum of 0.2206 at $x/L = 0$. Due to symmetry about $x/L = 0$, the pattern of changing curvature is mirrored about $x/L = 0$. This curvature pattern is different from the steeper

BeVERLi Hill [3], which exhibits shorter distances over which the interchanging curvatures act and a flat top after the upstream convex region. More comparisons between the speed bump, the BeVERLi Hill and other bump geometries can be found in [3]

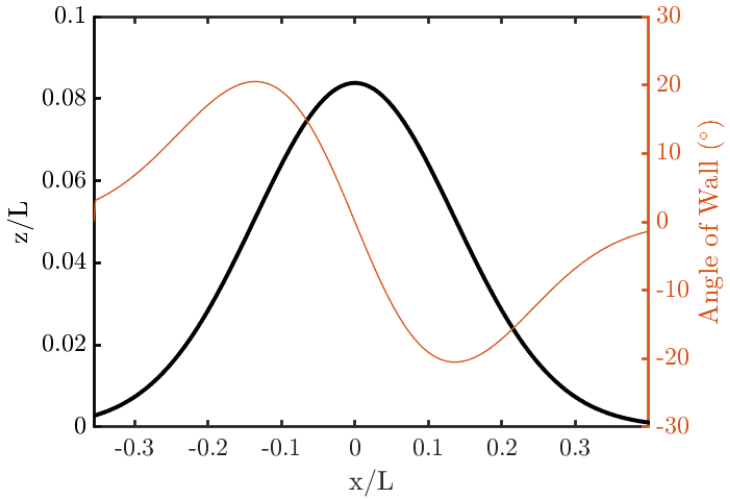
4.1.2 PIV Fields Summarizing Flowfield Characteristics

Several composite statistical flowfields were obtained via PIV across various Reynolds numbers and streamwise and spanwise locations. A summary of the test conditions is shown in Table 4.1 and the locations of the PIV frames in Setups 1 and 2 are shown in Figure 4.2.

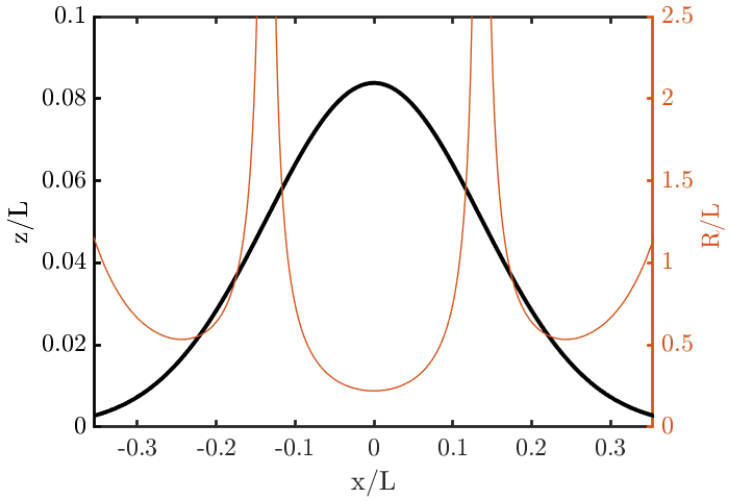
	Setup 1	Setup 2
Streamwise Extent of Data Capture (x/L)	(-0.35, 0.53)	(-0.05, 0.53)
Spanwise Planes (y/L)	$0, \pm 0.083, \pm 0.125, \pm 0.25$	$0, 0.083, 0.125$
Re_L (x10^6)	1.39, 1.97, 2.55, 3.17, 3.44	2.56, 3.48
PIV Frames Captured Per Streamwise Location	1000	10000

Table 4.1: Test conditions and streamwise and spanwise locations of PIV acquisition across Setups 1 and 2.

Composite statistical flowfields created from PIV obtained from using Setup 1 (4.1) are presented for the nominal case of Reynolds number $Re_L = 3.44x10^6$ in Figure 4.3 along y/L = 0.083. All properties are normalized by the friction velocity u_τ, calculated from the upstream boundary layer [10]. The mean streamwise velocity field in Figure 4.3a indicates significant flow acceleration near the bump peak as mass flow conservation and the favorable pressure gradient increase flow velocities. Downstream of the bump apex, the flow separates off the leeward surface after encountering an adverse pressure gradient. The separation triggers the formation of a shear layer that reattaches further downstream, creating a fluctuating separation bubble bounded by the separation and reattachment points and the shear layer above.

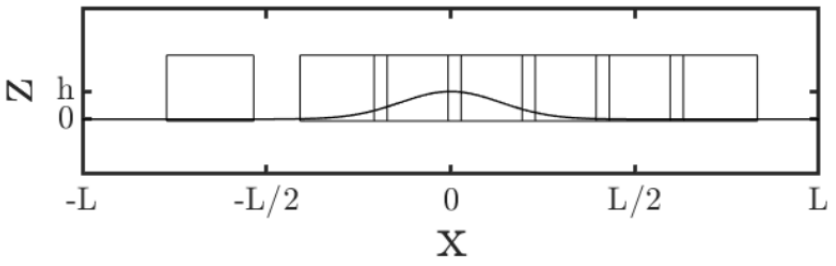


(a)

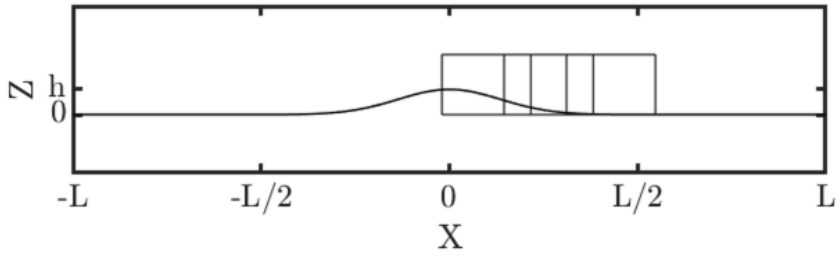


(b)

Figure 4.1: Evolution of (a) the angle at the bump wall and (b) the radius of curvature in the streamwise direction.



(a)



(b)

Figure 4.2: PIV fields of view acquired along a spanwise plane with large 9 $in.^2$ field of view for (a) Setup 1 [10] and (b) Setup 2.

There is a large region of streamwise flow with negative velocity, forming the aforementioned separation bubble. As shown in Figure 4.3b, the wall normal velocity downstream of separation seems to transition from negative to positive as the distance to the wall decreases, where the flow attains maximum W/u_τ just above the wall. This region of positive mean wall-normal flow is located below the shear layer, suggesting the flow is drawn upwards from the surface, into the separated region near the spanwise centerline, and transported downstream. The mean normalized turbulent shear stress $-uw/u_\tau^2$, illustrated in Figure 4.3c, is noticeably greater in the separated region, indicating turbulence production, until it starts to dissipate further downstream. u and w are the velocity fluctuations. Near flow detachment, there is a region close to the wall encapsulating the separation point envelope where high

turbulent shear stress is observed, suggesting significant streamwise and wall-normal velocity fluctuations that are also contributing to the production of turbulence. Similar to the shear stress, the mean in-plane turbulent kinetic energy, depicted in Figure 4.3d, is also noticeably greater within the separated region and then starts to dissipate further downstream.

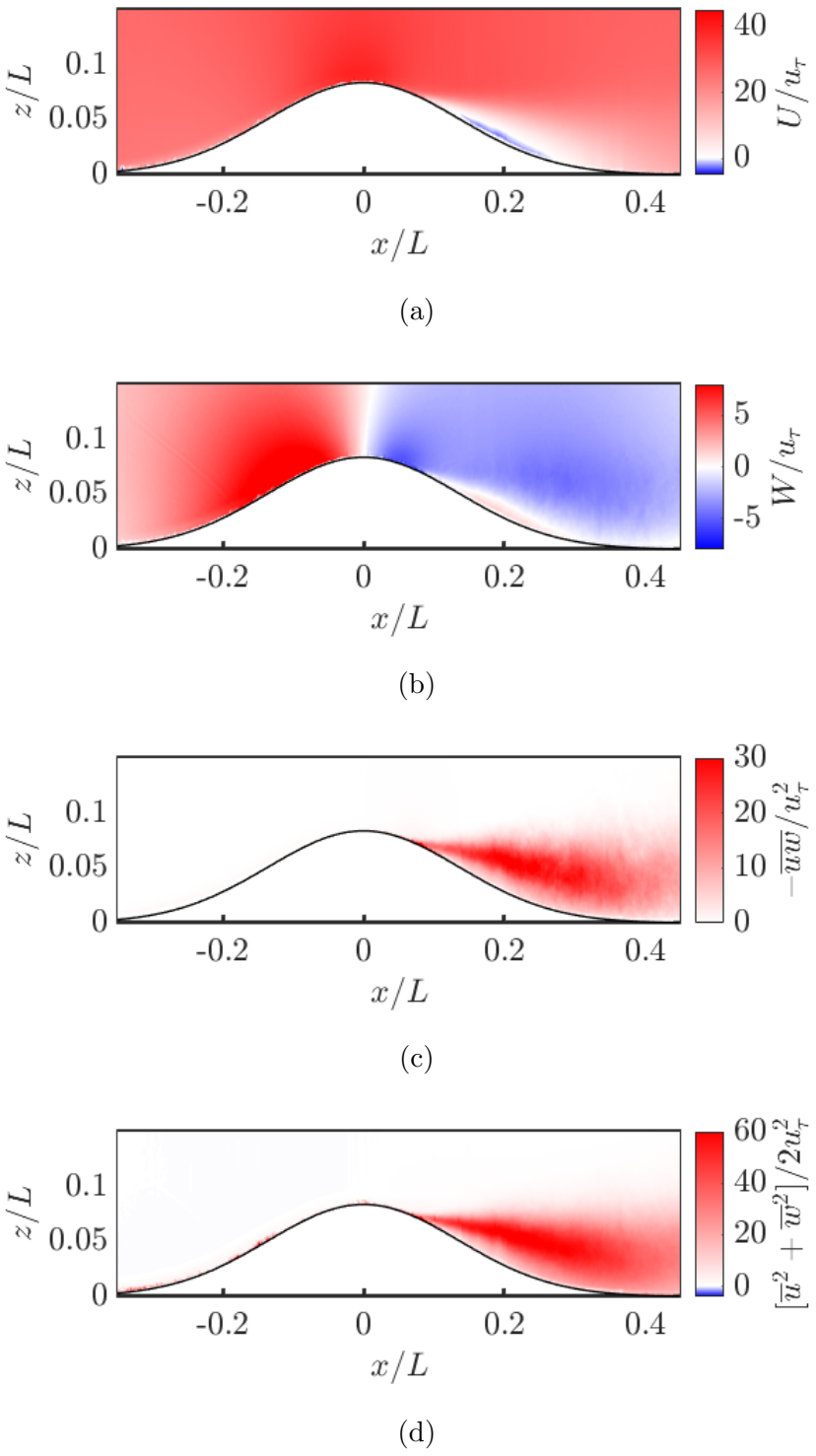
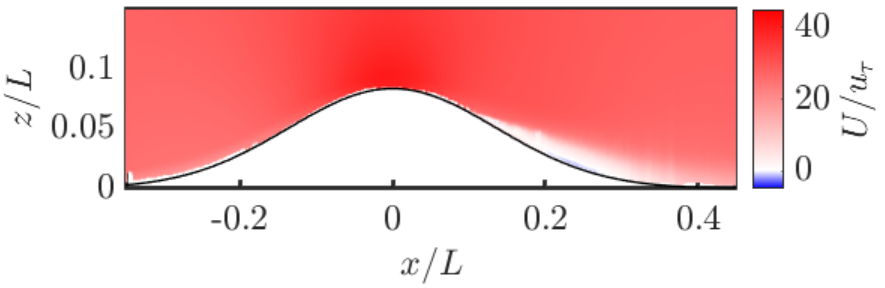
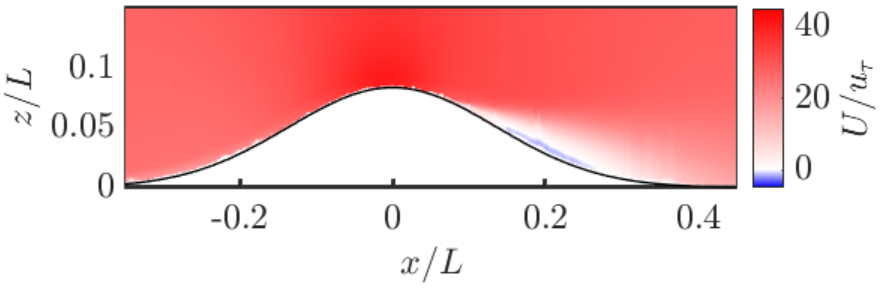


Figure 4.3: Mean (a) streamwise velocity, (b) wall-normal velocity, (c) Reynolds shear stress, (d) and in-plane TKE along at y/L = 0.083 for $Re_L = 3.44x10^6$.

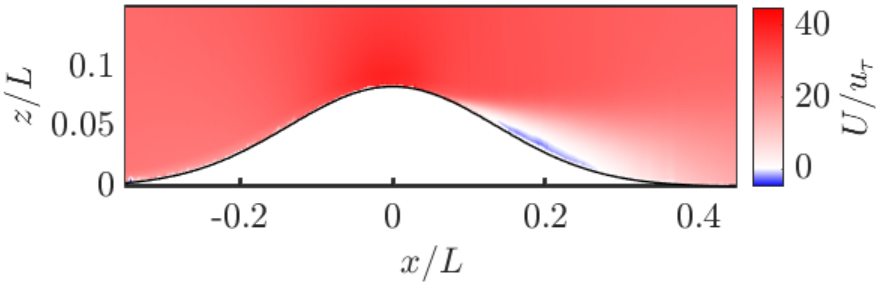
Williams et.al found the turbulent separated flow was found to be independent of Reynolds number above $Re_L = 2.55x10^6$ [4]. This is confirmed in Figures 4.4, 4.5, 4.6 and 4.7. Increasing Reynolds number promoted earlier separation, moving the mean separation point upstream along the spanwise plane y/L = 0.083, as shown in Figures 4.4a and b. The size and angle of the shear layer increased with Reynolds number but was similar for $Re_L \geq 2.55x10^6$, which can best seen in Figures 4.6 and 4.7. However, the region of maximum shear stress increased in size as Reynolds number increased past $Re_L = 2.55x10^6$. This suggests there are stronger turbulence producing motions that do not dissipate as quickly when the freestream Reynolds number increases in this separated flow. Similar trends were observed with the intensity of the in-plane turbulent kinetic energy in the separated region as well, shown in Figure 4.7.



(a)

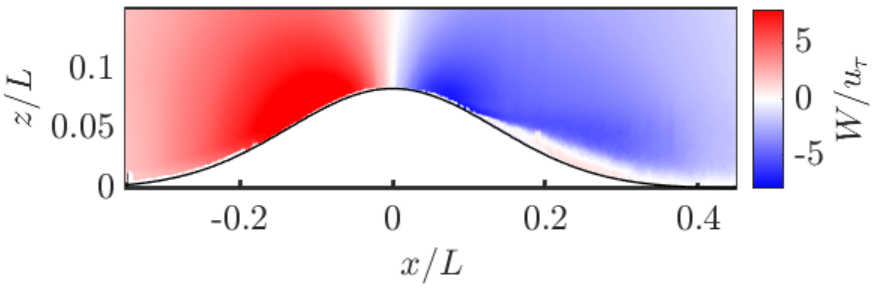


(b)

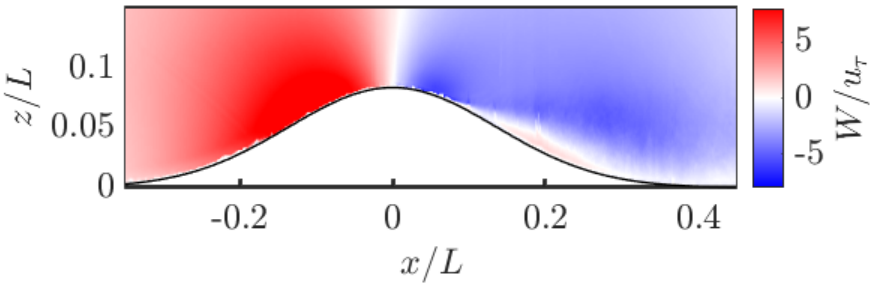


(c)

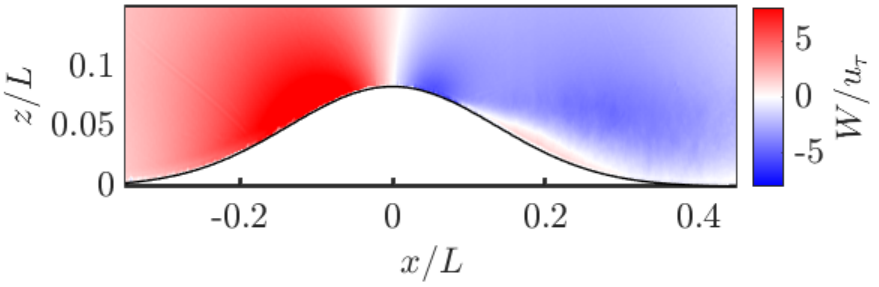
Figure 4.4: Mean streamwise velocity at y/L = 0.083 for (a) $Re_L = 1.39x10^6$, (b) $Re_L = 2.55x10^6$ and (c) $Re_L = 3.44x10^6$.



(a)

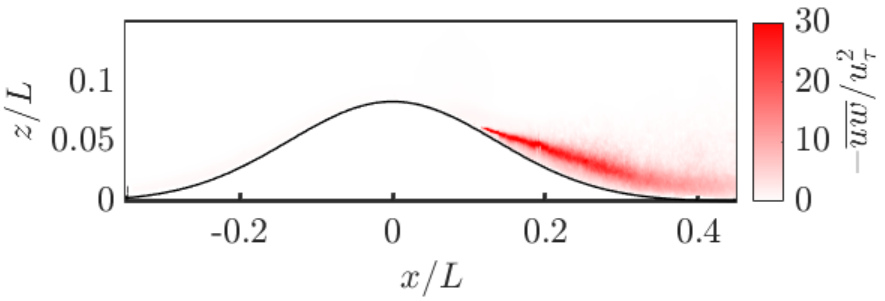


(b)

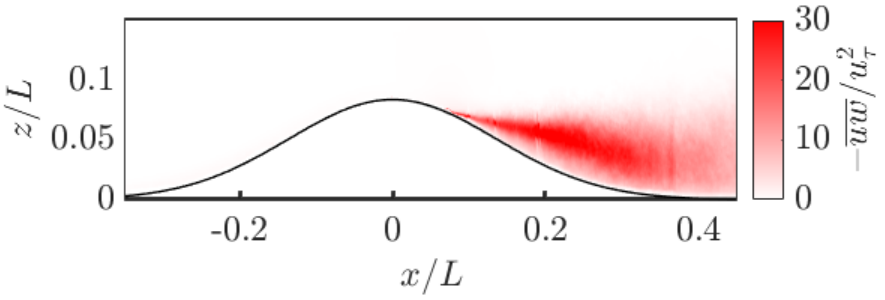


(c)

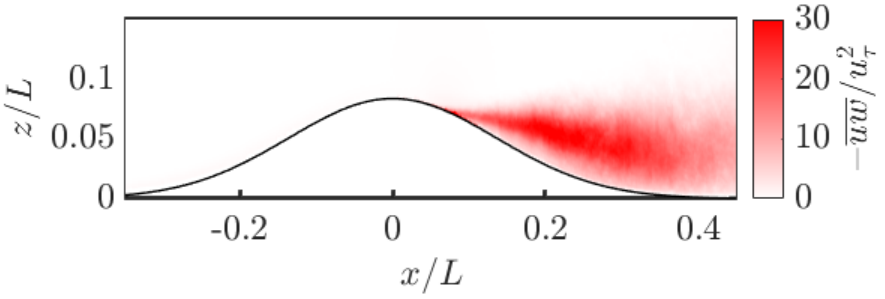
Figure 4.5: Mean wall normal velocity at y/L = 0.083 for (a) $Re_L = 1.39x10^6$, (b) $Re_L = 2.55x10^6$ and (c) $Re_L = 3.44x10^6$.



(a)

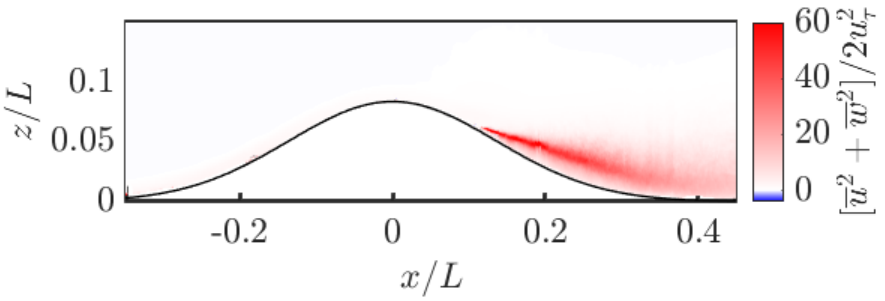


(b)

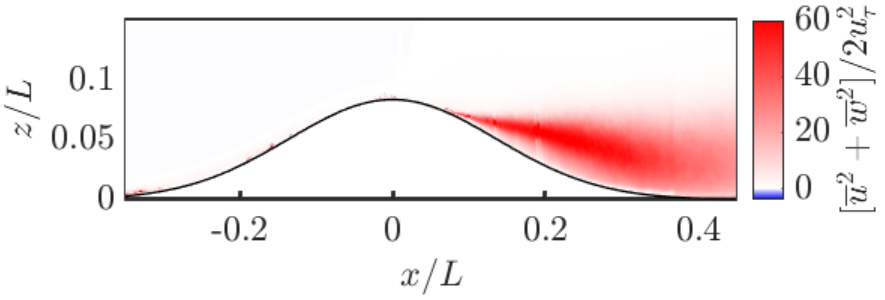


(c)

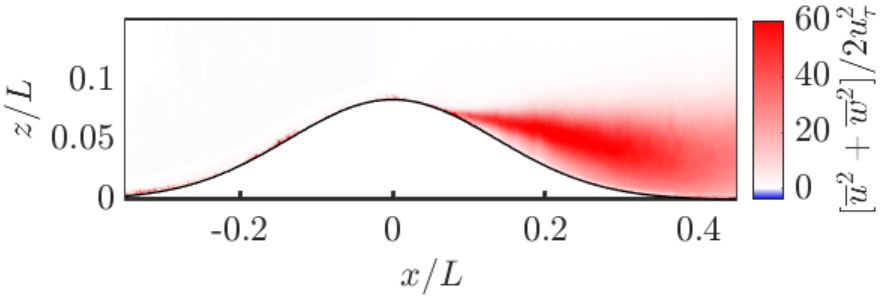
Figure 4.6: Mean shear stress at y/L = 0.083 for (a) $Re_L = 1.39x10^6$, (b) $Re_L = 2.55x10^6$ and (c) $Re_L = 3.44x10^6$.



(a)



(b)



(c)

Figure 4.7: Mean in-plane turbulent kinetic energy at y/L = 0.083 for (a) $Re_L = 1.39x10^6$, (b) $Re_L = 2.55x10^6$ and (c) $Re_L = 3.44x10^6$.

The shape of the separated region, especially the separation and reattachment points and the vertical extent of separation, are found to be heavily dependent on spanwise location. This is believed to be due to the presence of the spanwise vortices on the leeward side of the

bump, depicted in Figure 4.8.

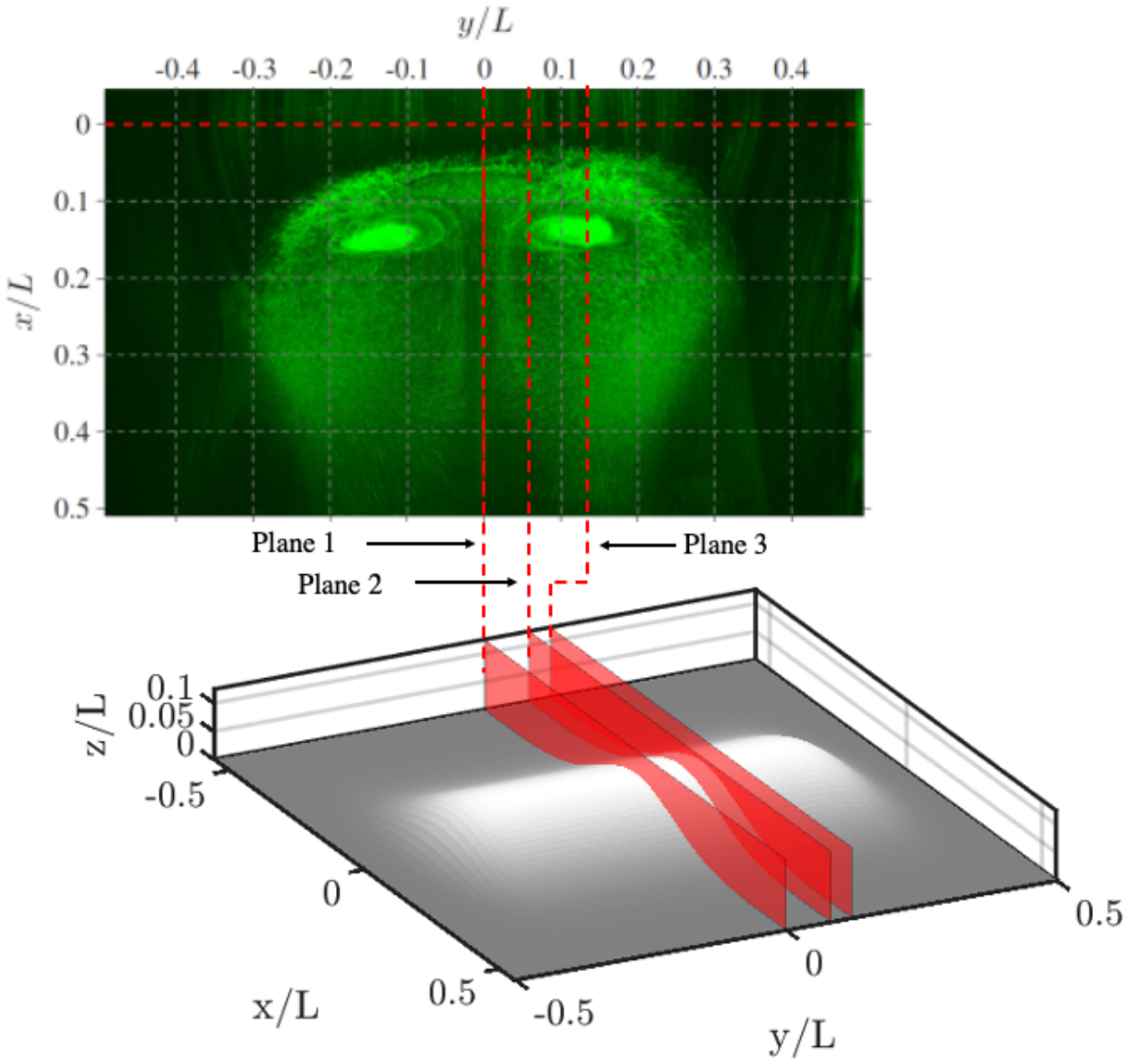
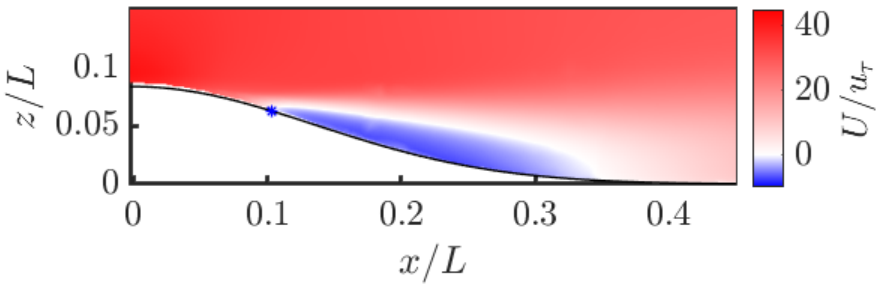
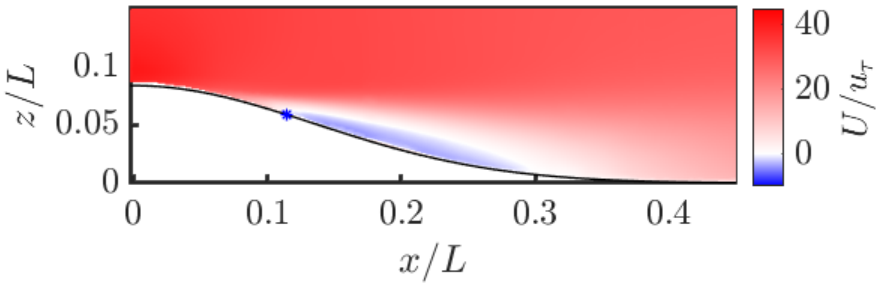


Figure 4.8: China clay visualization conducted by Williams et. al [11] of spanwise vortices downstream of separation on the leeward face of the bump at $Re_L = 3.44x10^6$. The flow is travelling from top to bottom. The red dashed lines correspond to planes located at y/L = 0, 0.083, and 0.125, labelled planes 1, 2 and 3, respectively, where the PIV data was acquired.

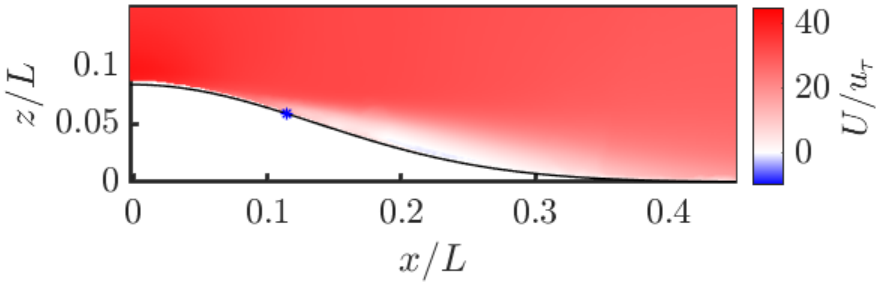
Composite statistical flowfields created from PIV obtained from using Setup 2 (refer Table 4.1) are presented at various spanwise locations. The location of the streamwise planes were adjusted between Setup 1 and 2 so the entire range of the separation point could be captured in one PIV frame. The spanwise dependency is primarily due to the three-dimensionality introduced by the taper of the model geometry near the tunnel side-walls. Figure 4.9 shows the mean separation occurred at $x/L = 0.103$ along the centerline at $z/L = 0.063$. Off the centerline, the separation point and shear layer initiation appear to move further downstream to $x/L = 0.114$ and $z/L = 0.059$ in planes 2 and 3. The separation locations were determined by plotting the $U/u_{\tau 0} = 0$ contour and finding the intersection of the contour with the bump. The angle, size and intensity of the shear layer decreased as the the distance from the spanwise centerline increased, shown in Figures 4.10 and 4.11. The lower magnitude of Reynolds shear stress and delay in separation as y/L increases suggests the flow undergoes varying levels of separation in the spanwise direction. This creates a three-dimensional separation line along the bump surface. It is interesting to note the significantly larger region of positive wall-normal velocity at the centerline compared to other spanwise positions as seen in Figure 4.5. This suggests there is a three-dimensional region of upward moving flow from the surface into the separated zone that tapers away from the centerline.



(a)

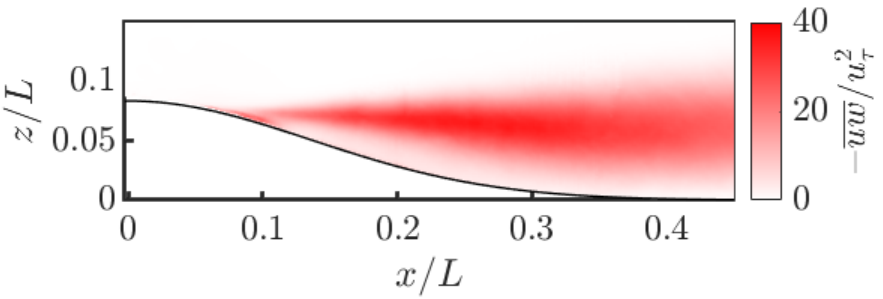


(b)

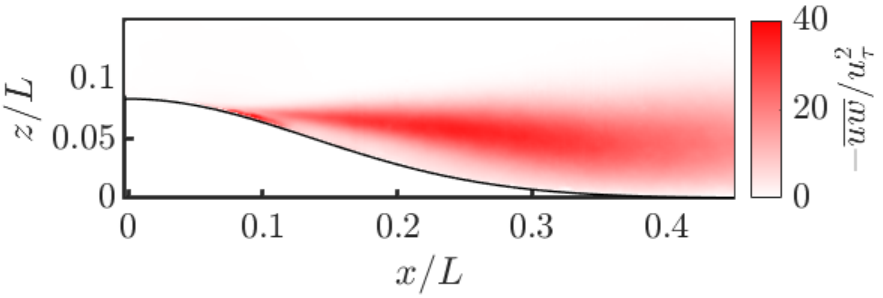


(c)

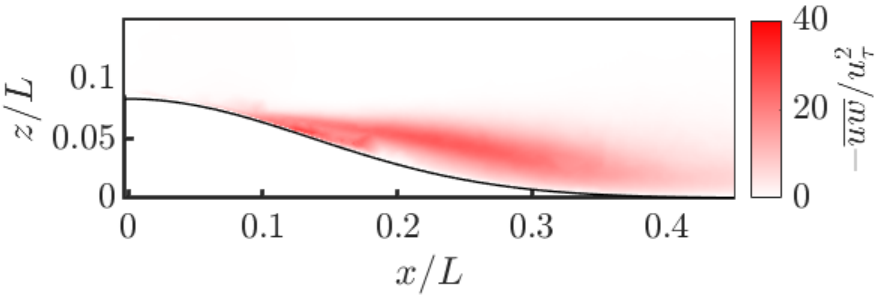
Figure 4.9: Mean streamwise velocity at (a) Plane 1, (b) Plane 2 and (c) Plane 3 for $Re_L = 2.56x10^6$. The marker roughly denotes the mean separation point.



(a)

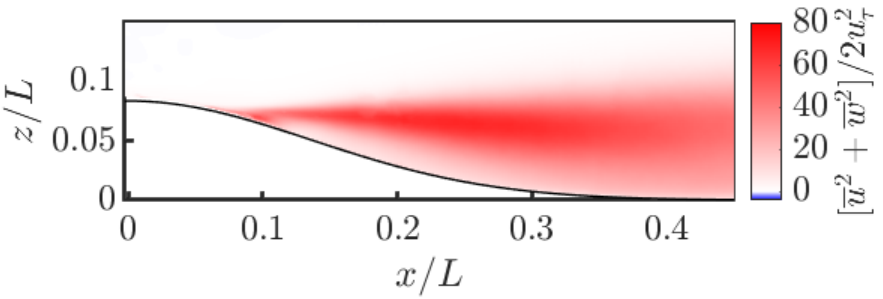


(b)

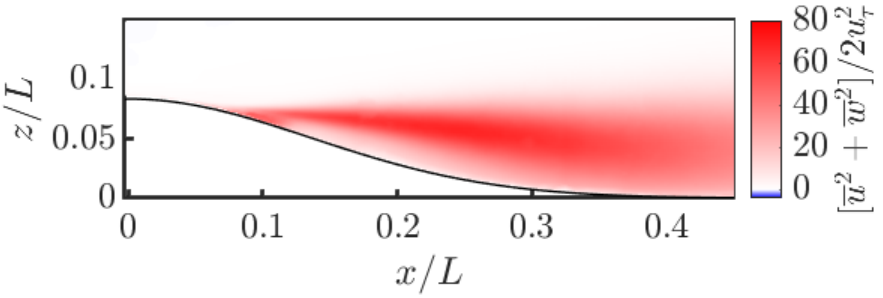


(c)

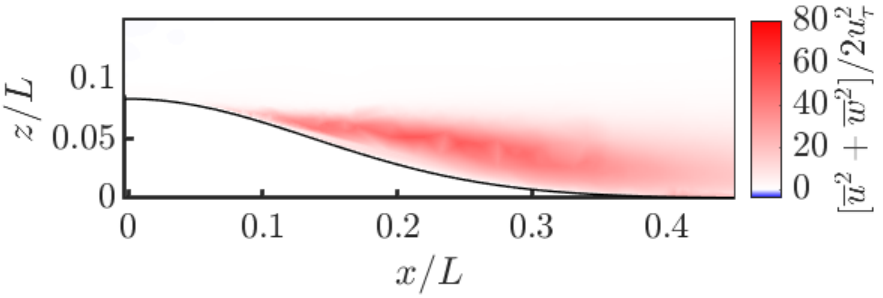
Figure 4.10: Mean turbulent shear stress at (a) Plane 1, (b) Plane 2 and (c) Plane 3 for $Re_L = 2.56x10^6$.



(a)

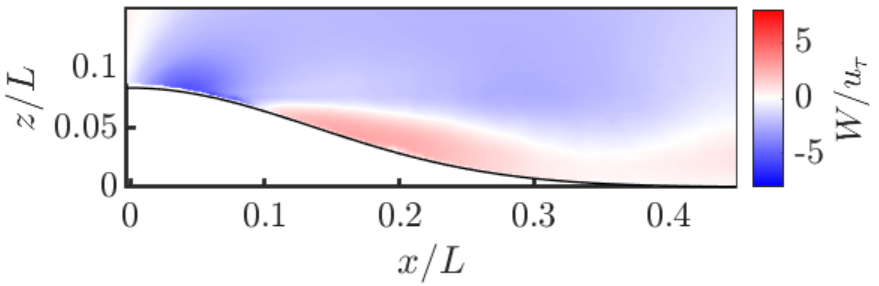


(b)

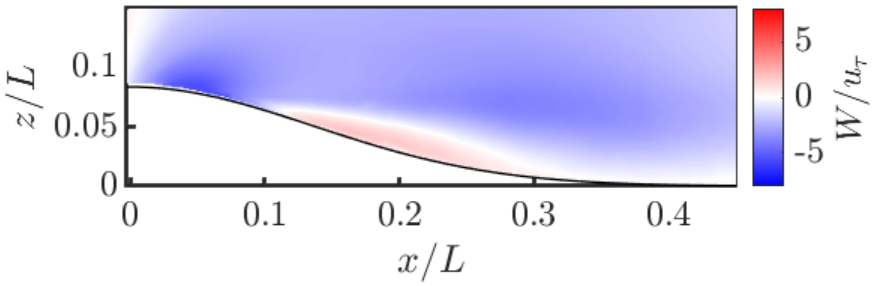


(c)

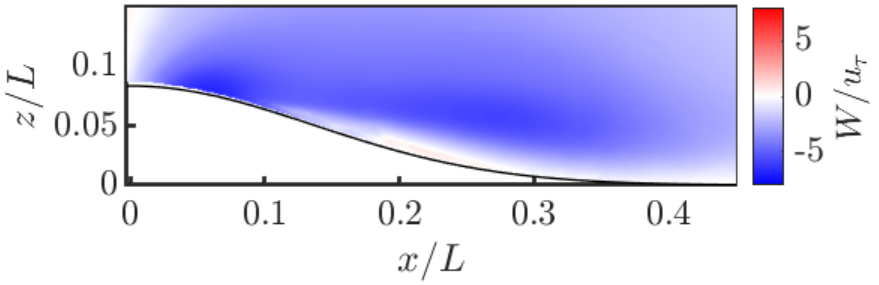
Figure 4.11: Mean in-plane TKE at (a) Plane 1, (b) Plane 2 and (c) Plane 3 for $Re_L = 2.56x10^6$.



(a)



(b)



(c)

Figure 4.12: Mean wall normal velocity at (a) Plane 1, (b) Plane 2 and (c) Plane 3 for $Re_L = 2.56x10^6$.

4.2 Boundary Layer Thickness Evolution in Presence of Simultaneously Varying Surface Curvature and Pressure Gradients

4.2.1 Surface Pressure and Curvature Interactions

The combination of varying pressure gradient histories and relative flow curvature profiles along the centerline of the bump, presents an opportunity to investigate the boundary layer evolution under the influence of both. It is known that the relative surface curvature has larger changes compared to the induced pressure gradient for test geometries with a larger upstream turbulent boundary layer relative to the bump height [3]. However, the timescale of the curvature effects is shorter than the large eddy turnover time of the boundary layer flow, making it unclear if the curvature influence is realized across the eddy and what effect it has. Therefore, there is a poor understanding of the non-equilibrium boundary layer flow which is undergoing many stabilizing and destabilizing influences.

Surface pressure surveys conducted by Samuell [12] and Robbins [10] indicated a series of favorable and adverse pressure gradients along the streamwise direction of the flow, as shown in Figure 4.13. Pressure initially increases as the flow encounters the retarding effect of the bump, after which mass continuity takes over and accelerates the flow over the bump apex, decreasing the surface pressure dramatically. The adverse pressure gradient is introduced after $x/L = 0$ and the pressure profile does not recover to the upstream reference pressure. The interchanging favorable and adverse pressure gradients were observed at all Reynolds numbers that were tested [11].

It has proven to be a challenge to define ubiquitous turbulent boundary layer length and velocity scales with changing surface curvature and pressure gradients. For this analysis, the boundary layer profiles were extracted from Figures 4.4a-c. Using the streamwise and wall-normal velocity fields, the spanwise vorticity, ω_y, was calculated using the Richardson extrapolation method to compute the gradients [46]. To ensure the wall-normal boundary layer properties were extracted, the local wall normal vector was calculated at each streamwise position along the wall. The boundary layer thickness is calculated following the method

proposed by Uzun and Malik [42] by choosing the location along the local wall-normal vector where the product $z_w\omega_y$ is less than 2% of $(z_w\omega_y)_{max}$. The variable z_w is the local wall normal distance. This definition was chosen as it is valid throughout the entire flow. The boundary layer edge velocity is calculated by taking the vector sum of the streamwise and wall-normal velocity at the location where $z_w\omega_y$ is less than 2% of $(z_w\omega_y)_{max}$. This closely follows calculating the edge velocity from integrating the spanwise vorticity presented by Spalart and Watmuff [47]. $(z_w\omega_y)$ was calculated near the wall across the entire composite PIV field and the global $(z_w\omega_y)_{max}$ was found to be 0.3843. Since the properties are extracted from PIV data, some uncertainty rises from the estimation of the maximum near-wall spanwise vorticity ω_y, which in turn affects $(z_w\omega_y)_{max}$. However, the profiles can still be used to understand the overall qualitative effects of pressure gradients and surface curvature on this flow. With this method, the boundary layer property profiles for $Re = 2.55x10^6$ compare very well to the DNS/LES performed by Uzun and Malik [13] at $Re = 2x10^6$, further corroborating the experimentally calculated boundary layer properties with the simulation results.

The evolution of the boundary layer thickness and the edge velocity magnitude in the streamwise direction are shown in Figures 4.14 and Figure 4.15. The local wall-normal boundary layer thickness δ_ω is normalized by the bump width L and compared across Reynolds numbers. As indicated in the aforementioned figures, all quantities experience significant changes as the flow moves over the bump. The incoming boundary layer encounters an adverse pressure gradient that initiates the growth of δ_ω/L shown at $x/L = -0.35$. The adverse pressure gradient strengthens towards the foot of the bump, causing U_e/U_∞ to decrease in this region, while δ_ω/L continues to rise. It is interesting to note the normalized boundary layer thickness in this initial adverse pressure gradient region decreases as the freestream Reynolds number increases. However, the edge velocity is highest for the lowest Reynolds number and the curves collapse onto each other for the higher Reynolds numbers.

The pressure gradient transitions to become strongly favorable from approximately $x/L = -0.2$ to near the bump apex at $x/L = 0$. The flow accelerates and U_e attains its maximum value of approximately $1.54U_\infty$ for $Re = 1.39 * 10^6$ and $1.5U_\infty$ for $Re = 2.55 * 10^6$ and

$3.44 * 10^6$. Simultaneously, δ_ω/L decreases in this favorable pressure gradient and collapses as it reaches a value of approximately 0.013. The pressure gradient switches back to adverse immediately after the apex and the flow rapidly decelerates, which leads to separation near $x/L = 0.1$, which is clearly indicated in Figure 4.14 as the lines diverge and rapidly increase. It is interesting to note the edge velocities also collapse after separation as shown in Figure 4.15. Downstream of separation, δ_ω/L continues to increase until it plateaus near the tail of the bump, where the pressure gradient becomes favorable, before continuing to increase again. This is because flow separation creates large structures that convect downstream, whose reattachment leads to the formation and evolution of another turbulent boundary layer. It should be noted that this method of calculating boundary layer thickness and edge velocity does not differentiate between separated and attached flows. However, the method provides great insight into the transitions undergone by the attached flow as it separates and then reattaches.

Previous work investigating the surface curvature effects on turbulent boundary layers has indicated that in the absence of pressure gradients, convex curvature has a destabilizing effect on turbulence by reducing Reynolds stresses and C_f ([26], [27]). Concave curvature has the opposite effect. However, in the case of the speed bump, there is varying surface curvature in the presence of alternating pressure gradients so the interaction between the two is investigated.

The evolution of δ_ω with respect to surface curvature R/L is shown in Figure 4.16 and compared across Reynolds numbers. Figure 4.17 illustrates the approximate regions of the adverse/favorable pressure gradients and concave/convex curvature acting on the boundary layer. As the pressure gradient and curvature change sign and strength, the boundary layer is pushed out of equilibrium and creates intricate internal layer dynamics. On the initial upstream slope, the combination of the adverse pressure gradient, which reduces C_f, and concave curvature, which increases C_f, causes an increase in δ_ω and reduction in C_f as seen in Figure 4.18. This indicates the adverse pressure gradient seems to overpower the effects of concave curvature on the upstream slope for this geometry. However, the weakening of

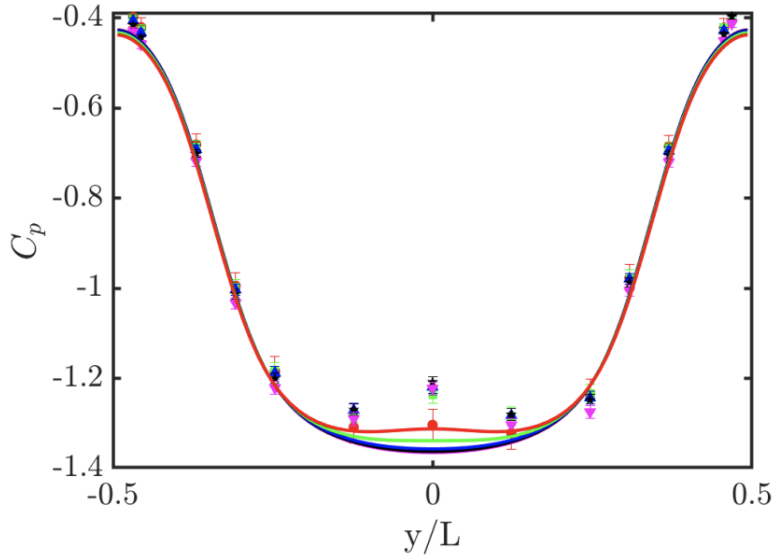
the adverse pressure gradient and increase in surface curvature causes C_f to plateau near $x/L = -0.32$ and increase thereafter, which is expected.

The pressure gradient then transitions to favorable, which increases C_f independently, while the surface curvature remains concave ($-0.2 < x/L < -0.135$). In this region, both seem to work together to reduce the rate at which the δ_ω grows and cause it to plateau, as the pressure gradient reduces flow deceleration. As expected, C_f rapidly rises. The surface curvature then switches to convex, which destabilizes turbulence when acting independently. However, δ_ω slightly decreases from the plateau, reaching a minimum just upstream of $x/L = 0$, signaling that the favorable pressure gradient dominates in this region by accelerating the flow to steepen the near-wall velocity gradient and work against the opposing effect of the convex curvature. Indicated in Figure 4.18, C_f was also observed to increase until just upstream of the bump apex, further corroborating the argument that the pressure gradient dominates the convex curvature [13]. δ_ω begins to increase just upstream of the bump peak as C_f decreases as shown in Figure 4.18 [13], suggesting the combined outcome of the convex curvature affecting the flow as the favorable pressure gradient ends and the adverse pressure gradient appears. Ultimately, δ_ω continues to grow as the adverse pressure gradient strengthens under convex curvature and the turbulent boundary layer separates off the bump surface.

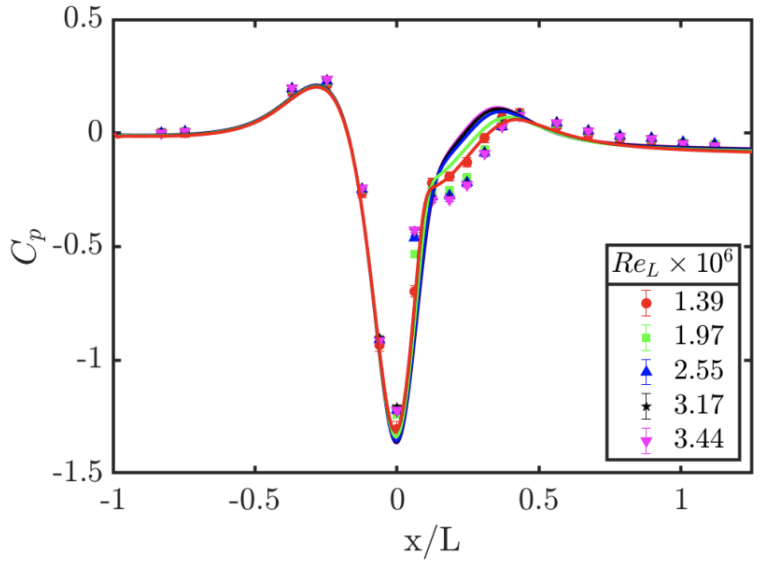
4.3 Proper Orthogonal Decomposition of Separated Region

Proper orthogonal decomposition (POD) of the separated region was performed to determine the presence of any dominant structures and motions and their contributions to the total energy of the flow. POD is a technique used widely throughout turbulent flow studies to decompose turbulent vector fields into a set of deterministic functions that capture portions of the total fluctuating kinetic energy in the flow. This allows for identification of coherent structures that populate flows, which are otherwise difficult to define and observe.

The fluctuating velocity in the flow is $\boldsymbol{u'}(\boldsymbol{x}, t)$, where $\boldsymbol{u'}$ is the velocity vector $\boldsymbol{U}$ minus the mean $\overline{\boldsymbol{U}}$. $\boldsymbol{x} = (x, z)$ is the position vector, $\boldsymbol{U} = (U, W)$ is the velocity vector and t



(a)



(b)

Figure 4.13: (a) Streamwise centerline and (b) spanwise along bump peak mean experimental surface pressure profiles at H/L = 0.5, for various Reynolds numbers from Robbins [10] compared with RANS simulations from Samuell [12].

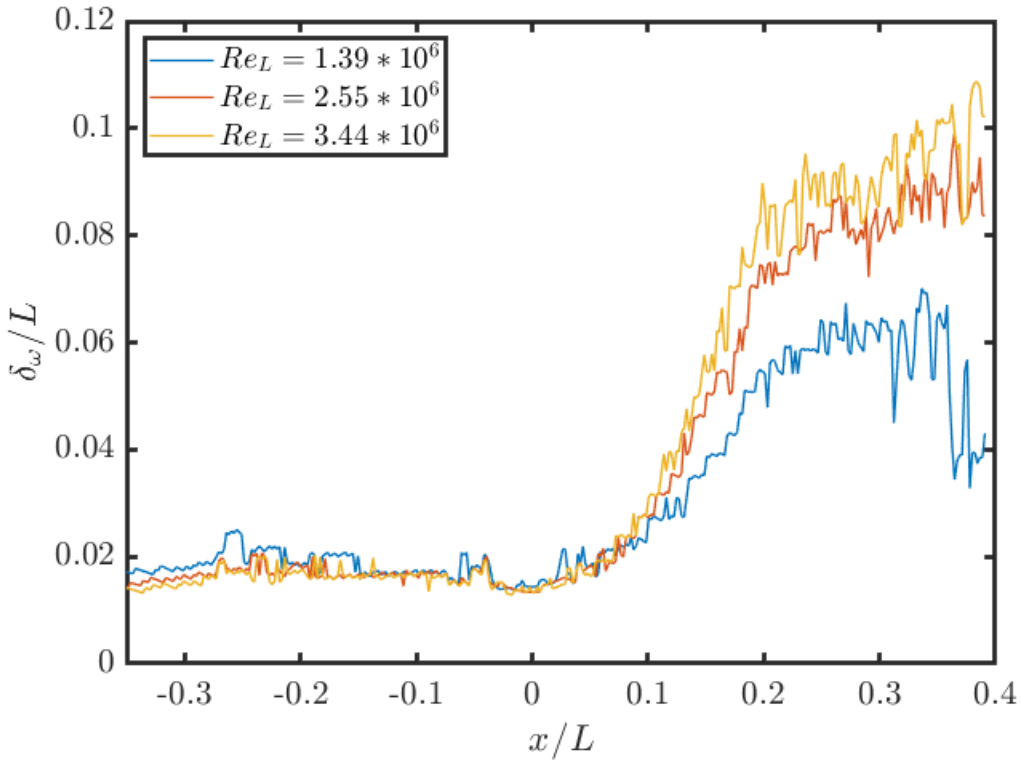


Figure 4.14: Boundary layer thickness evolution in streamwise direction based on vorticity thickness along centerline across increasing Reynolds numbers.

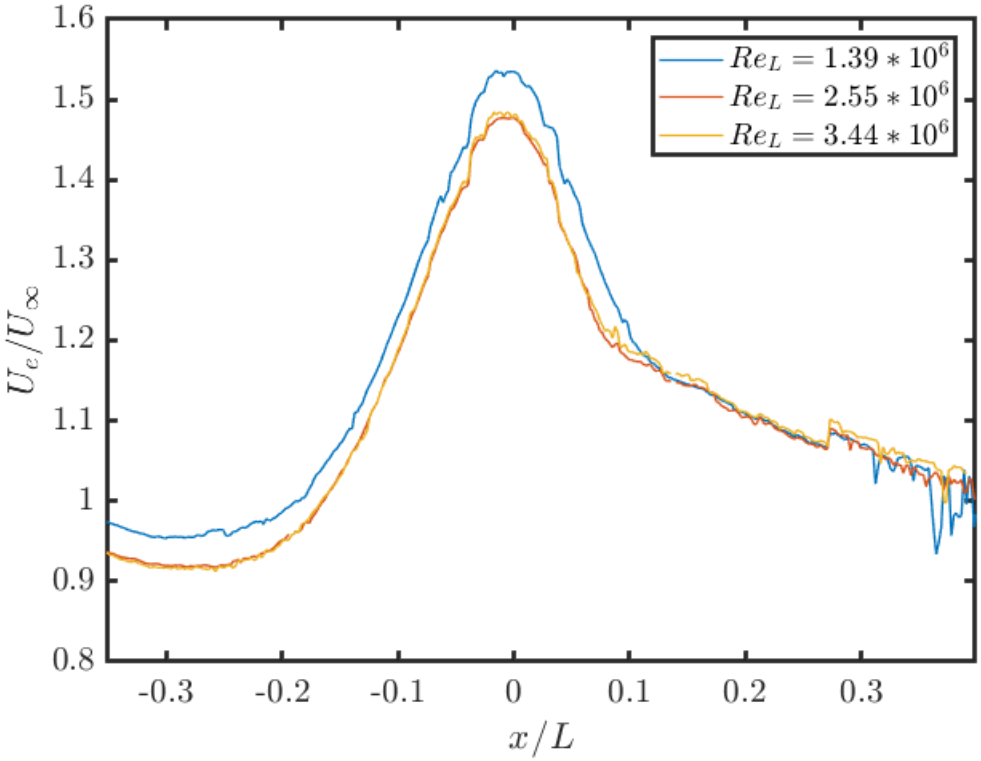


Figure 4.15: Velocity at edge of boundary layer in streamwise direction along centerline across increasing Reynolds numbers. Upstream of separation, when $Re_L = 1.39 * 10^6$, the edge velocity is higher when compared to the edge velocity when $Re_L = 2.55 * 10^6$ and $3.44 * 10^6$. However, after separation the edge velocities are independent of Reynolds number. The sudden increase in U_e near $x/L = 0.275$ for all Reynolds numbers may be due to some larger and faster moving structures located in the separated region.

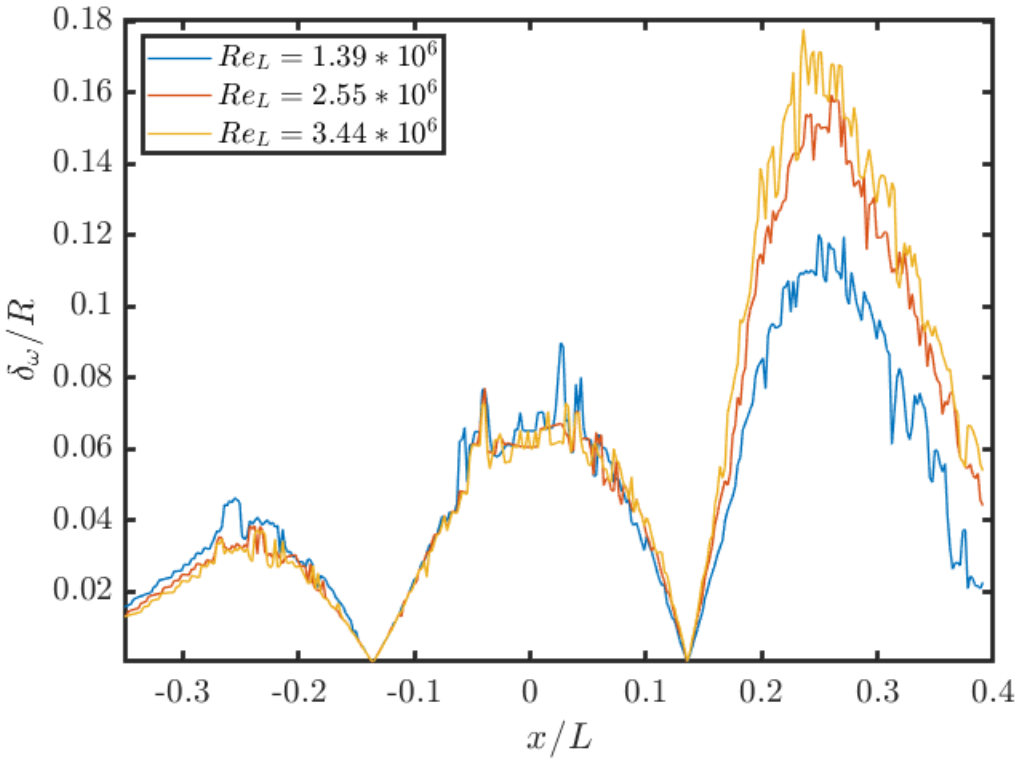


Figure 4.16: Boundary layer thickness evolution in streamwise direction based on vorticity thickness normalized against streamwise curvature along centerline across increasing Reynolds numbers.

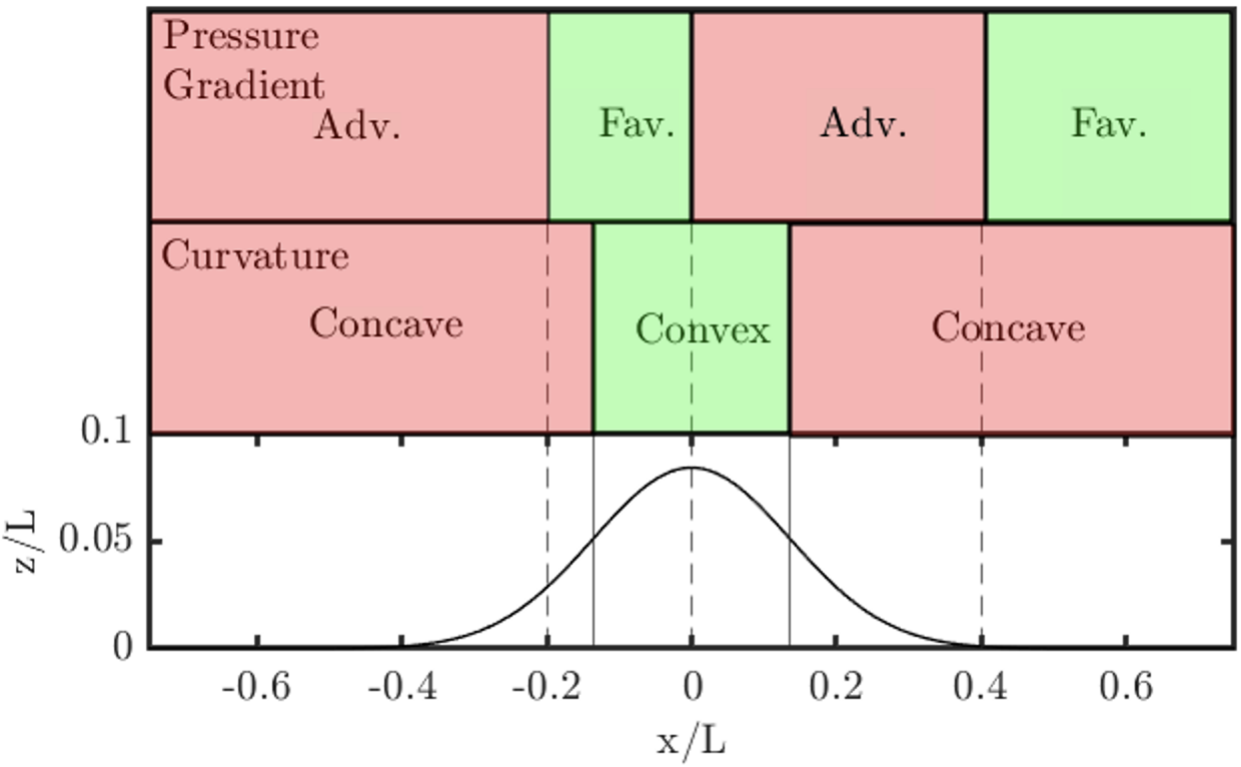


Figure 4.17: Schematic of simultaneously changing pressure gradients and surface curvature influences in streamwise direction over speed-bump.

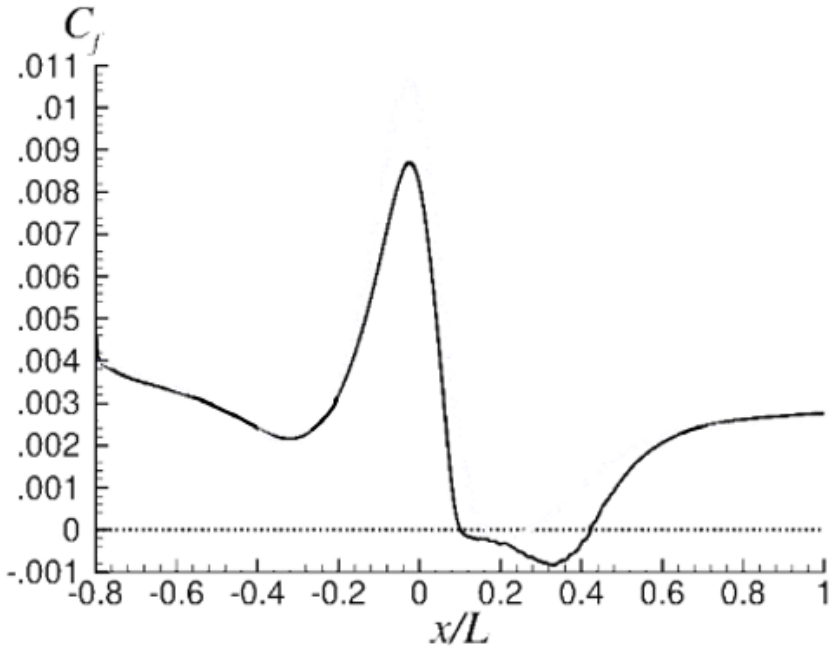


Figure 4.18: C_f in streamwise direction calculated by Uzun and Malik for $Re_L = 2x10^6$ with DNS/LES [13].

is time. The goal of POD is to decompose $\boldsymbol{u'}(\boldsymbol{x},t)$ into a set of spatial functions $\boldsymbol{\phi_k(x)}$ modulated by time coefficients $a_k(t)$ such that:

$$\boldsymbol{u'}(\boldsymbol{x},t) = \sum_{k=1}^{\infty} a_k(t)\boldsymbol{\phi_k(x)} \tag{4.1}$$

The $\boldsymbol{\phi_k(x)}$ are the POD spatial modes and $a_k(t)$ are their time coefficients [48].

For practical applications of POD towards large dimensional datasets of size n, the set of m fluctuating velocity fields is summarized into one mxn matrix $\boldsymbol{U}$ where each row represents one snapshot measured by the PIV system. In other words, each element (u_{ij}) of $\boldsymbol{U}$ is the velocity fluctuation at point j at time i. The eigenvalues and eigenvectors of the covariance matrix $\boldsymbol{C} = \frac{1}{m-1}\boldsymbol{U}^T\boldsymbol{U}$ are then calculated and the eigenvalues are ordered from largest to smallest. The n eigenvectors are arranged as columns in an nxn matrix $\boldsymbol{\phi}$:

$$\phi = \begin{pmatrix} \phi_{11} & \dots & \phi_{1n} \\ \phi_{21} & \dots & \phi_{2n} \\ \vdots & & \vdots \\ \phi_{n1} & \dots & \phi_{nn} \end{pmatrix} \tag{4.2}$$

and are the proper orthogonal modes of the dataset. Rearranging columns of $\boldsymbol{\phi}$ into the original dimensions of the PIV frames yields the spatial modes. The original dataset can also be projected onto each of the n modes to obtain the time coefficients by writing $\boldsymbol{A = U\phi}$. Rearranging this yields $\boldsymbol{U = A\phi^T}$, which can be applied towards the reconstructed PIV fields $\boldsymbol{\tilde{U}^k = A\phi^T}$ [48]. The dimensions of $\boldsymbol{\tilde{U}^k}$ are the same as $\boldsymbol{U}$ such that:

$$\boldsymbol{U} = \sum_{k=1}^{n} \boldsymbol{\tilde{U}^k} \tag{4.3}$$

Equation 4.3 implies the original velocity fluctuations are decomposed into a sum of n contributions from n proper orthogonal modes [49]. This is the finite-dimensional equivalent to the original infinite-dimensional POD theorem in equation 4.1. Various characteristics of the modes contributing the largest amount of energy to the flow can be explored and this is

one of the reasons why dimensionality reduction is very powerful in turbulent flows.

For this analysis, 40,000 frames of streamwise and wall-normal velocity calculated from PIV focused on the leeward side of the bump were initially used. However, convergence was found up to 25 modes with using just 25,000 frames. This was determined by running the POD analysis on increments of 5000 frames of PIV data (e.g 5000, 10000, 15000, etc.) and visually comparing the shapes of the streamwise and wall-normal velocity modes. It should be noted the mean streamwise and wall-normal velocities were subtracted from the fields, resulting in only the fluctuations being analyzed. However, it was found that the first 25 modes accounted for 73% of the total fluctuating energy across all number of frames tested, after which each succeeding mode only contributed less than 0.4% to the total fluctuating energy. Figure 4.19 shows the energy of each POD mode calculated from using 25,000 PIV frames containing streamwise and wall-normal velocity data.

The scaled mode energies quickly fall off after the first mode, which contributes 32.3% to the total fluctuating flow energy. This decreases to 7.9% for the second mode, 5.4% for the third mode and so on. This indicates there is a high energy mode that accounts for nearly 1/3 of the total separated flow energy that dominates the flow field. The following modes still contain high energy but their contribution is significantly less, suggesting other structures or motions are present in the separated region.

The first three POD modes of the streamwise and wall normal velocity fields in the separated region, denoted as ϕ_u and ϕ_w, respectively, are shown in Figure 4.20. In the case of the first mode, ϕ_u^1 and ϕ_w^1 are of uniform sign and seem to represent the low frequency contraction and expansion of the turbulent separated recirculation bubble. Streamlines indicate that this motion draws fluid from either upstream or downstream, bringing it up or down the bump face, and ejecting in either direction, suggesting a potential mechanism that controls the growth and decay of the separated region. This is similar to the POD results seen by Le'Floch et.al [25] and Taifour and Weiss [28]. In contrast to the first set of modes, the subsequent modes show a phase offset, which is indicative of convection of turbulent structures. Modes ϕ^2 and ϕ^3 seem to be paired and depict the convection of slow moving

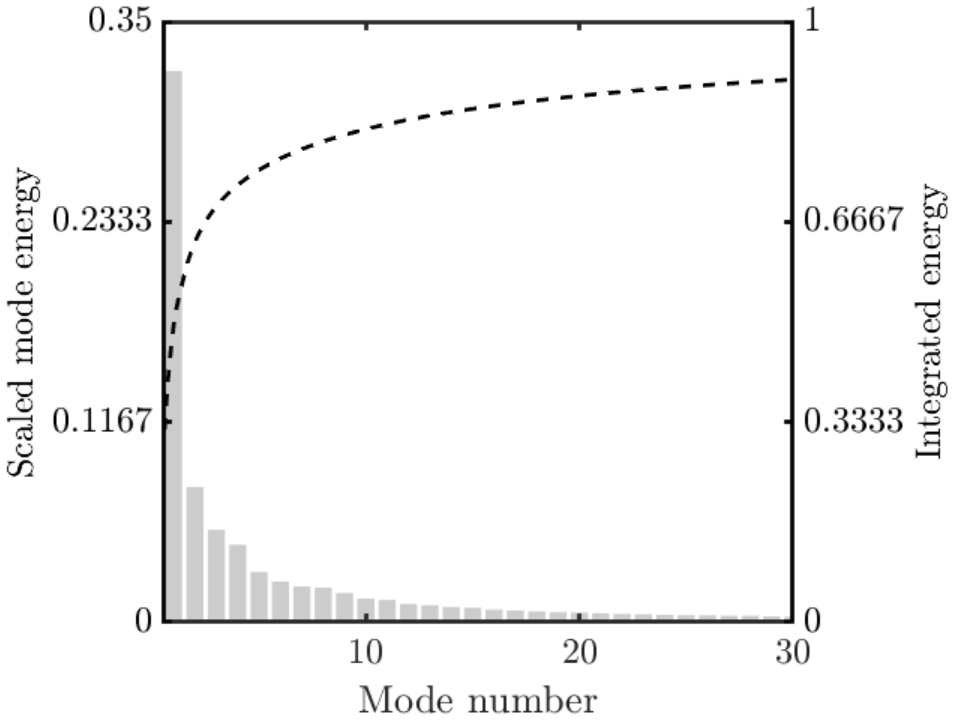


Figure 4.19: Energy of each mode from POD conducted on PIV frames focused along the spanwise centerline at $Re_L = 2.56x10^6$ in separated region. The scaled mode energy of the first mode is 32.3%, which then decreases to 7.9% for the second mode, and subsequently 5.4% for the third mode. The integrated energy is on the right vertical axis showing that just 19 modes accounts for 70% of the total flow energy.

vortical structures, shown by the motion of the large blue regions located most downstream in ϕ_u^2 and ϕ_u^3. Similarly, the large red regions in ϕ_w^2 and ϕ_w^3 located most downstream are offset, also indicating the slow convection of large vortices. The streamlines of ϕ^2 depict what may be the initiation of vortex roll-up off the bump surface as flow is drawn in from both the upstream and well above in the freestream outside the shear layer.

Modes ϕ^4 and ϕ^5, shown in Figure 4.21 also seem to be paired and appear to depict the convection of faster moving vortical structures compared to those in modes ϕ^2 and ϕ^3, which are offset by a smaller wavelength. The large energetic regions are shifted by what appears to be 1/2 of a wavelength between modes, which can be easily seen in ϕ_w^4 and ϕ_w^5. Streamlines in ϕ^4 suggest that after the vortices roll up off the bump surface, they could be feeding a region near the wall that is moving downstream. This region could also be drawing flow from the freestream above, potentially signalling a strong vortical structure that has already convected downstream that is pulling the flow downwards. However, this would have to be confirmed with looking at the mode activation of each frame. ϕ^6 did not appear to pair with any modes and shows interesting features that are not seen in any of the other modes examined. Out of the top 25 modes, the vertical striations of high energy regions in both ϕ_u^6 and ϕ_w^6 is unique. ϕ^6 looks very similar to a shear layer containing convecting vortices that appear to originate from the separation point with a recirculation bubble underneath, as indicated by the streamlines. This would explain the vertical striations in both the streamwise and wall-normal velocity modes. Plots of the higher order ϕ_u and ϕ_w can be found in Appendix C.

ϕ_{-u} and ϕ_w were multiplied together to obtain ϕ_{-uw}, which are the modes describing the predominant Reynolds shear stress motions. These are of interest because they are are related to turbulence production. Figures 4.22, 4.23, and 4.24 show the shear stress modes ϕ_{-uw}^1 to ϕ_{-uw}^{12}. A feature of interest noticeable across all modes except in ϕ_{-uw}^1 is that vortices and their cores are clearly identifiable by pairs of energy regions of opposite strength that converge at a central point. Vortices produce and contain velocity fluctuations so it is expected that they are prominently seen across majority of the modes. However, as previously stated, an

exception to this is ϕ^1_{-uw}, where the shear stress mode takes a shape very similar to ϕ^1_u and ϕ^1_w. ϕ^1_{-uw} indicates a strong region of $-uv$, suggesting the low frequency contraction and expansion of the separation bubble could be the largest turbulence producing mechanism in this flow.

4.4 Analysis of Separation and Reattachment Based on Reverse Flow Area Fraction

4.4.1 Area of Reverse Flow and Separated Region

In order to investigate the structure of the separated region and its response to upstream disturbances, instances of similar flow dynamics and behavior are grouped together to observe average flow behaviors. Conditional averaging is used for this purpose as it represents the best non-linear estimate of a quantity, given a specific event criteria [50]. The event criteria selected for this analysis of separated flows is the reverse flow area fraction at any instant in the PIV frame. For a velocity field with components $U(x,z,t)$ and $W(x,y,t)$ in a two-dimensional PIV field of view A at given time t, the reverse area flow fraction α is defined as

$$\alpha = \frac{\int_A H(U)dxdy}{A} \tag{4.4}$$

$$H(m) = \begin{cases} 0, & (U \geq 0 \\ 1, & (U < 0 \end{cases} \tag{4.5}$$

The set of all instants T for which $\alpha(t)$ has a value between two scalars [a,b] can then be expressed as

$$T_{[a,b]} = \{t | a < \alpha(t) < b\} \tag{4.6}$$

from which it can be determined that the conditional average of all $t \in T_{[a,b]}$ is

$$\langle U(t) \rangle_{t \in T_{[a,b]}} \tag{4.7}$$

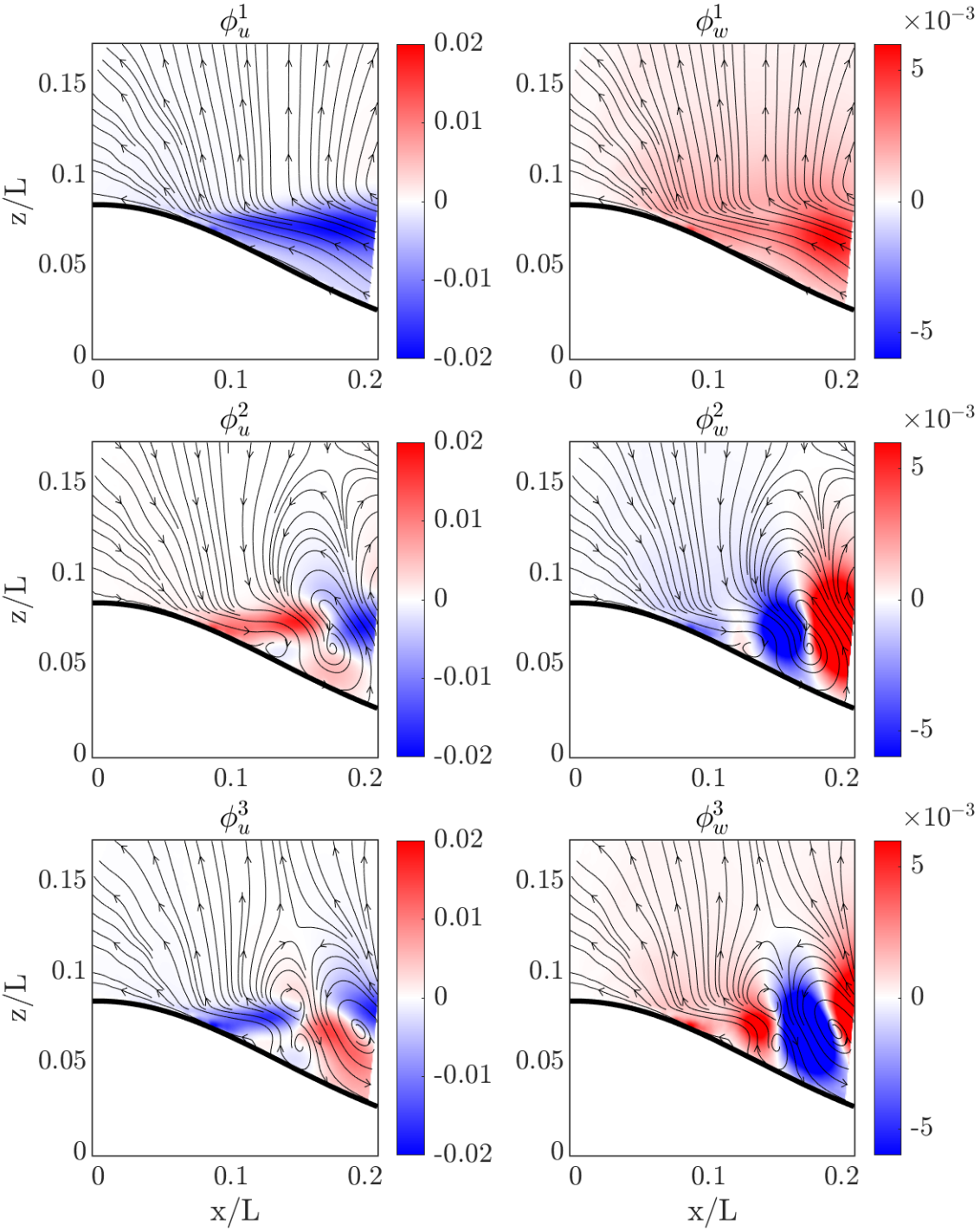


Figure 4.20: Composite figures of modes ϕ^1 to ϕ^3 of the streamwise, u, and wall-normal, w, velocity fluctuations. ϕ_u^1 and ϕ_w^1 suggest the presence of the separation bubble breathing mode. Modes ϕ^2 and ϕ^3 seem to be paired as they indicate convecting structures.

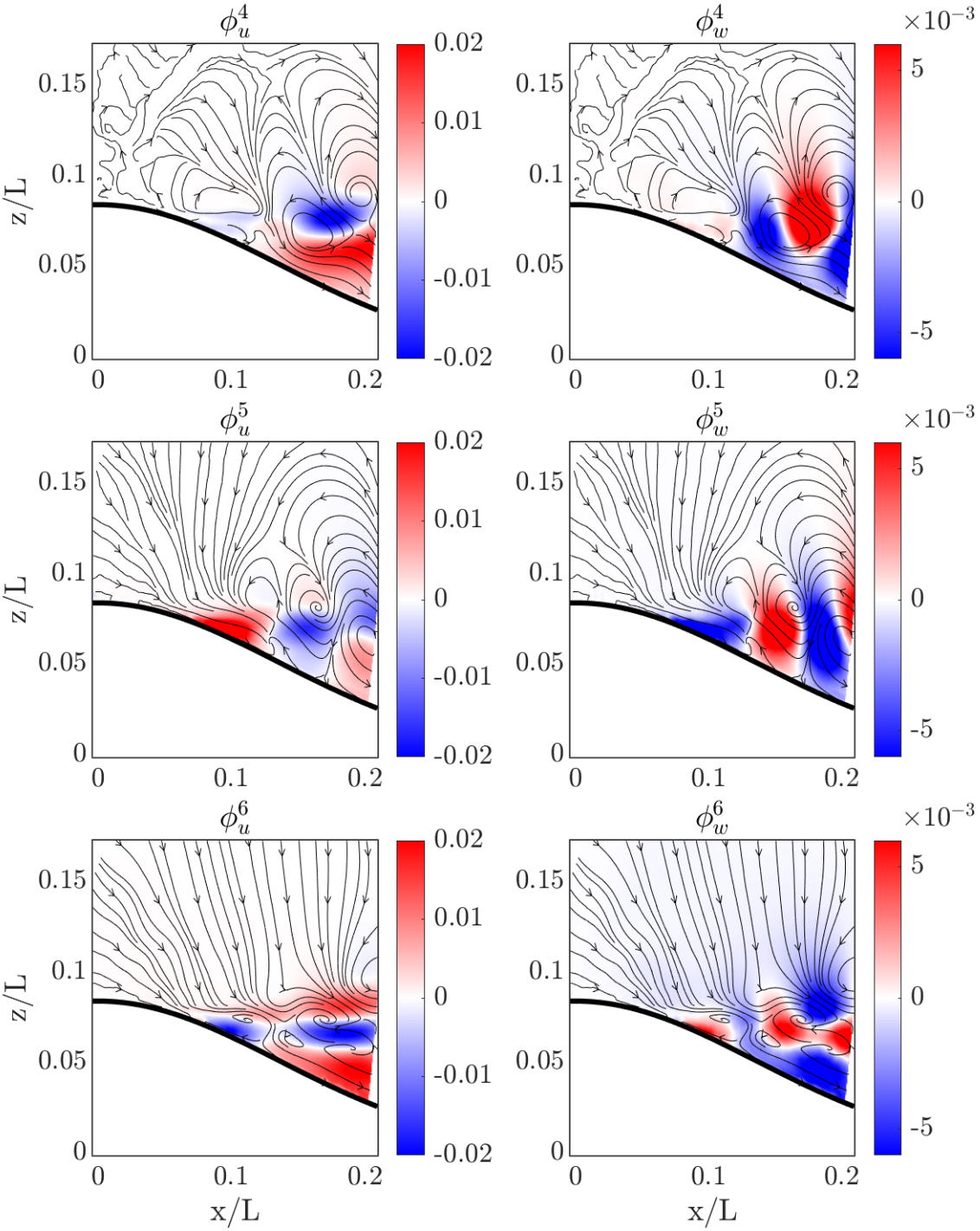


Figure 4.21: Composite figures of modes ϕ^4 to ϕ^6 of the streamwise, u, and wall-normal, w, velocity fluctuations. ϕ^4 and ϕ^5 seem to be paired as they indicate convecting structures. The vertical striations in ϕ_u^6 and vortical structures in ϕ_w^6 suggest the presence of a shear layer.

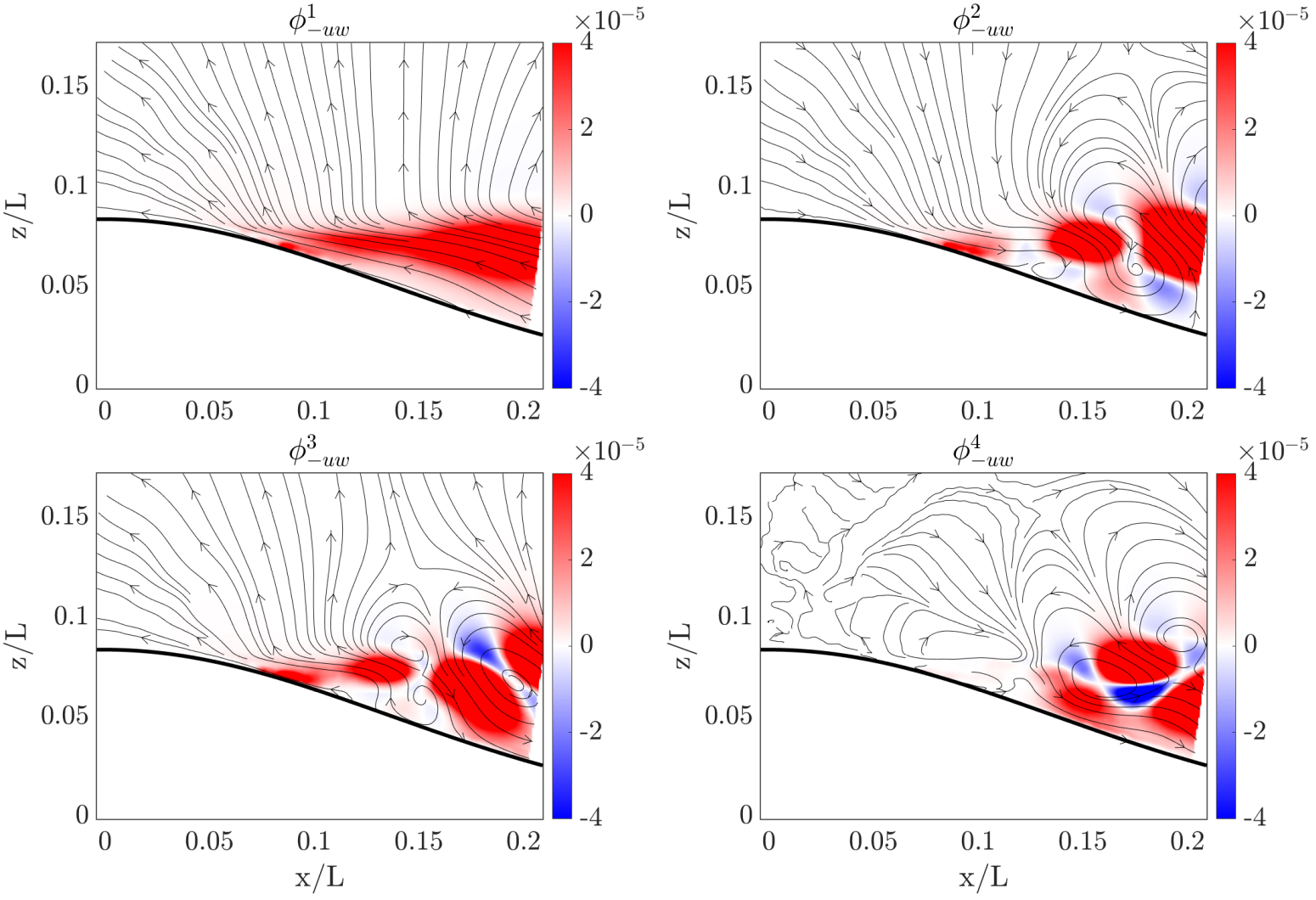


Figure 4.22: Shear stress modes ϕ^1_{-uw} to ϕ^4_{-uw} focused on the separated region.

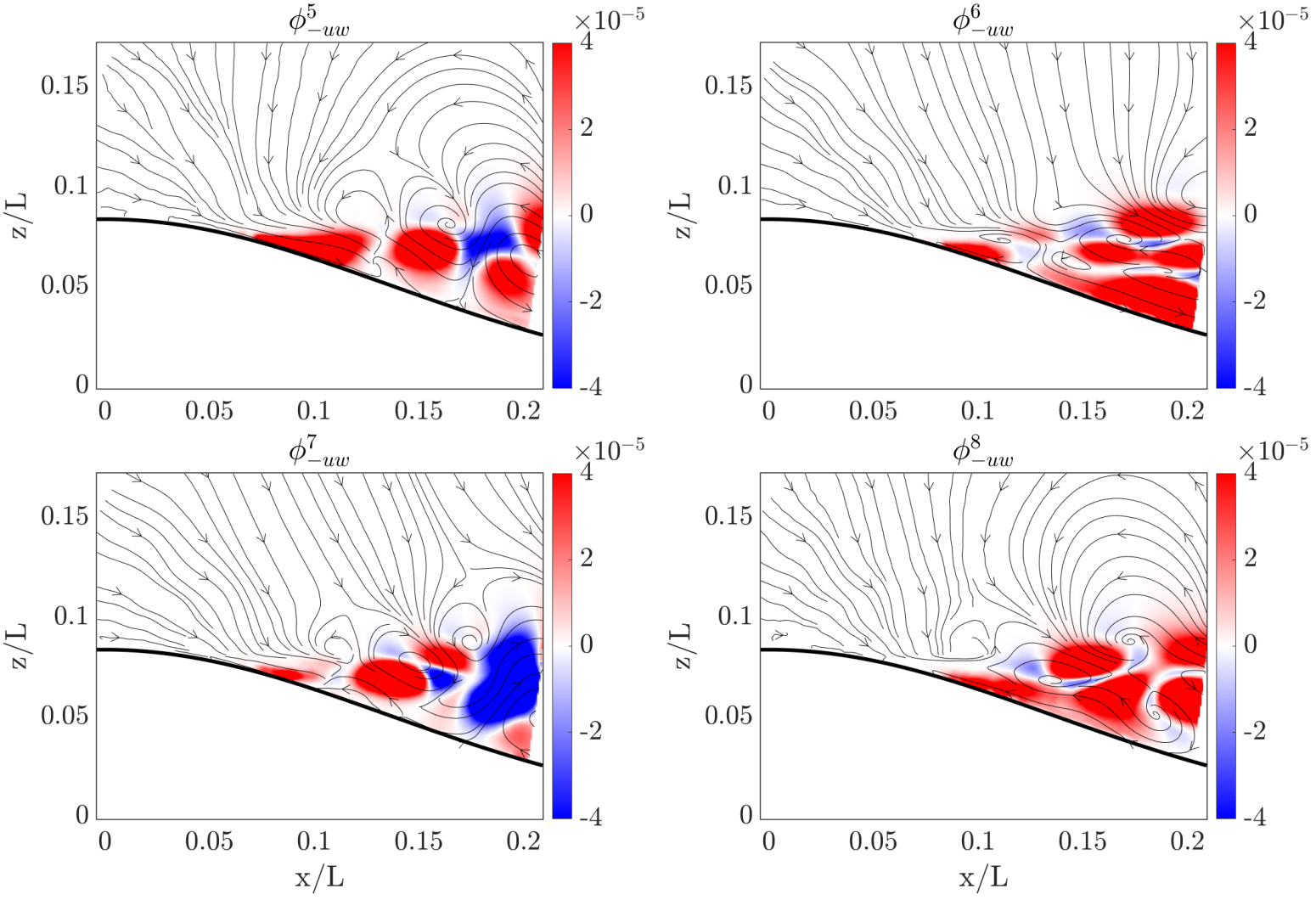


Figure 4.23: Shear stress modes ϕ^5_{-uw} to ϕ^8_{-uw} focused on the separated region.

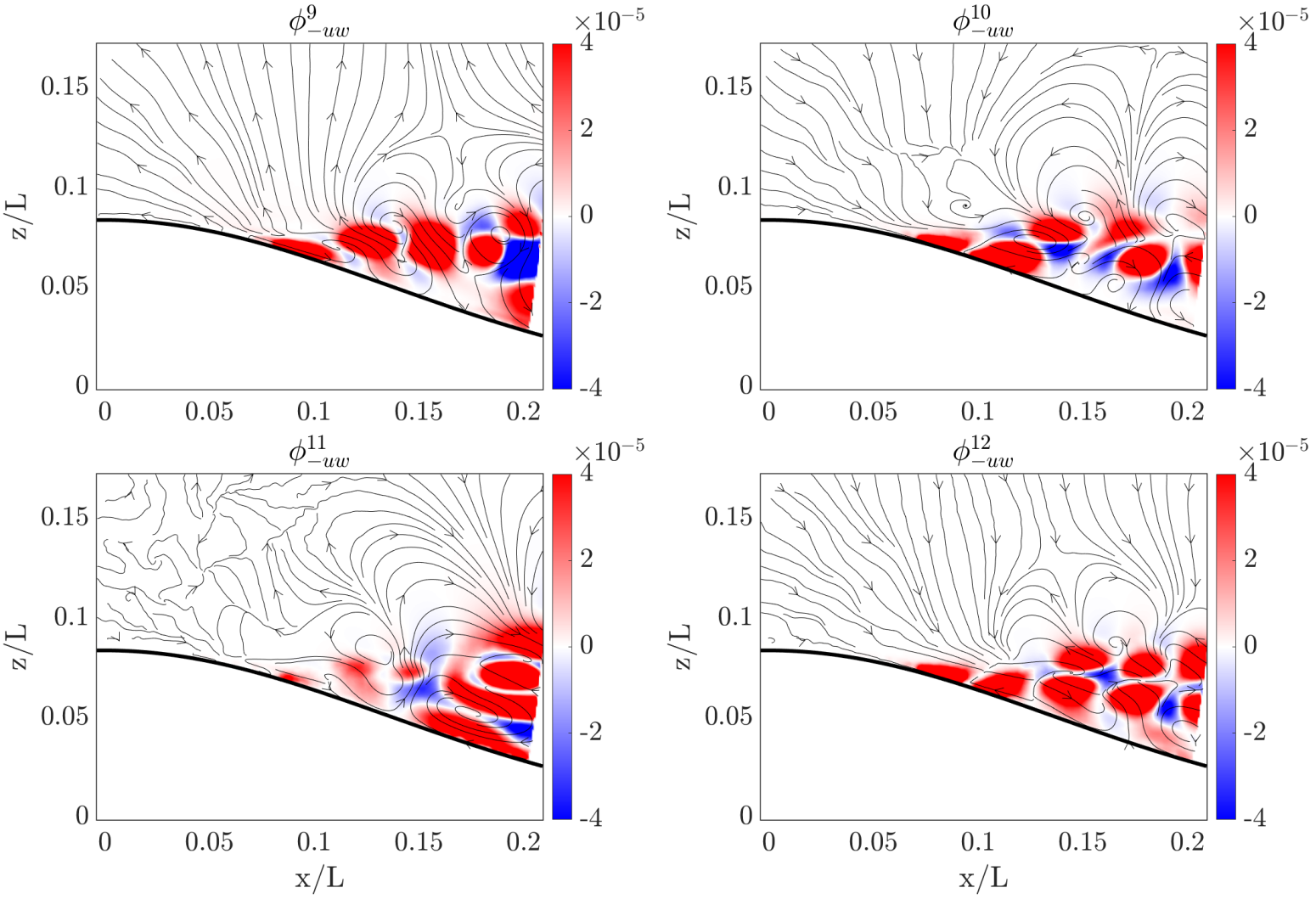


Figure 4.24: Shear stress modes ϕ^9_{-uw} to ϕ^{12}_{-uw} focused on the separated region.

where $\langle \cdot \rangle$ is the average calculated over all PIV fields that satisfy the given condition.

Figure 4.25 shows the probability density function (p.d.f) and cumulative distribution function (CDF) of α across 40,000 PIV frames taken along the bump centerline at Re = $2.56 * 10^6$. The p.d.f appears Gaussian in nature with the mean at $\alpha = 0.0766$ and $\sigma = 0.0219$. It is very slightly positively skewed with the median at $\alpha = 0.0772$. The maximum instance of reverse flow has an area equivalent to $0.172A$ and represents a large separation event occurring on the bump surface. The p.d.f. indicates there are zero PIV fields that show no reverse flow at all (i.e. $\alpha = 0$), suggesting that all frames contain boundary layer separation with regions of negative flow. To compare the effect of Reynolds number on the distribution functions, Figure 4.26 shows the p.d.f. and CDF of α across 10000 frames along the bump centerline at Re = $3.48 * 10^6$. The p.d.f. appears Gaussian in nature with a similar mean at $\alpha = 0.0792$ and $\sigma = 0.0251$. The maximum instance of reverse flow fraction was similar with an area equivalent to $0.165A$, and still represents a large separation event occurring on the bump surface. Similar to Figure 4.25, the p.d.f does not indicate any PIV fields with no reverse flow.

4.4.2 Statistical Determination of Separation Point Location

To learn more about the structure of separation at various α magnitudes and determine the separation points, it is useful to inspect the streamline pattern. While instantaneous turbulent velocity fields are very interesting, another avenue was explored to identify the dynamics governing separation. The streamlines of an averaged velocity field, conditioned over a small range of α were inspected and the separation locations were extracted.

Figure 4.27a shows the streamlines computed from the mean streamwise and wall-normal velocity fields, with the mean streamwise velocity field in the background for $Re_L = 2.56{*}10^6$. The white contour represents the $U/U_\infty = 0$ boundary, where within the contour, the flow is reversed. The separation region is clearly visible, with a large separation bubble, along with a shear layer with a steep velocity gradient. The streamlines indicate that in this mean field, flow is drawn up the bump face and rapidly turned around into the lower part of the

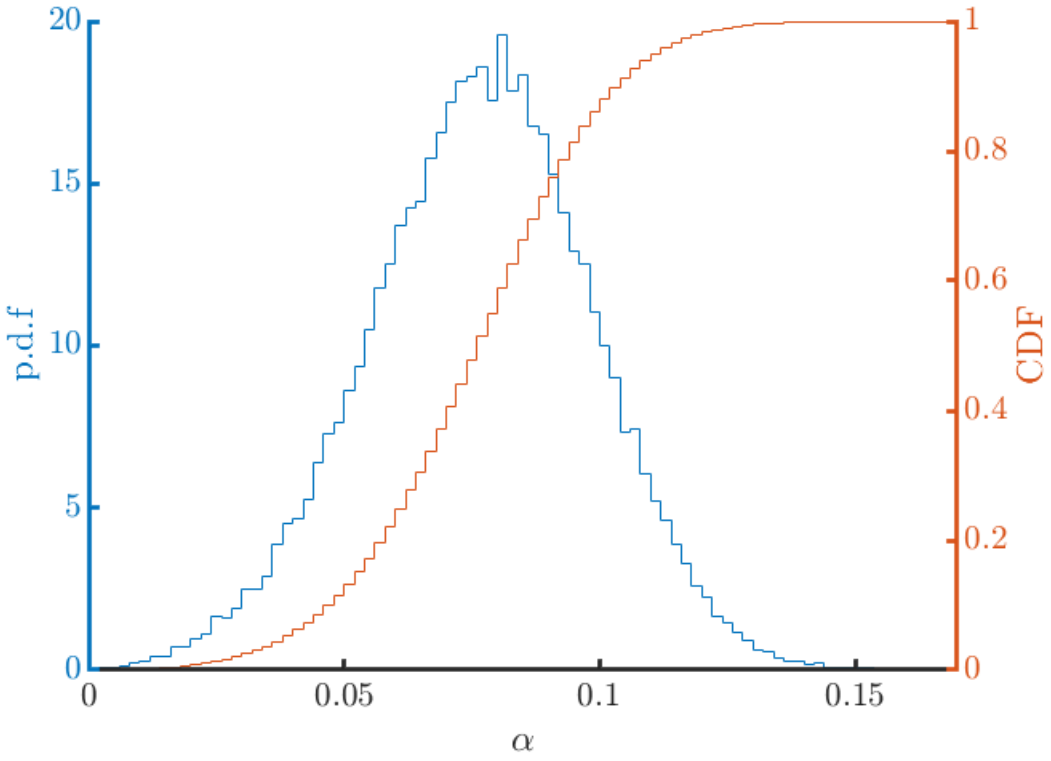


Figure 4.25: Probability density function (p.d.f) and cumulative density function (CDF) of α for $Re_L = 2.56 * 10^6$ focused on separated region.

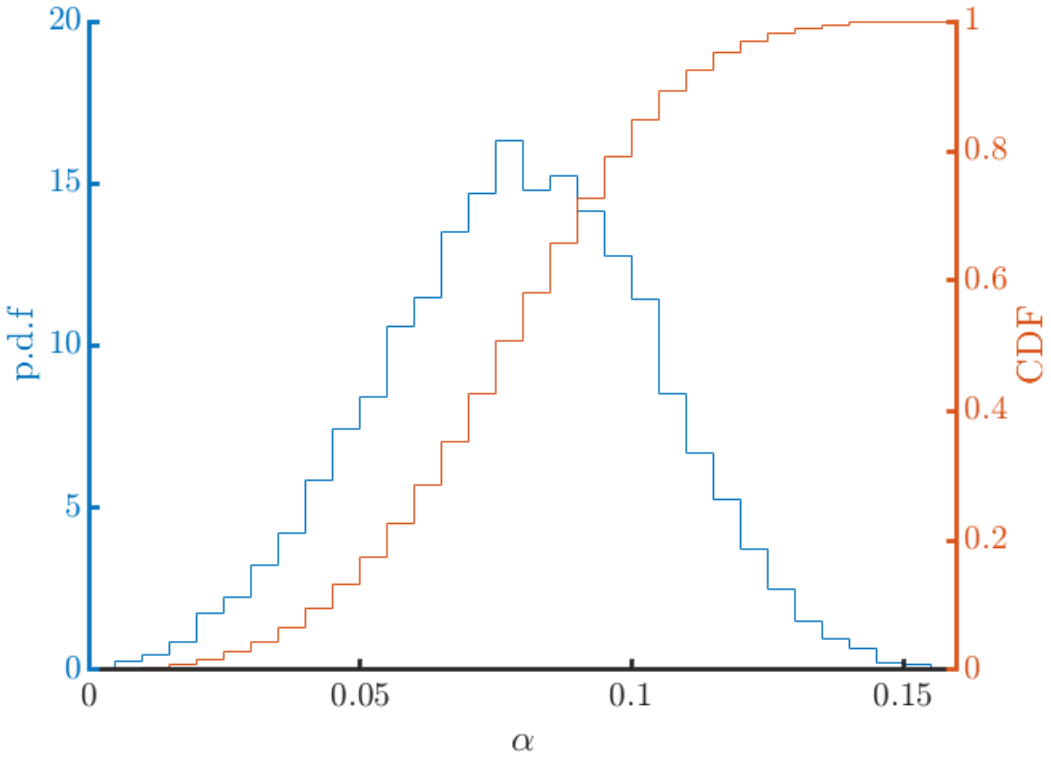


Figure 4.26: Probability density function (p.d.f) and cumulative density function (CDF) of α for $Re_L = 3.48 * 10^6$ focused on separated region.

shear layer. Due to limits on the extent of the field of view, the reattachment point is not visible on the bump surface.

The average streamlines of Figure 4.27a are sufficiently coherent that a direct approximation of the separation point is possible. However, due to the three-dimensional unsteadiness of this turbulent separated flow, identifying the exact separation point is not trivial. This difficult task is addressed in reviews by Simpson, which explains that in only steady two-dimensional flow does a turbulent separation point coincide with the classic definition: the location at which the wall shear stress, τ_{wall}, is zero [22]. Simpson instead proposed that a more reliable measure for turbulent flow is the fraction of time any wall location experiences reversed flow, denoted γ. So, to determine the location of separation in each of the conditionally averaged fields, the intersection of the $U/U_\infty = 0$ contour with the bump surface was used as a proxy due to near-wall acquisition difficulties with PIV. This corresponded to the location where $\gamma = 1$ in the mean. This yields a mean streamwise separation location of $x/L = 0.1047$ and wall normal separation location of $z/L = 0.0623$ when $\alpha_{mean} = 0.0766$.

Figure 4.27b depicts a low α case ($0.006 < \alpha < 0.008$) with very weak and delayed separation, relative to the mean. The small separated region appears to be very shallow to the surface in region that may be dominated by PIV uncertainty. The shear layer, whose angle with respect to the bump surface is smaller relative to the mean, extends down to the bump surface as majority of the flow has a positive streamwise velocity and moves downstream. The streamlines also curve downward indicating almost no positive wall-normal velocity, which compared to the mean streamlines in Figure 4.27a, indicate upward moving flow in the recirculation bubble.

Figure 4.27c shows the streamlines computed from conditionally averaging the streamwise and wall-normal velocity fields over the limits ($0.076 < \alpha < 0.078$). This bin contains α_{mean} and when comparing 4.27a, the streamlines and velocity fields are almost identical, although just only 1486 frames were used. The streamwise separation location for this bin is $x/L = 0.1044$ and wall normal separation location of $z/L = 0.0624$, signalling the robustness of the conditional averaging method.

Figure 4.27d depicts a high α case ($0.142 < \alpha < 0.144$) with strong and early separation, relative to the mean. There is a large separation bubble encapsulating a vast quantity of recirculating flow, only bounded by the separation point located on the bump surface in this PIV frame. The reattachment point is too far downstream to be captured simultaneously. The shear layer angle relative to the bump surface is much higher than that of the mean and the low α case. It does not extend to the bump surface as there is a large region of fast moving upstream flow beneath. The streamwise velocity of the flow in the recirculation bubble is also much higher relative to the mean separation case. This suggests there is a large amount of flow transported across the bump surface laterally, up the wall and into the shear layer, via strong spanwise vortices created by the tapered shoulders. This indicates that the spanwise vortices may vary in strength and affect the overall shape of separation with out-of-plane mass fluxes as demonstrated by Stuer et. al. [51].

As previously stated, the intersection of the $U/U_\infty = 0$ contour with the bump surface was defined to be the separation point and calculated automatically via MATLAB. Figure 4.28 is the pdf of streamwise separation points for $Re_L = 2.56 * 10^6$, along the bump centerline. The pdf is nearly Gaussian and positively skewed with a few separation locations of very little probability density located far downstream. The mean separation across the 40,000 PIV frames is located at $x/L = 0.1047$ with a standard deviation $\sigma = 0.0075$. To characterize the unsteady separation point motion, the CDF indicates that 95% of the separation points are located $\pm 2\sigma$ from the mean $x/L = 0.1047$. There are significant deviations from this region on the bump surface but those only account for 5% of the data.

The separation locations were plotted on the bump surface to serve as a visualization of the pdf, as shown in Figure 4.29. The points are colored based on the probability of separation occurring at that location. It is very evident from Figure 4.29 that majority of the separation point motion is centered about the mean as indicated with the dark red and orange points for $Re_L = 2.56 * 10^6$. There are points that extend as far downstream as $x/L = 0.2$, but these are likely to be very influenced by near-wall PIV uncertainty so they may not be true separation locations.

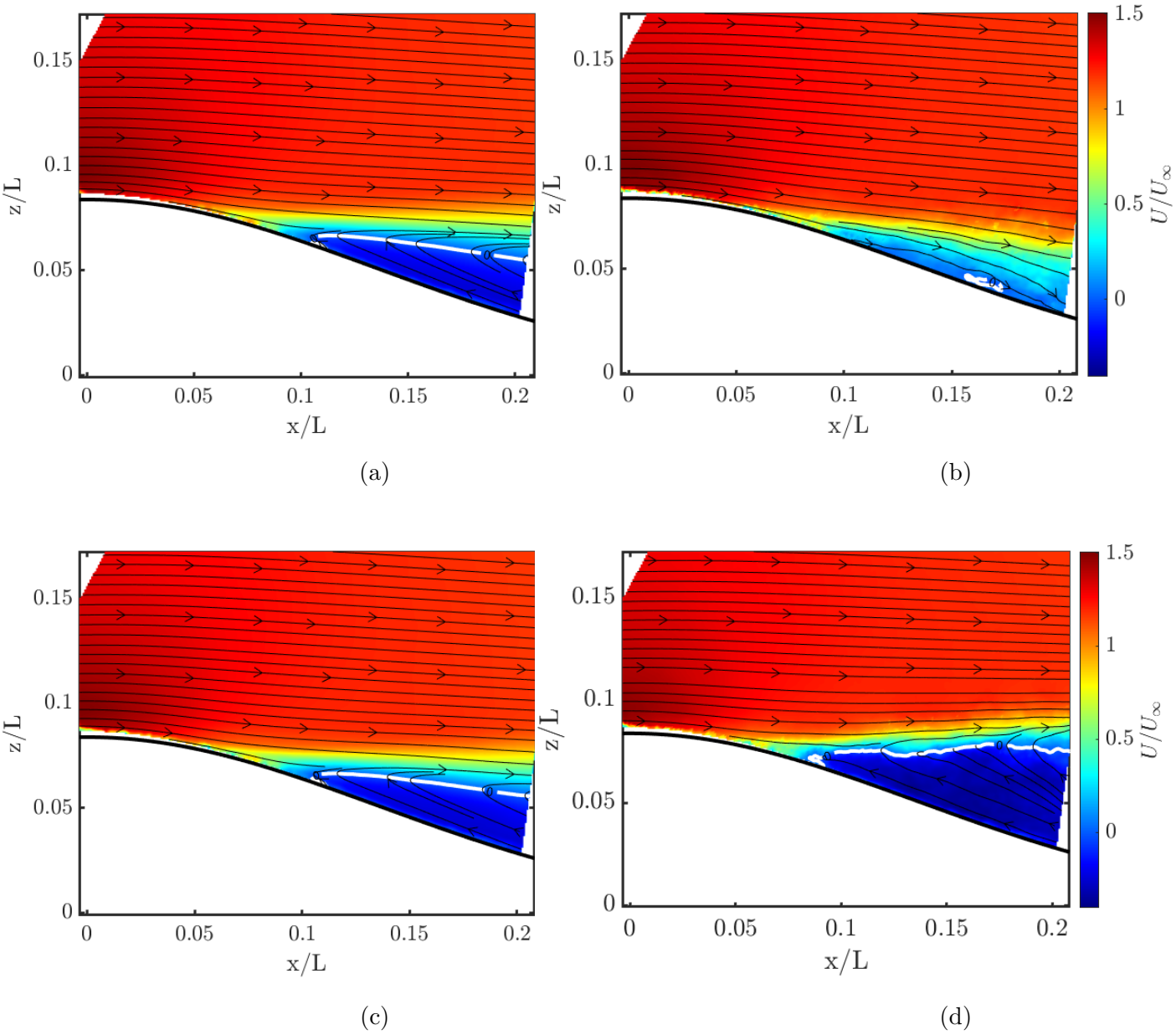


Figure 4.27: Streamlines and $U/U_\infty = 0$ contours overlaid on streamwise velocity fields for conditional averages of the flow based on α. As the streamwise position of contour moves upstream, different characteristics of the conditionally-averaged separation are observed: (a) streamlines for mean streamwise and wall-normal velocity field, containing 40,000 vector fields; (b) streamlines for bin $0.006 < \alpha < 0.008$, containing 11 vector fields; (c) streamlines for bin $0.076 < \alpha < 0.078$, containing 1486 vector fields; (d) streamlines for bin $0.142 < \alpha < 0.144$, containing 16 vector fields, at $Re_L = 2.56 * 10^6$.

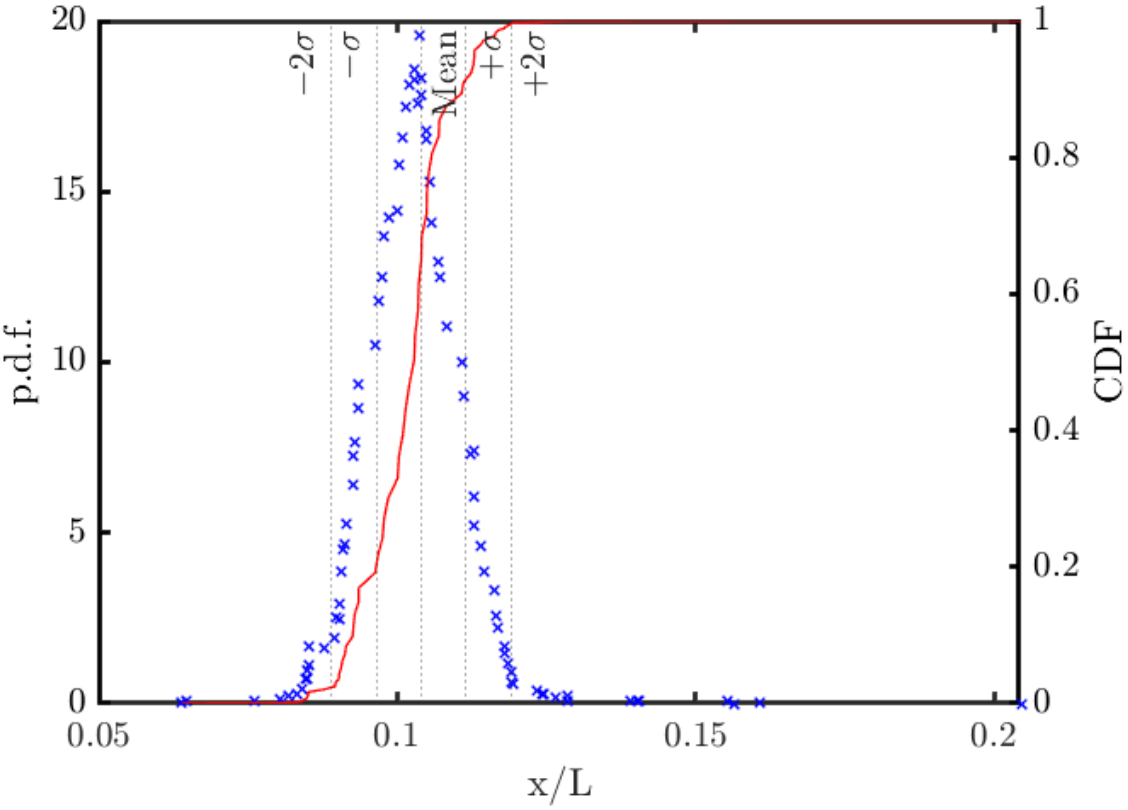


Figure 4.28: Probability density function of streamwise separation point location for $Re_L = 2.56 * 10^6$.

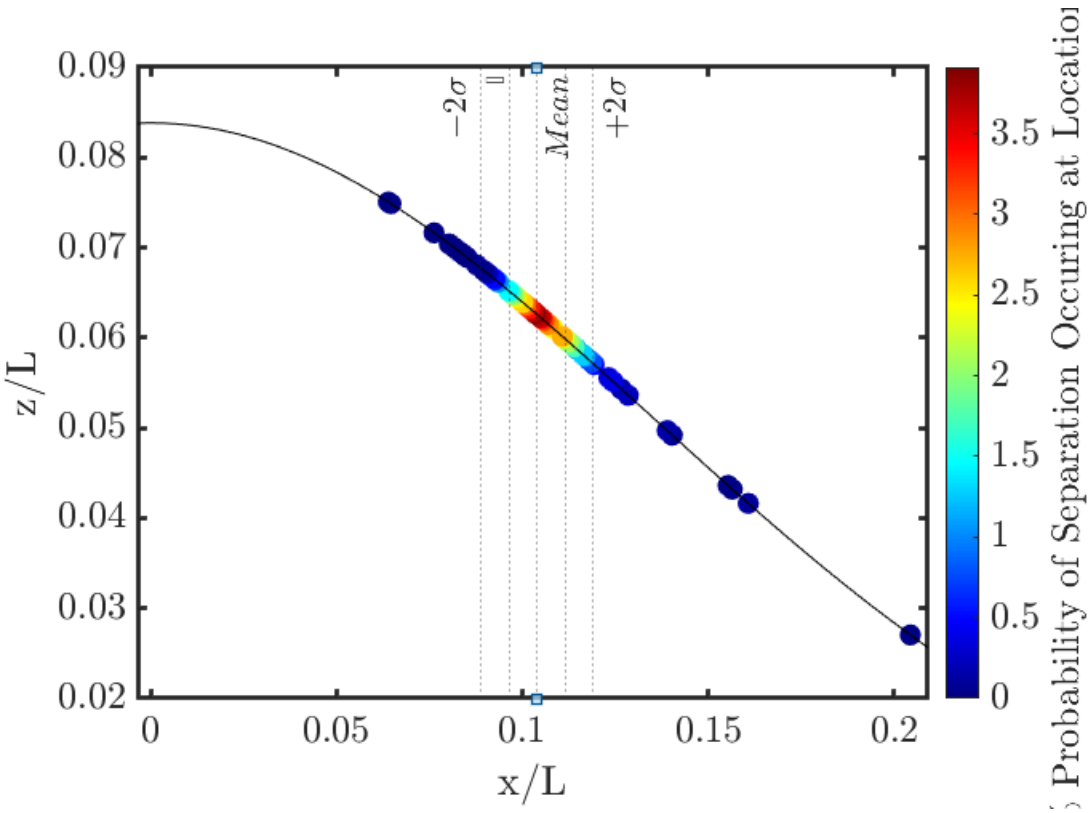


Figure 4.29: Separation locations plotted on bump surface along the bump centerline at $Re_L = 2.56 * 10^6$ colored by probability of separation occurring at that location.

4.4.3 Statistical Determination of Reattachment Point Location

Similar to the method used to calculate the separation point location, the intersection of the $U/U_\infty = 0$ contour in the conditionally averaged fields by α with the bump surface was defined to be the reattachment point. The PIV frames were focused on the downstream portion of the separated region to characterize the reattachment point motion and develop a statistical picture of its location.

Figure 4.30 shows the probability density function (p.d.f) and cumulative distribution function (CDF) of α across two FOVs of 10,000 PIV frames each taken along the bump centerline at Re $= 2.56 * 10^6$. The p.d.f of FOV 1 is Gaussian in nature with the mean and median at $\alpha = 0.1626$ and $\sigma = 0.0513$. The maximum instance of reverse flow has an area equivalent to $0.3434A$. The p.d.f. indicates there are zero PIV fields that show no reverse flow at all (i.e. $\alpha = 0$), suggesting that all frames contain boundary layer separation with regions of negative flow. The p.d.f of FOV 2, which is located downstream of FOV 1, is positively skewed with mean at $\alpha = 0.067$, median at $\alpha = 0.0619$, and $\sigma = 0.0381$. The maximum instance of reverse flow has an area equivalent to $0.2793A$. The p.d.f. indicates there are some PIV fields that show no reverse flow at all (i.e. $\alpha = 0$), suggesting that select frames contain no separated flow or the flow has already reattached upstream.

Figure 4.31 shows the p.d.f. and CDF of α in FOV 1 across 10000 frames along the bump centerline at Re $= 3.48 * 10^6$. The p.d.f. appears Gaussian in nature with a similar mean of $\alpha = 0.1627$, median of $\alpha = 0.1614$, and $\sigma = 0.0564$. The maximum instance of reverse flow fraction was also similar with an area equivalent to $0.369A$. Similar to Figure 4.30, the p.d.f does not indicate any PIV fields with no reverse flow.

The composite pdf of streamwise reattachment for $Re_L = 2.56 * 10^6$ shown in Figure 4.32 is calculated from two sets of 10,000 PIV frames that overlapped in the streamwise direction to capture the entire motion. Again, the pdf is nearly Gaussian but is negatively skewed with a few reattachment locations of low probability density located far upstream. The mean reattachment is located at $x/L = 0.3519$ with a standard deviation $\sigma = 0.0256$.

To characterize the unsteady reattachment point motion, the CDF indicates that nearly 98% of the reattachment points are located within $\pm 2\sigma$ from the mean. It is interesting to note the σ calculated for the reattachment point is nearly 3.5 times greater than the σ of the separation point. This suggests the reattachment point fluctuates more than the separation point and could potentially be influenced by a variety of factors such as for example, the oscillating shear layer in addition to the recirculation bubble entrainment.

The reattachment locations were plotted on the bump surface to serve as a visualization of the pdfs, as shown in Figure 4.33. The points are colored based on the probability of reattachment occurring at that location. It is very evident from Figure 4.33 that the reattachment point fluctuates from the middle of the downstream bump surface from $x/L = 0.300$ to $x/L = 0.403$, with majority of the motion centered about the mean as indicated with the red and orange points for $Re_L = 2.56 * 10^6$. There are points that extend as far upstream as $x/L = 0.18$, but these are likely to be very influenced by near-wall PIV uncertainty so they may not be true reattachment locations. Recall from Section 4.4.1 the mean streamwise separation location for $Re_L = 2.56 * 10^6$ is $x/L = 0.1044$. Knowing this and the mean reattachment location, the length of the separation bubble, L_{sep}, is 0.226m.

4.5 Correlation of Separation Point Motion with Upstream Disturbances

The PIV data was further analyzed to explore the possible influences of upstream disturbances on the movement of the separation point. The mean edge velocity $\frac{U_e}{U_\infty\, mean}$ at the edge of the mean turbulent boundary layer (determined by $z_w\omega_y$ method described in section 4.2.2) at the bump peak ($x/L = 0$) was calculated for the mean streamwise velocity field. The edge velocity $\frac{U_e}{U_\infty}$ at the same location of $\frac{U_e}{U_\infty\, mean}$ was then calculated for each streamwise velocity field conditioned by α as α was demonstrated to be directly related to separation point motion. The upstream velocity perturbation was then defined to be $\frac{U_e}{U_\infty}$-$\frac{U_e}{U_\infty\, mean}$ to center the distribution about 0. Similarly, α_{mean} was subtracted from α of the conditionally averaged fields to center the distribution about 0.

In red, Figure 4.34 shows the mean subtracted velocity perturbations in the boundary

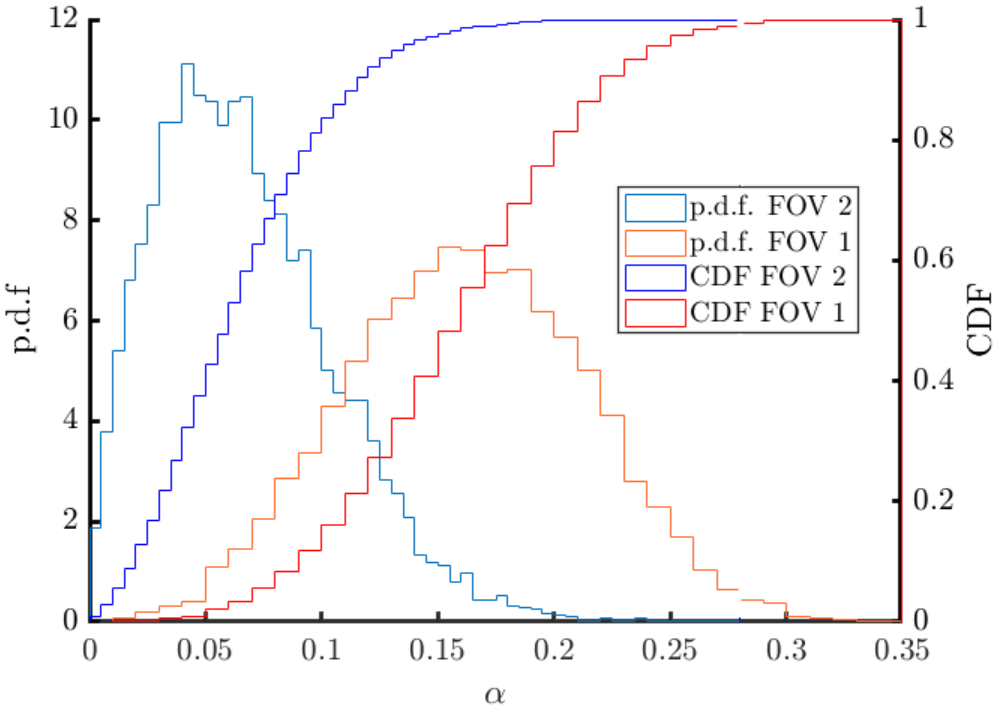


Figure 4.30: Probability density function (p.d.f) and cumulative density function (CDF) of α for $Re_L = 2.56 * 10^6$ focused on the reattachment region. Two FOVs were analyzed and their respective pdfs and CDFs are identified.

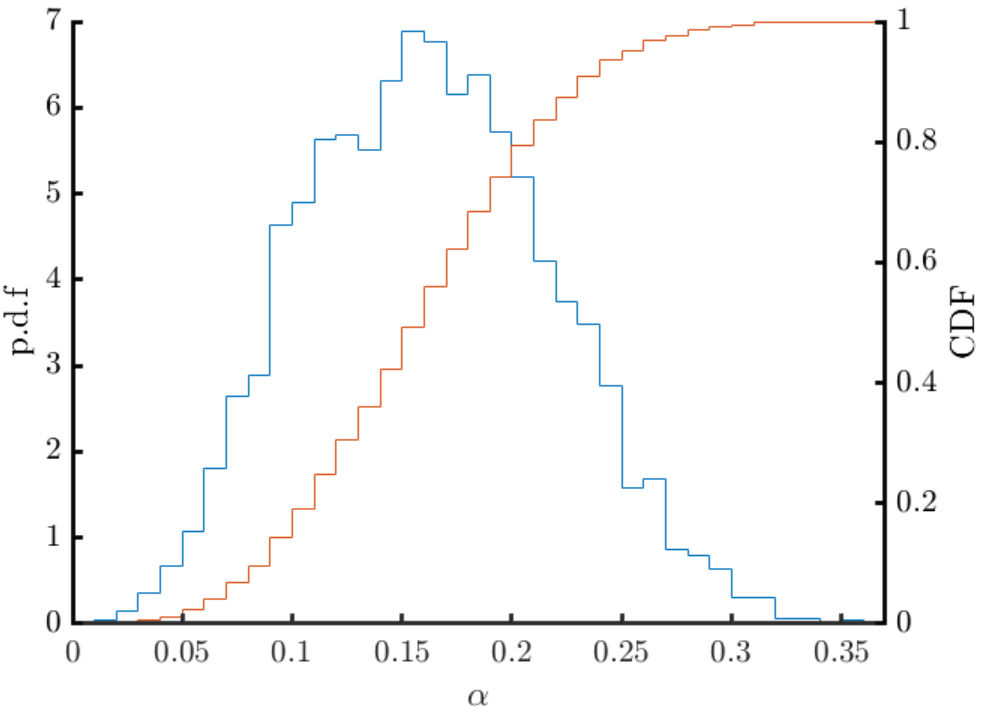


Figure 4.31: Probability density function (p.d.f) and cumulative density function (CDF) of α for $Re_L = 3.48 * 10^6$ focused on the reattachment region. .

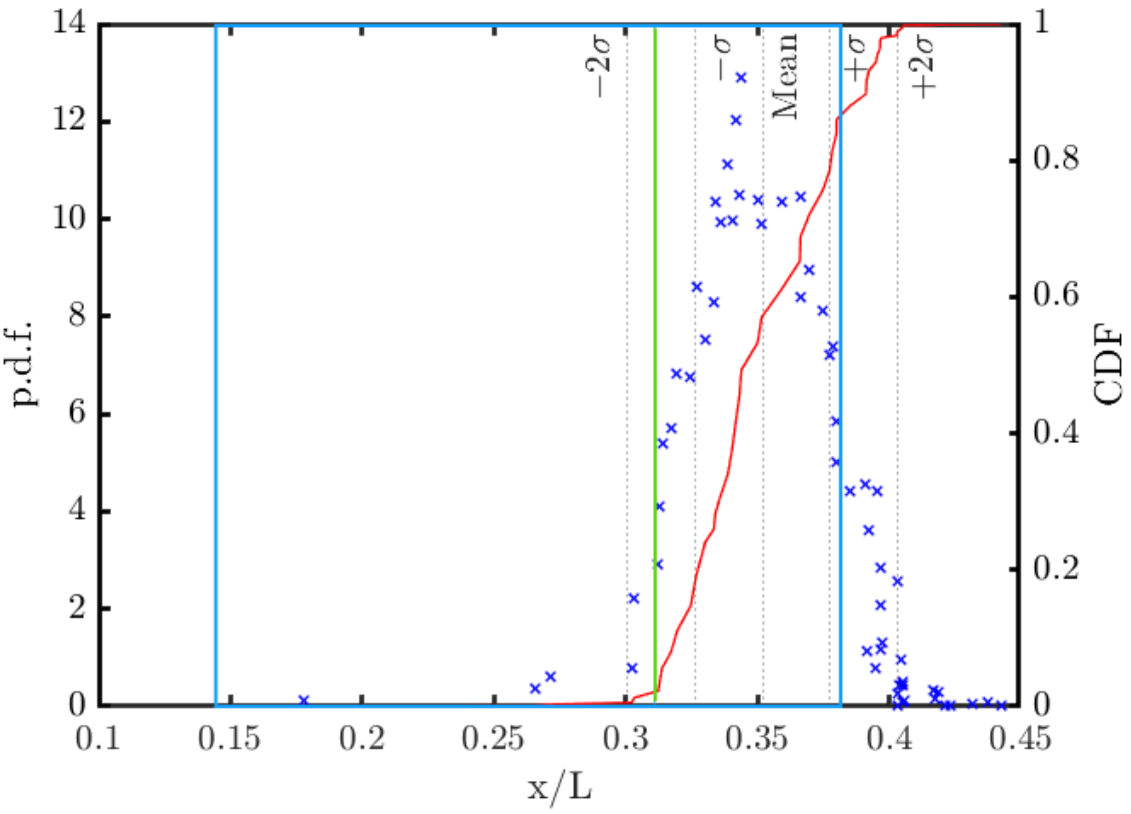


Figure 4.32: Probability density function of streamwise reattachment point location for $Re_L = 2.56 * 10^6$ along the bump centerline. The edges of the PIV field of views are shown in blue and green.

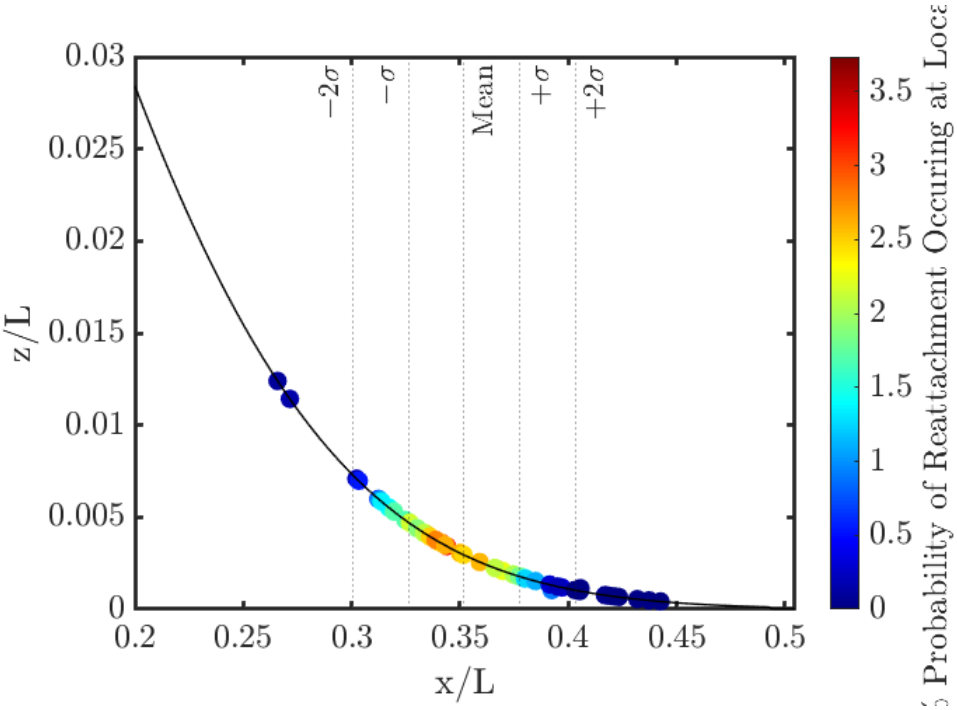


Figure 4.33: Reattachment locations plotted on bump surface along the bump centerline at $Re_L = 2.56 * 10^6$ colored by probability of reattachment occurring at that location.

layer above the bump peak against the mean subtracted α for each conditionally averaged field, centered about 0 along the bump centerline at $Re_L = 2.56 * 10^6$. The results of this analysis indicate a clear relationship between the upstream disturbance and reverse flow fraction (i.e. the separation point), as the reverse flow fraction decreases with increasing velocity perturbation, suggesting the separation point moves downstream from its mean location. This relationship is expected as higher velocity near the wall upstream of separation means the adverse pressure gradient must overcome greater near-wall momentum to force flow reversal, thereby delaying the separation event.

To further understand the relationship between the two quantities, Figure 4.34 shows the velocity perturbations against the individual α for all 40,000 PIV frames along the bump centerline at $Re_L = 2.56 * 10^6$, colored by distance to surrounding points. The number of samples in each α bin are shown in the upper plot. The distribution is approximately Gaussian similar to many turbulent processes. The elliptic point cloud is centered about the mean and rotated at an angle matching the slope of the mean subtracted velocity perturbations of each conditionally averaged α. It is clear from the point cloud that there are substantially more cases of high velocity perturbation-low α and low velocity perturbation-high α as seen in the top left and bottom right quadrants, further suggesting a relationship between the two quantities. The point cloud has a Pearson correlation coefficient, described in Equation 4.8, $r = -0.4886$, which indicates a significant correlation for a sample size of 40,000.

$$r = \frac{(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum(x_i - \overline{x})^2 \sum(y_i - \overline{y})^2}} \tag{4.8}$$

The data points that fall outside of the top left and bottom right quadrants suggest there are other forces acting in tandem or against the upstream disturbances such as the effects from the spanwise vortices and shear layer, which deserve further investigation. It is also possible the upstream boundary layer is modulating the intrinsic dynamics of the bubble.

With any subsonic flow, this observation drives the question as to whether the velocity perturbations cause the change in α, or if changes in α cause the velocity perturbations,

or if it is a combination of both. However, the separation events are located significantly downstream of the perturbation location, rendering it unlikely for the change in α to cause the perturbation itself. The low and high velocity perturbations seen in this analysis are of large scale as they are measured in the outer region of the turbulent boundary layer and this presents the question of how much the separation location is governed by these upstream disturbances versus separation bubble dynamics. Pearson et.al. found that in a subsonic flow, low-velocity perturbations are responsible only for the growth of the separation bubble, but not the subsequent decay. This means the influence of the convecting low-velocity region is only present for half the relevant time scale [17]. Performing time-resolved PIV on the separated region, while simultaneously calculating separation bubble growth and upstream velocity perturbations, may shed light on the governing mechanisms of separation point movement.

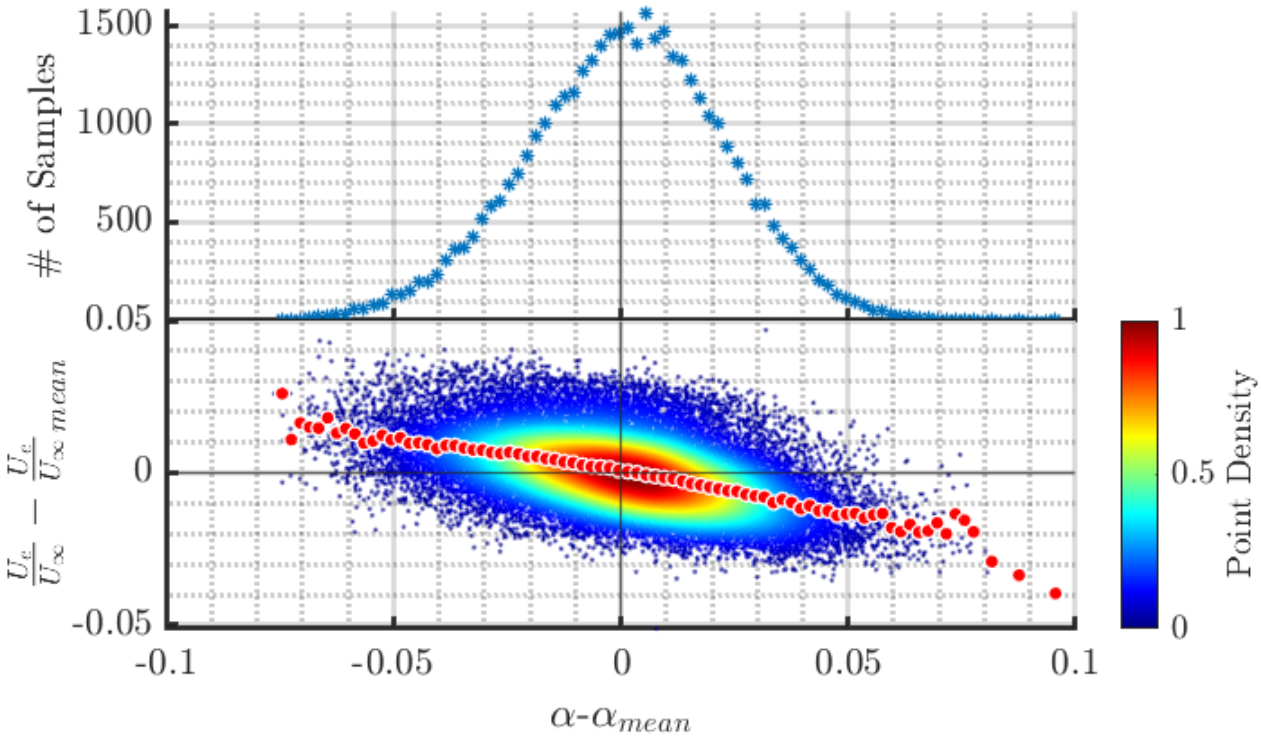


Figure 4.34: Above: Distribution of α. Below: The point cloud represents the velocity perturbation at the bump peak U_e/U_∞- $U_e/U_{\infty,mean}$ for 40,000 individual PIV frames plotted the against $\alpha - \alpha_{mean}$. The red dots indicate the same relationship for the PIV fields conditionally averaged by α. Both datasets are taken along the bump centerline at $Re_L = 2.56 * 10^6$.

Chapter 5

HIGH FREQUENCY PRESSURE ANALYSIS RESULTS AND DISCUSSION

5.1 High Frequency Pressure Data Analysis

This data was acquired in collaboration with Kevin Manohar from the University of Calgary and his and his advisors' support has been invaluable towards this project. Surface pressures were recorded at 48 locations on the splitter plate and over the speed bump located at series of streamwise planes and at symmetric spanwise locations. See Appendix A for a full list of the pressure tap locations. Pressure measurements were taken over the course of the PIV acquisition at 31250 Hz, yielding new high frequency data not acquired before on this test geometry. Surface pressures are presented as non-dimensional coefficients of pressure C_p. Although tap 1 was used to reference the pressure differential during measurement, C_p values are referenced to the pressure differential at tap 3 as per the following equation:

$$C_{px} = \frac{2RT_a\Delta P}{P_a U^2} = \frac{2RT_a((P_x - P_1) - (P_3 - P_1))}{P_a U^2} = \frac{2RT_a(P_x - P3)}{P_a U^2} \tag{5.1}$$

Pressure coefficients measured along the streamwise and spanwise centerline are shown in Figure 5.1 at $Re_L = 2.56 * 10^6$ in red, compared with the low-frequency pressure measurements taken by Robbins [10] at$Re_L = 2.55 * 10^6$ in black. The C_p results from 2020 were initially referenced to tap 0, so they were shifted to be offset by the respective tap 3 pressures. The results between the low and high frequency pressure acquisition match very well as the streamwise pressure gradients similarly vary as described in section 4.1 and the small asymmetry seen across the span by Robbins is substantially reduced.

Figure 5.2 shows pdfs of the high-frequency pressure data along the centerline taps for $Re_L = 2.56 * 10^6$. The most upstream tap in this plot, tap 7, is the pressure tap at the

bump peak and shows a bimodal pdf. Tap 7 exhibits the lowest C_p values as the flow remains attached and attains its highest velocity due to the favorable pressure gradient. As the adverse pressure gradient appears, there is a sharp increase in C_p, as seen in the rightward movement of the tap 8 pdf. Flow separation occurs in the vicinity of taps 9 and 10, whose pdfs are essentially on top of one another. This observation is depicted in Figure 5.1a as the plateau in C_p just downstream of the bump peak at $x/L = 0$. The pdfs of Taps 11-14 continue to march towards positive C_p as the flow reattaches and returns its freestream conditions. The streamwise location of Tap 11 ($x/L = 0.26$) nearly corresponds with the lower streamwise bound of mean reattachment ($x/L = 0.3$) presented in Figure 4.33, corroborating the previous statement. As Taps 11-14 move downstream towards the splitter plate, the pdfs become narrower and taller, indicating a smaller range and more frequently occurring mean C_p.

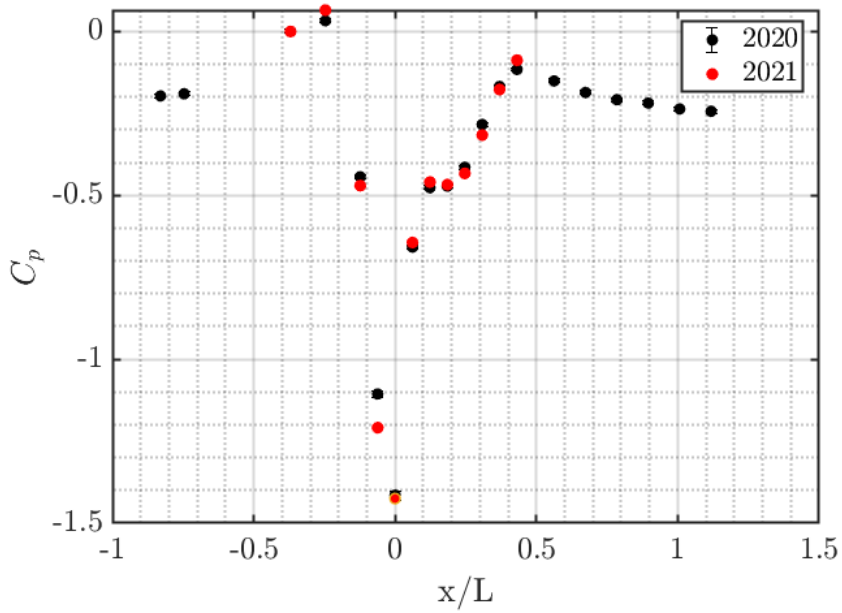
The premultiplied PSD of the fluctuating pressure signals calculated using Welch's method, is presented in Figure 5.3 for $0 < x/L < 0.4314$, measured across Taps 7 through 14. In this representation, the energy of each signal is proportional to the area under the semi-logarithmic curve. The measurement period was 240 seconds and the spectra were computed with 8 averaging windows, yielding a frequency resolution of $3.33 * 10^{-2}$ Hz. The frequency is presented in terms of the non-dimensional Strouhal number $St = fL_{sep}/U_\infty$ where $L_{sep} = 0.226m$ is the streamwise length of the mean separation bubble and $U_\infty = 44.6m/s$ is the freestream velocity over the splitter plate upstream of the bump. An expected similarity seen across all taps is the strong peak at $St = 2.187$, corresponding to the pressure tap cavity resonance frequency at 478 Hz.

For positions downstream of separation (Taps 11-14), the spectra are characterized by a broad peak centered at $St \approx 0.62$. The peak's height continuously decreases up to Tap 14. However, as this attenuation occurs, the signal's frequency shifts to a narrow peak centered at $St = 2.881$. Upstream of separation (Taps 7 and 8), there is virtually no energy in the pressure signal at $St = 0.62$. The Tap 7 PSD indicates there is some energy at the higher frequencies ($St \approx 10$) caused by the turbulent nature of the incoming boundary layer [25].

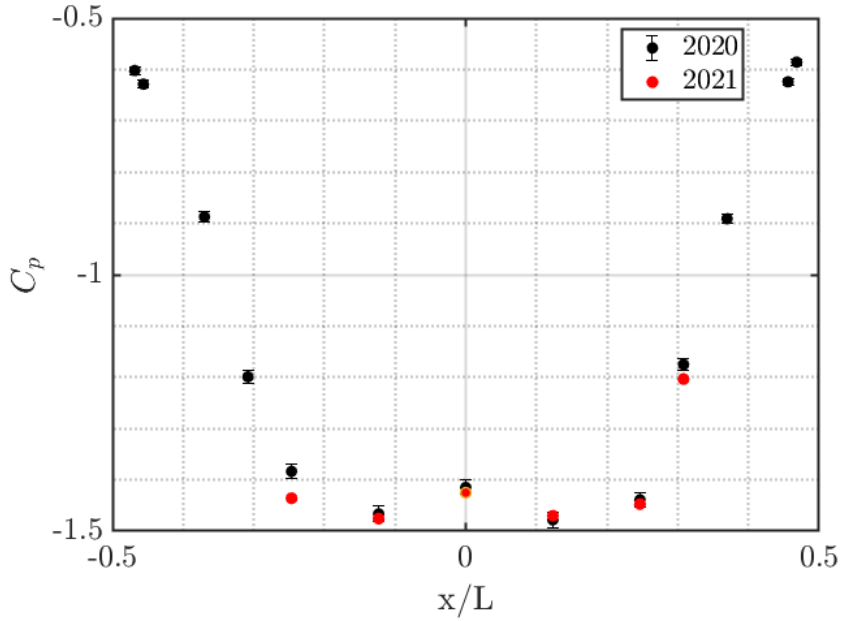
However, in Tap 8, there is another local maximum centered at $St \approx 0.05$. Although the amplitude of this wide peak is much smaller than that of the peak located at $St = 0.62$, its energy contribution to the signal is still high. There is no indication of any energy at $St \approx 0.05$ at the bump peak (Tap 7) and this region of high energy is present right before the boundary layer separates. After the separation occurs, there is negligible energy present at $St \approx 0.05$ as seen in the Tap 9 PSD.

The two dominant St numbers are indicative of various motions seen characteristic to turbulent separated flows in previous studies. The broad peak centered at $St = 0.05$ near mean separation at Tap 8, seems to be on the same order as the low frequency bubble breathing mode observed by Le'Floch et.al, at $St = fL_{sep}/U_\infty \approx 0.1$ [25]. The broadband peak centered at $St = 0.62$ near reattachment at Tap 10 seems to be consistent with the medium frequency shear layer and vortex shedding mode observed by Le'Floch et.al. at $St \approx 0.35$ [25] and by Hudy et.al. at $St \approx 0.7$ [52].

Examinations of the statistics of the surface pressure field along the centerline are presented for $Re_L = 2.56 * 10^6$ in Figure 5.4. The unsteadiness of the separated region is very evident in the visualization of the variance of C_p shown in Figure 5.4a. The variance suddenly increases just after $x/L = 0$ as the boundary layer starts to separate and the wall pressures start fluctuating as flow reversal, recirculation bubble growth and decay, and shear layer effects occur. The variance rapidly increases until the flow begins to reattach after $x/L = 0.2$, as the flow returns to freestream conditions. The skewness of the pressure distributions at each centerline location was also calculated and revealed a trend opposite to that of the variance. The skewness decreases on the upstream slope and the C_p distribution is nearly symmetrical at the bump peak. However, after separation the skewness drops considerably and continues to drop in the separated region as the pressure distributions skew towards higher C_p. Once the flow starts reattaching, the pressure distributions are almost symmetric at $x/L = 0.2465$, before the skewness rapidly increases as the pressure distributions skew toward lower C_p. This change in skewness is interesting to note and its coincidence with near reattachment deserves a closer look at its potential sources (i.e. shear layer effects).



(a)



(b)

Figure 5.1: (a) Streamwise centerline and (b) spanwise along bump peak mean experimental surface pressure profiles at H/L = 0.5, $Re_L = 2.56x10^6$ in 2021 compared to results taken in 2020.

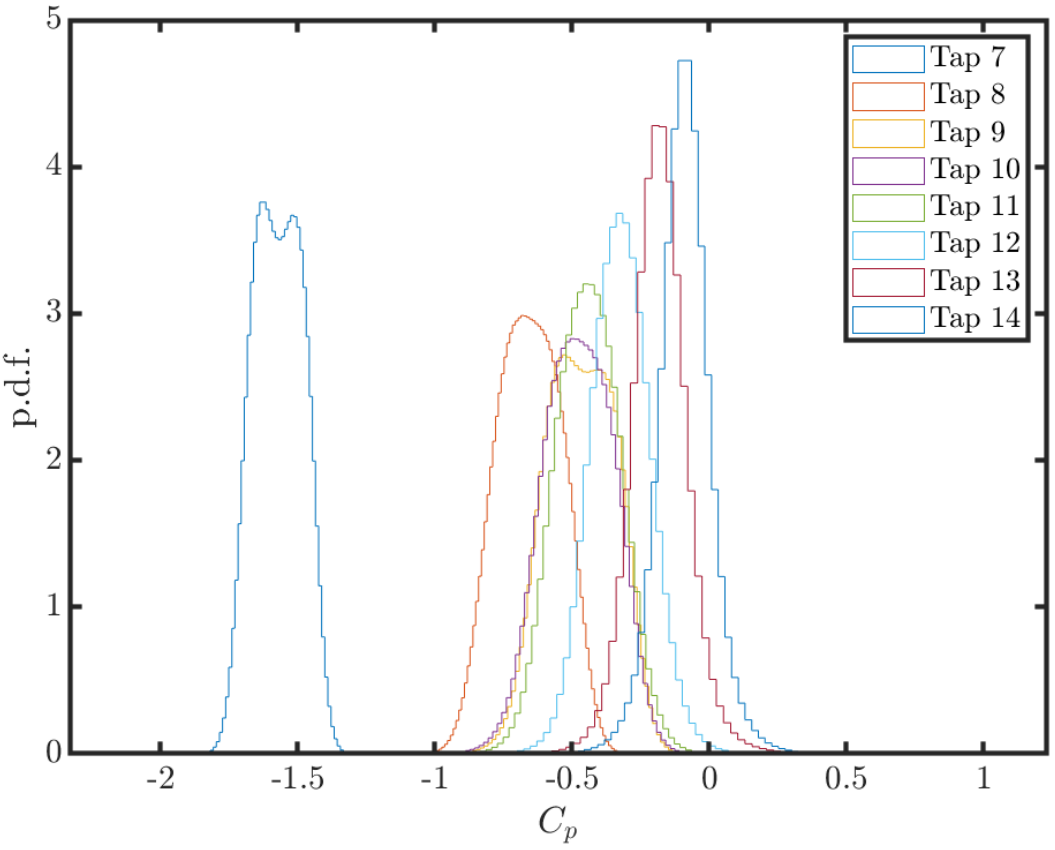


Figure 5.2: p.d.f.s of high-frequency pressure data along the centerline showing the shift of the C_p distributions in the streamwise direction at $Re_L = 2.56 * 10^6$ for $0 < x/L < 0.4314$.

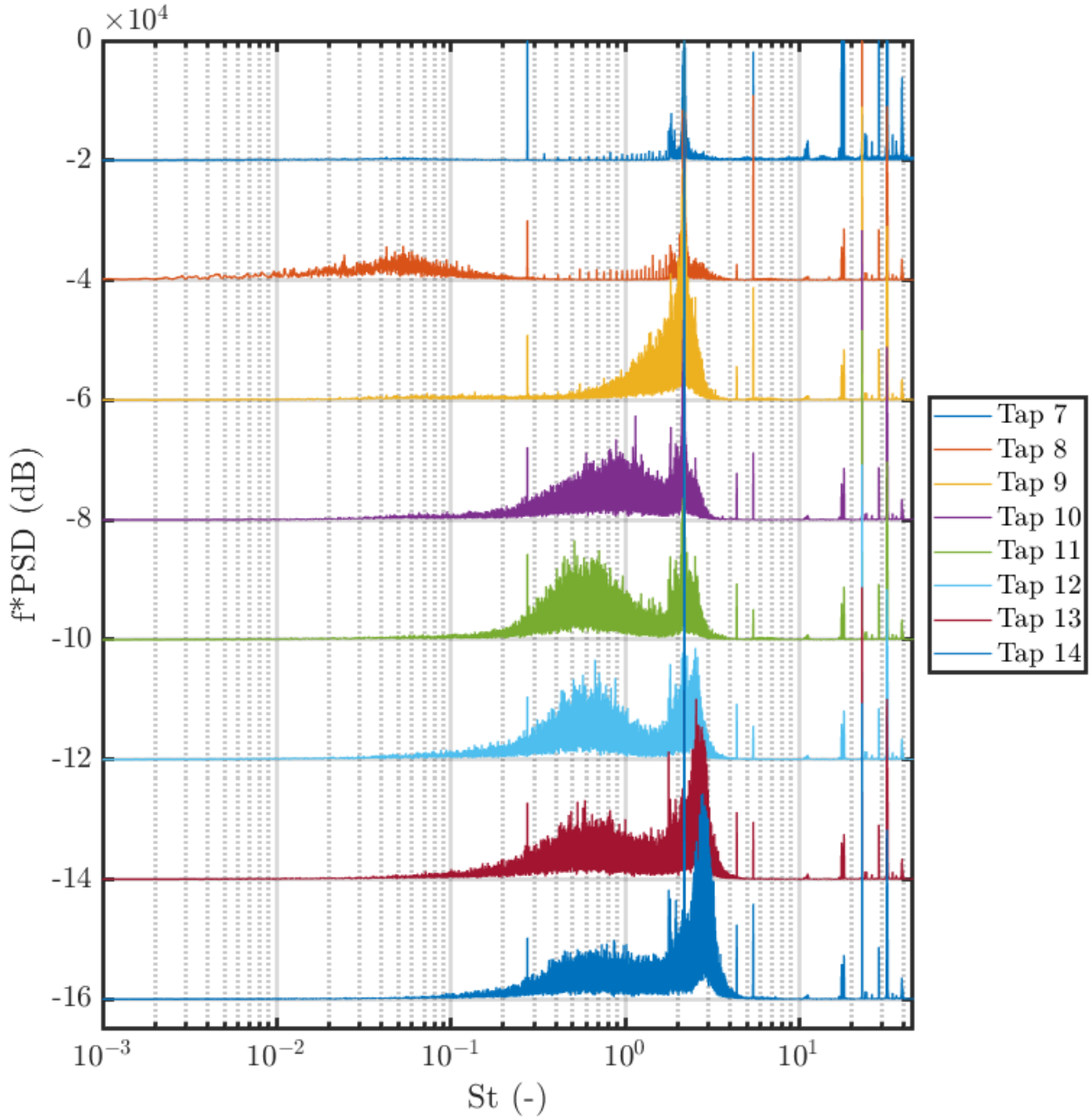
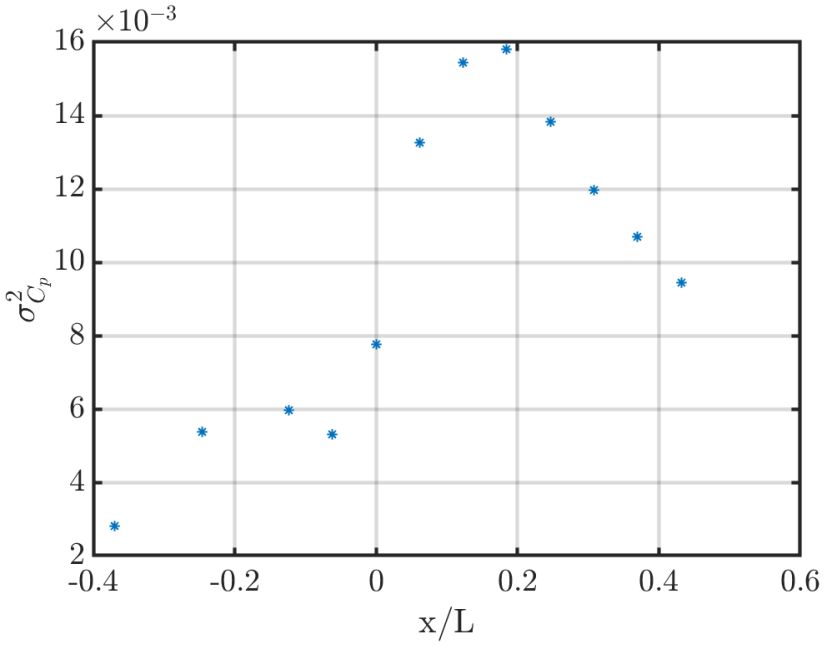
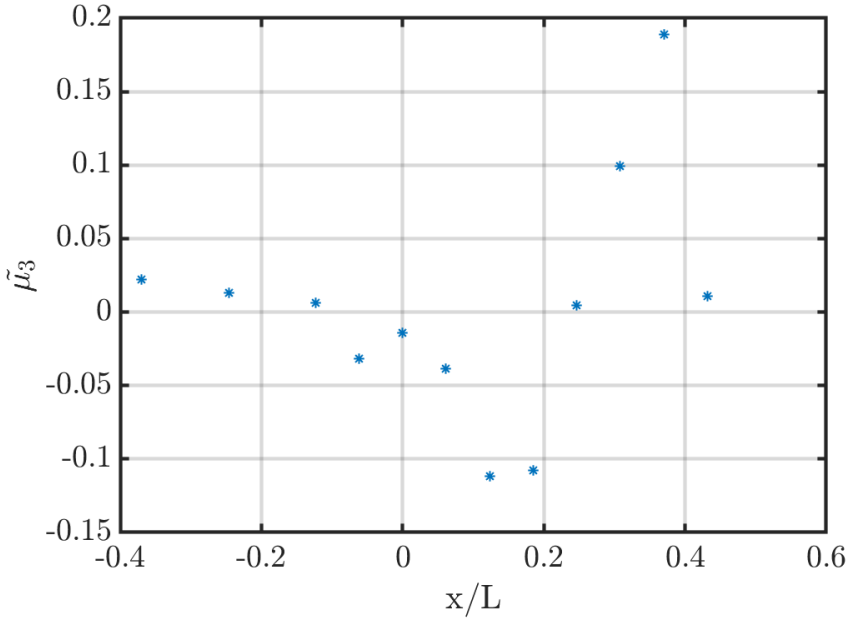


Figure 5.3: PSD of static-pressure fluctuations from $x/L = 0$ (Tap 7) to $x/L = 0.4134$ (Tap 14) at $Re_L = 2.56 * 10^6$ across Strouhal number $St = fL_{sep}/U_\infty$. Ordinates are shifted by 20000 dB for each position.



(a)



(b)

Figure 5.4: (a) Variance of C_p vs streamwise location and (b) Skewness of C_p vs streamwise location for $Re_L = 2.56 * 10^6$.

5.2 Analysis of Separation Point Based on PIV Synchronized with High Frequency Pressure Data

5.2.1 Pressure-based Conditional Averaging and Determination of Separation Point Location

Similar to the conditional averaging of PIV fields based on reverse flow fraction α, the PIV fields were conditionally averaged by instantaneous C_p to understand the effect of pressure on estimated separation point location. The PIV fields were synchronized to their respective high frequency pressure readings via matching the q-switch signals such that a series of readings across various taps were specifically associated with a given PIV field. To learn more about the structure of separation at various C_p magnitudes and determine the separation points, the streamline patterns of various averaged velocity fields conditioned over a small range of C_p were inspected and the separation locations were extracted. As with determining the separation location when the PIV fields were conditioned by α, the intersection of the $U/U_\infty = 0$ contour with the bump surface was defined to be the separation point.

Figure 5.5 depict pdfs of streamwise separation points for $Re_L = 2.56 * 10^6$, along the spanwise centerline, from PIV fields conditionally averaged by C_p across the centerline taps near separation. None of the pdfs have distributions that have any shape to them suggesting that the pressure signals are not correlating closely with separation location. As a result, majority of streamwise separation locations calculated across the conditionally averaged fields lie very near the mean. As expected, the mean streamwise separation location calculated across the averaged fields by all taps is $x/L \approx 0.1040$ and agrees well with the result calculated from conditionally averaging the fields by *alpha*, equal to $x/L = 0.1047$. As indicated in the figures, σ of the streamwise separation location is nearly constant as well across pdfs with a value of 0.0048. Although, this is substantially less than the σ calculated using the α method, the true dynamics of separation point motion is not currently captured in these pdfs.

However, it is expected that the separation point motion be dependent on wall surface

pressures as velocities near the wall fluctuate as the flow is drawn across the leeward bump face and up the recirculated region. Analysis of the surface pressures showed the variance of the pressure significantly increased in the separated region, further increasing the expectation of a relationship between the two quantities. A limitation with the current PIV data is the direction of the separation point motion is unknown for a given PIV frame, as time-resolved PIV could not be acquired. This could inform the C_p conditional averaging method by pairing the direction as another event criteria to know whether the pressure is decreasing or increasing as the separation point moves in a certain direction. This is avenue for further investigation that could yield very interesting results.

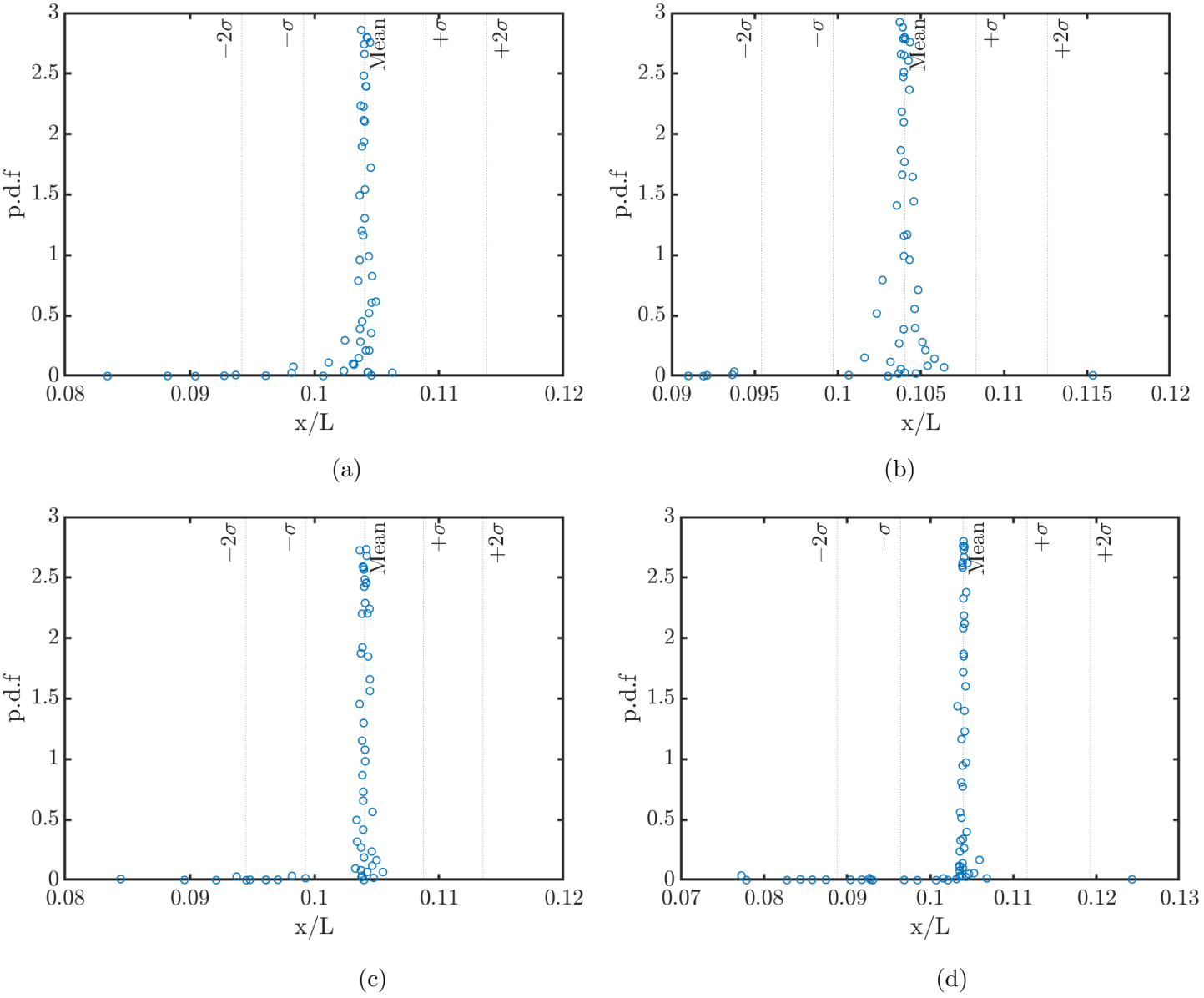


Figure 5.5: p.d.f.s of streamwise separation locations calculated from PIV fields conditionally averaged by C_p along the centerline for $Re_L = 2.56 * 10^6$ for taps located at (a) $x/L = 0$ (Tap 7), (b) $x/L = 0.0616$ (Tap 8), (c) $x/L = 0.1233$ (Tap 9), and (d) $x/L = 0.1849$ (Tap 10).

Chapter 6

CONCLUSION

Previous work in separated turbulent boundary layers has often focused on geometry where the separation location is fixed ([17], [18]), hill type geometries with varying surface curvature and pressure gradients such as the Byun/Simpson, FAITH and BeVERLi hills, ([31]-[38],[3]), and pressure-induced separation ([20], [25], [28]). The speed bump at the University of Washington is a novel geometry in which the incoming turbulent boundary layer undergoes a combination of varying pressure gradients and interchanging surface curvature and is only weakly three-dimensional at the centerline. This presents a new lens through which turbulent separation dynamics can be investigated and is a challenge to current turbulence models and CFD capabilities.

The size and shape of the separated region were thoroughly investigated through mean spatial flowfields measured using PIV. Flow separation was explored for Reynolds numbers of $Re_L = 1.39(20m/s)$, $2.56 * 14^6(40m/s)$ and $3.48 * 10^6(60m/s)$. The major structure found in both mean fields was a large separated region containing a shear layer and recirculation bubble. The separated region and line of separation were found to be three-dimensional, with separation occurring further upstream along the spanwise centerline. The size of the separated region was found to decrease as the distance increased from the centerline. The region of positive mean vertical velocity was larger at the centerline, suggesting significant flow is drawn upwards from the surface, into the separated region near the spanwise centerline, and transported downstream.

The streamwise velocity fields were conditionally averaged by the reverse flow fraction α to characterize the separation and reattachment motion and calculate distributions of their location on the bump surface. The mean separation point for $Re_L = 2.56 * 10^6$ was

found to be located at $x/L = 0.105$ and fluctuated between $0.089 < x/L < 0.120$ ($\pm 2\sigma$). The mean reattachment point was found to be located at $x/L = 0.3519$ and fluctuated between $0.3 < x/L < 0.403$ ($\pm 2\sigma$). This yields a mean streamwise separation bubble length of $0.226m$. Cumulative distribution functions of the separation and reattachment point locations indicated at least 95% of the points were located within $\pm 2\sigma$ from the respective mean values. The probability distribution functions also suggested that majority of the motion can be described as Gaussian, with positive skewness of separation and negative skewness of reattachment.

The relationship between upstream velocity perturbations and reverse flow fraction (i.e. separation location) across conditionally averaged fields was determined and showed that as velocity perturbations increased, the reverse flow fraction decreased, moving the separation event further downstream. The same analysis was conducted for the individual 40,000 PIV frames and the resulting point cloud had a statistically significant Pearson correlation coefficient r = -0.489. This suggests that the disturbances in the upstream boundary layer are influencing the dynamics of the separation bubble.

The evolution of the upstream boundary layer prior to separation was characterized as it underwent changes due to varying pressure gradients and interchanging surface curvatures. It is known that convex curvature has a destabilizing effect on turbulence and concave curvature has the opposite ([26], [27]). However, in the presence of an adverse pressure gradient with concave surface curvature as seen at the foot of the bump, δ_ω, was seen to increase and C_f from simulations was observed to decrease [42], indicating the dominance of the adverse pressure gradient. This is similar to the conclusion made by Uzun [42]. Similarly, the change in curvature from concave to convex under a favorable pressure gradient showed that δ_ω decreased and C_f increased [42], signaling the favorable pressure gradient dominated in this region, again similar to the conclusion drawn by Uzun [42]. Figures 4.14 and 4.18 suggest that the pressure gradient has a stronger affect on governing the non-equilibrium layer dynamics when under the influence of changing surface curvature for the speed bump.

POD of the fluctuating streamwise and wall-normal velocities in the separated region

revealed that the first 25 modes accounted for 73% of the total fluctuating energy across the 25,000 PIV frames. The scaled mode energies showed that ϕ^1, believed to represent the low frequency contraction and expansion of the recirculation bubble, contributed 32.3% to the energy. ϕ^2 and ϕ^3 seemed to represent the convection of slow moving vortical structures which could draw in flow from the upstream and freestream above the shear layer and account for 13.3% of the fluctuating energy. ϕ^4 and ϕ^5 appeared to depict faster moving vortical structures and ϕ^6 seemed to show the shear layer. The shear stress mode, ϕ^1_{-uv}, indicates the strong presence of $-uv$, suggesting the low frequency bubble motion could be the largest turbulence producing mechanism in this separated flow.

High frequency pressure data was acquired across various locations on the bump surface. Pressure distributions showed that the variance rapidly increased in the separated region and starkly reduced when the flow reattaches, signaling large pressure fluctuations that are characteristic of the separated region. The skewness follows an opposite trend and becomes more negative (more high C_p values) going into the separation region and then rapidly becomes positive (more low C_p values) after reattachment. The mean C_p values along the spanwise and streamwise centerlines agreed well with previous low frequency pressure data taken at identical locations. Spectra analysis of the high frequency data along the centerline taps revealed a wide broadband region of energy just downstream of the bump peak before separation at $St = fL_{sep}/U_\infty \approx 0.05$. This disappears further downstream. $St \approx 0.05$ compares well with the results from previous turbulent separated flow studies, suggesting it could be describing the low frequency motion associated with the bubble breathing. Another strong peak at $St \approx 0.62$ was observed downstream of separation and decreased in strength as the streamwise location increased. This peak was not seen upstream of separation. $St \approx 0.62$ compares well with the medium frequency motion associated with shear layer and vortex shedding observed in various separated flow studies.

REFERENCES

[1] Bell, J., Heineck, J. T., and Mehta, R. D., "Surface and flow field measurements on the FAITH Hill Model," 2012, 50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition.

[2] Lynch, K., "A CFD Validation Challenge for Transonic, Shock-Induced Separated Flow: Experimental Characterization," 2020, AIAA 2020-1309, Session: Transonic Shock-Induced Separated Flow.

[3] Williams, O., Annamalai, H., Ozoroski, T., Christopher, R., and Lowe, T., "Comparison of hill-type geometries for the validation andadvancement of turbulence models," 2022, AIAA SciTech Forum.

[4] Williams, O. J., Samuell, M., Robbins, M. L., Annamalai, H., and Ferrante, A., "Characterization of separated flowfield over Gaussian speed-bump CFD validation geometry," AIAA SciTech Forum 2021.

[5] UW Facility Management, D. D., 1992, Aerodynamics Lab Building Remodel Project 010-16403. Tech. rep.

[6] Chauvete and Sons, "Hurricane 1800 Flex User Manual, Version 3," In: 2019.

[7] Inc., L., "Image sCMOS Datasheet," In: March 2018.

[8] Quantel, b. L., "EverGreen 200 Laser Datasheet," In: January 2014.

[9] Inc., L., "Programmable Timing Unit (PTU X) Product Manual," In: November 2020.

[10] Robbins, M., *Detailed Characterization of Flowfields and Uncertainty in a Speed-Bump Turbulent Separated Flow Validation Experiment*, Master's thesis, University of Washington, 2020.

[11] Williams, O. J., Samuell, M., Sarwas, S., Robbins, M. L., and Ferrante, A., "Experimental Study of a CFD Validation Test Case for Turbulent Separated Flows," AIAA SciTech Forum 2020.

[12] Samuell, M. C., *Development of a Turbulent Separated Flow Validation Test Case: Experimental and Computational (RANS) Studies*, Master's thesis, University of Washington, 2020.

[13] Uzun, A. and Malik, M. R., "High-Fidelity Simulation of Turbulent Flow Past a Gaussian Bump," 11 2021.

[14] Inc., E. L. D., "Operating and Maintenance Instructions 36" Open Circuit Wind Tunnel," .

[15] Finley, P. J., "CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences," *Journal of Fluid Mechanics*, Vol. 26, No. 1, 1966, pp. 337–368.

[16] Wu, W., Meneveau, C., and Mittal, R., "Spatio-temporal dynamics of turbulent separation bubbles," *Journal of Fluid Mechanics*, Vol. 883, 2019, pp. A45.

[17] Pearson, D., Goulart, P., and Ganapathisubramani, B., "Turbulent separation upstream of aforward-facing step," *Journal of Fluid Mechanics*, Vol. 724, 2013, pp. 284–304.

[18] Fang, X. and F., T. M., "Spatio-temporal dynamics of flow separation induced by a forward-facing step submerged in athick turbulent boundary layer," *Journal of Fluid Mechanics*, Vol. 892, No. 4, 2020, pp. A40.

[19] Deck, S. and Thorigny, P., "Unsteadiness of an axisymmetric separating-reattaching flow: Numerical investigation," *Physics of Fluids*, Vol. 19, 2007, pp. A40.

[20] Weiss, J., Mohammed-Taifour, A., and Schwaab, Q., "Unsteady Behavior of a Pressure-Induced Turbulent Separation Bubble," *AIAA Journal*, Vol. 53, 2015, pp. 2364–2645.

[21] Clemens, N. T. and Narayanaswamy, V., "Low-Frequency Unsteadiness of Shock Wave/-Turbulent Boundary Layer Interactions," *Annural Review of Fluid Mechanics*, Vol. 46, 2014, pp. 469–492.

[22] Simpson, R. L., "Turbulent Boundary Layer Separation," *Annual Review of Fluid Mechanics*, Vol. 21, 1989, pp. 205–234.

[23] Deshpande, A. S. and Poggie, J., "Unsteady characteristics of compressible reattaching shear layers," *Physics of Fluids*, Vol. 32, 2020.

[24] Malm, J., Schlatter, P., and Sandham, N., "A vorticity stretching diagnostic for turbulent and transitional flows," *Theoretical and Computational Fluid Dynamics*, Vol. 26, 2012.

[25] Le'Floch, A., Weiss, J., Mohammed-Taifour, A., and Dufresne, L., "Measurements of pressure and velocity fluctuations in a family of turbulent separation bubbles," *Journal of Fluid Mechanics*, Vol. 902, 2020, pp. A13.

[26] Muck, K., Hoffmann, P., and Bradshaw, P., "The effect of convex surface curvature on turbulent boundary layers," *Journal of Fluid Mechanics*, Vol. 161, 1985, pp. 347–369.

[27] Hoffmann, P., Muck, K., and Bradshaw, P., "The Effect of concave Surface Curvature on Turbulent Boundary Layers," *Journal of Fluid Mechanics*, Vol. 161, 1985, pp. 371–403.

[28] Trunkle, J., Mohammed-Taifour, A., and Weiss, J., "Fluctuating pressure measurements in a turbulent separation bubble," *Comptes Rendus Mecanique*, Vol. 344, 2015.

[29] Coleman, G., Rumsey, C., and Spalart, P., "Numerical study of a turbulent separation bubble with sweep," *Journal of Fluid Mechanics*, Vol. 880, 2019, pp. 684–706.

[30] Webste, D., DeGraff, D., and Eaton, J., "Turbulence characteristics of a boundary layer over a swept bump," *Journal of Fluid Mechanics*, Vol. 323, 2006, pp. 1–22.

[31] Byun, G. and Simpson, R. L., "Structure of three-dimensional separated flow on an axisymmetric bump," *AIAA Journal*, Vol. 44, 2006.

[32] Byun, G., Simpson, R. L., and Long, C., "Study of Vortical Separation from Three-Dimensional Symmetric Bumps," *AIAA Journal*, Vol. 42, 2004, pp. 754–765.

[33] Byun, G. and Simpson, R. L., "Surface-Pressure Fluctuations from Separated Flow over an Axisymmetric Bump," *AIAA Journal*, Vol. 48, 2010, pp. 2397–2405.

[34] Patel, N., Stone, C., and Menon, S., "Large-Eddy Simulation of Turbulent Flow over an Axisymmetric Hill," 2003, 41st Aerospace Sciences Meeting and Exhibit.

[35] Castagna, J., Yao, Y., and Yao, J., "Numerical Simulation of a Turbulent Flow over an Axisymmetric Hill," 2012, 50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition.

[36] Husen, N. M., Liu, T., and Sullivan, P., "Luminescent Oil Film Flow Tagging Skin Friction Meter Applied to FAITH Hill," 2018, https://arc.aiaa.org/doi/abs/10.2514/1.J057114?journalCode=aiaaj.

[37] Bachalo, W. and Johnson, D., "Transonic, Turbulent Boundary-Layer Separation Generated on an Axisymmetric Flow Model," *AIAA Journal*, Vol. 24, 1986, pp. 437–443.

[38] Lynch, K. P., Beresh, S. J., Miller, N., Barone, M. F., Spillers, R. W., and Soehnel, M. M., "Revisiting Bachalo-Johnson: The Sandia Axisymmetric Transonic Hump and CFD Challenge." .

[39] Gargiulo, A., Duetsch-Patel, J., Ozoroski, T. A., Beardsley, C., Vishwanathan, V., Fristch, D., Borgoltz, A., Devenport, W., Roy, C., and Lowe, T., "Flow Field Features of the BEVERLI Hill Model," 2021, AIAA SciTech Forum 2021.

[40] Slotnick, J. P., "Integrated CFD validation experiments for prediction of turbulent separated flows for subsonic transport aircraft," 2019, In: In NATO Science and Technology Organization, Meeting Proceedings RDP, STO-MP-AVT-307.

[41] Sarwas, S., *Experimental Examination of New Separated Turbulent Flow Validation Test Geometry*, Master's thesis, University of Washington, January 2019.

[42] Uzun, A. and Malik, M. R., "Simulation of a turbulent flow subjected to favorable and adverse pressure gradients," *Theoretical and Computational Fluid Dynamics*, Vol. 35, 2021, pp. 293–329.

[43] Balin, R., Jansen, K., and Spalart, P., "Wall-Modelled LES of Flow over a Gaussian Bump with Strong Pressure Gradients and Separation," AIAA SciTech Forum 2020.

[44] Commission, I. E., "IEC 60584-1 Ed. 3.0 b:2013, Thermo- couples Part 1 EMF Specifications And Tolerances," https://www.thermocoupleinfo.com/thermocouple-accuracies.htm, 2022-05-2.

[45] Traceable, "DigitalMonitoringTraceableBarometer," https://www.traceable.com/6530-traceable-digital-barometer.html#product_tabs_additional_tabbed, 2022-05-2.

[46] Westerwheel, J., *Digital particle image velocimetery: theory and application*, Delft University Press, 1993.

[47] Spalart, P. R. and Watmuff, J. H., "Experimental and numerical study of a turbulent boundary layer with pressure gradients," *Journal of Fluid Mechanics*, Vol. 249, 2006, pp. 337–371.

[48] Weiss, J., "A Tutorial on the Proper Orthogonal Decomposition," 2019, AIAA Aviation Forum 2019.

[49] Kutz, J. N., *Data-driven modeling & scientific computation: methods for complex systems & big data*, Oxford University Press, 2013.

[50] Adrian, R. and Moin, P., "Stochastic estimation of organized turbulent structure:homogeneous shear flow," *Journal of Fluid Mechanics*, Vol. 190, 1988, pp. 531–559.

[51] Stuer, H., Gyr, A., and Kinzelbach, W., "Laminar separation on a forward facing step," *European Journal of Mechanics - B/Fluids*, Vol. 18, 1999, pp. 675–692.

[52] Hudy, L. M., Naguib, A. M., and Humphreys, W. M., "Wall-Pressure-Array Measurements Beneath a Separating/Reattaching Flow Region," *Physics of Fluids*, Vol. 15, 2003.

Appendix A

A.1 Manufacturer Data

Table A.1: Omega PX653 pressure sensor manufacturer specifications

Excitation (Vdc)	12 to 36
Output (Vdc)	1 to 5
Linearity (%FS)	0.3
Hysteresis (%FS)	0.02
Repeatability (%FS)	0.05
Operating Temperature (°C)	-29 to 72
Compensated Temperature (°C)	2 to 57
Thermal Effects (zero)(%FS/°F)	0.015
Thermal Effects (span)(%FS/°F)	0.015
Proof Pressure (psi)	15
Burst Pressure (psi)	20
Static Pressure (psi)	25
Gage Type	Capacitance
Supply Current (mA)	<5
Response Time (ms)	250

Table A.2: Baratron 226A pressure sensor manufacturer specifications

Resolution (% FS)	0.01
Accuracy (% FS)	0.30
Temperature Coefficient (zero)(%FS/°C)	0.01
Temperature Coefficient (span)(%FS/°C)	0.04
Ambient Operating Temperature (°C)	0 to 50
Maximum Overpressure (Measurement)	120%FS or 20 psi
Maximum Overpressure (Reference) (%FS)	120
Maximum Line Pressure (psig)	40

Table A.3: NI 9485 solid-state relay digital output module manufacturer output characteristics

Relay type	Normally open solid-state relay (SSR)
Switching voltage	60Vdc, 30Vrms max
Switching current, per channel (A)	1.2 max
Switching rate (90% duty cycle)	1 operation/second
Relay open time (ms)	0.5
Relay close time (ms)	9.0
On Resistance (mΩ)	200 max
Off stage leakage (µA)	30
MTBF (hours at 25°C)	2,172,740

Table A.4: NI cDAQ 9213 thermocouple module manufacturer input specifications and accuracy (high-resolution mode, 25 °C, type K thermocouple)

ADC resolution (bits)	24
Type of ADC	Delta-Sigma
Sampling Mode	Scanned
Voltage measurement range (mV)	±78.125
Conversion time (per channel) (ms)	55
Sample rate (all channels) (S/s)	1
Input bandwidth (Hz)	14.4
Noise rejection (dB)	60
Differential input impedance (M)	78
Input noise (nVrms)	200
Gain error	0.03% typical
Offset error (μV)	4 typical, 6 maximum
Offset error from source impedance	Add 0.55 μV/Ω when ¿50Ω
Cold-junction compensation accuracy (°C)	0.8 typical, 1.7 maximum
Measurement sensitivity (°C)	<0.02

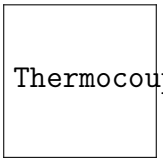


Figure A.1: NI cDAQ 9213 thermocouple Type K measurement error (accounts for gain errors, offset errors, differential and integral nonlinearity, quantization errors, noise errors, 50 lead wire resistance, and cold-junction compensation errors.

Table A.5: NI cDAQ 9205 voltage input module manufacturer analog input and accuracy specifications (± 5V input)

ADC resolution (bits)	16
DNL	No missing codes guaranteed
Conversion time (μs)	4.00 (250 kS/s)
Input couple	DC
Input impedance (GΩ)	<10 in parallel in 100 pF
Input bias current (pA)	±100
Crosstalk, at 100kHz (adjacent channels)(dB)	-65
Analog bandwidth (kHz)	370
Scaling coefficient (μV/LSB)	164.2
CMRR, DC to 60 Hz (dB)	100
Accuracy at full-scale (μV)	3230
Random Noise (μV RMS)	121
Sensitivity (μV)	46.4
Residual gain error (ppm of reading)	135
Gain tempco (ppm/°C)	11
Reference tempco (ppm/°C)	5
Residual offset error (ppm of range)	20
Offset tempco (ppm of range/°C)	47
INL error (ppm of range)	76

Table A.6: NI PCIe 6361 voltage DAQ manufacturer analog input and accuracy specifications (± 10V input)

ADC resolution (bits)	16
DNL	No missing codes guaranteed
Sample Rate (MS/s)	2.00 maximum
Timing resolution (ns)	10
Timing accuracy (ppm of sample rate)	50
Input coupling	DC
CMRR, DC to 60 Hz (dB)	100
Input impedance (GΩ)	<10 in parallel with 100 pF
Input bias current (pA)	±100
Crosstalk, at 100 kHz (adjacent channels) (dB)	-75
Small signal bandwidth (-3 dB) (MHz)	1.7
Input FIFO size (samples)	2,047
Data transfers	DMA (scatter-gather), programmed I/O
Settling time (±60 ppm of Step) (s)	1
Residual gain error (ppm of reading)	48
Residual offset error (ppm of range)	13
Offset tempco (ppm of range /°C)	21
Random noise (V RMS)	315
Absolute accuracy at full-scale (V)	1,660
Gain tempco (ppm/°C)	13
Reference tempco (ppm/°C)	1
Sensitivity (V)	46.4
INL error (ppm of range)	60

Table A.7: Advantech PCIE-1805 manufacturer analog input and accuracy specifications (± 5V input)

ADC resolution (bits)	16
Input coupling	DC
Maximum sample rate (fs)	1 MS/s shared by all enabled channels
Common-mode rejection ratio (CMRR, at 60 Hz)	85 dB
Input bias current	1pA
-3 dB bandwidth	3 MHz
RMS Noise	0.1 mV
Hysteresis	Software programmable
Operating temperature	0 to 60 °C (32 to 140 °F)
Operating humidity	10 to 90% RH, non-condensing

Table A.8: Programmable Timing Unit (PTU X) I/O specifications

Output drivers (Ω)	TTL 50
Time resolution (ns)	10
Typical jitter between all outputs	(ns) < 1
Jitter to external signals	(ns) $\pm$ 5
Trigger source	generator, external TTL input
Frequency strategy	direct static
Frequency generator range	0.01 Hz - 1 MHz
Reference times	1, 8
Digital inputs (kΩ)	TTL programmable polarity, 2.7 to GND

Table A.9: Imager sCMOS camera general system specifications

Double shutter	Two images with 120 ns (min) interframe time
Exposure time	15 s - 100 ms
Digital output (bit)	16
Interface	Frame grabber CLHS optical for PCIe 4 x slot
Lens mount	F-mount (optional C-mount)
Number of pixels	2560 x 2160
Pixel size (m)	6.5 x 6.5
Active area (mm)	16.6 x 14.0
Spectral range (nm)	370 - 1100
Quantum efficiency	typ. 53% @ 532 nm
Full well capacity (e)	30,000
Readout noise (e)	< 3 @ 286 MHz
Frame rate (fps)	50

Table A.10: LaVision 3D calibration target specifications

	SFOV	LFOV
Type	058-5	204-15
Dimensions(mm)	58x58	204x204
Dot Distance(mm)	5	15
Dot Diameter(mm)	1.2	3.2
Level separation(mm)	1	3

Table A.11: EverGreen 200 laser general system specifications

Wavelength (nm)	532
Pulse repetition rate (Hz)	15
Energy (mJ)	200
Pulse-to-pulse energy stability	< 2 % RMS
Energy drift over 8 hours	10 %
Pulse width (ns)	≤ 10
Near field beam diameter (mm)	< 6.35
Beam divergence (mrad)	< 4
Shot to shot point stability (rad)	< 100
Far field beam overlap (rad)	$\pm$ 100
Near field beam overlap (rad)	$\pm$ 100
Polarization	linearly polarized, vertical
Spectral purity	> 98 %
Near field beam profile	flat-top, uniform

Table A.12: Velmex BiSlide specifications

	Streamwise (x)	Spanwise (y)	Vertical (z)
Model	MB10-1000-M10-33,20	MN10-0400-M02-31,20	MN10-0150-M02-21
Drive	Belt	Lead Screw	Lead Screw
Travel (in)	100	40	15
Motor	Vexta PK296	Vexta PK296	Vexta PK264
Controller	VXM-2	VXM-2	VXM-1
Advance per step (mm)	0.03mm	5m	5m
Repeatability (in)	0.0002	0.0002	0.0002
Straight line accuracy	0.0007 in/ 10 in	0.003 in/ entire travel	0.003 in/ entire travel

Figure A.2: Depiction of calibration target (not to scale)

A.2 Calibrations and Facility

Table A.13: Omega PX653-10D5V (0-10 "WC range) pressure sensor (S/N X16380087) calibration report.

Pressure ("WC)	Output VDC	% Error (Span)	% Error (BSL)
-0.00018	1.001	0.03%	0.00%
2.50069	2.002	0.04%	0.03%
4.99958	2.999	-0.02%	-0.01%
7.49821	3.997	-0.06%	-0.03%
9.99997	4.999	-0.01%	0.04%
7.49877	3.997	-0.06%	-0.03%
5.00019	2.999	-0.02%	-0.01%
2.50087	2.002	0.03%	0.02%
-0.00033	1.001	0.03%	-0.01%

Table A.14: EverGreen 200 laser system as-measured calibration and beam parameters, courtesy of Quantel by Lumibird.

	Laser 1	Laser 2
Wavelength (nm)	532	532
Energy (mJ)	211	202
Near Field Beam Diameter (mm)	5.54	5.60
Spec Energy Voltage (V)	602	583
Voltage Min (V)	497	487
Voltage Limit (V)	608	589
Q-Switch Delay (μs)	135	135

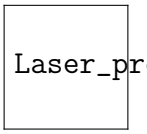


Figure A.3: Profile of Evergreen 200 laser. Both lasers shown.

Table A.15: PIV image calibrations for small and large fields of view (SFOV/LFOV).

	SFOV	LFOV
Mapping function	Pinhole	Pinhole
Fit error (pixels)	0.7439	0.1638
Size of dewarped image (pixels)	2571 x 2171	2567x2167
Camera translation offset (mm)	(9.756, -3.092)	(7.82, 20.75)
Camera rotation	(0.87, -1.07, 0.12)°	(-0.22,-0.33, 0.11)°
Image distortion principal point (pixels)	(904.52, 1038.09)	(1261.99, 1048.11)
Radial distortion, $\kappa 1$	0.711	0.0405
Radial distortion, $\kappa 2$	-55.63	0.8319
Tangential distortion, $\rho 1$	0.00034	$9.66x10^{-}5$
Tangential distortion, $\rho 2$	-0.0061	-0.00027
Camera focal length (mm)	215.38	63.38
Pixel size (mm)	.0065	0.0065
Pixel aspect ratio	1	1
Image origin (pixel)	(1303.58, 918.88)	(1352.39, 820.65)
Scale factor (pixel/mm)	40.35	11.14

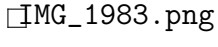


Figure A.4: Overview of LWST looking upstream.

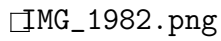


Figure A.5: Test Section in LWST without Bump Installed

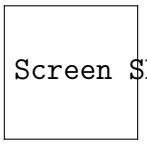


Figure A.6: LWST Wind Tunnel Schematic 1 [14]

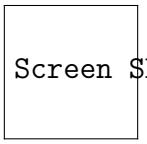


Figure A.7: LWST Wind Tunnel Schematic 2 [14]

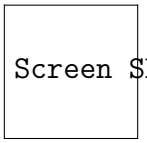


Figure A.8: LWST Wind Tunnel Schematic 3 [14]

A.3 Pressure Tap Locations

Figure A.9: Splitter plate and speed-bump pressure tap numbering system and locations.

Table A.16: Summary of pressure tap locations organized by spanwise plane number indicated in Figure A.9.

Tap No.	Plane	y/Lb	x/Lb	y/L	x/L
0	0	0.0000	-0.9261	0.0000	-0.9132
1		0.0000	-0.8415	0.0000	-0.8299
2		0.0000	-0.7570	0.0000	-0.7465
3		0.0000	-0.3750	0.0000	-0.3698
4		0.0000	-0.2500	0.0000	-0.2465
5		0.0000	-0.1250	0.0000	-0.1233
6		0.0000	-0.0625	0.0000	-0.0616
7		0.0000	0.0000	0.0000	0.0000
8		0.0000	0.0625	0.0000	0.0616
9		0.0000	0.1250	0.0000	0.1233
10		0.0000	0.1875	0.0000	0.1849
11		0.0000	0.2500	0.0000	0.2465
12		0.0000	0.3125	0.0000	0.3082
13		0.0000	0.3750	0.0000	0.3698
14		0.0000	0.4375	0.0000	0.4314
15		0.0000	0.5704	0.0000	0.5625
16		0.0000	0.6831	0.0000	0.6736
17		0.0000	0.7958	0.0000	0.7847
18		0.0000	0.9085	0.0000	0.8958
19		0.0000	1.0211	0.0000	1.0069
20		0.0000	1.1338	0.0000	1.1181
21	1	0.1250	0.0000	0.1233	0.0000
22		-0.1250	0.0000	-0.1233	0.0000
23		0.1250	0.1250	0.1233	0.1233

Table A.17: Pressure tap locations organized by spanwise plane number continued.

Tap No.	Plane	y/Lb	x/Lb	y/L	x/L
24		-0.1250	0.1250	-0.1233	0.1233
25		0.1250	0.2500	0.1233	0.2465
26		-0.1250	0.2500	-0.1233	0.2465
27		0.1250	0.3125	0.1233	0.3082
28		-0.1250	0.3125	-0.1233	0.3082
29		0.1250	0.3750	0.1233	0.3698
30		-0.1250	0.3750	-0.1233	0.3698
31	2	0.2500	-0.3750	0.2465	-0.3698
32		-0.2500	-0.3750	-0.2465	-0.3698
33		0.2500	0.0000	0.2465	0.0000
34		-0.2500	0.0000	-0.2465	0.0000
35		0.2500	0.1875	0.2465	0.1849
36		-0.2500	0.1875	-0.2465	0.1849
37	3	0.3125	0.0000	0.3082	0.0000
38		-0.3125	0.0000	-0.3082	0.0000
39	4	0.3750	0.0000	0.3698	0.0000
40		-0.3750	0.0000	-0.3698	0.0000
41	5	0.4625	-0.2500	0.4561	-0.2465
42		-0.4625	-0.2500	-0.4561	-0.2465
43		0.4625	0.0000	0.4561	0.0000
44		-0.4625	0.0000	-0.4561	0.0000
45		0.4625	0.2500	0.4561	0.2465
46		-0.4625	0.2500	-0.4561	0.2465
47	6	0.4750	0.0000	0.4684	0.0000
48		-0.4750	0.0000	-0.4684	0.0000

Appendix B

MATLAB REDUCTION SCRIPTS

Listing B.1: Import PIV from DaVis10 into MATLAB

```
%ImportDaVis10_BB_V8(PIV_Folder)

%Owen Williams, Hariprasad Annamalai, Matt Robbins

function ImportDaVis10_BB_V8_Fall2021(varargin)
close all

addpath('C:\Users\Lavision\Documents\MATLAB PIV Scripts\readimx-v2.1...
-win64');
addpath('C:\Users\Lavision\Documents\MATLAB PIV Scripts');

if isempty(varargin)
    Folder_PIV = uigetdir('','Select PIV test directory.');
else
    Folder_PIV = varargin{1};
end

saveFolder = strcat(Folder_PIV,'/MATLAB Processed/');
if exist(saveFolder, 'dir')~=7
    mkdir(saveFolder)
end

cd(Folder_PIV)

```

```
%% Determine case name
casename = extractAfter(Folder_PIV,regexp(Folder_PIV,'Vel*')-1);
fprintf('Currently processing: %s\n',casename)

%% Determine Small or Large FOV and set constants
if contains(Folder_PIV,'Small')
    SFOV = 1;
else
    SFOV = 0;
    max_station = 13;
end

if SFOV
    scale_factor = 38.44; %pixels/mm
    origin_shift_x = 315.24; %(pixels)
    origin_shift_y = 89.56; %(pixels)

    switch str2num(cell2mat(regexp(char(regexp(Folder_PIV,'Vel\d*...'
    ,'match')),'\d','match')))
        case 20
            dt = 7.04;
        case 30
            dt = 5.59;
        case 40
            dt = 5.33;
        case 60
            dt = 4.16;
    end


else
    scale_factor = 11.14; %pixels/mm
    origin_shift_x = 121.69*11.14; %114.03; %(pixels) %needs to be adjusted
```

```
58      origin_shift_y = 1343.5954 ; %112.52; %(pixels) %needs to be adjusted
59
60      switch str2num(cell2mat(regexp(char(regexp(Folder_PIV,'Vel\d*'...
61      ,'match')),'\d','match')))
62          case 40
63              dt = 25.35;
64          case 60
65              dt = 17.95;
66      end
67  end
68
69  fov_horz = 2560 / scale_factor; %(mm)
70  fov_vert = 2160 / scale_factor; %(mm)
71  z0 = 18*25.4; % (mm) z0 position for plate position=4
72
73  %% Load tunnel conditions/calibration information/test coordinates
74  load('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\vel_calib.mat');
75
76  coord_file = dir('**\PIV_Coords*.*');
77  coord = readmatrix(coord_file.name).*25.4; %(mm)
78
79  %% Bump definition (in)
80  L = 36;
81  L_b = 35.5;
82  x_0 = 0.195 * L_b;
83  y_0 = 0.06 * L_b;
84  h_0 = 0.085* L_b;
85  Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* (1 + ...
86  erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
87
88  %% Fix coordinates overlap issue that affected stitching - Fall 2021
89  old_dx = diff(coord(:,1));
90
```

```
91  if old_dx~= 0
92      new_dx = 2*old_dx(1) - fov_horz;
93  else
94      new_dx = 0;
95  end
96
97  %Station closest to top of bump (x = 0)
98  [~,stat_center] = min(abs(coord(:,1)));
99  % for i = stat_center-1:-1:1
100 %     coord(i,1) = coord(i+1,1)-new_dx;
101 % end
102 % for i = stat_center+1:length(coord)
103 %     coord(i,1) = coord(i-1,1)+new_dx;
104 % end
105 if casename == "Vel60_yPos0_Bump"
106     coord(:,1) = coord(:,1) - 20; %Only needed for LFOV. ...
107     %Bad offset measurement
108     coord(8:12,1) = coord(8,1)-fov_horz/4+33;
109 else
110     coord(:,1) = coord(:,1) - 20; %Only needed for LFOV. ...
111     %Bad offset measurement
112     coord(4:6,1) = coord(4,1)-fov_horz/4+33;
113     coord(7:9,1) = coord(9,1)-fov_horz/4+10;
114 end
115
116 %% Find actual z coordinates based on wall location in raw image
117
118 try
119     wall_offset = (2160-FindWall(Folder_PIV))./scale_factor; %(mm)
120     %Bottom of image to plate/bump apex
121     if contains(Folder_PIV,'OldImportCode','IgnoreCase',true)
122         fprintf('\nNo z coordinate correction applied (long sample).\n')
123     elseif contains(Folder_PIV,'inflow','IgnoreCase',true)
```

```
124         coord(:,3) = z0 + fov_vert/2 - wall_offset;
125     elseif contains(Folder_PIV,'Separation Bubble','IgnoreCase',true)
126         dz = (z0+fov_vert/2-wall_offset)-min(coord(:,3));
127         coord(:,3) = coord(:,3)+dz;
128     else
129         dz =(z0 + h_0*25.4 + fov_vert/2 - wall_offset) - max(coord(:,3))+...
130         1.5;
131         coord(:,3) = coord(:,3) + dz;
132         %coord(7:9,3) = coord(7:9,3)-0.1227;
133     end
134 
135 catch
136     fprintf('\nNo z coordinate correction applied.\n')
137 end
138 
139 %% Find cycle folder paths
140 ind = 1;
141 dir_loop = dir('Frame*.*');
142 if exist('max_station','var')
143     if max_station*2 < length(dir_loop)
144         k = max_station*2;
145     else
146         k = length(dir_loop);
147     end
148 else
149     k = length(dir_loop);
150 end
151 
152 for i = 1:k
153     if dir_loop(i).isdir
154         dir_piv = dir([dir_loop(i).folder '\' dir_loop(i)....
155         name'\SubOverTimeMin_sl=49\PIV*']);
156 
```

```
157         if contains(Folder_PIV,'long','IgnoreCase',true) %Long sample...
158         %Append PIV processed sets
159             for j = 1:length(dir_piv)
160                 if dir_piv(j).isdir
161                     P(ind).DataPath = [dir_piv(j).folder '\' dir_piv(j)...
162                     .name'\PostProc'];
163                     ind = ind + 1;
164                 end
165             end
166         else %Most tests. Take the most recent PIV processing set
167             P(ind).DataPath = [dir_piv(end-1).folder '\' dir_piv(end-1)....
168             name'\PostProc'];
169             ind = ind + 1;
170         end
171 
172         clear dir_piv
173     end
174 end
175 
176 
177 %% Process vector fields
178 for pos = 1:length(P)
179     fprintf('\tReading field tunnel conditions %d of %d...\n',pos,length(P))
180     cd(P(pos).DataPath)
181     cd ../../../
182     try
183         load('Tunnel Conditions/tunnel_cond.mat');
184         %Calibrated freestream velocity
185         Uinf = ((tunnel_cond(end).Pomegamean * px_calib.coeff(1)...
186             + px_calib.coeff(2)).*2./mean([tunnel_cond(end).rho])).^0.5;
187         tunnel_cond(end).Vcal = Uinf;
188         save('Tunnel Conditions/tunnel_cond.mat','tunnel_cond');
189     catch
```

```
190         warning('Tunnel conditions data not present')
191         Uinf= str2num(cell2mat(regexp(char(regexp(Folder_PIV,'Vel\d*'...
192         ,'match')),'\d','match')));
193     end
194
195     fprintf('\tReading field %d of %d...\n',pos,length(P))
196
197     d = dir([P(pos).DataPath '/*.vc7']);
198     d = {d.name};
199     numpics = length(d);
200
201     oldFolder = cd(P(pos).DataPath);
202     if numpics == 0
203         disp('No suitable files found')
204         cd(oldFolder)
205         return
206     end
207
208     %Take in span,stream, Z of PIV coordinate from test inputs
209     stream_coord = coord(pos,1);
210     span_coord = coord(pos,2);
211     z_coord = coord(pos,3);
212
213     %Initialize arrays
214     M = readimx(char(d(1)));
215     D = showimx(M.Frames{1});
216     U = zeros(size(D.U')); W = zeros(size(D.V'));
217     clear M D
218
219     for pict=1:numpics
220         M = readimx(char(d(pict)));
221         D = showimx(M.Frames{1});
222
```

```
223         if pict ==1
224             X = D.X';
225             Z = D.Y'; %Wall-normal in Davis
226         end
227 
228         U(:,:,pict) = D.U';
229         W(:,:,pict) = D.V';
230 
231         clear M D
232     end
233     close gcf
234     clear pict
235 
236     cd(oldFolder)
237 
238     %Convert to m
239     X = X/1000;
240     Z = Z/1000;
241 
242     %Shift origin from DaVis to line up y to 0
243     X = X+origin_shift_x/scale_factor/1000;
244     Z = Z+origin_shift_y/scale_factor/1000;
245 
246     %Offest data using PIV position inputs
247     X = X+(-stream_coord/1000)-fov_horz/2/1000;
248     Z = Z+((z_coord-z0)/1000)-fov_vert/2/1000;
249 
250 
251     %Flip Signs
252     X = -1.*X;
253     U = -1.*U;
254 
255     %Remove Zeros
```

```
256     Vel = (U.^2.+W.^2).^0.5;
257     U(Vel==0)=NaN;
258     W(Vel==0)=NaN;
259
260     %Remove Edges
261     trim = 20;
262         Z = Z(trim:end-trim/2,trim+5:end-trim-5);
263         X = X(trim:end-trim/2,trim+5:end-trim-5);
264         U = U(trim:end-trim/2,trim+5:end-trim-5,:);
265         W = W(trim:end-trim/2,trim+5:end-trim-5,:);
266
267     %Populate into data structure
268     P(pos).U = U/Uinf;
269     P(pos).W = W/Uinf;
270     P(pos).X = X;
271     P(pos).Z = Z;
272     P(pos).Input.Stream = stream_coord;
273     P(pos).Input.Span = span_coord;
274     P(pos).Input.Wall_Normal = z_coord;
275     P(pos).Input.Units = 'mm';
276     P(pos).Uinfy = Uinf;
277     Uinfinity(pos).Uinfy= Uinf;
278
279     clear U W d Umean Wmean Uvar Wvar Ustd Wstd UWprime pTKE TSS ...
280     rho_UW present_vectors_normalized present_vectors
281     clear U_Umean U_Wmean U_Speed U_Uvar U_Wvar U_Ustd U_Wstd U_UWprime...
282     U_TSS U_pTKE
283 end
284
285 %% Combine Frame and Batch data
286 if casename == "Vel40_yPos0_Bump_LongSample"
287     P_all(1).U = cat(3,P(1).U,P(2).U);
288     P_all(1).W = cat(3,P(1).W,P(2).W);
```

```
289      P_all(1).X = P(1).X;
290      P_all(1).Z = P(1).Z;
291      P_all(1).Input.Stream = P(1).Input.Stream;
292      P_all(1).Input.Span = P(1).Input.Span;
293      P_all(1).Input.Wall_Normal = P(1).Input.Wall_Normal;
294      P_all(1).Input.Units = 'mm';
295
296      for i =3:length(P)
297          P_all(1).U = cat(3,P(i).U,P_all(1).U);
298
299          P_all(1).W = cat(3,P(i).W,P_all(1).W);
300      end
301  elseif casename == "Vel60_yPos0_Bump"
302      for i=1:7
303          if i ==1
304              P_all(1).U = cat(3,P(i).U,P(i+1).U);
305              P_all(1).W = cat(3,P(i).W,P(i+1).W);
306          elseif i ==2
307              continue
308          else
309              P_all(1).U = cat(3,P(i).U,P_all(1).U);
310              P_all(1).W = cat(3,P(i).W,P_all(1).W);
311          end
312      end
313
314      P_all(1).X = P(1).X;
315      P_all(1).Z = P(1).Z;
316      P_all(1).Input.Stream = P(1).Input.Stream;
317      P_all(1).Input.Span = P(1).Input.Span;
318      P_all(1).Input.Wall_Normal = P(1).Input.Wall_Normal;
319      P_all(1).Input.Units = 'mm';
320
321       for i=8:12
```

```
322         if i ==8
323             P_all(2).U = cat(3,P(i).U,P(i+1).U);
324             P_all(2).W = cat(3,P(i).W,P(i+1).W);
325         elseif i==9
326             continue
327         else
328             P_all(2).U = cat(3,P(i).U,P_all(2).U);
329             P_all(2).W = cat(3,P(i).W,P_all(2).W);
330         end
331     end
332
333     P_all(2).X = P(10).X;
334     P_all(2).Z = P(10).Z;
335     P_all(2).Input.Stream = P(10).Input.Stream;
336     P_all(2).Input.Span = P(10).Input.Span;
337     P_all(2).Input.Wall_Normal = P(10).Input.Wall_Normal;
338     P_all(2).Input.Units = 'mm';
339
340 else
341     g = 1;
342     h = 3;
343
344     P_all(1).U = cat(3,P(g).U,P(g+1).U);
345     P_all(1).U = cat(3,P(h).U,P_all(1).U);
346
347     P_all(1).W = cat(3,P(g).W,P(g+1).W);
348     P_all(1).W = cat(3,P(h).W,P_all(1).W);
349
350     P_all(1).X = P(g).X;
351     P_all(1).Z = P(g).Z;
352     P_all(1).Input.Stream = P(g).Input.Stream;
353     P_all(1).Input.Span = P(g).Input.Span;
354     P_all(1).Input.Wall_Normal = P(g).Input.Wall_Normal;
```

```
355     P_all(1).Input.Units = 'mm';
356
357     if round(P_all(1).Input.Span) == 55 || round(P_all(1).Input.Span) == 112
358         g=4;
359         h=6;
360
361         P_all(2).U = cat(3,P(g).U,P(g+1).U);
362         P_all(2).U = cat(3,P(h).U,P_all(2).U);
363
364         P_all(2).W = cat(3,P(g).W,P(g+1).W);
365         P_all(2).W = cat(3,P(h).W,P_all(2).W);
366
367         P_all(2).X = P(g).X;
368         P_all(2).Z = P(g).Z;
369         P_all(2).Input.Stream = P(g).Input.Stream;
370         P_all(2).Input.Span = P(g).Input.Span;
371         P_all(2).Input.Wall_Normal = P(g).Input.Wall_Normal;
372         P_all(2).Input.Units = 'mm';
373
374         g=7;
375         h=9;
376
377         P_all(3).U = cat(3,P(g).U,P(g+1).U);
378         P_all(3).U = cat(3,P(h).U,P_all(3).U);
379
380         P_all(3).W = cat(3,P(g).W,P(g+1).W);
381         P_all(3).W = cat(3,P(h).W,P_all(3).W);
382
383         P_all(3).X = P(g).X;
384         P_all(3).Z = P(g).Z;
385         P_all(3).Input.Stream = P(g).Input.Stream;
386         P_all(3).Input.Span = P(g).Input.Span;
387         P_all(3).Input.Wall_Normal = P(g).Input.Wall_Normal;
```

```
388         P_all(3).Input.Units = 'mm';
389
390     else
391
392         g=4;
393         h=7;
394
395         P_all(2).U = cat(3,P(g).U,P(g+1).U);
396         P_all(2).U = cat(3,P(h-1).U,P_all(2).U);
397         P_all(2).U = cat(3,P(h).U,P_all(2).U);
398
399         P_all(2).W = cat(3,P(g).W,P(g+1).W);
400         P_all(2).W = cat(3,P(h-1).W,P_all(2).W);
401         P_all(2).W = cat(3,P(h).W,P_all(2).W);
402
403         P_all(2).X = P(g).X;
404         P_all(2).Z = P(g).Z;
405         P_all(2).Input.Stream = P(g).Input.Stream;
406         P_all(2).Input.Span = P(g).Input.Span;
407         P_all(2).Input.Wall_Normal = P(g).Input.Wall_Normal;
408         P_all(2).Input.Units = 'mm';
409
410         g=8;
411         h=10;
412
413         P_all(3).U = cat(3,P(g).U,P(g+1).U);
414         P_all(3).U = cat(3,P(h).U,P_all(3).U);
415
416         P_all(3).W = cat(3,P(g).W,P(g+1).W);
417         P_all(3).W = cat(3,P(h).W,P_all(3).W);
418
419         P_all(3).X = P(g).X;
420         P_all(3).Z = P(g).Z;
```

```
421         P_all(3).Input.Stream = P(g).Input.Stream;
422         P_all(3).Input.Span = P(g).Input.Span;
423         P_all(3).Input.Wall_Normal = P(g).Input.Wall_Normal;
424         P_all(3).Input.Units = 'mm';
425 
426     end
427 end
428 
429 %% Calculate Statistics
430 for pos = 1:length(P_all)
431     dim = size(P_all(pos).U);
432     Umean = nanmean(P_all(pos).U,3);
433     if contains(Folder_PIV,'long','IgnoreCase',false)
434         if pos == 1 || pos ==2
435             for i =1:75
436                 for j = 1:dim(1)
437                     if (isnan(Umean(j,i,:)) && ~isnan(Umean(j,i+1,:)))
438                         if j==1
439                             P_all(pos).U(j:j+5,i:i+10,:) = NaN;
440                             P_all(pos).W(j:j+5,i:i+10,:) = NaN;
441                         elseif j==dim(1)
442                             P_all(pos).U(j,i:i+10,:) = NaN;
443                             P_all(pos).W(j,i:i+10,:) = NaN;
444                         else
445                             P_all(pos).U(j-1:j+5,i-5:i+10,:) = NaN;
446                             P_all(pos).W(j-1:j+5,i-5:i+10,:) = NaN;
447                         end
448 
449                         break;
450                     end
451                 end
452             end
453         end
```

```
454     end
455
456
457
458     Umean = nanmean(P_all(pos).U,3);      %Mean streamwise velocity
459     Wmean = nanmean(P_all(pos).W,3);      %Mean normal velocity
460     Speed = sqrt(Umean.^2 + Wmean.^2);    %Total vector magnitude
461     Uvar = nanvar(P_all(pos).U,[],3); %Streamwise velocity variance; Rxx
462     Wvar = nanvar(P_all(pos).W,[],3); %Normal velocity variance; Rzz
463     Ustd = nanstd(P_all(pos).U,[],3);%Streamwise velocity standard deviation
464     Wstd = nanstd(P_all(pos).W,[],3);%Normal velocity standard deviation
465
466     %Calculate covariance; Reynolds shear stress
467     Uprime = zeros(dim);
468     Wprime = zeros(dim);
469     UW = zeros(dim);
470
471     for j=1:dim(3)
472         Uprime(:,:,j) = P_all(pos).U(:,:,j)-Umean(:,:);
473         Wprime(:,:,j) = P_all(pos).W(:,:,j)-Wmean(:,:);
474         UW(:,:,j) = Uprime(:,:,j).*Wprime(:,:,j);
475     end
476
477     UWprime = nanmean(UW,3);%(Streamwise/Normal velocity covariance; Rxz
478     rho_UW = UWprime./(Ustd.*Wstd); %(-) Streamwise/Normal velocity...
479     correlation coefficient
480
481     pTKE = (Uvar + Wvar)./2;   %Planar turbulent kinetic energy
482     TSS = sqrt(0.25.*(Wvar-Uvar).^2 + UWprime.^2); %Turbulent shear stress
483
484     %Calculate uncertainty
485     U_Umean = Ustd.*sqrt(1/dim(3)); %Streamwise mean velocity uncertainty
486     U_Wmean = Wstd.*sqrt(1/dim(3)); %Normal mean velocity uncertainty
```

```
487     U_Speed = sqrt((Umean.*U_Umean./Speed).^2 + (Wmean.*U_Wmean./Speed...
488     ).^2); %Total velocity magnitude uncertainty
489     U_Ustd = Ustd.*sqrt(1/(2*(dim(3)-1)))...
490     ; %Streamwise velocity standard deviation uncertainty
491     U_Wstd = Wstd.*sqrt(1/(2*(dim(3)-1)))...
492     ; %Normal velocity standard deviation uncertainty
493     U_Uvar = Uvar.*sqrt(2/(dim(3)-1)); %Rxx uncertainty
494     U_Wvar = Wvar.*sqrt(2/(dim(3)-1)); %Rzz uncertainty
495     U_UWprime = Ustd.*Wstd.*sqrt((1+rho_UW.^2)/(dim(3)-1)); %Rxz uncertainty
496     U_pTKE = sqrt((0.5.*U_Uvar).^2 + (0.5.*U_Wvar).^2); %pTKE uncertainty
497     U_TSS = (1./TSS).*sqrt((1/16).*(Uvar-Wvar).^2.*(U_Uvar.^2+U_Wvar.^2)+...
498         (UWprime.^2 .* U_UWprime.^2)); %Turbulent shear stress uncertainty
499
500
501     Stats(pos).X = P_all(pos).X;
502     Stats(pos).Z = P_all(pos).Z;
503     Stats(pos).U = Umean;
504     Stats(pos).W = Wmean;
505     Stats(pos).Speed = Speed;
506     Stats(pos).Ustd = Ustd;
507     Stats(pos).Wstd = Wstd;
508     Stats(pos).u2 = Uvar;
509     Stats(pos).w2 = Wvar;
510     Stats(pos).uw = UWprime;
511     Stats(pos).pTKE = pTKE;
512     Stats(pos).TSS = TSS;
513     Stats(pos).U_U = U_Umean;
514     Stats(pos).U_W = U_Wmean;
515     Stats(pos).U_Speed = U_Speed;
516     Stats(pos).U_Ustd = U_Ustd;
517     Stats(pos).U_Wstd = U_Wstd;
518     Stats(pos).U_u2 = U_Uvar;
519     Stats(pos).U_w2 = U_Wvar;
```

```
520     Stats(pos).U_uw = U_UWprime;
521     Stats(pos).U_pTKE = U_pTKE;
522     Stats(pos).U_TSS = U_TSS;
523 end
524 %% Create interpolant objects
525 fprintf('\tCreating interpolant objects...\n')
526 
527 % Meshgrid to coordinate vectors
528 ind = 1;
529 [Ny, Nx] = size(Stats(1).X);
530 for i = 1:length(P_all)
531     for j = 1:Ny
532         for k = 1:Nx
533             if ~isnan(Stats(i).U(j,k))
534                 VecX(ind) = Stats(i).X(j,k);
535                 VecZ(ind) = Stats(i).Z(j,k);
536                 VecU(ind) = Stats(i).U(j,k);
537                 VecW(ind) = Stats(i).W(j,k);
538                 VecSpeed(ind) = Stats(i).Speed(j,k);
539                 VecUstd(ind) = Stats(i).Ustd(j,k);
540                 VecWstd(ind) = Stats(i).Wstd(j,k);
541                 Vecu2(ind) = Stats(i).u2(j,k);
542                 Vecw2(ind) = Stats(i).w2(j,k);
543                 Vecuw(ind) = Stats(i).uw(j,k);
544                 VecpTKE(ind) = Stats(i).pTKE(j,k);
545                 VecTSS(ind) = Stats(i).TSS(j,k);
546 
547                 VecU_U(ind) = Stats(i).U_U(j,k);
548                 VecU_W(ind) = Stats(i).U_W(j,k);
549                 VecU_Speed(ind) = Stats(i).U_Speed(j,k);
550                 VecU_Ustd(ind) = Stats(i).U_Ustd(j,k);
551                 VecU_Wstd(ind) = Stats(i).U_Wstd(j,k);
552                 VecU_u2(ind) = Stats(i).U_u2(j,k);
```

```
553                 VecU_w2(ind) = Stats(i).U_w2(j,k);
554                 VecU_uw(ind) = Stats(i).U_uw(j,k);
555                 VecU_pTKE(ind) = Stats(i).U_pTKE(j,k);
556                 VecU_TSS(ind) = Stats(i).U_TSS(j,k);
557                 ind = ind + 1;
558             end
559         end
560     end
561 end
562
563
564 FullField.U = scatteredInterpolant(VecX',VecZ',VecU','linear','none');
565 FullField.W = scatteredInterpolant(VecX',VecZ',VecW','linear','none');
566 FullField.Speed = scatteredInterpolant(VecX',VecZ',VecSpeed','linear'...
567 ,'none');
568 FullField.Ustd = scatteredInterpolant(VecX',VecZ',VecUstd','linear'...
569 ,'none');
570 FullField.Wstd = scatteredInterpolant(VecX',VecZ',VecWstd','linear'...
571 ,'none');
572 FullField.u2 = scatteredInterpolant(VecX',VecZ',Vecu2','linear','none');
573 FullField.w2 = scatteredInterpolant(VecX',VecZ',Vecw2','linear','none');
574 FullField.uw = scatteredInterpolant(VecX',VecZ',Vecuw','linear','none');
575 FullField.pTKE = scatteredInterpolant(VecX',VecZ',VecpTKE','linear','none');
576 FullField.TSS = scatteredInterpolant(VecX',VecZ',VecTSS','linear','none');
577
578 FullField.U_U = scatteredInterpolant(VecX',VecZ',VecU_U','linear','none');
579 FullField.U_W = scatteredInterpolant(VecX',VecZ',VecU_W','linear','none');
580 FullField.U_Speed = scatteredInterpolant(VecX',VecZ',VecU_Speed',...
581 'linear','none');
582 FullField.U_Ustd = scatteredInterpolant(VecX',VecZ',VecU_Ustd',...
583 'linear','none');
584 FullField.U_Wstd = scatteredInterpolant(VecX',VecZ',VecU_Wstd',...
585 'linear','none');
```

```
586 FullField.U_u2 = scatteredInterpolant(VecX',VecZ',VecU_u2',...
587 'linear','none');
588 FullField.U_w2 = scatteredInterpolant(VecX',VecZ',VecU_w2',...
589 'linear','none');
590 FullField.U_uw = scatteredInterpolant(VecX',VecZ',VecU_uw',...
591 'linear','none');
592 FullField.U_pTKE = scatteredInterpolant(VecX',VecZ',VecU_pTKE',...
593 linear','none');
594 FullField.U_TSS = scatteredInterpolant(VecX',VecZ',VecU_TSS',...
595 'linear','none');
596
597 %% Plotting prep
598
599 saveFolder_plots_fig = strcat(saveFolder,'Plots/Fig/');
600 saveFolder_plots_png = strcat(saveFolder,'Plots/Png/');
601
602 if exist(saveFolder_plots_fig, 'dir')~=7
603     mkdir(saveFolder_plots_fig)
604 end
605
606 if exist(saveFolder_plots_png, 'dir')~=7
607     mkdir(saveFolder_plots_png)
608 end
609
610 fprintf('\tStarting to Plot...\n')
611
612 % QC: Stitched data (non-interpolated)
613 figure(2)
614 quiver(VecX,VecZ,VecU,VecW)
615 hold on
616
617 %plot FOV edges
618 if casename ~= "Vel60_yPos0_Bump"
```

```
619     for i = 1:length(P_all)
620         rectangle('Position',[(coord(3*i,1)-fov_horz/2)/1000,...
621             (coord(3*i,3)-z0-fov_vert/2)/1000,fov_horz/1000,fov_vert/1000]);
622     end
623 else
624      for i = 1:length(P_all)
625         rectangle('Position',[(coord(4*i,1)-fov_horz/2)/1000,...
626             (coord(4*i,3)-z0-fov_vert/2)/1000,fov_horz/1000,fov_vert/1000]);
627     end
628 end
629
630
631 xlabel('x (m)')
632 ylabel('z (m)')
633 savename = 'QC_StitchedFlowfield';
634 savefile = strcat(saveFolder, savename, '.png');
635 print(savefile, '-dpng');
636
637
638 %% Interpolated grid
639 dens = 1;       %pt/mm   (max 3 pt/mm)
640 nx = ceil((max(VecX)-min(VecX))*1000*dens); nz = ceil((max(VecZ)-...
641 min(VecZ))*1000*dens);
642 [x_plt,z_plt] = meshgrid(linspace(min(VecX),max(VecX),nx),...
643     fliplr(linspace(min(VecZ),max(VecZ),nz)));  %(m)
644
645 %Create alphashape of non-masked points
646 nonmasked_x = []; nonmasked_z = [];
647
648 for i = 1:length(Stats)
649     nonmasked_x = [nonmasked_x; Stats(i).X(~isnan(Stats(i).U))];
650     nonmasked_z = [nonmasked_z; Stats(i).Z(~isnan(Stats(i).U))];
651 end
```

```
652
653 nonmasked_shape = alphaShape(nonmasked_x, nonmasked_z);
654
655 %Trim values of interpolated grid in masked area and outside FOV
656 for i = 1:nx
657     for j = 1:nz
658         if ~inShape(nonmasked_shape,x_plt(j,i),z_plt(j,i))
659             x_plt(j,i) = nan; z_plt(j,1) = nan;
660         end
661     end
662 end
663
664 %Create bump/plate line
665 z_0 = 0;
666 x_wall = linspace(min(VecX),max(VecX),nx);
667 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
668 z_wall = zeros(1,length(x_wall));
669 z_wall(1:min(bump_ind)-1) = z_0;
670 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
671 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
672 ones(1,length(bump_ind)).*coord(1,2)*0.0393701);
673
674 %% Plotting
675
676 %Inflow plotting
677 if contains(Folder_PIV,'inflow','IgnoreCase',true)
678
679     for i = 1:length(Stats)
680         figure
681         colormap(jet(100))
682         pcolor(Stats(i).X./(L*0.0254),Stats(i).Z./(L*0.0254),Stats(i).U)
683         shading interp
684         d=colorbar;
```

```
685         caxis([1/2, 1.25])
686         ylabel(d, '$\overline{U}/U_\infty$', 'interpreter', 'latex')
687         d.FontSize = get(gca,'FontSize');
688         d.TickLabelInterpreter = 'latex';
689         drawnow
690         hold on
691         plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',1)
692         xlabel('$x/L$', 'interpreter', 'latex','FontSize',...
693         get(gca,'FontSize'))
694         ylabel('$z/L$', 'interpreter', 'latex','FontSize',...
695         get(gca,'FontSize'))
696         xlim([min(min(x_plt))/(L*0.0254)-0.001 max(max(x_plt))/...
697         (L*0.0254)+0.001])
698         ylim([min(min(z_plt))/(L*0.0254)-0.001 max(max(z_plt))/...
699         (L*0.0254)+0.001])
700
701         savename = strcat(casename,'_Streamwise_Velocity_Station',num2str(i));
702         savefile = strcat(saveFolder_plots_fig, savename, '.fig');
703         savefig(savefile)
704         savefile = strcat(saveFolder_plots_png, savename, '.png');
705         print(savefile,'-dpng');
706     end
707 else
708
709     figure
710     colormap(jet(100))
711     pcolor(x_plt./(L*0.0254),z_plt./(L*0.0254),FullField.U(x_plt,z_plt))
712     shading interp
713     axis equal
714     d=colorbar;
715     caxis([-7/P(1).Uinfy, 2])
716     ylabel(d, '$\overline{U}/U_\infty$', 'interpreter', 'latex')
717     d.FontSize = get(gca,'FontSize');
```

```
718     d.TickLabelInterpreter = 'latex';
719     drawnow
720     hold on
721     plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',1)
722     xlabel('$x/L$', 'interpreter', 'latex','FontSize',get(gca,'FontSize'))
723     ylabel('$z/L$', 'interpreter', 'latex','FontSize',get(gca,'FontSize'))
724     xlim([0 max(max(x_plt))/(L*0.0254)])
725     ylim([0 0.15])
726
727     savename = strcat(casename, '_Streamwise_Velocity');
728     savefile = strcat(saveFolder_plots_fig,savename,'.fig');
729     savefig(savefile)
730     savefile = strcat(saveFolder_plots_png, savename, '.png');
731     print(savefile,'-dpng');
732
733 %     xlim([-0.05 max(max(x_plt))/(L*0.0254)])
734 %     ylim([0 0.1368])
735 %
736 %     savename = strcat(casename, '_Streamwise_Velocity_Sep_Focused');
737 %     savefile = strcat(saveFolder_plots_fig, savename, '.fig');
738 %     savefig(savefile)
739 %     savefile = strcat(saveFolder_plots_png, savename, '.png');
740 %     print(savefile,'-dpng');
741 %
742
743     figure
744     colormap(jet(100))
745     pcolor(x_plt./(L*0.0254),z_plt./(L*0.0254),FullField.W(x_plt,z_plt))
746     shading interp
747     axis equal
748     d=colorbar;
749     caxis([-10/P(1).Uinfy, 0.4])
750     ylabel(d, '$\overline{W}/U_\infty$', 'interpreter', 'latex')
```

```
751     d.FontSize = get(gca,'FontSize');
752     d.TickLabelInterpreter = 'latex';
753     drawnow
754     hold on
755     plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',1)
756     xlabel('$x/L$', 'interpreter', 'latex','FontSize',get(gca,'FontSize'))
757     ylabel('$z/L$', 'interpreter', 'latex','FontSize',get(gca,'FontSize'))
758     xlim([0 max(max(x_plt))/(L*0.0254)])
759     ylim([0 0.15])
760
761     savename = strcat(casename, '_Normal_Velocity');
762     savefile = strcat(saveFolder_plots_fig, savename, '.fig');
763     savefig(savefile)
764     savefile = strcat(saveFolder_plots_png, savename, '.png');
765     print(savefile,'-dpng');
766
767 %     xlim([-0.05 max(max(x_plt))/(L*0.0254)])
768 %     ylim([0 0.1368])
769 %
770 %     savename = strcat(casename, '_Normal_Velocity_Sep_Focused');
771 %     savefile = strcat(saveFolder_plots_fig, savename, '.fig');
772 %     savefig(savefile)
773 %     savefile = strcat(saveFolder_plots_png, savename, '.png');
774 %     print(savefile,'-dpng');
775
776     figure
777     colormap(jet(50))
778     pcolor(x_plt./(L*0.0254),z_plt./(L*0.0254),(FullField.u2(x_plt...
779     ,z_plt)+FullField.w2(x_plt,z_plt))/2)
780     shading interp
781     axis equal
782     d=colorbar;
783     ylabel(d, '$[\overline{u}^2+\overline{w}^2]/2U_\infty^2$'...
```

```
784     ,'interpreter', 'latex')
785     d.FontSize = get(gca,'FontSize')-6;
786     d.TickLabelInterpreter = 'latex';
787     caxis([0 0.1])
788     drawnow
789     hold on
790     plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',1)
791     xlabel('$x/L$','interpreter', 'latex','FontSize',get(gca,'FontSize'))
792     ylabel('$z/L$','interpreter', 'latex','FontSize',get(gca,'FontSize'))
793     xlim([0 max(max(x_plt))/(L*0.0254)])
794     ylim([0 0.15])
795
796
797     savename = strcat(casename, '_TKE');
798     savefile = strcat(saveFolder_plots_fig, savename, '.fig');
799     savefig(savefile)
800     savefile = strcat(saveFolder_plots_png, savename, '.png');
801     print(savefile,'-dpng');
802
803 %     xlim([-0.05 max(max(x_plt))/(L*25.4/1000)])
804 %
805 %     savename = strcat(casename, '_TKE_Downstream');
806 %     savefile = strcat(saveFolder_plots_fig, savename, '.fig');
807 %     savefig(savefile)
808 %     savefile = strcat(saveFolder_plots_png, savename, '.png');
809 %     print(savefile,'-dpng');
810
811 %     xlim([min(min(x_plt))/(L*25.4/1000) 0.05])
812 %     ylim([0 0.1368])
813 %     caxis([0 0.0025])
814 %
815 %     savename = strcat(casename, '_TKE_Upstream');
816 %     savefile = strcat(saveFolder_plots_fig, savename, '.fig');
```

```
817 %     savefig(savefile)
818 %     savefile = strcat(saveFolder_plots_png, savename, '.png');
819 %     print(savefile,'-dpng');
820
821
822
823     figure
824     colormap(jet(50))
825     pcolor(x_plt./(L*0.0254),z_plt./(L*0.0254),-FullField.uw(x_plt,z_plt))
826     shading interp
827     axis equal
828     d=colorbar;
829     caxis([0 0.05])
830     ylabel(d, '$-\overline{uw}/U_\infty^2$','interpreter', 'latex');
831     d.FontSize = get(gca,'FontSize')-2;
832     d.TickLabelInterpreter = 'latex';
833     drawnow
834     hold on
835     plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',1)
836     xlabel('$x/L$','interpreter', 'latex','FontSize',get(gca,'FontSize'))
837     ylabel('$z/L$','interpreter', 'latex','FontSize',get(gca,'FontSize'))
838     xlim([0 max(max(x_plt))/(L*0.0254)])
839     ylim([0 0.15])
840
841     savename = strcat(casename, '_Re_Shear_Stress');
842     savefile = strcat(saveFolder_plots_fig, savename, '.fig');
843     savefig(savefile)
844     savefile = strcat(saveFolder_plots_png, savename, '.png');
845     print(savefile,'-dpng');
846
847
848 %     xlim([-0.05 max(max(x_plt))/(L*25.4/1000)])
849 %
```

```
850 %     savename = strcat(casename, '_Re_Shear_Stress_Downstream');
851 %     savefile = strcat(saveFolder_plots_fig, savename, '.fig');
852 %     savefig(savefile)
853 %     savefile = strcat(saveFolder_plots_png, savename, '.png');
854 %     print(savefile,'-dpng');
855 %     xlim([min(min(x_plt))/(L*25.4/1000) 0.05])
856 %     ylim([0 0.1368])
857 %     caxis([0 0.0005])
858 %
859 %     savename = strcat(casename, '_Re_Shear_Stress_Upstream');
860 %     savefile = strcat(saveFolder_plots_fig, savename, '.fig');
861 %     savefig(savefile)
862 %     savefile = strcat(saveFolder_plots_png, savename, '.png');
863 %     print(savefile,'-dpng');
864
865
866     fprintf('\tPlots Saved...\n')
867
868 end
869
870
871 %% Save reduced data
872
873 fprintf('\tSaving data...\n')
874
875 if SFOV
876     fileStats = 'SFOV_MATLAB_Processed_Stat.mat';
877     fileAll = 'SFOV_MATLAB_Processed.mat';
878 else
879     fileStats = 'LFOV_MATLAB_Processed_Stat.mat';
880     fileAll = 'LFOV_MATLAB_Processed.mat';
881 end
882
```

```
883 if exist('Tunnel Conditions/tunnel_cond.mat') ==2
884     save(strcat(saveFolder, casename, fileStats),'Stats',...
885         'FullField','coord','px_calib','tunnel_cond','-v7.3')
886     save(strcat(saveFolder, casename, fileAll),'P_all','Stats',...
887         'FullField','coord','px_calib','tunnel_cond','-v7.3')
888     save(strcat(saveFolder, casename, 'Uinf.mat'),'Uinfinity','-v7.3')
889 else
890     save(strcat(saveFolder, casename, fileStats),'Stats',...
891         'FullField','coord','px_calib','-v7.3')
892     save(strcat(saveFolder, casename, fileAll),'P_all','Stats',...
893         'FullField','coord','px_calib','-v7.3')
894     save(strcat(saveFolder, casename, 'Uinf.mat'),'Uinfinity','-v7.3')
895 end
896
897 fprintf('\tProcessing Complete...\n')
```

Listing B.2: Boundary Layer Property Calculations Richardson Uzun

```
1  clear
2  filename = uigetfile({'*.mat'}, 'Select POD mat file');
3  load(filename);
4  %%
5  %Calculate vorticity from vector fields
6  span = 0.5;
7  Uinf = 60;
8  start_pos = 1;
9  end_pos = 1;
10 %pos_size = length(P);
11 for pos =start_pos:end_pos
12     if pos == 7
13         dz = 0.00085; %normalized units, correction
14     else
15         dz = 0;
```

```
16      end
17
18       L = 36;
19  %     U = Stats(pos).U/60;
20  %     V = Stats(pos).W/60;
21  %     X = Stats(pos).X;
22  %     Y = Stats(pos).Z;
23  %
24  %     [curlz,cav] = curl(X/(L*0.0254),Y/(L*0.0254)+dz,U,V);
25  %     figure
26  %     H = pcolor(X/(L*0.0254),Y/(L*0.0254)+dz,curlz);
27      for i = 1:1000
28          U = P(pos).U(:,:,i)/Uinf;
29          V = P(pos).W(:,:,i)/Uinf;
30          X = Stats(pos).X;
31          Y = Stats(pos).Z;
32          %[curlz,cav] = curl(X/(L*0.0254),Y/(L*0.0254)+dz,U,V);
33          [r c] = size(U);
34
35          for p = 1:c-1
36              if p ==1
37                  V_del(:,p) = V(:,p+1)-V(:,p);
38                  X_del(:,p) = X(:,p+1)-X(:,p);
39                  V_grad(:,p) = V_del(:,p)./(X_del(:,p)/(L*0.0254));
40
41              elseif p == 2
42                  V_del(:,p) = V(:,p+1)-V(:,p-1);
43                  X_del(:,p) = X(:,p+1)-X(:,p);
44                  V_grad(:,p) = V_del(:,p)./(2*(X_del(:,p)/(L*0.0254)));
45
46
47
48              elseif p == c-1
```

```
49                  V_del(:,p) = V(:,p+1)-V(:,p-1);
50                  X_del(:,p) = X(:,p+1)-X(:,p);
51                  V_grad(:,p) = V_del(:,p)./(2*(X_del(:,p)/(L*0.0254)));
52
53              else
54                  V_del(:,p) = V(:,p-2)-8*V(:,p-1)+8*V(:,p+1)-V(:,p+2);
55                  X_del(:,p) = X(:,p+1)-X(:,p);
56                  V_grad(:,p) = V_del(:,p)./(12*(X_del(:,p)/(L*0.0254)));
57
58              end
59
60          end
61
62          for m = 1:r-1
63              if  m ==1
64                  U_del(m,:) = U(m+1,:)-U(m,:);
65                  Y_del(m,:) = Y(m+1,:)-Y(m,:);
66                  U_grad(m,:) = U_del(m,:)./(Y_del(m,:)/(L*0.0254));
67
68              elseif m == 2
69                  U_del(m,:) = U(m+1,:)-U(m-1,:);
70                  Y_del(m,:) = Y(m+1,:)-Y(m,:);
71                  U_grad(m,:) = U_del(m,:)./(2*(Y_del(m,:)/(L*0.0254)));
72
73              elseif m == r-1
74                  U_del(m,:) = U(m+1,:)-U(m-1,:);
75                  Y_del(m,:) = Y(m+1,:)-Y(m,:);
76                  U_grad(m,:) = U_del(m,:)./(2*(Y_del(m,:)/(L*0.0254)));
77
78              else
79                  U_del(m,:) = U(m-2,:)-8*U(m-1,:)+8*U(m+1,:)-U(m+2,:);
80                  Y_del(m,:) = Y(m+1,:)-Y(m,:);
81                  U_grad(m,:) = U_del(m,:)./(12*(Y_del(m,:)/(L*0.0254)));
```

```
82              end
83
84          end
85
86  %         U_grad(:,:) = U_grad(1:m,1:p);
87  %         V_grad(:,:) = V_grad(1:m,1:p);
88  %         X_grad(:,:) = X_grad(1:m,1:p);
89  %         Y_grad(:,:) = Y_grad(1:m,1:p);
90
91          curl_mat(:,:,i) = V_grad(1:m,1:p)-U_grad(1:m,1:p);
92
93          %curl_mat(:,:,i) = curlz;
94          %cav_mat(:,:,i) = cav;
95          clear U_grad V_grad
96      end
97
98      curlz_mean = mean(curl_mat,3,'omitnan');
99      %cav_mean = mean(cav_mat,3,'omitnan');
100
101     figure
102     %H = pcolor(X/(L*0.0254),Y/(L*0.0254)+dz,curlz);
103     %H = pcolor(X/(L*0.0254),Y/(L*0.0254)+dz,curlz_mean);
104     H = pcolor(X(1:m,1:p)/(L*0.0254),Y(1:m,1:p)/(L*0.0254)+dz,curlz_mean);
105
106     shading interp; grid off; colormap(jet)
107     d = colorbar;
108     caxis([-200 0])
109     ylabel(d, '$\omega/U_\infty$', 'interpreter', 'latex')
110     xlabel('$x/L$','interpreter','latex')
111     ylabel('$z/L$','interpreter','latex')
112     ax = gca;
113     hold(ax);
114
```

```
115     x_data = get(H,'xdata');
116     z_data = get(H,'ydata');
117     w_data = get(H,'CData');
118     %[M,c] = contour(x_data,z_data,w_data,[0,0],'ShowText','on');
119     [M,c] = contour(x_data,z_data,w_data,[-0.05,-0.05],'ShowText','on');
120     c.LineWidth = 1;
121     c.LineColor = 'white';
122
123     line = streamslice(X/(L*0.0254),Y/(L*0.0254)+dz,U,V,4);
124     set(line,'LineWidth',0.05);
125     set(line,'Marker','none');
126     set(line,'Color','black');
127     pause(0.5)
128
129     L = 36;
130     L_b = 35.5;
131     x_0 = 0.195 * L_b;
132     y_0 = 0.06 * L_b;
133     h_0 = 0.085* L_b;
134     Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .*...
135     (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
136     z_0 = 0;
137     x_wall = linspace(min(min(X)),max(max(X)),length(curlz_mean));
138     bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
139     z_wall = zeros(1,length(x_wall));
140     z_wall(1:min(bump_ind)-1) = z_0;
141     z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
142
143     z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,ones...
144     (1,length(bump_ind)).*span*0.0393701);
145
146     % Calculate local normal vectors to bump surface
147     L = 36;
```

```
148     L_b = 35.5;
149     x_0 = 0.195 * L_b;
150     y_0 = 0.06 * L_b;
151     h_0 = 0.085* L_b;
152     Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .*...
153     (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
154     z_0 = 0;
155     x_wall = linspace(min(min(X)),max(max(X)),length(curlz_mean));
156     bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
157     z_wall = zeros(1,length(x_wall));
158     z_wall(1:min(bump_ind)-1) = z_0;
159     z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
160
161     z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,ones...
162     (1,length(bump_ind)).*span*0.0393701);
163
164     hold on
165     plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
166
167     clear slope slope_normal
168     for i=2:length(z_wall)-1
169         x1 = x_wall(i-1)/(L*0.0254);
170         x2 = x_wall(i)/(L*0.0254);
171         x3 = x_wall(i+1)/(L*0.0254);
172
173         z1 = z_wall(i-1)./L;
174         z2 = z_wall(i)./L;
175         z3 = z_wall(i+1)./L;
176
177         m = (z3-z1)/(x3-x1);
178         slope(i) = m;
179
180         m_normal = -1/m;
```

```
181         slope_normal(i) = m_normal;
182
183         theta(i) = 90-abs(atan(m_normal)*180/pi);
184
185     end
186     %%
187     % Calculate thickness
188     clear endpoint_x endpoint_y end_index_x end_index_y x_bl y_bl
189     if pos ==4
190         [r c] = size(curlz_mean);
191         c = c-1;
192         for j = c:-1:1
193             for i = r:-1:1
194                 if isnan(curlz_mean(i,j)) && ~isnan(curlz_mean(i-1,j))
195                     y_bl = i-1;
196                     x_bl = j;
197
198                     if x_bl <= 0
199                         x_bl = NaN; %x_bl+1;
200                         y_bl = NaN; %y_bl + round(slope_normal(c));
201                         case_number = 2;
202                         break
203                     end
204
205                     if y_bl <= 0
206                         x_bl = NaN; %x_bl+1;
207                         y_bl = NaN; %y_bl + round(slope_normal(c));
208                         case_number = 2;
209                         break
210                     end
211
212                     while x_bl>0 && y_bl>0 && y_bl<= r && x_bl<=(c+1)
213                         if slope_normal(j) < 0
```

```
214                         x_bl = x_bl-1;
215                         y_bl = y_bl+round(slope_normal(j));
216                     else
217                         x_bl = x_bl-1;
218                         y_bl = y_bl-round(slope_normal(j));
219                         plot(x_bl,-y_bl,'*')
220                         hold on
221                     end
222
223                     if x_bl >= c+1
224                         x_bl = NaN; %x_bl-1;
225                         y_bl = NaN;% y_bl - round(slope_normal(c));
226                         case_number = 2;
227                         break
228                     end
229
230                     if x_bl <= 0
231                         x_bl = NaN; %x_bl+1;
232                         y_bl = NaN; %y_bl + round(slope_normal(c));
233                         case_number = 2;
234                         break
235                     end
236
237                     if y_bl <= 0
238                         x_bl = NaN; %x_bl+1;
239                         y_bl = NaN; %y_bl + round(slope_normal(c));
240                         case_number = 2;
241                         break
242                     end
243
244                     if y_bl >= r
245                         x_bl = NaN; %x_bl-1;
246                         y_bl = NaN; %y_bl-round(slope_normal(c));
```

```
247                         case_number = 2;
248                         break;
249                     end
250
251                     dist_local = sqrt((mean(X(:,x_bl))-...
252                     mean(X(:,j))).^2+(Y(y_bl,1)-Y(i,:)).^2);
253
254                     if slope_normal(j) < 0
255                         dist_Fmax = sqrt((mean(X(:,j+1))-...
256                         mean(X(:,j))).^2+(Y(i-1,1)-Y(i,:)).^2);
257                     else
258                         dist_Fmax = sqrt((mean(X(:,j-1))-...
259                         mean(X(:,j))).^2+(Y(i-1,1)-Y(i,:)).^2);
260                     end
261
262                     if slope_normal(j) < 0
263                         curl_max = curlz_mean(i-1,j+1);
264                     else
265                         curl_max = curlz_mean(i-1,j-1);
266                     end
267
268                     epsilon = 0.02;
269
270                     if -dist_local*curlz_mean(y_bl,x_bl) < -epsilon...
271                     *dist_Fmax*curl_max
272                         if slope_normal(j) < 0
273                             x_bl = x_bl;%+1;
274                             y_bl = y_bl;%-round(slope_normal(c));
275                             case_number = 1;
276                         else
277                             x_bl = x_bl-1;
278                             y_bl = y_bl+round(slope_normal(j));
279                             case_number = 1;
```

```
280                         end
281
282                         if x_bl == 0
283                             x_bl = NaN;
284                             case_number = 2;
285                         elseif y_bl == 0
286                             y_bl = NaN;
287                             case_number = 2;
288                         end
289
290                         if x_bl == 1
291                             case_number = 0;
292                         elseif y_bl == 1
293                             case_number = 0;
294                         end
295
296                         break
297                     end
298                 end
299
300                 end_index_x(j) = x_bl;
301                 end_index_y(j) = y_bl;
302
303                 if case_number == 1
304                     if slope_normal(j) < 0
305                         endpoint_y(j) = (Y(y_bl,1)+Y(y_bl-...
306                         round(slope_normal(j)),1))/2;
307                         endpoint_x(j) = (mean(X(:,x_bl))+...
308                         mean(X(:,x_bl+1)))/2;
309                     else
310                         endpoint_y(j) = (Y(y_bl,1)+Y(y_bl+...
311                         round(slope_normal(j)),1))/2;
312                         endpoint_x(j) = (mean(X(:,x_bl)+...
```

```
313                             mean(X(:,x_bl+1))))/2;
314                         end
315
316                     elseif case_number == 0
317                         endpoint_x(j) = mean(X(:,x_bl));
318                         endpoint_y(j) = Y(y_bl,1);
319                     else
320                         endpoint_x(j) = NaN;
321                         endpoint_y(j) = NaN;
322                     end
323
324                 end
325                 clear x_bl y_bl
326             end
327         end
328     end
329
330     if pos ==5
331         [r c] = size(curlz_mean);
332         c = c-1;
333         for j = c:-1:1
334             for i = r:-1:1
335                 if isnan(curlz_mean(i,j)) && ~isnan(curlz_mean(i-1,j))
336                     y_bl = i-1;
337                     x_bl = j;
338
339                     if x_bl <= 0
340                         x_bl = NaN; %x_bl+1;
341                         y_bl = NaN; %y_bl + round(slope_normal(c));
342                         case_number = 2;
343                         break
344                     end
345
```

```
346                     if y_bl <= 0
347                         x_bl = NaN; %x_bl+1;
348                         y_bl = NaN; %y_bl + round(slope_normal(c));
349                         case_number = 2;
350                         break
351                     end
352
353                     while x_bl>0 && y_bl>0 && y_bl<= r && x_bl<=(c+1)
354
355                         x_bl = x_bl;
356                         y_bl = y_bl-1;
357
358                         if x_bl >= c+1
359                             x_bl = NaN; %x_bl-1;
360                             y_bl = NaN;% y_bl - round(slope_normal(c));
361                             case_number = 2;
362                             break
363                         end
364
365                         if x_bl <= 0
366                             x_bl = NaN; %x_bl+1;
367                             y_bl = NaN; %y_bl + round(slope_normal(c));
368                             case_number = 2;
369                             break
370                         end
371
372                         if y_bl <= 0
373                             x_bl = NaN; %x_bl+1;
374                             y_bl = NaN; %y_bl + round(slope_normal(c));
375                             case_number = 2;
376                             break
377                         end
378
```

```
379                         if y_bl >= r
380                             x_bl = NaN; %x_bl-1;
381                             y_bl = NaN; %y_bl-round(slope_normal(c));
382                             case_number = 2;
383                             break;
384                         end
385
386                         dist_local = sqrt((mean(X(:,x_bl))-...
387                         mean(X(:,j))).^2+(Y(y_bl,1)-Y(i,:)).^2);
388
389
390                         if curlz_mean(y_bl,x_bl) >0
391                             if slope_normal(j) < 0
392                                 x_bl = x_bl;%+1;
393                                 y_bl = y_bl;%-round(slope_normal(c));
394                                 case_number = 1;
395                             else
396                                 x_bl = x_bl;
397                                 y_bl = y_bl;
398                                 case_number = 1;
399                             end
400
401                             if x_bl == 0
402                                 x_bl = NaN;
403                                 case_number = 2;
404                             elseif y_bl == 0
405                                 y_bl = NaN;
406                                 case_number = 2;
407                             end
408
409                             if x_bl == 1
410                                 case_number = 0;
411                             elseif y_bl == 1
```

```
412                                 case_number = 0;
413                             end
414
415                             break
416                         end
417                     end
418
419                     end_index_x(j) = x_bl;
420                     end_index_y(j) = y_bl;
421
422                     if case_number == 1
423                         if slope_normal(j) < 0
424                             endpoint_y(j) = (Y(y_bl,1));
425                             endpoint_x(j) = (mean(X(:,x_bl)));
426                         else
427                             endpoint_y(j) = (Y(y_bl,1));
428                             endpoint_x(j) = (mean(X(:,x_bl)));
429                         end
430
431                     elseif case_number == 0
432                         endpoint_x(j) = mean(X(:,x_bl));
433                         endpoint_y(j) = Y(y_bl,1);
434                     else
435                         endpoint_x(j) = NaN;
436                         endpoint_y(j) = NaN;
437                     end
438
439                 end
440                 clear x_bl y_bl
441             end
442         end
443     end
444
```

```
445
446     [r c] = size(curlz_mean);
447     c = c-1;
448     for j = c:-1:1
449
450         if pos ==4 || pos ==5
451             break
452         end
453
454         for i = r:-1:1
455             if isnan(curlz_mean(i,j)) && ~isnan(curlz_mean(i-1,j))
456                 y_bl = i-5;
457                 x_bl = j;
458
459                 if x_bl <= 0
460                     x_bl = NaN; %x_bl+1;
461                     y_bl = NaN; %y_bl + round(slope_normal(c));
462                     case_number = 2;
463                     break
464                 end
465
466                 if y_bl <= 0
467                     x_bl = NaN; %x_bl+1;
468                     y_bl = NaN; %y_bl + round(slope_normal(c));
469                     case_number = 2;
470                     break
471                 end
472
473                 while x_bl>0 && y_bl>0 && y_bl<= r && x_bl<=(c+1)
474                     if slope_normal(c) < 0
475                         x_bl = x_bl-1;
476                         y_bl = y_bl+round(slope_normal(c));
477                     else
```

```
478                         x_bl = x_bl+1;
479                         y_bl = y_bl-round(slope_normal(c));
480                     end
481
482                     if x_bl >= c+1
483                         x_bl = NaN; %x_bl-1;
484                         y_bl = NaN;% y_bl - round(slope_normal(c));
485                         case_number = 2;
486                         break
487                     end
488
489                     if x_bl <= 0
490                         x_bl = NaN; %x_bl+1;
491                         y_bl = NaN; %y_bl + round(slope_normal(c));
492                         case_number = 2;
493                         break
494                     end
495
496                     if y_bl <= 0
497                         x_bl = NaN; %x_bl+1;
498                         y_bl = NaN; %y_bl + round(slope_normal(c));
499                         case_number = 2;
500                         break
501                     end
502
503                     if y_bl >= r
504                         x_bl = NaN; %x_bl-1;
505                         y_bl = NaN; %y_bl-round(slope_normal(c));
506                         case_number = 2;
507                         break;
508                     end
509
510                     dist_local = sqrt((mean(X(:,x_bl))-mean(X(:,j))).^2+...
```

```
511                     (Y(y_bl,1)-Y(i,:)).^2);
512
513                     if slope_normal(j) <= 0
514                         dist_Fmax = sqrt((mean(X(:,j+1))-mean(X(:,j))).^2...
515                         +(Y(i-1,1)-Y(i,:)).^2);
516                     else
517                         dist_Fmax = sqrt((mean(X(:,j-1))-mean(X(:,j))).^2...
518                         +(Y(i-1,1)-Y(i,:)).^2);
519                     end
520
521                     if slope_normal(j) <= 0
522                         curl_max = curlz_mean(i-1,j+1);
523                     else
524                         curl_max = curlz_mean(i-1,j-1);
525                     end
526
527                     epsilon = 0.02;
528
529                     if -dist_local*curlz_mean(y_bl,x_bl) < -epsilon...
530                     *dist_Fmax*curl_max
531
532                         if slope_normal(c) < 0
533                             x_bl = x_bl;%+1;
534                             y_bl = y_bl;%-round(slope_normal(c));
535                             case_number = 1;
536                         else
537                             x_bl = x_bl;%-1;
538                             y_bl = y_bl;%-round(slope_normal(c));
539                             case_number = 1;
540                         end
541
542                         if x_bl == 0
543                             x_bl = NaN;
```

```
544                         case_number = 2;
545                     elseif y_bl == 0
546                         y_bl = NaN;
547                         case_number = 2;
548                     end
549
550                     if x_bl == 1
551                         case_number = 0;
552                     elseif y_bl == 1
553                         case_number = 0;
554                     end
555
556                     break
557                 end
558             end
559
560            end_index_x(j) = x_bl;
561             end_index_y(j) = y_bl;
562
563             if case_number == 1
564                 if slope_normal(c) < 0
565                     endpoint_y(j) = (Y(y_bl,1)+Y(y_bl-...
566                     round(slope_normal(c)),1))/2;
567                     endpoint_x(j) = (mean(X(:,x_bl))...
568                     +mean(X(:,x_bl+1)))/2;
569                 else
570                     endpoint_y(j) = (Y(y_bl,1)+Y(y_bl+...
571                     round(slope_normal(c)),1))/2;
572                     endpoint_x(j) = (mean(X(:,x_bl))+...
573                     mean(X(:,x_bl+1)))/2;
574                 end
575
576             elseif case_number == 0
```

```
577                     endpoint_x(j) = mean(X(:,x_bl));
578                     endpoint_y(j) = Y(y_bl,1);
579                 else
580                     endpoint_x(j) = NaN;
581                     endpoint_y(j) = NaN;
582                 end
583
584             end
585             clear x_bl y_bl
586         end
587     end
588
589
590
591     %%
592     hold on
593     plot(endpoint_x/(0.0254*L),endpoint_y/(L*0.0254),'*')
594
595      saveas(gcf,sprintf('Pcolor_frame_%d.png',pos))
596      close
597 %% Calcuate BL thickness in curvilinear coordinates
598     x_wall_normalized = flip((x_wall./(L*0.0254)));
599     z_wall_normalized = flip((z_wall./L));
600     endpoint_x_normalized = endpoint_x./(L*0.0254);
601     endpoint_y_normalized = endpoint_y./(L*0.0254);
602
603     for i = 1:c
604         BL_thickness(i) = sqrt((x_wall_normalized(i)-...
605         endpoint_x_normalized(i))^2+(z_wall_normalized(i)-...
606         endpoint_y_normalized(i))^2);
607     end
608
609     BL_thickness = flip(BL_thickness);
```

```
610
611     % Calculate radius of curvature
612     L = 36;
613     L_b = 35.5;
614     x_0 = 0.195 * L_b;
615     y_0 = 0.06 * L_b;
616     h_0 = 0.085* L_b;
617
618     Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) ...
619     .* (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
620     Z_bump_der_x1 = @(x_bump,y_bump) (h_0/2).*(-2.*x_bump/x_0.^2).*...
621     exp(-(x_bump./x_0).^2) .* (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
622     Z_bump_der_x2 = @(x_bump,y_bump) (h_0/2).*(-2/x_0.^2).* ...
623     exp(-(x_bump./x_0).^2) .* (1 + erf((L_b/2 - 2*y_0 - ...
624     abs(y_bump))./y_0)) + (h_0/2).*(4.*x_bump.^2./x_0.^4).* ...
625     exp(-(x_bump./x_0).^2) .* (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
626
627     z_0 = 0;
628     x_wall = linspace(min(min(X)),max(max(X)),length(curlz_mean));
629     %bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
630     bump_ind = 1:1:c+1;
631     z_wall = zeros(1,length(x_wall));
632     z_wall(1:min(bump_ind)-1) = z_0;
633     z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
634
635     z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
636     ones(1,length(bump_ind)).*span*0.0393701);
637     z_wall_der_x1 = Z_bump_der_x1(x_wall(1,bump_ind).*39.3701,...
638     ones(1,length(bump_ind)).*span*0.0393701);
639     z_wall_der_x2 = Z_bump_der_x2(x_wall(1,bump_ind).*39.3701,...
640     ones(1,length(bump_ind)).*span*0.0393701);
641
642     r_curve = (1+z_wall_der_x1.^2).^(3/2)./abs(z_wall_der_x2);
```

```
643
644     Boundary_layers_cal(pos).radius = r_curve(1:c);
645     Boundary_layers_cal(pos).layer_thickness = BL_thickness;
646     Boundary_layers_cal(pos).end_x = endpoint_x/(0.0254*L);
647     Boundary_layers_cal(pos).end_y = endpoint_y/(0.0254*L);
648     Boundary_layers_cal(pos).streamwise_loc = x_wall/(0.0254*L);
649     Boundary_layers_cal(pos).normal_loc = z_wall/(0.0254*L);
650
651     clear r_curve BL_thickness endpoint_x endpoint_y x_wall_normalized...
652     z_wall_normalized endpoint_x_normalized endpoint_y_normalized
653 end
654 %%
655 %for pos = 1:length(Boundary_layers_cal)
656 figure
657 for pos = 1:end_pos
658     hold on
659     plot(Boundary_layers_cal(pos)...
660     .radius/L,Boundary_layers_cal(pos).layer_thickness,'*')
661 end
662 xlim([0 50])
663 xlabel('$R/L$','interpreter','latex')
664 ylabel('$"\delta_\omega/L$"','interpreter','latex')
665 legend('Frame 1', 'Frame 2', 'Frame 3','Frame 4','Frame 5')
666
667 saveas(gcf,'BL_radius_curvature_comparison.png')
668 close
669
670 %%
671 figure
672 for pos = 1:end_pos
673     X = Stats(pos).X;
674     Y = Stats(pos).Z;
675     hold on
```

```
676     plot(Boundary_layers_cal(pos).end_x,Boundary_layers_cal(pos).end_y,'*')
677 end
678
679 L = 36;
680 L_b = 35.5;
681 x_0 = 0.195 * L_b;
682 y_0 = 0.06 * L_b;
683 h_0 = 0.085* L_b;
684 Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
685 (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
686 z_0 = 0;
687 x_wall = linspace(min(min(Stats(1).X)),max(max(Stats(pos).X)),...
688 length(curlz_mean));
689 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
690 z_wall = zeros(1,length(x_wall));
691 z_wall(1:min(bump_ind)-1) = z_0;
692 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
693
694 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
695 ones(1,length(bump_ind)).*span*0.0393701);
696
697 hold on
698 plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
699
700 xlabel('$x/L$','interpreter','latex')
701 ylabel('$z/L$','interpreter','latex')
702 legend('Frame 1', 'Frame 2', 'Frame 3','Frame 4','Frame 5')
703
704 saveas(gcf,'Boundary_layer_plot_over_bump.png')
705 close
706 %%
707 figure
708 for pos = 1:end_pos
```

```
709     X = Stats(pos).X;
710     Y = Stats(pos).Z;
711     hold on
712     plot(X(:,1:c)/(0.0254*L),flip(Boundary_layers_cal(pos)....
713     layer_thickness),'*')
714 end
715 xlabel('$x/L$','interpreter','latex')
716 ylabel('$"\delta_\omega/L$"','interpreter','latex')
717 legend('Frame 1', 'Frame 2', 'Frame 3','Frame 4','Frame 5')
718 saveas(gcf,'BL_thickness vs frame data.png')
719 close
720
721 %%
722 figure
723 for pos = start_pos:end_pos
724
725 plot(Boundary_layers_cal(pos).streamwise_loc(1:c),...
726 Boundary_layers_cal(pos).layer_thickness(1:c),'-k')
727 xlim([min(Boundary_layers_cal(start_pos).streamwise_loc) ...
728 max(Boundary_layers_cal(end_pos).streamwise_loc)])
729 hold on
730 end
731
732 xlabel('$x/L$','interpreter','latex')
733 ylabel('$"\delta_\omega/L$"','interpreter','latex')
734
735 saveas(gcf,'Boundary_layer_evolution_in_x.png')
736 close
```

Listing B.3: Conditionally Average PIV Fields by Reverse Flow Fraction

```
1 addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
2 Vel40_yPos0_Bump\MATLAB Processed')
```

```
addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Vel40_yPos0_Bump_LongSample\MATLAB Processed')
addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Vel40_yPos0.5_Bump\MATLAB Processed')
addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Vel40_yPos1_Bump\MATLAB Processed')
addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Vel60_yPos0_Bump\MATLAB Processed')

filename = uigetfile('.mat','Select Processed PIV .mat file');
load(filename)
%% Bump Properties
L = 36;
font = 12; %16 for presentations, 12 for papers
Uinf = 60;

for pos=1:length(P_all)
    U_frames = P_all(pos).U;
    W_frames = P_all(pos).W;
    X = P_all(pos).X;
    Z = P_all(pos).Z;

% uncomment below section if small fov separation bubble centerline 60m/s
%     if pos==8
%         U_frames = U_frames(:,65:end,:);
%         W_frames = W_frames(:,65:end,:);
%         X = X(:,65:end);
%         Z = Z(:,65:end);
%     end

    [r c d] = size(U_frames);
    U_normalized = U_frames;
    U_average = sum(U_normalized,3,'omitnan')/d;
```

```
36
37      W_normalized = W_frames;
38      W_average = sum(W_normalized,3,'omitnan')/d;
39
40      for i = 1:r
41          for j= 1:c
42              if U_average(i,j) == 0
43                  U_average (i,j) = NaN;
44                  W_average(i,j) = NaN;
45              end
46          end
47      end
48
49      for k=1:length(U_frames)
50          Atot = r*c;
51          Aneg = 0;
52          Apos = 0;
53          for i = 1:r
54              for j = 1:c
55                  if ~isnan(U_normalized(i,j,k))
56                      if U_normalized(i,j,k) < 0
57                          Aneg = Aneg+1;
58                      else
59                          Apos = Apos+1;
60                      end
61                  end
62              end
63          end
64          Area_total= Aneg+Apos;
65          Negative_area(k) = Aneg/(Area_total);
66          Positive_area(k) = Apos/(Area_total);
67      end
68      %% Historgram of Negative and Positive Flow Area Fractions
```

```
69      figure
70      neg_hist = histogram(Negative_area);
71      xlabel('Negative Flow Area Fraction Per Frame','interpreter',...
72      'latex','FontSize',font)
73      ylabel('Frames','interpreter','latex','FontSize',16)
74      neg_fig_name = strcat('Neg_Flow_Area_Fraction_Frame_',num2str(pos)...
75      ,'_Uinf_',num2str(Uinf),'.fig');
76      neg_png_name = strcat('Neg_Flow_Area_Fraction_Frame_',num2str(pos)...
77      ,'_Uinf_',num2str(Uinf),'.png');
78      savefig(neg_fig_name)
79      saveas(gcf,neg_png_name)
80      figure
81      pos_hist = histogram(Positive_area);
82      xlabel('Positive Flow Area Fraction Per Frame','interpreter',...
83      'latex','FontSize',font)
84      ylabel('Frames','interpreter','latex','FontSize',16)
85      pos_fig_name = strcat('Pos_Flow_Area_Fraction_Frame_',num2str(pos)...
86      ,'_Uinf_',num2str(Uinf),'.fig');
87      pos_png_name = strcat('Pos_Flow_Area_Fraction_Frame_',num2str(pos)...
88      ,'_Uinf_',num2str(Uinf),'.png');
89      savefig(pos_fig_name)
90      saveas(gcf,pos_png_name)
91
92      %% Extract Indicies of Velocity Fields in negative flow fraction bin
93      neg_bins = get(neg_hist,'BinEdges');
94      neg_counts = get(neg_hist,'BinCounts');
95      neg_indices = [];
96      for i = 1:(length(neg_bins)-1)
97          for j = 1:length(Negative_area)
98              if (Negative_area(j) >= neg_bins(i) && Negative_area(j) <...
99              neg_bins(i+1))
100                 neg_index = j;
101                 neg_indices = [neg_indices neg_index];
```

```
102             else
103
104             end
105         end
106     end
107
108     %% Group binned fields together and average fields
109     for p = 1:length(neg_counts)
110         if p == 1
111             groups = neg_indices(1:neg_counts(p));
112             U_conditional_avg(:,:,p) = sum(U_normalized(:,:,groups),3...
113             ,'omitnan')/length(groups);
114             W_conditional_avg(:,:,p) = sum(W_normalized(:,:,groups),3...
115             ,'omitnan')/length(groups);
116             U_cond_avg_std(:,:,p) = std(U_normalized(:,:,groups),0,3...
117             ,'omitnan');
118             W_cond_avg_std(:,:,p) = std(W_normalized(:,:,groups),0,3...
119             ,'omitnan');
120             for i = 1:r
121                 for j= 1:c
122                     if U_conditional_avg(i,j,p) == 0
123                         U_conditional_avg (i,j,p) = NaN;
124                         W_conditional_avg (i,j,p) = NaN;
125                         U_cond_avg_std(i,j,p) = NaN;
126                         W_cond_avg_std(i,j,p) = NaN;
127                     end
128                 end
129             end
130         else
131             groups = neg_indices(sum(neg_counts(1:p-1))+1:...
132             sum(neg_counts(1:p)));
133             U_conditional_avg(:,:,p) = sum(U_normalized(:,:,groups),3...
134             ,'omitnan')/length(groups);
```

```
135             W_conditional_avg(:,:,p) = sum(W_normalized(:,:,groups),3.
136             ...
137             ,'omitnan')/length(groups);
138             U_cond_avg_std(:,:,p) = std(U_normalized(:,:,groups),0,3...
139             ,'omitnan');
140             W_cond_avg_std(:,:,p) = std(W_normalized(:,:,groups),0,3...
141             ,'omitnan');
142             for i = 1:r
143                 for j= 1:c
144                     if U_conditional_avg(i,j,p) == 0
145                         U_conditional_avg (i,j,p) = NaN;
146                         W_conditional_avg (i,j,p) = NaN;
147                         U_cond_avg_std(i,j,p) = NaN;
148                         W_cond_avg_std(i,j,p) = NaN;
149                     end
150                 end
151             end
152         end
153     end
154 Conditionally_averaged(pos).U = U_conditional_avg;
155 Conditionally_averaged(pos).W = W_conditional_avg;
156 Conditionally_averaged(pos).Ustd = U_cond_avg_std;
157 Conditionally_averaged(pos).Wstd = W_cond_avg_std;
158 Conditionally_averaged(pos).X = X;
159 Conditionally_averaged(pos).Z = Z;
160 Conditionally_averaged(pos).negative_area = Negative_area;
161 Conditionally_averaged(pos).positive_area = Positive_area;
162 Conditionally_averaged(pos).bins = neg_bins;
163 Conditionally_averaged(pos).counts = neg_counts;
164 Conditionally_averaged(pos).indices = neg_indices;
165 clear Negative_area Positive_area neg_hist U_conditional_avg ...
166 W_conditional_avg groups neg_bins neg_counts ...
167 neg_indices U_cond_avg_std W_cond_avg_std
```

```
end
close all
%%
fileStats = strcat('Conditionally_Avg_', num2str(Uinf),'.mat');
save([fileStats],'Conditionally_averaged','-v7.3')
```

Listing B.4: Create p.d.f.s and CDFs of reverse flow fraction

```
clear
filename = uigetfile('select conditionally averaged field?');
load(filename)

%%
for i = 2
    hold on
    yyaxis left
    histogram(Conditionally_averaged(i).negative_area,...
    'Normalization','pdf'...
        ,'DisplayStyle','stairs')
    xlabel('$\alpha$','interpreter','latex')
    ylabel('p.d.f')

    hold on
    yyaxis right
    histogram(Conditionally_averaged(i).negative_area,...
    'Normalization','cdf'...
        ,'DisplayStyle','stairs')
    xlabel('$\alpha$','interpreter','latex')
    ylabel('CDF')

    xlim([0 0.35])
    set(gca,'FontSize',16)
    set(gca,'box','off')
```

```
26
27      hold on
28      legend(['pdf FOV ' num2str(i-1)],['CDF FOV ' num2str(i-1)])
29  end
30
31  set(gca,'FontSize',16)
32  %% Statistics
33  mean_negative_area = mean(Conditionally_averaged(2).negative_area)
34  median_negative_area = median(Conditionally_averaged(2).negative_area)
35  std_negative_area = std(Conditionally_averaged(2).negative_area)
36  max(Conditionally_averaged(2).negative_area)
```

Listing B.5: Create Conditionally Averaged PIV Fields and Automatic Determination of Separation and Reattachment Point

```
1  %% Large FOV, centerline, 40/s, LongSample, separation
2  clear all
3  clc
4  addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
5  Vel40_yPos0_Bump_LongSample\MATLAB Processed')
6
7  load('Conditionally_Avg_40.mat')
8  load('Vel40_yPos0_Bump_LongSampleLFOV_MATLAB_Processed.mat')
9
10 %% Large FOV, centerline, 40/s, 10000 frames, separation and reattachment
11 clear all
12 clc
13 addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
14 Vel40_yPos0_Bump\MATLAB Processed')
15
16 load('Conditionally_Avg_40.mat')
17 load('Vel40_yPos0_BumpLFOV_MATLAB_Processed.mat')
```

```
18
19 %% Large FOV, p0.5, 40/s, 10000 frames, separation and reattachment
20 clear all
21 clc
22 addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
23 Vel40_yPos0.5_Bump\MATLAB Processed')
24
25 load('Conditionally_Avg_40.mat')
26 load('Vel40_yPos0.5_BumpLFOV_MATLAB_Processed.mat')
27
28 %% Large FOV, p1, 40/s, 10000 frames, separation and reattachment
29 clear all
30 clc
31 addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
32 Vel40_yPos1_Bump\MATLAB Processed')
33
34 load('Conditionally_Avg_40.mat')
35 load('Vel40_yPos1_BumpLFOV_MATLAB_Processed.mat')
36
37 %% Large FOV, centerline 60/s, 10000 frames, separation and reattachment
38 clear all
39 clc
40 addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
41 Vel60_yPos0_Bump\MATLAB Processed')
42
43 load('Conditionally_Avg_60.mat')
44 load('Vel60_yPos0_BumpLFOV_MATLAB_Processed.mat')
45 %% Animation of all frames
46 font = 12; %16 for Presentations, 12 for papers
47 L=36;
48
49 for pos = 1:length(Conditionally_averaged)
50     bins = Conditionally_averaged(pos).bins;
```

```
51      counts = Conditionally_averaged(pos).counts;
52      U_avg_test = Conditionally_averaged(pos).U;
53      W_avg_test = Conditionally_averaged(pos).W;
54      X = Conditionally_averaged(pos).X; %m
55      Z = Conditionally_averaged(pos).Z; %m
56
57      for bin_number = 1:length(counts)
58          U_plot = U_avg_test(:,:,bin_number);
59          W_plot = W_avg_test(:,:,bin_number);
60          H = pcolor(X/(L*0.0254),Z/(L*0.0254), U_plot);
61          colormap(jet), set(H,'edgecolor','none')
62          shading interp
63          ax = gca;
64          hold(ax);
65          x_data = get(H,'xdata');
66          z_data = get(H,'ydata');
67          u_data = get(H,'CData');
68          [M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
69          c.LineWidth = 5;
70          c.LineColor = 'white';
71          xlabel('x/L','interpreter','latex','FontSize',font)
72          ylabel('z/L','interpreter','latex','FontSize',font)
73          bar = colorbar;
74          bar.FontSize = font;
75          bar.Label.Interpreter = 'latex';
76          bar.Label.String = '$U/U_\infty$';
77          bar.TickLabelInterpreter = 'latex';
78          title(['Aneg = ', num2str(bins(bin_number)), '-' ,...
79          num2str(bins(bin_number+1)), ', Frame Count = ',...
80          num2str(counts(bin_number))],'FontSize',font)
81          hold(ax);
82          pause(1)
83      end
```

```
84 end
85 close all
86
87 %% save conditionally averaged fields as png fig focused on separation zone
88 font = 12; %16 for Presentations, 12 for papers
89 L=36;
90 for pos=1:length(Conditionally_averaged)
91     bins = Conditionally_averaged(pos).bins;
92     counts = Conditionally_averaged(pos).counts;
93     U_avg_test = Conditionally_averaged(pos).U; %6th frame
94     W_avg_test = Conditionally_averaged(pos).W; %6th frame
95     for bin_number = 1:length(counts)
96         U_plot = U_avg_test(:,:,bin_number);
97         W_plot = W_avg_test(:,:,bin_number);
98         X = Conditionally_averaged(pos).X; %m
99         Z = Conditionally_averaged(pos).Z; %m
100
101         H = pcolor(X/(L*0.0254),Z/(L*0.0254), U_plot);
102         colormap(jet), set(H,'edgecolor','none')
103         shading interp
104         ax = gca;
105         hold(ax);
106         x_data = get(H,'xdata');
107         z_data = get(H,'ydata');
108         u_data = get(H,'CData');
109         [M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
110         c.LineWidth = 3;
111         c.LineColor = 'white';
112         xlabel('x/L','interpreter','latex','FontSize',font)
113         ylabel('z/L','interpreter','latex','FontSize',font)
114         bar = colorbar;
115         bar.FontSize = font;
116         bar.Label.Interpreter = 'latex';
```

```
117         bar.Label.String = '$U/U_\infty$';
118         bar.TickLabelInterpreter = 'latex';
119         hold(ax);
120 
121         L = 36;
122         L_b = 35.5;
123         x_0 = 0.195 * L_b;
124         y_0 = 0.06 * L_b;
125         h_0 = 0.085* L_b;
126         Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
127         (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
128         z_0 = 0;
129         x_wall = linspace(min(min(X)),max(max(X)),200);
130         bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
131         z_wall = zeros(1,length(x_wall));
132         z_wall(1:min(bump_ind)-1) = z_0;
133         z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
134 
135         span = P_all(1).Input.Span/25.4;
136 
137         z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
138         ones(1,length(bump_ind)).*span*0.0393701);
139         hold on
140         plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
141         close all
142     end
143 end
144 %% Save standard deviation plots for each bin
145 font = 12; %16 for Presentations, 12 for papers
146 L=36;
147 
148 for pos=1:length(Conditionally_averaged)
149     bins = Conditionally_averaged(pos).bins;
```

```
150     counts = Conditionally_averaged(pos).counts;
151     U_cond_std = Conditionally_averaged(pos).Ustd; %6th frame
152     W_cond_std = Conditionally_averaged(pos).Wstd; %6th frame
153     for bin_number = 1:length(counts)
154
155         U_plot_std = U_cond_std(:,:,bin_number);
156         W_plot_std = W_cond_std(:,:,bin_number);
157         X = Conditionally_averaged(pos).X; %m
158         Z = Conditionally_averaged(pos).Z; %m
159
160         H = pcolor(X/(L*0.0254),Z/(L*0.0254), U_plot_std);
161         colormap(jet), set(H,'edgecolor','none')
162         shading interp
163         ax = gca;
164         hold(ax);
165         x_data = get(H,'xdata');
166         z_data = get(H,'ydata');
167         u_data = get(H,'CData');
168         xlabel('x/L','interpreter','latex','FontSize',font)
169         ylabel('z/L','interpreter','latex','FontSize',font)
170         bar = colorbar;
171         bar.FontSize = font;
172         bar.Label.Interpreter = 'latex';
173         bar.Label.String = '${U/U_\infty}_{sigma}$';
174         bar.TickLabelInterpreter = 'latex';
175         hold(ax);
176
177         L = 36;
178         L_b = 35.5;
179         x_0 = 0.195 * L_b;
180         y_0 = 0.06 * L_b;
181         h_0 = 0.085* L_b;
182         Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .*...
```

```
183         (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
184         z_0 = 0;
185         x_wall = linspace(min(min(X)),max(max(X)),200);
186
187         bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
188         z_wall = zeros(1,length(x_wall));
189         z_wall(1:min(bump_ind)-1) = z_0;
190         z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
191
192         span = P_all(1).Input.Span/25.4;
193
194         z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
195         ones(1,length(bump_ind)).*span*0.0393701);
196         hold on
197         plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
198
199         close all
200     end
201 end
202
203 %% Add streamlines to separation frames
204 font = 12; %16 for Presentations, 12 for papers
205 L=36;
206
207 for pos = 1:length(P_all)
208
209     bins = Conditionally_averaged(pos).bins;
210     counts = Conditionally_averaged(pos).counts;
211     U_avg_test = Conditionally_averaged(pos).U;
212     W_avg_test = Conditionally_averaged(pos).W;
213
214     for bin_number = 1:length(counts)
215
```

```
figure
U_plot = U_avg_test(:,:,bin_number);
W_plot = W_avg_test(:,:,bin_number);
X = Conditionally_averaged(pos).X; %m
Z = Conditionally_averaged(pos).Z; %m

H = pcolor(X/(L*0.0254),Z/(L*0.0254), U_plot);
colormap(jet), set(H,'edgecolor','none')
shading interp
ax = gca;
hold(ax);
x_data = get(H,'xdata');
z_data = get(H,'ydata');
u_data = get(H,'CData');
[M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
c.LineWidth = 3;
c.LineColor = 'white';
xlabel('x/L','interpreter','latex','FontSize',font)
ylabel('z/L','interpreter','latex','FontSize',font)
bar = colorbar;
bar.FontSize = font;
bar.Label.Interpreter = 'latex';
bar.Label.String = '$U/U_\infty$';
bar.TickLabelInterpreter = 'latex';
hold(ax);


 line = streamslice(X/(L*0.0254),Z/(L*0.0254),U_plot,W_plot);
 set(line,'Color','black','LineWidth',0.05,'MarkerSize',0.01);


L = 36;
L_b = 35.5;
```

```
249         x_0 = 0.195 * L_b;
250         y_0 = 0.06 * L_b;
251         h_0 = 0.085* L_b;
252         Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
253         (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
254         z_0 = 0;
255         x_wall = linspace(min(min(X)),max(max(X)),200);
256
257         bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
258         z_wall = zeros(1,length(x_wall));
259         z_wall(1:min(bump_ind)-1) = z_0;
260         z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
261
262         span = P_all(1).Input.Span/25.4;
263
264         z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
265         ones(1,length(bump_ind)).*span*0.0393701);
266         hold on
267         plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
268
269         xlim([min(min(X))./(L*0.0254) max(max(X))./(L*0.0254)-0.001])
270
271
272     end
273     close all
274 end
275
276 %% Determine mean fields
277 figure
278
279 pos = 1;
280 L = 36;
281 font = 12;
```

```
282
283 Umean = Stats(pos).U;
284 Wmean = Stats(pos).W;
285 X = Stats(pos).X;
286 Z = Stats(pos).Z;
287
288 H = pcolor(X/(0.0254*L),Z/(0.0254*L),Umean); %Uinf from calibrated data
289 colormap(jet), set(H,'edgecolor','none')
290 shading interp
291 ax = gca;
292 hold(ax);
293 x_data = get(H,'xdata');
294 z_data = get(H,'ydata');
295 u_data = get(H,'CData');
296 [M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
297 c.LineWidth = 3;
298 c.LineColor = 'w';
299 xlabel('x/L','interpreter','latex','FontSize',font)
300 ylabel('z/L','interpreter','latex','FontSize',font)
301 bar = colorbar;
302 bar.FontSize = font;
303 bar.Label.Interpreter = 'latex';
304 bar.Label.String = '$U/U_\infty$';
305 bar.TickLabelInterpreter = 'latex';
306 hold(ax)
307 %
308 line = streamslice(X/(L*0.0254),Z/(L*0.0254),Umean,Wmean);
309 set(line,'Color','black','LineWidth',0.05,'MarkerSize',0.01)
310
311 L = 36;
312 L_b = 35.5;
313 x_0 = 0.195 * L_b;
314 y_0 = 0.06 * L_b;
```

```
315 h_0 = 0.085* L_b;
316 Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
317 (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
318 z_0 = 0;
319 x_wall = linspace(min(min(X)),max(max(X)),200);
320
321 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
322 z_wall = zeros(1,length(x_wall));
323 z_wall(1:min(bump_ind)-1) = z_0;
324 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
325
326 span = P_all(1).Input.Span/25.4;
327
328 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
329 ones(1,length(bump_ind)).*span*0.0393701);
330 hold on
331 plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
332
333 set(gca,'FontSize',font)
334
335 close all
336
337 %% Determine separation locations across frames
338 clear x_sep z_sep
339 font = 12; %16 for Presentations, 12 for papers
340 L=36;
341
342 for pos = 1:1
343
344     bins = Conditionally_averaged(pos).bins;
345     counts = Conditionally_averaged(pos).counts;
346     U_avg_test = Conditionally_averaged(pos).U;
347     W_avg_test = Conditionally_averaged(pos).W;
```

```
348
349     for bin_number = 1:length(counts)
350
351         figure
352         U_plot = U_avg_test(:,:,bin_number);
353         W_plot = W_avg_test(:,:,bin_number);
354         X = Conditionally_averaged(pos).X; %m
355         Z = Conditionally_averaged(pos).Z; %m
356
357         H = pcolor(X/(L*0.0254),Z/(L*0.0254), U_plot);
358         colormap(jet), set(H,'edgecolor','none')
359         shading interp
360         ax = gca;
361         hold(ax);
362         x_data = get(H,'xdata');
363         z_data = get(H,'ydata');
364         u_data = get(H,'CData');
365         [M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
366         c.LineWidth = 3;
367         c.LineColor = 'w';
368
369         xlabel('x/L','interpreter','latex','FontSize',font)
370         ylabel('z/L','interpreter','latex','FontSize',font)
371         bar = colorbar;
372         bar.FontSize = font;
373         bar.Label.Interpreter = 'latex';
374         bar.Label.String = '$U/U_\infty$';
375         bar.TickLabelInterpreter = 'latex';
376         caxis([-0.3 1.5])
377         hold(ax);
378
379         x_contour_locations = sort(M(1,:));
380         i=1;
```

```
381         if isempty(x_contour_locations)
382             x_sep(bin_number,pos) = NaN;
383         else
384             while x_contour_locations(i) == 0
385                 i=i+1;
386             end
387             x_sep(bin_number,pos) = x_contour_locations(i);
388         end
389 
390         L = 36;
391         L_b = 35.5;
392         x_0 = 0.195 * L_b;
393         y_0 = 0.06 * L_b;
394         h_0 = 0.085* L_b;
395         Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
396         (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
397         z_0 = 0;
398         x_wall = linspace(min(min(X)),max(max(X)),200);
399 
400         bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
401         z_wall = zeros(1,length(x_wall));
402         z_wall(1:min(bump_ind)-1) = z_0;
403         z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
404 
405         span = P_all(1).Input.Span/25.4;
406 
407         z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
408         ones(1,length(bump_ind)).*span*0.0393701);
409         hold on
410         plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
411 
412         xlim([min(min(X))./(L*0.0254) max(max(X))./(L*0.0254)-0.001])
413         z_sep(bin_number,pos) = Z_bump(x_sep(bin_number,pos)*(L),...
```

```
414         span*0.0393701)/(L);
415     end
416     close all
417 end
418 %% Calculate Mean Separation
419 clear x_contour_locations
420 figure
421
422 pos = 1;
423 L = 36;
424 font = 12;
425
426 Umean = Stats(pos).U;
427 Wmean = Stats(pos).W;
428 X = Stats(pos).X;
429 Z = Stats(pos).Z;
430
431 H = pcolor(X/(0.0254*L),Z/(0.0254*L),Umean); %Uinf from calibrated data
432 colormap(jet), set(H,'edgecolor','none')
433 shading interp
434 ax = gca;
435 hold(ax);
436 x_data = get(H,'xdata');
437 z_data = get(H,'ydata');
438 u_data = get(H,'CData');
439 [M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
440 c.LineWidth = 3;
441 c.LineColor = 'w';
442 xlabel('x/L','interpreter','latex','FontSize',font)
443 ylabel('z/L','interpreter','latex','FontSize',font)
444 bar = colorbar;
445 bar.FontSize = font;
446 bar.Label.Interpreter = 'latex';
```

```
447 bar.Label.String = '$U/U_\infty$';
448 bar.TickLabelInterpreter = 'latex';
449 hold(ax)
450
451 x_contour_locations = sort(M(1,:));
452 i=1;
453 if isempty(x_contour_locations)
454     x_sep_mean(pos) = NaN;
455 else
456     while x_contour_locations(i) == 0
457         i=i+1;
458     end
459     x_sep_mean(pos) = x_contour_locations(i);
460 end
461
462
463 L = 36;
464 L_b = 35.5;
465 x_0 = 0.195 * L_b;
466 y_0 = 0.06 * L_b;
467 h_0 = 0.085* L_b;
468 Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
469 (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
470 z_0 = 0;
471 x_wall = linspace(min(min(X)),max(max(X)),200);
472
473 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
474 z_wall = zeros(1,length(x_wall));
475 z_wall(1:min(bump_ind)-1) = z_0;
476 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
477
478 span = P_all(1).Input.Span/25.4;
479
```

```
480 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
481 ones(1,length(bump_ind)).*span*0.0393701);
482
483 z_sep_mean(pos) = Z_bump(x_sep_mean(pos)*(L),span*0.0393701)/(L);
484 close all
485 %% "histogram" separation points counts divided by total count to get pdf
486 if pwd == "E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
487 Hari_cond_avg\LFOV\Vel40_p0_LongSample_cond_avg"
488     x_sep(41) = 0.10348;
489     z_sep(41) = 0.0627;
490     std_dev_x = 0.0075;
491     std_dev_z = 0.0026;
492 elseif pwd =="E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
493 Hari_cond_avg\LFOV\Vel60_p0_cond_avg"
494     x_sep_mean(1) = .10045;
495     z_sep_mean(1) = Z_bump(x_sep_mean(pos)*(L),span*0.0393701)/(L);
496     std_dev_x = 0.0094;
497     std_dev_z = 0.0032;
498 end
499 %%
500 clear bar
501 font = 16;
502 pos = 1;
503 yyaxis left
504 plot(x_sep,Conditionally_averaged(pos).counts'/...
505 sum(Conditionally_averaged(pos).counts)/(bins(2)-bins(1)),'o');
506 xlabel('x/L','interpreter','latex','FontSize',font)
507 ylabel('p.d.f','interpreter','latex','FontSize',font)
508
509 hold on
510 xline(x_sep_mean-2*std_dev_x,':','$-2\sigma$',...
511 'LabelHorizontalAlignment','left','FontSize',font,'interpreter','latex')
512 hold on
```

```
513 xline(x_sep_mean-std_dev_x,':','$-\sigma$',...
514 'LabelHorizontalAlignment','left','FontSize',font,'interpreter','latex')
515 hold on
516 xline(x_sep_mean,':','Mean','interpreter','latex','FontSize',font)
517 hold on
518 xline(x_sep_mean+std_dev_x,':','$+\sigma$','FontSize',font,...
519 'interpreter','latex')
520 hold on
521 xline(x_sep_mean+2*std_dev_x,':','$+2\sigma$',...
522 'LabelHorizontalAlignment','right','FontSize',font,'interpreter','latex')
523
524 cdf_individual =flip(x_sep).*Conditionally_averaged(pos).counts'/...
525 sum(Conditionally_averaged(pos).counts)/(bins(2)-bins(1));
526 cdf_individual= cdf_individual/sum(cdf_individual,'omitnan');
527
528 for i = 1:length(cdf_individual)
529     if i == 1
530         cdf_total_xsep(i) = cdf_individual(i);
531     else
532         cdf_total_xsep(i) = cdf_individual(i)+cdf_total_xsep(i-1);
533     end
534 end
535
536 cdf_total_xsep(isnan(cdf_total_xsep)) = 1;
537
538 yyaxis right
539 plot(sort(x_sep),sort(cdf_total_xsep),'-');
540 ylabel('CDF','interpreter','latex','FontSize',font)
541
542 set(gca,'FontSize',font)
543 set(gca,'box','off')
544 %%
545 png_name = strcat('xsep',num2str(pos), '.png');
```

```
546 saveas(gcf,[pwd png_name])
547
548 fig_name = strcat('xsep',num2str(pos), '.fig');
549 saveas(gcf,[pwd fig_name])
550 %%
551
552 figure
553 plot(z_sep,Conditionally_averaged(pos).counts'/...
554 sum(Conditionally_averaged(pos).counts)/(bins(2)-bins(1)),'o');
555 xlabel('z/L','interpreter','latex','FontSize',font)
556 ylabel('p.d.f','interpreter','latex','FontSize',font)
557
558 hold on
559 xline(z_sep_mean-2*std_dev_z,':','$-2\sigma$',...
560 'LabelHorizontalAlignment','left','FontSize',font,'interpreter','latex')
561 hold on
562 xline(z_sep_mean-std_dev_z,':','$-\sigma$','...
563 LabelHorizontalAlignment','left','FontSize',font,'interpreter','latex')
564 hold on
565 xline(z_sep_mean,':','Mean','interpreter','latex')
566 hold on
567 xline(z_sep_mean+std_dev_z,':','$+\sigma$','FontSize',font,...
568 'interpreter','latex')
569 hold on
570 xline(z_sep_mean+2*std_dev_z,':','$+2\sigma$',...
571 'LabelHorizontalAlignment','right','FontSize',font,'interpreter','latex')
572 set(gca,'FontSize',font)
573
574
575 png_name = strcat('zsep',num2str(pos), '.png');
576 saveas(gcf,[pwd png_name])
577
578 fig_name = strcat('zsep',num2str(pos), '.fig');
```

```
saveas(gcf,[pwd fig_name])

close all

%% Plot separation locations on Bump Space
separation_properties(:,1) = x_sep';
separation_properties(:,2) = z_sep';
separation_properties(:,3) = counts;

scatter(x_sep(1:1:end),z_sep(1:1:end),100,counts(1:1:end)/...
sum(counts)*100,'filled')
H = colorbar;
H.Label.String = '\% Probability of Separation Occuring';
H.Label.Interpreter = 'latex';
H.Label.FontSize = font;
H.TickLabelInterpreter = 'latex';
H.FontSize = font;

%hold on
%plot(x_sep_mean, z_sep_mean,'r*','MarkerSize',10)

hold on
xline(x_sep_mean-2*std_dev_x,':','$-2\sigma$',...
'LabelHorizontalAlignment','left','FontSize',font,'interpreter','latex')
hold on
xline(x_sep_mean-std_dev_x,':','$-\sigma$',...
'LabelHorizontalAlignment','left','FontSize',font,'interpreter','latex')
hold on
xline(x_sep_mean,':','Mean','interpreter','latex','FontSize',font)
hold on
xline(x_sep_mean+std_dev_x,':','$+\sigma$','FontSize',font,...
'interpreter','latex')
hold on
```

```
612 xline(x_sep_mean+2*std_dev_x,':','$+2\sigma$',...
613 'LabelHorizontalAlignment','right','FontSize',font,'interpreter','latex')
614
615 xlabel('x/L','interpreter','latex','FontSize',font)
616 ylabel('z/L','interpreter','latex','FontSize',font)
617
618 L = 36;
619 L_b = 35.5;
620 x_0 = 0.195 * L_b;
621 y_0 = 0.06 * L_b;
622 h_0 = 0.085* L_b;
623 Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
624 (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
625 z_0 = 0;
626 x_wall = linspace(min(min(X)),max(max(X)),200);
627
628 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
629 z_wall = zeros(1,length(x_wall));
630 z_wall(1:min(bump_ind)-1) = z_0;
631 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
632
633 span = P_all(1).Input.Span/25.4;
634
635 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,ones...
636 (1,length(bump_ind)).*span*0.0393701);
637 hold on
638 plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',0.5)
639 xlim([min(min(X))./(L*0.0254) max(max(X))./(L*0.0254)])
640
641 set(gca,'FontSize',font)
642 png_name = strcat('PlotonBump_sep',num2str(pos), '.png');
643 saveas(gcf,[pwd png_name])
644
```

```
645 fig_name = strcat('PlotonBump_sep',num2str(pos), '.fig');
646 %%
647 saveas(gcf,[pwd fig_name])
648 close all
649
650 %% Determine reattachment locations across frames
651 clear x_sep z_sep x_re z_re
652 font = 12; %16 for Presentations, 12 for papers
653 L=36;
654
655 for pos = 2:3
656
657     bins = Conditionally_averaged(pos).bins;
658     counts = Conditionally_averaged(pos).counts;
659     U_avg_test = Conditionally_averaged(pos).U;
660     W_avg_test = Conditionally_averaged(pos).W;
661
662     for bin_number = 1:length(counts)
663
664         figure
665         U_plot = U_avg_test(:,:,bin_number);
666         W_plot = W_avg_test(:,:,bin_number);
667         X = Conditionally_averaged(pos).X; %m
668         Z = Conditionally_averaged(pos).Z; %m
669
670         H = pcolor(X/(L*0.0254),Z/(L*0.0254), U_plot);
671         colormap(jet), set(H,'edgecolor','none')
672         shading interp
673         ax = gca;
674         hold(ax);
675         x_data = get(H,'xdata');
676         z_data = get(H,'ydata');
677         u_data = get(H,'CData');
```

```
678         [M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
679         c.LineWidth = 3;
680         c.LineColor = 'w';
681 
682         xlabel('x/L','interpreter','latex','FontSize',font)
683         ylabel('z/L','interpreter','latex','FontSize',font)
684         bar = colorbar;
685         bar.FontSize = font;
686         bar.Label.Interpreter = 'latex';
687         bar.Label.String = '$U/U_\infty$';
688         bar.TickLabelInterpreter = 'latex';
689         caxis([-0.3 1.5])
690         hold(ax);
691 
692         x_contour_locations = sort(M(1,:));
693 
694         if isempty(x_contour_locations)
695             x_re(bin_number,pos) = NaN;
696         else
697             x_re(bin_number,pos) = max(x_contour_locations);
698         end
699 
700         L = 36;
701         L_b = 35.5;
702         x_0 = 0.195 * L_b;
703         y_0 = 0.06 * L_b;
704         h_0 = 0.085* L_b;
705         Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
706         (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
707         z_0 = 0;
708         x_wall = linspace(min(min(X)),max(max(X)),200);
709 
710         bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
```

```
711         z_wall = zeros(1,length(x_wall));
712         z_wall(1:min(bump_ind)-1) = z_0;
713         z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
714
715         span = P_all(1).Input.Span/25.4;
716
717         z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
718         ones(1,length(bump_ind)).*span*0.0393701);
719         hold on
720         plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
721
722         xlim([min(min(X))./(L*0.0254) max(max(X))./(L*0.0254)-0.001])
723         z_re(bin_number,pos) = Z_bump(x_re(bin_number,pos)*(L),...
724         span*0.0393701)/(L);
725     end
726     close all
727 end
728
729 %% Calculate Mean Reattachment
730 clear x_contour_locations
731 figure
732
733 L = 36;
734 font = 12;
735
736 for pos = 2:3
737     Umean = Stats(pos).U;
738     Wmean = Stats(pos).W;
739     X = Stats(pos).X;
740     Z = Stats(pos).Z;
741
742     H = pcolor(X/(0.0254*L),Z/(0.0254*L),Umean); %Uinf from calibrated data
743     colormap(jet), set(H,'edgecolor','none')
```

```
744     shading interp
745     ax = gca;
746     hold(ax);
747     x_data = get(H,'xdata');
748     z_data = get(H,'ydata');
749     u_data = get(H,'CData');
750     [M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
751     c.LineWidth = 3;
752     c.LineColor = 'w';
753     xlabel('x/L','interpreter','latex','FontSize',font)
754     ylabel('z/L','interpreter','latex','FontSize',font)
755     bar = colorbar;
756     bar.FontSize = font;
757     bar.Label.Interpreter = 'latex';
758     bar.Label.String = '$U/U_\infty$';
759     bar.TickLabelInterpreter = 'latex';
760     hold(ax)
761
762     x_contour_locations = sort(M(1,:));
763
764     if isempty(x_contour_locations)
765         x_re_mean(pos)= NaN;
766     else
767         x_re_mean(pos)= max(x_contour_locations);
768     end
769
770
771     L = 36;
772     L_b = 35.5;
773     x_0 = 0.195 * L_b;
774     y_0 = 0.06 * L_b;
775     h_0 = 0.085* L_b;
776     Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
```

```
777      (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
778      z_0 = 0;
779      x_wall = linspace(min(min(X)),max(max(X)),200);
780
781      bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
782      z_wall = zeros(1,length(x_wall));
783      z_wall(1:min(bump_ind)-1) = z_0;
784      z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
785
786      span = P_all(1).Input.Span/25.4;
787
788      z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
789      ones(1,length(bump_ind)).*span*0.0393701);
790
791      z_re_mean(pos) = Z_bump(x_re_mean(pos)*(L),span*0.0393701)/(L);
792      close all
793  end
794  %%
795  if pwd == "E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
796  Hari_cond_avg\LFOV\Vel40_p0.5_cond_avg"
797      x_re_mean(2) = 0.314;
798      z_re_mean(2) = Z_bump(x_re_mean(2)*(L),span*0.0393701)/(L);
799
800      x_re_mean(3) = 0.3033;
801      z_re_mean(3) = Z_bump(x_re_mean(3)*(L),span*0.0393701)/(L);
802
803  elseif pwd == "E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
804  Hari_cond_avg\LFOV\Vel40_p0_cond_avg"
805      x_re_mean(2) = NaN;
806      z_re_mean(2) = NaN;
807
808      x_re_mean(3) = 0.3506;
809      z_re_mean(3) = Z_bump(x_re_mean(3)*(L),span*0.0393701)/(L);
```

```
810
811      x_re_mean_all = 0.3519;
812      z_re_mean_all = 0.0032;
813      std_dev_re_x = 0.0256;
814      std_dev_re_z = 0.0016;
815
816  elseif pwd == "E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
817  Hari_cond_avg\LFOV\Vel60_p0_cond_avg"
818      x_re_mean(2) =.3576;
819      z_re_mean(2) = Z_bump(x_re_mean(2)*(L),span*0.0393701)/(L);
820
821      x_re_mean_all = 0.3406;
822      z_re_mean_all = 0.0038;
823
824      std_dev_re_x = 0.0187;
825      std_dev_re_z = 0.0015;
826
827  end
828  %% "histogram" reattachment points counts divided by total count to get pdf
829  font = 16;
830  clear bar
831  figure
832  yyaxis left
833  pos = 2;
834  end_count = length(Conditionally_averaged(pos).counts);
835  plot(x_re(1:end_count,pos),Conditionally_averaged(pos).counts'/...
836  sum(Conditionally_averaged(pos).counts)/(bins(2)-bins(1)),'o','Color',...
837  [0 0.4470 0.7410]);
838
839  hold on
840  pos = 3;
841  end_count = length(Conditionally_averaged(pos).counts);
842  plot(x_re(1:end_count,pos),Conditionally_averaged(pos).counts'...
```

```
/sum(Conditionally_averaged(pos).counts)/(bins(2)-bins(1)),'o','Color',...
[0 0.4470 0.7410]);

xlabel('x/L','interpreter','latex','FontSize',font)
ylabel('p.d.f','interpreter','latex','FontSize',font)

x_re_total=[x_re(1:13,2); x_re(3:48,3)];
counts_total = [Conditionally_averaged(2).counts(1:13)';...
Conditionally_averaged(3).counts(3:48)'];

cdf_individual =x_re_total.*counts_total/sum(counts_total)/...
(bins(2)-bins(1));
cdf_individual= cdf_individual/sum(cdf_individual,'omitnan');

for i = 1:length(cdf_individual)
    if i == 1
        cdf_total_xre(i) = cdf_individual(i);
    else
        cdf_total_xre(i) = cdf_individual(i)+cdf_total_xre(i-1);
    end
end

cdf_total_xre(isnan(cdf_total_xre)) = 1;

yyaxis right
plot(sort(x_re_total),sort(cdf_total_xre),'-')
ylabel('CDF','interpreter','latex')
ylim([0 1])
xlim([0.226200000000000 0.4425])
hold on
xline(x_re_mean_all-2*std_dev_re_x,':','$-2\sigma$',...
'LabelHorizontalAlignment','left','FontSize',font,'interpreter','latex')
hold on
```

```
xline(x_re_mean_all-std_dev_re_x,':','$-\sigma$',...
'LabelHorizontalAlignment','left','FontSize',font,'interpreter','latex')
hold on
xline(x_re_mean_all,':','Mean','interpreter','latex','FontSize',font,...
'interpreter','latex')
hold on
xline(x_re_mean_all+std_dev_re_x,':','$+\sigma$','FontSize',font,...
'interpreter','latex')
hold on
xline(x_re_mean_all+2*std_dev_re_x,':','$+2\sigma$','LabelHorizontalAlignment','right','
,'interpreter','latex')

set(gca,'FontSize',font)
set(gca,'box','off')

png_name = strcat('xre_all', '.png');
saveas(gcf,[pwd png_name])

fig_name = strcat('xre_all', '.fig');
saveas(gcf,[pwd fig_name])
%%
figure
pos = 2;
end_count = length(Conditionally_averaged(pos).counts);
plot(z_re(1:end_count,pos),Conditionally_averaged(pos).counts'/...
sum(Conditionally_averaged(pos).counts)/(bins(2)-bins(1)),'o','Color',...
[0 0.4470 0.7410]);

hold on
pos = 3;
end_count = length(Conditionally_averaged(pos).counts);
plot(z_re(1:end_count,pos),Conditionally_averaged(pos).counts'/...
sum(Conditionally_averaged(pos).counts)/(bins(2)-bins(1)),'o','Color',...
```

```
[0 0.4470 0.7410]);

xlabel('z/L','interpreter','latex','FontSize',font)
ylabel('p.d.f','interpreter','latex','FontSize',font)

hold on
xline(z_re_mean_all-2*std_dev_re_z,':','-2\sigma',...
'LabelHorizontalAlignment','left','FontSize',font)
hold on
xline(z_re_mean_all-std_dev_re_z,':','-\sigma',...
'LabelHorizontalAlignment','left','FontSize',font)
hold on
xline(z_re_mean_all,':','Mean','interpreter','latex','FontSize',font)
hold on
xline(z_re_mean_all+std_dev_re_z,':','+\sigma','FontSize',font)
hold on
xline(z_re_mean_all+2*std_dev_re_z,':','+2\sigma',...
'LabelHorizontalAlignment','right','FontSize',font)

set(gca,'FontSize',font)

png_name = strcat('zre_all', '.png');
saveas(gcf,[pwd png_name])

fig_name = strcat('zre_all', '.fig');
saveas(gcf,[pwd fig_name])

close all

%% Plot reattachment locations on Bump Space
if pwd == "E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Hari_cond_avg\LFOV\Vel40_p0_cond_avg"
    re_properties(:,1) = [x_re(1:34,2); x_re(1:56,3)];
```

```
    re_properties(:,2) = [z_re(1:34,2); z_re(1:56,3)];
    re_properties(:,3) = [Conditionally_averaged(2).counts';...
        Conditionally_averaged(3).counts'];
elseif pwd == "E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Hari_cond_avg\LFOV\Vel60_p0_cond_avg"
    re_properties(:,1) = x_re(:,2);
    re_properties(:,2) = z_re(:,2);
    re_properties(:,3) = counts';
end
%%
scatter(re_properties(:,1),re_properties(:,2),100,re_properties(:,3)/...
sum(re_properties(:,3))*100,'filled')
H = colorbar;
H.Label.String = '\% Probability of Reattachment Occuring';
H.Label.Interpreter = 'latex';
H.Label.FontSize = font;
H.TickLabelInterpreter = 'latex';
H.FontSize = font;

%hold on
%plot(x_sep_mean, z_sep_mean,'r*','MarkerSize',10)

hold on
xline(x_re_mean_all-2*std_dev_re_x,':','-2\sigma',...
'LabelHorizontalAlignment','left','FontSize',font)
hold on
xline(x_re_mean_all-std_dev_re_x,':','-\sigma',...
'LabelHorizontalAlignment','left','FontSize',font)
hold on
xline(x_re_mean_all,':','Mean','interpreter','latex','FontSize',font)
hold on
xline(x_re_mean_all+std_dev_re_x,':','+\sigma','FontSize',font)
hold on
```

```
975 xline(x_re_mean_all+2*std_dev_re_x,':','+2\sigma',...
976 'LabelHorizontalAlignment','right','FontSize',font)
977
978 xlabel('x/L','interpreter','latex','FontSize',font)
979 ylabel('z/L','interpreter','latex','FontSize',font)
980
981 L = 36;
982 L_b = 35.5;
983 x_0 = 0.195 * L_b;
984 y_0 = 0.06 * L_b;
985 h_0 = 0.085* L_b;
986 Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
987 (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
988 z_0 = 0;
989 x_wall = linspace(0.15,max(max(X))+0.5,200);
990
991 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
992 z_wall = zeros(1,length(x_wall));
993 z_wall(1:min(bump_ind)-1) = z_0;
994 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
995
996 span = P_all(1).Input.Span/25.4;
997
998 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
999 ones(1,length(bump_ind)).*span*0.0393701);
1000
1001 hold on
1002 plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',1)
1003 xlim([0.2 max(max(X))./(L*0.0254)])
1004 ylim([0 0.03])
1005
1006 set(gca,'FontSize',font')
1007
```

```
png_name = strcat('PlotonBump_re',num2str(pos), '.png');
saveas(gcf,[pwd png_name])

fig_name = strcat('PlotonBump_re',num2str(pos), '.fig');
saveas(gcf,[pwd fig_name])

close all
```

Listing B.6: Upstream Disturbance Correlation Analysis of Conditionally Averaged PIV Fields

```
%% Large FOV, centerline, 40/s, LongSample, separation
clear all
clc
addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Vel40_yPos0_Bump_LongSample\MATLAB Processed')
addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Hari_cond_avg\LFOV\Vel40_p0_LongSample_cond_avg')
load('Conditionally_Avg_40.mat')
load('Vel40_yPos0_Bump_LongSampleLFOV_MATLAB_Processed.mat')
load('Vel40_yPos0_Bump_LongSampleUinf.mat')
%% Animation of curl of all frames
font = 12; %16 for Presentations, 12 for papers
L=36;
Uinf =  44.7156;
for pos = 1:length(Conditionally_averaged)
    bins = Conditionally_averaged(pos).bins;
    counts = Conditionally_averaged(pos).counts;
    U_avg_test = Conditionally_averaged(pos).U;
    W_avg_test = Conditionally_averaged(pos).W;
    X = Conditionally_averaged(pos).X; %m
    Z = Conditionally_averaged(pos).Z; %m
```

```
22
23      for bin_number = 1:length(counts)
24          U_plot = U_avg_test(:,:,bin_number);
25          W_plot = W_avg_test(:,:,bin_number);
26          [curlz,cav] = curl(X/(L*0.0254),Z/(L*0.0254),U_plot,W_plot);
27      end
28  end
29  close all
30
31  %% Determine mean fields
32  figure
33
34  pos = 1;
35  L = 36;
36  font = 12;
37
38  Umean = Stats(pos).U;
39  Wmean = Stats(pos).W;
40  X = Stats(pos).X;
41  Z = Stats(pos).Z;
42
43  [curlz_mean,cav] = curl(X/(L*0.0254),Z/(L*0.0254),Umean,Wmean);
44
45  H = pcolor(X/(L*0.0254),Z/(L*0.0254), curlz_mean);
46  colormap(jet), set(H,'edgecolor','none')
47  shading interp
48
49  d = colorbar;
50  caxis([-200 0])
51  ylabel(d, '$\omega/U_\infty$', 'interpreter', 'latex')
52  xlabel('$x/L$','interpreter','latex')
53  ylabel('$z/L$','interpreter','latex')
54  x_data = get(H,'xdata');
```

```
z_data = get(H,'ydata');
w_data = get(H,'CData');
%[M,c] = contour(x_data,z_data,w_data,[0,0],'ShowText','on');
hold on
[M,c] = contour(x_data,z_data,w_data,[0,0],'ShowText','on');
c.LineWidth = 1;
c.LineColor = 'white';
hold on
L = 36;
L_b = 35.5;
x_0 = 0.195 * L_b;
y_0 = 0.06 * L_b;
h_0 = 0.085* L_b;
Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
(1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
z_0 = 0;
x_wall = linspace(min(min(X)),max(max(X)),200);

bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
z_wall = zeros(1,length(x_wall));
z_wall(1:min(bump_ind)-1) = z_0;
z_wall(max(bump_ind)+1:length(x_wall)) = z_0;

span = P_all(1).Input.Span/25.4;

z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
ones(1,length(bump_ind)).*span*0.0393701);
hold on
plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)

% set(gca,'FontSize',font)
%
% png_name = strcat(num2str(pos), ' Mean Field.png');
```

```
% saveas(gcf,[pwd '/', num2str(pos),'/',png_name])
%
% fig_name = strcat(num2str(pos), ' Mean Field.fig');
% saveas(gcf,[pwd '/', num2str(pos),'/',fig_name])
%
% close all

%% Calc reverse fraction for mean field
[r,c] = size(curlz_mean);
Atot = r*c;
Aneg = 0;
Apos = 0;
for i = 1:r
    for j = 1:c
        if ~isnan(Umean(i,j))
            if Umean(i,j) < 0
                Aneg = Aneg+1;
            else
                Apos = Apos+1;
            end
        end
    end
end
Area_total= Aneg+Apos;
Negative_area_mean = Aneg/(Area_total);
Positive_area_mean = Apos/(Area_total);

%% Calc BL edge velocity at bump peak for mean field
clear endpoint_x endpoint_y end_index_x end_index_y x_bl y_bl
F = zeros(100,c);
for j = 267
    for i = r:-1:2
        if (isnan(curlz_mean(i,j)) && ~isnan(curlz_mean(i-1,j))) || ...
```

```
121         (~isnan(curlz_mean(end,j)))
122             y_bl_start(j,1) = i-1;
123             x_bl_start(j,1) = j;
124             y_bl = i;
125             x_bl = j;
126
127             while x_bl>0 && y_bl>0 && y_bl<= r && x_bl<=(c+1)
128                 x_bl = x_bl;
129                 y_bl = y_bl-1;
130
131                 epsilon = 0.02;
132                 if curlz_mean(y_bl,x_bl) > 2.13
133                     case_number =1;
134                     break
135                 else
136                     case_number = 0;
137
138                 end
139             end
140
141             end_index_x(j) = x_bl;
142             end_index_y(j) = y_bl;
143
144             Fmax_overall = .3843;
145
146             if case_number==1
147                 if (Z(y_bl_start(j),2) ~= Z(end_index_y(j),2))
148                     xq = linspace(mean(X(1,x_bl_start(j)),'omitnan')...
149                     ,mean(X(1,end_index_x(j)),'omitnan'),100);
150                     yq = linspace(mean(Z(y_bl_start(j),2),'omitnan')...
151                     ,mean(Z(end_index_y(j),2),'omitnan'),100);
152                     dist_local(:,j) = yq-mean(Z(i,:),'omitnan');
153                     curl_interp(:,j) = interp1([Z(y_bl_start(j),2)/...
```

```
154                     (L*0.0254) Z(end_index_y(j),2)/(L*0.0254)],...
155                     [curlz_mean(y_bl_start(j),x_bl_start(j)) ...
156                     curlz_mean(end_index_y(1,j),end_index_x(1,j))],...
157                     yq/(L*0.0254),'linear');
158                     U_interp(:,j) = interp1([Z(y_bl_start(j),2)/...
159                     (L*0.0254) Z(end_index_y(j),2)/(L*0.0254)],...
160                     [Umean(y_bl_start(j),x_bl_start(j))...
161                     Umean(end_index_y(1,j),end_index_x(1,j))],...
162                     yq/(L*0.0254),'linear');
163                     F(:,j) = -dist_local(:,j).*curl_interp(:,j);
164                     for k = 1:99
165                         if F(k+1) > epsilon*Fmax_overall
166                             endpoint_x(j) = xq(k);
167                             endpoint_y(j) = yq(k);
168                             U_e(j) = U_interp(k,j);
169                             Umean_edge =  Umean(y_bl,x_bl);
170                             break
171                         else
172                             endpoint_x(j) = xq(k+1);
173                             endpoint_y(j) = yq(k+1);
174                             U_e(j) = U_interp(k+1,j);
175                             Umean_edge =  Umean(y_bl,x_bl);
176                             break
177                         end
178                     end
179
180                 else
181                     endpoint_x(j) = mean(X(:,x_bl),'omitnan');
182                     endpoint_y(j) = mean(Z(y_bl,2),'omitnan');
183                     U_e(j) = Umean(y_bl,x_bl);
184                     Umean_edge =  Umean(y_bl,x_bl);
185                     break
186                 end
```

```
187             end
188
189         end
190     end
191     clear x_bl y_bl
192 end
193 %% Calculate mean BL thickness at bump peak
194 mean_bl_thickness = sqrt((endpoint_x(j)-(X(1,x_bl_start(j))))^2+...
195 (endpoint_y(j)-Z(y_bl_start(j),2))^2)+0.00372;
196
197 %% Calc BL edge velocity at bump peak for each conditionally averaged field
198 clear endpoint_x endpoint_y end_index_x end_index_y x_bl y_bl
199 %F = zeros(100,c);
200
201 for m = 1:86
202     clear y_bl_start curlz cav x_bl_start y_bl x_bl case_number end_index_x
203     clear end_index_z xq yq dist_local U_interp curl_interp F endpoint_x
204     clear endpoint_y U_e
205     U = U_avg_test(:,:,m);
206     W = W_avg_test(:,:,m);
207     [curlz cav] = curl(X/(L*0.0254),Z/(L*0.0254),U,W);
208
209     U_e = U(87,267);
210
211     Ue_allframes_cond(m) = U_e;
212 end
213 Ue_allframes_cond = Ue_allframes_cond';
214 %%
215 %plot(BL_thickness_allframes,'*')
216 hold on
217 plot(bins(2:end)-Negative_area_mean,Ue_allframes_cond-Umean_edge,'*')
218 %ylim([-.25 .15])
219
```

```
scatter(bins(2:end)-Negative_area_mean,Ue_allframes_cond-Umean_edge...
    ,30,counts(1:1:end)/sum(counts)*100,'filled')
H = colorbar;
H.Label.String = '\% Probability of Negative Flow Fraction';
H.Label.Interpreter = 'latex';
H.Label.FontSize = font;
H.TickLabelInterpreter = 'latex';
H.FontSize = font;
grid on
set(gca,'XMinorGrid','on');
set(gca,'YMinorGrid','on');
set(gca,'FontSize',font);
%leg = legend('Instantaneous','Condtionally Averaged');
xlabel('$\frac{A_{neg}}{A_{total}}-\frac{A_{negmean}}{A_{total}}$',...
    'Interpreter','latex','FontSize',20)
ylabel('$\frac{U_{e}}{U_{\infty}}-\frac{U_{emean}}{U_{\infty}}$',...
    'Interpreter','latex','FontSize',20)
```

Listing B.7: Upstream Disturbance Correlation Analysis of All Individual PIV Frames

```
%% Large FOV, centerline, 40/s, LongSample, separation
clear all
clc
addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Vel40_yPos0_Bump_LongSample\MATLAB Processed')
addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Hari_cond_avg\LFOV\Vel40_p0_LongSample_cond_avg')
load('Conditionally_Avg_40.mat')
load('Vel40_yPos0_Bump_LongSampleLFOV_MATLAB_Processed.mat')
load('Vel40_yPos0_Bump_LongSampleUinf.mat')
%% Animation of curl of all frames
font = 12; %16 for Presentations, 12 for papers
L=36;
```

```
14 Uinf =  44.7156;
15 for pos = 1:length(Conditionally_averaged)
16     bins = Conditionally_averaged(pos).bins;
17     counts = Conditionally_averaged(pos).counts;
18     U_avg_test = Conditionally_averaged(pos).U;
19     W_avg_test = Conditionally_averaged(pos).W;
20     X = Conditionally_averaged(pos).X; %m
21     Z = Conditionally_averaged(pos).Z; %m
22 
23     for bin_number = 1:length(counts)
24         U_plot = U_avg_test(:,:,bin_number);
25         W_plot = W_avg_test(:,:,bin_number);
26         [curlz,cav] = curl(X/(L*0.0254),Z/(L*0.0254),U_plot,W_plot);
27     end
28 end
29 close all
30 
31 %% Determine mean fields
32 figure
33 
34 pos = 1;
35 L = 36;
36 font = 12;
37 
38 Umean = Stats(pos).U;
39 Wmean = Stats(pos).W;
40 X = Stats(pos).X;
41 Z = Stats(pos).Z;
42 
43 [curlz_mean,cav] = curl(X/(L*0.0254),Z/(L*0.0254),Umean,Wmean);
44 
45 H = pcolor(X/(L*0.0254),Z/(L*0.0254), curlz_mean);
46 colormap(jet), set(H,'edgecolor','none')
```

```
47 shading interp
48
49 d = colorbar;
50 caxis([-200 0])
51 ylabel(d, '$\omega/U_\infty$', 'interpreter', 'latex')
52 xlabel('$x/L$','interpreter','latex')
53 ylabel('$z/L$','interpreter','latex')
54 x_data = get(H,'xdata');
55 z_data = get(H,'ydata');
56 w_data = get(H,'CData');
57 %[M,c] = contour(x_data,z_data,w_data,[0,0],'ShowText','on');
58 hold on
59 [M,c] = contour(x_data,z_data,w_data,[0,0],'ShowText','on');
60 c.LineWidth = 1;
61 c.LineColor = 'white';
62 hold on
63 L = 36;
64 L_b = 35.5;
65 x_0 = 0.195 * L_b;
66 y_0 = 0.06 * L_b;
67 h_0 = 0.085* L_b;
68 Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
69 (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
70 z_0 = 0;
71 x_wall = linspace(min(min(X)),max(max(X)),200);
72
73 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
74 z_wall = zeros(1,length(x_wall));
75 z_wall(1:min(bump_ind)-1) = z_0;
76 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
77
78 span = P_all(1).Input.Span/25.4;
79
```

```
z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
ones(1,length(bump_ind)).*span*0.0393701);
hold on
plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)


%% Calc reverse fraction for mean field
[r,c] = size(curlz_mean);
Atot = r*c;
Aneg = 0;
Apos = 0;
for i = 1:r
    for j = 1:c
        if ~isnan(Umean(i,j))
            if Umean(i,j) < 0
                Aneg = Aneg+1;
            else
                Apos = Apos+1;
            end
        end
    end
end
Area_total= Aneg+Apos;
Negative_area_mean = Aneg/(Area_total);
Positive_area_mean = Apos/(Area_total);

%% Calc BL edge velocity at bump peak for mean field
clear endpoint_x endpoint_y end_index_x end_index_y x_bl y_bl
F = zeros(100,c);
for j = 267
    for i = r:-1:2
        if (isnan(curlz_mean(i,j)) && ~isnan(curlz_mean(i-1,j))) || ...
        (~isnan(curlz_mean(end,j)))
```

```
113             y_bl_start(j,1) = i-1;
114             x_bl_start(j,1) = j;
115             y_bl = i;
116             x_bl = j;
117
118             while x_bl>0 && y_bl>0 && y_bl<= r && x_bl<=(c+1)
119                 x_bl = x_bl;
120                 y_bl = y_bl-1;
121
122                 epsilon = 0.02;
123                 if curlz_mean(y_bl,x_bl) > 2.13
124                     case_number =1;
125                     break
126                 else
127                     case_number = 0;
128
129                 end
130             end
131
132             end_index_x(j) = x_bl;
133             end_index_y(j) = y_bl;
134
135             Fmax_overall = .3843;
136
137             if case_number==1
138                 if (Z(y_bl_start(j),2) ~= Z(end_index_y(j),2))
139                     xq = linspace(mean(X(1,x_bl_start(j)),'omitnan'),...
140                     mean(X(1,end_index_x(j)),'omitnan'),100);
141                     yq = linspace(mean(Z(y_bl_start(j),2),'omitnan'),...
142                     mean(Z(end_index_y(j),2),'omitnan'),100);
143                     dist_local(:,j) = yq-mean(Z(i,:),'omitnan');
144                     curl_interp(:,j) = interp1([Z(y_bl_start(j),2)/...
145                     (L*0.0254) Z(end_index_y(j),2)/(L*0.0254)],...
```

```
146                     [curlz_mean(y_bl_start(j),x_bl_start(j)) ...
147                     curlz_mean(end_index_y(1,j),end_index_x(1,j))],...
148                     yq/(L*0.0254),'linear');
149                     U_interp(:,j) = interp1([Z(y_bl_start(j),2)/...
150                     (L*0.0254) Z(end_index_y(j),2)/(L*0.0254)],...
151                     [Umean(y_bl_start(j),x_bl_start(j))...
152                     Umean(end_index_y(1,j),end_index_x(1,j))],...
153                     yq/(L*0.0254),'linear');
154                     F(:,j) = -dist_local(:,j).*curl_interp(:,j);
155                     for k = 1:99
156                         if F(k+1) > epsilon*Fmax_overall
157                             endpoint_x(j) = xq(k);
158                             endpoint_y(j) = yq(k);
159                             U_e(j) = U_interp(k,j);
160                             Umean_edge =  Umean(y_bl,x_bl);
161                             break
162                         else
163                             endpoint_x(j) = xq(k+1);
164                             endpoint_y(j) = yq(k+1);
165                             U_e(j) = U_interp(k+1,j);
166                             Umean_edge =  Umean(y_bl,x_bl);
167                             break
168                         end
169                     end
170
171                 else
172                     endpoint_x(j) = mean(X(:,x_bl),'omitnan');
173                     endpoint_y(j) = mean(Z(y_bl,2),'omitnan');
174                     U_e(j) = Umean(y_bl,x_bl);
175                     Umean_edge =  Umean(y_bl,x_bl);
176                     break
177                 end
178             end
```

```
179
180         end
181     end
182     clear x_bl y_bl
183 end
184 %% Calculate mean BL thickness at bump peak
185 mean_bl_thickness = sqrt((endpoint_x(j)-(X(1,x_bl_start(j))))^2+...
186 (endpoint_y(j)-Z(y_bl_start(j),2))^2)+0.00372;
187
188 %% Calc BL edge velocity at bump peak for each conditionally averaged field
189 clear endpoint_x endpoint_y end_index_x end_index_y x_bl y_bl
190
191 for m = 1:40000
192     clear y_bl_start curlz cav x_bl_start y_bl x_bl case_number end_index_x
193     clear end_index_z xq yq dist_local U_interp curl_interp F endpoint_x
194     clear endpoint_y U_e
195     U = P_all.U(:,:,m);
196     W = P_all.W(:,:,m);
197     [curlz cav] = curl(X/(L*0.0254),Z/(L*0.0254),U,W);
198     U_e = U(87,267);
199     Ue_allframes(m) = U_e;
200 end
201 Ue_allframes = Ue_allframes';
202 %%
203 %plot(BL_thickness_allframes,'*')
204 hold on
205 plot(Conditionally_averaged.negative_area-...
206     Negative_area_mean,Ue_allframes-Umean_edge,'*','MarkerSize',0.5)
207 %ylim([-.1 .075])
208 grid on
209 xlabel('$\frac{A_{neg}}{A_{total}}-\frac{A_{negmean}}{A_{total}}$',...
210 'Interpreter','latex')
211 ylabel('$\frac{U_{e}}{U_{\infty}}-\frac{U_{emean}}{U_{\infty}}$',...
```

```
'Interpreter','latex')
set(gca,'FontSize',font)

%% Plot linear regression of data
mdl = fitlm(Conditionally_averaged.negative_area-...
    Negative_area_mean,Ue_allframes-Umean_edge);

figure
plot(mdl)
grid on
xlabel('$\frac{A_{neg}}{A_{total}}-\frac{A_{negmean}}{A_{total}}$',...
'Interpreter','latex')
ylabel('$\frac{U_{e}}{U_{\infty}}-\frac{U_{emean}}{U_{\infty}}$',...
'Interpreter','latex')
title('')

set(gca,'FontSize',font)
%%  Plot scatter plot with density of points
ind = ~isnan(Ue_allframes-Umean_edge);
yn = Ue_allframes(ind)-Umean_edge;
xn = Conditionally_averaged.negative_area(ind)-Negative_area_mean;

figure

s1= subplot(2,1,1);
plot(bins(2:end)-Negative_area_mean, counts,'*')
set(gca,'FontSize',font)
xlabel('$\frac{A_{neg}}{A_{total}}-\frac{A_{negmean}}{A_{total}}$',...
'Interpreter','latex','FontSize',18)
ylabel('Number of Samples','interpreter','latex','FontSize',font)
hold on
xline(0,'linewidth',2)
set(gca,'XTickLabel',[])
```

```
grid on
set(gca,'XMinorGrid','on');
set(gca,'YMinorGrid','on');

s2 = subplot(2,1,2);
scatplot(xn,yn,'circles');
hcb = colorbar;
set(gca,'FontSize',font)
grid on
set(gca,'XMinorGrid','on');
set(gca,'YMinorGrid','on');
xlabel('$\frac{A_{neg}}{A_{total}}-\frac{A_{negmean}}{A_{total}}$',...
'Interpreter','latex','FontSize',18)
ylabel('$\frac{U_{e}}{U_{\infty}}-\frac{U_{emean}}{U_{\infty}}$',...
'Interpreter','latex','FontSize',18)
ylabel(hcb, 'Point Density', 'interpreter', 'latex')
set(hcb,'TickLabelInterpreter', 'latex')

hcb.Label.Interpreter = 'latex';
%ylim([-0.228703831415398,0.14])

hold on
hold on
plot(bins(2:end)-Negative_area_mean,Ue_allframes_cond-Umean_edge,'ro', ...
'MarkerFaceColor','red','MarkerEdgeColor','white')

hold on
yline(0,'linewidth',2)
hold on
xline(0,'linewidth',2)
set(s1,'Box','off')
set(s2,'Box','off')
set(s1,'position',[0.13,0.561837209302326,0.721428571428571,...
```

```
0.341162790697675]);

set(s2,'position',[0.13,0.22,0.721428571428571,0.341162790697675]);


%% Correlation Coefficient Calculation
numerator = (40000*sum(xn.*yn,'omitnan')-sum(xn,'omitnan')*...
sum(yn,'omitnan'));
denominator = sqrt((40000*sum(xn.^2,'omitnan')-(sum(xn,'omitnan')).^2)*...
    (40000*sum(yn.^2,'omitnan')-(sum(yn,'omitnan')).^2));
r = numerator/denominator
```

Listing B.8: Proper Orthogonal Decomposition of PIV Fields

```
%function sPOD
% Author: Leo Hellstoem and Owen Williams
%
% This file primarily calculates POD modes and reconstructed velocity
% fields. As currently coded, it acts an energy filter, removing noisy
% strucutures from the data. POD modes are calculated, and then the data is
% projected on a chosen number to arrive at a de-noised field
%
%function sPOD(File, numModes, Reconstruct, 2ndProjFile)

function sPOD(varargin)

maxi=1;

set(0,'DefaultFigureColormap',feval('jet'))

close all

if nargin ~= 0%~isempty(varargin)
```

```
20      load(varargin{1});
21      Folder = [pwd '/'];
22      File = varargin{1};
23      Reconst = varargin{2};
24      if Reconst
25          UsePercent = varargin{3};
    %Use pecentage of total energy when reconstructing
26          num_modes = varargin{4};
27          if length(varargin)==5
28              SecondProjFile = varargin{5};
29          end
30      else
31          UsePercent = 0;
32          num_modes = varargin{3};
33      end
34  else
35      [File,Folder]=uigetfile('./*.mat', 'Pick dataset: ');
36      Reconst = 1;
37  end
38
39  load(strcat(Folder, File));
40
41  if ~isempty(varargin)
42      Name = [];
43  else
44      Name = input('Save as (default: same as input): ');
45  end
46
47  clear Condtionally_averaged px_calib Stats
48
49  if ~isempty(varargin)
50      if Reconst && numel(varargin) == 6
51          saveFolder = varargin{6};
```

```
52          rnames = {Name};
53      elseif ~Reconst && numel(varargin) == 4
54          saveFolder = varargin{4};
55          rnames = {Name};
56      else
57
58          if isempty(Name)
59              saveFolder = [pwd  '/Results/POD'];
60              rnames = {[pwd '/Results/POD']};
61          else
62              saveFolder = [Folder Name];
63              rnames = {[Folder Name]};
64          end
65      end
66  else
67      if isempty(Name)
68          saveFolder = [pwd  '/Results/POD'];
69          rnames = {[pwd  '/Results/POD']};
70      else
71          saveFolder = [Folder Name];
72          rnames = {[Folder Name]};
73      end
74  end
75
76
77  if exist(saveFolder, 'dir')~=7
78      mkdir(saveFolder)
79  end
80
81  if exist([saveFolder '/MatFigs/'], 'dir')~=7
82      mkdir([saveFolder '/MatFigs/'])
83  end
84
```

```
85 %Crop near wall
86 if exist('Cond','var')
87     if isfield(Cond,'ymin')
88         Yall = repmat(Y,[1 1 size(U,3)]);
89         U(Yall<1.1*Cond.ymin) = NaN;
90         V(Yall<1.1*Cond.ymin) = NaN;
91         clear Yall
92    end
93 end
94
95 %remember where the nans are
96 U = P_all.U;
97 V = P_all.W;
98 X = P_all.X;
99 Y = P_all.Z;
100 mask = U;
101 mask(:) = 0;
102 mask(isnan(U)) = 1;
103
104 %Find plotting bounds
105 Uf = U-repmat(nanmean(U,3),[1,1,size(U,3)]);
106 Vf = V-repmat(nanmean(V,3),[1,1,size(V,3)]);
107
108 Umean = nanmean(U,3);
109 Vmean = nanmean(V,3);
110
111 %must remove all nans
112 Vf(isnan(Vf)) = 0;
113 Uf(isnan(Uf)) = 0;
114
115 %% create matrix will all fluctuating velocity components for ...
116 %%each snapshot in a column
117 [uSize] = size(Uf);
```

```
118 Uall=[reshape(Uf,uSize(1)*uSize(2),uSize(3));reshape(Vf,uSize(1)*...
119 uSize(2),uSize(3))];
120
121
122 %Do POD analysis
123
124 R=Uall'*Uall; % Autocovariance matrix
125 clear Uall
126 [eV,D]=eig(R); % solve: eV is eigenvectors, D is eigenvalues in ...
127 %diagonal matrix
128 clear R
129 [L,I]=sort(diag(D)); % sort eigenvalues in ascending order - I ...
130 %is sorted index vector
131 for i=1:length(D)
132     eValue(length(D)+1-i)=L(i); % Eigenvalues sorted in descending order
133     eVec(:,length(D)+1-i)=eV(:,I(i)); % Eigenvectors sorted in same order
134     POD.eValue = eValue;
135     POD.eVec = eVec;
136     if rem(i,10) == 0
137         i
138     end
139 end
140
141 eValue(length(eValue))=0; % last eigenvalue should be zero
142 menergy=eValue/sum(eValue); % relative energy associated with mode m
143 POD.menergy = menergy;
144
145 menergy_sum = zeros(length(menergy),1);
146 for ii = 1:length(menergy)
147     menergy_sum(ii) = sum(menergy(1:ii));
148     POD.menergy_sum = menergy_sum;
149 end
150
```

```
151 %---------------Plot the energy of the modes--------------------------
152 end_mode = 200;
153 figure
154 [AX, H1, H2] = plotyy(1:end_mode,menergy(1:end_mode),1:end_mode,...
155 menergy_sum(1:end_mode),'bar','plot');
156 % title('Mode Energy','Interpreter','latex','fontsize',14)
157 xlabel('Mode number','Interpreter','latex','fontsize',18)
158 set(H1,'FaceColor',[.8 .8 .8])
159 set(H2,'Color','k','LineStyle','--','Linewidth',2)
160 set(AX(1),'YColor','k','xlim',[0.5 end_mode+.5],'FontSize',18,'ylim',...
161 [0 round(1.1*max(menergy)*100)/100],'YTick',linspace(0,round(1.1*max(menergy)*100)/100,4
162 % set(AX(1),'YColor','k','xlim',[0.5 end_mode+.5],'FontSize',18,'ylim',...
163 [0 0.09],'YTick',linspace(0,0.09,4))
164 set(AX(2),'YColor','k','xlim',[0.5 end_mode+.5],'FontSize',18,'ylim',...
165 [0 1],'YTick',linspace(0,1,4))
166 ylabel(AX(1),'Scaled mode energy','Interpreter','latex','FontSize',18)
167 ylabel(AX(2),'Integrated energy','Interpreter','latex','FontSize',18)
168 %xlabel(AX(2),'Mode number','Interpreter','latex','fontsize',18)
169
170 set(AX,'units', 'normalized', 'position', [0.22 0.15 0.6 0.8])
171 set(gcf, 'units', 'centimeters', 'PaperPosition', [1 1 4 7]);
172 % ylabel(AX(2),'Y2','rotation',0)
173
174 savefile = [saveFolder '/ModeEnergies', '.tif'];
175 set(gcf,'PaperPositionMode','auto')
176 print(savefile, '-dtiff');
177 saveas(gcf,[saveFolder '/MatFigs/ModeEnergies.fig'])
178
179 %% calculate the first n modes
180 if isempty(varargin)
181     while true
182         num_modes = input('How many modes would you like to use? ...
183         (default: 50) ');
```

```
184         UsePercent = input('Number or percent? (Number=0, Percent=1) ');
185
186         if isempty(num_modes)
187             num_modes = 50;
188             break
189         else
190             if num_modes <= uSize(3)
191                 break
192             else
193                 disp('There are not that many!')
194             end
195         end
196     end
197 end
198
199 if UsePercent
200    pTag = num_modes;
201    num_modes =  find(menergy_sum-num_modes/100 == ...
202    min(abs(menergy_sum-num_modes/100)));
203 end
204 pTag = num_modes;
205 eVec = eVec(:,1:num_modes);
206 %phi = zeros(uSize(1)*uSize(2)*uSize(3),length(num_modes));
207 phi = zeros(uSize(1)*uSize(2)*2,length(num_modes));
208
209 Uall=[reshape(Uf,uSize(1)*uSize(2),uSize(3));...
210 reshape(Vf,uSize(1)*uSize(2),uSize(3))];
211
212 for i=1:num_modes
213     tmp=Uall*eVec(:,i); % find mode... it is a hypothesis of snapshot POD
214                         % the mode is the eigenvector multiplied by the
215                         % initial velocity field
216     phi(:,i)=tmp/norm(tmp); % normalize mode
```

```
217                         % Must normalize the modes so that the eigenvalue
218                         % is the energy of the mode
219 end
220
221 %% plotting and saving the modes!
222 L = 36;
223 mode_plots = 25;
224 if mode_plots > uSize(3)
225     mode_plots = uSize(3);
226 end
227
228 % phi_mode_U= U(:,:,1:mode_plots);
229 % phi_mode_U(:) = NaN;
230 % phi_mode_V(:) = NaN;
231
232 vel{1} = 'U';
233 vel{2} = 'V';
234
235 for ii = 1 : mode_plots
236
237     for kk = 1:length(vel)
238         eval(['phi_mode_' vel{kk} ' = reshape(phi((kk-1)*uSize(1)*...
239         uSize(2)+1:kk*uSize(1)*uSize(2),ii), uSize(1), uSize(2));'])
240         if kk==1
241             POD.modes(ii).U = reshape(phi((kk-1)*uSize(1)*uSize(2)+1:kk..
242             *uSize(1)*uSize(2),ii), uSize(1), uSize(2));
243         else
244             POD.modes(ii).V = reshape(phi((kk-1)*uSize(1)*uSize(2)+1:kk..
245             *uSize(1)*uSize(2),ii), uSize(1), uSize(2));
246         end
247     end
248
249     figure(1)
```

```
250     clf
251     for jj = 1: length(vel)
252         subplot(1, length(vel),jj)
253         eval(['pcolor(X/(0.0254*L),Y/(0.0254*L),phi_mode_' vel{jj} ')'])
254         title([vel{jj} ' - # ', num2str(ii)])
255         colorbar('SouthOutside')
256         shading interp
257         axis('equal', 'tight')
258         %set(gca, 'XDir', 'reverse')
259         eval(['bound = max([max(phi_mode_'...
260         vel{jj} ') abs(min(phi_mode_' vel{jj} '))]);'])
261         %bound = bound*0.5;
262         caxis([-1*bound, bound])
263         xlabel('x/L')
264         ylabel('z/L')
265     end
266
267     savefile = [saveFolder '/ModeEnergies_' num2str(ii) '.tif'];
268     set(gcf,'PaperPositionMode','auto')
269     print(savefile, '-dtiff');
270     saveas(gcf,[saveFolder '/MatFigs/ModeEnergies_' num2str(ii) '.fig'])
271 end
272
273 Reconst = 1;
274
275 %% Reconstructing the velocities (projecting on original data)
276 if Reconst
277
278         %uTot = loadingVel(velFile{ii},vel,coordSys, (y_mode.^2+z_mode.^2));
279         alpha = phi'*Uall;
280                                 % alpha says when each mode is active in time.
281                                 % A column of alpha is the activity in time for
282                                 % a mode which is equal to the column
```

```
283         %clear uTot
284         uRec = phi*alpha;   %Calculates reconstructed field by multiplying
285                             %each mode by when it is active in each
286                             %velocity field
287
288         alpha = shiftdim(alpha,1);
289
290         for kk = 1:length(vel)
291             eval([vel{kk} '_rec = reshape(uRec((kk-1)*uSize(1)*uSize(2)...
292             +1:kk*uSize(1)*uSize(2),:), uSize);'])
293         end
294
295         clear uRec
296
297         %Add mean field back into solution
298         for i = 1:uSize(3)
299             U(:,:,i) = U_rec(:,:,i) + Umean;
300             V(:,:,i) = V_rec(:,:,i) + Vmean;
301         end
302
303         %Apply all the NaNs back into the data
304         U(mask==1) = NaN;
305         V(mask==1) = NaN;
306
307         POD.phi = phi;
308         POD.uSize = uSize;
309         POD.alpha = alpha;
310
311         if UsePercent
312             save([Folder '/' File '-POD-' num2str(pTag) 'p.mat'], 'X',..
313             'Y','U','V','POD','-v7.3')
314         else
315             save([Folder '/' File '-POD-' num2str(num_modes) '.mat'], ...
```

```
316             'X','Y','U','V','POD','-v7.3')
317         end
318
319
320         clear U V X Y Cond PercentMissing Source Uf Vf Uall U_rec V_rec
321         if exist('SecondProjFile','var')
322             load(SecondProjFile);
323
324             %remember where the nans are
325             mask = U;
326             mask(:) = 0;
327             mask(isnan(U)) = 1;
328
329             %must remove all nans
330             V(isnan(V)) = 0;
331             U(isnan(U)) = 0;
332
333             uSize = size(U);
334             Umean = mean(U,3);
335             Vmean = mean(V,3);
336
337             for i = 1:size(U,3)
338                 Uf(:,:,i) = U(:,:,i)-Umean;
339                 Vf(:,:,i) = V(:,:,i)-Vmean;
340             end
341
342
343             [uSize] = size(Uf);
344             Uall=[reshape(Uf,uSize(1)*uSize(2),uSize(3));reshape(Vf,...
345             uSize(1)*uSize(2),uSize(3))];
346
347             alpha = phi'*Uall;
348             uRec = phi*alpha;
```

```
            alpha = shiftdim(alpha,1);
            for kk = 1:length(vel)
                eval([vel{kk} '_rec = reshape(uRec((kk-1)*uSize(1)*...
                uSize(2)+1:kk*uSize(1)*uSize(2),:), uSize);'])
            end

            clear uRec

            for i = 1:uSize(3)
                U(:,:,i) = U_rec(:,:,i) + Umean;
                V(:,:,i) = V_rec(:,:,i) + Vmean;
            end

            %Apply all the NaNs back into the data
            U(mask==1) = NaN;
            V(mask==1) = NaN;

            if usePercent
                save([Folder '/' SecondProjFile '-POD-' num2str(pTag) ...
                'p.mat'], 'X','Y','U','V','POD','-v7.3')
            else
                save([Folder '/' SecondProjFile '-POD-'...
                num2str(num_modes) '.mat'], 'X','Y','U','V','POD','-v7.3')
            end
        end
end
```

Listing B.9: Mean Streamwise and Spanwise Pressure Plots

```
% Create Mean Pressure Plots for 40ms
clear
load('UWBUMPSP2020_EXP.mat')
font = 18;
```

```
5
6  streamwise = [];
7  spanwise = [];
8  cp = [];
9  cp_error=[];
10
11 for i = 1:48
12 streamwise = [streamwise UWBumpSP2020_EXP.S4_V40(i).X_L];
13
14 spanwise = [spanwise UWBumpSP2020_EXP.S4_V40(i).Y_L];
15
16 cp = [cp UWBumpSP2020_EXP.S4_V40(i).C_P];
17
18 cp_error = [cp_error UWBumpSP2020_EXP.S4_V40(i).delta_C_P];
19 end
20
21 figure
22 errorbar(streamwise(1:20),cp(1:20),cp_error(1:20),'ko',...
23 'MarkerFaceColor','k')
24 xlabel('x/L','interpreter','latex')
25 ylabel('$C_{p}$','interpreter','latex')
26 grid on
27 grid(gca,'minor')
28 set(gca,'FontSize',font)
29 set(gca,'TickLabelInterpreter','latex')
30 set(gca, 'LineWidth', 1.5)
31 %%
32 savefig(gcf,'Streamwise_Mean_Pressure_Centerline_Vel40.fig')
33 saveas(gcf,'Streamwise_Mean_Pressure_Centerline_Vel40.png')
34
35 close all
36
37 peak_indices = [7 21 22 33 34 37 38 39 40 43 44 47 48];
```

```
38
39 figure
40 errorbar(spanwise(peak_indices),cp(peak_indices),cp_error...
41 (peak_indices),'ko','MarkerFaceColor','k')
42 xlabel('y/L','interpreter','latex')
43 ylabel('$C_{p}$','interpreter','latex')
44 grid on
45 grid(gca,'minor')
46 set(gca,'FontSize',font)
47 set(gca,'TickLabelInterpreter','latex')
48 set(gca, 'LineWidth', 1.5)
49
50 savefig(gcf,'BumpPeak_Mean_Pressure_Centerline_Vel40.fig')
51 saveas(gcf,'BumpPeak_Mean_Pressure_Centerline_Vel40.png')
52
53 close all
54
55 %% Create Mean Pressure Plots for 60ms
56 clear
57 load('UWBUMPSP2020_EXP.mat')
58 font = 18;
59
60 streamwise = [];
61 spanwise = [];
62 cp = [];
63 cp_error=[];
64
65 for i = 1:48
66 streamwise = [streamwise UWBumpSP2020_EXP.S4_V60(i).X_L];
67
68 spanwise = [spanwise UWBumpSP2020_EXP.S4_V60(i).Y_L];
69
70 cp = [cp UWBumpSP2020_EXP.S4_V60(i).C_P];
```

```
71
72 cp_error = [cp_error UWBumpSP2020_EXP.S4_V60(i).delta_C_P];
73 end
74
75 figure
76 errorbar(streamwise(1:20),cp(1:20),cp_error(1:20),'ko',...
77 'MarkerFaceColor','k')
78 xlabel('x/L','interpreter','latex')
79 ylabel('$C_{p}$','interpreter','latex')
80 grid on
81 grid(gca,'minor')
82 set(gca,'FontSize',font)
83 set(gca,'TickLabelInterpreter','latex')
84 set(gca, 'LineWidth', 1.5)
85
86 savefig(gcf,'Streamwise_Mean_Pressure_Centerline_Vel60.fig')
87 saveas(gcf,'Streamwise_Mean_Pressure_Centerline_Vel60.png')
88
89 close all
90
91 peak_indices = [7 21 22 33 34 37 38 39 40 43 44 47 48];
92
93 figure
94 errorbar(spanwise(peak_indices),cp(peak_indices),cp_error...
95 (peak_indices),'ko','MarkerFaceColor','k')
96 xlabel('y/L','interpreter','latex')
97 ylabel('$C_{p}$','interpreter','latex')
98 grid on
99 grid(gca,'minor')
100 set(gca,'FontSize',font)
101 set(gca,'TickLabelInterpreter','latex')
102 set(gca, 'LineWidth', 1.5)
103
```

```
savefig(gcf,'BumpPeak_Mean_Pressure_Centerline_Vel60.fig')
saveas(gcf,'BumpPeak_Mean_Pressure_Centerline_Vel60.png')

close all
```

Listing B.10: Calculation of Pre-Multiplied Power Spectral Density Pressure

```
clear
clc

addpath 'E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Hari_cond_avg\Pressure\Synchronized_pressure'
%%
P_signal = [];
time_signal = [];
%tap = input('What tap do you want?');
tap = [1 2 3 4 5 6 7 8 9 10 11 12];
for i = 1:12
%tunnelconditionname = ['tunnel_cond_' num2str(i) '.mat'];
%syncname = ['SensorSync_' num2str(i) '.mat'];
LSname = ['LS' num2str(i) '_P.mat'];

%load(tunnelconditionname);
%load(syncname);
load(LSname);
P_signal = [P_signal P.p_v(tap,:)];
time_signal = [time_signal P.time];

clear tunnelconditionname syncname LSname P1_amb Uinf meanP delta_P ...
Cp idxPIV nPIVlost P qflag
clear tunnel_cond
end
%%
```

```
%plot(time_signal,P_signal)
clear pxx fxx
pxx = [];
fxx = [];

for i = 1:length(tap)
[pxx_temp fxx_temp] = pwelch(P_signal...
(i,1:7500000)-mean(P_signal(i,1:7500000)),[],[],[],31250);
pxx(i,:) = pxx_temp;
fxx(i,:) = fxx_temp;

clear pxx_temp fxx_temp
end

%% Total view pre-multipled vs F
font = 16;
load('Vel40_yPos0_Bump_LongSampleUinf.mat')
for i = 1:12
    Uinf_all = Uinfinity(i).Uinfy;
end

Uinf_mean = mean(Uinf_all);
Lsep_x = 0.2235; %x/L units
Lsep_x = Lsep_x*36*0.0254; %m

for i = 1:12
%subplot(3,1,i)
%figure
%semilogx(fxx(i,:),fxx(i,:).*pxx(i,:)-20000*i)
plot(log10(fxx(i,:)),...
    fxx(i,:).*pxx(i,:)-20000*i)
grid on

```

```
hold on
end
legend('Tap 3', 'Tap 4','Tap 5','Tap 6','Tap 7', 'Tap 8 ', 'Tap 9',...
'Tap 10','Tap 11', 'Tap 12', 'Tap 13', 'Tap 14',...
    'interpreter','latex')
%legend('Tap 7', 'Tap 8 ', 'Tap 9', 'Tap 10'...
%    ,'Tap 11', 'Tap 12', 'Tap 13', 'Tap 14',...
%    'interpreter','latex')
%xlim([min(min(fxx)), 1000])
ylim([-275000 0])
xlim([0 3.25])
ylabel('f*PSD (dB)','interpreter','latex')
set(gca,'FontSize',font)
xlabel('Frequency(Hz)','interpreter','latex')


%% Total view pre-multipled vs Strouhal All
font = 16;
load('Vel40_yPos0_Bump_LongSampleUinf.mat')
for i = 1:12
    Uinf_all = Uinfinity(i).Uinfy;
end

Uinf_mean = mean(Uinf_all);
Lsep_x = 0.2235; %x/L units
Lsep_x = Lsep_x*36*0.0254; %m

for i = 1:12
%subplot(3,1,i)
%figure
plot(log10(fxx(i,:)*Lsep_x/Uinf_mean),...
    fxx(i,:).*pxx(i,:)-20000*i)
grid on
```

```
93
94  hold on
95  end
96
97  legend('Tap 3', 'Tap 4','Tap 5','Tap 6','Tap 7', 'Tap 8 ', 'Tap 9', ...
98  'Tap 10','Tap 11', 'Tap 12', 'Tap 13', 'Tap 14',...
99      'interpreter','latex')
100 %legend('Tap 7', 'Tap 8 ', 'Tap 9', 'Tap 10'...
101 %    ,'Tap 11', 'Tap 12', 'Tap 13', 'Tap 14',...
102 %    'interpreter','latex')
103 %xlim([min(min(fxx)), 1000])
104 ylim([-275000 0])
105 %xlim([0 3.25])
106 ylabel('f*PSD (dB)','interpreter','latex')
107 set(gca,'FontSize',font)
108 xlabel('St (-)','interpreter','latex')
109
110 %% Total view pre-multipled vs Strouhal Upstream Taps
111 font = 16;
112 load('Vel40_yPos0_Bump_LongSampleUinf.mat')
113 for i = 1:4
114     Uinf_all = Uinfinity(i).Uinfy;
115 end
116
117 Uinf_mean = mean(Uinf_all);
118 Lsep_x = 0.2235; %x/L units
119 Lsep_x = Lsep_x*36*0.0254; %m
120
121 for i = 1:4
122 %subplot(3,1,i)
123 %figure
124 semilogx((fxx(i,:)*Lsep_x/Uinf_mean),...
125     fxx(i,:).*pxx(i,:)-20000*i)
```

```
126 grid on
127
128 hold on
129 end
130
131 legend('Tap 3', 'Tap 4','Tap 5','Tap 6'...
132     ,'interpreter','latex')
133 %xlim([min(min(fxx)), 1000])
134 ylim([-90000 0])
135 %xlim([0 3.25])
136 ylabel('f*PSD (dB)','interpreter','latex')
137 set(gca,'FontSize',font)
138 xlabel('St (-)','interpreter','latex')
139
140 %% Total view pre-multipled vs Strouhal Downstream Taps
141 font = 16;
142 load('Vel40_yPos0_Bump_LongSampleUinf.mat')
143 for i = 5:12
144     Uinf_all = Uinfinity(i).Uinfy;
145 end
146
147 Uinf_mean = mean(Uinf_all);
148 Lsep_x = 0.247; %x/L units
149 Lsep_x = Lsep_x*36*0.0254; %m
150
151 for i = 5:12
152 %subplot(3,1,i)
153 %figure
154 %plot(log10(fxx(i,:)*Lsep_x/Uinf_mean),...
155     %fxx(i,:).*pxx(i,:)-20000*i+80000)
156  semilogx((fxx(i,:)*Lsep_x/Uinf_mean), fxx(i,:).*pxx(i,:)-20000*i+80000)
157 grid on
158
```

```
159 hold on
160 end
161
162 legend('Tap 7', 'Tap 8 ', 'Tap 9', 'Tap 10','Tap 11', 'Tap 12'...
163     ,'Tap 13','Tap 14', 'interpreter','latex')
164 %xlim([min(min(fxx)), 1000])
165 ylim([-160000 0])
166 %xlim([0 3.25])
167 ylabel('f*PSD (dB)','interpreter','latex')
168 set(gca,'FontSize',font)
169 xlabel('St (-)','interpreter','latex')
170 %% Total view pre-multipled vs Strouhal Downstream Taps 11-14
171 font = 16;
172 load('Vel40_yPos0_Bump_LongSampleUinf.mat')
173 for i = 1:4
174     Uinf_all = Uinfinity(i).Uinfy;
175 end
176
177 Uinf_mean = mean(Uinf_all);
178 Lsep_x = 0.2235; %x/L units
179 Lsep_x = Lsep_x*36*0.0254; %m
180
181 for i = 9:12
182 %subplot(3,1,i)
183 %figure
184 plot(log10(fxx(i,:)*Lsep_x/Uinf_mean),...
185     fxx(i,:).*pxx(i,:)-20000*i+160000)
186 grid on
187
188 hold on
189 end
190
191 legend('Tap 9', 'Tap 10', 'Tap 11', 'Tap 12'...
```

```
    ,'interpreter','latex')
%xlim([min(min(fxx)), 1000])
ylim([-80000 0])
%xlim([0 3.25])
ylabel('f*PSD (dB)','interpreter','latex')
set(gca,'FontSize',font)
xlabel('St (-)','interpreter','latex')
```

Listing B.11: Create Histograms of High Frequency Pressure Data

```
addpath 'E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Hari_cond_avg\Pressure\Synchronized_pressure\Conditionally_averaged\Tap 7'
addpath 'E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Hari_cond_avg\Pressure\Synchronized_pressure\Conditionally_averaged\Tap 8'
addpath 'E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Hari_cond_avg\Pressure\Synchronized_pressure\Conditionally_averaged\Tap 9'
addpath 'E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Hari_cond_avg\Pressure\Synchronized_pressure\Conditionally_averaged\Tap 10'
addpath 'E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Hari_cond_avg\Pressure\Synchronized_pressure\Conditionally_averaged\Tap 11'
addpath 'E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Hari_cond_avg\Pressure\Synchronized_pressure\Conditionally_averaged\Tap 12'
addpath 'E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Hari_cond_avg\Pressure\Synchronized_pressure\Conditionally_averaged\Tap 13'
addpath 'E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Hari_cond_avg\Pressure\Synchronized_pressure\Conditionally_averaged\Tap 14'
%%
figure
font = 12;
for i = 7:14
name = ['Tap',num2str(i),'Conditionally_Avg_pressure.mat'];
load(name)
hold on
```

```
pressure_hist = histogram(Conditionally_averaged_pressure.pressure_info);
xlabel('Differential Pressure (Pa)','interpreter','latex','FontSize',font)
ylabel('Frames','interpreter','latex','FontSize',16)
end

set(gca,'FontSize',font)

legend('Tap 7', 'Tap 8', 'Tap 9' , 'Tap 10', 'Tap 11', 'Tap 12', ...
'Tap 13', 'Tap 14')
```

Listing B.12: Pressure Distribution Statistics Calculation

```
clear
clc

addpath 'E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Hari_cond_avg\Pressure\Synchronized_pressure'
font = 14;
load('Vel40_yPos0_Bump_LongSampleUinf.mat')
%%
P_signal = [];
time_signal = [];
%tap = input('What tap do you want?');
tap = [5 6 7 8 9 10 11 12];
for i = 1:8
tunnelconditionname = ['tunnel_cond_' num2str(i) '.mat'];
syncname = ['SensorSync_' num2str(i) '.mat'];
LSname = ['LS' num2str(i) '_P.mat'];

load(tunnelconditionname);
load(syncname);
load(LSname);
i
```

```
mean(tunnel_cond(end).rho)
meanP = mean(P.p_v,2);

temp = (P.p_v(tap,:)-meanP(1))./(1/2*mean(tunnel_cond(end).rho)*...
(Uinfinity(i).Uinfy).^2);

P_signal = [P_signal temp];
time_signal = [time_signal P.time];

clear temp tunnelconditionname syncname LSname P1_amb Uinf meanP ...
delta_P Cp idxPIV nPIVlost P qflag
clear tunnel_cond
end
%% All histograms together
for i = 1:8
    histogram(P_signal(i,:),100,'DisplayStyle','stairs','normalization'...
        ,'pdf')
    hold on
    xlabel('$C_{p}$','interpreter','latex','FontSize',font)
    ylabel('p.d.f.','interpreter','latex','FontSize',font)

end

legend('Tap 7', 'Tap 8 ', 'Tap 9', 'Tap 10'...
    ,'Tap 11', 'Tap 12', 'Tap 13', 'Tap 14',...
    'interpreter','latex','Location','NorthEast')

set(gca,'TickLabelInterpreter','latex')
set(gca,'FontSize',14)

png_name = strcat('All_histograms_centerline.png');
saveas(gcf,[pwd '\AllTogether\' png_name])

```

```
55 fig_name = strcat('All_histograms_centerline.fig');
56 saveas(gcf,[pwd '\AllTogether\'  fig_name])
57 set(gca,'interpreter',font)
58
59 close all
60 %% Individual histograms
61 for i = 1:8
62     histogram(P_signal(i,:),100,'DisplayStyle','stairs','normalization'...
63         ,'pdf')
64     hold on
65     xlabel('$C_{p}$','interpreter','latex','FontSize',font)
66     ylabel('p.d.f.','interpreter','latex','FontSize',font)
67     set(gca,'TickLabelInterpreter','latex')
68     set(gca,'FontSize',font)
69
70     png_name = strcat('Pressure_Dist_Tap ', num2str(i+2),'.png');
71     saveas(gcf,[pwd '\Pngs\' png_name])
72
73     fig_name = strcat('Pressure_Dist_Tap ', num2str(i+2),'.fig');
74     saveas(gcf,[pwd '\Figs\' fig_name])
75     close all
76
77 end
78
79 %% Statistics calculation
80 P_signal = [];
81 time_signal = [];
82 %tap = input('What tap do you want?');
83 tap = [1 2 3 4 5 6 7 8 9 10 11 12];
84 for i = 1:12
85 tunnelconditionname = ['tunnel_cond_' num2str(i) '.mat'];
86 syncname = ['SensorSync_' num2str(i) '.mat'];
87 LSname = ['LS' num2str(i) '_P.mat'];
```

```
88
89 load(tunnelconditionname);
90 load(syncname);
91 load(LSname);
92
93 meanP = mean(P.p_v,2);
94
95 temp = (P.p_v(tap,:)-meanP(1))./(1/2*mean(tunnel_cond(end).rho)*...
96 (Uinfinity(i).Uinfy).^2);
97
98 P_signal = [P_signal temp];
99 time_signal = [time_signal P.time];
100
101 clear temp tunnelconditionname syncname LSname P1_amb Uinf meanP delta_P ...
102 Cp idxPIV nPIVlost P qflag
103 clear tunnel_cond
104 end
105
106 %%
107 P_signal = P_signal';
108 %%
109 font = 18;
110 mean_all = mean(P_signal,1,'omitnan');
111 standard_deviation = std(P_signal,1,'omitnan');
112 skew = skewness(P_signal(:,1:12)-mean_all);
113 variance = standard_deviation.^2;
114 %% Plot variance and skewness against streamwise location
115 tap_location_info = readtable('UWBUMPSP2020_EXP_TapLocations.csv');
116 plot(tap_location_info.X_L(4:15),variance,'*')
117 xlabel('x/L','interpreter','latex')
118 ylabel('$\sigma_{C_{p}}^2$','interpreter','latex')
119 grid on
120 %set(gca,'XMinorGrid','on');
```

```
121 %set(gca,'YMinorGrid','on');
122 set(gca,'FontSize',font)
123
124 png_name = strcat('Variance_xL.png');
125 saveas(gcf,[pwd '\Pngs\' png_name])
126
127 fig_name = strcat('Variance_xL.fig');
128 saveas(gcf,[pwd '\Figs\' fig_name])
129 close all
130
131 tap_location_info = readtable('UWBUMPSP2020_EXP_TapLocations.csv');
132 yyaxis right
133 plot(tap_location_info.X_L(4:15),variance,'*')
134 xlabel('x/L','interpreter','latex')
135 ylabel('$\sigma_{C_{p}}^2$','interpreter','latex')
136
137 L = 36;
138 L_b = 35.5;
139 x_0 = 0.195 * L_b;
140 y_0 = 0.06 * L_b;
141 h_0 = 0.085* L_b;
142 Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
143 (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
144 z_0 = 0;
145 x_wall = linspace(-0.3698,.4314,200);
146 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
147 z_wall = zeros(1,length(x_wall));
148 z_wall(1:min(bump_ind)-1) = z_0;
149 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
150
151 span = 0;
152
153 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
```

```
154 ones(1,length(bump_ind)).*span*0.0393701);
155 hold on
156 yyaxis left
157 plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
158 ylabel('z/L','interpreter','latex')
159 xlim([-0.3698 .4314])
160 hold on
161 grid on
162 %set(gca,'XMinorGrid','on');
163 %set(gca,'YMinorGrid','on');
164 set(gca,'FontSize',font)
165
166 png_name = strcat('Variance_with_bump.png');
167 saveas(gcf,[pwd '\Pngs\' png_name])
168
169 fig_name = strcat('Variance_with_bump.fig');
170 saveas(gcf,[pwd '\Figs\' fig_name])
171 close all
172 %%
173 tap_location_info = readtable('UWBUMPSP2020_EXP_TapLocations.csv');
174 plot(tap_location_info.X_L(4:15),skew,'*')
175 xlabel('x/L','interpreter','latex')
176 ylabel('$\tilde{\mu_{3}}$','interpreter','latex')
177 grid on
178 %set(gca,'XMinorGrid','on');
179 %set(gca,'YMinorGrid','on');
180 set(gca,'FontSize',font)
181
182 png_name = strcat('Skew_xL.png');
183 saveas(gcf,[pwd '\Pngs\' png_name])
184
185 fig_name = strcat('Skew_xL.fig');
186 saveas(gcf,[pwd '\Figs\' fig_name])
```

```
187 close all
188
189 tap_location_info = readtable('UWBUMPSP2020_EXP_TapLocations.csv');
190 yyaxis right
191 plot(tap_location_info.X_L(4:15),skew,'*')
192 xlabel('x/L','interpreter','latex')
193 ylabel('$\tilde{\mu_{3}}$','interpreter','latex')
194 grid on
195
196 L = 36;
197 L_b = 35.5;
198 x_0 = 0.195 * L_b;
199 y_0 = 0.06 * L_b;
200 h_0 = 0.085* L_b;
201 Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
202 (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
203 z_0 = 0;
204 x_wall = linspace(-0.3698,.4314,200);
205 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
206 z_wall = zeros(1,length(x_wall));
207 z_wall(1:min(bump_ind)-1) = z_0;
208 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
209
210 span = 0;
211
212 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
213 ones(1,length(bump_ind)).*span*0.0393701);
214 hold on
215 yyaxis left
216 plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
217 ylabel('z/L','interpreter','latex')
218 xlim([-0.3698 .4314])
219 hold on
```

```
grid on
%set(gca,'XMinorGrid','on');
%set(gca,'YMinorGrid','on');
set(gca,'FontSize',font)


%set(gca,'XMinorGrid','on');
%set(gca,'YMinorGrid','on');
set(gca,'FontSize',font)

hold on
xline(0)
hold on
yline(0)

png_name = strcat('Skewness_with_bump.png');
saveas(gcf,[pwd '\Pngs\' png_name])

fig_name = strcat('Skewness_with_bump.fig');
saveas(gcf,[pwd '\Figs\' fig_name])
close all
```

Listing B.13: Conditionally Average PIV Fields High Frequency Pressure Data

```
addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Vel40_yPos0_Bump_LongSample\MATLAB Processed')
filename = uigetfile('.mat','Select Processed PIV .mat file');
addpath('E:\Initial Testing - 2021\BB-Fall2021_Streamwise_LFOV\...
Hari_cond_avg\Pressure\Synchronized_pressure')
load(filename)
%%
clearvars -except FullField P_all Stats frame_data_all

```

```
%% Bump Properties
L = 36;
font = 12; %16 for presentations, 12 for papers
Uinf = 40;

frames_count = [2000 4000 2000 4000 4000 4000 4000 2000 4000 2000 ...
4000 4000];

for pos = 1:12
    if pos == 1
        frame_data_all(pos).U = P_all(1).U(:,:,1:frames_count(1));
        frame_data_all(pos).W = P_all(1).W(:,:,1:frames_count(1));
        frame_data_all(pos).X = P_all(1).X;
        frame_data_all(pos).Z = P_all(1).Z;
%         start_index = 1
%         end_index = frames_count(1)
    else
    frame_data_all(pos).U = P_all(1).U(:,:,sum(frames_count(1:pos-1))...
    +1:sum(frames_count(1:pos)));
    frame_data_all(pos).W = P_all(1).W(:,:,sum(frames_count(1:pos-1))...
    +1:sum(frames_count(1:pos)));
    frame_data_all(pos).X = P_all(1).X;
    frame_data_all(pos).Z = P_all(1).Z;
%     start_index = sum(frames_count(1:pos-1))+1
%     end_index = sum(frames_count(1:pos))
    end
end
%%
tap_cond_avg = input('Which number tap do you want to use?');

U_frames = [];
W_frames = [];
pressure_data = [];
```

```
for pos = 1:12
    tunnelconditionname = ['tunnel_cond_' num2str(pos) '.mat'];
    syncname = ['SensorSync_' num2str(pos) '.mat'];
    LSname = ['LS' num2str(pos) '_P.mat'];

    load(tunnelconditionname);
    load(syncname);
    load(LSname);

    X = frame_data_all(pos).X;
    Z = frame_data_all(pos).Z;

    tap_index = find(P.taps == tap_cond_avg);

    U_frames_temp = frame_data_all(pos).U(:,:,idxPIV);
    W_frames_temp = frame_data_all(pos).W(:,:,idxPIV);

    pressure_data_temp = P.p_v(tap_index,qFlag);

    U_frames = cat(3,U_frames, U_frames_temp);
    W_frames = cat(3,W_frames, W_frames_temp);
    pressure_data = [pressure_data pressure_data_temp];
end

for pos=1:1
    %% Historgram of Pressures
    figure
    pressure_hist = histogram(pressure_data);
    xlabel('Differential Pressure (Pa)','interpreter','latex',...
    'FontSize',font)
    ylabel('Frames','interpreter','latex','FontSize',16)
    pressure_fig_nam = strcat('Tap', num2str(tap_cond_avg),...
    'Pressure',num2str(pos),'_Uinf_',num2str(Uinf),'.fig');
```

```
76      pressure_png_nam = strcat('Tap', num2str(tap_cond_avg), ...
77      'Pressure',num2str(pos),'_Uinf_',num2str(Uinf),'.png');
78      savefig(pressure_fig_nam)
79      saveas(gcf,pressure_png_nam)
80
81      %% Extract Indicies of Velocity Fields in each pressure bin
82      pressure_bins = get(pressure_hist,'BinEdges');
83      pressure_counts = get(pressure_hist,'BinCounts');
84      pressure_indices = [];
85      for i = 1:(length(pressure_bins)-1)
86          for j = 1:length(pressure_data)
87              if (pressure_data(j) >= pressure_bins(i) && pressure_data(j) ...
88              < pressure_bins(i+1))
89                  pressure_index = j;
90                  pressure_indices = [pressure_indices pressure_index];
91              else
92
93              end
94          end
95      end
96
97      %% Group binned fields together and average fields
98      for p = 1:length(pressure_counts)
99          if p == 1
100             groups = pressure_indices(1:pressure_counts(p));
101             U_conditional_avg(:,:,p) = sum(U_frames(:,:,groups),3,...
102             'omitnan')/length(groups);
103             W_conditional_avg(:,:,p) = sum(W_frames(:,:,groups),3,...
104             'omitnan')/length(groups);
105             U_cond_avg_std(:,:,p) = std(U_frames(:,:,groups),0,3,...
106             'omitnan');
107             W_cond_avg_std(:,:,p) = std(W_frames(:,:,groups),0,3,...
108             'omitnan');
```

```
109             for i = 1:222
110                 for j= 1:272
111                     if U_conditional_avg(i,j,p) == 0
112                         U_conditional_avg (i,j,p) = NaN;
113                         W_conditional_avg (i,j,p) = NaN;
114                         U_cond_avg_std(i,j,p) = NaN;
115                         W_cond_avg_std(i,j,p) = NaN;
116                     end
117                 end
118             end
119         else
120             groups = pressure_indices(sum(pressure_counts(1:p-1))+...
121             1:sum(pressure_counts(1:p)));
122             U_conditional_avg(:,:,p) = sum(U_frames(:,:,groups),3,...
123             'omitnan')/length(groups);
124             W_conditional_avg(:,:,p) = sum(W_frames(:,:,groups),3,...
125             'omitnan')/length(groups);
126             U_cond_avg_std(:,:,p) = std(U_frames(:,:,groups),0,3,...
127             'omitnan');
128             W_cond_avg_std(:,:,p) = std(W_frames(:,:,groups),0,3,...
129             'omitnan');
130             for i = 1:222
131                 for j= 1:272
132                     if U_conditional_avg(i,j,p) == 0
133                         U_conditional_avg (i,j,p) = NaN;
134                         W_conditional_avg (i,j,p) = NaN;
135                         U_cond_avg_std(i,j,p) = NaN;
136                         W_cond_avg_std(i,j,p) = NaN;
137                     end
138                 end
139             end
140         end
141     end
```

```
142 Conditionally_averaged_pressure(pos).U = U_conditional_avg;
143 Conditionally_averaged_pressure(pos).W = W_conditional_avg;
144 Conditionally_averaged_pressure(pos).Ustd = U_cond_avg_std;
145 Conditionally_averaged_pressure(pos).Wstd = W_cond_avg_std;
146 Conditionally_averaged_pressure(pos).X = X;
147 Conditionally_averaged_pressure(pos).Z = Z;
148 Conditionally_averaged_pressure(pos).pressure_info = pressure_data;
149 Conditionally_averaged_pressure(pos).bins = pressure_bins;
150 Conditionally_averaged_pressure(pos).counts = pressure_counts;
151 Conditionally_averaged_pressure(pos).indices = pressure_indices;
152 clear Negative_area Positive_area neg_hist U_conditional_avg ...
153 W_conditional_avg groups neg_bins neg_counts neg_indices U_cond_avg_std...
154 W_cond_avg_std
155 end
156 close all
157
158 fileStats = strcat('Tap', num2str(tap_cond_avg),...
159 'Conditionally_Avg_pressure.mat');
160 save([fileStats],'Conditionally_averaged_pressure','-v7.3')
```

Listing B.14: Create Conditionally Averaged PIV Fields from Pressure Data and Automatic Determination of Separation and Reattachment Point

```
1 clearvars -except FullField P_all Stats
2 tap_cond_avg = input('tap number?')
3 %% Velocity fields
4 font = 12; %16 for Presentations, 12 for papers
5 L=36;
6 for pos=1:length(Conditionally_averaged_pressure)
7     bins = Conditionally_averaged_pressure(pos).bins;
8     counts = Conditionally_averaged_pressure(pos).counts;
9     U_avg_test = Conditionally_averaged_pressure(pos).U; %6th frame
```

```
10      W_avg_test = Conditionally_averaged_pressure(pos).W; %6th frame
11      for bin_number = 1:length(counts)
12          U_plot = U_avg_test(:,:,bin_number);
13          W_plot = W_avg_test(:,:,bin_number);
14          X = Conditionally_averaged_pressure(pos).X; %m
15          Z = Conditionally_averaged_pressure(pos).Z; %m
16
17          H = pcolor(X/(L*0.0254),Z/(L*0.0254), U_plot);
18          colormap(jet), set(H,'edgecolor','none')
19          shading interp
20          caxis([-.3 1.5]);
21          ax = gca;
22          hold(ax);
23          x_data = get(H,'xdata');
24          z_data = get(H,'ydata');
25          u_data = get(H,'CData');
26          [M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
27          c.LineWidth = 3;
28          c.LineColor = 'white';
29          xlabel('x/L','interpreter','latex','FontSize',font)
30          ylabel('z/L','interpreter','latex','FontSize',font)
31          bar = colorbar;
32          caxis([-.3 1.5]);
33          bar.FontSize = font;
34          bar.Label.Interpreter = 'latex';
35          bar.Label.String = '$U/U_\infty$';
36          bar.TickLabelInterpreter = 'latex';
37          hold(ax);
38
39          L = 36;
40          L_b = 35.5;
41          x_0 = 0.195 * L_b;
42          y_0 = 0.06 * L_b;
```

```
43          h_0 = 0.085* L_b;
44          Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
45          (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
46          z_0 = 0;
47          x_wall = linspace(min(min(X)),max(max(X)),200);
48          bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
49          z_wall = zeros(1,length(x_wall));
50          z_wall(1:min(bump_ind)-1) = z_0;
51          z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
52
53          span = P_all(1).Input.Span/25.4;
54
55          z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
56          ones(1,length(bump_ind)).*span*0.0393701);
57          hold on
58          plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
59
60          close all
61      end
62  end
63  %% Save standard deviation plots for each bin
64  font = 12; %16 for Presentations, 12 for papers
65  L=36;
66
67  for pos=1:length(Conditionally_averaged_pressure)
68      bins = Conditionally_averaged_pressure(pos).bins;
69      counts = Conditionally_averaged_pressure(pos).counts;
70      U_cond_std = Conditionally_averaged_pressure(pos).Ustd; %6th frame
71      W_cond_std = Conditionally_averaged_pressure(pos).Wstd; %6th frame
72      for bin_number = 1:length(counts)
73
74          U_plot_std = U_cond_std(:,:,bin_number);
75          W_plot_std = W_cond_std(:,:,bin_number);
```

```
76          X = Conditionally_averaged_pressure(pos).X; %m
77          Z = Conditionally_averaged_pressure(pos).Z; %m
78
79          H = pcolor(X/(L*0.0254),Z/(L*0.0254), U_plot_std);
80          colormap(jet), set(H,'edgecolor','none')
81          shading interp
82          ax = gca;
83          hold(ax);
84          x_data = get(H,'xdata');
85          z_data = get(H,'ydata');
86          u_data = get(H,'CData');
87          xlabel('x/L','interpreter','latex','FontSize',font)
88          ylabel('z/L','interpreter','latex','FontSize',font)
89          bar = colorbar;
90          bar.FontSize = font;
91          bar.Label.Interpreter = 'latex';
92          bar.Label.String = '${U/U_\infty}_{sigma}$';
93          bar.TickLabelInterpreter = 'latex';
94          hold(ax);
95
96          L = 36;
97          L_b = 35.5;
98          x_0 = 0.195 * L_b;
99          y_0 = 0.06 * L_b;
100         h_0 = 0.085* L_b;
101         Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
102         (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
103         z_0 = 0;
104         x_wall = linspace(min(min(X)),max(max(X)),200);
105
106         bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
107         z_wall = zeros(1,length(x_wall));
108         z_wall(1:min(bump_ind)-1) = z_0;
```

```
109         z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
110
111         span = P_all(1).Input.Span/25.4;
112
113         z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
114         ones(1,length(bump_ind)).*span*0.0393701);
115         hold on
116         plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
117
118         close all
119
120     end
121 end
122
123 %% Add streamlines to separation frames
124 font = 12; %16 for Presentations, 12 for papers
125 L=36;
126
127 for pos = 1:length(P_all)
128
129     bins = Conditionally_averaged_pressure(pos).bins;
130     counts = Conditionally_averaged_pressure(pos).counts;
131     U_avg_test = Conditionally_averaged_pressure(pos).U;
132     W_avg_test = Conditionally_averaged_pressure(pos).W;
133
134     for bin_number = 1:length(counts)
135
136         figure
137         U_plot = U_avg_test(:,:,bin_number);
138         W_plot = W_avg_test(:,:,bin_number);
139         X = Conditionally_averaged_pressure(pos).X; %m
140         Z = Conditionally_averaged_pressure(pos).Z; %m
141
```

```
142         H = pcolor(X/(L*0.0254),Z/(L*0.0254), U_plot);
143         colormap(jet), set(H,'edgecolor','none')
144         shading interp
145         ax = gca;
146         hold(ax);
147         x_data = get(H,'xdata');
148         z_data = get(H,'ydata');
149         u_data = get(H,'CData');
150         [M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
151         c.LineWidth = 3;
152         c.LineColor = 'white';
153         xlabel('x/L','interpreter','latex','FontSize',font)
154         ylabel('z/L','interpreter','latex','FontSize',font)
155         bar = colorbar;
156         caxis([-.3 1.5]);
157         bar.FontSize = font;
158         bar.Label.Interpreter = 'latex';
159         bar.Label.String = '$U/U_\infty$';
160         bar.TickLabelInterpreter = 'latex';
161         hold(ax);
162
163
164          line = streamslice(X/(L*0.0254),Z/(L*0.0254),U_plot,W_plot);
165          set(line,'Color','black','LineWidth',0.05,'MarkerSize',0.01);
166
167
168         L = 36;
169         L_b = 35.5;
170         x_0 = 0.195 * L_b;
171         y_0 = 0.06 * L_b;
172         h_0 = 0.085* L_b;
173         Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
174         (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
```

```
175          z_0 = 0;
176          x_wall = linspace(min(min(X)),max(max(X)),200);
177
178          bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
179          z_wall = zeros(1,length(x_wall));
180          z_wall(1:min(bump_ind)-1) = z_0;
181          z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
182
183          span = P_all(1).Input.Span/25.4;
184
185          z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
186          ones(1,length(bump_ind)).*span*0.0393701);
187          hold on
188          plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
189
190          xlim([min(min(X))./(L*0.0254) max(max(X))./(L*0.0254)-0.001])
191
192          set(gca,'FontSize',font)
193          close all
194      end
195      close all
196  end
197
198  %% Determine mean fields
199  figure
200
201  pos = 1;
202  L = 36;
203  font = 12;
204
205  Umean = Stats(pos).U;
206  Wmean = Stats(pos).W;
207  X = Stats(pos).X;
```

```
208 Z = Stats(pos).Z;
209
210 H = pcolor(X/(0.0254*L),Z/(0.0254*L),Umean); %Uinf from calibrated data
211 colormap(jet), set(H,'edgecolor','none')
212 shading interp
213 ax = gca;
214 hold(ax);
215 x_data = get(H,'xdata');
216 z_data = get(H,'ydata');
217 u_data = get(H,'CData');
218 [M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
219 c.LineWidth = 3;
220 c.LineColor = 'w';
221 xlabel('x/L','interpreter','latex','FontSize',font)
222 ylabel('z/L','interpreter','latex','FontSize',font)
223 bar = colorbar;
224 caxis([-.6 1.5]);
225 bar.FontSize = font;
226 bar.Label.Interpreter = 'latex';
227 bar.Label.String = '$U/U_\infty$';
228 bar.TickLabelInterpreter = 'latex';
229 hold(ax)
230 %
231 line = streamslice(X/(L*0.0254),Z/(L*0.0254),Umean,Wmean);
232 set(line,'Color','black','LineWidth',0.05,'MarkerSize',0.01)
233
234 L = 36;
235 L_b = 35.5;
236 x_0 = 0.195 * L_b;
237 y_0 = 0.06 * L_b;
238 h_0 = 0.085* L_b;
239 Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
240 (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
```

```
241 z_0 = 0;
242 x_wall = linspace(min(min(X)),max(max(X)),200);
243
244 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
245 z_wall = zeros(1,length(x_wall));
246 z_wall(1:min(bump_ind)-1) = z_0;
247 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
248
249 span = P_all(1).Input.Span/25.4;
250
251 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
252 ones(1,length(bump_ind)).*span*0.0393701);
253 hold on
254 plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
255
256 set(gca,'FontSize',font)
257
258 % png_name = strcat('Pressure Mean Field.png');
259 % saveas(gcf,png_name)
260 %
261 % fig_name = strcat('Pressure Mean Field.fig');
262 % saveas(gcf,fig_name)
263
264 close all
265
266 %% Determine separation locations across frames
267 clear x_sep z_sep
268 font = 12; %16 for Presentations, 12 for papers
269 L=36;
270
271 for pos = 1:1
272
273     bins = Conditionally_averaged_pressure(pos).bins;
```

```
274     counts = Conditionally_averaged_pressure(pos).counts;
275     U_avg_test = Conditionally_averaged_pressure(pos).U;
276     W_avg_test = Conditionally_averaged_pressure(pos).W;
277
278     for bin_number = 1:length(counts)
279
280         figure
281         U_plot = U_avg_test(:,:,bin_number);
282         W_plot = W_avg_test(:,:,bin_number);
283         X = Conditionally_averaged_pressure(pos).X; %m
284         Z = Conditionally_averaged_pressure(pos).Z; %m
285
286         H = pcolor(X/(L*0.0254),Z/(L*0.0254), U_plot);
287         colormap(jet), set(H,'edgecolor','none')
288         shading interp
289         ax = gca;
290         hold(ax);
291         x_data = get(H,'xdata');
292         z_data = get(H,'ydata');
293         u_data = get(H,'CData');
294         [M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
295         c.LineWidth = 3;
296         c.LineColor = 'w';
297
298         xlabel('x/L','interpreter','latex','FontSize',font)
299         ylabel('z/L','interpreter','latex','FontSize',font)
300         bar = colorbar;
301         bar.FontSize = font;
302         bar.Label.Interpreter = 'latex';
303         bar.Label.String = '$U/U_\infty$';
304         bar.TickLabelInterpreter = 'latex';
305         caxis([-0.3 1.5])
306         hold(ax);
```

```
307
308         x_contour_locations = sort(M(1,:));
309         i=1;
310         if isempty(x_contour_locations)
311             x_sep(bin_number,pos) = NaN;
312         else
313             while x_contour_locations(i) == 0
314                 i=i+1;
315             end
316             x_sep(bin_number,pos) = x_contour_locations(i);
317         end
318
319         L = 36;
320         L_b = 35.5;
321         x_0 = 0.195 * L_b;
322         y_0 = 0.06 * L_b;
323         h_0 = 0.085* L_b;
324         Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
325         (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
326         z_0 = 0;
327         x_wall = linspace(min(min(X)),max(max(X)),200);
328
329         bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
330         z_wall = zeros(1,length(x_wall));
331         z_wall(1:min(bump_ind)-1) = z_0;
332         z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
333
334         span = 0;
335
336         z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
337         ones(1,length(bump_ind)).*span*0.0393701);
338         hold on
339         plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
```

```
340         z_sep(bin_number,pos) = Z_bump(x_sep(bin_number,pos)*(L),...
341         span*0.0393701)/(L);
342         hold on
343         plot(x_sep(bin_number,pos),z_sep(bin_number,pos),'*')
344         xlim([min(min(X))./(L*0.0254) max(max(X))./(L*0.0254)-0.001])
345     end
346     close all
347 end
348 
349 %% Calculate Mean Separation
350 clear x_contour_locations
351 figure
352 
353 pos = 1;
354 L = 36;
355 font = 12;
356 
357 Umean = Stats(pos).U;
358 Wmean = Stats(pos).W;
359 X = Stats(pos).X;
360 Z = Stats(pos).Z;
361 
362 H = pcolor(X/(0.0254*L),Z/(0.0254*L),Umean); %Uinf from calibrated data
363 colormap(jet), set(H,'edgecolor','none')
364 shading interp
365 ax = gca;
366 hold(ax);
367 x_data = get(H,'xdata');
368 z_data = get(H,'ydata');
369 u_data = get(H,'CData');
370 [M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
371 c.LineWidth = 3;
372 c.LineColor = 'w';
```

```
373 xlabel('x/L','interpreter','latex','FontSize',font)
374 ylabel('z/L','interpreter','latex','FontSize',font)
375 bar = colorbar;
376 bar.FontSize = font;
377 bar.Label.Interpreter = 'latex';
378 bar.Label.String = '$U/U_\infty$';
379 bar.TickLabelInterpreter = 'latex';
380 hold(ax)
381
382 x_contour_locations = sort(M(1,:));
383 i=1;
384 if isempty(x_contour_locations)
385     x_sep_mean(pos) = NaN;
386 else
387     while x_contour_locations(i) == 0
388         i=i+1;
389     end
390     x_sep_mean(pos) = x_contour_locations(i);
391 end
392
393
394 L = 36;
395 L_b = 35.5;
396 x_0 = 0.195 * L_b;
397 y_0 = 0.06 * L_b;
398 h_0 = 0.085* L_b;
399 Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
400 (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
401 z_0 = 0;
402 x_wall = linspace(min(min(X)),max(max(X)),200);
403
404 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
405 z_wall = zeros(1,length(x_wall));
```

```
406 z_wall(1:min(bump_ind)-1) = z_0;
407 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
408
409 span = 0;
410
411 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
412 ones(1,length(bump_ind)).*span*0.0393701);
413
414 z_sep_mean(pos) = Z_bump(x_sep_mean(pos)*(L),span*0.0393701)/(L);
415 hold on
416 %plot(x_sep_mean(pos),z_sep_mean(pos),'*')
417 close all
418 %% "histogram" separation points counts divided by total count to get pdf
419 std_dev_x = std(x_sep,'omitnan');
420 std_dev_z = std(z_sep,'omitnan');
421
422 %%
423 font = 18;
424 clear bar
425 pos = 1;
426 plot(x_sep,Conditionally_averaged_pressure(pos).counts'/...
427 sum(Conditionally_averaged_pressure(pos).counts)/((bins(2)-bins(1))...
428 /(1/2*1.16*44.6^2)),'o');
429 xlabel('x/L','interpreter','latex','FontSize',font)
430 ylabel('p.d.f','interpreter','latex','FontSize',font)
431
432 hold on
433 xline(x_sep_mean-2*std_dev_x,':','$-2\sigma$',...
434 'LabelHorizontalAlignment','left','FontSize',font,'interpreter','latex')
435 hold on
436 xline(x_sep_mean-std_dev_x,':','$-\sigma$',...
437 'LabelHorizontalAlignment','left','FontSize',font,'interpreter','latex')
438 hold on
```

```
xline(x_sep_mean,':','Mean','interpreter','latex','FontSize',font)
hold on
xline(x_sep_mean+std_dev_x,':','$+\sigma$','FontSize',font,...
'interpreter','latex')
hold on
xline(x_sep_mean+2*std_dev_x,':','$+2\sigma$',...
'LabelHorizontalAlignment','right','FontSize',font,...
'interpreter','latex')

set(gca,'FontSize',font)
%%
png_name = strcat('Tap', num2str(tap_cond_avg), 'xsep_pressure.png');
saveas(gcf,[png_name])

fig_name = strcat('Tap', num2str(tap_cond_avg), 'xsep_pressure.fig');
saveas(gcf,[fig_name])
close all
%%
figure
plot(z_sep,Conditionally_averaged_pressure(pos).counts'/...
sum(Conditionally_averaged_pressure(pos).counts),'o');
xlabel('z/L','interpreter','latex','FontSize',font)
ylabel('p.d.f','interpreter','latex','FontSize',font)

hold on
xline(z_sep_mean-2*std_dev_z,':','$-2\sigma$',...
'LabelHorizontalAlignment','left','FontSize',font,...
'interpreter','latex')
hold on
xline(z_sep_mean-std_dev_z,':','$-\sigma$',...
'LabelHorizontalAlignment','left','FontSize',font,...
'interpreter','latex')
hold on
```

```
472 xline(z_sep_mean,':','Mean','interpreter','latex')
473 hold on
474 xline(z_sep_mean+std_dev_z,':','$+\sigma$','FontSize',font,...
475 'interpreter','latex')
476 hold on
477 xline(z_sep_mean+2*std_dev_z,':','$+2\sigma$',...
478 'LabelHorizontalAlignment','right','FontSize',font,...
479 'interpreter','latex')
480 set(gca,'FontSize',font)
481
482 png_name = strcat('Tap', num2str(tap_cond_avg), 'zsep_pressure.png');
483 saveas(gcf,[png_name])
484
485 fig_name = strcat('Tap', num2str(tap_cond_avg), 'zsep_pressure.fig');
486 saveas(gcf,[fig_name])
487
488 close all
489
490 %% Plot separation locations on Bump Space
491 separation_properties(:,1) = x_sep';
492 separation_properties(:,2) = z_sep';
493 separation_properties(:,3) = counts;
494
495 scatter(x_sep(1:1:end),z_sep(1:1:end),100,counts(1:1:end)/sum(counts)*...
496 100,'filled')
497 H = colorbar;
498 H.Label.String = '\% Probability of Separation Occuring at Location';
499 H.Label.Interpreter = 'latex';
500 H.Label.FontSize = font;
501 H.TickLabelInterpreter = 'latex';
502 H.FontSize = font;
503
504 %hold on
```

```
505 %plot(x_sep_mean, z_sep_mean,'r*','MarkerSize',10)
506
507 hold on
508 xline(x_sep_mean-2*std_dev_x,':','$-2\sigma$',...
509 'LabelHorizontalAlignment','left','FontSize',font,...
510 'interpreter','latex')
511 hold on
512 xline(x_sep_mean-std_dev_x,':','$-\sigma$',...
513 'LabelHorizontalAlignment','left','FontSize',font,...
514 'interpreter','latex')
515 hold on
516 xline(x_sep_mean,':','Mean','interpreter','latex','FontSize',font)
517 hold on
518 xline(x_sep_mean+std_dev_x,':','$+\sigma$','FontSize',font,...
519 'interpreter','latex')
520 hold on
521 xline(x_sep_mean+2*std_dev_x,':','$+2\sigma$',...
522 'LabelHorizontalAlignment','right','FontSize',font,...
523 'interpreter','latex')
524
525 xlabel('x/L','interpreter','latex','FontSize',font)
526 ylabel('z/L','interpreter','latex','FontSize',font)
527
528 L = 36;
529 L_b = 35.5;
530 x_0 = 0.195 * L_b;
531 y_0 = 0.06 * L_b;
532 h_0 = 0.085* L_b;
533 Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
534 (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
535 z_0 = 0;
536 x_wall = linspace(min(min(X)),max(max(X)),200);
537
```

```
538 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
539 z_wall = zeros(1,length(x_wall));
540 z_wall(1:min(bump_ind)-1) = z_0;
541 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
542
543 span = P_all(1).Input.Span/25.4;
544
545 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
546 ones(1,length(bump_ind)).*span*0.0393701);
547 hold on
548 plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',0.5)
549 xlim([min(min(X))./(L*0.0254) max(max(X))./(L*0.0254)])
550
551 set(gca,'FontSize',font)
552 close all
```

Listing B.15: Create Conditionally Averaged PIV Fields from Pressure Standard Deviations and Automatic Determination of Separation and Reattachment Point

```
1  clear
2  filename = uigetfile('Select conditionally averaged file');
3  load(filename)
4  %%
5  font = 18;
6  std_deviation = std(Conditionally_averaged_pressure.pressure_info);
7  mean_pressure=mean(Conditionally_averaged_pressure.pressure_info,'omitnan');
8  tap_cond_avg = input('Which Tap?')
9  % four regions, greater than 1std, mean to 1st, mean to -1st, greater than
10 %-1st
11 %% Region 1: points > than 1 std
12 region1 = Conditionally_averaged_pressure.bins > mean_pressure+...
13 std_deviation;
```

```
k = 1;
for i=1:length(region1)-1
    if region1(i) == 1
        U_region1(:,:,k) = Conditionally_averaged_pressure.U(:,:,i);
        W_region1(:,:,k) = Conditionally_averaged_pressure.W(:,:,i);
        counts_region_1(k) = Conditionally_averaged_pressure.counts(i);
        k = k+1;
    end
end

U_region1_mean = nanmean(U_region1,3);
W_region1_mean = nanmean(W_region1,3);
X = Conditionally_averaged_pressure.X ;
Z = Conditionally_averaged_pressure.Z;

L = 36;
L_b = 35.5;
x_0 = 0.195 * L_b;
y_0 = 0.06 * L_b;
h_0 = 0.085* L_b;
Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
(1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
z_0 = 0;
x_wall = linspace(min(min(X)),max(max(X)),200);

bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
z_wall = zeros(1,length(x_wall));
z_wall(1:min(bump_ind)-1) = z_0;
z_wall(max(bump_ind)+1:length(x_wall)) = z_0;

span = 0;

z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
```

```
ones(1,length(bump_ind)).*span*0.0393701);


figure
plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)

hold on

H = pcolor(X/(L*0.0254),Z/(L*0.0254), U_region1_mean);
colormap(jet), set(H,'edgecolor','none')
shading interp
ax = gca;
hold(ax);
hold on;
x_data = get(H,'xdata');
z_data = get(H,'ydata');
u_data = get(H,'CData');
[M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
c.LineWidth = 3;
c.LineColor = 'w';

xlabel('x/L','interpreter','latex','FontSize',font)
ylabel('z/L','interpreter','latex','FontSize',font)
bar = colorbar;
bar.FontSize = font;
bar.Label.Interpreter = 'latex';
bar.Label.String = '$U/U_\infty$';
bar.TickLabelInterpreter = 'latex';
caxis([-0.3 1.5])
hold(ax);

x_contour_locations = sort(M(1,:));
i=1;
```

```
if isempty(x_contour_locations)
    x_sep(1) = NaN;
else
    while x_contour_locations(i) == 0
        i=i+1;
    end
    x_sep(1) = x_contour_locations(i);
end

L = 36;
L_b = 35.5;
x_0 = 0.195 * L_b;
y_0 = 0.06 * L_b;
h_0 = 0.085* L_b;
Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
(1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
z_0 = 0;
x_wall = linspace(min(min(X)),max(max(X)),200);

bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
z_wall = zeros(1,length(x_wall));
z_wall(1:min(bump_ind)-1) = z_0;
z_wall(max(bump_ind)+1:length(x_wall)) = z_0;

span = 0;

z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
ones(1,length(bump_ind)).*span*0.0393701);
hold on
z_sep(1) = Z_bump(x_sep(1)*(L),span*0.0393701)/(L);
close all
%% Region 2: mean < points < than 1 std
region2 = (mean_pressure < Conditionally_averaged_pressure.bins) &...
```

```
113     (Conditionally_averaged_pressure.bins < (mean_pressure+std_deviation));
114 k = 1;
115 for i=1:length(region2)-1
116     if region2(i) == 1
117         U_region2(:,:,k) = Conditionally_averaged_pressure.U(:,:,i);
118         W_region2(:,:,k) = Conditionally_averaged_pressure.W(:,:,i);
119         counts_region_2(k) = Conditionally_averaged_pressure.counts(i);
120         k = k+1;
121     end
122 end
123
124 U_region2_mean = nanmean(U_region2,3);
125 W_region2_mean = nanmean(W_region2,3);
126 X = Conditionally_averaged_pressure.X ;
127 Z = Conditionally_averaged_pressure.Z;
128
129 L = 36;
130 L_b = 35.5;
131 x_0 = 0.195 * L_b;
132 y_0 = 0.06 * L_b;
133 h_0 = 0.085* L_b;
134 Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
135 (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
136 z_0 = 0;
137 x_wall = linspace(min(min(X)),max(max(X)),200);
138
139 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
140 z_wall = zeros(1,length(x_wall));
141 z_wall(1:min(bump_ind)-1) = z_0;
142 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
143
144 span = 0;
145
```

```
146 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
147 ones(1,length(bump_ind)).*span*0.0393701);
148
149 figure
150 plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
151
152 hold on
153
154 H = pcolor(X/(L*0.0254),Z/(L*0.0254), U_region2_mean);
155 colormap(jet), set(H,'edgecolor','none')
156 shading interp
157 ax = gca;
158 hold(ax);
159 hold on;
160 x_data = get(H,'xdata');
161 z_data = get(H,'ydata');
162 u_data = get(H,'CData');
163 [M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
164 c.LineWidth = 3;
165 c.LineColor = 'w';
166
167 xlabel('x/L','interpreter','latex','FontSize',font)
168 ylabel('z/L','interpreter','latex','FontSize',font)
169 bar = colorbar;
170 bar.FontSize = font;
171 bar.Label.Interpreter = 'latex';
172 bar.Label.String = '$U/U_\infty$';
173 bar.TickLabelInterpreter = 'latex';
174 caxis([-0.3 1.5])
175 hold(ax);
176
177 x_contour_locations = sort(M(1,:));
178 i=1;
```

```
179 if isempty(x_contour_locations)
180     x_sep(2) = NaN;
181 else
182     while x_contour_locations(i) == 0
183         i=i+1;
184     end
185     x_sep(2) = x_contour_locations(i);
186 end
187 
188 L = 36;
189 L_b = 35.5;
190 x_0 = 0.195 * L_b;
191 y_0 = 0.06 * L_b;
192 h_0 = 0.085* L_b;
193 Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
194 (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
195 z_0 = 0;
196 x_wall = linspace(min(min(X)),max(max(X)),200);
197 
198 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
199 z_wall = zeros(1,length(x_wall));
200 z_wall(1:min(bump_ind)-1) = z_0;
201 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
202 
203 span = 0;
204 
205 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
206 ones(1,length(bump_ind)).*span*0.0393701);
207 hold on
208 z_sep(2) = Z_bump(x_sep(2)*(L),span*0.0393701)/(L);
209 
210 close all
211 %% Region 3: -1 std < points < mean
```

```
region3 = (mean_pressure-std_deviation) < Conditionally_averaged_pressure...
bins &  Conditionally_averaged_pressure.bins <...
mean_pressure;
k = 1;
for i=1:length(region3)-1
    if region3(i) == 1
        U_region3(:,:,k) = Conditionally_averaged_pressure.U(:,:,i);
        W_region3(:,:,k) = Conditionally_averaged_pressure.W(:,:,i);
        counts_region_3(k) = Conditionally_averaged_pressure.counts(i);
        k = k+1;

    end
end

U_region3_mean = nanmean(U_region3,3);
W_region3_mean = nanmean(W_region3,3);
X = Conditionally_averaged_pressure.X ;
Z = Conditionally_averaged_pressure.Z;

L = 36;
L_b = 35.5;
x_0 = 0.195 * L_b;
y_0 = 0.06 * L_b;
h_0 = 0.085* L_b;
Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
(1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
z_0 = 0;
x_wall = linspace(min(min(X)),max(max(X)),200);

bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
z_wall = zeros(1,length(x_wall));
z_wall(1:min(bump_ind)-1) = z_0;
z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
```

```
245
246 span = 0;
247
248 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
249 ones(1,length(bump_ind)).*span*0.0393701);
250
251 figure
252 plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
253
254 hold on
255
256 H = pcolor(X/(L*0.0254),Z/(L*0.0254), U_region3_mean);
257 colormap(jet), set(H,'edgecolor','none')
258 shading interp
259 ax = gca;
260 hold(ax);
261 hold on;
262 x_data = get(H,'xdata');
263 z_data = get(H,'ydata');
264 u_data = get(H,'CData');
265 [M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
266 c.LineWidth = 3;
267 c.LineColor = 'w';
268
269 xlabel('x/L','interpreter','latex','FontSize',font)
270 ylabel('z/L','interpreter','latex','FontSize',font)
271 bar = colorbar;
272 bar.FontSize = font;
273 bar.Label.Interpreter = 'latex';
274 bar.Label.String = '$U/U_\infty$';
275 bar.TickLabelInterpreter = 'latex';
276 caxis([-0.3 1.5])
277 hold(ax);
```

```
278
279 x_contour_locations = sort(M(1,:));
280 i=1;
281 if isempty(x_contour_locations)
282     x_sep(3) = NaN;
283 else
284     while x_contour_locations(i) == 0
285         i=i+1;
286     end
287     x_sep(3) = x_contour_locations(i);
288 end
289
290 L = 36;
291 L_b = 35.5;
292 x_0 = 0.195 * L_b;
293 y_0 = 0.06 * L_b;
294 h_0 = 0.085* L_b;
295 Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
296 (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
297 z_0 = 0;
298 x_wall = linspace(min(min(X)),max(max(X)),200);
299
300 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
301 z_wall = zeros(1,length(x_wall));
302 z_wall(1:min(bump_ind)-1) = z_0;
303 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
304
305 span = 0;
306
307 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
308 ones(1,length(bump_ind)).*span*0.0393701);
309 hold on
310 z_sep(3) = Z_bump(x_sep(3)*(L),span*0.0393701)/(L);
```

```
close all
%% Region 4: points < -1 std
region4 = Conditionally_averaged_pressure.bins < mean_pressure-...
std_deviation;
k = 1;
for i=1:length(region4)-1
    if region4(i) == 1
        U_region4(:,:,k) = Conditionally_averaged_pressure.U(:,:,i);
        W_region4(:,:,k) = Conditionally_averaged_pressure.W(:,:,i);
        counts_region_4(k) = Conditionally_averaged_pressure.counts(i);
        k = k+1;
    end
end

U_region4_mean = nanmean(U_region4,3);
W_region4_mean = nanmean(W_region4,3);
X = Conditionally_averaged_pressure.X;
Z = Conditionally_averaged_pressure.Z;

L = 36;
L_b = 35.5;
x_0 = 0.195 * L_b;
y_0 = 0.06 * L_b;
h_0 = 0.085* L_b;
Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
(1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
z_0 = 0;
x_wall = linspace(min(min(X)),max(max(X)),200);

bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
z_wall = zeros(1,length(x_wall));
z_wall(1:min(bump_ind)-1) = z_0;
z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
```

```
344
345 span = 0;
346
347 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
348 ones(1,length(bump_ind)).*span*0.0393701);
349
350 figure
351 plot(x_wall./(L*0.0254),z_wall./L,'k','LineWidth',3)
352
353 hold on
354
355 H = pcolor(X/(L*0.0254),Z/(L*0.0254), U_region4_mean);
356 colormap(jet), set(H,'edgecolor','none')
357 shading interp
358 ax = gca;
359 hold(ax);
360 hold on;
361 x_data = get(H,'xdata');
362 z_data = get(H,'ydata');
363 u_data = get(H,'CData');
364 [M,c] = contour(x_data,z_data,u_data,[0,0],'ShowText','on');
365 c.LineWidth = 3;
366 c.LineColor = 'w';
367
368 xlabel('x/L','interpreter','latex','FontSize',font)
369 ylabel('z/L','interpreter','latex','FontSize',font)
370 bar = colorbar;
371 bar.FontSize = font;
372 bar.Label.Interpreter = 'latex';
373 bar.Label.String = '$U/U_\infty$';
374 bar.TickLabelInterpreter = 'latex';
375 caxis([-0.3 1.5])
376 hold(ax);
```

```
377
378 x_contour_locations = sort(M(1,:));
379 i=1;
380 if isempty(x_contour_locations)
381     x_sep(4) = NaN;
382 else
383     while x_contour_locations(i) == 0
384         i=i+1;
385     end
386     x_sep(4) = x_contour_locations(i);
387 end
388
389 L = 36;
390 L_b = 35.5;
391 x_0 = 0.195 * L_b;
392 y_0 = 0.06 * L_b;
393 h_0 = 0.085* L_b;
394 Z_bump = @(x_bump,y_bump) (h_0/2) .* exp(-(x_bump./x_0).^2) .* ...
395 (1 + erf((L_b/2 - 2*y_0 - abs(y_bump))./y_0));
396 z_0 = 0;
397 x_wall = linspace(min(min(X)),max(max(X)),200);
398
399 bump_ind = find(x_wall>-(L_b*0.0254)/2 & x_wall<(L_b*0.0254)/2);
400 z_wall = zeros(1,length(x_wall));
401 z_wall(1:min(bump_ind)-1) = z_0;
402 z_wall(max(bump_ind)+1:length(x_wall)) = z_0;
403
404 span = 0;
405
406 z_wall(bump_ind) = Z_bump(x_wall(1,bump_ind).*39.3701,...
407 ones(1,length(bump_ind)).*span*0.0393701);
408 hold on
409 z_sep(4) = Z_bump(x_sep(4)*(L),span*0.0393701)/(L);
```

```
410
411
412 close all
413 %% pdf of separation points
414 x_sep_hist = [ones(sum(counts_region_1),1)*x_sep(1); ...
415 ones(sum(counts_region_2),1)*...
416     x_sep(2); ones(sum(counts_region_3),1)...
417     *x_sep(3); ones(sum(counts_region_4),1)...
418     *x_sep(4)];
419
420 z_sep_hist = [ones(sum(counts_region_1),1)*z_sep(1);...
421 ones(sum(counts_region_2),1)*...
422     z_sep(2); ones(sum(counts_region_3),1)*...
423     z_sep(3); ones(sum(counts_region_4),1)...
424     *z_sep(4)];
425
426 yyaxis left
427 histogram(x_sep_hist,'normalization','pdf')
428 xlabel('x/L','interpreter','latex')
429 ylabel('p.d.f','interpreter','latex')
430
431 yyaxis right
432 histogram(x_sep_hist,'normalization','cdf','DisplayStyle','Stairs')
433 ylabel('CDF','interpreter','latex')
```

Listing B.16: Create p.d.f.s and CDFs of reverse flow fraction

```
1 clear
2 filename = uigetfile('select conditionally averaged field?');
3 load(filename)
4
5 %%
6 for i = 2
```

```
7       hold on
8       yyaxis left
9       histogram(Conditionally_averaged(i).negative_area,...
10      'Normalization','pdf'...
11          ,'DisplayStyle','stairs')
12      xlabel('$\alpha$','interpreter','latex')
13      ylabel('p.d.f')
14
15      hold on
16      yyaxis right
17      histogram(Conditionally_averaged(i).negative_area,...
18      'Normalization','cdf'...
19          ,'DisplayStyle','stairs')
20      xlabel('$\alpha$','interpreter','latex')
21      ylabel('CDF')
22
23      xlim([0 0.35])
24      set(gca,'FontSize',16)
25      set(gca,'box','off')
26
27      hold on
28      legend(['pdf FOV ' num2str(i-1)],['CDF FOV ' num2str(i-1)])
29  end
30
31  set(gca,'FontSize',16)
32  %% Statistics
33  mean_negative_area = mean(Conditionally_averaged(2).negative_area)
34  median_negative_area = median(Conditionally_averaged(2).negative_area)
35  std_negative_area = std(Conditionally_averaged(2).negative_area)
36  max(Conditionally_averaged(2).negative_area)
```

Appendix C

C.1 Ancillary Modes Produced by Proper Orthogonal Decomposition

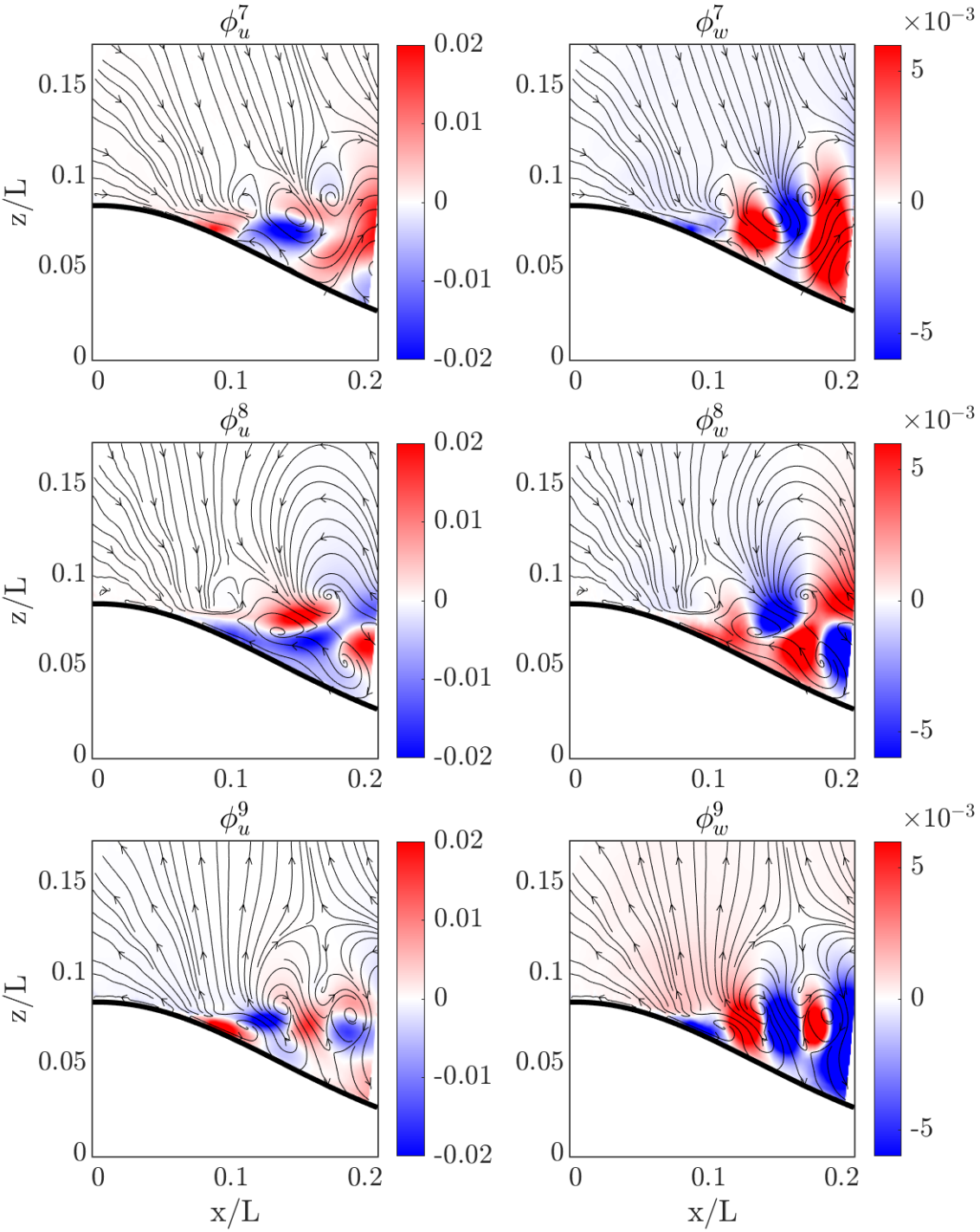


Figure C.1: Composite figures of modes ϕ^7 to ϕ^9 of the streamwise, u, and wall-normal, w, velocity fluctuations.

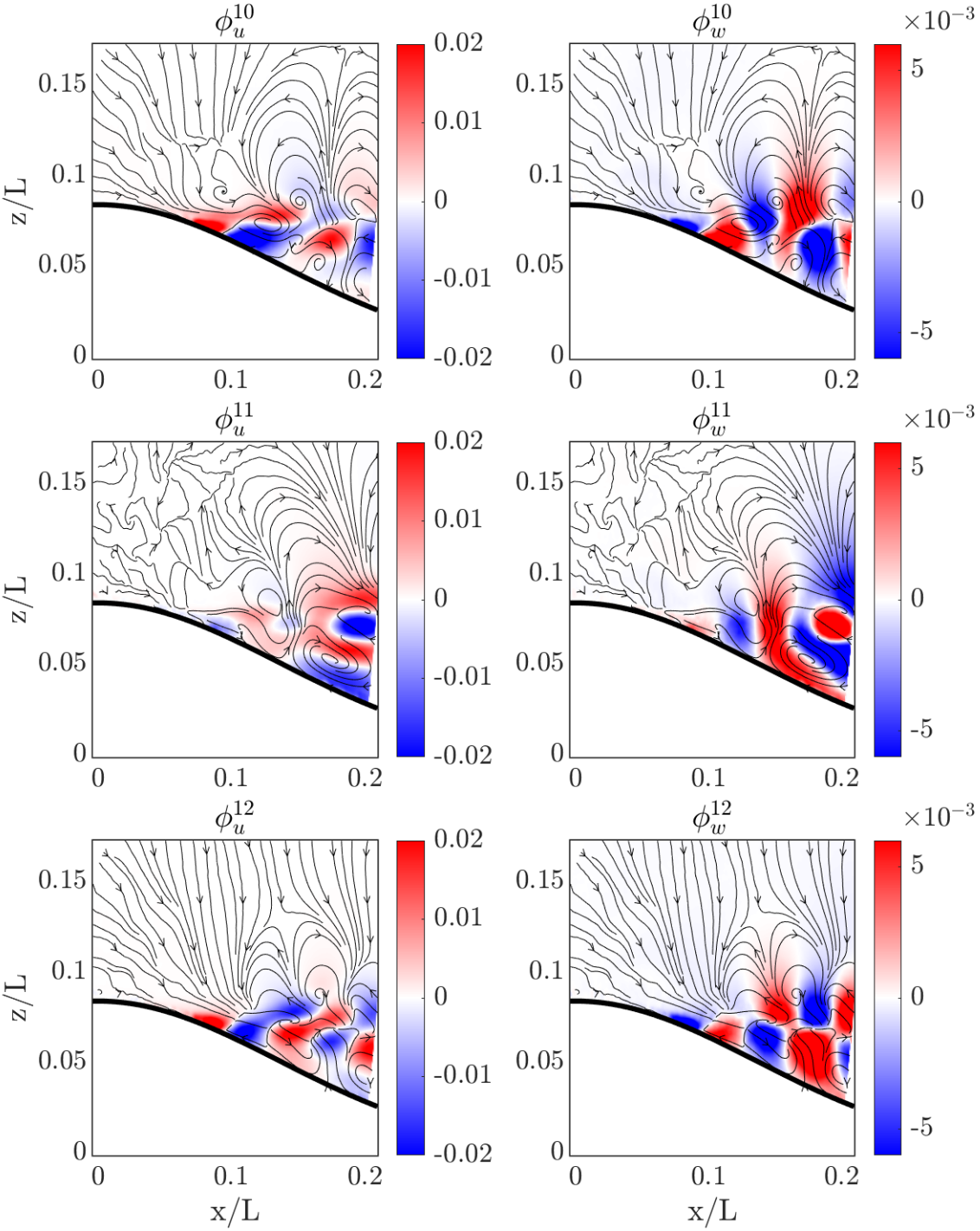


Figure C.2: Composite figures of modes ϕ^{10} to ϕ^{12} of the streamwise, u, and wall-normal, w, velocity fluctuations.

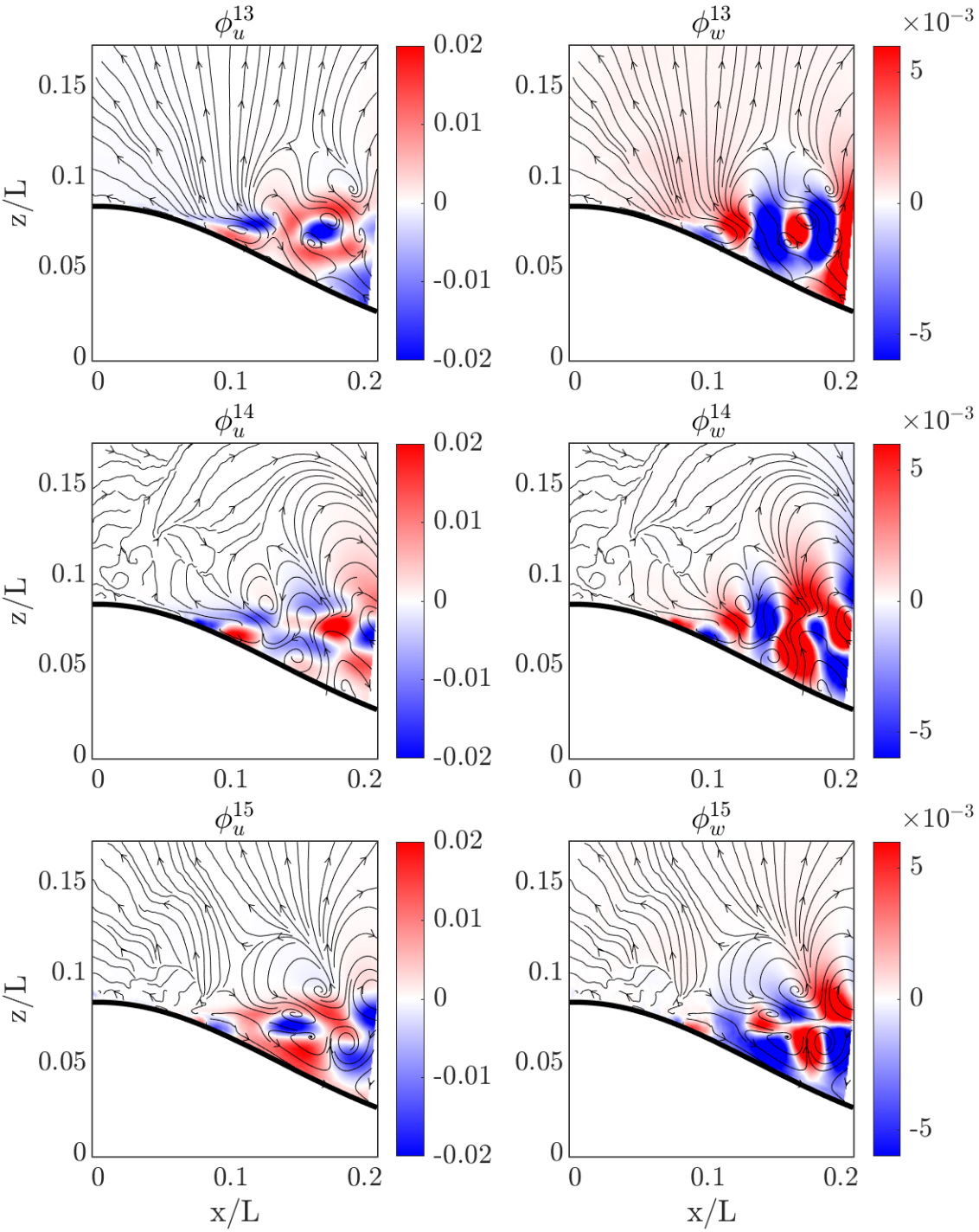


Figure C.3: Composite figures of modes ϕ^{13} to ϕ^{15} of the streamwise, u, and wall-normal, w, velocity fluctuations.

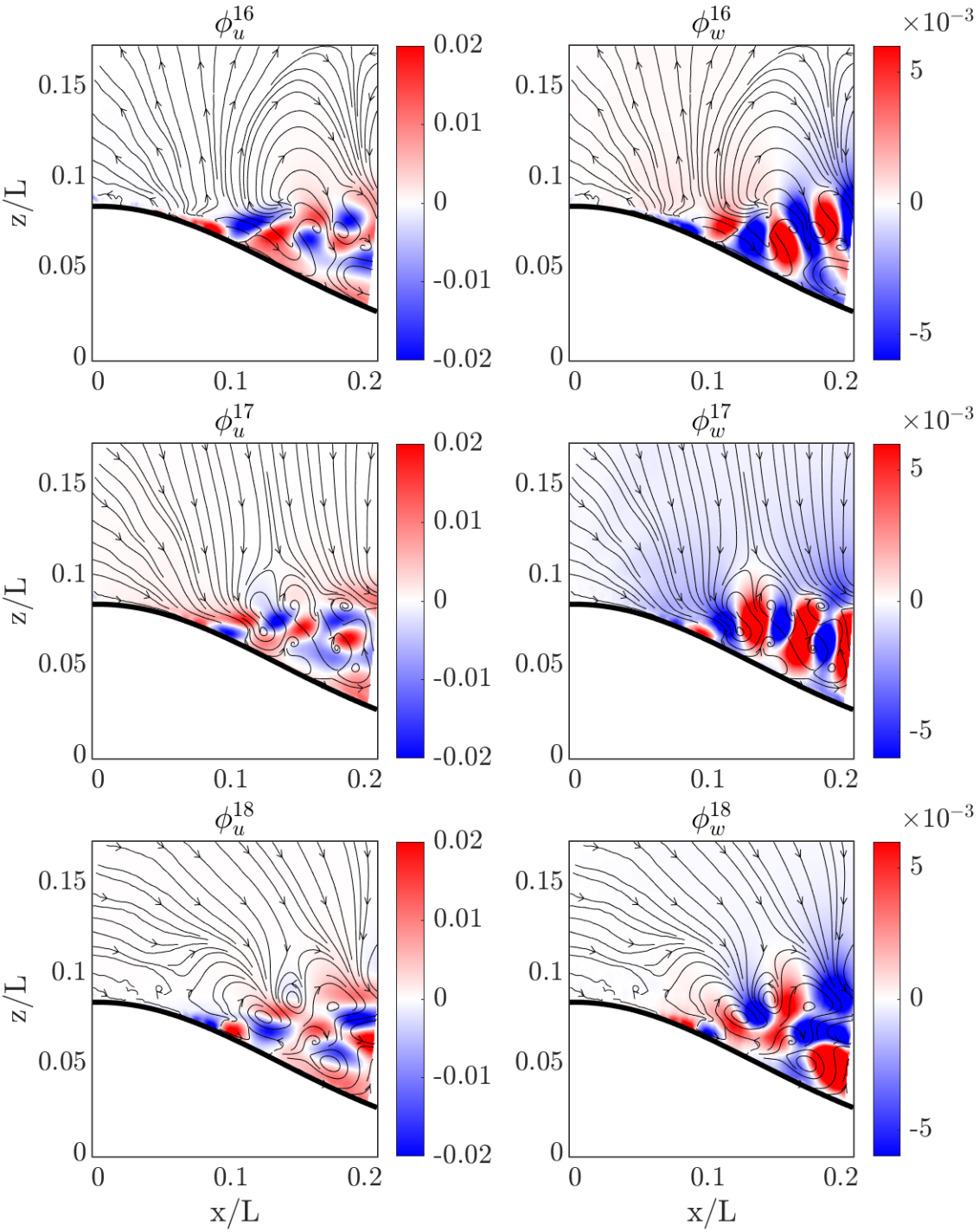


Figure C.4: Composite figures of modes ϕ^{16} to ϕ^{18} of the streamwise, u, and wall-normal, w, velocity fluctuations.

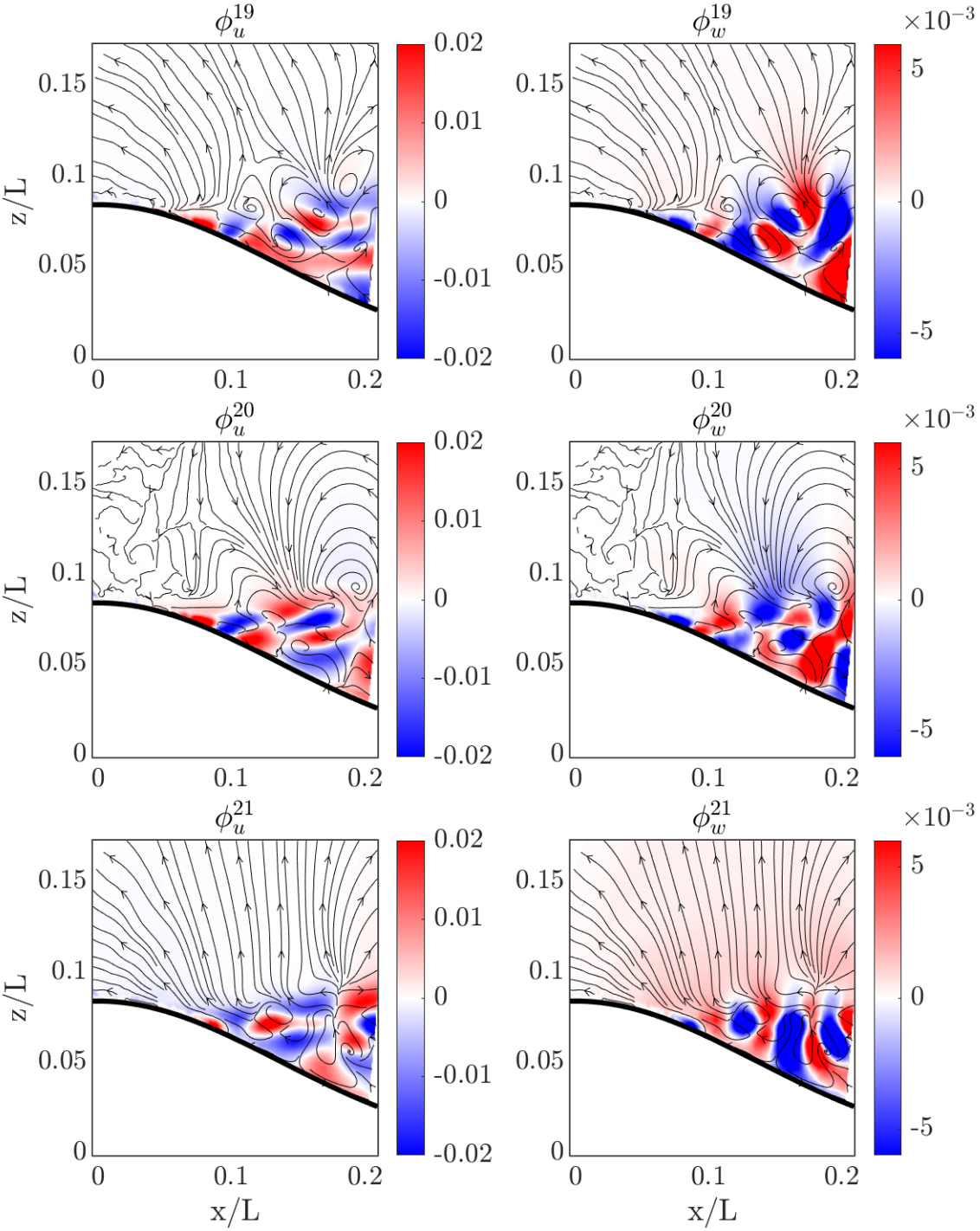


Figure C.5: Composite figures of modes ϕ^{19} to ϕ^{21} of the streamwise, u, and wall-normal, w, velocity fluctuations.

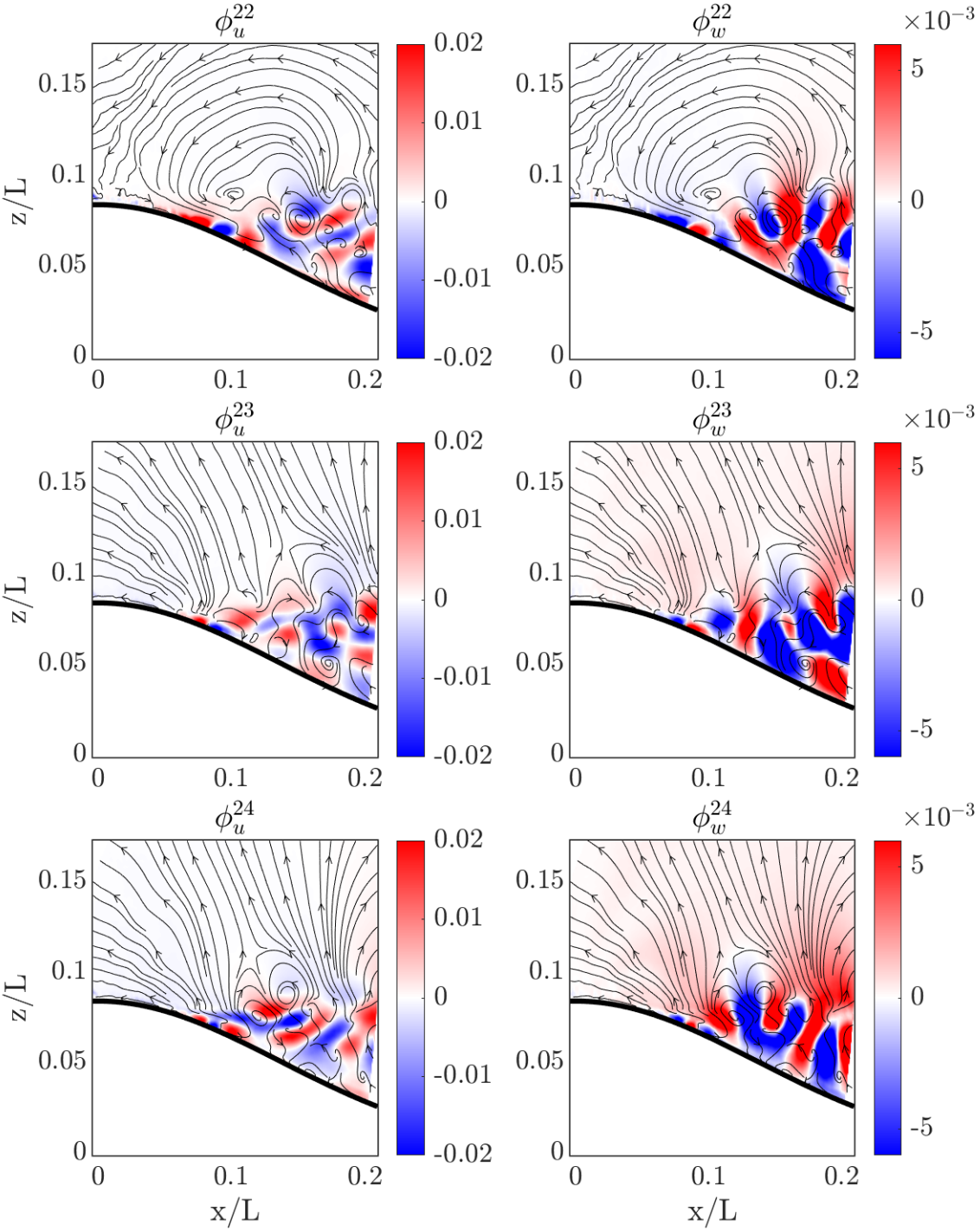


Figure C.6: Composite figures of modes ϕ^{22} to ϕ^{24} of the streamwise, u, and wall-normal, w, velocity fluctuations.

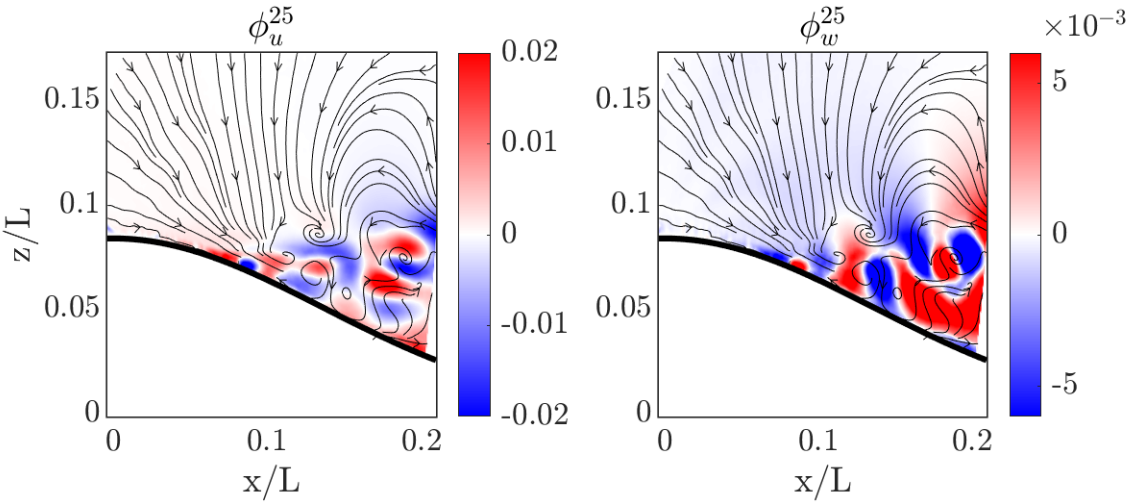


Figure C.7: Composite figures of mode ϕ^{25} of the streamwise, u, and wall-normal, w, velocity fluctuations.

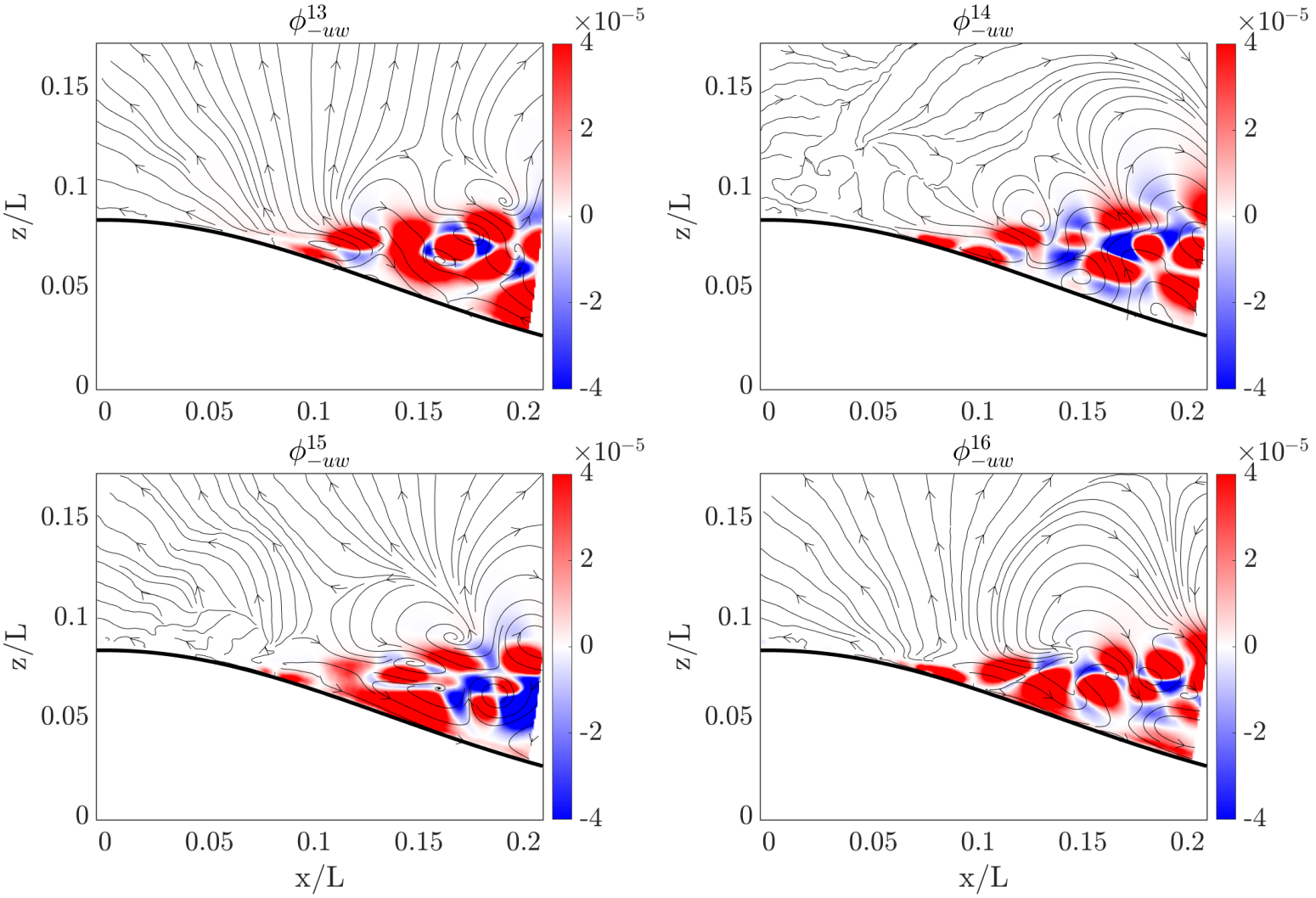


Figure C.8: Shear stress modes ϕ^{13}_{-uw} to ϕ^{16}_{-uw} focused on the separated region.

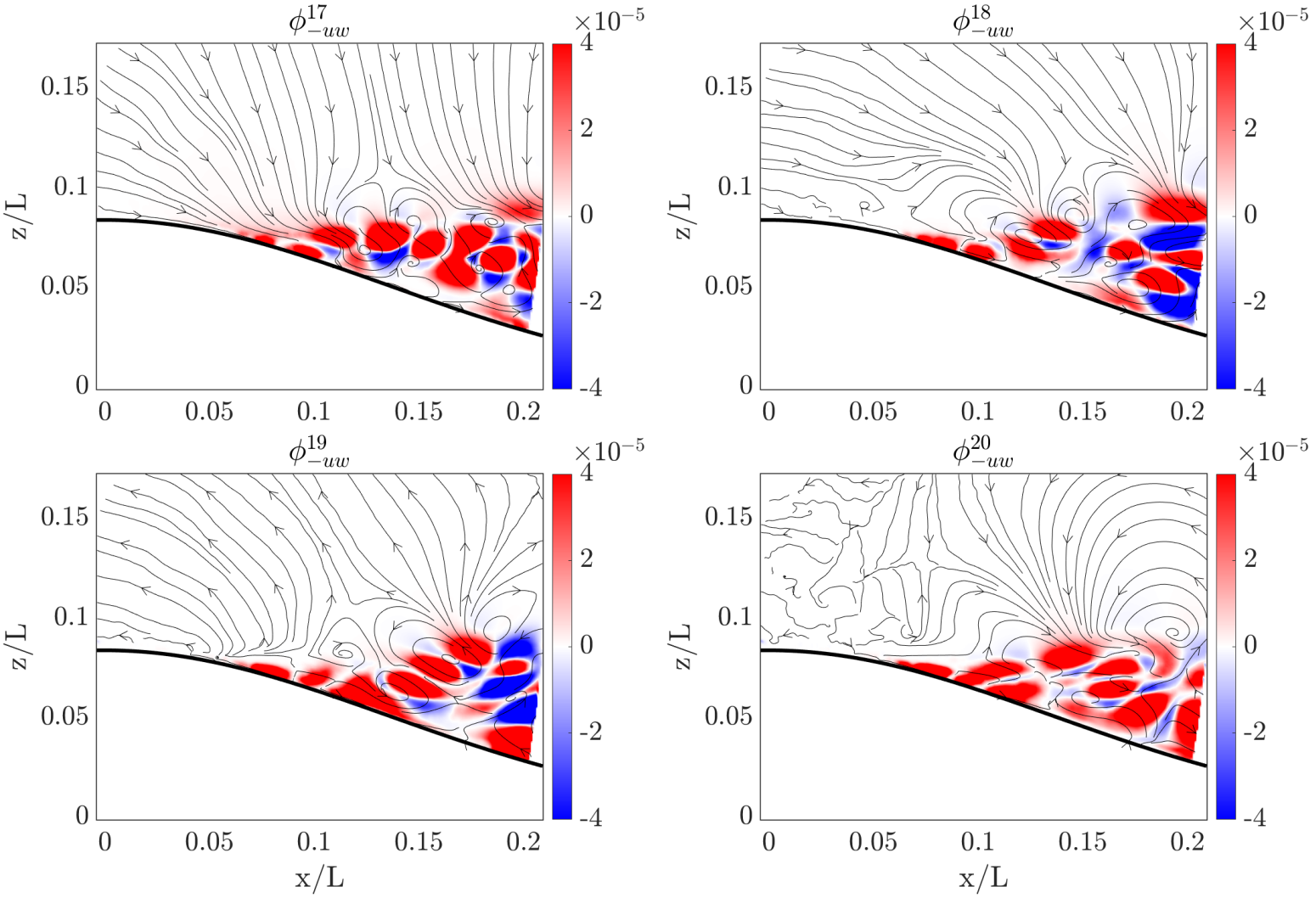


Figure C.9: Shear stress modes ϕ^{17}_{-uw} to ϕ^{20}_{-uw} focused on the separated region.

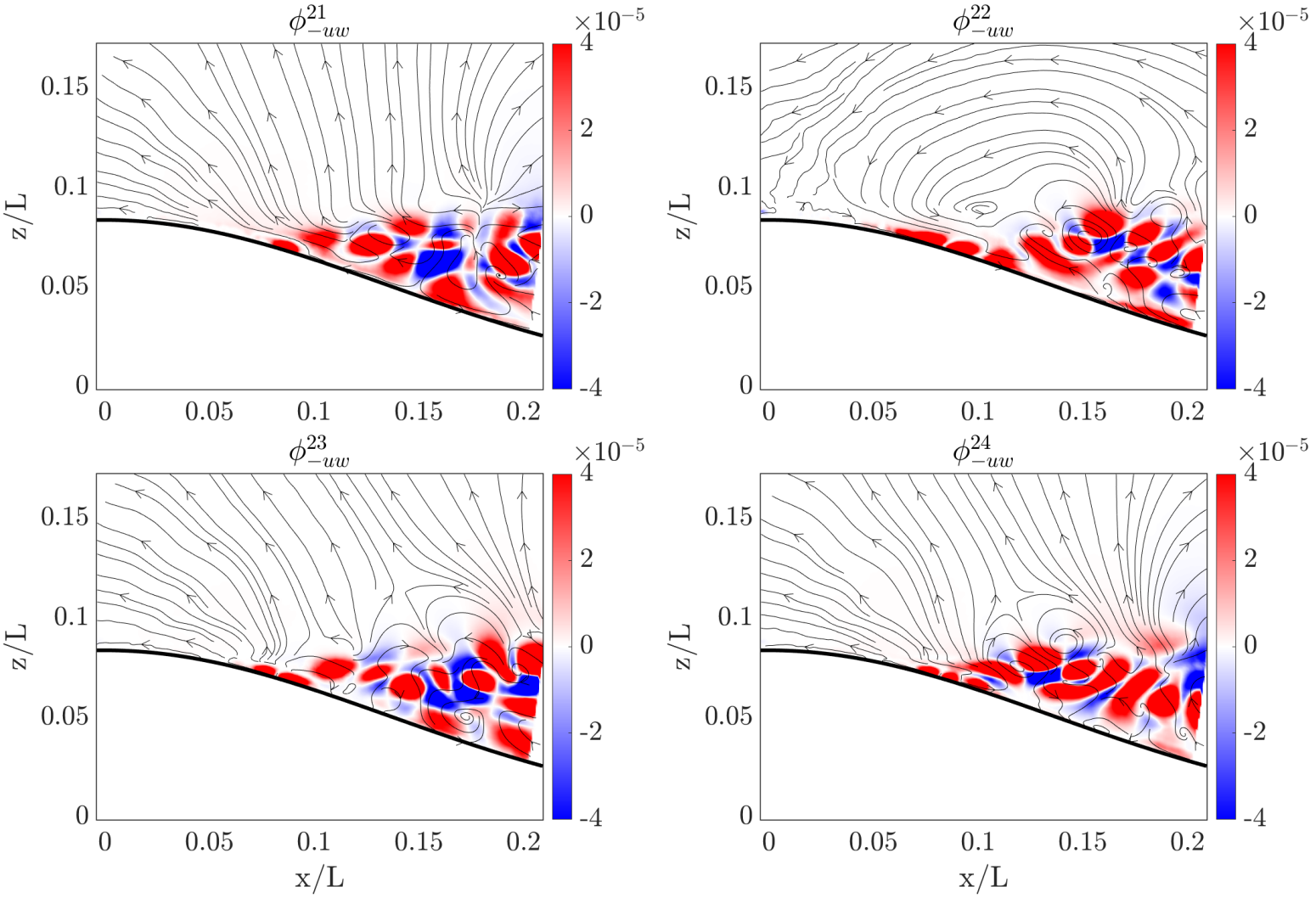


Figure C.10: Shear stress modes ϕ^{21}_{-uw} to ϕ^{24}_{-uw} focused on the separated region.

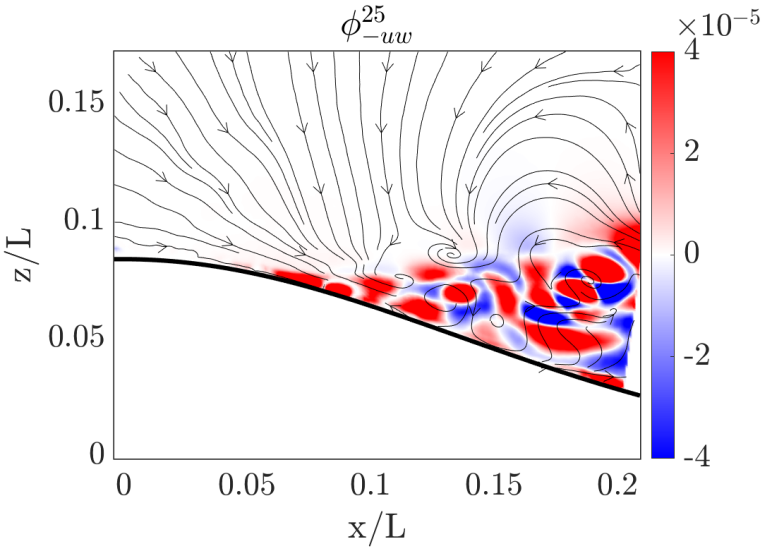


Figure C.11: Shear stress mode ϕ^{25}_{-uw} focused on the separated region.