Turbulent Boundary Layer Control with Discrete Actuators Using Wall Information

by

Gerardo Ernesto Colmenero, B.S.

Thesis

Presented to the Faculty of the Graduate School

of The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

The University of Texas at Austin

December 2004

Turbulent Boundary Layer Control with Discrete Actuators Using Wall Information

Approved by

Supervising Committee:

_____

_____

To my family.

Acknowledgments

I would like to thank my supervisor, Dr. David B. Goldstein, for his support and guidance throughout the course of my studies and Dr. Laxminarayan Raja for reviewing this work and providing feed back. I gratefully acknowledge the funding provided by the Air Force Office of Scientific Research, under grant number F49620-02-1-0093 and the computation time provided by the Texas Advanced Computing Center, TACC, in support of this project. Helpful conversions with Dr. Conrad Lee are also gratefully acknowledged.

GERARDO E. COLMENERO

The University of Texas at Austin

December 2004

Turbulent Boundary Layer Control with Discrete Actuators Using Wall Information

Gerardo Ernesto Colmenero, M.S.E.

The University of Texas at Austin, 2004

Supervisor: David B. Goldstein

Flow control may be achieved by using Micro Electro-Mechanical Systems (MEMS) to alter the fine scale flow structures within a boundary layer. In this study, a direct numerical simulation of slot jet MEMS in a turbulent channel flow is used to investigate flow control. This report consists of the results of an array of discrete wall-normal actuators coupled with a control algorithm that uses wall information upstream of the jets to sense oncoming streamwise vortices. These vortices are a key phenomenon in the development and regeneration of turbulent flow and are therefore targeted to provide drag reduction over a channel wall. Results show that detection of a spanwise shear gradient at the channel surface provides an effective detection scheme as a drag reduction of 3.0% ±2.1% is achieved using an actuation strength such that in the slot exit plane $v_{rms} \approx 0.28u^*$. In addition, the optimization of a second detection algorithm measuring streamwise shear gradients near the wall is outlined and examined. The goal is to detect high- and low-streamwise shear stress

regions just upstream of a micro-actuator in order to manipulate the near-wall flow. Preliminary results suggest that this second control algorithm is not as effective as the first. The investigation finds that with optimal time delay and actuator signal gain there is an *increase* of the average drag. Preliminary results show a 3.2% ±2.3% drag *increase* suggesting that either the optimized settings have not yet been determined or that this algorithm may not be as effective as the first method. This result is achieved with an exit velocity at the slot exit plane having $v_{rms} \approx 0.07u^*$. For both control algorithms the total control area of the current array of actuators is equivalent to approximately 15% of the total wetted surface suggesting a larger reduction, using the first algorithm, may be achieved by the addition of more actuators.

Contents

List of Tables

List of Figures

Nomenclature

General

$C_{deflection}$	membrane deflection scaling factor
dt	time step
h	channel half-height with virtual surface
l^*	wall units or viscous length scale, ν/u^*
L	channel length
ΔL	streamwise resolution
P	pressure
$Pamp$	gain constant of the actuator response
$Re_{channel}$	centerline Reynolds number, $u_{cl}h/\nu$
R^*	turbulent Reynolds number, u^*h/ν
t^*	viscous time scale, l^*/u^*
T	total flow duration
Δt	time delay constant, number of iterations
u^*	friction velocity, $(\tau_w/\rho)^{1/2}$

u_{cl}	channel centerline mean velocity
u, v, w	velocity in streamwise, normal and spanwise directions
W	channel width
ΔW	spanwise resolution
x, y, z	streamwise, normal and spanwise directions
x^+, y^+, z^+	streamwise, normal and spanwise directions in wall units, x/l^*

Greek Symbols

μ	absolute viscosity
v	kinematic viscosity
ρ	density
τ_w	wall shear stress on unmanipulated wall, $\Delta u/\Delta y$
$\omega_x, \omega_y, \omega_z$	streamwise, normal and spanwise vorticity components

Chapter 1

Introduction

A turbulent boundary layer is characterized by coherent vortical structures that arise, evolve and decay in a quasi-periodic fashion. The structures, which are dominant in the near-wall region, occupy only 25% of this region but are responsible for approximately 50% of the total turbulence production [1]. Hence, the goal of many researchers and the present research is to actively weaken the coherent structures in the near-wall region to achieve drag reductions over a channel wall. There exist three regions in boundary layer flow that can be categorized by how certain terms in the turbulent energy equation compare, specifically how the rate of turbulent energy production compares to its dissipation. The outer region of the boundary layer can be characterized as a place where turbulent energy dissipation is greater than the rate of production [2]. In this region turbulent shear, associated with eddy viscosity and turbulence dissipation, is dominant over the molecular viscosity effects, which are associated with viscous shear and the no-slip boundary condition of the wall [3]. In

the logarithmic region of a boundary layer, the energy dissipation and the rate of energy production of are of equal magnitude and in general nothing happens in this region except the passing of eddy structures from one region to another [2]. The third region, closest to the wall where $y^+ < 100$, consists of the viscous sub-layer below the inner portion of the log layer. In this buffer layer the rate of production is dominant over dissipation. It is the manipulation of this inner region that interests researchers since it provides the greatest potential for reduction of turbulent energy production. Furthermore, reducing the rate of production also reduces the passage of energy away from the wall which helps to maintain turbulence. For wall-bounded flows, it is suggested [2] that a quasi-periodic turbulence cycle exists in the near-wall region ($20 < y^+ < 60$) and it is independent of flow in the interior of the channel. Therefore, understanding of the physics of the regeneration cycle in this region is important and this cycle is briefly described.

Jimenez *et al.* [2] suggests that the near-wall region is dominated by streamwise velocity streaks superimposed on the mean shear, where the mean shear is maintained by the no-slip boundary condition of the wall. It is well known that streaks can be very long, $x^+ \approx 1000$, have a width of 20 - 40l^*, and have an average spanwise spacing of $z^+ \approx 100$ [4]. In addition to streaks, quasi-streamwise vortices dominate the near-wall region; however, because the vortices are not aligned exactly parallel to the wall they only remain in this region for approximately $x^+ \approx 200$ [2]. Due to the shorter length of the quasi-streamwise vortices it seems that several

vortices can be associated with each velocity streak. This suggests that the physical vortex/streak interaction is crucial for the life of both structures; and in general, it suggests there are more vortices in the near-wall region than there are velocity streaks. The explanation of how the velocity streaks are created is roughly as follows; as the pairs of quasi-streamwise vortices travel downstream, they tend to pump high momentum fluid from the center of the channel (sweep events) towards the wall where low momentum fluid exists due to the no-slip condition of the channel boundaries. In addition the quasi-streamwise vortices pump low momentum fluid from the near-wall region towards the center of the channel creating an ejection event of low momentum fluid. The sweep and ejection events result in the alternating streaks of streamwise velocity. Furthermore, the generation of the quasi-streamwise vortices results from inflectional instabilities introduced by the streaks which affect the velocity profiles [2] thereby generating a self-sustaining mechanism for creating vortices.

The mixing of the high and low momentum fluid in the channel, resulting primarily from the sweep and ejection events, helps to create a fuller mean velocity profile characteristic of a turbulent channel flow. The steeper velocity gradient near the channel wall results in a much higher viscous drag than would laminar flow at the same bulk channel velocity [5]. Therefore it is argued that the weakening of the streamwise vortices in the region $20 < y^+ < 60$ may achieve reductions in skin-friction drag. This effect has been observed with passive devices such as riblets. In

the case of passive devices, the streamwise vortices are weakened by increasing the spanwise friction above the riblet crests [6] and by the restriction of the vortices such that only a limited area of the riblet is exposed to the sweep event that the vortices induce [7]. However, the objective of this project is the active control of the near-wall region to accomplish drag reduction. In contrast to passive devices, active control weakens the coherent vortical structures through more direct mechanisms.

Active control of the turbulent boundary layer has been applied using numerous techniques. The overall objective is the use of small sensors and actuators to provide effective control based on measurable flow quantities. The progression of research in this area started with physical, intuitive arguments of the boundary layer features but then shifted towards the use of parametric approaches with the end goal of weakening of the near-wall coherent vortical structures and thereby reducing drag. A brief summary follows describing the numerical and experimental work that has led to the development of the current simulation of turbulent boundary layer control.

Chapter 2

Literature Review

2.1 Progress in turbulent boundary layer active control

2.1.1 Numerical research

Choi *et al.* [8] uses a physical intuition in his approach to create a numerical opposition control method. In this method, depicted in figure 2.1, the vertical motion of the near-wall turbulent flow, which is thought of as resulting from the quasi-streamwise vorticity, is sensed at $y^+ \approx 15$ and countered by an equal but opposite blowing/suction distribution velocity on the wall. The technique results in a 25% drag reduction. However, one major drawback of the method is that it requires knowledge of the flow variables within the flow domain. Also, the opposition control is instantaneously applied at every point along the wall throughout the entire channel domain. These drawbacks make the method impractical to implement physically. The work, however, sparked others to develop more practical methods for detection and actuation.

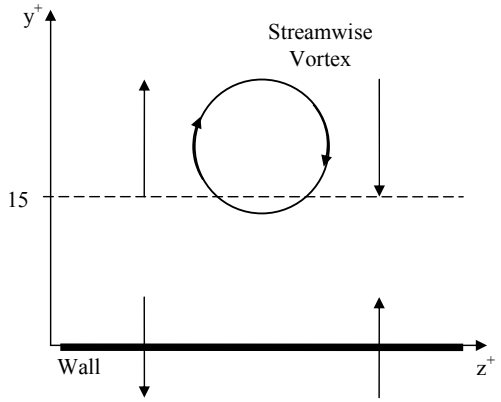


Figure 2.1: Choi *et al.* [8] opposition control scheme to weaken streamwise vortices in the near-wall region of turbulent channel flow. Vertical motion induced by the streamwise vortex is sensed, say, at $y^+ \approx 15$. An opposing blowing/suction velocity distribution is applied on the channel wall.

Koumoutsakos *et al.* [9] sought a numerical feedback control algorithm that uses flow information detected at the wall. The actuating mechanism of this approach is effectively the blowing/suction velocity distribution at the wall, similar to the method of Choi *et al.* [8]. In order to maintain a reasonable level of actuation, actuation strength is limited to a threshold value of 5 - 15% of the mean bulk velocity. The control scheme is based on the manipulation of the spanwise and streamwise vorticity flux components obtained by measuring the instantaneous pressure at the wall and calculating its gradient. The equations relating the two quantities, shown below, were derived from the momentum equation evaluated at the wall.

$$\upsilon\left(\frac{\partial \omega_x}{\partial y}\right)_{wall} = \frac{1}{\rho}\left(\frac{\partial P}{\partial z}\right)_{wall}, \quad -\upsilon\left(\frac{\partial \omega_z}{\partial y}\right)_{wall} = \frac{1}{\rho}\left(\frac{\partial P}{\partial x}\right)_{wall}, \tag{1}$$

where P is the pressure and ω_x and ω_z are the streamwise and spanwise vorticity components. Results using this approach show up to 40% drag reduction for low Reynolds number turbulent channel flow.

Kang *et al.* [7] and Carlson *et al.* [10] sought a different actuation mechanism for suppressing the strength of the near-wall streamwise vortices. In Carlson *et al.* [10] a Gaussian shaped bump actuator approximately $12l^*$ high is installed on the wall. Carlson *et al.* [8] show that raising the actuator below a high-speed streak decreased skin-friction drag by allowing the associated low-speed fluid region to expand. In comparison, the approach of Kang *et al.* [7] reduces skin-friction drag by locally deforming the wall with basic coordinate transformations, thereby inducing a blowing/suction velocity distribution based on two previously imposed feedback control strategies [8, 11]. Results show that for a channel flow at $R^* = 140$, 13 - 17% drag reduction is obtained using the active wall motions. This reduction is smaller than the reductions of Choi *et al.* [8] and Lee *et al.* [11] possibly due to the limitation of the height of the wall deformations in the study [7]. In addition, the results show that active control using wall deformations shifts the mean velocity away from the wall within the logarithmic region and reduces the turbulence intensities; this upward shift appears common in turbulent flows with skin-friction drag reduction [7]. An interesting observation is that the root-mean-squared amplitudes of the wall deformations are approximately $3.2l^*$ and resemble riblets because the wall

deformations extend in the streamwise direction. However, Kang *et al.* [7] found that the mechanism resulting in a reduction of skin-friction drag is due to the direct suppression of the coherent near-wall structures through induced blowing/suction at the wall and not by the passive mechanisms characterized by riblets.

Endo *et al.* [12] developed a numerical feedback control method with an array of sinusoidal deformable wall actuators roughly $172l^*$ by $60l^*$ to minimize the near-wall coherent structures. When using the opposition control scheme similar to Choi *et al.* [8] to determine the local wall velocity, the drag is decreased by ~12% with a wall deformation magnitude on the order of $1l^*$. In addition, a second scheme was developed using only wall information. Endo *et al.* [12] describes the spanwise meandering of the near-wall streamwise streaks as playing an important role in the quasi-cyclic turbulence regeneration process, and he uses the meandering of the streaks to argue that quasi-streamwise vortices accompanied with the meandering streaks can be detected by measuring the streamwise and spanwise gradient of the wall shear stresses. They conclude that by using only wall information they can actuate on quasi-streamwise vortices $50l^*$ downstream from the wall sensors. Through the active wall deformation of arrayed sensors and actuators using only wall information a 10% drag reduction can be achieved.

In addition to the above applications, research on the control of turbulent boundary layers has made use of control theory to examine the control algorithms. Lee *et al.* [11] developed a sub-optimal feedback control law that requires pressure

or shear stress information only at the wall. Using the blowing/suction actuation of Choi *et al.* [8], the numerical method was applied using as the detection variables the local gradients of pressure and shear stress in a turbulent channel flow at $R^* = 110$ resulting in a 16 - 22% reduction in skin-friction drag. Measurements of the pressure gradients at the wall resulted from the observation of Lee *et al.* [11] that opposition control of the wall-normal velocity component of a streamwise vortex near the wall increases the pressure gradient in the spanwise direction directly below a streamwise vortex structure (see figure 2.2).

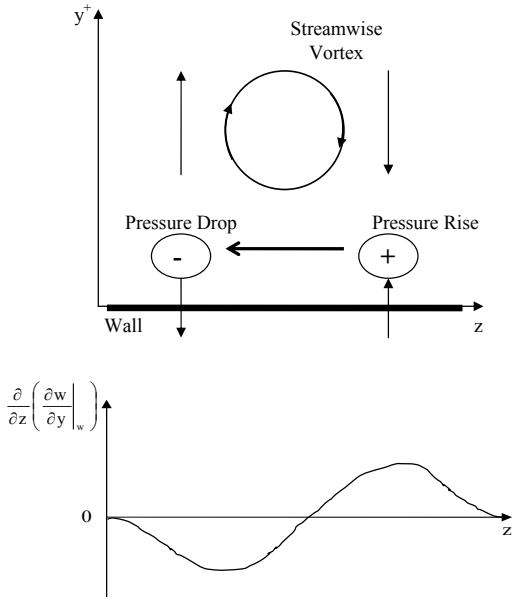


Figure 2.2: A key observation by Lee *et al.* [11] is the ability to measure a spanwise pressure gradient at the wall resulting from the interaction of a vortex near the wall. This suggests a spanwise increase in pressure gradient directly below streamwise vortices. In addition, the spanwise gradient of the spanwise wall shear stress (profile shown above) can be used to detect a streamwise vortex.

However, of the two detection quantities used by Lee *et al.* [11], they found that the spanwise derivative of the spanwise shear at the wall (shown in figure 2.2) is a slightly better quantity to use as a control input as it results in a 22% reduction in skin-friction drag.

Lee *et al.* [13] make use of a neural network based on the spanwise wall shear stresses to activate the blowing/suction velocity distribution at the wall, achieving a 20% reduction of skin-friction. The numerical control scheme detects edges of local high-shear stress regions, which are elongated in the streamwise direction, by measuring the spanwise variation of the spanwise shear stress. They investigate how wall shear stresses correlate with wall actuations and make use of a neural network to approximate the correlation which then predicts the optimal wall actuation. Lee *et al.* [13] find that the detection of the spanwise shear stress at several points across a spanwise distance of $z^+ = 90$ is enough to achieve good performance of the control algorithm. Furthermore, the correlations suggest that the root-mean-squared value of the actuation should be approximately $0.15u^*$ for suppression of the near-wall streamwise vortices.

2.1.2 Experimental research

Experimental investigations into the active control of the near-wall turbulent boundary layer also play an important role in finding practical methods of weakening the near-wall coherent structures that produce turbulence. However, experiments on turbulent boundary layer control are difficult due to the small length

and time scales that characterize turbulent flow, making the design and fabrication of sensors and actuators very difficult [14]. Rathnasingham and Breuer [14] investigate the active control of the near-wall turbulent boundary layer by using the key assumption that the dynamics of the large-scale coherent structures can be described as a linear process for a short period of time. The assumption is based on the observation that the mean shear of the near-wall turbulent flow will dominate during the short time it takes for the flow perturbations to evolve [15]. In addition, it is important to note that the assumption holds only for the time it takes a structure to convect from a sensor to an actuator, and does not imply that turbulence production is a linear mechanism [14]. Using an array of upstream flush-mounted sensors and flush-mounted resonant membrane-type actuators, diagram provided in figure 2.3, a series of experimental observations [1, 14] of the boundary layer provide the optimal transfer functions to predict the downstream characteristics of the streamwise velocity fluctuations. This process predicted, in contrast to the numerical correlations of Lee *et al.* [13], that the control jet amplitude is approximately three times stronger resulting in an optimal range with root-mean-squared values of 0.45 - 0.55u^* [1].

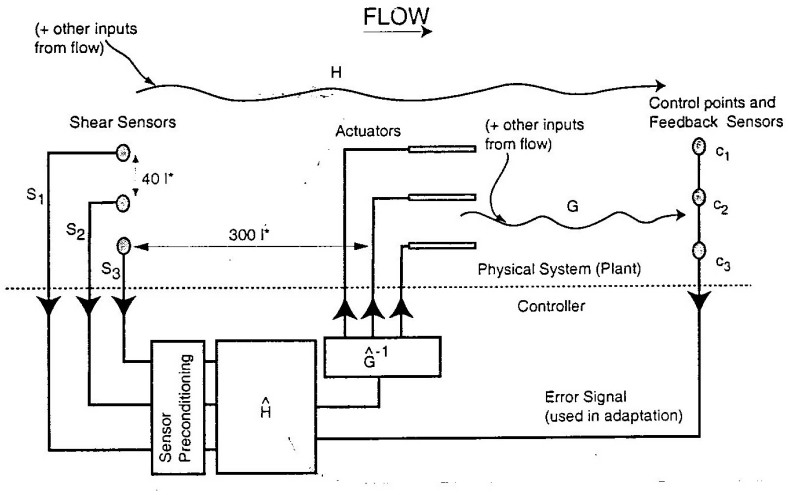


Figure 2.3: Schematic showing the detection sensors (s1, s2, s3), actuators (a1,a2,a3) and downstream control points (c1, c2, c3) [1, 14].

The difference in actuation strengths may be due to the observation that in the work of Breuer *et al.* [1] the actuators are discrete jets that do not cover the entire domain and therefore may need stronger actuations locally while the actuation mechanism of Lee *et al.* [13] is an instantaneous blowing/suction velocity distribution of the entire domain wall. In addition to finding the optimal transfer functions, the detection techniques were optimized to isolate the large-scale turbulent motion and improve the downstream correlations. Control results show a maximum reduction of streamwise velocity fluctuations of 30%, with the reduction spanning a region $100l^*$ downstream of the actuator, $50l^*$ in the spanwise direction, and $150l^*$ in the wall normal direction. Furthermore, results show wall pressure fluctuations were reduced by 15%, the local mean wall shear stresses were reduced by 7% and the bursting frequency, associated with sweep events, is reduced by up to 23% [14].

Rebbeck *et al*. [16] experimentally investigates how the near-wall turbulence structure of the boundary layer is modified when opposition control is attempted with a piston-type actuator. The actuator produces a wall-normal jet to cancel the downwash of high-momentum fluid associated with a sweep event. The effectiveness of the control method is judged from the observed changes in the burst intensity of the near-wall events. It is found that the performance of the control is very sensitive to the phase lag between detection and actuation. Specifically, the investigation shows a similar magnitude of burst intensity reduction when the opposition control is applied slightly late. However, the effectiveness is significantly reduced when the jet is issued early.

Another experimental investigation by Lew *et al.* [17] uses a linear array of MEMS surface shear stress sensors and a micro-machined pneumatic flap actuator to eliminate streak-like regions of high shear stress before their natural dissipation occurs along the channel wall. Open-loop actuation tests show that over an actuation cycle a net reduction of surface shear stress results. It is also found that the reduction is proportional to the actuation amplitude in relation to the boundary layer thickness [17]. Furthermore, to ensure interaction with the coherent structure, the actuation amplitude of the flap was limited to $y^+ < 7$ which puts it just beyond the viscous sub-layer. In addition, they show that the high shear stress regions occupy approximately 40% of the surface area and contain approximately 70% of the total surface shear.

Jacobson *et al.* [18] develop a piezoelectric cantilever flush-mounted with the wall to investigate active control of transitional and turbulent boundary layers. Actuation is applied by allowing a part of the flow surface to oscillate in and out of a cavity in the surface. The objective is to demonstrate control of steady and unsteady streamwise vortex disturbances in a laminar boundary layer, with the disturbances acting similar to eddies in the wall region of a turbulent boundary layer. The main idea is to draw fluid into the wall and pump it back out in a controlled manner in order to modify the near-wall flow. The piezoelectric actuator acts like a controllable vortex pair generator with no net mass flow through the boundary surface. The strength of the generated vortices is controlled by the amplitude of the actuator. Results show that the vortices are localized over the actuator and decay quickly downstream while the associated high- and low-speed streaks remain far downstream of the actuator [18]. Other interesting results of the investigation are the dimensional scales found necessary to implement active control successfully. Jacobson *et al.* [18] suggests the spanwise dimension of the control module should be of order $20l^*$ and in the streamwise dimension of order $200l^*$, totaling a control area of $Ac^+ \approx 4000$.

This review has brought to light recent progress made in turbulent boundary layer control research. The goal of the present work is to use aspects of the successful control techniques discussed above to develop a practical control method using the current numerical scheme [19] to best model a realistic simulation of

turbulent flow control. The average length and time scales of the near-wall coherent phenomena appear well documented, as are appropriate detection methods and dimensions of a successful control algorithm. In particular, the results of Breuer *et al.* [1] and Lee *et al.* [13] suggest that the root-mean-squared control jet amplitude should be in the range of 0.15 - 0.55u^*, with perhaps a tendency toward the stronger actuations if one is modeling discrete actuators versus using a uniform distribution of wall-normal velocity on the domain wall. In addition, a control algorithm should closely model the control modules on the dimensions of those in Rathnasingham *et al.* [14], where the sensors used are hot-wires aligned in the streamwise direction and the resonant membrane-type actuators are narrow in the spanwise dimension (~10l^*) and long in the streamwise dimension (~150l^*). These dimensions are also roughly the same as those suggested by Jacobson *et al.* [18]. Moreover, the findings of Rathnasingham *et al.* [14] suggest that the average convection speed of the large-scale structures of turbulent flow is approximately $u^+ = 10.7$. This allows a control algorithm to account for the time between the detection event upstream of the actuators and the actuation event imposed on the structures downstream. Finally, if the control algorithm requires flow variables measured at the wall only, a good detection scheme to model is that of Lee *et al.* [11]; this scheme detects the spanwise derivative of the spanwise shear stress at the wall and has been shown to work successfully by Endo *et al.* [12].

2.2 Current turbulent boundary layer control methodology

As described above, substantial reductions in drag can be achieved by wall-mounted actuators operating on the vortices or streaks to either stabilize the flow or reduce shear [20]. In order to do so, practical devices able to achieve fine flow control at the small scales near the surface need to be developed and their detailed interaction with a boundary layer needs to be studied. Therefore, the two control schemes shown together in figure 2.4 are developed; the first a control algorithm based on the detection methodology of Lee *et al.* [11, 13] and another based on the experiments of Lew *et al.* [17]. In addition, an array of discrete wall-normal MEMS micro-actuators that act upon oncoming streamwise vortices is used.

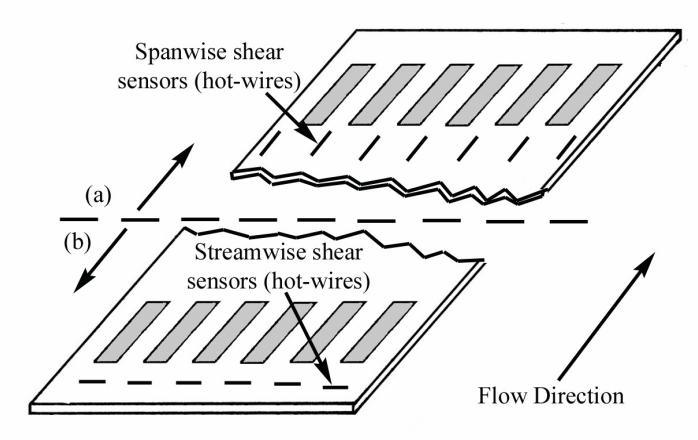


Figure 2.4: Schematic of the current detection schemes. Modeled as hot-wires approximately $2l^*$ above the surface; (a) detection of spanwise gradient of spanwise shear (based on Lee *et al.* [11, 13]); (b) detection of regions of high and low wall shear stress (based on Lew *et al.* [17]).

Early numerical studies in our group [21] have shown that when continuously operated in a turbulent boundary layer, small MEMS devices can substantially affect structures well beyond the buffer layer but such strong actuation was not found to decrease drag on the surface. Further studies with a single row of actuators [22] show that the physics of flow induced by an array of actuators differs considerably from the idealized case of uniform suction/blowing at the surface as in Koumoutsakos *et al.* [5] and Choi *et al*. [8]. However, the research reviewed in the previous section hints at the potential of discrete actuators in achieving some form of flow control. Thus the aim of this study is to simulate a physically realistic method of turbulent boundary layer control. The use of wall information upstream of individual actuators to detect oncoming streamwise vortices modifies a previous velocity opposition control method in Lee and Goldstein [22]. In the previous approach, which is based on the opposition control method of Choi *et al.* [8], the wall-normal velocity was sensed directly above each actuator slot along a plane $10.6l^*$ above the wall as shown in figure 2.5. An instantaneous response from the slot jets counteracted the detected wall-normal velocity.

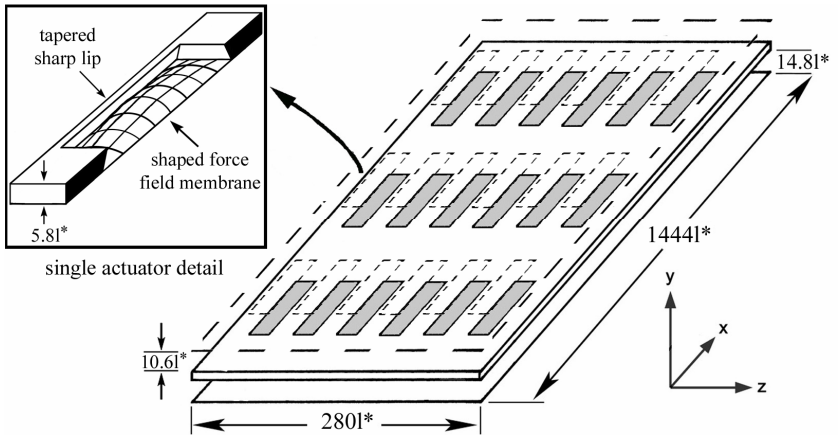


Figure 2.5: Schematic of manipulated surface with array of actuators and actuator detail. Sensing regions of Lee and Goldstein [22] (based on the Choi *et al.* [8] approach) above each slot are indicated with dashed lines.

The objective of the current study is to compare the reduction in drag of that method [22] with the more feasible method of upstream detection of wall information. It will also be interesting to see how the results of this current simulation will compare to those of Endo *et al.* [12] since the actuating mechanisms are similar; the main difference is the use of sub-surface sinusoidal membranes within cut out actuator slots. A brief discussion of the computational method and domain is followed by a detailed discussion of the results.

Chapter 3

Methodology

3.1 DNS code description

The spectral method initially used by Kim *et al.* [23] expands the spatial variables of the incompressible Navier-Stokes equations with Fourier and Chebyshev polynomials. The equations are solved with a Chebyshev-tau method with cosine grid clustering in the wall-normal direction. Time stepping is done with an Adams-Bashforth scheme for the non-linear terms and Crank-Nicholson for the viscous terms. A localized force field similar to the one described by Goldstein *et al.* [19] is used to simulate stationary and moving boundaries that make up the various parts of the actuators. Please refer to the work of Lee and Goldstein [21, 22, and 24], as well as the work by Lee [26] for a discussion of the details and issues related to the formulation of the governing equations and the issues related to code validation and convergence.

3.2 Flow field conditions and computational domain

The full computational domain, shown in figure 3.1, consists of a rectangular channel with mean flow in the x-direction. Flow is periodic in both the x and z directions while the horizontal top and bottom y-normal planes are defined as the channel boundaries.

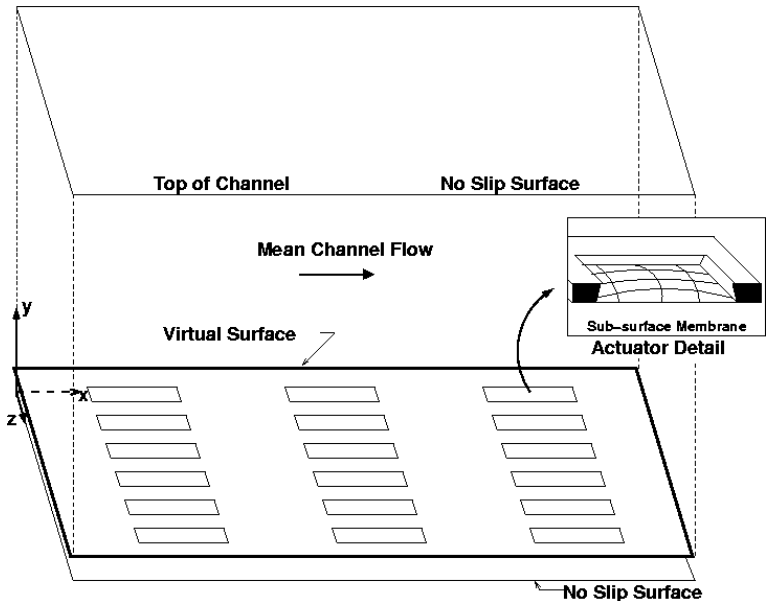


Figure 3.1: The full computational domain is shown including the no-slip channel boundary condition along the wall-normal axis. Also shown is the virtual surface created with the method of Goldstein *et al.* [19] just above the bottom boundary. A close up of the actuator is shown including the sinusoidal sub-surface membrane that creates the actuator pumping/suction action. Note: the bottom boundary is a slip surface for the second detection scheme to ease any Gibbs phenomena near the actuator edges.

The discrete actuator (inset of figure 3.1), is configured to be similar to those being tested by Rathnasingham and Breuer [14] and Wu and Breuer [25]. Individual rectangular holes are cut in a raised plate mounted above the lower boundary of the channel. The lips of each slot are tapered to be wider at the membrane location and narrower at the exit plane. The sharp lips are used to promote vortex separation [25] which was thought to be especially useful if modeling jet interactions with the turbulent boundary layer.

The membranes are modeled flush at the bottom of each cutout to make up the driving mechanism of each actuator. Membrane deflection is scaled using a factor, $C_{deflection}$, to match the volumetric displacement of a piston-like motion. That is, a unit displacement input to the membrane results in a larger than unity peak deflection that produces that same amount of ejected fluid as the membrane undergoing a piston-like displacement of one unit. Additionally, while each slot has its own driving mechanism, all slots share a common subsurface cavity that supplies fluid for the pumping/suction action.

For the domain in this study, unless otherwise stated, Re_{channel} was about 2,118 while R^* was about 116. For consistency, the friction velocity u^*, viscous length scale l^* and viscous time scale t^* were taken from the opposing top wall of the channel that contained no actuators. Those values were taken as constants and used throughout the study whenever friction properties were needed to normalize data or figures. With these parameters, the computational domain measured $280.2l^*$ in

width and 1,443.7l^* in length. Drag results for a second case examining a slightly higher $Re_{channel}$ of about 2,553 with an R^* of approximately 130 are also reported. For this case the friction velocity, viscous length scale and viscous time scale for an $R^* = 130$ are used. A summary of the relevant flow parameters for the turbulent channel flow is given in table 3.1.

Table 3.1: Summary of relevant flow parameters for the turbulent channel flow.

Quantity	$R^* = 116$	$R^* = 130$
Centerline Reynolds number - $Re_{channel}$	~2,118	~2,553
Turbulent Reynolds number - R^*	~116	~130
Friction velocity - u^*	0.03007	0.03350
Viscous length scale - l^*	0.00831	0.00724
Viscous time scale - t^*	0.27635	0.21619
Channel height - 2h	233.6l^*	268.0l^*
Channel length - L	1,443.7l^*	1,656.6l^*
Channel width - W	280.2l^*	321.6l^*
Streamwise resolution - ΔL	~7.5l^*	~8.6l^*
Spanwise resolution – ΔW	~1.5l^*	~1.7l^*
Time step – dt	0.0075	0.0075
Number of grid points – x, y, z	128, 64, 128	128, 64, 128

Chapter 4

Control algorithms based on wall information

4.1 Upstream detection of $\partial\tau_z/\partial z$

4.1.1 Formulation

In this section the results are discussed for the detection methodology which is based on that of Lee *et al.* [11, 13]. In particular, the spanwise gradient of the spanwise shear stress is used to detect the near-wall quasi-streamwise vortices upstream of an array of MEMS micro-actuators. Here the detailed dimensional information of the control algorithm is given. Using the immersed boundary technique of Goldstein *et al.* [19] the control module, i.e. the actuator and detection mechanism, is closely modeled after that of Rathnasingham and Breuer [14]. In their study, a pair of hot-wires aligned in the streamwise direction was placed upstream of the actuator to detect the spanwise wall shear stress. By differencing the pair signal, the derivative of the spanwise shear can be approximated. The spanwise spacing between hot-wires is equivalent to the characteristic streak width of $50l^*$. In the current control

algorithm such hot-wires are modeled by detecting the spanwise component of velocity at two different points, approximately $2l^*$ above the wall and approximately $47l^*$ apart. Since the detection points are located deep in the viscous sub-layer, measuring the wall shear stress is simply a matter of detecting the velocity component of interest, in the current case it is the spanwise velocity component, and dividing by a constant wall-normal height ($2l^*$). Knowing the exact spanwise spacing of the detection points allows us to calculate an approximation of the spanwise gradient of the spanwise shear stress at the wall. A typical profile of this quantity is shown in figure 2.2 for a streamwise vortex. As shown in that schematic, $\partial\tau_z/\partial z$ can be a clear indication of the presence of a streamwise vortex. The numerical approximation of the measured quantity used is given as:

$$\left.\frac{\partial}{\partial z}\left(\frac{\partial w}{\partial y}\right)\right|_{approximated} \cong \left[\frac{w_1}{\Delta y} - \frac{w_2}{\Delta y}\right]\cdot\frac{1}{\Delta z}; \qquad \text{where } \Delta y = 2l^*,\ \Delta z = 47l^* \tag{2}$$

Multiplication by a gain constant, *Pamp*, of the $\partial\tau_z/\partial z|_{wall}$ and membrane deflection scaling factor yields the opposition blowing/suction magnitude of the actuators.

$$\left(v_{blowing/suction}\right)_{time=t} = (Pamp)\cdot(C_{deflection})\cdot\left[\frac{\partial}{\partial z}\left(\frac{\partial w}{\partial y}\right)\right]_{approximated\ at\ time=t-\Delta t} \tag{3}$$

The dimensions of the actuator, at the wall surface, are approximately $23l^*$ wide and $143l^*$ long which roughly models the $10l^*$ by $150l^*$ of Ref. 14. The size of the current control module also follows that of Jacobson *et al.* [18] who suggests control modules of roughly $20l^*$ wide and $200l^*$ in length. The current algorithm is extended

into an array of 18 actuators each with a pair of upstream wall shear stress sensors to detect the oncoming streamwise vortices. The array, as shown in figures 2.4, 2.5, and 3.1, consists of three rows spaced evenly along the streamwise direction. Each row contains six actuators placed such that the actuators are aligned directly behind each other and have a pitch of approximately $47l^*$. The mean flow is periodic in the streamwise and spanwise directions. In addition, the total cut-out area covered by the array of actuators corresponds to 15% of the total surface. The time between the detection event upstream of the actuators and the actuation response is accounted for in the present control method. Using the findings of Rathnasingham and Breuer [14], which suggest that the average convection speed of the large-scale turbulent structures is approximately $u^+ = 10.7$, and coupled with the finding that quasi-streamwise vortices were successfully detected $50l^*$ upstream of an actuation location by Endo *et al.* [12], the current detection points were placed approximately $53l^*$ upstream of the leading edge of the actuator. A delay time variable, Δt, was introduced into the control algorithm to account for the convection time necessary for the detected vortical structure to reach the specified actuator. That time delay is parametrically examined below. A general summary of the control system as described above is shown as a schematic in figure 4.1.

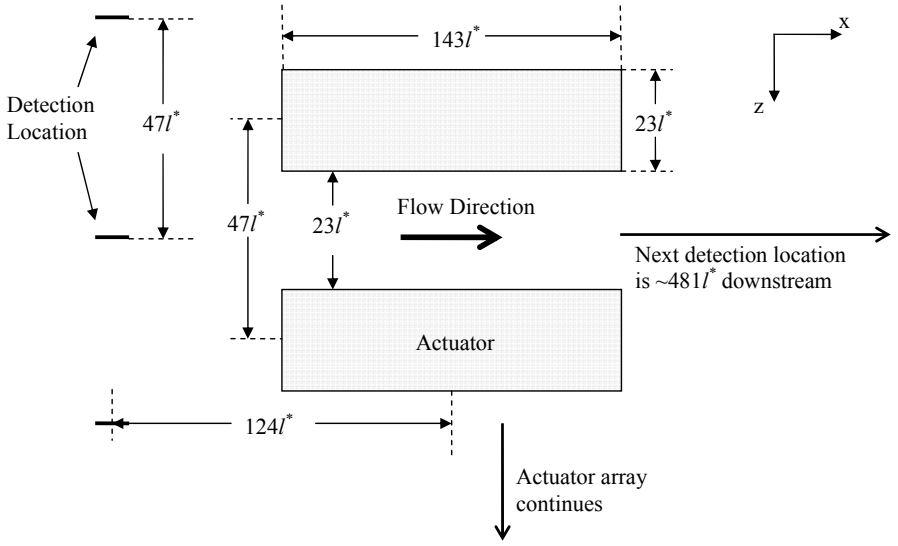


Figure 4.1: Summary of detection scheme including spatial dimensions for a portion of the controlled surface. Detection points (modeled as hot-wires aligned in the streamwise direction) are spaced to best sense a streamwise vortex. A time delay variable is introduced into the control algorithm to account for the time it takes a structure to convect the distance from the sensor to the actuator. Length and width of the actuator follow dimensions suggested by Rathnasingham and Breuer [14] and Jacobson *et al.* [18].

4.1.2 Optimization of signal gain

A study was performed to determine the size of a gain constant, *Pamp*, used to determine the strength of the actuating mechanism based on Eq. 3. The results of Breuer *et al.* [1] and Lee *et al.* [13] suggest that an actuation gain is applied such that the root-mean-squared control jet amplitude is in the range of 0.15 - 0.55u^*. Since the current scheme models discrete actuators, a control jet toward the stronger end of this amplitude range was chosen. A brief parametric study of the *Pamp* constant was performed to determine the best value, i.e. the case which resulted in the largest short-term drag reduction, was used in the long-term simulation run. Figure 4.2

displays the range of the actuator gains examined along with the number of time steps of delay. The figure shows the region of near-wall flow stability for various test cases. Stability was determined through the observation of the near-wall flow over the controlled surface during a series of short simulations.

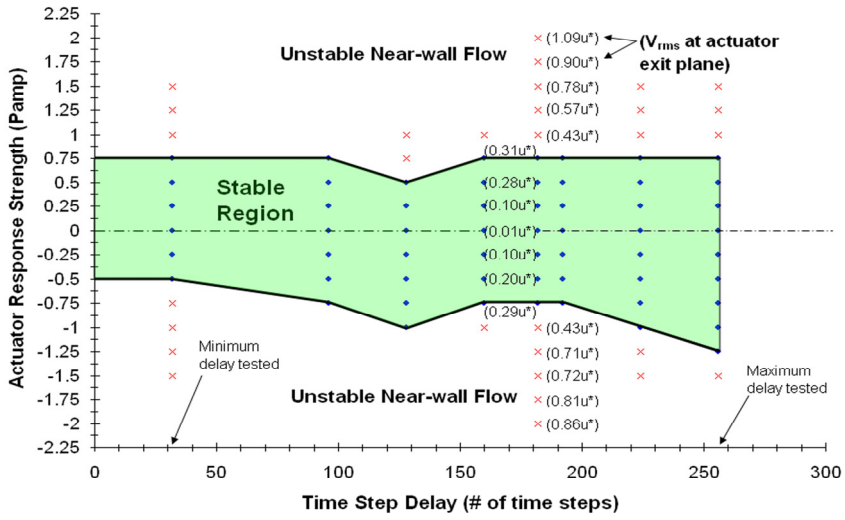


Figure 4.2: Stability plot of actuator gain constant, Pamp, and time delay, Δt, parametric study. Stability of the near-wall flow was determined through observation of the near-wall flow over the controlled surface during a short simulation. The stable region is shaded in green and blue dots represent stable test points. Red x's represent test points where the near-wall flow of the controlled surface was visually unstable. V_{rms} data at the actuator exit plane is given, in ( ), for the entire range of *Pamp* in the test matrix.

The tests showed that if *Pamp* is set too large, above ~0.75, the actuating membranes are forced too hard. This causes near-wall flow instability as well as CFL failure in the more extreme cases. These gains are outside of the green shaded stable region of figure 4.2. As will be discussed shortly, it was found that where the root-mean-square of the control jet amplitude is approximately $0.28u^*$ the drag appeared

minimized. This corresponds to a *Pamp* = 0.5, and as shown in figure 4.2, this value of gain is in the stable region for all values of time delay examined.

Figure 4.3 shows a series of drag traces for different gains compared to the inactive case of *Pamp* = 0.0. The drag ratio parameter is defined as the drag of the actuated surface divided by the drag of the opposing flat wall. A time delay of Δt = 182 time steps was imposed in these example simulations. The traces show that a *Pamp* = 0.5 results in a large and consistent drag minimization throughout the simulation.

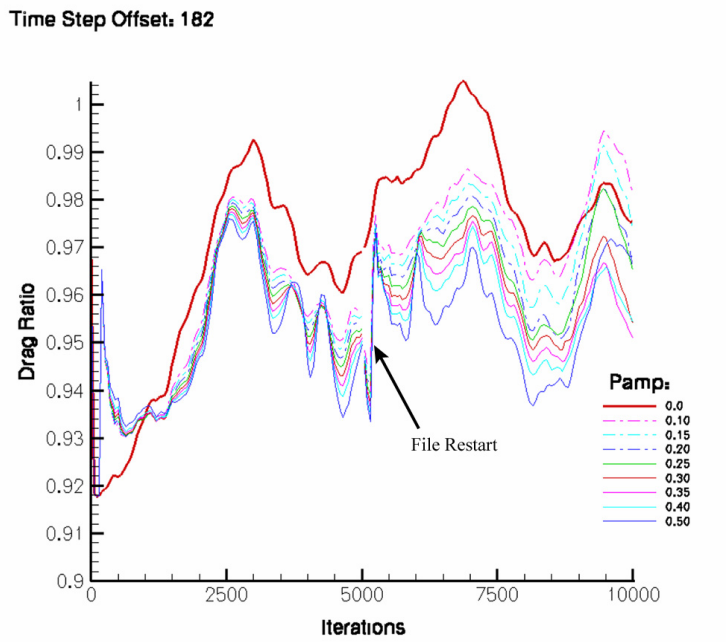
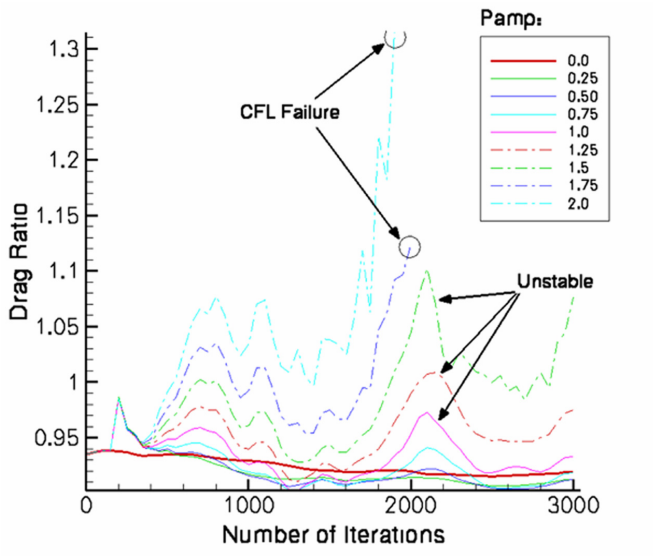
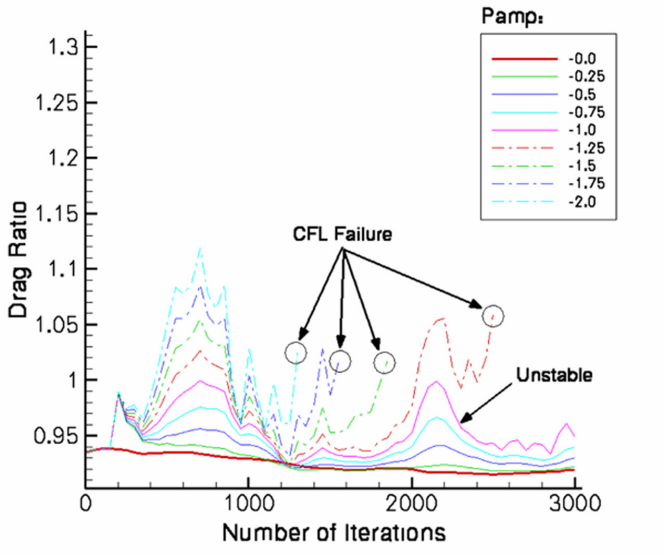


Figure 4.3: Drag trace of positive *Pamp* values within the stable region of Fig. 7 with Δt = 182. The short run shows using a *Pamp* = 0.5 consistently results in the largest drag reduction. The glitches at iteration 1 and 5000 are due to simulation restart points that did not account for the final 182 time steps. Such glitches are corrected in the long-term simulations.

In figure 4.4a the drag traces are shown for the full range of positive gains examined in the parametric study. In figure 4.4b example drag traces using the negative *Pamp* values are shown. In both figures 4.4a and b, some CFL failures result indicating the occurrence of flow instability for *Pamp* values above 1.0. In figure 4.4b, note that drag *increases* are generally shown for the smaller magnitude (negative) *Pamp* gains suggesting that a positive value is necessary. Based on such a parametric study, it was determined that $Pamp_{\text{optimal}} \approx 0.5$. In the sections that follow the inactive (*Pamp* = 0.0) control case is compared to the active case where *Pamp* is set to 0.5, unless otherwise stated.



a)



b)

Figure 4.4: (a) Drag traces of entire range of positive *Pamp* examined in parametric study at the particular case of Δt = 182. The traces support the observation of near-wall flow instabilities on the controlled surface for large *Pamp* values. (b) Traces showing the negative *Pamp* values. In general, no negative *Pamp* resulted in a consistent drag reduction and most resulted in unstable near-wall flow thereby supporting the use of a positive *Pamp* for the optimal drag reduction.

4.1.3 Optimization of time delay

A brief parametric study is also performed to determine the optimal time delay, or lag, between the detection of the oncoming turbulent structure and the actuation of the sub-surface membrane. This is done by fixing the streamwise distance of the upstream sensors and varying the time delay variable, Δt.

It is found that the time delay variations of the study, oddly, have very little effect over a range of $\Delta t = 0.87 - 6.95t^*$. In figure 4.5, the drag ratio of the inactive case is compared to that of the active case for various Δt.

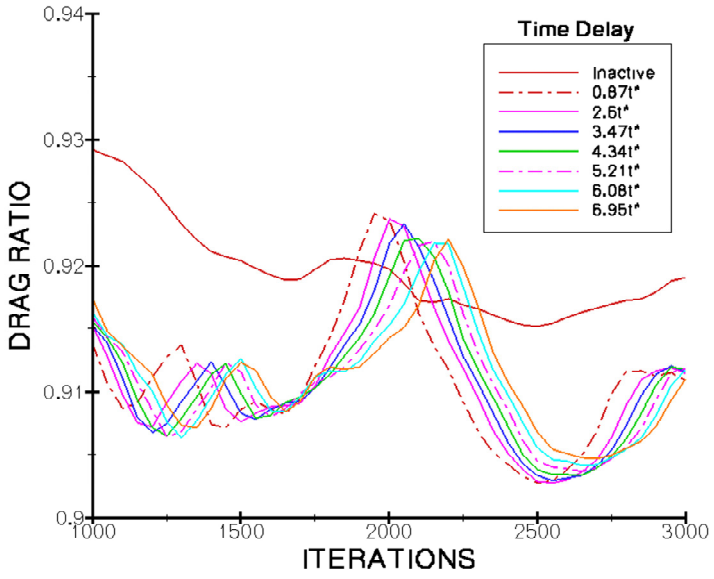


Figure 4.5: The results of a short simulation show the drag ratio of inactive case compared to that of the active case with various time delay constants (Δt) imposed. Results show varying the time delay has very little effect on the reduction mechanism. Longer delays result in the drag trace to be shifted more to the right, however the general characteristic of the resulting reduction mechanism is similar to that at any other delay, supporting the finding that the time delay imposed on the simulation does not effect the general result of a drag reduction.

Note how changing the time delay simply shifts the data to the right, indicating a longer delay between detection and actuation events. This zoomed in portion of a short simulation supports the fact that, at least for the delays considered, a very small effect is observed. A possible explanation for the lack of a significant effect of Δt is that typical lengths of the velocity streaks can extend up to $1000l^*$ in the streamwise direction [4] which corresponds to approximately $93t^*$ when assuming $u_c^+ = 10.7$ [14] as the average streamwise convection speed of the large-scale turbulent structures. Since the range of the time delay variable used in this parametric study is short compared to the duration of an individual streak it is possible that the range examined is too small and results in no net effect. At first this seems to contradict the study of Rebbeck *et al.* [16] where they find the control performance to be very sensitive to the phase lag between detection and actuation; however, in that study they detect bursting frequencies which are associated with sweep events. Sweep events are known to be much shorter in streamwise length, typically between 20 - $90l^*$ [4], corresponding to a range of time delay of approximately 2 - $8t^*$. However, since the current method seeks to stabilize the near-wall flow with the detection of long low-speed streaks, time delay may have little effect. As a result, the delay is set to $\Delta t = 182$ iterations (or approximately $5t^*$) which is equal to the convection time of a structure from the detection location to the actuator slot leading edge.

4.1.4 Randomization test case

In the previous two sections, the optimal signal gain (*Pamp* = 0.5) was determined and the time delay (Δt = 182 time steps) was found to be not tightly constrained for the current control algorithm. To determine if random blowing/suction resulted in any drag reductions, a randomization test of the sensors to actuators was conducted. That is, each sensor pair was connected to a random actuator instead of to the actuator in its own control unit. The random 'wiring' used for the current test case is shown in figure 4.6 where the sensor input to the actuators with matching numbers are swapped.

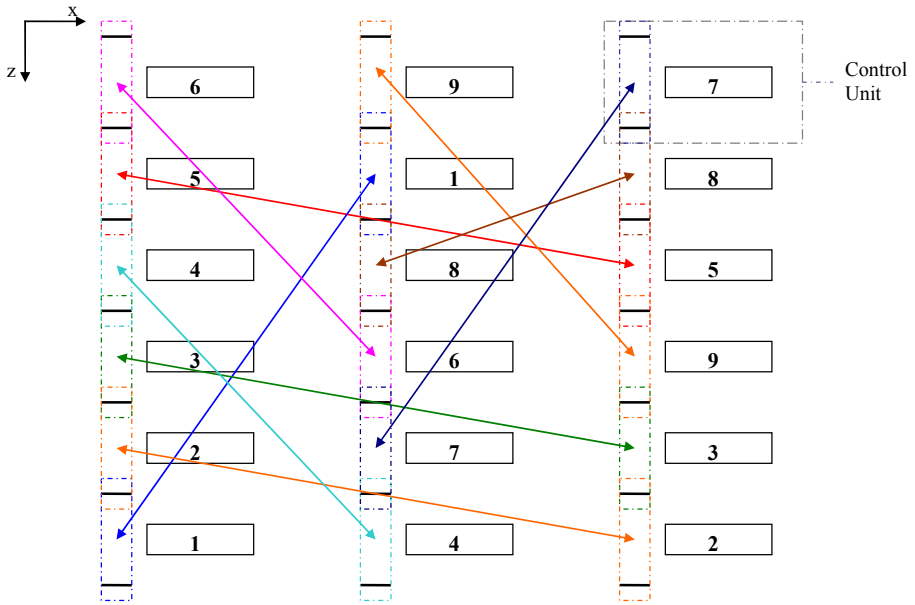


Figure 4.6: Random "wiring" of sensor-actuator control units. Control unit shown above is split up such that the sensor input signal associated with a specific actuator is swapped with an actuator of the matching number creating a randomly assigned sensor-actuator relationship for the actuator array.

If drag reduction occurred using this randomization, it would contradict the argument that a control algorithm specifically designed to reduce drag is needed for effective control. The random assignment of sensors to actuators was performed and a short simulation of approximately 15,000 iterations was conducted.

The results confirm the effectiveness of the actual control technique as the randomization process resulted in no large affect on the drag. Figure 4.7 shows the drag trace of this short test whereby control in the active and random cases where both turned on at the first iteration and both have the same gain, *Pamp* = 0.5.

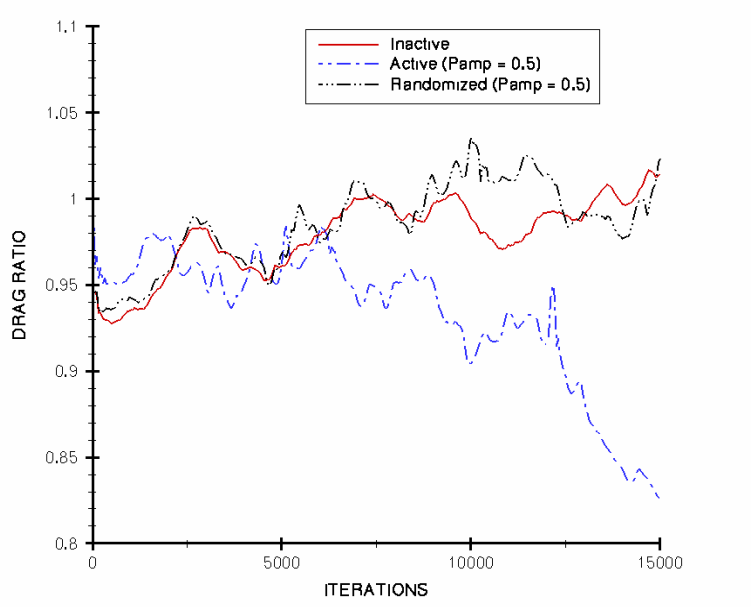


Figure 4.7: Drag traces comparing the inactive, active, and random cases all beginning at the same restart point at iteration 0. Control for the active and random cases (Pamp = 0.5) are both begun at iteration 0. As expected, the random control results in a trace very similar to the inactive case and supports the case that a well-defined control algorithm can reduce the drag on a controlled surface.

These cases are compared to the inactive case. One can see, from figure 4.7, that the random case closely follows the inactive case, at best. Comparison to the active case shows the difference between the current algorithm and that of random blowing/suction and suggests that a well-defined control algorithm has the potential to produce a drag reduction.

Using these results the next step was to run a longer simulation to 96,000 iterations (or $2605t^*$) to view the long-term effects of the control algorithm on the mean near-wall channel flow.

4.1.5 Mean and root-mean-squared profiles

The mean and root-mean-squared velocity profiles are first examined by averaging the results over time and across the spanwise and streamwise directions. The resulting profiles are compared with those of Kim *et al.* [23] and with those on the opposing flat wall of the channel. Figure 4.8 contains profiles showing three cases of interest which include; inactive control, active control, and the flow over the opposing smooth flat wall. By inactive control it is meant that the driving membranes are present but do not move. These mean velocity profiles show negligible differences from one another.

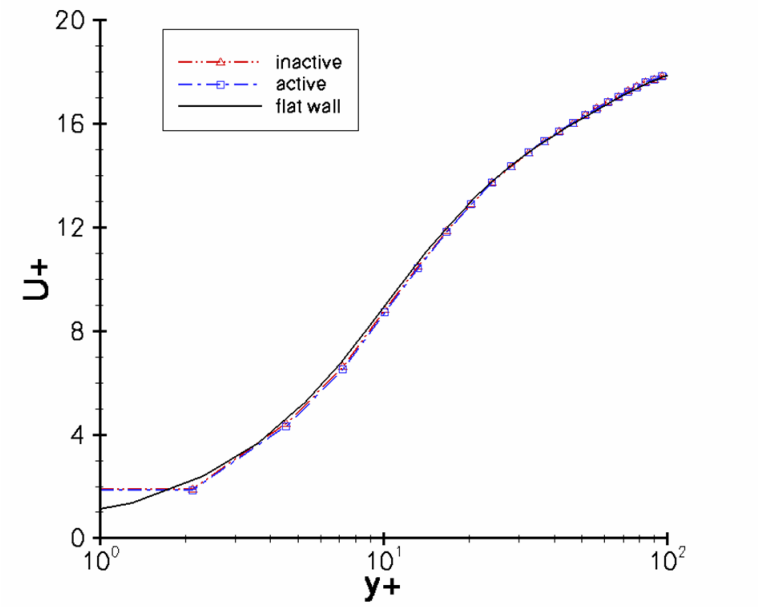


Figure 4.8: Mean velocity profile of streamwise velocity normalized by u^*. The profiles for the inactive, active, and opposing channel flat wall are shown using a semi-log plot.

The active and inactive profiles do not show data points below $y^+ \approx 2.1$ due to the resolution difference of the grid near the controlled surface and opposing flat wall. Since the controlled surface is created using the immersed boundary method of Goldstein *et al.* [19] above the computational domain boundary (see figures 2.5 and 3.1) the grid cells are larger than at the opposing flat wall. The flat portion of the profile very close to the wall is a result of grid resolution.

In figure 4.9, root-mean-square velocity fluctuations are normalized by u^*. In comparing the characteristics of the flat wall profile to those of Kim *et al.* [23] excellent agreement is found for the u_{rms} data.

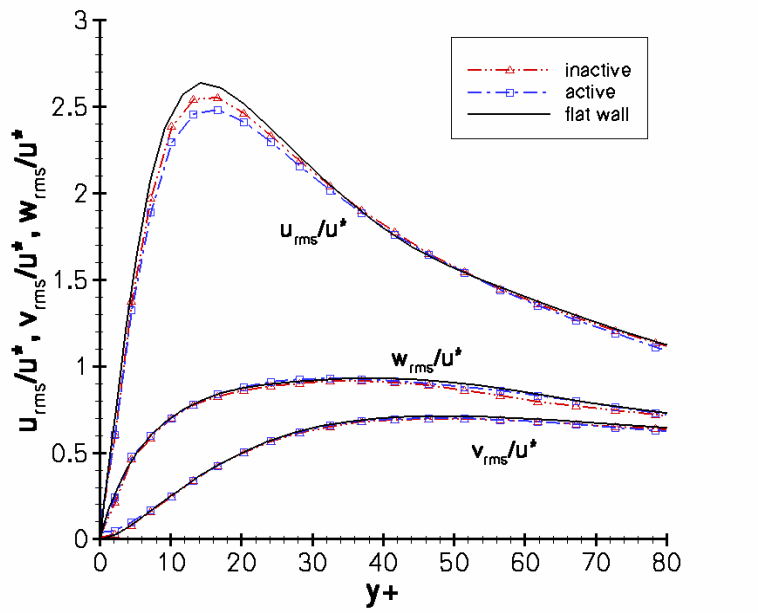


Figure 4.9: Root-mean-square velocity fluctuations normalized by the friction velocity, u^*, and shown in wall coordinates. Near-wall profiles of opposing channel flat wall are compared to the inactive and active control case.

A slight difference in the v_{rms} and w_{rms} data is observed in that the peak magnitude of these profiles is ≈ 0.2 less then those of Kim *et al.* [23]. However, this is most likely due to the choice of using a consistent friction velocity and viscous length scales for normalizing all the data, and to the fact that $R^* = 116$ is very low for turbulent channel flow. The main point is that the general trends follow those of a turbulent channel flow. In comparing the three cases of figure 4.9, one can see how simply having the altered control surface with inactive actuators may result in a slight decrease in the streamwise velocity fluctuations near the surface ($y^+ < 34$). With active control, this profile is further reduced. No significant changes for the other two velocity fluctuation components are noticed in the near-wall region except for

the small deviation from the flat wall profile for v_{rms}^{+} at $y^{+} < 3$. This deviation is due to the fact that the controlled surface contains an array of actuator slots which weakly blow/suck fluid through portions of the wall (the actuators). This could result in all three velocity fluctuation components having a non-zero mean value at $y^{+} = 0$. Table 4.1 summarizes the values of the root-mean-squared components of velocity at $y^{+} = 0$.

Table 4.1: Summary of velocity data at the control surface (y+ = 0).

Quantity	*Inactive case*	*Active case*	*Flat wall*
u_{rms}^{+}	0.066	0.102	0
v_{rms}^{+}	0.008	0.364	0
w_{rms}^{+}	0.022	0.026	0
$<-uv>/u^{*2}$	0.0005	0.017	0
$\omega_{xrms}t^{*}$	0.042	0.068	0.16
$\omega_{yrms}t^{*}$	0.012	0.018	0
$\omega_{zrms}t^{*}$	0.116	0.123	0.345

The Reynolds shear stress normalized by u^{*2} is shown in figure 4.10. In comparison with the results of Kim *et al.* [23], the peak Reynolds shear is slightly less (~ 0.1) which is consistent with the slightly smaller root-mean-squared (rms) velocity fluctuation profiles of figure 4.10. The location of peak Reynolds shear occurs at $y^{+} \approx 28$ which is also consistent with Kim *et al.* [23]. Very close to the surface there is little difference between the flat wall and inactive cases; however, one can see that the active case Reynolds shear stress is somewhat larger than the others at $y^{+} < 5$. This increase in Reynolds shear stress near the surface suggests an

increase in eddy viscosity near the wall which is not clearly consistent with a control producing a small skin-friction drag reduction. However, it does makes sense in terms of the Reynolds shear stress not having to be zero right at $y^+ = 0$ due to weak flows in and out through the slot exit planes.

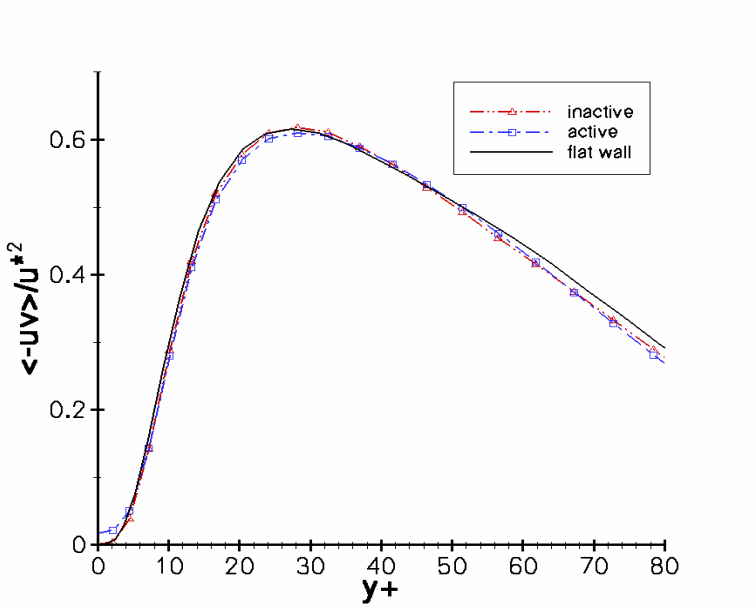


Figure 4.10: Reynolds shear stress for flat wall, inactive control, and active control normalized by the friction velocity, u^* and shown in wall coordinates.

Figure 4.11 shows the root-mean-square vorticity fluctuations normalized by t^*. The flat wall profiles are in excellent agreement with those of Kim *et al.* [23]. The characteristic trends are all similar with a slight difference between the slotted and flat surface in peak values for the x- and z-components. The inactive and active cases differ only slightly from the flat wall profile below $y^+ \approx 8$ perhaps partly due to

different grid resolution. Of course, a small portion of slip-surface over each actuator is averaged into each of these quantities.

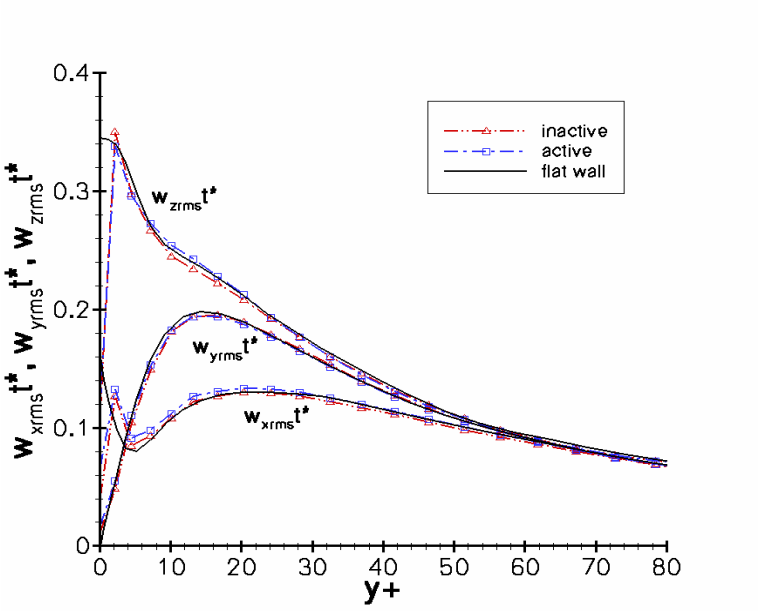


Figure 4.11: Root-mean-square vorticity fluctuations normalized by t^* and shown in wall coordinates. Profiles of a smooth flat wall are compared to the inactive and active control case.

The sharp decrease in the x-and z-components immediately below $y^+ \approx 2$ is explained by the large grid cells of these cases so close to the surface. Between $y^+ \approx 2$ and $y^+ \approx 8$ the x-component is increased for the inactive and active cases with the active case being slightly larger than the inactive. In the case of inactive actuators, this slightly elevated ω_{xrms} may be due to the actuators in the surface increasing the streamwise vorticity fluctuations simply by causing the slight dipping-down of fluid over an actuator slot. For the active case, this increase is perhaps explained by the

same cause as well as by the operation of the actuators which produce vorticity along the actuator slot lips when activated.

Having examined the time-averaged profiles one finds there is little to distinguish between the three cases. The profile characteristics agree fairly well with those established by Kim *et al.* [23] for turbulent channel flow. The main observation however is that, for the low Reynolds number, the profiles contain the characteristics of a turbulent channel flow.

In the next section an attempt is made to gain an understanding of the physical mechanisms at play in the near-wall region. The review of the time averaged, spanwise and streamwise averaged data show that the near-wall region is only weakly affected by the current array of actuators for both the inactive and active cases. First, the cross-sectional views of the computational domain are examined to try to put physical meaning into the characteristics shown above.

4.1.6 Cross-sectional results

It is beneficial to examine the physics of what is occurring in the near-wall region. An attempt to show the phenomena is done by using six sets of cross sectional views of the 3-D domain at various instances in time. Figure 4.12 consists of a wall-parallel plane located at $y^+ \approx 2.125$, which coincides with the height of the detection hot wires. On this plane, contours of wall-normal velocity are shown to clearly display the timing and strength of the actuation events. Crossing this plane are three cross sections of data located at the streamwise location of detection ($x^+ = 609$),

across the center of the row of actuators ($x^+ = 729$), and finally at a symmetrical distance downstream of the actuator row ($x^+ = 849$). These three planes show enstrophy ($|\omega|^2$) to easily view regions of low and high shear stress near the surface. Black vectors indicate the wall-normal velocity at the slot exit plane; their size is exaggerated compared to the white vectors in the three y-z planes. A final important note is that only the middle row of actuators is shown in the figure. The choice of which row to zoom-in on is arbitrary but such a close-up is necessary to view the details of the flow over the array of actuators.

The illustrations of figure 4.12 cover a small instance in time toward the beginning of a simulation run using $\Delta t = 425$.

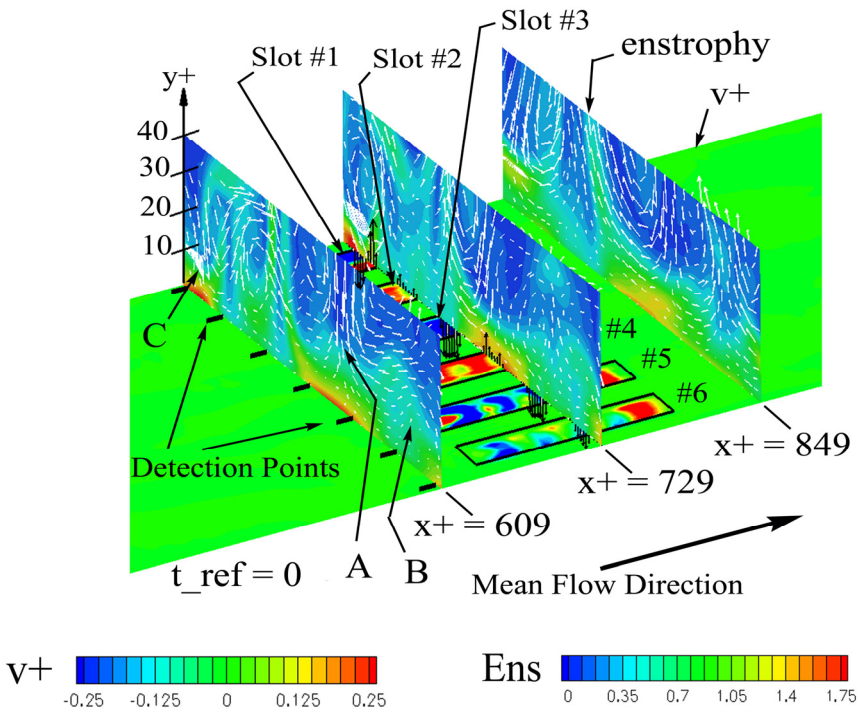


Figure 4.12a: Full caption on page 59. Reference time, $t_{ref} = 0.0t^*$.

This amount of delay accounts for the convection of the detected structure to the center of an actuator. The first frame (figure 4.12a) with $t_{ref} = 0$ corresponds to t = $11.5t^*$ after control was first turned on. In this frame a downward sweeping structure identified as "A" is shown at the detection region and is shown to approach slot #4. Observation of the near-wall enstrophy just below structure "A" reveals a large high-shear stress region on the surface. The actuation event associated with the detection of structure "A" is shown in figure 4.12b with $t_{ref} = 11.5t^*$.

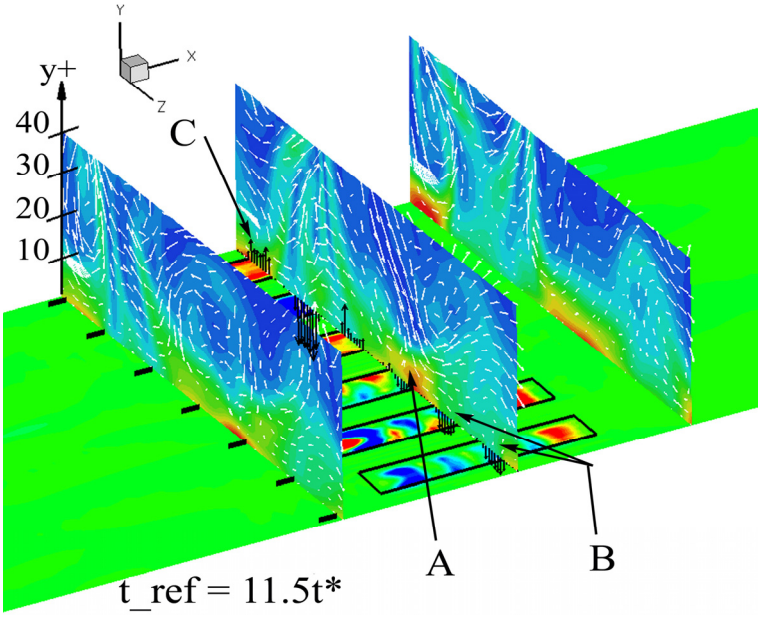


Figure 4.12b: Full caption on page 59. Reference time, $t_{ref} = 11.5t^*$.

The expected response of actuator #4 is a blowing action since it seems logical to blow at sweep events to deflect high-speed fluid away from the surface. At $t_{ref} = 11.5t^*$ one sees the strong sweep event dominates over the slot actuation resulting in

a net down flow over most of the actuator. However, a blowing action seems to have been performed by the actuator which can be seen with the small red contour region to the left side of actuator #4. This is a key observation for the current control algorithm because the magnitude of the actuator gain, *Pamp*, was set such that strong individual jet-pulses of fluid exiting or entering the actuator are not expected. Instead, the current method aims to create small disturbances or ripple-like effects on the control surface to make the near-wall flow less unstable; the goal is not the immediate local reduction of high-shear stress regions to reduce the skin-friction drag.

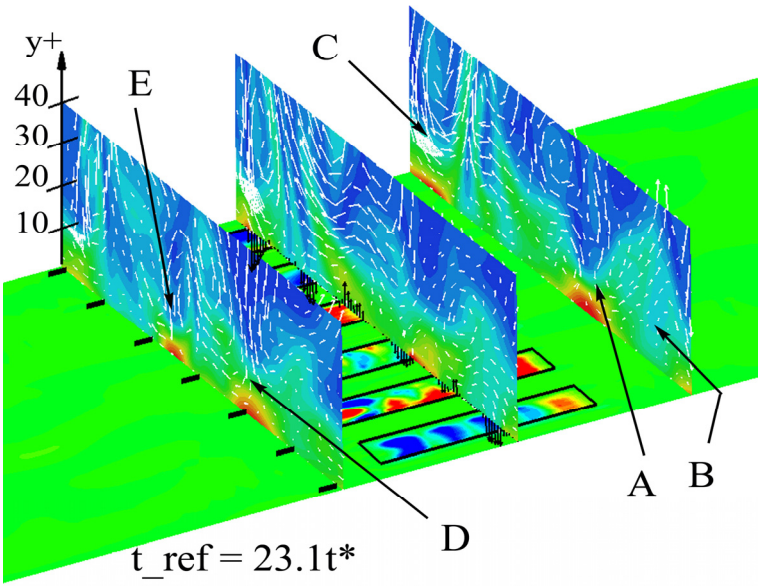


Figure 4.12c: Full caption on page 59. Reference time, $t_{ref} = 23.1t^*$.

The effects of the actuation are seen in figure 4.12c, $t_{ref} = 23.1t^*$, where a high-shear region "A" still exists downstream of slot #4; however, one should note how along

the control surface the spanwise width of the high-shear region has decreased. At x^+ = 849 it appears the high-shear region is shorter in the spanwise direction but taller in the wall-normal direction than it did at t_{ref} = 0 when the structure was located at x^+ = 609.

Referring back to figure 4.12a, t_{ref} = 0, structure "B" is illustrated as a general low-speed region with velocity vectors slightly pointed up but mainly directed in the positive z-direction. One can see the corresponding low enstrophy region below "B" on the wall. At $t_{ref} = 11.5t^*$ the proper control, suction, is applied to counter the weak counter-clockwise vortex above slots #4 and #5. This suction allows the low-shear region at the wall to continue downstream which is observed at x^+ = 849 at $t_{ref} = 23.1t^*$. Structure "C", pointed out at t_{ref} = 0, is another example of a high enstrophy or high-shear stress region at the wall with an associated sweep event of high-speed fluid. This structure seems to approach slot #1 and appears to have reached the actuator at $t_{ref} = 11.5t^*$. As expected the actuator response is a blowing action which seems to effectively counter the sweep event and keep the high-shear region away from the control surface. The high-shear region persists as seen at $t_{ref} = 23.1t^*$ at the downstream cross plane. While the velocity vectors appear deflected away from the wall, high-shear stress is still observed just below the structure.

Another set of interesting structures are shown at $t_{ref} = 23.1t^*$. Structure "E" seems to be counter-clockwise vortex which has a sweep region approaching slot #3 and causing a high-shear region at the wall. The appropriate blowing response is

shown at t_{ref} = 34.6t^* (figure 4.12d) with the addition of a suction response of actuator #2, both seem to counter the wall-normal velocity vectors just above the surface. However, the actuator response does not noticeably effect the size of the high-shear region which persists at t_{ref} = 46.1t^* (figure 4.12e).

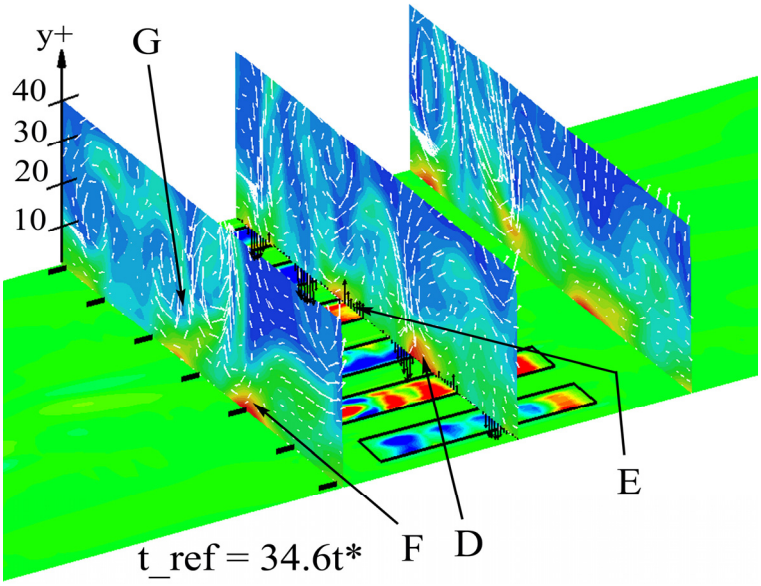


Figure 4.12d: Full caption on page 59. Reference time, t_{ref} = 34.6t^*.

Structure "D" (figure 4.12c) is a net downwash region associated with a pair of counter-rotating vortices. A high-shear region is shown to approach between slot #4 and #5 with a clockwise vortex toward the left of the sweep event and centered at approximately y^+ = 20. A weaker counter-clockwise begins to form to the right. At t_{ref} = 34.6t^* a pair of counter-rotating vortices is clearly observed over slot #4. The suction of actuator #4 and blowing from actuator #5 seems to have strengthened; such a response from actuators #4 and #5 seem inconsistent with the overall control

goal of velocity opposition in the near-wall region. Observing the structure at the downstream plane at $t_{ref} = 46.1t^*$ one observes a stronger, more dominant counter-clockwise vortex centered over the spanwise location of slot #4.

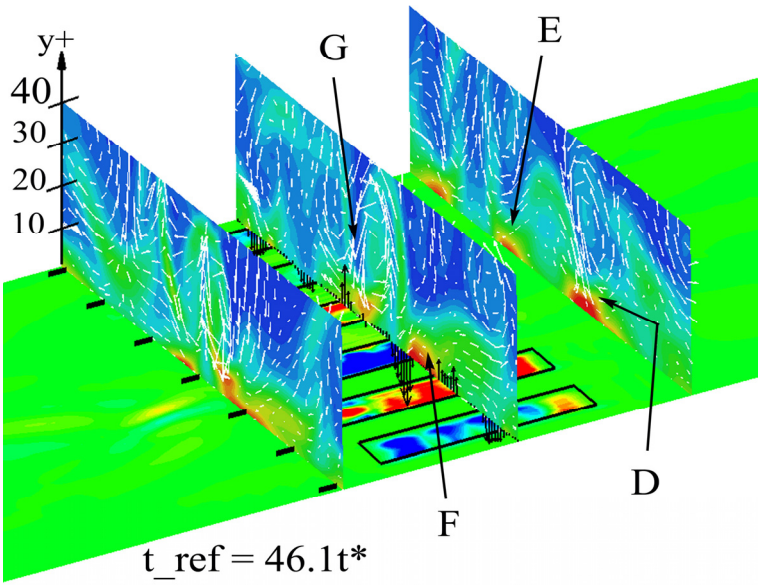


Figure 4.12e: Full caption on page 59. Reference time, $t_{ref} = 46.1t^*$.

Both of the structures "G" and "F" illustrated in $t_{ref} = 34.6t^*$ show high-shear regions at the wall. Structure "G" is clearly a pair of counter-rotating vortices while "F" seems to show a very weak pair of vortices. The actuator response is consistent with those already observed. Structure "F" approaches between slots #4 and #5 and the actuator pair again shows opposite actuation between neighboring actuators with fluid exiting at slot #5 and suction at slot #4 at $t_{ref} = 46.1t^*$. Observing this high-shear region "F" at $t_{ref} = 57.7t^*$ in figure 4.12f shows that the structure has been pulled toward actuator #5. The final structure illustrated here is "G" which is a pair

of counter-rotating vortices approaching slots #2 - #4 ($t_{ref} = 34.6t^*$). This time the suction of actuator #2 and #4 and blowing of #3 is consistent with the velocity opposition approach as each seem to counter the wall-normal velocities at $y^+ = 10$. This opposition is shown at $t_{ref} = 46.1t^*$ with varying degrees of actuation across the three actuators. The clockwise vortex dissipates by the time the structure reaches $x^+ = 849$ at $t_{ref} = 57.7t^*$ and a single counter-clockwise vortex remains with an associated high-shear region just below.

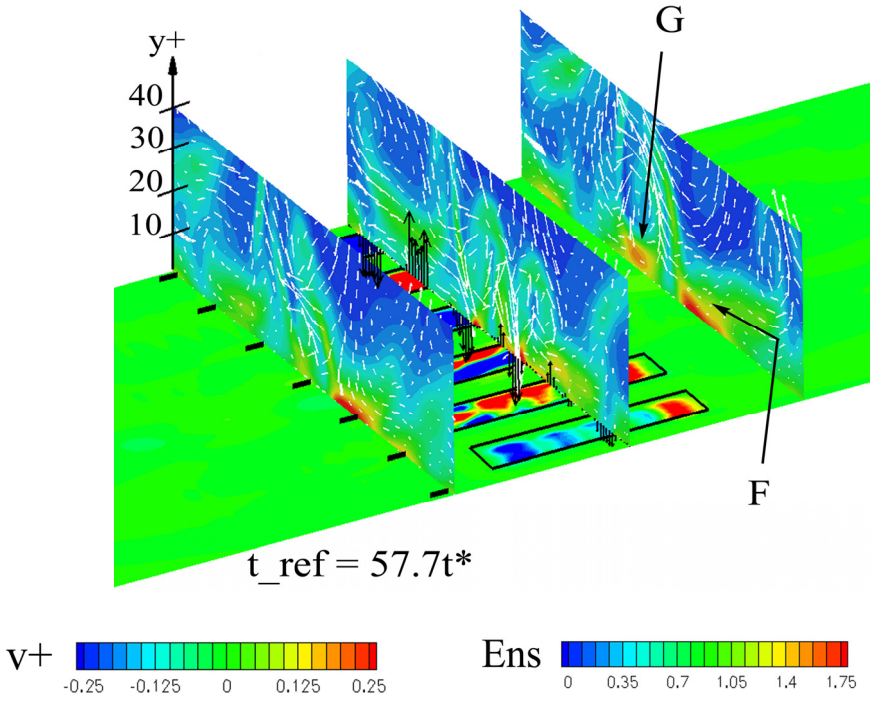


Figure 4.12f, continued: Ref. time, $t_{ref} = 57.7t^*$. Cross-sectional compilation of data to illustrate near wall vortical structures. Detection events at $x^+ = 609$ are shown with the corresponding actuation event at $x^+=729$. Downstream effects of actuation shown at $x^+ = 849$. Slices along the x-z plane display contours of wall-normal velocity while y-z slices show contours of enstrophy. This series of illustrations begins $11.5t^*$ after the control was first turned on. Vectors show velocities in plane. Black vectors showing slot suction or blowing are exaggerated compared to the white vectors.

While the series of illustrations of figure 4.12 do not show a completely consistent trend towards the reduction of eddies downstream of the actuator row, specifically in the near-wall region, the picture may become somewhat more clear if the time-averaged data over the entire simulation duration is examined at a wall-normal height of approximately $y^+ = 2$ as illustrated in figure 4.13. In figure 4.13 a time average of the entire $2606t^*$ of data is taken and the streamwise velocity component is plotted, normalized by the friction velocity.

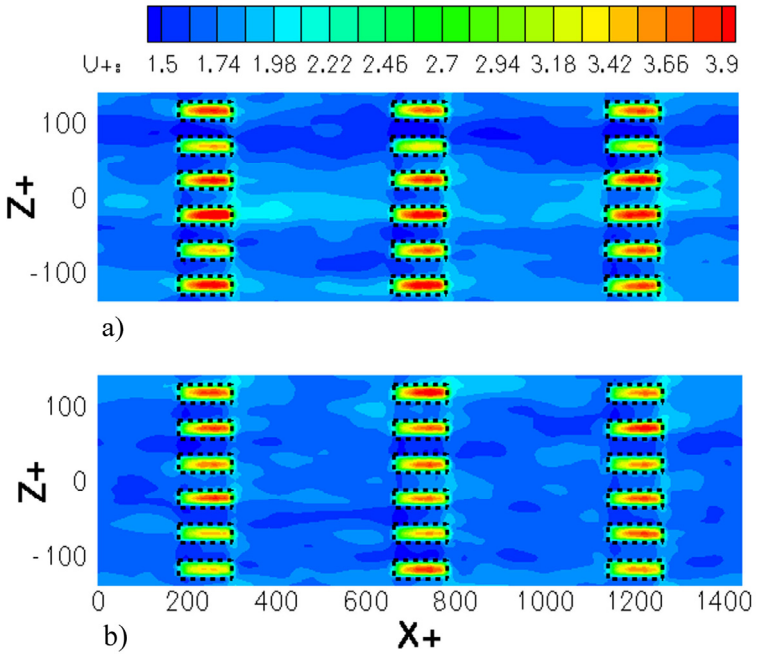


Figure 4.13: Time-averaged contours of streamwise velocity on an *xz*-plane located at $y^+ \approx 2.1$ above the controlled surface for: (a) inactive case (*Pamp* = 0.0) and (b) active case (*Pamp* = 0.5).

One immediately sees the streamwise acceleration of fluid over each actuator. This is consistent with the notion of altering the no-slip boundary with small portions of a slip-like boundary condition over each recessed cavity. Moreover, even with the

active case, the boundary condition over the individual actuators is slip-like and regions of higher streamwise velocity result. In comparing the two cases, figure 4.13a shows very persistent, long streamwise structures covering the entire length of the domain and spaced approximately $50l^*$ in the spanwise direction. This seems to indicate the presence of the long low-speed streaks characteristic of turbulent wall flow. They are seen simply because the simulation was not run long enough in this time-averaged data set. Figure 4.13b shows the active case and one observes how the long streamwise streaky structures are broken up into shorter, thinner structures which extend at most a third of the domain length. This may indicate that small amplitude actuation is enough to break up the long coherent near-wall structures typical of turbulence.

Figure 4.14 shows the time and spanwise averaged contours of streamwise velocity on the x-z plane located at $y^+ \approx 2$. By averaging in the spanwise direction the long streaks are no longer present and a better view of the flow near the actuators is gained. As was observed in figure 4.13, streamwise acceleration of the flow is again seen over the actuators. Also notice that beside each actuator is a region of lower speed fluid which occurs as the acceleration of fluid over an actuator slightly entrains the surrounding flow resulting in an inward turning of the velocity streamlines towards the center of the actuator. The spreading streamlines result in a smaller streamwise component of velocity for the regions adjacent to the actuators.

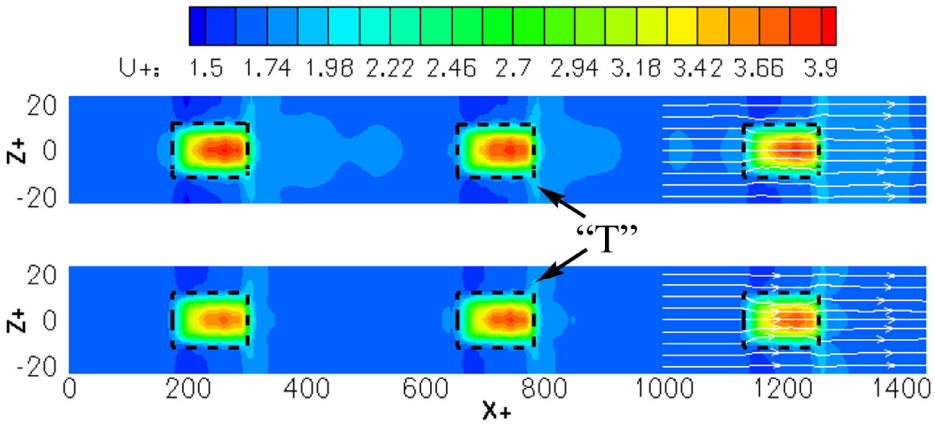


Figure 4.14: Time and span-averaged contours of streamwise velocity on an *xz*-plane located at $y^+ \approx 2.1$ above the controlled surface for: (a) inactive case and (b) active case. Note the stretched z^+ length scale.

Also noticeable is that these low-speed regions seem to be dominant towards the upstream half of the actuators. This may be explained by the dipping of the streamlines into a cavity. The streamlines dip down just aft of the cavity leading edge and entrain the surrounding fluid which causes the deceleration of fluid between actuators. At the downstream end of the cavity the streamlines rise up and out. There, the upward shift in the streamlines compresses the streamlines beside the slot thus producing faster moving fluid surrounding the downstream end of the actuator. This is seen as the characteristic "T" shape of the surface shear stress patterns at the trailing edges of the actuators. A final observation of figure 4.14 is, in the case of active control, the general shortening of the high speed fluid regions directly downstream of the actuators. This is consistent with regions of reduced shear stress and the overall reduction of skin-friction drag.

4.1.7 Drag reduction

To quantify the performance of the control algorithms a simple ratio is used to compare the shear stresses on the top, un-actuated channel surface and the bottom surface which includes the flow control mechanism. This drag ratio is

$$\text{Drag Ratio} = \frac{D_{\text{actuated side}}}{D_{\text{top wall}}} \tag{4}$$

A drag ratio of less then 1 means the control algorithm is working to reduce the skin-friction drag of the manipulated bottom surface. Figure 4.15 shows the drag ratio trace and running average for the current control algorithm. The figure compares an inactive versus an active case. The simulation was run to 96,000 iterations, which is equivalent to $2605t^*$.

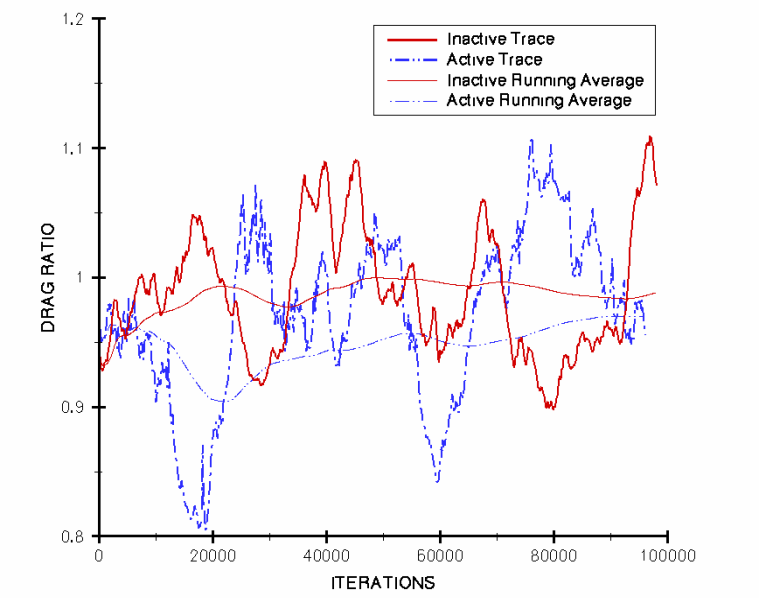


Figure 4.15: Drag reductions for the inactive and active cases at $R^* = 116$. Shown is the drag trace and running average for each for time duration of approximately $2606t^*$.

The running averages of both cases suggest a small drag reduction throughout the duration of the simulation. As seen in the figure the drag trace for the inactive case jumps by ±10% while that of the active case peaks at +10% (increase) and dips as low as -20% (reduction). In general, there seems to be an intermittency period of roughly 15,000 iterations. This period is equivalent to the flow traveling almost three times the length of the domain using the average convection speed of structures at y^+ ~10. The general trend shown in the running average is that while the initial control resulted in a large reduction (~10%), over longer periods of time the control will settle to a very small drag reduction value of just a couple of percent.

For the duration of the simulation above, active control provides a small drag reduction of 3% ±2.1% whereas the average reduction of the inactive case is 1.2% ±1.4%. Here, the error estimates are calculated by using an autocorrelation of the drag ratio data to determine the length of an independent realization. This realization number is then used to determine the number of statistically independent samples in a data set and the period of these realizations which can be inserted into a statistics program that calculates the standard error. These results are in good agreement with those of Lee [26] which uses the same immersed boundary layer technique and actuators but a different control methodology of detecting vertical fluctuations over the slots and applying opposition control. In addition, figure 4.16 shows the results for another active case run at a slightly higher $Re_{channel}$ = 2,552 or R^* = 130 also agree with the above results suggesting that a small increase in the

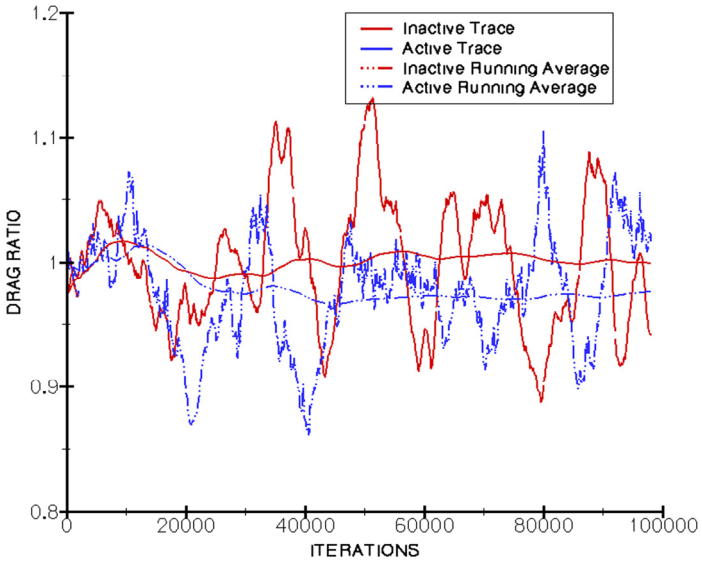


Figure 4.16: Drag reductions for the inactive and active cases at $R^* = 130$. Shown is the drag trace and running average for each for time duration of approximately $3400t^*$.

Reynolds Number using the current detection method does not make a significant difference in the drag reduction. In this $R^* = 130$ case, a small reduction of 2.4% ±0.97% was calculated for an equivalent *Pamp* of 0.5 and the simulation was run to 98,000 iterations which is equivalent to $3400t^*$. These data are summarized in Table 4.2. A final note is that the figure 4.15 shows that the running average is still not completely stable at 96,000 iterations which suggests that the simulation has not been run long enough and a simulation out to ~300,000 iterations may be needed to achieve a better statistical average.

Table 4.2: Average drag reductions for different cases compared to Lee [26]. The number of independent realizations is determined through an auto correlation of the drag.

Case	*Indep. Realizations*	*Average Drag Reduction*	*Standard error*
Inactive (*Pamp* = 0.0)	12	1.2%	±1.4%
Active (*Pamp* = 0.5, R^* = 116)	10	3.0%	±2.1%
Active (*Pamp* = 0.5, R^* = 130)	22	2.4%	±1.0%
Inactive, Lee result[26]	10	1.1%	±3.8%
Active, Lee result[26]	38	4.6%	±1.3%

In the next section an investigation is performed for a second control algorithm which also utilizes flow information at the wall. The algorithm is easier to comprehend physically as it will target the streamwise component of the high- and low-shear stress regions upstream of the actuators which correspond to sweep and ejection events. Comparison of the two control algorithms may help to confirm that multiple detection quantities can be associated with the coherent near-wall structures and that the use of one over the other may result in larger drag reduction.

4.2 Upstream detection of $\partial u/\partial y$

4.2.1 Formulation

Here preliminary results are presented using a second detection method based on the experiments of Lew *et al.* [17]. The main objective of this scheme is to detect high- and low-streamwise shear stress regions just upstream of a micro-actuator and manipulate the near-wall flow such that the near-wall coherent structures are weakened. In contrast to the previous detection method, the sensors are modeled as

hot-wires aligned in the spanwise direction. The streamwise component of velocity is averaged over 16 detection points covering a spanwise width of $23l^*$, equivalent to the width of a single actuator, at approximately $4.51l^*$ above the surface. This height of detection sensors is approximately double that of the previous algorithm in an attempt to avoid the potential detection or amplification of Gibbs phenomena near the edges of the actuator. Furthermore, the boundary condition along the bottom of the computational domain is changed from a no-slip to a slip boundary to ease the Gibbs phenomena. The high- and low- shear regions are determined by comparing an averaged velocity ($u_{avg,\ measured}$) to an optimized mean streamwise velocity. The optimized mean streamwise velocity is taken over the entire actuator plate surface, and is measured as $3.98u^*$. The optimization of this reference velocity is further described in the following section. Equation 5 summarizes the calculation involved in determining whether a high- or low-shear region is approaching an actuator.

$$\frac{u^+_{avg,\ measured}}{u^+\big|_{\substack{\text{optimized mean}\\ \text{over plate surface}}}} - 1 \quad = \quad \begin{Bmatrix} > 0 & \text{high} - \text{shear region} \\ = 0 & \\ < 0 & \text{low} - \text{shear region} \end{Bmatrix} \tag{5}$$

$$\left(v_{blowing\,/\,suction}\right)_{time=t} = (Pamp)\cdot(C_{deflection})\cdot\left[\frac{\frac{u_{avg,measured}}{u^*}}{3.98} - 1\right]_{at\ time=t-\Delta t} \tag{6}$$

Equation 6 shows that a positive value in Eq. 5 will result in a blowing of the actuator in order to deflect away some of the high-speed fluid sweeping towards the wall. Conversely, a negative value in Eq. 5 will direct the actuator to provide

suction to a low-speed streak thereby reducing the ejection of this fluid away from the wall, hopefully weakening the streak and perhaps keeping it stable. As shown in figure 4.17, the placement of the sensors is approximately $75l^*$ upstream of the actuator leading edge. The actuating mechanism and spacing between actuators remains the same as the previous scheme.

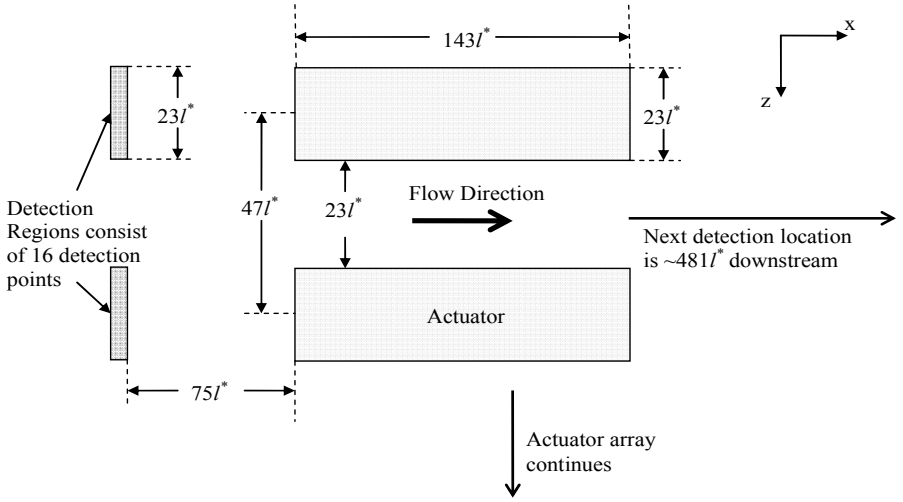


Figure 4.17: Summary of spatial dimensions of detection scheme. Shown is only a partial section of the controlled surface. The average over 16 detection points (modeled as a grey hot-wire aligned in the spanwise direction) is used to sense high- and low- shear regions just upstream of an actuator. A time delay variable is introduced into the control algorithm so that the time it takes a structure to convect from the sensor to the center of the actuator ($147l^*$) is accounted for. Actuator dimensions follow those suggested by Rathnasingham and Breuer [14] and Jacobson *et al.* [18].

The next section discusses the optimization of the current control scheme. Specifically, the parameters examined are the time delay and actuator gain signal which are similar to the previous control algorithm. In addition to these two parameters, the optimization of the reference velocity value for the current scheme is examined.

4.2.2 Optimization process

Similar to the previous control scheme, an optimization process is necessary to find the test case that presents that greatest drag benefit. The examination of an additional parameter is important due to the control methodology stated in equations 5 and 6. Here we discuss the optimization of the mean streamwise velocity over the actuator plate which is selected as our reference velocity in Eq. 5. The quantitative comparison between each local measured streamwise velocity and a fixed reference velocity determines whether approaching flow should be considered a low- or high-speed fluid and therefore this quantity is important to finding a control scheme that works to reduce the drag over the surface. The most obvious idea would be to compare the measured velocity at each wire located at $y^+ = 4.51$ to the mean x-velocity on the $y^+ = 4.51$ plane. But it is found that choosing the instantaneous mean velocity in a plane a specific height above the actuator surface results in an unstable system. Specifically, the reference value in this case appears consistently too low compared to the measured velocity thereby communicating to the actuator that a high-shear stress region is always approaching the actuators and therefore they consistently blow more fluid out of the slot. Instead of an instantaneous reference value a series of fixed reference values were attempted including a reference velocity of $4.51u^*$, equivalent to the height of the sensor placement above the surface ($4.51l^*$), and another reference velocity slightly lower $3.98u^*$ obtained by calculating the mean streamwise velocity three planes above the surface. In brief simulations of

7000 time steps with a nominal range of *Pamp* and a $\Delta t = 382$, both reference values produce a stable response unlike the instantaneous test case and both give similar drag results. However, using $3.98u^*$ as the reference velocity produces a slightly better drag benefit. Therefore, based on this short simulation, a reference velocity of $3.98u^*$ is chosen to produce the greatest drag benefit and discussion of the appropriate time delay and signal gain follows.

A series of time delays and actuator signal gains are used to determine the best test case for achieving an overall drag reduction. Three time delays are chosen which correspond to the time it takes the fluid to convect at a speed of $10.7u^*$ from the upstream sensors to three positions [14]; to the actuator leading edge, to $1/4^{th}$ the actuator length, and to 1/2 the actuator length. More extreme cases such as early actuation (convection time to half the distance to the actuator leading edge) and delayed actuation (convection time to the trailing edge of the actuator) are not investigated to save on computation time in the optimization process. However, a complete range of actuator signal gains (*Pamp*) were tested which cover similar actuation strengths to those used in the previous control scheme. The gains attempted are shown below in figure 4.18 and gives an overall view of the various test cases included in the parametric study.

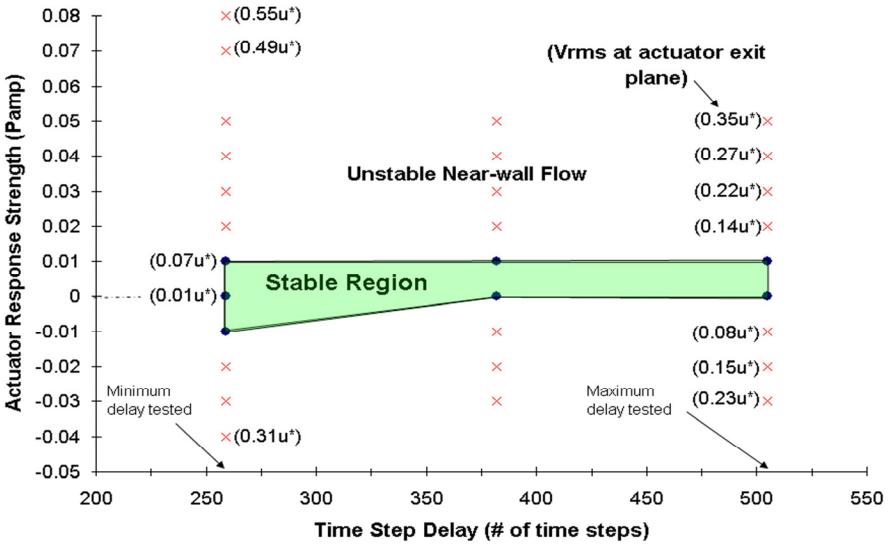


Figure 4.18: Plot of actuator gain constant, *Pamp*, and time delay, Δt, parametric test matrix. Stability of the near-wall flow was determined through observation of the near-wall flow over the controlled surface during a short simulation. The stable region is shaded in green and blue dots represent stable test points. Red x's represent test points where the near-wall flow of the controlled surface was visually unstable. V_{rms} data at the actuator exit plane is given, in ( ), for the entire range of *Pamp* in the test matrix.

CFL instability is not a major factor in the various test cases. However, the flow near the edges of the actuators may become unstable possibly due to Gibbs phenomena around the edges of the actuator geometry.

One observes from the figure above and by comparing the drag ratio vs. time plots that varying the time delay parameter has no major affect on the effectiveness of the control scheme. This was also seen in the previous control scheme and is most likely explained by the natural streamwise low-speed velocity streak lengths involved in near-wall channel flow which can extend up to approximately $1000l^*$, O(10x) the length of our sensor to actuator distance. Also, one should be aware of

the possibility that actuation on the fluid may result in a shortening of these low-speed streamwise velocity streaks such that the lengths are O(1x) the length of our sensor to actuator distance. In this case, fine tuning of the time delay parameter may be necessary to achieve the greatest drag benefit, as well as keeping the system stable.

Results of the parametric study show that in a short simulation of 54,000 iterations, equivalent to $1466t^*$, the optimal time delay is Δt = 259 iterations. This delay is equivalent to the time it takes the flow to travel, assuming $u_c^+ = 10.7$ [14], from the detection sensors to the leading edge of the actuator. A very small *Pamp* = 0.01 is the optimal case among those examined above producing a very small $v_{rms} \approx 0.07u^*$ at the slot exit plane.

While a finely resolved parametric study of the actuator gain signal, *Pamp*, and the time delay variable, Δt, has yet to be completed, the next section provides preliminary drag results using the current optimal values of *Pamp* = 0.01 and Δt = 259.

4.2.3 Drag reduction

Preliminary results of the current control method are given by calculating the drag ratio term. The drag ratio is defined in section 4.1.7. It compares the shear stresses on the top, un-actuated channel surface to the bottom surface which includes the flow control mechanism and is a measure of the performance of the control algorithm. In a short simulation of 54,000 iterations, the current algorithm results in

an average drag *increase* of 3.2% ±2.3%. This is the active test case result using the optimal *Pamp* = 0.01. These results suggest that either the true optimized settings are not yet determined or that this algorithm is not as effective as the first method in reducing the drag. However, because only 54,000 time steps have been completed, definite conclusions about the effectiveness of this control algorithm should not be considered until after a simulation of sufficient length is performed.

The inactive test case (*Pamp* = 0.0) also provides unexpected results compared to those of the previous control algorithm. Once again the result is an average drag *increase* of 1.3% ±1.5% which barely falls within previous simulation results when including error bars. Since this case is no different than any of the other inactive test cases of the previous control algorithm and those of Lee[26], this suggests that the current simulation may be too short and needs to be run longer for the calculation of more accurate results or it suggests that there is something essentially wrong with the current simulation. Direct comparison with the previous control results and the data of Lee[26] can be made by referring to Table 4.2.

Table 4.3: Average drag reductions for preliminary test run.

Case	*Indep. Realizations*	*Average Drag Reduction**	*Standard error*
Inactive (*Pamp* = 0.0)	8	-1.3%	±1.5%
Active (*Pamp* = 0.01)	9	-3.2%	±2.3%

*a negative number indicates a drag increase.

Chapter 5

Summary and Discussion

The objective of this paper was to study the physics of a physically realizable device having physically realistic control algorithms based on previous numerical and experimental work in the field. A control design was aimed at producing small disturbances very close to the control surface with the goal of quieting the near-wall vortical structures responsible for the regeneration and maintenance of turbulence. As a result, two control algorithms were examined, both based on detection of a quantity at the wall. The first method used detection of $\partial/\partial z(\partial w/\partial y)|_{wall}$. In addition, preliminary results of a second control scheme using the detection of $(\partial u/\partial y)|_{wall}$ were examined. Results after 96,000 iterations using control based on the detection of $\partial/\partial z(\partial w/\partial y)|_{wall}$, show a small 3% ±2.1% reduction in drag was found with actuation turned on while a smaller 1.2% ±1.4% reduction was observed for the inactive actuators. Both results are comparable to the reductions of similar test cases produced by Lee [26]. In those cases, the same immersed boundary technique of

Goldstein *et al.* [19] was used but Lee's control algorithm for detection and actuation was based on that of Choi *et al.* [8]. The main objective of Lee [26] was to dampen the velocity fluctuations just above an actuator by sampling *v*-velocity at a detection area above each actuator, but well within the flow, and apply a continuous blowing/suction to achieve *v*-velocity fluctuation dampening in these regions. Both control designs, that of Lee[26] and the method of $\partial/\partial z(\partial w/\partial y)|_{wall,}$ cover approximately 15% of the total flat surface by using an array of 18 actuators evenly spaced in three rows on the control surface. The small percentage of control surface covered by actuators explains, in part, why the drag reductions achieved in both control designs are small compared to the ~25% drag reductions reported by Choi *et al.* [8]. The method of Choi *et al.* [8] dampened the *v*-velocity fluctuations at *every* grid point on a detection plane and continuous blowing/suction was applied over the *entire* channel surface. Therefore, if the array of actuators used herein were expanded such that the total control surface was covered, the drag reductions reported here may be approximately 6.5 times larger, a reduction comparable to that of Choi *et al.* [8].

A series of parametric studies was performed to obtain an optimal actuation gain and to find the optimal time delay between the detection and actuation events. It was found that an actuation gain which resulted in $v_{rms} \approx 0.28u^*$ in the slot plane provided the greatest drag reduction over a series of short-term tests. This value falls within the root-mean-squared control jet amplitude range of $0.15 - 0.55u^*$ found in

tests by Breuer *et al.* [1] and Lee *et al.* [13]. Surprisingly, the optimal time delay, or lag, between detection and actuation events was found to be not very important for the current control method. This may be due to the typical lengths of the velocity streaks compared to the small range of delays examined in the parametric study. Therefore, a delay corresponding to the time it takes a large scale turbulent structure to convect in the streamwise direction, at an average speed of $10.7u^*$ [14], from the detection location to the leading portion of the actuator was satisfactory.

Furthermore, in an effort to confirm the effectiveness of the control algorithm, a randomization test case was run and compared for the inactive and active cases of the $\partial/\partial z(\partial w/\partial y)|_{wall}$ control. A short-term test was performed in which each pair of sensors was randomly assigned to one of the actuators located somewhere in the array. Results show that, at best, the randomization case matched the slight reductions produced by the inactive test case. The larger drag reductions observed for the active case were not expected in the random case because randomization does not target specific coherent structures. This is reasonably consistent with the results of Dahlburg *et al.* [27] who found no drag reduction from introducing randomized body forces in a turbulent channel flow. Therefore, in examining the random case it was found first, that $\partial/\partial z(\partial w/\partial y)|_{wall}$ control is an effective algorithm and, secondly, that the idea of targeting specific turbulent structures to weaken the near-wall vortical structures responsible for turbulence regeneration works.

In addition, preliminary data were reported using the second control method which detected a $(\partial u/\partial y)|_{wall}$ quantity at the wall. While the optimal actuation gain and time delay have not yet been investigated in fine detail, actuation resulting in $v_{rms} \approx 0.07u^*$ (for $Pamp = 0.01$) with a time delay to match the convection of the detected structure to the leading edge of the actuator ($\Delta t = 259$) were used to gather the preliminary data. The potential for drag reduction using this method, surprisingly, does not match that of the first control method examined. With a 1.3% ±1.5% and 3.2% ±2.3% drag *increase* reported thus far for the inactive and active test cases, it seems that either the current algorithm is not as effective as the first, or that the current simulation has not been run long enough to achieve reliable results.

In conclusion, the current research supports the idea of weakening the near-wall coherent structures of turbulent channel flow through the use of physically practical control devices and algorithms which make use of flow information on the wall. While the reductions reported seem small compared to other the numerical studies, it should be kept in mind that only 15% of the total surface was covered in the current cases. Furthermore, by increasing the actuator population density, which is feasible with current progress in MEMS technology, the potential for a significant drag reduction over a surface remains promising.

Chapter 6

Future Directions

Using the current numerical approach, there are numerous possibilities for further research regarding turbulent boundary layer control. For starters, due to computation limitations at the present time, the current turbulent channel flow simulations were limited to low Reynolds number channel flow. With the use of parallel processing future simulations can be performed to investigate higher Reynolds flow control with the goal of determining whether the same mechanisms targeted in the current runs can be detected and whether a successful control algorithm can be designed.

Future work can also be performed in determining how the actuator coverage area of the control surface is related to the amount of drag reduction achieved. Increasing the coverage area can be done in at least two ways; first, one may want to increase the actuator density by modeling smaller control units (i.e. actuators and their corresponding sensors). With this approach the coverage area may be increased significantly. Secondly, one may modestly increase the coverage area of the

actuators by modifying the arrangement of the array. Another area worth examining is whether neural networking or the self-learning algorithms of Lee *et al*. [13] can be implemented into the current numerical scheme. This work can be performed in order to examine whether such a control algorithm can produce results consistent those of Lee *et al*. [13].

Appendix

Guide for DNS code and modifications

Presented here is a brief guide on how to run the DNS code and an explanation of modifications performed on the code since January 2003. The code is written in FORTRAN (f77) and compiled with an xlf compiler on the IBM Power4 System (*longhorn*). The two versions of the code discussed in this section correspond to the different detection methods used in the current report. The code versions are named: chevron.spanwise.f and chevron.streamwise.f. Descriptions of the various files (input/output) is followed by a brief description of how to run the code; as well as a brief discussion of the modifications to each code. Finally, brief descriptions of other codes used for statistical analysis of the data are provided.

Input Files

fort.10 formatted input file specifying a restarted run (zero for cold start, one for restarted run), number of iterations, physical domain sizes, gain values, data print out frequencies, and cross-sectional output locations

fort.12 restart file with flow field data for channel geometry

fort.48 user supplied geometry of the bottom surface. The file format is in terms of zero (flow driven by a gravity force), one (solid surface), two (slip surface), and four (pumping/ suction membrane)

fort.79 renamed from a fort.78 output file. Used to read in last of *vslotm* variable (measured or detected quantity of control scheme) for a restart. Accounts for the time delay between detection of the quantity and its use in determining the strength of an actuator response

Output Files

fort.22 restart data file which should be renamed fort.12 if used in a restart

fort.30 binary file with fluctuating flow quantities to be used by the analysis program to get the turbulence statistical quantities

fort.50 3D flow field data output for roughly the bottom third on the domain

fort.51 contains flow information at a certain xy plane. File written in subroutine VCW3D and location of z-plane is given in subroutine forcepre, variable *jc*

fort.52 flow information at a certain xz plane. Written in subroutine VCW3D and location of y plane given by variable *kc* in forcepre subroutine

fort.53 flow information at a zy plane specified by variable *xloc* in VCW3D

fort.54 data at a second xz plane specified by variable *kctwo* in VCW3D

fort.55 data at a second zy plane specified by variable *xloctwo* in VCW3D

fort.70 detection information (*vslotm*) at a y plane at channel surface

fort.71 v-velocity information (*vdum*) sampled in center of actuator slot

fort.78 detection information (*vslotg*) at a y plane at channel surface accounts for lag between detection quantity calculation and strength of actuator application in restarted runs

Mpi Launcher Files

paramlist provides directory location and name of program used in run. Useful for running codes in parallel for parametric studies

qrun provides directory location and name of program used in run. Used for running a code in series (no parallel run) and replaces *paramlist*, *launcher*, and *launcher.poe*

launcher must be created with *Makefile* and *launcher.c* by typing, '*make launcher* '

launcher.poe specifies number of processors, memory allocation and number of tasks per processor as well as the directory path to the work directory in which the input files are located for the specific run

makefile used to create *launcher* from *launcher.c* file

Code Compilation

xlf –qrealsize=8 –qdpc=e program_name.f fft_init.f fft_mult.f myfftlib3.f

This is the command to compile code for the IBM Power4 System (*longhorn*). The *qrealsize* and *qdpc* flags are used specifically for the current two codes. *fft_init.f*, *fft_mult.f* and *myfftlib3.f* are needed in addition to the current code to compile correctly. After compilation an *a.out* file is created and can be renamed to *chevron.spanwise.p,* for example, for submission and execution in the *longhorn* system. The executable programs are below:

chevron.spanwise.p

chevron.streamwise.p

Run Submission

The following files must be included in a work directory of the *longhorn* system to submit a run: *fort.10*, *fort.12* (if restarting a run), *fort.48*, *fort.79*, *paramlist*, *launcher*, *launcher.poe*, *chevron.spanwise.p*. *Fort.79* is included only if you have the delay

activated between detection event and actuation like the two current codes above since this file accounts for the lag of information over a restarted run. Note: if running a serial job, *paramlist*, *launcher* and *launcher.poe* are replaced with *qrun* in the list above.

The command to submit a run to the batch queue of the longhorn system and to check queue status are:

llsubmit launcher.poe (parallel run)

or *llsubmit qrun* (serial run)

llq

The output files are created in the same work directory as the input files. Further information on how to use the longhorn system can also be found at www.tacc.utexas.edu.

Modifications to chevron.spanwise.f ($\partial\tau_z/\partial y$ Method)

This section describes the changes made to the above program. The differences between chevron.spanwise.f and chevron.streamwise.f are very small, however the key modifications will be discussed separately. The changes described below are arranged into the different subroutines of the code in which they exist.

Subroutine SETSTUFF

In this subroutine an additional loop (below) was added to the code to allow for reading in *fort.79*. The variable *offset* is read in from the *fort.10* file and specifies the delay of information in terms of number of iterations.

```
do i = 1,offset
      read(79,*)
          itg, vslotg(1,1,itg), vslotg(2,1,itg), …
end do
```

This data array is given the variable *vslotg* and accounts for the delay or lag of detection information upstream of the actuators, where the sensors are located, to the actuator location. The dimension *itg* of the data array is simply a counter to keep track of the relay of information from one run to another. This array exists due to the fact that for a restarted run the first certain number of iterations, depending on the length of the information delay, was calculated in the previous run and were stored in that run as the *fort.78* output file. In order to accurately continue a smooth transition between restart runs this data needs to be read into the new run as *fort.79* and this is done here.

Subroutine VCW3D

A few repeating loops were added to this subroutine to calculate the detected quantity upstream of the actuators. For the current program we detect an approximation of $\partial\tau_z/\partial z$; this approximation is calculated with the following portion of code upstream of each of the eighteen actuators:

```
ktemp = 9
if (it.le.1) ktemp = 8
if (k .eq. ktemp) then
        vslotm(1,1,it) = 0.
        vmeas1 = 0.
        vmeas2 = 0.
        do jj = 1, 33, 32
                do ii= 17, 17
                        vmeas1 = vmeas2
                        vmeas2 = wp(jj,ii)
                end do
        end do
 vslotm(1,1,it) = -(vmeas2-vmeas1)/.388267
```

Except for the first iteration, the values are detected one plane (9th y plane) above the solid flat surface which is what the *ktemp* variable declares. The first iteration is never used in a restarted run because of the inability to calculate a first order time derivative of a velocity needed to calculate the drag ratio. Using the nested do loops you can see that at the 17th i-index (streamwise location upstream of the actuator) two values of *w*-velocity are sampled, the first at the 1st j-index and the second at the 33rd j-index (spanwise locations). The variables *vmeas1* and *vmeas2* store these two *w*-velocity values and use them in the final equation which calculates an approximation of the spanwise gradient of spanwise shear with the hard-wired constant 0.388267 equivalent to the spanwise distance between the 1st j-index and the 33rd j-index (or dz). Note that the dw term (*vmeas2 – vmeas1*) is not divided by the constant dy, equal to the distance between the 8th and 9th y planes. This can be added later, in fact both the dy and dz terms are always constant and therefore only the difference in velocity is important. This portion of the code is repeated for each

actuator and a new variable array *vslotm* is created to store the detected values which will be used to determine the actuation sign and strength of the actuator within the same control unit.

Another modification includes the sampling of *v*-velocity at points over each actuator opening to keep track of the strength of the actuator response. Again a similar loop, with different indices, is repeated so that data from each actuator can be obtained. This is included below with *vdum* being the new variable array. *Vp* is the *v*-component of velocity and *delxm* and *delzm* are cell sizes in the streamwise and spanwise directions. The loop calculates an average *v*-velocity over the area above an actuator opening with 0.1365 equal to the total area over an actuator opening..

```
vdum(1,1,it) = 0.
        do jj = 11, 22
                do ii = 26, 40
                        vdum(1,1,it) = vdum(1,1,it) + vp(jj,ii)*delxm*delzm
                end do
         end do
vdum(1,1,it) = -vdum(1,1,it)/ .1365
```

Output files *fort.70*, *fort.71*, and *fort.78* are also written in this subroutine. The coding for these loops is very basic and will not be discussed further here except that it should be noted that in writing the *fort.78* file, an argument in the *if*-statement allows only the *vslotm* values (which for the next run are read in as *vslotg*) at the very end of the run to be written out.

A final modification to the subroutine is when to use the *vslotg* values read in from a *fort.79* file allowing a smooth transition between restarts and when to use

vslotm values which are calculated in the current run. Again, arguments included in the *if*-statements separate when to use *vslotg* and *vslotm* in the calculation of *wdes2*.

Modifications to chevron.streamwise.f (∂u/∂y Method)

The modifications to this program are very similar to those of chevron.spanwise.f. The changes in the subroutine SETSTUFF that are made in the above section are also made in this program. The main modification to this program occurs in VCW3D and is discussed below.

Subroutine VCW3D

The modifications of this subroutine in the previous code can also be included here. The main difference is the series of *do*-loops which calculate the detected quantity upstream of each actuator. For the current program the upstream quantity of ∂u/∂y is targeted and approximated by sampling *u*-velocity values at the 9th y-plane, upstream of each actuator. The portion of code below is repeated for each actuator.

```
vslotm(1,1,it) = 0.
vmeas1 = 0.
do jj = 9, 24
        vmeas1 = vmeas1 + up(jj,19)
end do
vmeas1 = (vmeas1)/16.
vslotm(1,1,it)= (((vmeas1)/0.030077296)/2.12520561403) – 1
```

The code above initializes the *vslotm* array and *vmeas1* variable before calculating the average of 16 *u*-velocity values upstream of each actuator (in this case the

streamwise location for all of the actuators of the first row is the 19th i-index). The *do*-loop adds *u*-velocity values for points between 9th and 24th j-index for the actuator within its control unit and is divided by the number of samples, 16, immediately after the *do*-loop. Once again, the velocity is the key quantity and division by a constant dy term, which is the difference in heights between the 9th and 8th y-planes, is not important. The values calculated are then stored in the *vslotm* array for use in determining the strength and sign of actuation of the corresponding actuator.

This concludes the discussion of modifications to the two programs described in this report. The modifications are very similar except in the calculation of the targeted detection quantity. In the following section a brief explanation of a few programs designed for statistical analysis is given.

Additional Codes (Statistical)

newpostsimple.c

Used to calculate the running average, average ± one standard deviation, and an autocorrelation of the drag data (*drag ratio* variable). The input file for the program is created by compiling all of the data one wants to calculate statistics on in Tecplot, for example, to create one large data input file (*data.out*). Compilation of the program is done with a c-compiler on the local (b47, b52, arthur, etc.) machines:

cc newpostsimple.c –o newpostsimple.p –lm

This produces newpostsimple.p which can be typed in the prompt as:

newpostsimple.p data.out > results.out

The program will run and output the data into *results.out*. An important quantity that should be examined is the autocorrelation. Plotting it can give an approximation to the *independent realization interval* of the drag results. Together with the total number of iterations and drag ratio printout frequency allows for the calculation of *nsamples* and *period* variables that are needed in the two programs below to accurately calculate the standard error. These two quantities are calculated as follows:

$$\text{nsamples} = \frac{\text{total \# iterations}}{\text{indep. realization interval}}$$

$$\text{period} = \frac{\text{indep. realization interval}}{\text{drag ratio printout frequency}}$$

avg_SD_vdum.f / avg_SD_drag.f

Both of these programs are very similar with the goal of calculating the running average, overall average, standard deviation, standard error bars, etc. of the drag and *vdum* variable, which is the *v*-velocity directly over an actuator opening. The input file for each of the programs is made by compiling all of the data one wants to calculate statistics on in Tecplot, for example, to create one large data input file that needs to be named *fort.25*. The first line in the *fort.25* file should declare the number of samples the file contains. In addition, to calculate the standard error accurately

one must use the results of the newpostsimple.c program to calculate the *period* variable that is hard-wired into each of the current two programs. Note: although *nsamples* is calculated it should always remain hard-wired as 1 in both programs. Once the input file and hard-wired variables have been set the compilation command is as follows and performed on the local (b47, b52, arthur, etc.) machines:

f77 –o avg_SD_drag.p avg_SD_drag.f

Running the program (avg_SD_drag.p) will produce an output file, *fort.26*, with the above statistics calculated.

newpostyallspan.f

This section discusses a post-process program which takes the *fort.30* file as an input file to average consecutive restarted runs so that the data can be collapsed from the 3D computational domain (*fort.90*) down to a single xy plane of data (*fort.91*) and then again down to a single profile (*fort.92*) over the entire length of a simulation. Figure A.1 provides a graphical representation of the above described collapsing of data.

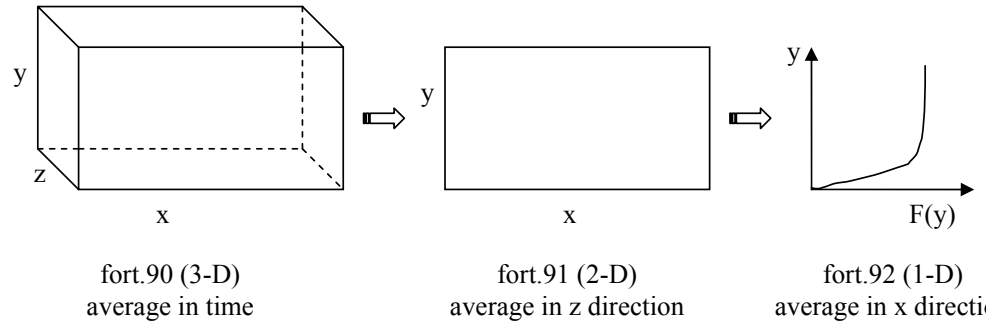


Figure A.1: Graphical representation of the output files for newpostyallspan.f which shows the collapsing of data from a 3-D representation to a single profile.

However, if one is only interested in examination of the 3D data set and any cross-sections of the domain, the data can be stacked or averaged in blocks called master units in the streamwise and spanwise directions by changing the *xs* and *zs* variable within this program. These variables represent the number of cells in each direction on a 3/2 grid that makes up the master unit. For example, this master unit can be arranged such that one can sub-divide the array of 18 actuators and "stack" or average over them to get the average values of the data over only one actuator or one row of actuators similar to that of figure 4.14. Another example is to set the master units (*xs, zs*) to the full domain lengths (*mx, mz*) which will result in no sub-division of the array of actuators and result in figure 4.13. Both of these figures where created by examination of the *fort.90* file and manually created cross-sections using Tecplot.

Two more graphical examples are given in figure A.2 and A.3 to clearly explain master units and how the data is "stacked" or averaged in the *fort.90* file.

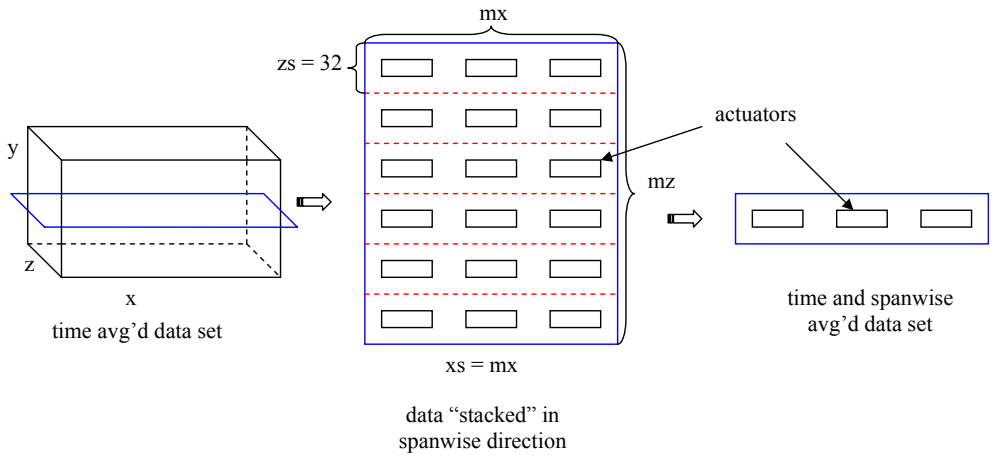


Figure A.2: Graphical representation of the *fort.90* output file for newpostyallspan.f with (*xs*, *zs*) = (*mx*, *32*). This diagram shows the "stacking" or averaging of data that occurs when running the newpostyallspan.f program with master units not equal to the spanwise and streamwise domain maximums (*mx*, *mz*).

In the above example the master units (*xs*, *zs*) are set to (*mx*, *32*). As one can see this sub-divides the *xz* plane into 6 sections along the spanwise direction, having *xs* = *mx* results in no sub-division of data in the streamwise direction. The result is a time and spanwise averaged data set showing a single row of actuators which is found in the *fort.90* file and using Tecplot to manually extract the *xz* planes desired (for example $3l^*$ above the array surface). Figure A.3 shows a similar example only with "stacking" in the streamwise direction, in addition to in the spanwise direction. As one can see the data is sub-divided in both the spanwise and streamwise directions resulting in a data set that is time, spanwise, and streamwise averaged over a single actuator. The desired plane of data (usually only a few wall units above the control surface) is extracted by extracting these planes in the *fort.90* file using Tecplot.

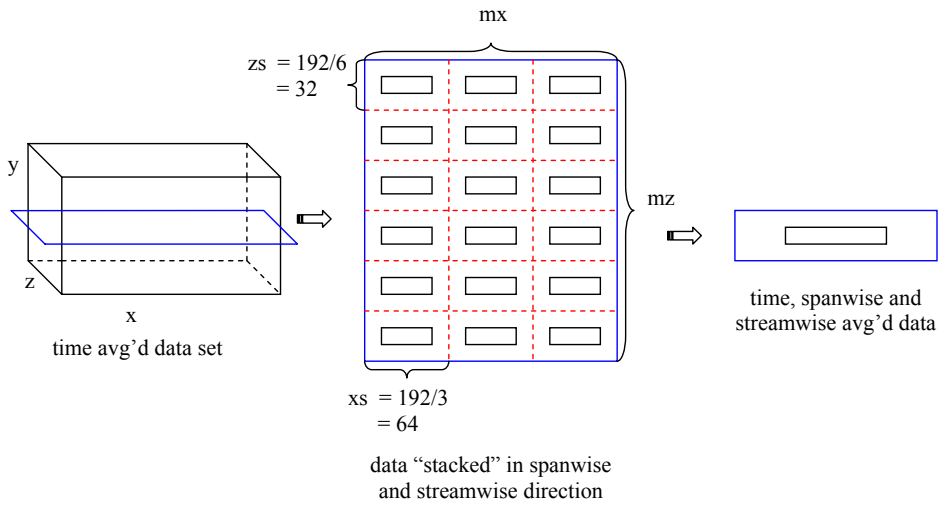


Figure A.3: Graphical representation of the *fort.90* output file for newpostyallspan.f with (*xs*, *zs*) = (*64*, *32*). This diagram shows the "stacking" or averaging of data that occurs when running the newpostyallspan.f program with master units not equal to the spanwise and streamwise domain maximums (*mx*, *mz*).

Compilation of this post-process program is as follows and is performed on the *longhorn* system:

xlf –qrealsize=8 –qdpc=e –q64 newpostyallspan.f fft_init.f fft_mult.f myfftlib3.f

An a.out file results and can be renamed if desired. The program requires consecutive *fort.30* files from the restarted runs along with *fort.10*, which is best shown by an example. Also keep in mind that if "stacking" of the data is desired you must change the master units (*xs*, *zs*) in the program before compilation and execution.

Example: a simulation is run and restarted twice leaving three sets of output data for the entire simulation. In each of the three runs the simulation is run for 6000 iterations and printed every 2000 iterations. For each run a *fort.30* file is created and

must be stored. Let's say the three files are named: *fort.001.30*, *fort.002.30* and *fort.003.30*. Now one wants to perform the post-process program for the entire simulation. The *fort.30* files must be named such that the first is *fort.30* followed by *fort.31*, *fort.32*, etc. Therefore *fort.001.30* is renamed *fort.30*, *fort.002.30* renamed to *fort.31*, and *fort.003.30* to *fort.32*. Now the following files must be included in a single work directory: *a.out*, *fort.10*, *fort.30*, *fort.31*, *fort.32*. Within *fort.10,* three variables must be changed: *nfile*, *nrlz*, *nskip*. Since 6000 divided by the printout frequency of 2000 equals 3, there are 3 realizations per run (*nrlz* should then be set to 9). Since there were three runs performed *nfile* should be set to 3 and *nskip* should always be set to 1. All one needs to do now is run the code by typing a.out at the prompt. The output files are then given as *fort.90*, *fort.91*, and *fort.92*.

newpostyallspan_symm.f

This program is the same to the one above however mirroring or stacking of the data about a centerline is performed. Again manipulating the master units in this program to (*xs=mx, zs=32*), where *mx* is the full domain length, will sub-divide the array of actuators in the 3/2 grid with an additional stacking along a symmetry line. The result is that of figure 4.14. If you look closely at this figure one observes that the symmetry line is along the spanwise center of the actuator. Figure A.4 shows a graphical representation of this additional averaging of the data.

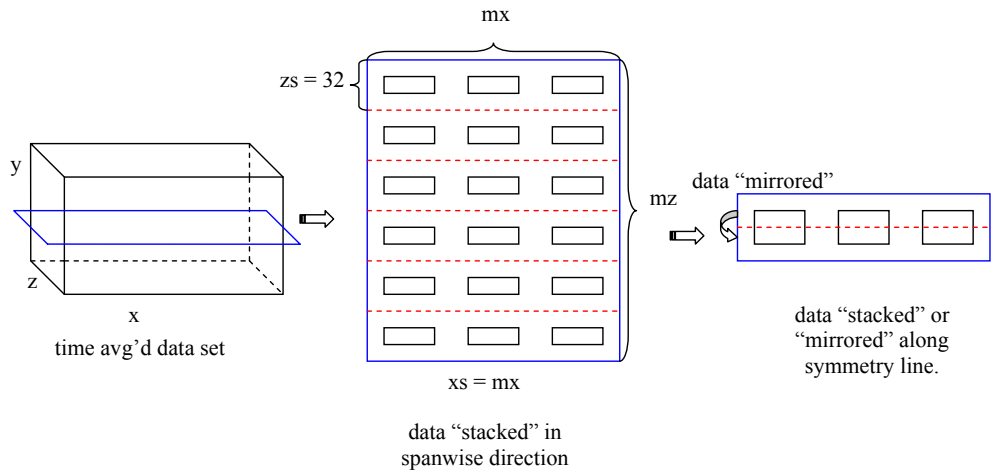


Figure A.4: Graphical representation of the *fort.90* output file for newpostyallspan_sym.f with (*xs*, *zs*) = (*mx*, *32*). This diagram shows the "stacking" or averaging of data that occurs when running the newpostyallspan_sym.f program with master units not equal to the spanwise and streamwise domain maximums (*mx*, *mz*). In addition, a "stacking" or "mirroring" of the data along the centerline of the spanwise master unit is performed.

References

[1] Breuer, K. S., Amonlirdviman, K., and Rathnasingham, R. "Adaptive Feed Forward Control of Turbulent Boundary Layers," AIAA paper 98-1025, 1998.

[2] Jimenez, J. and Pinelli, A., "The autonomous cycle of near-wall turbulence," *Journal of Fluid Mechanics*, 1999, **389**:335-359.

[3] White, F. M., *Viscous Fluid Flow*, 2nd ed., McGraw-Hill Series in Mechanical Engineering, 1991, Chap. 6.

[4] Panton, R., Self-Sustaining Mechanisms of Wall Turbulence, Computational Mechanics Publications, 1997, Chap. 2, 10.

[5] Koumoutsakos, P., Bewley, P., Hammond, E. P., and Moin, P., "Feedback Algorithms for Turbulence Control – Some Recent Developments," AIAA paper 97-2008, 1997.

[6] Goldstein, D. B. and Tuan, T.-C., "Secondary flow induced by riblets," *Journal of Fluid Mechanics*, 1998, **363**:115-151.

[7] Kang, S., and Choi, H., "Active wall motions for skin-friction drag reduction," *Physics of Fluids*, 2000, **12**, No. 12:3301-3304.

[8] Choi, H., Moin, P., and Kim, J., "Active Turbulence Control for Drag Reduction in Wall-Bounded Flows," *Journal of Fluid Mechanics*, 1994, **262**:75-110.

[9] Koumoutsakos, P., "Vorticity flux control for a turbulent channel flow," *Physics of Fluids*, 1999, **11**, No. 2:248-250.

[10] Carlson, H. A. and Lumley, J. L., "Active control in the turbulent wall layer of a minimal flow unit," *Journal of Fluid Mechanics*, 1996, **329**:341.

[11] Lee, C., Kim, J., and Choi, H., "Suboptimal control of turbulent channel flow for drag reduction," *Journal of Fluid Mechanics*, 1998, **358**:245-258.

[12] Endo, T., Kasagi, N., and Suzuki, Y., "Feedback control of wall turbulence with wall deformation," *International Journal of Heat and Fluid Flow*, 2000, **21**:568-575.

[13] Lee, C., Kim, J., Babcock, D., and Goodman, R., "Application of neural networks to turbulence control for drag reduction," *Physics of Fluids*, 1997, **9**, No. 6:1740-1747.

[14] Rathnasingham, R. and Breuer, K. S., "System Identification and Active Control of a Turbulent Boundary Layer," AIAA paper 97-1793, 1997.

[15] Rathnasingham, R. and Breuer, K. S., "Closed-Loop Control of Turbulent Boundary Layers," *Unpublished*, Division of Engineering, Brown University, Providence, RI, 2002.

[16] Rebbeck, H. and Choi, K-S., "Opposition control of near-wall turbulence with a piston-type actuator," *Physics of Fluids*, 2001, **13**, No. 8:2142-2145.

[17] Lew, J., Huang, A., Ho, C-M., Jiang, F., and Tai, Y. C., "Surface Shear Stress Reduction with MEMS Sensors/Actuator in Turbulent Boundary Layers," *Unpublished*, Univ. of California, Los Angeles and Caltech.

[18] Jacobson, S. A., and Reynolds, W. C., "Active control of streamwise vortices and streaks in boundary layers," *Journal of Fluid Mechanics*, 1998, **360**:179-211.

[19] Goldstein, D. B., Handler, R., and Sirovich, L., "Modeling a no-slip flow boundary with an external force field," *Journal of Computational Physics*, 1995, **105**:354-366.

[20] Gad-el-Hak, M., "Interactive Control of Turbulent Boundary Layers: a Futuristic Overview," *AIAA Journal*, 1994, **32.9**:1753-1765.

[21] Lee, C. and Goldstein, D. B., "DNS of Microjets for Turbulent Boundary Layer Control," AIAA paper 2001-1013, 2001.

[22] Lee, C. and Goldstein, D. B., "Simulation of MEMS Suction and Blowing for Turbulent Boundary Layer Control," AIAA paper 2002-2831, 2002.

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

[24] Lee, C. Y. and Goldstein, D. B., "Two-Dimensional Synthetic Jet Simulation," *AIAA Journal*, **40.3**:510-516.

[25] Wu, K. E. and Breuer, K. S., "Dynamics of Synthetic Jet Actuator Arrays for Flow Control," AIAA paper 2003-4257, 2003.

[26] Lee, C. Y., "Direct Numerical Simulation of Microjets for Turbulent Boundary Layer Control," Ph.D. Dissertation, Dept. of Aerospace Engineering, Univ. of Texas at Austin, Austin, TX, 2004.

[27] Dahlburg, R. B., Sandberg, W. C., Handler, R. A, and Sirovich, L., "Direct numerical simulations of drag modifications using randomized force fields ," Proceedings of the international symposium on seawater drag reduction, Newport, RI, July 22-23 1998, pp. 131-133.

Vita

Gerardo Ernesto Colmenero was born in Milwaukee, Wisconsin, on October 15, 1978, the son of Gerardo G. Colmenero and Maria A. Colmenero. After graduating from Cudahy High School, Cudahy, Wisconsin, in 1997, he entered The University of Wisconsin–Madison. During his education at The University of Wisconsin-Madison, he worked as a professional co-op for NASA - Johnson Space Center in Houston over the course of eighteen months. In December 2002 he received his Bachelor of Science with honors in Mechanical Engineering. In January 2003 he then entered the Aerospace Engineering Department at The University of Texas at Austin. His research during his time at The University of Texas was supported by the Air Force Office of Scientific Research under grant number F49620-02-1-0093.

Permanent Address: 3416 E. Armour Avenue

Cudahy, WI 53110

This thesis was typed by the author.