Dynamics of Vortices in Numerically Simulated Turbulent Channel Flow

by

Praveen Kumar Parthasarathy

A Thesis Presented in Partial Fulfillment of the Requirements for the Degree Master of Science

Approved July 2011 by the Graduate Supervisory Committee:

Ronald Adrian, Chair
Huei-Ping Huang
Marcus Herrmann

ARIZONA STATE UNIVERSITY
August 2011
ABSTRACT

The evolution of single hairpin vortices and multiple interacting hairpin vortices are studied in direct numerical simulations of channel flow at Re$_{\tau}$=395. The purpose of this study is to observe the effects of increased Reynolds number and varying initial conditions on the growth of hairpins and the conditions under which single hairpins autogenerate hairpin packets. The hairpin vortices are believed to provide a unified picture of wall turbulence and play an important role in the production of Reynolds shear stress which is directly related to turbulent drag. The structures of the initial three-dimensional vortices are extracted from the two-point spatial correlation of the fully turbulent direct numerical simulation of the velocity field by linear stochastic estimation and embedded in a mean flow having the profile of the fully turbulent flow. The Reynolds number of the present simulation is more than twice that of the Re$_{\tau}$=180 flow from earlier literature and the conditional events used to define the stochastically estimated single vortex initial conditions include a number of new types of events such as quasi-streamwise vorticity and Q4 events. The effects of parameters like strength, asymmetry and position are evaluated and compared with existing results in the literature. This study then attempts to answer questions concerning how vortex mergers produce larger scale structures, a process that may contribute to the growth of length scale with increasing distance from the wall in turbulent wall flows. Multiple vortex interactions are studied in detail.
DEDICATION

To my mother and father.
ACKNOWLEDGMENTS

It gives me great pleasure to remember and thank all the people who have guided me towards the successful completion of this scientific endeavor. Firstly, I would like to sincerely thank Prof. Ronald J. Adrian for his guidance and support during this work. Our discussions have inspired many a late night’s worth of work. I also wish to thank Dr Marcus Hermann for his thoughts and ideas on this work. Valuable inputs from Dr B.J. Balakumar and Dr Kyoungyoun Kim are appreciated. The influence of Dr B.S.V Prasad Patnaik, Mr Ramachandran K.S, Mr Sundaram, Mrs Shobha Raman and Mr Kothandaraman M. is gratefully acknowledged with deep respect.

I would like to thank my parents Geetha and Parthasarathy and my brother Prashanth for their love, affection and wisdom. Talking to them seemed to solve many a problem. This work would not have been possible without them. I owe the deepest gratitude to my cousins Jana, Jayanthi and Karthik. The many conversations we have had were always a source of happiness and inspiration.

I had the pleasure of working with Jon Baltzer, Isaac Ziskin and Stefano Discetti during my stay at our laboratory. Their hardwork and knowledge served as a great source of motivation. I wish to thank Durella Donell for her timely help with the administrative work.

I also would like to thank Dr Gil Speyer, Dr Scott Menor and the entire high performance computing center for their assistance with running my simulations in the university’s super computer.
Last but not the least, I wish to thank my friends who have always stood by me during the highs and lows I have had. They have been a great source of strength and encouragement throughout and our conversations had served as starting points of many a train of thought.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF SYMBOLS / NOMENCLATURE</td>
<td>xi</td>
</tr>
<tr>
<td>Chapter 1</td>
<td>1</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1. Channel flow model</td>
<td>3</td>
</tr>
<tr>
<td>1.1.1 Geometry</td>
<td>3</td>
</tr>
<tr>
<td>1.1.2 Governing Equations</td>
<td>4</td>
</tr>
<tr>
<td>1.2 Numerical Methods</td>
<td>5</td>
</tr>
<tr>
<td>1.2.1 Temporal and Spatial Discretization</td>
<td>5</td>
</tr>
<tr>
<td>1.2.2 Boundary conditions</td>
<td>6</td>
</tr>
<tr>
<td>1.2.3 Solution procedure</td>
<td>7</td>
</tr>
<tr>
<td>1.2.4 Grid Independence study</td>
<td>7</td>
</tr>
<tr>
<td>Chapter 2</td>
<td>11</td>
</tr>
<tr>
<td>METHODOLOGY</td>
<td>11</td>
</tr>
<tr>
<td>2.1 Turbulent mean properties</td>
<td>11</td>
</tr>
<tr>
<td>2.2 Correlation</td>
<td>16</td>
</tr>
<tr>
<td>2.3 Joint Probability Distribution functions</td>
<td>18</td>
</tr>
</tbody>
</table>
2.4 Linear Stochastic estimation ................................................................. 21

2.5 Vortex visualization ........................................................................... 22

Chapter 3 ................................................................................................. 27

SINGLE VORTEX EVOLUTION ................................................................. 27

3.1 Effect of strength ............................................................................... 28

3.2 Effect of y-normal position ............................................................... 32

3.3 Effect of asymmetry .......................................................................... 34

3.4 Evolution into a fully turbulent flow ................................................. 38

Chapter 4 ................................................................................................. 42

MULTIPLE VORTEX INTERACTION .......................................................... 42

4.1 Streamwise interaction between 2 Q2 events ................................... 42

4.1.1 Interaction between 2 Q2 events having the same strength ......... 42

4.1.2 Interaction between 2 Q2 events having the different strengths: ... 43

4.2 Interaction between 3 Q2 events ................................................. 45

4.3 Spanwise growth of vortices ............................................................ 47

4.4 Interaction between Q2 and Q4 events ............................................ 49

4.5 Interaction between vortices at different y+ locations .................... 50

4.6 Interaction between vortices in a staggered arrangement ............... 52

Chapter 5 ................................................................................................. 54
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>The threshold $\lambda_{ci}$ for the 3 different grids at various $t^+$. $\lambda_{ci}$ is the complex eigenvalue of the velocity gradient tensor and $t^+$ is the non-dimensional time computed from equation 1.2.</td>
<td>8</td>
</tr>
<tr>
<td>2.1</td>
<td>Quadrant IV events which maximize the product of Reynolds stress and Probability of occurrence [Moin, Adrian and Kim (1987)]. $u_m$ and $v_m$ denote the maximum values of fluctuating $u$ and $v$ velocities; $\sigma_u$ and $\sigma_v$ denote the variances in the $u$ and $v$ direction.</td>
<td>18</td>
</tr>
<tr>
<td>2.2</td>
<td>Quadrant II events which maximize the product of Reynolds stress and Probability of occurrence [Moin, Adrian and Kim (1987)].</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>Initial conditions used in this study for single vortex evolution. All computations were done at $Re_\tau=395$ for 128 x 129 x 128 grid.</td>
<td>23</td>
</tr>
<tr>
<td>2.4</td>
<td>Initial conditions used in this study for vortex interactions. All computations were done at $Re_\tau=395$ for 128 x 129 x 128 grid.</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>The time ($t^+$) taken for the vortex to disappear when a sub-critical strength is used for computation. These computations were done at $y^+=46.6$. $x^+$ denotes the non-dimensionalized streamwise spacing.</td>
<td>31</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 The channel geometry. The x, y and z coordinates show the streamwise, wall-normal and spanwise directions. The streamwise and spanwise directions are respectively $2\pi h$ and $\pi h$ long which is 2480.6 and 1240.9 in wall units. The 2 infinite parallel walls are spaced 2$h$ apart (790 wall units).</td>
<td>4</td>
</tr>
<tr>
<td>1.2 The plot between the threshold $\lambda_{ci}$ and $t^+$ for the 3 different grids. $\lambda_{ci}$ and $t^+$ are defined in equations 1.2 and 1.3 respectively and denote the complex eigen value of the velocity gradient tensor and the non-dimensional time.</td>
<td>8</td>
</tr>
<tr>
<td>1.3 The evolved hairpin vortex structure at $t^+ = 250$ for (a) 96 x 97 x 96 grid; (b) 128 x 129 x 128 grid; and (c) 256 x 257 x 256 grid. (b) and (c) are qualitatively similar from the above figure. The initial condition is shown in equation 1.4.</td>
<td>9</td>
</tr>
<tr>
<td>1.4 Plots of streamwise correlation vs the streamwise spacing. The correlation is defined in equation 1.5. $R_{uu}$ is the streamwise correlation at $(\Delta x^+, y^+=37.9, y'^+=37.9, \Delta z^+=0)$ and is non-dimensionalized with the correlation at zero streamwise spacing $(\Delta x^+=0)$. These agree closely with the results of Moser, Kim and Mansour’s (1999) computation on a finer grid (256x257x256).</td>
<td>10</td>
</tr>
</tbody>
</table>

2.1 The mean velocity profile for the channel flow plotted with the law of the wall. The superscript + indicates a non-dimensional quantity scaled by the wall
variables; \( y^+ = y u^*/v \) is the viscous height of the channel where \( v \) is the kinematic viscosity and \( u^* = (\tau_w / \rho)^{1/2} \) is the wall shear velocity.............. 11

2.2 Vertical profiles of the resolvable mean Reynolds shear stress \( \vec{u}v \). \( Re_\tau=395 \). The grid adopted is 128 x 129 x 128. The stress was validated with the results of Moser, Kim and Mansour (1999) as shown in figure 2.4 a. ........ 12

2.3 Plots of the root mean square components of velocity against the wall-normal distance normalized with \( Re_\tau=395 \). Validation with the \( Re_\tau=395 \) result of Moser, Kim and Mansour (1999) is shown in figure 2.4 b........................... 13

2.4 Validation of computations with the \( Re_\tau=395 \) results of Moser, Kim and Mansour (1999) (a) The magnitude of reynolds stress obtained from the Reynolds stress tensor as a function of the non-dimensionalized wall-normal distance (\( y^+ \)) upto \( y^+=395 \). (b) ___: \( u_{\text{rms}} \), ___: \( v_{\text{rms}} \), ___: \( w_{\text{rms}} \).° represents Moser et al.’s results for a finer (256 x 257 x 256) grid.......................... 14

2.5 Plots of (a) Skewness and (b) Flatness for the channel flow data. \( u' \), \( v' \) and \( w' \) represent the velocity components in the streamwise, normal and spanwise directions respectively. \( S(w') \) and \( F(w') \) are predominantly 0 and 3 respectively. .......................................................... 16

2.6 Plots of velocity correlations as a function of the normalized (a) streamwise distance; \( R_{uu}, R_{vv}, R_{ww} \) are computed at \( (\Delta x^+, y^+=11.8, y'^+=11.8, \Delta z^+=0) \) and is non-dimensionalized by the correlation values at \( \) and (b) spanwise distance. .......................................................... 17
2.7 Joint probability distributions at various $y^+$ values (a) $y^+=11.8$; (b) $y^+=46.6$; (c) $y^+=66.6$; (d) $y^+=109$; (e) $y^+=217$; (f) $y^+=395$................................. 19

2.8 Plots of (a) $y^+$ vs 4th quadrant angles (in degrees) and (b) $y^+$ vs 2nd quadrant angles (in degrees). The plots are validated with the results of Moin, Adrian and Kim (1987). The 4th and 2nd quadrant angles were obtained from table 2.1 and 2.2 respectively. These angles make the maximum contribution to the Reynolds stress tensor. The present computations were done at $Re_\tau=395$ and Moin et al’s results were at $Re_\tau=180$..................................................... 20

2.9 The angle of the Q2 vector as a function of distance from the wall obtained from Kim, Moin and Moser (1987). Inset: Method of defining the Q2 event $(u_m,v_m,0)$........................................................................................................ 21

2.10 List of figures showing initial vortex shapes. Mathematical representation shown in table 2.3 ................................................................. 24

2.11 List of figures showing initial vortex shapes. Mathematical representation shown in table 2.4 ................................................................. 26

3.1 Generation of secondary hairpin vortices depends on the strength of initial vortical structures and location of the event vector used to extract the initial vortical structure. (●) Case with new hairpins. (○) Case without new hairpins [Kim, Sung and Adrian (2008)]................................................................. 29

3.2 Vortex evolution at $t^+=150$ for different strengths (a) $\alpha=2$; (b) $\alpha=2.5$; (c) $\alpha=3$; (d) $\alpha=3.5$. The initial velocity field specified was $u=\alpha(u_m,v_m,0)$ where $u_m$ and
were obtained from the joint probability density function and were taken
to be (-1.6,1.4,0)………………………………………………………………………………….. 30

3.3 A comparison between the lengths of the eddy (x^+) at various t^+ values........ 31

3.4 Evolution of the hairpin vortex at various values of y^+; (a) y^+ = 11.8; (b) y^+ =
46.6; (c) y^+ = 66.6; (d) y^+ = 217; the initial vortex was located at the center
of the xz plane…………………………………………………………………………… 34

3.5 Effect of asymmetry on vortex evolution (a) β = 0.2; (b) β=0.4; (c) β=0.5; (d)
β=0.6; (e) β=0.8; α = 2.5 was used for all the computations. The initial field
specified was u=(-4,3.5,0) for all the cases considered. ............................... 36

3.6 Growth of a single vortex into a fully turbulent field. α = 2.5 was used for all
the computation. The initial field specified was u=(-4,3.5,0). (a) The
evolution at t^+=400; (b) The evolution at t^+=750; (c) The evolution at
t^+=1000; (d) The evolution at t^+=1250; (e) The evolution at t^+=1500………40

3.7 Fully turbulent channel flow at t^+=2000……………………………………..41

4.1 Evolution of 2 Q2 events initially separated by x^+ = 100 units. α=2.5; β=0;
The initial condition was considered at the center of the xz plane and at
y^+=46.6. (a) The initial vortex obtained from linear stochastic estimation. A
and B are Q2 events having the initial velocity vectors (-4,3.5,0) based on the
joint probability density function (b) The evolution structure at t^+=150 (c)
The evolution structure at t^+= 375. ............................................................... 43

4.2 The effect of varying the strength of the vortex. Strength is denoted by α
which was defined earlier in the study (chapter 3.1) The vortex A is stronger
than vortex B. (a) $\alpha_A=2.5; \alpha_B=2; \beta=0$; Hence, the 1st event vector (vortex A) is ................................................................. 44

4.3 The vortex B is stronger than vortex A. (a) The initial vortex at $t^+=0$, $\alpha_A=2; \alpha_B=2.5; \beta=0$; Hence, event vector for vortex A is (-3.2,2.8,0) and the event vector for vortex B is (-4,3.5,0) (b) Evolution after $t^+=150$......................... 44

4.4 Evolution of 3 Q2 events initially separated by $x^+=100$ units. $\alpha=2.5; \beta=0$; The initial condition was considered at the center of the xz plane and at $y^+=46.6$. (a) The initial vortex obtained from linear stochastic estimation (b) The evolution structure at $t^+=150$ (c) The evolution structure at $t^+=375$. .... 45

4.5 The effect of varying the strength of the vortices. (a) The initial vortex at $t^+=0$. $\alpha_A=3; \alpha_B=2.5; \alpha_C=2, \beta=0$; (b) Evolution after $t^+=150$ ............................... 46

4.6 (a) The initial vortex at $t^+=0$. $\alpha_A=2; \alpha_B=2.5; \alpha_C=3, \beta=0$; (b) Evolution after $t^+=150$ ........................................................................................................... 47

4.7 The vortex structure at $t^+=750$ for a single Q2 event evolution at $(x^+=0,y^+=46.6, z^+=0)$................................................................................. 47

4.8 Evolution of 2 spanwise vortices separated by $z^+=100$ at (a) $t^+=25; t^+=225$; (b) $t^+=350; t^+=500$. Both vortices have strength $\alpha=2.5$ and no inclination to the z axis ($\beta=0$) ................................................................................................................. 48

4.9 Evolution of a Q2 event and a Q4 event together. The vortices are initially separated by $x^+=100$ units. $\alpha=2.5; \beta=0$; The initial condition was considered at the center of the xz plane and at $y^+=46.6$. (a) Q4-Q2 combination where A(i) represents the Q4 vortex (4,-3.5,0) at $(x^+=0,y^+=46.6,$
z^+=0) and A(ii) represents the Q2 vortex (-4,3.5,0) at (x^+=100,y^+=46.6, z^+=0); (b) Q2-Q4 combination where A(i) represents the Q2 vortex and A(ii)
represents the Q4 vortex. B represents the structure at t^+=250 and C is the
structure at t^+=500.......................................................................................... 50

4.10 (a) 2 vortices separated by 100 x^+ units at t^+=0. Vortex A is at y^+=46.6 and
vortex B is at y^+=11.8; (b) the vortex structure at t^+=50; (c) The vortex
structure at t^+=50 when the vortex B is absent; ............................................. 51

4.11 (a) Evolution of the single vortex at y^+=46.6 and (b) the evolution of the dual
vortices at t^+=400. The similarity in structure leads us to believe that vortex
B doesn’t have a major role to play in the evolution. ................................. 51

4.12 Evolution of 5 Q2 events placed in a staggered arrangement. The schematic
diagram of the arrangement is shown in figure 4.13. α=2.5; β=0; All the
initial vortices are at y^+=46.6. (a) The evolution structure at t^+=25 (b) The
evolution structure at t^+=175 (c) The evolution structure at t^+=375 (d) The
evolution structure at t^+=500..............................................................53

4.13 A schematic arrangement of staggered vortices in channel flow............54
# LIST OF SYMBOLS

Symbol

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{ik}$</td>
<td>linear coefficients for stochastic estimation</td>
</tr>
<tr>
<td>$h$</td>
<td>half-channel height</td>
</tr>
<tr>
<td>$i$</td>
<td>unit imaginary number</td>
</tr>
<tr>
<td>$j$</td>
<td>component index</td>
</tr>
<tr>
<td>$k$</td>
<td>component index</td>
</tr>
<tr>
<td>$k_x$</td>
<td>streamwise wave number</td>
</tr>
<tr>
<td>$k_y$</td>
<td>wall-normal spectral mode number</td>
</tr>
<tr>
<td>$k_z$</td>
<td>spanwise wave number</td>
</tr>
<tr>
<td>$l$</td>
<td>component index</td>
</tr>
<tr>
<td>$L_x$</td>
<td>length of the computational box in the streamwise direction</td>
</tr>
<tr>
<td>$L_z$</td>
<td>length of the computational box in the streamwise direction</td>
</tr>
<tr>
<td>$p$</td>
<td>fluctuating pressure</td>
</tr>
<tr>
<td>$r_x$</td>
<td>distance between x and x’</td>
</tr>
<tr>
<td>$R_{jl}$</td>
<td>two-point, second order spatial correlation tensor</td>
</tr>
<tr>
<td>$Re_{\tau}$</td>
<td>Reynolds number based on wall friction velocity and half-channel height time</td>
</tr>
<tr>
<td>$u$</td>
<td>fluctuating velocity component in the streamwise direction</td>
</tr>
<tr>
<td>$U$</td>
<td>mean streamwise velocity</td>
</tr>
<tr>
<td>$x$</td>
<td>streamwise position</td>
</tr>
<tr>
<td>$y$</td>
<td>wall-normal position</td>
</tr>
</tbody>
</table>
\( z \) spanwise position

Greek Symbols

\( \gamma_{II} \) the angle between the event vector and the negative streamwise axis for quadrant II events

\( \gamma_{IV} \) the angle between the event vector and the negative streamwise axis for quadrant IV events

\( \nu \) kinematic viscosity

\( \rho \) density

\( \tau_w \) shear stress evaluated at the wall

Superscripts

\( + \) denotes that the quantity is non-dimensionalized with viscous scales fluctuating quantity

Other Notation

\( <f> \) denotes ensemble average of the quantity \( f \)
Chapter 1

INTRODUCTION

One of the most fundamental properties of wall turbulence is that the length scale, defined in various ways increases with distance from the wall. Starting with the mean spanwise spacing of low speed streaks at the wall, the length scale grows slightly through the buffer layer and then grows linearly throughout the logarithmic layer. This study attempts to answer the question on whether the vortex mergers would produce self similar vortices or a new class of structures. Channel flow was chosen since both experimental and theoretical investigations of complex turbulence interactions near the wall can be carried out.

Various studies by Bandhopadhyay (1980) and Smith (1984) ascertain the presence of vortex packets in the turbulent boundary layer. In this study, the vortex packet is shown to evolve out of single and multiple hairpin vortices generated through linear stochastic estimation. Hydrogen bubble and dye visualization by Haidari and Smith (1994) and inviscid models by Smith et al. (1991) attempted to address the natural formation of the vortex packets more closely. Although processes like vortex stretching and tilting were described by the inviscid models, a complete picture on vortex breakup and reconnection were not considered. Zhou, Adrian and Balachandar (1996) and Zhou, Adrian, Balachandar and Kendall (1999) performed direct numerical simulations in channel flow at $\text{Re}_c=180$, and found that a single hairpin vortex is capable of creating successive upstream hairpins, providing that the strength of the first hairpin exceeds a critical value. This process, called ‘autogeneration’ leads to the
formation of a packet of hairpins travelling together, with the first hairpin being tallest and the last hairpin being shortest. The first hairpin generates a secondary hairpin, the secondary generates a tertiary, and so on for succeeding generations. If the initial hairpin is symmetric about a wall-normal plane through its middle, the resulting packet is also symmetric. But, if there is asymmetry, the hairpins assume the shape of a cane, and the packet structure tends to alternate from right-handed to left-handed canes. If the initial hairpin contains noise, the autogeneration leads to chaotic packets [Adrian (2007)]. Kim and Adrian (1999) proposed that the organization of hairpin vortices into packets and the interactions between these packets are characteristic features of wall turbulence that explain many observations like the large amount of streamwise kinetic energy residing in very long streamwise wavelengths. The formation of new streamwise vortices and the characteristic angles of inclined hairpins were further explained by Adrian, Meinhart and Tomkins (2000).

Hairpin vortex packets play an important role in the production of the Reynolds shear stress, which is directly related to the turbulent drag. Ganapathisubramani, Longmire and Marusic (2003) showed that about 25% of the total production of Reynolds shear stress in the log layer of turbulent boundary layers is attributed to vortex packets. In a hairpin packet, the total turbulent Reynolds stress can be thought of as arising from the incoherent component and the coherent component. The incoherent component is the sum of the momentum transfers by each individual vortex and the coherent component is the sum of the momentum transfers produced by vortex interactions. In addition
to the experimental observation of hairpin packets in instantaneous flow fields using particle image velocimetry (PIV), statistical evidence of hairpin packets has been reported by Christensen and Adrian (2001) and Hambleton, Hutchins and Marusic (2006). Zhou, Adrian and Balachandar (1996) used the direct Numerical simulation of the Navier-stokes equation to study the evolution of a hairpin vortex in a unidirectional mean flow obtained from the low-Reynolds number turbulent channel flow of Kim, Moin and Moser (1987). Their approach is adopted in the present study. The initial vortex structure without the presence of the other eddies (i.e. in a clean turbulent mean flow environment) has made it possible to visualize clearly the auto generation of new hairpin vortices.

1.1. Channel flow model

1.1.1 Geometry

The channel is composed of two infinite parallel walls, spaced a distance 2h apart. The streamwise and spanwise directions are $2\pi h$ and $\pi h$ respectively (2480.6 and 1240.9 in wall units). The computation is carried out with 2113536 grid points (128 x 129 x 128, in x, y, z) for a Reynolds number of 395 based on the wall shear velocity $u^*$. The model assumes that the flow is periodic in the plane of the walls. Thus, a finite sized section can be used to model the infinite channel. The section used in this study is shown in Figure 1.1.
Figure 1.1 The channel geometry. The x, y and z coordinates show the streamwise, wall-normal and spanwise directions. The streamwise and spanwise directions are respectively $2\pi h$ and $\pi h$ long which is 2480.6 and 1240.9 in wall units. The 2 infinite parallel walls are spaced 2h apart (790 wall units).

With this computational domain, the grid spacing’s in the streamwise and spanwise directions are respectively $\Delta x^+ \approx 19.37$ and $\Delta z^+ \approx 9.69$ in wall units. Non-uniform meshes are used in the normal direction with $y_j = \cos \theta_j$, for $\theta_j = (j-1)\pi/(N-1)$, $j = 1, 2, ..., N$. Here $N$ is the number of grid points in the $y$-direction.

### 1.1.2 Governing Equations

The initial turbulent flow field is evolved in time by solving the Navier-Stokes equation along with the incompressibility condition. The equations used are the same as used in the thesis by Kendall (1992). Written in non-dimensional form, the equations can be represented as

$$\frac{\partial \tilde{u}}{\partial \tilde{x}} + \frac{\partial \tilde{v}}{\partial \tilde{y}} + \frac{\partial \tilde{w}}{\partial \tilde{z}} = 0$$

(1.1a)
In the governing equations, the channel half-height \( h \) is used as the length scale. Wall friction velocity \( u^* = (u(\partial u/\partial y)_{y=\pm h})^{1/2} \) is used as the velocity scale. The characteristic pressure and time scales are \( \rho u^*^2 \) and \( h/ u^* \) respectively. This scaling results in the non-dimensional parameter of Reynolds number based on friction velocity, \( Re_t = u^* h/\nu \).

1.2 Numerical Methods

1.2.1 Temporal and Spatial Discretization

Fourier expansions are used as part of the spectral collocation methodology for the periodic directions and a Chebyshev expansion is used for the non-periodic wall normal direction with Gauss-Lobatto points for spatial discretization. A time-splitting technique was employed for the decoupling of the pressure computations in the time advancement of the flow field. At each time step, the following discrete equations are solved:

\[
\frac{\partial \tilde{u}}{\partial t} + \frac{\partial \tilde{u}}{\partial x} + \frac{\partial \tilde{v}}{\partial y} + \frac{\partial \tilde{w}}{\partial z} = -\frac{\partial \tilde{p}}{\partial x} + \frac{1}{Re_t} \left( \frac{\partial^2 \tilde{u}}{\partial y^2} + \frac{\partial^2 \tilde{u}}{\partial z^2} + \frac{\partial^2 \tilde{u}}{\partial x^2} \right)
\]

\[
\frac{\partial \tilde{v}}{\partial t} + \frac{\partial \tilde{u}}{\partial x} + \frac{\partial \tilde{v}}{\partial y} + \frac{\partial \tilde{w}}{\partial z} = -\frac{\partial \tilde{p}}{\partial y} + \frac{1}{Re_t} \left( \frac{\partial^2 \tilde{v}}{\partial y^2} + \frac{\partial^2 \tilde{v}}{\partial z^2} + \frac{\partial^2 \tilde{v}}{\partial x^2} \right)
\]

\[
\frac{\partial \tilde{w}}{\partial t} + \frac{\partial \tilde{u}}{\partial x} + \frac{\partial \tilde{v}}{\partial y} + \frac{\partial \tilde{w}}{\partial z} = -\frac{\partial \tilde{p}}{\partial z} + \frac{1}{Re_t} \left( \frac{\partial^2 \tilde{w}}{\partial y^2} + \frac{\partial^2 \tilde{w}}{\partial z^2} + \frac{\partial^2 \tilde{w}}{\partial x^2} \right)
\]

\[
\tilde{u} = \frac{u}{u^*}
\]

\[
v^* = \frac{v}{u}
\]

\[
w^* = \frac{w}{u}
\]
step, first an intermediate velocity field is computed with only the advection and
diffusion effects taken into account. This intermediate velocity field is not
divergence free. In the second step, an appropriate pressure is computed by
solving a Poisson equation for pressure, based on which a pressure correction is
applied to the intermediate velocity field to make it divergence free. Here, we
employ a third order Runge Kutta scheme for the advection term and an implicit
Crank Nicholson scheme for the diffusion term. The pressure effect is considered
to be fully implicit in order to guarantee zero divergence at the end of the full
timestep. The details of the numerical procedure used in this channel-flow
simulation are elaborated in Kendall (1992).

1.2.2. Boundary conditions

The periodic conditions in the streamwise and spanwise directions are
automatically satisfied by the use of fourier expansions. The no slip and the
incompressibility conditions cannot be satisfied simultaneously because the time
splitting scheme separates the momentum equation into two parts. In order to
minimize the slip, a proper choice of the intermediate boundary condition must be
made. The boundary condition for pressure is specified during the pressure-
poisson step. It can be shown that a self-consistent, pure Neumann condition will
allow slip velocity to be minimized.
1.2.3. Solution procedure

The Helmholtz equations for the three components of velocity are solved for each combination of horizontal Wave numbers to solve for the entire flow field. The equations are listed in Kendall’s thesis (1992).

1.2.4. Grid Independence study

Grid refinement study was done for three different grids: 96x97x96, 128x129x128 (present grid) and 256x257x256. From figure 1.2, 128x129x128 grid is seen to be optimum for this computation since there is not much difference in \( \lambda_{ci} \) with the 256x257x256 grid. \( \lambda_{ci} \), referred to as the swirling strength is the complex eigenvalue of the velocity gradient tensor (\( \mathbf{D} = \nabla \mathbf{u} \)) and it is a good measure of the vortex structure since it is frame independent and discriminates against shear layers which have vorticity but no swirling motion [Chong, Perry and Cantwell (1990), Chakraborty, Balachandar and Adrian (2006)]. \( t^+ \) is the non-dimensional time and is computed in equation 1.2. The change in time, \( dt \) is taken to be 1.25e-04 and the number of iterations is typically 10,000 although the value was increased for some computations to study the physics at a later time.

\[
t^+ = \frac{dt \times \text{Number of iterations}}{(h/ u^*)} \tag{1.2}
\]

The swirling strength is obtained from the characteristic equation of the velocity gradient tensor which is given by

\[
\lambda^3 + P\lambda^2 + Q\lambda + R = 0 \tag{1.3}
\]

Where, \( P = -\text{div} \mathbf{u} \); \( Q = \frac{1}{2}[P^2 - \text{tr(DD)}] \); and \( R = -\text{det(D)} \)
Table 1.1 The threshold $\lambda_{ci}$ for the 3 different grids at various $t^+$. $\lambda_{ci}$ is the complex eigen value of the velocity gradient tensor and $t^+$ is the non-dimensional time computed from equation 1.2.

<table>
<thead>
<tr>
<th>$t^+$</th>
<th>Maximum $\lambda_{ci}$ (Grid: 96 x 97 x 96)</th>
<th>Maximum $\lambda_{ci}$ (Grid: 128 x 129 x 128)</th>
<th>Maximum $\lambda_{ci}$ (Grid: 256 x 257 x 256)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>16.1136</td>
<td>60.6377</td>
<td>54.937</td>
</tr>
<tr>
<td>50</td>
<td>16.3566</td>
<td>65.3198</td>
<td>61.9359</td>
</tr>
<tr>
<td>100</td>
<td>18.5097</td>
<td>75.223</td>
<td>80.9264</td>
</tr>
<tr>
<td>150</td>
<td>21.9789</td>
<td>53.2664</td>
<td>55.0533</td>
</tr>
<tr>
<td>200</td>
<td>28.3971</td>
<td>54.1164</td>
<td>51.6953</td>
</tr>
<tr>
<td>250</td>
<td>26.3907</td>
<td>50.3841</td>
<td>50.4075</td>
</tr>
<tr>
<td>300</td>
<td>25.4552</td>
<td>51.8427</td>
<td>47.8462</td>
</tr>
<tr>
<td>350</td>
<td>25.7557</td>
<td>49.6056</td>
<td>48.9717</td>
</tr>
<tr>
<td>400</td>
<td>25.9998</td>
<td>43.6902</td>
<td>47.2204</td>
</tr>
<tr>
<td>450</td>
<td>25.747</td>
<td>39.759</td>
<td>44.3926</td>
</tr>
<tr>
<td>500</td>
<td>25.3215</td>
<td>33.3265</td>
<td>39.6914</td>
</tr>
</tbody>
</table>

Figure 1.2 The plot between the threshold $\lambda_{ci}$ and $t^+$ for the 3 different grids. $\lambda_{ci}$ and $t^+$ are defined in equations 1.2 and 1.3 respectively and denote the complex eigen value of the velocity gradient tensor and the non-dimensional time.

The initial condition for figure 1.2 and 1.3 is defined as

$$u(x,t=0) = \langle u(x) | u'(y_m^+ = 46.6) = 3(u_m, v_m, 0) \rangle$$  \hspace{1cm} (1.4)
Figure 1.3 The evolved hairpin vortex structure at $t^+ = 250$ for (a) $96 \times 97 \times 96$ grid; (b) $128 \times 129 \times 128$ grid; and (c) $256 \times 257 \times 256$ grid. (b) and (c) are qualitatively similar from the above figure. The initial condition is shown in equation 1.4.
The 1D streamwise correlations plotted as a function of the non-dimensionalized streamwise spacing ($\Delta x^+$) further shows the adequacy of grid (figure 1.5).

![Figure 1.4 Plots of streamwise correlation vs the streamwise spacing. The correlation is defined in equation 1.5. $R_{uu}$ is the streamwise correlation at $(\Delta x^+, y^+=37.9, y'^+=37.9, \Delta z^+=0)$ and is non-dimensionalized with the correlation at zero streamwise spacing ($\Delta x^+=0$). These agree closely with the results of Moser, Kim and Mansour’s (1999) computation on a finer grid (256x257x256).](image)

A detailed discussion of the properties of the initial velocity fields and the initial structure extraction using linear stochastic estimation is given in chapter 2. In chapter 3, the evolution of a single hairpin vortex in the channel flow is discussed and the results are compared with literature. Multiple vortex interactions are studied in chapter 4. Finally, in chapter 5, the conclusions obtained from this research program are summarized and some recommendations for future work in the area of conditional vortex dynamics are given.
Chapter 2

METHODOLOGY

2.1 Turbulent mean properties

Figure 2.1 The mean velocity profile for the channel flow plotted with the law of the wall. The superscript + indicates a non-dimensional quantity scaled by the wall variables; \( y^+ = y u^* / \nu \) is the viscous height of the channel where \( \nu \) is the kinematic viscosity and \( u^* = (\tau_w / \rho )^{1/2} \) is the wall shear velocity.

Starting from the initial velocity field, the governing equations were integrated forward in time until the numerical solutions reached statistically steady states. The calculations were considered to be complete when the time-averaged turbulence quantities became stationary. The profile of the mean velocity non-dimensionalized by the wall-shear velocity is shown in figure 2.1. The collapse of
the mean-velocity profiles corresponding to the upper and lower half of the channel indicates the adequacy of the sample taken here for the average.

Figure 2.2 Vertical profiles of the resolvable mean Reynolds shear stress $\overline{uv}$. $Re_e=395$. The grid adopted is 128 x 129 x 128. The stress was validated with the results of Moser, Kim and Mansour (1999) as shown in figure 2.4 a.

The profile in figure 2.2 indicate that the average Reynolds shear-stress profile has attained the equilibrium shape that balances the downstream mean pressure gradient in the regions away from the walls. In the vicinity of the walls, the viscous stresses are significant, and they, together with the total Reynolds stress, balance the mean pressure gradient. The symmetry of the profile about the channel centre line indicates that the total averaging time and statistical sample are adequate. The other characteristic properties of the flow, like the root mean square velocity were also plotted.
Figure 2.3 Plots of the root mean square components of velocity against the wall-normal distance normalized with $Re_\tau=395$. Validation with the $Re_\tau=395$ result of Moser, Kim and Mansour (1999) is shown in figure 2.4 b
Figure 2.4 Validation of computations with the $Re_t=395$ results of Moser, Kim and Mansour (1999) (a) The magnitude of Reynolds stress obtained from the Reynolds stress tensor as a function of the non-dimensionalized wall-normal distance ($y^+$) upto $y^+=395$. (b) ___: $u_{\text{rms}}$, ___: $v_{\text{rms}}$, ___: $w_{\text{rms}}$. ° represents Moser et al.’s results for a finer (256 x 257 x 256) grid.

Once again, the symmetry of the calculated turbulence intensities about the centre line of the channel indicates that the total averaging time was sufficient for an adequate statistical sample. 2\textsuperscript{nd} order statistics like skewness and flatness which are important parameters in a turbulent flow [Davidson (2007)] are defined as

\begin{align}
S &= \frac{\overline{u_i^3}}{\overline{u_i^2}^{3/2}} \\
F &= \frac{\overline{u_i^4}}{\overline{u_i^2}^2}
\end{align}

(i=1,2,3; no summation) (2.1a, 2.1b)

The flatness factors of all the velocity components reach their maxima at the wall. This indicates that in the vicinity of the wall, the turbulence is highly intermittent. Throughout an appreciable portion of the channel cross-section, $F(w')$ and $S(w')$
are approximately equal to three and zero respectively. These values correspond to the flatness and skewness factors of a Gaussian distribution. Near the wall, $S(u')$ is positive, whereas away from the wall it is negative. This indicates that near the wall the large-amplitude $u$-fluctuations are primarily due to arrival of high-speed fluid from regions away from the wall. On the other hand, away from the wall the large-amplitude $u$-fluctuations are most probably associated with low-speed fluid leaving the wall region. This is encouraging considering the significant contribution of small-scale turbulence to these quantities and the difficulties associated with their measurements.
Figure 2.5 Plots of (a) Skewness and (b) Flatness for the channel flow data. $u'$, $v'$ and $w'$ represent the velocity components in the streamwise, normal and spanwise directions respectively. $S(w')$ and $F(w')$ are predominantly 0 and 3 respectively.

2.2 Correlation

In order to perform a linear estimate of the velocity field given a set of velocity conditions, the full two-point, second-order spatial correlation tensor, equation (2.2) is needed. This tensor was calculated using equation 2.2.

$$R_{ji}(x,x') = <u_j(x) u_i(x')>$$  \hspace{1cm} (2.2)
Figure 2.6 Plots of velocity correlations as a function of the normalized (a) streamwise distance; (b) spanwise distance. $R_{uu}$, $R_{vv}$, $R_{ww}$ are computed at $(\Delta x^+, y^+=11.8, y'^+=11.8, \Delta z^+=0)$ and is non-dimensionalized by the correlation values at $\Delta x^+=0$.

These profiles show that, the longitudinal correlation in the streamwise direction extends over much longer distances than do all other correlations. The slow decay of $R_{uu}$ with increasing $x^+$ indicates that near the wall, the eddies are highly
elongated in the streamwise direction. On the other hand, the profiles of figure 2.4 (b) shows that the spanwise extent of turbulence structures near the wall is much smaller than for those away from the wall. It hence appears that, near the walls the computed flow field consists of elongated streaky structures.

2.3 Joint Probability Distribution functions

The streamwise and wall normal velocity components of the event vector are chosen based on their contribution to mean Reynolds shear stress. The events, \( u(x,t=0) \), studied in this work are chosen such that the product of the simultaneous Reynolds stress and the probability of occurrence of events are maximized[(Moin, Adrian and Kim (1987)]. Second (Q2) and fourth quadrant (Q4) events are studied.

Table 2.1 Quadrant IV events which maximize the product of Reynolds stress and Probability of occurrence [Moin, Adrian and Kim (1987)]. \( u_m \) and \( v_m \) denote the maximum values of fluctuating \( u \) and \( v \) velocities; \( \sigma_u \) and \( \sigma_v \) denote the variances in the \( u \) and \( v \) direction.

<table>
<thead>
<tr>
<th>Q4 event</th>
<th>( \frac{u_m}{\sigma_u} )</th>
<th>( \frac{v_m}{\sigma_v} )</th>
<th>( \frac{u_m v_m}{\sigma_u \sigma_v} )</th>
<th>( \tan^{-1}(\frac{u_m v_m}{\sigma_u \sigma_v}) ) degrees</th>
<th>( y^+ )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>-0.8</td>
<td>-0.58</td>
<td>-33.67</td>
<td>11.8</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>-0.78</td>
<td>-44.98</td>
<td>46.6</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>-1</td>
<td>-0.69</td>
<td>-39.78</td>
<td>66.6</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>-1.2</td>
<td>-0.78</td>
<td>-44.98</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>-0.78</td>
<td>-44.98</td>
<td>217</td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>-0.8</td>
<td>-0.78</td>
<td>-44.98</td>
<td>395</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2 Quadrant II events which maximize the product of Reynolds stress and Probability of occurrence [Moin, Adrian and Kim (1987)].

<table>
<thead>
<tr>
<th>Q2 event</th>
<th>( \frac{u_m}{\sigma_u} )</th>
<th>( \frac{v_m}{\sigma_v} )</th>
<th>( \frac{u_m v_m}{\sigma_u \sigma_v} )</th>
<th>( \tan^{-1}(\frac{u_m v_m}{\sigma_u \sigma_v}) ) degrees</th>
<th>( y^+ )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.4</td>
<td>0.8</td>
<td>-0.51</td>
<td>-80.25</td>
<td>11.8</td>
<td></td>
</tr>
<tr>
<td>-1.6</td>
<td>1.4</td>
<td>-0.71</td>
<td>-91.63</td>
<td>46.6</td>
<td></td>
</tr>
<tr>
<td>-1.4</td>
<td>1.4</td>
<td>-0.78</td>
<td>-80.18</td>
<td>66.6</td>
<td></td>
</tr>
<tr>
<td>-1.4</td>
<td>1.2</td>
<td>-0.70</td>
<td>-80.18</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>-1.4</td>
<td>1.4</td>
<td>-0.78</td>
<td>-80.18</td>
<td>217</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>-0.78</td>
<td>-57.27</td>
<td>395</td>
<td></td>
</tr>
</tbody>
</table>
All the following plots are contour plots of $<u'v'>$ probability density function at various values of $y^+$. 

Figure 2.7 Joint probability distributions at various $y^+$ values (a) $y^+$ = 11.8; (b) $y^+$ = 46.6; (c) $y^+$ = 66.6; (d) $y^+$ = 109; (e) $y^+$ = 217; (f) $y^+$ = 395
Figure 2.8  Plots of (a) $y^+$ vs 4th quadrant angles (in degrees) and (b) $y^+$ and $y/h$ vs 2nd quadrant angles (in degrees). The plots are validated with the results of Moin, Adrian and Kim (1987). The 4th and 2nd quadrant angles were obtained from table 2.1 and 2.2 respectively. These angles make the maximum contribution to the Reynolds stress tensor. The present computations were done at $Re_\tau=395$ and Moin et al’s results were at $Re_\tau=180$.
As is seen from figure 2.8, the profiles for $Re_{\tau}=395$ agree well with $Re_{\tau}=180$ [Moin, Adrian and Kim (1987)]. But there is deviation away from the wall ($y^+>100$). The abrupt change in the flow angle which occurs in the buffer layer indicates transition from streamwise oriented wall layer structures to hairpin vortices characterizing the outer layer.

![Figure 2.9 The angle of the Q2 vector as a function of distance from the wall obtained from Kim, Moin and Moser (1987). Inset: Method of defining the Q2 event ($u_m, v_m, 0$)](image)

2.4 Linear Stochastic estimation

Stochastic estimation is a simple procedure by which conditional averages are approximated in terms of unconditional correlation functions (Moin, Adrian and Kim). Linear stochastic estimation is accomplished by expressing the conditional average as a linear function of its data and solving a set of linear algebraic equations for the expansion coefficients. The initial condition consists of a
conditional vortex or a set of conditional vortices superposed onto a turbulent mean velocity profile. The conditional vortex is evaluated using Linear Stochastic estimation. The estimation procedure is briefed in Zhou, Adrian, Balachandar and Kendall (1999) and is described in detail in the appendix at the end of the current study. The choice of a symmetric Q2 event vector results in a vortical structure that resembles a near-wall quasi-streamwise vortex pair when the event is specified close to the wall and resembles a hairpin vortex when the event is specified sufficiently far away from the wall [Moin, Adrian and Kim (1987)]. The linear estimate of the conditional average \( <u(x',t)|u(x,t)> \) is calculated from equation (2.3) where \( A_{jk} \) are the estimation coefficients. For each value of the component \( j \), the \( A_{jk} \) are determined by solving the 3x3 linear algebraic equations shown in equation (2.4). The location in the homogenous directions, \( x \) and \( z \), may be selected arbitrarily and each estimate is evaluated for a given value of \( y \) as a function of the distance \( r = x' - x \).

\[
\begin{align*}
\phi_j(x',t) &= A_{jk}(x',x) \phi_k(x,t) \\
R_{kl}(x,x') A_{jk}(x,x') &= R_{lj}(x,x') = R_{lj}(r,y) \\
&j,k,l = 1,2,3
\end{align*}
\]

By virtue of being extracted from the correlation tensor, the initial structure has length scales, shape and vorticity consistent with eddies that occur in the fully turbulent channel flow.

2.5 Vortex visualization

According to Zhou, Adrian and Balachandar (1996), a vortex usually refers to a tube-like structure with persistent and coherent rotation about its spine. Robinson
(1991) definition of a vortex explains the inadequacy of mathematical quantities like helicity and vorticity to characterize a vortex. On the other hand, a number of techniques for the identification of vortices have been proposed. Although a variety of techniques have been used in the past, the method of Chong, Perry and Cantwell (1990) is used in the current study due to the advantages which include frame independence and the display of shear layers which have vorticity but no swirling motion. The choice of $\lambda_{ci}$ for this study was made so that the various vortical structures would be easily identifiable with minimal background noise, eliminating sensitivity dependence.

Table 2.3 Initial conditions used in this study for single vortex evolution. All computations were done at $Re_\tau=395$ for 128 x 129 x 128 grid.

<table>
<thead>
<tr>
<th>Run</th>
<th>X=(x+,y+,z+)</th>
<th>u=(u,v,w)</th>
<th>Movie location folder (on DVD)</th>
<th>Figure number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0,46.6,0)</td>
<td>(-1.6,1.4,0)</td>
<td>Strength=1</td>
<td>1a</td>
</tr>
<tr>
<td>2</td>
<td>(0,46.6,0)</td>
<td>(-2.175,0)</td>
<td>Strength=1.25</td>
<td>1b</td>
</tr>
<tr>
<td>3</td>
<td>(0,46.6,0)</td>
<td>(-2.4,2.1,0)</td>
<td>Strength=1.5</td>
<td>1c</td>
</tr>
<tr>
<td>4</td>
<td>(0,46.6,0)</td>
<td>(-3.2,2.8,0)</td>
<td>Strength=2</td>
<td>1d</td>
</tr>
<tr>
<td>5</td>
<td>(0,46.6,0)</td>
<td>(-4.3,5,0)</td>
<td>Strength=2.5</td>
<td>1e</td>
</tr>
<tr>
<td>6</td>
<td>(0,46.6,0)</td>
<td>(-4.8,4.2,0)</td>
<td>Strength=3</td>
<td>1f</td>
</tr>
<tr>
<td>7</td>
<td>(0,46.6,0)</td>
<td>(-6.4,5.6,0)</td>
<td>Strength=4</td>
<td>1g</td>
</tr>
<tr>
<td>8</td>
<td>(0.118,0)</td>
<td>(-4.3,5,0)</td>
<td>y+=11.8</td>
<td>2a</td>
</tr>
<tr>
<td>9</td>
<td>(0.66,0)</td>
<td>(-4.3,5,0)</td>
<td>y+=66.6</td>
<td>2b</td>
</tr>
<tr>
<td>10</td>
<td>(0.217,0)</td>
<td>(-4.3,5,0)</td>
<td>y+=217</td>
<td>2c</td>
</tr>
<tr>
<td>11</td>
<td>(0.395,0)</td>
<td>(-4.3,5,0)</td>
<td>y+=395</td>
<td>2d</td>
</tr>
<tr>
<td>12</td>
<td>(0.466,0)</td>
<td>(-4,-3.5,0)</td>
<td>Q4</td>
<td>3a</td>
</tr>
<tr>
<td>13</td>
<td>(0.46,0)</td>
<td>(-4.3,5,0)</td>
<td>Beta=0.2</td>
<td>4a</td>
</tr>
<tr>
<td>14</td>
<td>(0.46,0)</td>
<td>(-4.3,5,0)</td>
<td>Beta=0.4</td>
<td>4b</td>
</tr>
<tr>
<td>15</td>
<td>(0.46,0)</td>
<td>(-4.3,5,0)</td>
<td>Beta=0.5</td>
<td>4c</td>
</tr>
<tr>
<td>16</td>
<td>(0.46,0)</td>
<td>(-4.3,5,0)</td>
<td>Beta=0.6</td>
<td>4d</td>
</tr>
<tr>
<td>17</td>
<td>(0.46,0)</td>
<td>(-4.3,5,0)</td>
<td>Beta=0.8</td>
<td>4e</td>
</tr>
<tr>
<td>18</td>
<td>(0.46,0)</td>
<td>(-4,0,0)</td>
<td>u00</td>
<td>5a</td>
</tr>
<tr>
<td>19</td>
<td>(0.46,0)</td>
<td>(0,3.5,0)</td>
<td>0v0</td>
<td>5b</td>
</tr>
</tbody>
</table>
Figure 2.10 List of figures showing initial vortex shapes. Mathematical representation shown in table 2.3.
Table 2.4 Initial conditions used in this study for vortex interactions. All computations were done at $Re_{\tau}=395$ for 128 x 129 x 128 grid.

<table>
<thead>
<tr>
<th>Run</th>
<th>$X=(x,y,z)$</th>
<th>$u=(u,v,w)$</th>
<th>Movie location (on DVD)</th>
<th>Figure number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0,46.6,0) (100,46.6,0)</td>
<td>(-4.3.5,0) (-4.3.5,0)</td>
<td>Streamwise vortex interaction</td>
<td>7a</td>
</tr>
<tr>
<td>2</td>
<td>(0,46.6,0) (100,46.6,0)</td>
<td>(-4.3.5,0) (-3.2.2,8,0)</td>
<td>Decreasing strength</td>
<td>7b</td>
</tr>
<tr>
<td>3</td>
<td>(0,46.6,0) (100,46.6,0)</td>
<td>(-4.3.5,0) (-4.3.5,0)</td>
<td>Increasing strength</td>
<td>7c</td>
</tr>
<tr>
<td>4</td>
<td>(0,46.6,0) (0.46.6,100)</td>
<td>(-4.3.5,0) (-4.3.5,0)</td>
<td>Spanwise vortex interaction</td>
<td>8a</td>
</tr>
<tr>
<td>5</td>
<td>(0,46.6,0) (100,46.6,0) (200,46.6,0)</td>
<td>(-4.3.5,0) (-4.3.5,0)</td>
<td>3 vortices/same strength</td>
<td>9a</td>
</tr>
<tr>
<td>6</td>
<td>(0,46.6,0) (100,46.6,0) (200,46.6,0)</td>
<td>(-4.8.4,2,0) (-4.3.5,0) (-3.2.2,8,0)</td>
<td>3 vortices/decreasing strength</td>
<td>9b</td>
</tr>
<tr>
<td>7</td>
<td>(0,46.6,0) (100,46.6,0) (200,46.6,0)</td>
<td>(-3.2.2,8,0) (-4.3.5,0) (-4.8.4,2,0)</td>
<td>3 vortices/increasing strength</td>
<td>9c</td>
</tr>
<tr>
<td>8</td>
<td>(0,46.6,0) (100,46.6,0)</td>
<td>(-4.3.5,0) (-4.3.5,0)</td>
<td>21_11</td>
<td>10a</td>
</tr>
<tr>
<td>9</td>
<td>(0,46.6,0) (100,46.6,0)</td>
<td>(-4.3.5,0) (-4.3.5,0)</td>
<td>1Q2Q4</td>
<td>11a</td>
</tr>
<tr>
<td>10</td>
<td>(0,46.6,0) (100,46.6,0)</td>
<td>(4.3.5,0) (4.3.5,0)</td>
<td>1Q4Q2</td>
<td>11b</td>
</tr>
<tr>
<td>11</td>
<td>(0,46.6,0) (0.46.6,100) (0.46.6,200) (0.46.6,50) (0.46.6,150)</td>
<td>(-4.3.5,0) (-4.3.5,0) (-4.3.5,0) (-4.3.5,0)</td>
<td>Staggered</td>
<td>12a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 a</td>
<td>7 b</td>
<td>7 c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 a</td>
<td>9 b</td>
<td>9 c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 a</td>
<td>11 b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.11 List of figures showing initial vortex shapes. Mathematical representation shown in table 2.4.
Chapter 3

SINGLE VORTEX EVOLUTION

Zhou, Adrian, Balachandar and Kendall (1999) studied the evolution of a symmetric pair of quasistreamwise vortical structures extracted from the two-point correlation tensor of turbulent channel flow data by linear stochastic estimation procedure. The initial structure evolves into a hairpin-like vortical structure which can, in turn, generate streamwise vortices, thus providing a mechanism for continual regeneration of new vortices. It is recognized that the strength of the initial structure can play an important role, especially in the nonlinear stages of the evolution. Therefore, the effect of strength on vortex evolution is considered in section 3.1. Also, in the present study, the wall normal location, $y^+$ of the event vector will be varied from near the boundary to the middle of the channel (section 3.2). The symmetric event vector is specified as $u = \alpha u_m$, $v = \alpha v_m$ and $w = 0$, where the multiplicative factor $\alpha$ referred as ‘strength’ of the initial structure, is varied from 1.0 to 3.5. Zhou et al. (1999) showed that asymmetric initial vortices grow more rapidly than symmetric ones and hence are likely to be the most common form found in natural wall turbulence. The effect of asymmetry for various values of $\beta$ is shown in section 3.3. Section 3.4 discusses the evolution of a single vortex into a fully turbulent field.
3.1 Effect of strength

Kim, Sung and Adrian (2008) examined the autogeneration process by which new hairpin vortices are created from a sufficiently strong hairpin vortex, leading to the formation of a hairpin packet. It is observed that while stronger initial vortices result in the formation of a hairpin packet, weaker initial vortical structures, which live long and maintain their integrity, do not participate in the autogeneration of additional hairpins. Owing to the linear nature of the estimation procedure, the entire velocity field of the initial structure scales linearly with $\alpha$. As the strength of the initial event vector $\alpha$ is changed, the initial structure always rolls-up into a hairpin vortex, but its strength and accordingly its subsequent evolution differs. The main effect is on the length of the resulting hairpin vortex along the streamwise direction. The formation process of the primary hairpin vortex remains the same qualitatively. Whereas the initial structure evolves into an $\Omega$-shaped primary vortex, irrespective of its initial strength $\alpha$, and initial location $y^+$, the autogeneration of secondary and tertiary vortices is quite sensitive to the amplitude. From the following figure, it appears that the threshold amplitude reaches a minimum for an initial location $y^+$ of around 30.
Figure 3.1 Generation of secondary hairpin vortices depends on the strength of initial vortical structures and location of the event vector used to extract the initial vortical structure. (●) Case with new hairpins. (∘) Case without new hairpins [Kim, Sung and Adrian (2008)].

Computations were done to see if the downstream vortex affects the upstream vortex in autogeneration. From figure 3.1, the threshold for auto-generation for $\text{Re}_t=180$ is between 0.5 and 1, though there is no auto-generation evident at $\alpha=1$ for $\text{Re}_t=395$. Auto-generation for $\text{Re}_t=395$ exists between $\alpha=1.25$ and $\alpha=1.5$. Figure 3.2 shows the hairpin structure at $t^*=150$ for various strengths.
Figure 3.2 Vortex evolution at $t^+=150$ for different strengths (a) $\alpha=2$; (b) $\alpha=2.5$; (c) $\alpha=3$; (d) $\alpha=3.5$. The initial velocity field specified was $u=\alpha(u_m,v_m,0)$ where $u_m$ and $v_m$ were obtained from the joint probability density function and were taken to be (-1.6, 1.4, 0)
Even though the growth of the vortices tends to be qualitatively similar for all strengths greater than the threshold strength, the disturbances (or the tongue) in the downstream side of the primary vortex are more pronounced as we increase the strength. These disturbances can be considered as numerical errors and are hence more visible as we increase the values of fluctuating u and v velocities.

Table 3.1 The time ($t^+$) taken for the vortex to disappear when a sub-critical strength is used for computation. These computations were done at $y^+ = 46.6$. $x^+$ denotes the non-dimensionalized streamwise spacing.

<table>
<thead>
<tr>
<th>$t^+$</th>
<th>$x^+$ (strength = 1)</th>
<th>$x^+$ (strength = 1.25)</th>
<th>$x^+$ (strength = 1.5)</th>
<th>$x^+$ (strength = 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>360</td>
</tr>
<tr>
<td>50</td>
<td>240</td>
<td>280</td>
<td>320</td>
<td>560</td>
</tr>
<tr>
<td>150</td>
<td></td>
<td>240</td>
<td>400</td>
<td>1280</td>
</tr>
<tr>
<td>175</td>
<td></td>
<td>120</td>
<td>360</td>
<td>2380</td>
</tr>
<tr>
<td>200</td>
<td></td>
<td>80</td>
<td>280</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td></td>
<td></td>
<td>440</td>
<td></td>
</tr>
<tr>
<td>650</td>
<td></td>
<td></td>
<td>680</td>
<td></td>
</tr>
<tr>
<td>800</td>
<td></td>
<td></td>
<td>280</td>
<td></td>
</tr>
<tr>
<td>900</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.3 A comparison between the lengths of the eddy ($x^+$) at various $t^+$ values.
The hairpin vortex at alpha = 1 disappears very quickly as can be seen from figure 3.3. Increasing the strength makes the length of the hairpin grow faster.

3.2 Effect of y-normal position

The choice of a symmetric Q2 event vector results in a vortical structure that resembles a near-wall quasi-streamwise vortex pair when the event is specified close to the wall and resembles a hairpin vortex when the event is specified sufficiently far away from the wall [Moin, Adrian and Kim (1987)]. It can be observed that there exists a bridge of vorticity across the two streamwise vortices at the point where the event vector is specified. The strength of the bridge is weak when the event vector is close to the wall but is relatively stronger when the event vector is farther away from the wall. The average inclination of the initial structure decreases (or increases) as the y-location of the event vector is lowered (or raised), but the spanwise separation at the upstream end remains at about 100 viscous wall units approximately independent of y+. This is consistent with the accepted mean low-speed streak spacing of about 100 viscous wall units in the near-wall region. The location of the spanwise bridge is slightly upstream of the downstream tip of the quasi-streamwise vortices. In other words, the quasi-streamwise vortices extend slightly beyond their spanwise bridge. The spanwise bridge becomes stronger as the location of the event vector, y+ increases and the initial structure resembles more closely a hairpin vortex.

The presence of an optimum distance from the wall for the initial structure can be explained as followed. The optimum distance is a balance between self- and
mutual-induced motion of the quasi-streamwise vortex legs which tends to lift-up and curl back the vortices and the influence of mean shear which stretches along the streamwise direction and intensifies the vortices. Very close to the wall, viscous effects are also important. The enhanced viscous effects result in an increase in the threshold amplitude for initial vortices starting very close to the boundary. Away from the wall, the induced motion is determined by the strength of the vortex structure and streamwise stretching by the mean shear. With increasing distance from the wall, the mean shear rapidly reduces, thereby decreasing the intensification of the initial vortex structure by stretching. Thus, an initial hairpin vortex farther away from the boundary needs to be of sufficiently higher strength to generate subsequent hairpin vortices.
Figure 3.4 Evolution of the hairpin vortex at various values of $y^+$; (a) $y^+ = 11.8$; (b) $y^+ = 46.6$; (c) $y^+ = 66.6$; (d) $y^+ = 217$; the initial vortex was located at the center of the xz plane.

3.3 Effect of asymmetry

The streamwise alignment of the hairpins is the result of the spanwise symmetric nature of the initial vortex structure. Perfect symmetry however cannot be expected and the hairpins are not usually observed to possess two counter-rotating vortex legs of equal strength. The effect of asymmetry on the initial vortical structure evolution and its development into a hairpin packet has been studied here. Asymmetry was introduced in the initial vortical structure with an asymmetric event in the stochastic estimation procedure. The magnitude of the event vector was kept constant to maintain the initial vortex strength, while the
spanwise component of the event vector was increased from zero at the expense of the u and v components. As the $\beta$ increases, the strength of the event vector is still the same.

\[
\begin{align*}
    u &= u_m(1-\beta^2)^{1/2} \\
    v &= v_m(1-\beta^2)^{1/2} \\
    w &= \beta^*(u^2 + v^2)^{1/2}
\end{align*}
\]

where $\beta$ is the asymmetry parameter which measures the strength of asymmetry. For $\beta = 0$ there is no asymmetry and the initial vortex structure is the same as that shown in figure 6(a).
Figure 3.5 Effect of asymmetry on vortex evolution (a) $\beta = 0.2$; (b) $\beta=0.4$; (c) $\beta=0.5$; (d) $\beta=0.6$; (e) $\beta=0.8$; $\alpha = 2.5$ was used for all the computations. The initial field specified was $u=(-4,3.5,0)$ for all the cases considered.
Vortical structure corresponding to an asymmetry parameter of $\beta = 0.2$ is initially considered. The resulting initial structure has a pair of quasi-streamwise legs connected by a weak spanwise bridge at the downstream end, but one of the quasi-streamwise legs is much stronger, higher, and longer than the other. The influence of asymmetry on the overall evolution of the hairpin structures remains negligibly small though for $\beta = 0.2$. The initial structure has developed into a primary hairpin followed by the generation of secondary and downstream hairpins. The resulting hairpin packet is nearly symmetric and it closely resembles the hairpin packet generated under symmetric initial conditions. Thus, the mechanisms responsible for autogeneration of new hairpin vortices leading to the formation of a hairpin packet remain largely unaffected by small asymmetry in the initial development.

With sufficiently strong asymmetry in the initial event vector, the effects can be distinguished in the initial structure as well as in the evolution. The effects of $\beta = 0.4, 0.5, 0.6$ and 0.8 are compared.

For $t^* = 150$ and beta=0.5, in addition to the primary hairpin, secondary and tertiary hairpin-like structures can also be seen. The right-hand leg of the secondary hairpin can be seen, while the other quasi-streamwise vortex leg is so weak that it is not seen. On the other hand, in the case of the tertiary hairpin only the left-hand quasi-streamwise leg is strong and visible. Therefore, the secondary and tertiary hairpins resemble the asymmetric one-sided cane- or hook-like hairpin vortices referred in literature. Robinson (1991) pointed out that the preferred arrangement for hairpin vortices in a turbulent boundary layer is to be asymmetric and one-
sided. These spanwise asymmetric one-sided hairpins are also known as `canes' [Guezennec & Choi (1989)]. The present results suggest that experimentally observed asymmetry is possibly due to the influence of local spanwise velocity. These cane-like secondary and tertiary structures at $t^+ = 150$ are clearly visible in and can be compared with the corresponding symmetric case with initial event vector of $\alpha = 2.5$ specified at $y^+ = 46.6$. For the symmetric case, the streamwise distance between the primary and secondary hairpins was found to be 340 viscous units. In the asymmetric case the streamwise distance between the primary and secondary and between the secondary and tertiary hairpin heads is about 220 and 165 viscous wall units, respectively. These streamwise separations compare better with the experimental measurements of Meinhart, Adrian and Tomkins (1999) who observed the spacing to be around 150 wall units. Furthermore, in the asymmetric case the formation of tertiary hairpin is nearly complete by $t^+ = 150$. In the symmetric case the tertiary hairpin has not even begun to form by this time.

In general, it is observed that asymmetry aids in the formation of subsidiary hairpins and the initial threshold amplitude for the formation of secondary and tertiary hairpins is found to be lower with asymmetry. Under asymmetry, the new hairpins form in rapid succession and their streamwise separation is smaller, and hence better compare with the experiments.

3.4 Evolution into a fully turbulent flow

Computations were done to study the vortex evolution into a fully turbulent flow. The linear stochastic estimate at $y^+ = 46.6$ was used as the initial condition.
Figure 3.6 Growth of a single vortex into a fully turbulent field. $\alpha = 2.5$ was used for all the computation. The initial field specified was $u=(-4,3.5,0)$. (a) The evolution at $t^+=400$; (b) The evolution at $t^+=750$; (c) The evolution at $t^+=1000$; (d) The evolution at $t^+=1250$; (e) The evolution at $t^+=1500$.

The single vortex at $y^+=46.6$ auto-generates into the structure in figure 3.6 (a) at $t^+=400$. These vortices then start growing spanwise apart from growing in height, (figure 3.6 (b)) eventually leading to the complex feature in figure 3.6 (c). This
repeated spanwise interaction and auto-generation results in the structure in figure 3.6 (e) where a chain of vortices on the top is evident. The flow ultimately becomes fully turbulent and occupies the entire channel at around $t^+=2000$ (figure 3.7).

Figure 3.7 Fully turbulent channel flow at $t^+=2000$. 
Chapter 4

MULTIPLE VORTEX INTERACTION

Vortex interactions are important to study since they make understanding on a turbulent field much easier. Since hairpins typically occur in packets, understanding how certain distinct arrangements of vortices evolve helps understand how the entire packet would evolve. In this study, the distinct arrangements like 2 Q2 events, 3 Q2 events and combination of Q2 and Q4 events are studied. Strength plays an important part in this study since the vortices might gain or lose velocity during the process of evolution. Hence, variation of strength for multiple vortex interaction is studied in detail.

4.1 Streamwise interaction between 2 Q2 events

4.1.1 Interaction between 2 Q2 events having the same strength
Figure 4.1 Evolution of 2 Q2 events initially separated by $x^+ = 100$ units. $\alpha=2.5$; $\beta=0$; The initial condition was considered at the center of the $xz$ plane and at $y^+=46.6$. (a) The initial vortex obtained from linear stochastic estimation. A and B are Q2 events having the initial velocity vectors (-4,3.5,0) based on the joint probability density function (b) The evolution structure at $t^+=150$ (c) The evolution structure at $t^+=375$.

4.1.2 Interaction between 2 Q2 events having the different strengths:

Case I:
Figure 4.1 The effect of varying the strength of the vortex. Strength is denoted by $\alpha$ which was defined earlier in the study (chapter 3.1). The vortex A is stronger than vortex B. (a) $\alpha_A=2.5; \alpha_B=2; \beta=0$; Hence, the 1st event vector (vortex A) is $(-4,3.5,0)$ and the vortex B event vector is $(-3.2,2.8,0)$ (b) Evolution after $t' = 150$

Case II:

Figure 4.2 The vortex B is stronger than vortex A. (a) The initial vortex at $t'=0$, $\alpha_A=2; \alpha_B=2.5; \beta=0$; Hence, event vector for vortex A is $(-3.2,2.8,0)$ and the event vector for vortex B is $(-4,3.5,0)$ (b) Evolution after $t' = 150$

A stronger vortex moves slower than the weaker vortex and hence in case I (figure 4.2), the vortices A and B are separated while they evolve. In case II
(figure 4.3) though, vortex A catches up with vortex B and interacts with it earlier than figure 4.2. This can explain the differences in structure at $t^+=150$ (figures 4.2 (b) and 4.3 (b)). Due to the same reason, the vortex combination in figure 4.2 fills up the length of the channel faster than the weaker-stronger case. It is difficult to quantify the interaction processes due to the non-linearity of the problem.

4.2 Interaction between 3 Q2 events:

Figure 4.3 Evolution of 3 Q2 events initially separated by $x^+=100$ units. $\alpha=2.5$; $\beta=0$; The initial condition was considered at the center of the xz plane and at $y^+=46.6$. (a) The initial vortex obtained from linear stochastic estimation (b) The evolution structure at $t^+=150$ (c) The evolution structure at $t^+=375$. 

45
Figure 4.4 The effect of varying the strength of the vortices. (a) The initial vortex at $t^+=0$. $\alpha_A=3$; $\alpha_B=2.5$; $\alpha_C=2$, $\beta=0$; (b) Evolution after $t^+=150$
As explained in section 4.2, the stronger vortex moves slower than the weaker vortex. The interaction can be explained better if the non-linearity in the problem is mathematically modeled.

4.3 Spanwise growth of vortices

Figure 4.7 The vortex structure at $t^+=750$ for a single Q2 event evolution at $(x^+=0, y^+=46.6, z^+=0)$. 
From figure 4.7, it is evident that, as the vortex evolves, it not only generates daughter vortices but also leads to spanwise vortices which interact with each other in a complicated way. This necessitates the study of spanwise vortex interaction.

Figure 4.8 Evolution of 2 spanwise vortices separated by $z^+=100$ at (a) $t^+=25$; $t^+=225$; (b) $t^+=350$; $t^+=500$. Both initial vortices have strength $\alpha=2.5$ and no inclination to the $z$ axis ($\beta=0$).
Lateral interaction between hairpins must be an important ingredient in the spanwise scaling of the hairpin vortices as they grow along the streamwise and wall-normal directions. As the packets expand in the spanwise direction they must ultimately interact by vortex encounters. Encounters also occur due to larger, faster packets running over smaller, slower packets. In lateral encounters, the opposing vorticity in adjacent legs of two hypothetically identical hairpins could annihilate them, resulting in a larger hairpin of the same height, but double the width of the original hairpins. As hypothesized in Adrian, Balachandar and Liu (2001), the merger between 2 spanwise vortices (‘A’ in figure 4.8 (a)) leads to a larger vortex of the same height (‘B’ in figure 4.8 (a)). This large vortex autogenerates resulting in asymmetric vortices inclined to the z axis (‘C’ and ‘D’ in figure 4.8 (b)).

4.4 Interaction between Q2 and Q4 events

Case I

(a)
Case II

Figure 4.9 Evolution of a Q2 event and a Q4 event together. The vortices are initially separated by \( x^+ = 100 \) units. \( \alpha = 2.5; \beta = 0 \); The initial condition was considered at the center of the xz plane and at \( y^+ = 46.6 \). (a) Q2-Q4 combination where A(i) represents the Q2 vortex and A(ii) represents the Q4 vortex. B represents the structure at \( t^+ = 250 \) and C is the structure at \( t^+ = 500 \) (b) Q4-Q2 combination where A(i) represents the Q4 vortex \((4, -3.5, 0)\) at \((x^+ = 0, y^+ = 46.6, z^+ = 0, t^+ = 0)\) and A(ii) represents the Q2 vortex \((-4, 3.5, 0)\) at \((x^+ = 100, y^+ = 46.6, z^+ = 0, t^+ = 0)\). B represents the structure at \( t^+ = 250 \) and C is the structure at \( t^+ = 500 \).

The initial condition in case I grows into a complex structure with the Q4 event developing into 2 quasi-streamwise vortices and the Q2 event auto-generating into daughter vortices although the interaction is non-linear. In case II however, the Q4 vortex rapidly dissipates and a single Q2 hairpin vortex is formed at \( t^+ = 250 \) and \( t^+ = 500 \) (B and C in figure 4.8).

4.5 Interaction between vortices at different \( y^+ \) locations

Computations were done with 1 vortex at \( y^+ = 46.6 \) and the other at \( y^+ = 11.8 \) separated by \( x^+ = 100 \) units. The higher vortex consumes the lower one at a very early time (\( t^+ = 50 \)) and the combination behaves similar to the single vortex evolution (figure 4.11).
Figure 4.10 (a) 2 vortices separated by 100 x^+ units at t^+=0. Vortex A is at y^+=46.6 and vortex B is at y^+=11.8; (b) the vortex structure at t^+=50; (c) The vortex structure at t^+=50 when the vortex B is absent;

Figure 4.11 (a) Evolution of the single vortex at y^+=46.6 and (b) the evolution of the dual vortices at t^+=400. The similarity in structure leads us to believe that vortex B doesn’t have a major role to play in the evolution.
4.6 Interaction between vortices in a staggered arrangement

(a)

(b)

(c)
Figure 4.12 Evolution of 5 Q2 events placed in a staggered arrangement. The schematic diagram of the arrangement is shown in figure 4.13. $\alpha = 2.5; \beta = 0$; All the initial vortices are at $y^+ = 46.6$. (a) The evolution structure at $t^+ = 25$ (b) The evolution structure at $t^+ = 175$ (c) The evolution structure at $t^+ = 375$ (d) The evolution structure at $t^+ = 500$.

Figure 4.13 A schematic arrangement of staggered vortices in channel flow

The vortices in figure 4.12 (a) grow asymmetrically (with inclination to the z axis) till $t^+ = 175$. The canes formed in figure 4.12 (b) then dissipate leading to the vortex structure in figure 4.12 (d). After $t^+ = 175$, there is no more cane formation.
CONCLUSIONS AND RECOMMENDATIONS

This study attempted to answer questions concerning how vortex mergers produce larger scale structures, a process that may contribute to the growth of length scale with increasing distance from the wall in turbulent wall flows. This would aid in modeling the von Karman constant which is crucial in drag related studies.

The dynamics of hairpin vortices in turbulent channel flow have been studied using direct numerical simulation. The two-point spatial correlation of the fully turbulent velocity field was initially studied in detail and compared to existing literature. Linear stochastic estimation was then used to estimate the structures of the initial three-dimensional vortices. The vortices were visualized using the iso-surface of the imaginary part of the conjugated complex eigenvalues of the local velocity gradient tensor ($\lambda_{ci}$). The Reynolds number of the present simulation is more than twice that of the Re$_{\tau}$=180 flow studied by Zhou et al. (1999), and a number of new types of events such as quasi-streamwise vorticity and Q4 events were studied in this work. The larger Re$_{\tau}$ also made it possible to simulate the evolution of the vortices over longer periods of time, and correspondingly larger head heights.

The effect of asymmetry, $y^+$ position and strength were evaluated for single vortices. In order to study the complex non-linear interactions between vortices, various parameters such as spanwise inclination and strength were varied.
Grid independence study was performed to choose the optimum grid. The following are the conclusions from this study.

1. Autogeneration is insensitive to $Re_\tau$, as results change little from $Re_\tau=180$ (Zhou et al. 1999) to $Re_\tau=395$. The forms of the eddies at $Re_\tau=395$ are similar to those at $Re_\tau=180$, although there is no auto-generation evident upto $\alpha=1.25$. Hence, the auto-generation threshold is shifted from $\alpha=1$ for $Re_\tau=180$ to $\alpha=1.25$ for $Re_\tau=395$.

2. Single vortex evolution: Just like the strength threshold for autogeneration, there exists asymmetry threshold for cane formation. Canes are not produced till $\beta=0.4$. For a symmetric evolution, the flow becomes fully turbulent and occupies the entire channel around $t^+=2000$.

3. Multiple vortex interaction
   
   a. Larger $Q_2$ overtakes smaller $Q_2$
   
   b. Smaller $Q_2$ behind a larger $Q_2$ just separates.
   
   c. $Q_2$ behind $Q_4$ leads to auto-generation with the $Q_4$ event becoming 2 quasi-streamwise vortices at $t^+=500$.
   
   d. $Q_4$ behind $Q_2$ rapidly dissipates the $Q_4$ vortex.
   
   e. Lateral vortices merge in $t^+=100$.
   
   f. Staggered vortices merge in $t^+=175$.
   
   g. Two vortices, one at $y^+=46.6$ and the other at $y^+=11.8$, separated by $x^+=100$ evolve in a similar fashion to a single vortex at $y^+=46.6$; i.e. the vortex at a lower $y^+$ value does not play a significant part in the evolution.
Future work would include specifying $d(x,t)$ in addition to $u(x,t)$ which would lead to a more detailed picture. The full potential of stochastic estimation is realized when all the components of the given data $u(x,t)$ and possibly $d(x,t)$ are specified. Attempts also need to be made to separate the linear and non-linear effects to simplify the problem. Higher Reynolds numbers and bigger domains (e.g. doubling the length of the channel) are recommended based on the computational resources available.
REFERENCES


Kim, K., Sung, H.J. & Adrian, R.J. 2008 Effects of background noise on generating coherent packets of hairpin vortices, Phys. Fluids 20, 105107


Meinhart, C. D., Adrian, R. J. & Tomkins, C. D. 2000 Vortex organization in the outer region of a turbulent boundary layer, J. Fluid Mech, 424, 1-54


APPENDIX A

LINEAR STOCHASTIC ESTIMATION DERIVATION
Let $g(x')$ be any quantity associated with the turbulent flow, and let $E_1(x_1), E_2(x_2), E_3(x_3), \ldots, E_N(x_N)$ be $N$ random whose value assume specified event values at (possibly) $N$ different points. The conditionally averaged flow field is the averaged flow field given that the specified events occur [Zhou et. al. (1999)]:

$$<g(x')|E_1(x_1), E_2(x_2), E_3(x_3), \ldots, E_N(x_N)>$$  \hspace{1cm} (A1)

It is the best estimate of the flow field in terms of the known event, in the mean square sense. To streamline the notation, we often let $E$ be the $N$ dimensional event vector

$$E=[f_1\leq E_1(x_1)<f_1+df_1 \text{ and } \ldots \text{ and } f_N\leq E_N(x_N)<f_N+df_N]$$  \hspace{1cm} (A2)

The *linear stochastic estimate* of a conditional average is found by expanding the conditional average in a power series about the event $E=0$, and truncating the expansion at some level,

$$<g_i|E>=L_i E_1+E_m E_{m+}\ldots$$  \hspace{1cm} (A3)

The unknown coefficients $L, N$ etc. are determined by requiring that the mean-square error between the approximation and the conditional average be minimized. In the case of linear estimation only the first term is retained and the minimization leads to a set of linear algebraic equations for $L_i$,

$$<E_m(x_m)E_l(x_l)> L_{il} = E_m(x_m) g_l(x')>$$  \hspace{1cm} (A4)

Where $l=1,2,3,\ldots,N$ and $m=1,2,3,\ldots,N$. We assume that the event and the estimated quantity have zero mean in equations (A3) and (A4).

Equation (A4) can be written as

$$AL_i = b_i$$  \hspace{1cm} (A5)

Since the streamwise ($x$) and spanwise ($z$) directions are homogenous in the periodic channel flow,

$$A_{ml} = R_{EmE_l}(x_l-x_m, y_m, z_l-z_m)$$  \hspace{1cm} (A6)

and

$$b_{im} = R_{Emg_l}(x'-x_m, y_m, y', z'-z_m)$$  \hspace{1cm} (A7)

$L_i$ can be obtained from solving the matrix equation with $N \times N$ symmetric coefficient matrix $A$. Finally, the linear stochastic estimation (LSE) of the conditional average is

$$<g_i|E> \sim L_i(x';x_1, x_2, \ldots, x_N)E_l(x_l)$$  \hspace{1cm} (A8)
$$L_{nl} = \begin{bmatrix}
\langle u_1 u_1 \rangle & \langle u_2 u_1 \rangle & \langle u_3 u_1 \rangle \\
\langle u_1 u_2 \rangle & \langle u_2 u_2 \rangle & \langle u_3 u_2 \rangle \\
\langle u_1 u_3 \rangle & \langle u_2 u_3 \rangle & \langle u_3 u_3 \rangle 
\end{bmatrix} \quad (A9)$$
APPENDIX B

LINEAR STOCHASTIC ESTIMATION CODE
This code uses correlation functions to produce a conditional vortex using linear stochastic estimation.

Grid: 128 x 129 x 128;

Re$_t$ = 395;

Language: Fortran 95;

Machine it ran on: Saguaro (ASU high performance computing center);

Number of processors: 1;

Input parameters: u, v and w components of velocity, strength $\alpha$, asymmetric factor $\beta$, position $y^+$;

Output parameters: up.dat, vp.dat, wp.dat (velocity fields in .dat format), l_ci.dat ($\lambda_{ci}$ in .dat format)

```fortran
include 'param.h'
common/LSE/nv_evn(N_env), nv_est(N_est), multi(N_env,3), j1
common/domain/sx, sz
common/para/re

real*8 event(N_env)
real*8 AI(N_env,N_env)

real*8 b(N_env,nx,nyp,nz)
real*8 CL(N_env,nx,nyp,nz)
character*8 dummy8
character*45 dummy45

re = 395.

Pi = acos(-1.0)
sx = 2.*pi
sz = 1./1.*pi

open(70, file='lse.set', status='old', action='read')
read(70,102) dummy45
write(*,102) dummy45
read(70,100) dummy8, alpha
write(*,100) dummy8, alpha
read(70,100) dummy8, beta
write(*,100) dummy8, beta
100 format(a8,e14.8)
101 format(a8,i5)
```
call setup

c--- set event or condition variables and locations

    nv_evn(1) = 1  ! u'
    nv_evn(2) = 2  ! v'
    nv_evn(3) = 3  ! u'
    nv_evn(4) = 1  ! u'
    nv_evn(5) = 2
    nv_evn(6) = 3
    nv_evn(7) = 1
    nv_evn(8) = 2
    nv_evn(9) = 3
    nv_evn(10) = 1
    nv_evn(11) = 2
    nv_evn(12) = 3
    nv_evn(13) = 1
    nv_evn(14) = 2
    nv_evn(15) = 3

    event(:,)= 0.0

    read(70,102) dummy45  
    write(*,102) dummy45
    read(70,101) dummy8,j1  
    write(*,101) dummy8,j1

    c--- event variables are normalized by wall units

    do ll=1,N_evn
        read(70,103) dummy8,event(ll),(multi(ll,k),k=1,3)
        write(*,103) dummy8,event(ll),(multi(ll,k),k=1,3)
    103        format(a8,e12.5,3i3)
    enddo

    c--- for multi location event -> Read from lse.set file
    c        multi(ll,1)  ! relative x location of ll-th event w.r.t 1st event location
    c        multi(ll,2)  ! relative y location of ll-th event w.r.t 1st event location
    c        multi(ll,3)  ! relative z location of ll-th event w.r.t 1st event location

    c--- impose asymmetry in z dir.
    Event(3) = beta*(event(1)**2 + event(2)**2)**0.5  ! w_m=(u_m^2+v_m^2)^0.5
    event(1) = event(1)*(1.0-beta**2)**0.5
    event(2) = event(2)*(1.0-beta**2)**0.5

    c--- multiply strength factor to event vector
    event = alpha*event
do ll=1,N_evn
   write(*,104) ll,event(ll),(multi(ll,k),k=1,3)
104      format(i5,e12.5,3i3)
enddo

c--- set the quantities which will be estimated by LSE

nv_est(1) = 1 ! u'
nv_est(2) = 2 ! v'
nv_est(3) = 3 ! w'

c----
call set_coef_AI(AI) ! AI = inverse of A
do i_est = 1, N_est
   call read_b(b,i_est)
call get_CL(CL,AI,b)
call do_LSE(CL,event,i_est)
enddo ! N_est
call out_put(event)

stop
end

c----+--------------------------------------------------------
subroutine setup
include 'param.h'
common/mesh/y(nyp),dx,dz
common/domain/sx,sz
pi = acos(-1.0)
do j=1,nyp
   y(j)=1.-cos(pi*real(j-1)/real(nyp-1))
enddo
dx = sx/real(nx)
dz = sz/real(nz)
return
end

c----+--------------------------------------------------------
subroutine set_coef_AI(AI)
include 'param.h'
common/LSE/nv_evn(N_evn),nv_est(N_est),multi(N_evn,3),j1
real*8 r(N_evn),x(N_evn)
real*8 A(N_evn,N_evn)
real*8 AI(N_evn,N_evn)
real*8 test(N_evn,N_evn)

character*50 filename
real*8 E(N_evn,nx,nyp,nz)
real*8 Em(N_evn),Eq(N_evn,N_evn)
real*8 t(nx,nyp,nz)

A(:,:,)= 0.0
AI(:,:,)= 0.0

c---

do m=1,N_evn
  do l=m,N_evn
    i_m = multi(m,1)
    j_m = j1 + multi(m,2)
    k_m = multi(m,3)
    i_l = multi(l,1)
    j_l = j1 + multi(l,2)
    k_l = multi(l,3)

    filename = '../03_corr/R_'
    nn=index(filename,'R')
    write(unit=filename(nn+2:),fmt='(bn,i2.2)') nv_evn(m)
    write(unit=filename(nn+4:),fmt='(bn,a1)')   '_'
    write(unit=filename(nn+5:),fmt='(bn,i2.2)') nv_evn(l)
    write(unit=filename(nn+7:),fmt='(bn,a2)')   'Y_'
    write(unit=filename(nn+9:),fmt='(bn,i3.3)') j_m
    write(*,*), filename
    open(10,file=filename,form='unformatted')
    read(10) (((t(I,j,k),i=1,nx),j=1,nyp),k=1,nz)
    close(10)

    A(m,l) = t(nx/2+i_l-i_m,j_l,nz/2+k_l-k_m)
  enddo
enddo

c----

c---  A(m,l) should be symmetric.

Do m=2,N_evn
  do l=1,m-1
    A(m,l)=A(l,m)
  enddo
enddo

c---  calculate the inverse of A
call FindInv(A, AI, N_evn, ErrorFlag)

c--- check inversion of A
  test = 0.0  
do j=1, N_evn
    do i=1, N_evn
      do m=1,N_evn
        test(I,j) = test(I,j) + A(I,m)*AI(m,j)
      enddo
    enddo
  enddo
write(*,*) 'check the inversion'
do I = 1, N_evn
  write(*,*) (test(I,j),j=1,N_evn)
enddo

c---
  do m=1,N_evn
    write(*,100) (A(m,l),l=1,N_evn)
  enddo
  write(*,*)
  do m=1,N_evn
    write(*,100) (AI(m,l),l=1,N_evn)
  enddo
100   format(4(E12.5,x))
return
end

c----+--------------------------------------------------------
subroutine read_b(b,i_est)
include 'param.h'
common/LSE/nv_evn(N_evn),nv_est(N_est),multi(N_evn,3),j1
real*8 t(N_evn,nx,nyp,nz)
real*8 b(N_evn,nx,nyp,nz)
character*50 filename

do m = 1, N_evn
  j_m = j1 + multi(m,2)
  filename = '../03_corr/R_'
nn=index(filename,'R')
  write(unit=filename(nn+2:),fmt='(bn,i2.2)') nv_evn(m)
  write(unit=filename(nn+4:),fmt='(bn,a1)') '_'
  write(unit=filename(nn+5:),fmt='(bn,i2.2)') nv_est(i_est)
  write(unit=filename(nn+7:),fmt='(bn,a2)') 'Y_'
  write(unit=filename(nn+9:),fmt='(bn,i3.3)') j_m
  write(*,*) filename

68
open(10, file=filename, form='unformatted')
read(10) (((t(m,I,j,k),i=1,nx),j=1,nyp),k=1,nz)
close(10)

do j=1,nyp
  do k=1,nz
    do i=1,nx
      i_m = i - multi(m,1)
k_m = k - multi(m,3)
      if (i_m.lt.1) i_m = i_m + nx
      if (k_m.lt.1) k_m = k_m + nz
  b(m,I,j,k) = t(m,i_m,j,k_m)
  enddo
enddo
enddo
return
de

subroutine get_CL(CL,AI,b)
include 'param.h'
common/LSE/nv_evn(N_evn),nv_est(N_est),multi(N_evn,3),j1
real*8 AI(N_evn,N_evn)
real*8 b(N_evn,nx,nyp,nz)
real*8 CL(N_evn,nx,nyp,nz)
CL(:,,:,:,:)= 0.0

do l=1,N_evn
  do k=1,nz
    do j=1,nyp
      do i=1,nx
        do m=1,N_evn
          CL(1,l,i,j,k)=CL(1,l,i,j,k)+AI(l,m)*b(m,i,j,k)
        enddo
      enddo
    enddo
  enddo
enddo
return
de

subroutine do_LSE(CL,event,i_est)
include 'param.h'
common/LSE/nv_evn(N_evtn),nv_est(N_est),multi(N_evtn,3),j1
common/domain/sx,sz
common/mesh/y(nyp),dx,dz

real*8 event(N_evtn)
real*8 CL(N_evtn,nx,nyp,nz)

real*8 g(nx,nyp,nz) ! estimated quantity

character*50 filename ! estimated quantity

g(:,:,:) = 0.0

do k=1,nz
  do j=1,nyp
    do i=1,nx
      do l=1,N_evtn
        g(I,j,k) = g(I,j,k) + CL(l,I,j,k)*event(l)
      enddo
    enddo
  enddo
enddo

filename = 'output_LSE'
nn=index(filename,'E')
write(unit=filename(nn+1:),fmt='(bn,i1.1)') i_est
write(*,*) filename
open(10,file=filename,status='unknown',form='unformatted')
write(10) (((g(I,j,k),i=1,nx),j=1,nyp),k=1,nz)
close(10)

return
end

 subroutine out_put(event)
 include 'param.h'
 common/LSE/nv_evn(N_evtn),nv_est(N_est),multi(N_evtn,3),j1
 common/mesh/y(nyp),dx,dz
 common/domain/sx,sz
 common/para/re

 real*8 event(N_evtn)

 real*8 g(N_est,nx,nyp,nz)
     ! -> usually N_est=1,2,3 denote u,v,w

 real*8 um(nyp)
 character*50 filename

 real*8 l_ci(nx,nyp,nz)
c--- read the estimated field from file

    g(:,:,,:) = 0.0

do i_est=1,N_est
    filename = 'output_LSE'
nn=index(filename,'E')
    write(unit=filename(nn+1:),fmt=('bn,i1.1')) i_est
    write(*,*) filename
    open(10,file=filename,status='unknown',form='unformatted')
    read(10) (((g(i_est,I,j,k),i=1,nx),j=1,nyp),k=1,nz)
    close(10)
endo

c--- read averaged statistics
open(10,file='../01_mean/output.dat',status='old')
do j=1,nyp
    read(10,200)
    dummy,um(j),dummy,dummy,dummy,dummy,dummy,
& dummy,dummy,dummy,dummy,dummy,dummy
endo
200 format(13(e12.5,x))
close(10)

c--- total velocity
    do k=1,nz
    do j=1,nyp
    do i=1,nx
        g(1,i,j,k) = g(1,i,j,k)+um(j)
    enddo
    enddo
    enddo

c--- write estimated velocity field
    open(10,file='u.dat',status='unknown',form='unformatted')
    write(10) (((g(1,I,j,k),i=1,nx),j=1,nyp),k=1,nz)
    close(10)

    open(10,file='v.dat',status='unknown',form='unformatted')
    write(10) (((g(2,I,j,k),i=1,nx),j=1,nyp),k=1,nz)
    close(10)

    open(10,file='w.dat',status='unknown',form='unformatted')
    write(10) (((g(3,I,j,k),i=1,nx),j=1,nyp),k=1,nz)
    close(10)

c--- calculate lambda_ci
    call get_lambda_ci(g,l_ci)

    open(11,file='l_ci.dat',status='unknown')
write(11,*) "variables=x,y,z,lci,u,v,w,uf"
write(11,*)
>zone i='nx/3*2-nx/3+1', j='nyp', k='nz/3*2-nz/3+1', f=point'
do k=nz/3,nz/3*2
do j=1,nyp
do i=nx/3,nx/3*2
  yy=y(j)*re
  xx=real(i-1)*dx*re
  zz=real(k-1)*dz*re
  write(11,102) xx,yy,zz,l_ci(I,j,k)
102 format(8(e12.5,x))
enddo
enddo
enddo
close(11)
return
dend

subroutine get_lambda_ci(g,l_ci)
includ e'param.h'
common/mesh/y(nyp),dx,dz
common/domain/sx,sz
common/para/re
real*8 g(3,nx,nyp,nz)
  !-> 1,2,3 denote u,v,w
real*8 l_ci(nx,nyp,nz)
real*8 d11(nx,nyp,nz),d12(nx,nyp,nz),d13(nx,nyp,nz)
real*8 d21(nx,nyp,nz),d22(nx,nyp,nz),d23(nx,nyp,nz)
real*8 d31(nx,nyp,nz),d32(nx,nyp,nz),d33(nx,nyp,nz)
real*8 q1(nx,nyp,nz)
real*8 q2(nx,nyp,nz)
real*8 q3(nx,nyp,nz)
c--- d_ij = dq_i/dx_j
call init_partial
call fftw_ini
do k=1,nz
do j=1,nyp
do i=1,nx
  q1(I,j,k) = g(1,I,j,k)
enddo
enddo
enddo

call partial(1,q1,d11)
call partial(2,q1,d12)
call partial(3,q1,d13)

do k=1,nz
do j=1,nyp
do i=1,nx
    q2(I,j,k) = g(2,I,j,k)
enddo
enddo
enddo

call partial(1,q2,d21)
call partial(2,q2,d22)
call partial(3,q2,d23)

do k=1,nz
do j=1,nyp
do i=1,nx
    q3(I,j,k) = g(3,I,j,k)
enddo
enddo
enddo

call partial(1,q3,d31)
call partial(2,q3,d32)
call partial(3,q3,d33)

c--- calculating lambda_ci

l_ci(:,:, :) = 0.0d0 ! l_ci

do 1 j=1,nyp
do 1 k=1,nz
  do 1 i=1,nx
    e11 = d11(i,j,k)
e12 = d12(i,j,k)
e13 = d13(i,j,k)
e21 = d21(i,j,k)
e22 = d22(i,j,k)
e23 = d23(i,j,k)
e31 = d31(i,j,k)
e32 = d32(i,j,k)
e33 = d33(i,j,k)
p = - (e11 + e22 + e33)
q = 0.5*(p**2 - (e11**2 + e22**2 + e33**2 + e12*e21*2.0 + e13*e31*2.0 + e23*e32*2.0)
& )
& )
&    r = -( - e13*e22*e31 + e12*e23*e31
& + e13*e21*e32 - e11*e23*e32
& - e12*e21*e33 + e11*e22*e33
& )

r0 = r + 2./27.*p**3 - 1./3.*p*q
q0 = q - 1./3. *p**2
dis = (r0/2.)**2 + (q0/3.)**3
if (dis.gt.0.0) then
    reg1 = sqrt(dis)
    reg2 = reg1 - r0/2.0
    reg3 = reg1 + r0/2.0
    if (reg2 .gt. 0.0) then
        reg2 = reg2**(1./3.)
    else
        reg2 = -(-reg2)**(1./3.)
    endif
    if (reg3 .gt. 0.0) then
        reg3 = reg3**(1./3.)
    else
        reg3 = -(-reg3)**(1./3.)
    endif
    l_ci(I,j,k) = sqrt(3.)/2.0*(reg2 + reg3)
else
    l_ci(I,j,k) = 0.0
endif
1    continue
return
end
APPENDIX C

NAVIER-STOKES SOLUTION
This code solves the incompressible, constant viscosity Navier-Stokes equation in a channel flow geometry of height $h$ and imposed pressure gradient $dP/dx=1$. The variables in the code are all made non-dimensional by the wall friction velocity and the viscous length scale, c.f. equations (1.1 a) and (1.1 b) in the text. The solution is performed by Fourier spectral decomposition in the x- and z-directions, and Chebychev polynomials in the y-direction. The grid is $128 \times 129 \times 128$.

Input variable to the code are:

u.ini, v.ini, w.ini (the initial condition given by the LSE code)

Output variables are fluctuating u, v and w velocities and pressure for every “idmpfrq” iterations (the u,v,w and p values are typically written every 500 iterations which is equivalent to 25 time units ).idmpfrq and the total number of timesteps are defined in the APPENDIX D code.

The time step is given by $dt=1.25e-04$. It is set in APPENDIX D.

The code calls the initial condition from the folder ic_data (refer “set directory from argument” in code below).

!c-- 07/22/06
!c irstrt is removed.
!c to preserve second-order temporal accuracy between succesive runs,
!c the nonlinear terms are read from the file such as fu.ini, fcxx.ini, etc.
!c time history of cij is added.

!c-- 07/31/07
!c write pressure is added
!c n+1 and n-1 step fields are written to calculate time-derivatives

!c-- 09/19/09
!c actual pressure is written instead of dt*p
!c not to write fu.ini, fcxx.ini, etc

!c-- 09/27/09
!c read n_ini, time for succesive calculation

!c-- 10/09/09
!c change dPdx linearly in time

!c-- 10/19/09
!c employing dump_data logical variable

!c=================================  
== program main
use parameters
use new_derivatives
use wave_numbers_stuf
use general_stuf
use fftw_routines
use xyzfft
!
main program for turbulent channel flow
allowing to split 1 or 2 dimensions over the processors
data storage: us(nyp,kcomy) = us(nyp,nkz,kxh) etc

implicit none
include 'mpif.h'

complex(8), dimension (1:nyp,1:kcomy) :: us, vs, ws
complex(8), dimension (1:nyp,1:kcomy) :: u, v, w
complex(8), dimension (1:nyp,1:kcomy) :: pressure
complex(8), dimension (1:nyp,1:kcomy) :: temp1, temp2, temp3
complex(8), dimension (1:nyp,1:kcomy) :: temp

non-linear term at (n) and (n-1) steps
complex(8), dimension (1:nyp,1:kcomy) :: fnm, gnm, hnm
complex(8), dimension (1:nyp,1:kcomy) :: fn, gn, hn
common/block1/ fnm, gnm, hnm
common/block5/ fn, gn, hn

boundary conditions
complex(8), dimension (1:kcomy) :: bctop, bcbot, pbctop, pbcbot

influence matrix for helmholtz eq
real(8), dimension (0:ny,1:kcomy) :: a, ag, ac
real(8), dimension (1:kcomy) :: wn, wng, wnc
real(8), dimension (1:ny) :: wd, wi, wr

the flow variables in physical space
real(8), dimension (1:nx,1:kcomx) :: srxxp, srxy, srxzp
real(8), dimension (1:nx,1:kcomx) :: sryyp, sryzp, srxzp
real(8), dimension (1:nx,1:kcomx) :: dxtp, dytp, dztp
real(8), dimension (1:nx,1:kcomx) :: rxxp, rxy, rxzp
real(8), dimension (1:nx,1:kcomx) :: ryp, ryzp, rzzp
real(8), dimension (1:nx,1:kcomx) :: omxp, omyp, omzp
real(8), dimension (1:nx,1:kcomx) :: up, vp, wp

real(8) :: cfl_max, div_max, re_m, time

for cpu time measuring
real(4) :: cpu_start, cpu_end, cpu_proc, cpu_sum, cpu_max

mpi related constant
integer ierr, nprocmpi
integer mynum
common/cbpar2/ mynum
!c--- indicies and coefficients
integer :: i, iy, izx, it, in, n_ini
integer :: ibp, ib, error
real(8) :: g, sg, sa, con_1, con_2

!c--- input and output file names
character(len=70), dimension (1:99) :: namein, nameout

!c--- fene-p model
complex(8), dimension (1:nyp,1:kcomy) :: cxx,cxy,cxz,cyy,cyz,czz
complex(8), dimension (1:nyp,1:kcomy) :: c_fnxx, c_fnxy, c_fnxz
complex(8), dimension (1:nyp,1:kcomy) :: c_fnyy, c_fnyz, c_fnzz
complex(8), dimension (1:nyp,1:kcomy) :: c_fnmxx, c_fnmxy, c_fnmxz
complex(8), dimension (1:nyp,1:kcomy) :: c_fnmyy, c_fnmyz, c_fnmzz

!c--- pressure gradient change
integer :: time_region
real(8) :: re_tau_time
real(8) :: dpdx_time
logical :: dump_data

!c--- directory input (JRB)
character*30 :: dirarg
integer :: iargc

!c---------------------------------------------------------------------
!c    mpi initializations
!c---------------------------------------------------------------------
if ( nproc > 1 ) then
  call mpi_init( ierr )
  if ( ierr /= 0 ) stop "init 1"
  call mpi_comm_rank( mpi_comm_world, mynum, ierr )
  if ( ierr /= 0 ) stop "init 2"
  call mpi_comm_size( mpi_comm_world, nprocmpi, ierr )
  if ( ierr /= 0 ) stop "init 3"
  if ( nprocmpi /= nproc ) stop 'error nproc'
  if ( mod(nxh,nproc) /= 0 ) stop 'error nproc'
  if ( mod(nz,nproc) /= 0 ) stop "invalid nproc: see nxh (1)"
else
  mynum = 0
endif

!c----------------------------
!c     set directory from argument
!c--------------------------------
if (iargc().ne.1) stop "must set argument: <program> <#####>"
call getarg(1,dirarg)
folder_in="ic_data"//trim(dirarg)="/"
folder_out="/scratch/pkvraman/output"//trim(dirarg)="/"
write(*,*) "folder_in > ", trim(folder_in)
write(*,*) "folder_out > ", trim(folder_out)

!c---------------------------------------------------------------------
--
!c    setup
!c---------------------------------------------------------------------
--
call setstuf
if( .not.solve_fenep_model .and. beta /= 1.d0 ) stop "beta"

!c---------------------------------------------------------------------
--
!c    define the input/output arrays
!c---------------------------------------------------------------------
--
namein(1) = "u"
namein(2) = "v"
namein(3) = "w"
namein(4) = "cxx"
namein(5) = "cxy"
namein(6) = "cxz"
namein(7) = "cyy"
namein(8) = "cyz"
namein(9) = "czz"
namein(10) = "p"
namein(11) = "fu"
namein(12) = "fv"
namein(13) = "fw"
namein(14) = "fcxx"
namein(15) = "fcxy"
namein(16) = "fcxz"
namein(17) = "fcyy"
namein(18) = "fcyz"
namein(19) = "fczz"
nameout = namein

do i = 1, 19
   namein(i)  = trim(folder_in)//trim(namein(i)//".ini"
   nameout(i) = trim(folder_out)//trim(nameout(i)//"."
enddo

!c---------------------------------------------------------------------
--
!c    initialisations for fft routines
!c---------------------------------------------------------------------
--
call xyzfft_ini
call ccosexp_trig

!c---------------------------------------------------------------------
--
!c    set up the boundary conditions at y = [-1, 1] for channel flow
always check for consistency the pressure bcs for the zero mode
(dp/dy(1)-dp/dy(-1))*dyde = v(1) - v(-1)

---

bctop  = dcmplx(0.0d0, 0.0d0)
bcbot  = dcmplx(0.0d0, 0.0d0)
pbctop = dcmplx(0.0d0, 0.0d0)
pbcbot = dcmplx(0.0d0, 0.0d0)

n_ini = 0    ! if n_ini is not 0, the initial files for nonlinear
terms at (n-1) step are required.
time = 0.0d0

--- for interactive job
read (*,*) n_ini
read (*,*) time
n_ini=0
time=0.0

write(*,*) mynum, n_ini, time

---

--- read initial data
---

if ( solve_fenep_model ) then
    call var_scatter( cxx, namein(4) )
    call var_scatter( cxy, namein(5) )
    call var_scatter( cxz, namein(6) )
    call var_scatter( cyy, namein(7) )
    call var_scatter( cyz, namein(8) )
    call var_scatter( czz, namein(9) )
    if ( n_ini .ne. 0 ) then
        call var_scatter( c_fnmxx, namein(14) )
        call var_scatter( c_fnmxy, namein(15) )
        call var_scatter( c_fnmxz, namein(16) )
        call var_scatter( c_fnmyy, namein(17) )
        call var_scatter( c_fnmyz, namein(18) )
        call var_scatter( c_fnmzz, namein(19) )
    endif
endif
!c---------------------------------------------------------------------
!c    initialize the influence matrix in initial.
!c---------------------------------------------------------------------

call initial( pressure, temp, &
    bctop, bcbot, a, ag, ac, wn, wng, wnc, wd, wl, wr,
    pbctop, pbcbot)

!c--- write simulation parameters

if ( nproc > 1 ) call mpi_barrier( mpi_comm_world, ierr )
if ( mynum == 0 ) then
    write(*,*) '------------------------------------', &
    ' parameters ', &
    '------------------------------------'
    write(*,*) 're_tau = ', re_tau
    if ( scale_by_pi ) then
        write(*,*) 'len_x = ', xl * acos(-1.0)
        write(*,*) 'len_z = ', zl * acos(-1.0)
    else
        write(*,*) 'len_x = ', xl
        write(*,*) 'len_z = ', zl
    endif
    write(*,*) 'nx = ', nx
    write(*,*) 'ny = ', ny
    write(*,*) 'nz = ', nz
    write(*,*) 'dt = ', dt
    write(*,*) 'nproc = ', nproc
    if ( solve_fenep_model ) then
        write(*,*) 'we_tau = ', we_tau
        write(*,*) 'beta   = ', beta
        write(*,*) 'l_max  = ', lmax
        write(*,*) 'diffusivity = ', diffusivity
    endif
    write(*,*)
endif
write(*,*) '------------------------------------', &
    ' program starts ', &
    '------------------------------------'
write(*,*)

!c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++
!c    main time stepping loop
!c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++
call cpu_time( cpu_start )

    do it = n_ini + 1, n_ini + nsteps
!c---------------------------------------------------------------------
!c    calculate the vorticity, strain-rate tensor and velocity in
physical domain.
!c---------------------------------------------------------------------
---
call vort1( u, v, w, omxp, omyp, omzp )
call gadot( u, v, w, srxxp,srxyp,srxzp,sryyp,sryzp,srzzp )
call xyzfftsp( u, up )
call xyzfftsp( v, vp )
call xyzfftsp( w, wp )
call divergence( div_max, srxxp, sryyp, srzzp )
call cfl_number( cfl_max, up, vp, wp )
if ( mynum == 0 ) call mean_reynolds_number( re_m, u, time )
if ( mynum == 0 ) call time_history( up, vp, wp, time )
if ( mynum == 0 ) then
  write(*,100) it-1, time, re_m, div_max, cfl_max
100     format(' step =',i7,2x,':  t =',e15.9,2x,' re_m
  =',e15.9,2x, &
  ' div =',e15.9,2x,' cfl =',e15.9)
endif

time = time + dt
---
!c    adams-bashforth  for it > 1
!c    bacward euler    for it = 1 and read the array of filled zero
!c---------------------------------------------------------------------
---
!c
if ( it == 1 ) then
  con_1 = dt
  con_2 = 0.d0
else
  con_1 = + 1.5d0*dt
  con_2 = - 0.5d0*dt
endif
---
!c--- store spectral coefficient u^(n) to us
forall( izx=1:kcomy, iy=1:nyp )
  us(iy,izx) = u(iy,izx)
  vs(iy,izx) = v(iy,izx)
  ws(iy,izx) = w(iy,izx)
endforall
if ( solve_fenep_model ) then
    call get_polymer_stress ( cxx, cyy, czz, &
        cxy, cxz, cyz,
        c_fnxx, c_fnyy, c_fnzz, &
        c_fnxy, c_fnxz, c_fnyz, &
        c_fnmxx, c_fnmxy, c_fnym, &
        c_fnmxz, c_fnmyz, c_fnmzz, &

        c_fnmxy, c_fnmxz, c_fnmxy,
        omxp, omyp, omzp,
        srxxp, sryyp, srzzp,
        srxy, srxz, sryz,
        up, vp, wp,
        con_1, con_2,
        ac, wnc, wd, wi, wr,
        time )
endif

!c--- update the polymer stress contribution

sa = (1.d0-beta)*(dt/2.d0)/re_tau

do i = 1, kcomy
    u(:,i) = u(:,i) + sa*( x_der_1(c_fnxx(:,i),i) &
        y_der_1(c_fnxy(:,i)) &
        z_der_1(c_fnxz(:,i),i))

    v(:,i) = v(:,i) + sa*( x_der_1(c_fnxy(:,i),i) &
        y_der_1(c_fnyy(:,i)) &
        z_der_1(c_fnyz(:,i),i))

    w(:,i) = w(:,i) + sa*( x_der_1(c_fnxz(:,i),i) &
        y_der_1(c_fnyz(:,i)) &
        z_der_1(c_fnzz(:,i),i))
endo
do i = 1, kcomx

!c--- -( u.grad (u))/2 part

    dxtp(:,i) = -(   up(:,i)* srxxp(:,i) &
      + vp(:,i)*(srxyp(:,i)-omzp(:,i)) &
      + wp(:,i)*(srxzp(:,i)+omyp(:,i)) &
    )/4.0d0
    dytp(:,i) = -(   vp(:,i)* sryyp(:,i) &
      + up(:,i)*(srxyp(:,i)+omzp(:,i)) &
      + wp(:,i)*(sryzp(:,i)-omxp(:,i)) &
    )/4.0d0
    dztp(:,i) = -(   wp(:,i)* srzzp(:,i) &
      + up(:,i)*(srxzp(:,i)-omyp(:,i)) &
      + vp(:,i)*(sryzp(:,i)+omxp(:,i)) &
    )/4.0d0

!c--- -(div (uu))/2 part : gradient will be applied later

    rxxp(:,i) = -up(:,i)*up(:,i)/2.d0
    rxyp(:,i) = -up(:,i)*vp(:,i)/2.d0
    rxzp(:,i) = -up(:,i)*wp(:,i)/2.d0
    ryyp(:,i) = -vp(:,i)*vp(:,i)/2.d0
    ryzp(:,i) = -vp(:,i)*wp(:,i)/2.d0
    rzzp(:,i) = -wp(:,i)*wp(:,i)/2.d0

enddo

call xyzfftps( dxtp, fn )
call xyzfftps( dytp, gn )
call xyzfftps( dztp, hn )
call xyzfftps( rxxp, c_fnxx )
call xyzfftps( rxyp, c_fnxy )
call xyzfftps( rxzp, c_fnxz )
call xyzfftps( ryyp, c_fnyy )
call xyzfftps( ryzp, c_fnyz )
call xyzfftps( rzzp, c_fnzz )

g = dt/(2*re_tau) * beta ! for newtonian fluid beta = 1.d0

do i = 1, kcomy

    fn(:,i) = fn(:,i) + x_der_1(c_fnxx(:,i),i) +
    y_der_1(c_fnxy(:,i)) + z_der_1(c_fnxz(:,i),i),i) +
    gn(:,i) = gn(:,i) + x_der_1(c_fnyy(:,i)) +
    y_der_1(c_fnxy(:,i)) + z_der_1(c_fnyz(:,i),i),i) +
    hnn(:,i) = hnn(:,i) + x_der_1(c_fnxz(:,i),i) +
    y_der_1(c_fnxy(:,i)) + z_der_1(c_fnzz(:,i),i),i) +

!c--- adams - bashforth integration

    u(:,i) = u(:,i) + con_1*fn(:,i) + con_2*fnm(:,i)
    v(:,i) = v(:,i) + con_1*gn(:,i) + con_2*gnm(:,i)
    w(:,i) = w(:,i) + con_1*hnn(:,i) + con_2*hnm(:,i)
!c--- save the convection term for the next time-step in adams-bashford

\[
\begin{align*}
\text{fnm}( :, i ) &= \text{fn}( :, i ) \\
\text{gnm}( :, i ) &= \text{gn}( :, i ) \\
\text{hnm}( :, i ) &= \text{hn}( :, i )
\end{align*}
\]

!c--- add viscous corrections to u(n), v(n), w(n)

\[
\begin{align*}
\text{u}( :, i ) &= \text{u}( :, i ) + \\
& \quad g * ( x \_\text{der} \_2(\text{us}( :, i ), i ) + y \_\text{der} \_1( y \_\text{der} \_1(\text{us}( :, i ))) + z \_\text{der} \_2(\text{us}( :, i ), i ))) \\
\text{v}( :, i ) &= \text{v}( :, i ) + \\
& \quad g * ( x \_\text{der} \_2(\text{vs}( :, i ), i ) + y \_\text{der} \_1( y \_\text{der} \_1(\text{vs}( :, i ))) + z \_\text{der} \_2(\text{vs}( :, i ), i ))) \\
\text{w}( :, i ) &= \text{w}( :, i ) + \\
& \quad g * ( x \_\text{der} \_2(\text{ws}( :, i ), i ) + y \_\text{der} \_1( y \_\text{der} \_1(\text{ws}( :, i ))) + z \_\text{der} \_2(\text{ws}( :, i ), i )))
\end{align*}
\]
enddo

!c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++
!c    stage 2
!c    the pressure step (n + 1/3 to n + 2/3)
!c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++

call get_pressure ( pressure, u, v, w, &
\text{pbctop}, \text{pbcbot}, a, \text{wn}, \text{wd}, \text{wl}, \text{wr}, &
\text{bctop}, \text{bcbot}, \text{ag}, \text{wng})

!c--- update velocity at (n+2/3)
forall( i = 1:kcomy )
\begin{align*}
\text{u}( :, i ) &= \text{u}( :, i ) - \text{x} \_\text{der} \_1( \text{pressure}( :, i ), i ) \\
\text{v}( :, i ) &= \text{v}( :, i ) - \text{y} \_\text{der} \_1( \text{pressure}( :, i ) ) \\
\text{w}( :, i ) &= \text{w}( :, i ) - \text{z} \_\text{der} \_1( \text{pressure}( :, i ), i )
\end{align*}
endforall

!c--- apply constant pressuregradient in x-direction
!c if ( mynum==0 ) u(1,1) = u(1,1) + dt

!c--- pressure gradient change
if ( mynum==0 ) then
\begin{align*}
\text{time} \_\text{region} &= 1 \\
\text{if ( time.ge.time_s .and. time.lt.time_f) time} \_\text{region} &= 2 \\
\text{if ( time.ge.time_f ) time} \_\text{region} &= 3
\end{align*}
\begin{align*}
\text{select case(time} \_\text{region)} \\
\text{case}(1): \\
\text{dpx} \_\text{time} &= 1.0 \\
\text{case}(2): \\
\text{dpx} \_\text{time} &= ((\text{re} \_\text{tau} \_\text{final}/\text{re} \_\text{tau})^2 - 1.0)/(\text{time} \_f-\text{time} \_s) \times (\text{time} \_f-\text{time} \_s)+1.0 \\
\text{case}(3): \\
\text{dpx} \_\text{time} &= (\text{re} \_\text{tau} \_\text{final}/\text{re} \_\text{tau})^2
\end{align*}
endselect

end
end select

\[ u(1,1) = u(1,1) + dt \times dpdx\_time \]

endif

!c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ 
+++++
!c    stage 3
!c    calculate velocities at \((n + 1)\)
!c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ 
+++++

!c--- \(u^{(n+1)}\)
\[ sg = -2.0d0 \times \text{re\_tau}/dt/beta \]
forall( i = 1:kcomy )
\[ \text{temp}(;i) = sg \times u(;i) \]
\[ \text{temp}(nyp-3:nyp,i) = (0.d0,0.d0) \]
endforall

in = 0; g = 2.0d0 \times \text{re\_tau}/dt/beta; ib = 0
call solve( temp, u, \& ! input/output
g, dyde, in, \& ! input
bctop, bcbot,ib,wavz, wavx, ag, wng, wd, wl,
wr ) ! input/output

!c--- \(v^{(n+1)}\)
\[ sg = -2.0d0 \times \text{re\_tau}/dt/beta \]
forall( i = 1:kcomy )
\[ \text{temp}(;i) = sg \times v(;i) \]
\[ \text{temp}(nyp-3:nyp,i) = (0.d0,0.d0) \]
endforall

in = 0; g = 2.0d0 \times \text{re\_tau}/dt/beta; ib = 0
call solve( temp, v, \& ! input/output
g, dyde, in, \& ! input
bctop, bcbot,ib,wavz, wavx, ag, wng, wd, wl,
wr ) ! input/output

!c--- \(w^{(n+1)}\)
\[ sg = -2.0d0 \times \text{re\_tau}/dt/beta \]
forall( i = 1:kcomy )
\[ \text{temp}(;i) = sg \times w(;i) \]
\[ \text{temp}(nyp-3:nyp,i) = (0.d0,0.d0) \]
endforall

in = 0; g = 2.0d0 \times \text{re\_tau}/dt/beta; ib = 0
call solve( temp, w, \& ! input/output
g, dyde, in, \& ! input
bctop, bcbot,ib,wavz, wavx, ag, wng, wd, wl,
wr ) ! input/output

!c---------------------------------------------------------------
--
!c    output of data required for restart

86
dump_data = .false.
dump_data = ((mod(it,idmpfrq) == 0) .or. (it == nsteps + n_ini))

if (time.ge.time_s .and. time.le.time_f) then
    if ( mod( int((time-time_s)/dt+0.5)*60, int((time_f-time_s)/dt+0.5) )==0 &
        ) dump_data  = .true.
endif

if ( dump_data  &
    !     if ( mod(it,idmpfrq) == 0 .or. it == nsteps + n_ini &
    !         if ( mod(it-1,idmpfrq) == 0 .or. mod(it+1,idmpfrq) == 0 &
    !c        ) then
    call dooutputs( u, nameout(1), it )
call dooutputs( v, nameout(2), it )
call dooutputs( w, nameout(3), it )
call dooutputs( pressure/dt, nameout(10), it )
    !                   call dooutputs( fnm, nameout(11), it )
    !                   call dooutputs( gnm, nameout(12), it )
    !                   call dooutputs( hnm, nameout(13), it )
    if ( solve_fenep_model ) then
        call dooutputs( cxx, nameout(4), it )
call dooutputs( cxy, nameout(5), it )
call dooutputs( cxz, nameout(6), it )
call dooutputs( cyy, nameout(7), it )
call dooutputs( cyz, nameout(8), it )
call dooutputs( czz, nameout(9), it )
    !                   call dooutputs( c_fnmxx, nameout(14), it )
    !                   call dooutputs( c_fnmxy, nameout(15), it )
    !                   call dooutputs( c_fnmxz, nameout(16), it )
    !                   call dooutputs( c_fnmyy, nameout(17), it )
    !                   call dooutputs( c_fnmyz, nameout(18), it )
    !                   call dooutputs( c_fnmzz, nameout(19), it )
    endif
endif
enddo  ! time-steps

!c--- cpu time for all processors
call cpu_time( cpu_end )
cpu_proc = cpu_end - cpu_start

if ( nproc > 1 ) then
    call mpi_reduce( cpu_proc, cpu_sum, 1, mpi_real, &
        mpi_sum, 0, mpi_comm_world, ierr)
call mpi_reduce( cpu_proc, cpu_max, 1, mpi_real, &
else
  cpu_sum = cpu_proc
  cpu_max = cpu_proc
endif

if ( mynum == 0 ) then
  write(*,*)
  write(*,*) ' total cpu time over all processors =',int(cpu_sum/60.+1),' mins'
  write(*,*) ' wall clock time =',int(cpu_max/60.+1),' mins'
  write(*,*)
endif

!c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++
!c    end main loop
!c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++

if ( nproc > 1 ) then
  call mpi_barrier( mpi_comm_world, ierr )
  call mpi_finalize( ierr )
endif
end program main

!c======================================================================
subroutine setstuf
use parameters
use wave_numbers_stuf
use general_stuf
implicit none
integer mynum
common/cbpar2/ mynum
integer j, jstart, jz, k, keff, nxi, nyi, nzi
real(8) wavzall(nz)
real(8) alpha, bhta, rj, rn

!c---------------------------------------------------------------------
!c    calculate the resolvable wave nos. in x:
!c    assumes length xl has been non-dimensionalized with the length yhl.
!c---------------------------------------------------------------------
pi = 4.d0 * datan(1.d0)
if ( scale_by_pi ) then

alpha = 2.0d0/xl
bhta  = 2.0d0/zl
else
  alpha = 2.0d0*pi/xl
  bhta  = 2.0d0*pi/zl
endif

!c---------------------------------------------------------------------
--
!c    each processor, in the spectral domain consists of data in
!c    the form : complex a(nyp,kcomy) = a(nyp,nkz,kxh)
!c    so for the x-derivative, between 1 and nxh/nproc wave numbers
!c    are used per processor for nproc larger and smaller than
!c    nxh respectively
!c    for the z derivatives, between (nz*nxh)/nproc and nz wave numbers
!c    are used per processor for nproc larger and smaller than
!c    nxh respectively
!c---------------------------------------------------------------------
--
do k = 1, kxh
  keff  = (mynum*nxh)/nproc + k - 1
  wavx(k) = dfloat(keff)*alpha
  cwavx(k) = dcmplx(0.0d0, 1.0d0)*wavx(k)
  wavx2(k) = -wavx(k)*wavx(k)
enddo

!c---------------------------------------------------------------------
--
!c    first calculate all possible wavz
!c    then select proper values for current processor mynum
!c    distinguish nproc > nxh and nproc <= nxh
!c---------------------------------------------------------------------
--
do j = 1, max0(1,nz/2)
  wavzall(j) = dfloat(j-1)*bhta
enddo
do j = nz/2 + 1, nz
  wavzall(j) = dfloat(j-2*(nz/2)-1)*bhta
enddo
if ( kxh == 1 ) then
  jz = mod(mynum,kproc_yz)
  jstart = jz * kcomy
else
  jstart = 0
endif
do j = 1, nkz
  wavz(j) = wavzall(j+jstart)
  cwavz(j) = dcmplx(0.0d0, 1.0d0)*wavz(j)
  wavz2(j) = -wavz(j)*wavz(j)
enddo
end subroutine setstuf

!c=====================================================================
subroutine diverg( u, v, w, div )
!
!c---------------------------------------------------------------------
--
!c    this subroutine calculates fourier/chebyshev coeff of divergence
!c    input and output are fourier/chebyshev coefficients.
!c---------------------------------------------------------------------
--
use parameters, only : nyp,kcomy
use new_derivatives
implicit none
complex(8),intent(in),  dimension (nyp,kcomy) :: u, v, w
complex(8),intent(out), dimension (nyp,kcomy) :: div
integer :: i
forall( i = 1:kcomy )
div(:,i)=x_der_1(u(:,i),i)+y_der_1(v(:,i))+z_der_1(w(:,i),i)
endforall
end subroutine diverg
!
!c=====================================================================
==
subroutine vort1( ux, uy, uz, omxp, omyp, omzp )
!
!c---------------------------------------------------------------------
--
!c    calculate vorticity components and transformed to physical values
!c---------------------------------------------------------------------
--
use parameters
use new_derivatives
use xyzfft
implicit none
complex(8),intent(in), dimension (nyp,kcomy):: ux, uy, uz
real(8)   ,intent(out),dimension (nx,kcomx) :: omxp,omyp,omzp
complex(8),dimension (nyp,kcomy) :: temp
integer :: i
forall( i = 1:kcomy )
    temp(:,i) = x_der_1(uy(:,i),i) - y_der_1(ux(:,i))
endforall
call xyzfftsp(temp, omzp)
forall( i = 1:kcomy )
    temp(:,i) = y_der_1(uz(:,i)) - z_der_1(uy(:,i),i)
endforall
call xyzfftsp(temp, omxp)
forall( i = 1:kcomy )
    temp(:,i) = z_der_1(ux(:,i),i) - x_der_1(uz(:,i),i)
endforall
call xyzfftsp(temp, omyp)
end subroutine vort1

!c=====================================================================
===
subroutine gadot(ux,uy,uz,srxxp,srxyp,srxzp,sryyp,sryzp,srzsp)
!c---------------------------------------------------------------
--
!c calculate strain-rate tensors and transformed to physical values
!c (e.g.) srxyp = dux/dy + duy/dx
!c---------------------------------------------------------------
--
use parameters
use new_derivatives
use xyzfft
implicit none

complex(8),intent(in),dimension(nyp,kcomy) :: ux, uy, uz
real(8),  intent(out),dimension(nx,kcomx)  :: srxxp, srxyp, srxzp
&
sryyp, sryzp, srzsp

complex(8),dimension(nyp,kcomy) :: temp
integer :: i
forall( i = 1:kcomy )
temp(:,i) = 2.d0*x_der_1(ux(:,i),i)
endforall
call xyzfftsp(temp,srxxp)
forall( i = 1:kcomy )
temp(:,i) = 2.d0*y_der_1(uy(:,i))
endforall
call xyzfftsp(temp,sryyp)
forall( i = 1:kcomy )
temp(:,i) = 2.d0*z_der_1(uz(:,i),i)
endforall
call xyzfftsp(temp,srzsp)
forall( i = 1:kcomy )
temp(:,i) = x_der_1(uy(:,i),i) + y_der_1(ux(:,i))
endforall
call xyzfftsp(temp,srxyp)
forall( i = 1:kcomy )
temp(:,i) = x_der_1(uz(:,i),i) + z_der_1(ux(:,i),i)
endforall
call xyzfftsp(temp,srxzp)
forall( i = 1:kcomy )
temp(:,i) = z_der_1(uy(:,i),i) + y_der_1(uz(:,i))
endforall
call xyzfftsp(temp,sryzp)
end subroutine gadot

!c=====================================================================
==
subroutine var_scatter( varo, nameins )
use parameters
implicit none
include 'mpif.h'

complex(8) varo(nyp,kcomy)
character(len=*), intent(in) :: nameins

integer mynum
common/cbpar2/ mynum

complex(8), dimension (nyp,nz,nxh) :: vari
integer i, icomy, ierr, j, k

if ( mynum == 0 ) then
  write(*,*) ' reading initial data : ', nameins
  open(21,file=nameins,status="old",action="read",form='unformatted')
  read(21) (((vari(j,k,i),j=1,nyp),k=1,nz),i=1,nxh)
  close(21)
endif

if ( nproc > 1 ) then
  call mpi_barrier( mpi_comm_world, ierr )
  call mpi_scatter( vari, kdata, mpi_double_complex, varo, kdata, mpi_double_complex, &
                 0, mpi_comm_world, ierr )
elseif ( nproc == 1 ) then
  do i = 1, nxh
    do k = 1, nz
      do j = 1, nyp
        icomy = k + (i-1)*nz
        varo(j,icomy) = vari(j,k,i)
      enddo
    enddo
  enddo
endif

end subroutine var_scatter

!c=====================================================================
subroutine dooutputs( qs, nameouts, it )
!c---------------------------------------------------------------------
!c    gather the data of the processors to processor 0 for output
!c    and write in a file
!c---------------------------------------------------------------------
use parameters
use general_stuf
use xyzfft
implicit none
include 'mpif.h'

complex(8), intent(in), dimension (nyp,kcomy) :: qs
character(len=70), intent(in) :: nameouts
character(len=70)            :: filename

integer mynum
common/cbpar2/ mynum
complex(8), dimension (nyp,nz,nxh) :: qsall
integer :: i, icomy, ierr, j, k, it

if ( nproc > 1 ) then
  call mpi_barrier( mpi_comm_world, ierr )
  call mpi_gather( qs, kdata, mpi_double_complex, qsall,
    kdata, mpi_double_complex, 0, &
    mpi_comm_world, ierr )
elseif ( nproc == 1 ) then
  do i = 1, nxh
    do k = 1, nz
      do j = 1, nyp
        icomy = k + (i-1)*nz
        qsall(j,k,i) = qs(j,icomy)
      enddo
    enddo
  enddo
endif

if ( mynum == 0 ) then
  filename = nameouts
  i=index(filename,'.')
  !           write(unit=filename(i+1:),fmt='(bn,i5.5)') it
  !           write(unit=filename(i+1:),fmt='(bn,i6.6)') it
  write(unit=filename(i+1:),fmt='(bn,i7.7)') it
  write(*,*) 'writing file: ', filename
  open(31,file=filename,status="unknown",action="write",form="unformatted"
  )
  write(31) (((qsall(j,k,i),j=1,nyp),k=1,nz),i=1,nxh)
  close(31)
endif

if ( nproc > 1 ) call mpi_barrier( mpi_comm_world, ierr )
end subroutine dooutputs

!c=====================================================================
!      subroutine mean_reynolds( spec, name, time )
subroutine mean_reynolds_number( re_m, spec, time )
use parameters
use general_stuf
implicit none

complex(8), intent(in), dimension (nyp,kcomy) :: spec
! character(len=*), intent(in) :: name
real(8)  time

real(8) :: re_m  ! = u_m*(2h)/nu
integer :: iy

re_m = 0.d0
do iy = 0, nyp-1, 2
   re_m = re_m + real(spec(iy+1,1),8)/dble(1-iy*iy)*2.0d0
enddo
re_m = re_m * re_tau

open (10, file=trim(folder_out)//'time_hist_re_m.dat',
position="append", action="write")
write(10,"(e15.9,x,e15.9)") time, re_m
close(10,status="keep")

end subroutine mean_reynolds_number

!c=====================================================================

!c=====================================================================
subroutine cfl_number( cfl_max, up, vp, wp )
use parameters
use general_stuf
implicit none
include 'mpif.h'

real(8), dimension (1:nx,1:kcomx) :: up, vp, wp
real(8) :: cfl_max, cfl_max_proc, cfl_local
real(8) :: delta_x, delta_z, delta_y
integer i, j, k, icomx, ierr

pi = 4.d0 * datan(1.d0)

if ( scale_by_pi ) then
delta_x = xl * pi / dble(nx)
delta_z = zl * pi / dble(nz)
else
delta_x = xl / dble(nx)
delta_z = zl / dble(nz)
endif

cfl_max_proc = 0.d0

do icomx = 1, kcomx
   do i = 1, nx
      j = mod( icomx - 1, nyp) + 1

      delta_y = cos(dble(j-1)*pi/dble(ny)) &
                 - cos(dble(j  )*pi/dble(ny))

      cfl_local = abs(up(i,icomx))/delta_x &
                  + abs(vp(i,icomx))/delta_y &
+ abs(wp(i,icomx))/delta_z
if ( cfl_local .gt. cfl_max_proc )
cfl_max_proc = cfl_local
enddo
enddo

cfl_max_proc = cfl_max_proc * dt

if ( nproc > 1 ) then
  call mpi_reduce( cfl_max_proc, cfl_max, 1,
    mpi_double_precision, &
    mpi_max, 0, mpi_comm_world, ierr)
else
  cfl_max = cfl_max_proc
endif
end subroutine cfl_number

!c=====================================================================
subroutine time_history( up, vp, wp, time )
use parameters
use general_stuf
implicit none
real(8), dimension (1:nx,1:kcomx) :: up, vp, wp
real(8) time
integer :: i, j, k, icomx(4)
!c--- monitoring points
i = 1
k = 1  ! should be less than kz
j = 2
icomx(1) = (k - 1)*nyp + j
j = 10
icomx(2) = (k - 1)*nyp + j
j = 30
icomx(3) = (k - 1)*nyp + j
j = 65
icomx(4) = (k - 1)*nyp + j
open (10, file=trim(folder_out)//'time_hist_u.dat',
  position="append", action="write")
write(10, 100) time, (up(i,icomx(k)),k=1,4)
close(10, status="keep")
open (10, file=trim(folder_out)//'time_hist_v.dat',
  position="append", action="write")
write(10, 100) time, (vp(i,icomx(k)),k=1,4)
close(10, status="keep")
open (10, file=trim(folder_out)//'time_hist_w.dat',
    position="append", action="write")
write(10, 100) time, (wp(i,icomx(k)),k=1,4)
close(10, status="keep")

100 format(5(e15.9,x))

end subroutine time_history

!c=====================================================================
subroutine divergence( div_max, srxxp, sryyp, srzzp )
use parameters
use general_stuf
implicit none
include 'mpif.h'
real(8), dimension (1:nx,1:kcomx) :: srxxp,sryyp,srzzp
real(8) :: div_max, div_max_proc, div_local
integer :: i, icomx, ierr

div_max_proc = 0.d0

do icomx = 1, kcomx
   do i = 1, nx
      div_local = srxxp(i,icomx) + sryyp(i,icomx) +
                  srzzp(i,icomx)
      if ( div_local .gt. div_max_proc )  div_max_proc =
         div_local
   enddo
endo

div_max_proc = 0.5 * div_max_proc

if ( nproc > 1 ) then
   call mpi_reduce( div_max_proc, div_max, 1,
   mpi_double_precision, &
      mpi_max, 0, mpi_comm_world, ierr)
else
   div_max = div_max_proc
endif

end subroutine divergence
APPENDIX D
This code defines all the parameters used in APPENDIX C.

```plaintext
!======================================================================
module parameters
!======================================================================

!----------------------------------------------------------------------
!
! parameters for a specific problem
!
! nype array size in y-direction
!
! nproc number of processors, which has to fulfil
! 1) nxh * nz / nproc is integer
! 2) nxh / nproc is integer
! 3) nype / nproc is integer
! 4) nz / nproc is integer
!
!----------------------------------------------------------------------
!
integer, parameter :: nproc = 64    ! number of processors
integer, parameter :: nx = 128  ! number of points in x-direction
integer, parameter :: ny = 128 ! number of points in y-direction
integer, parameter :: nz = 128 ! number of points in z-direction
integer, parameter :: nxh = nx/2
integer, parameter :: nyh = ny/2
integer, parameter :: nyp = ny+1
!
integer, parameter :: nyp_n = ny_n+1
integer, parameter :: nype = nyp
!
!----------------------------------------------------------------------
!
! derived parameters
!
! kcomx number of yz data per proc for x-array
!
! kcomy number of zx data per proc for y-array
!
! kcomz number of xy data per proc for z-array
!
! kdata number of data per proc
!
! kproc_yz number of procs for the yz communication
!
! kproc_zx number of procs for the yz communication = nproc /
!
! kproc_zx
!
!----------------------------------------------------------------------
!
integer, parameter :: kcomx = (nz*nype)/nproc
integer, parameter :: kcomy = (nz*nxh)/nproc
integer, parameter :: kcomz = (nype*nxh)/nproc
integer, parameter :: kdata = nyp*kcomy
integer, parameter :: kproc_yz = 1+(nproc-1)/nxh
integer, parameter :: kproc_zx = nxh
!
integer, parameter :: kxh = 1 + (kcomy-1)/nz
integer, parameter :: kz = nz / nproc
integer, parameter :: nkz = kcomy/kxh
!
end module parameters
```

98
module general_stuf
!
character*8 :: dirarg
! integer :: iargc
!
character(len=100),parameter::folder_in="ic_data"//trim(dirarg)//"/
! character(len=100),parameter::folder_out="/../scratch/output"//trim(dirarg)//"/
    character(len=100) :: folder_in
    character(len=100) :: folder_out
real(8), parameter :: re_tau = 395.d0  ! friction reynolds number
integer, parameter :: nsteps = 10000 ! total time steps
integer, parameter :: idmpfrq = 500 ! frequency to dump restart files
    real(8), parameter :: dt = 1.25e-04 ! time step for integration
real(8), parameter :: re_tau_final = 395.d0 ! re_tau will be changed to this value
real(8), parameter :: time_s = 1d8 ! re_tau will be changed from this time
real(8), parameter :: time_f = 1d8 + 10.0 ! re_tau change will be terminated at this time
logical, parameter :: scale_by_pi = .true. ! if true, actual xl = xl*pi
    real(8), parameter :: xl = 2.0d0 ! length in x - periodic direction
    real(8), parameter :: zl = 1.0d0 ! length in z - periodic direction
    real(8), parameter :: yhl = 2.0d0 ! length in nonhomogeneous (y) direction
    real(8), parameter :: dyde = 2.0d0/yhl ! y-scaling factor
real(8) :: pi

!----------------------------------------------------------------------
--
fenep model parameters
!----------------------------------------------------------------------
--
logical, parameter :: solve_fenep_model = .false. ! if false, set beta to be 1.d0
    real(8), parameter :: we_tau = 25.00d0
    real(8), parameter :: beta = 1.00d0
    real(8), parameter :: lmax = 30.00d0 ! b=lmax**2
real(8), parameter :: diffusivity = 0.02d0
real(8), parameter :: we = we_tau / re_tau
real(8), parameter :: diffusivity_factor = 2.0 / dt /
diffusivity
real(8), parameter :: lmax_square = lmax * lmax
end module general_stuf

!c=================================================================================
==
module wave_numbers_stuf
!c=================================================================================
==
use parameters, only : nkz, kxh
implicit none
private
public :: wavz, wavx, cwavz, cwavx, wavz2, wavx2
real(8), dimension (nkz) :: wavz
real(8), dimension (kxh) :: wavx
complex(8), dimension (nkz) :: cwavz
complex(8), dimension (kxh) :: cwavx
real(8), dimension (nkz) :: wavz2
real(8), dimension (kxh) :: wavx2
end module wave_numbers_stuf

!c=================================================================================
=====
module new_derivatives
!c=================================================================================
=====
use parameters
use wave_numbers_stuf
implicit none
private
public :: x_der_1, x_der_2, y_der_1, z_der_1, z_der_2
contains
!----------------------------------------------------------------------
--------
!              fisrt and second derivatives in x-dir
!              f and df are fourier/chebyshev coefficients.
!----------------------------------------------------------------------
--------
complex(8) pure function x_der_1( f, icomy ) result (df)
implicit none

integer, intent(in) :: icomy
complex(8), intent(in), dimension (1:nyp) :: f
dimension df(1:nyp)
integer ix

ix = ( icomy - 1 ) / nz + 1

df = cwavx(ix)*f

end function x_der_1

complex(8) pure function x_der_2( f, icomy ) result ( ddf )

implicit none

integer, intent(in) :: icomy
complex(8), intent(in), dimension (1:nyp) :: f
dimension ddf(1:nyp)
integer ix

ix = ( icomy - 1 ) / nz + 1

ddf = wavx2(ix)*f

end function x_der_2

!----------------------------------------------------------------------
!               df = df/dy   (y = the chebyshev direction)
!               f and df are fourier/chebyshev coefficients.
!----------------------------------------------------------------------

complex(8) pure function y_der_1( f ) result ( df )

implicit none

complex(8), intent(in), dimension (1:nyp) :: f
dimension df(1:nyp)
integer :: iy

df(ny+1) = (0.0d0,0.0d0)
df(ny)   = dble(2*ny)*f(nyp)
do iy = ny-1, 2, -1
   df(iy) = df(iy+2) + dble(2*iy)*f(iy+1)
enddo
df(1) = 0.5d0*df(3) + f(2)

end function y_der_1

!----------------------------------------------------------------------
! first and second derivatives in z-dir

!----------------------------------------------------------------------
complex(8) pure function z_der_1(f, icomy) result(df)
implicit none
integer, intent(in) :: icomy
complex(8), intent(in), dimension (1:nyp) :: f
dimension df(1:nyp)
integer ix, iz
iz = mod(icomy-1, nz) + 1
ix = (icomy - 1) / nz + 1
df = cwavz(iz)*f
end function z_der_1

complex(8) pure function z_der_2(f, icomy) result(ddf)
implicit none
integer, intent(in) :: icomy
complex(8), intent(in), dimension (1:nyp) :: f
dimension ddf(1:nyp)
integer ix, iz
iz = mod(icomy-1, nz) + 1
ix = (icomy - 1) / nz + 1
ddf = wavz2(iz)*f
end function z_der_2

end module new_derivatives
APPENDIX E

VISUALIZATION CODE
This code calculates $\lambda_{ci}$ (complex eigen value of velocity gradient tensor) for visualizing vortices. It also writes the output velocity files in a readable (tecplot) format.

Grid: 128 x 129 x 128;

$Re = 395$;

Language: Fortran 95;

Machine it ran on: Saguaro (ASU high performance computing center);

Number of processors: 1.

Input parameters: u, v, and w components of velocity, components of velocity gradient tensor

Output parameters: $\lambda_{ci}$ for various $t^+$. 

```fortran
C--- write relative value of lambda_ci to its maximum at each flow field
C
program channel_post

include 'param.h'

common/mesh/y(nyp), dx, dz
common/domain/sx, sz
common/para/re
common/nstep/n_start, n_final, n_skip

! directory input (JRB)
character*30 :: curdir
integer :: iargc

if (iargc().ne.1) stop "must set argument: <program> <#####>
Call getarg(1, curdir)
Write(*,*) "current directory > ", trim(curdir)

C--- simulation parameters
re = 395.
! re = 180.
! re = 110.

pi = acos(-1.0)
! sx = 4.*pi !2.*pi !4.*pi !2.*pi ! 4.*pi
! sz = 4.*pi/3 !1.*pi !4.*pi/3 !1./1.*pi 1./1.*pi
```
sx=2.*pi
sz=1.*pi

c---------------------------------------------
!      read(*,*) n_start
!      read(*,*) n_final
!      read(*,*) n_skip

n_start = 500 !0
n_final = 10000 !5000   !1000
n_skip  = 500 !100 !50

C---------------------------------------------
call get_grid

call calc_rci(curdir)  ! calculate lambda_ci

stop
end

C------------------------------------------------------------------
subroutine get_grid
include 'param.h'
common/mesh/y(nyp),dx,dz
common/domain/sx,sz
common/para/re
pi = acos(-1.0)
do j=1,nyp
   y(j) = 1.0-cos(pi*real(j-1)/real(nyp-1))
enddo

dx = sx/real(nx)
dz = sz/real(nz)
return
end

C------------------------------------------------------------------
subroutine calc_rci(curdir)
include 'param.h'
common/mesh/y(nyp),dx,dz
common/domain/sx,sz
common/para/re
common/nstep/n_start, n_final, n_skip
character*50 filename
character*30 :: curdir
real*8 d11(nx,nyp,nz),d12(nx,nyp,nz),d13(nx,nyp,nz)
real*8 d21(nx,nyp,nz),d22(nx,nyp,nz),d23(nx,nyp,nz)
real*8 d31(nx,nyp,nz),d32(nx,nyp,nz),d33(nx,nyp,nz)
real*8 q1(nx,nyp,nz)
real*8 q2(nx,nyp,nz)
real*8 q3(nx,nyp,nz)
real*8 uf(nx,nyp,nz)
real*8 e11,e12,e13,e21,e22,e23,e31,e32,e33
real*8 p,q,r,q0,r0,dis,reg1,reg2,reg3
real*8 p_max
real*8 ramda_ci(nx,nyp,nz)
real*8 r_ci_max

c
real*8 q1_xz(nyp)
real*8 q2_xz(nyp)
c
TECPOIT STUFF
integer i,j,k,imax,jmax,kmax
integer debug,ier,itot
integer tecini,tecdat,teczne,tecnod,tecfil,tecend
integer visdouble,disdouble
character*1 nulchar
real*8 xt(nx,nyp,nz)
real*8 yt(nx,nyp,nz)
real*8 zt(nx,nyp,nz)
nulchar = char(0)
dependence = 0
visdouble = 0
disdouble = 1
imax = nx
jmax = nyp
kmax = nz

do 90 k=1,nz
   do 90 j=1,nyp
       do 90 i=1,nx
          c--- with Fortran 90 we can just fill the arrays...
          xt(i,j,k) = real(i-1)*dx*re
          yt(i,j,k) = y(j)*re
          zt(i,j,k) = real(k-nz/2)*dz*re
         90      continue

do ntime=n_start,n_final,n_skip
   c--- read dij
   call get_filename_dij(filename,ntime,1,1,curdir)
   write(*,*) filename
   open(10,file=filename,status='old',form='unformatted'
      ,action='read')
   read(10) (((d11(i,j,k),i=1,nx),j=1,nyp),k=1,nz)
   close(10)
   call get_filename_dij(filename,ntime,1,2,curdir)
   write(*,*) filename
   open(10,file=filename,status='old',form='unformatted'
      ,action='read')
read(10) (((d12(i,j,k),i=1,nx),j=1,nyp),k=1,nz)
close(10)
call get_filename_dij(filename,ntime,1,3,curdir)
write(*,*) filename
open(10,file=filename,status='old',form='unformatted'
&         ,action='read')
read(10) (((d13(i,j,k),i=1,nx),j=1,nyp),k=1,nz)
close(10)
call get_filename_dij(filename,ntime,2,1,curdir)
write(*,*) filename
open(10,file=filename,status='old',form='unformatted'
&         ,action='read')
read(10) (((d21(i,j,k),i=1,nx),j=1,nyp),k=1,nz)
close(10)
call get_filename_dij(filename,ntime,2,2,curdir)
write(*,*) filename
open(10,file=filename,status='old',form='unformatted'
&         ,action='read')
read(10) (((d22(i,j,k),i=1,nx),j=1,nyp),k=1,nz)
close(10)
call get_filename_dij(filename,ntime,2,3,curdir)
write(*,*) filename
open(10,file=filename,status='old',form='unformatted'
&         ,action='read')
read(10) (((d23(i,j,k),i=1,nx),j=1,nyp),k=1,nz)
close(10)
call get_filename_dij(filename,ntime,3,1,curdir)
write(*,*) filename
open(10,file=filename,status='old',form='unformatted'
&         ,action='read')
read(10) (((d31(i,j,k),i=1,nx),j=1,nyp),k=1,nz)
close(10)
call get_filename_dij(filename,ntime,3,2,curdir)
write(*,*) filename
open(10,file=filename,status='old',form='unformatted'
&         ,action='read')
read(10) (((d32(i,j,k),i=1,nx),j=1,nyp),k=1,nz)
close(10)
call get_filename_dij(filename,ntime,3,3,curdir)
write(*,*) filename
open(10,file=filename,status='old',form='unformatted'
&         ,action='read')
read(10) (((d33(i,j,k),i=1,nx),j=1,nyp),k=1,nz)
close(10)

ramda_ci(:,:,:,:) = 0.0d0 ! ramda_ci

do 1 j=1,nyp
do 1 k=1,nz
do 1 i=1,nx
   e11 = d11(i,j,k)
e12 = d12(i,j,k)
e13 = d13(i,j,k)
e21 = d21(i,j,k)
e22 = d22(i,j,k)

107
\[ e_{23} = d_{23}(i,j,k) \]
\[ e_{31} = d_{31}(i,j,k) \]
\[ e_{32} = d_{32}(i,j,k) \]
\[ e_{33} = d_{33}(i,j,k) \]
\[ p = -(e_{11} + e_{22} + e_{33}) \]
\[ q = 0.5*(p**2 - (e_{11}**2 + e_{22}**2 + e_{33}**2 + e_{12}*e_{21}*2.0 + e_{13}*e_{31}*2.0 + e_{23}*e_{32}*2.0)) \]
\[ r = -(e_{13}*e_{22}*e_{31} + e_{12}*e_{23}*e_{31} + e_{13}*e_{21}*e_{32} - e_{11}*e_{23}*e_{32} - e_{12}*e_{21}*e_{33} + e_{11}*e_{22}*e_{33}) \]

if (abs(p).gt.p_max) then
    p_max = abs(p)
endif
r0 = r + 2./27.*p**3 - 1./3.*p*q
go = q - 1./3.*p**2
dis = (r0/2.0)**2 + (q0/3.0)**3
if (dis.gt.0.0) then
    reg1 = sqrt(dis)
    reg2 = reg1 - r0/2.0
    reg3 = reg1 + r0/2.0
    if (reg2 .gt. 0.0) then
        reg2 = reg2**(1./3.)
    else
        reg2 = -(-reg2)**(1./3.)
    endif
    if (reg3 .gt. 0.0) then
        reg3 = reg3**(1./3.)
    else
        reg3 = -(-reg3)**(1./3.)
    endif
    ramda_ci(i,j,k) = sqrt(3.)/2.0*(reg2 + reg3)
else
    ramda_ci(i,j,k) = 0.0
endif
1 continue
write(*,*) 'maximum du_i/dx_i*h/u_tau =', p_max

--- find the maximum r_ci
r_ci_max = 0.0
do k=1,nz
do j=1,nyp
do i=1,nx
    r_ci_max = amax1(r_ci_max, ramda_ci(i,j,k))
enddo
dendo
c       write(*,*) r_ci_max
!
   filename='rci'
!   nn=index(filename,'i')
!   write(unit=filename(nn+1:),fmt='(bn,i5.5)') ntime
!   write(*,*) filename
!   open(10,file=filename,status='unknown')
!c    &      ,',k=',nz/4*3-nz/4+1,',f=point'
!
   filename='rss'
!   nn=index(filename,'s')
!   write(unit=filename(nn+2:),fmt='(bn,i5.5)') ntime
!   write(*,*) filename
!   open(13,file=filename,status='unknown')
!c     &    ,',k=',nz/4*3-nz/4+1,',f=point'
!
   filename='xy'
!   nn=index(filename,'y')
!   write(unit=filename(nn+1:),fmt='(bn,i5.5)') ntime
!   write(*,*) filename
!   open(11,file=filename,status='unknown')
!c    &      ,',k=',1,',f=point'
!
   filename='xy_uf'
!   nn=index(filename,'f')
!   write(unit=filename(nn+1:),fmt='(bn,i5.5)') ntime
!   write(*,*) filename
!   open(12,file=filename,status='unknown')
!c     &     ,',k=',1,',f=point'
!
   c--- calculate x-z mean of u and fpi

   q1_xz(:) = 0.0
   q2_xz(:) = 0.0
   do j = 1, nyp
   do k = 1, nz
   do i = 1, nx
   q1_xz(j) = q1_xz(j) + q1(i,j,k)/real(nx*nz)
   q2_xz(j) = q2_xz(j) + q2(i,j,k)/real(nx*nz)
   enddo
   enddo
   enddo

   do j = 1, nyp
   uf(:,j,:) = q1(:,j,:) - q1_xz(j)
   end do
!
   c---

   do 2 k=nz/4,nz/4*3
do 2 j=1,nyp/2+1
do 2 i=1,nx
  rx=real(i-1)*dx*re
  rz=real(k-nz/2)*dz*re
  ry=y(j)*re
  c           write(10,100) rx,ry,rz,ramda_ci(i,j,k)
c           write(13,100) rx,ry,rz,
c &           (q1(i,j,k)-q1_xz(j))*(q2(i,j,k)-q2_xz(j))
  if( k .eq. nz/2 )  then
    rz2=real(nz/4-nz/2)*dz*re
    c               write(11,101) rx,ry,rz2
    c &                      ,q1(i,j,k)- 0.8*q1_xz((nyp+1)/2) !-0.8*20.157 ! substract 80% centerline velocity
    c &                      ,q2(i,j,k)
c &                      ,q3(i,j,k)
c! &                      ,0.0
    c               write(12,100) rx,ry,rz2
    c &                      ,q1(i,j,k)- q1_xz(j) ! substract xz mean velocity
  endif
  c &
  2      continue
100       format(4(e12.5,x))
101       format(6(e12.5,x))
!
! Write the zone header information.
!
ier = teczne('Velocity Field'/nulchar,
  &  'x,y,z,u,v,w,ufluc,dudx,dudy,dudz,dvdx,
  &  dvdy,dvdz,dwdx,dwdy,dwdz,lambdaci'/nulchar,
  &  filename(1:(nn-1))//'.plt'//nulchar,
  &  '.'//nulchar,
  &  debug,visdouble)

! Write the zone header information.
!
ier = teczne('Velocity Field'/nulchar,
  &  imax,jmax,kmax,
  &  'BLOCK'/nulchar,nulchar)
! Write out the field data.

itot = imax*jmax*kmax
ier = tecdat(itot,xt,disdouble)
ieri = tecdat(itot,yt,disdouble)
ieri = tecdat(itot,zt,disdouble)
ieri = tecdat(itot,q1,disdouble)
ieri = tecdat(itot,q2,disdouble)
ieri = tecdat(itot,q3,disdouble)
ieri = tecdat(itot,uf,disdouble)
ieri = tecdat(itot,d11,disdouble)
ieri = tecdat(itot,d12,disdouble)
ieri = tecdat(itot,d13,disdouble)
ieri = tecdat(itot,d21,disdouble)
ieri = tecdat(itot,d22,disdouble)
ieri = tecdat(itot,d23,disdouble)
ieri = tecdat(itot,d31,disdouble)
ieri = tecdat(itot,d32,disdouble)
ieri = tecdat(itot,d33,disdouble)
ieri = tecdat(itot,ramda_ci,disdouble)
ieri = tecend()
enddo ! ntime

treturn
end

subroutine get_vel(u,v,w,ntime,curdir)

character*30 :: curdir

real*8 u(nx,nyp,nz)
real*8 v(nx,nyp,nz)
real*8 w(nx,nyp,nz)

call get_filename_disk5(filename,ntime,1,curdir)
write(*,*) filename
open(10,file=filename,status='old',form='unformatted'
& ,action='read')
read(10) (((u(i,j,k),i=1,nx),j=1,nyp),k=1,nz)
close(10)
c--- read v

call get_filename_disk5(filename,ntime,2,curdir)
write(*,*) filename
open(10,file=filename,status='old',form='unformatted'
 & action='read')
read(10) (((v(i,j,k),i=1,nx),j=1,nyp),k=1,nz)
close(10)

c--- read w

call get_filename_disk5(filename,ntime,3,curdir)
write(*,*) filename
open(10,file=filename,status='old',form='unformatted'
 & action='read')
read(10) (((w(i,j,k),i=1,nx),j=1,nyp),k=1,nz)
close(10)

return
end

!c------------------------------------------------------------------
! subroutine get_filename_disk5(filename,ntime,nv)
! nv=1 : u
! 2 : v
! 3 : w
!
! character*50 filename
!
! filename='../../../scratch/data_vel/'
! nn=index(filename,'/')
! if (nv.eq.1) write(unit=filename(nn+10:),fmt='(bn,a5)') 'u1.00'
! if (nv.eq.2) write(unit=filename(nn+10:),fmt='(bn,a5)') 'u2.00'
! if (nv.eq.3) write(unit=filename(nn+10:),fmt='(bn,a5)') 'u3.00'
! write(unit=filename(nn+15:),fmt='(bn,i5.5)') ntime
!
! return
! end

!c------------------------------------------------------------------
subroutine get_filename_disk5(filename,iseq,nv,curdir)

nv=1 : u
2 : v
3 : w

implicit none
character*50 filename
integer iseq, nv, nn
character*30 curdir

if (nv.le.4) then
directory where DNS results are stored.

nn=index(filename,'@')
if (nv.eq.1) write(unit=filename(nn:],fmt='(bn,a3)') 'u1.'
if (nv.eq.2) write(unit=filename(nn:],fmt='(bn,a3)') 'u2.'
if (nv.eq.3) write(unit=filename(nn:],fmt='(bn,a3)') 'u3.'
if (nv.eq.4) write(unit=filename(nn:],fmt='(bn,a3)') 'pp.'
write(unit=filename(nn+3:],fmt='(bn,i7.7)') iseq
endif
return
end

!c------------------------------------------------------------------
!      subroutine get_filename_dij(filename,iseq,nv1,nv2)
!!c       nv1=1 : u  nv2 = x
!!c       nv1=2 : v  nv2 = y
!!c       nv1=3 : w  nv2 = z
!c       implicit none
!c       character*50 filename
!c       integer iseq, nv1,nv2,nn
!c       filename='../../../scratch/data_dij/d'
!c       nn=index(filename,'/')
!c       write(unit=filename(nn+11:],fmt='(bn,i1.1)') nv1
!c       write(unit=filename(nn+12:],fmt='(bn,i1.1)') nv2
!c       write(unit=filename(nn+13:],fmt='(bn,a1)') '.'
!c       write(unit=filename(nn+14:],fmt='(bn,i7.7)') iseq
!c       return
!c       end

!c------------------------------------------------------------------
 subroutine get_filename_dij(filename,iseq,nv1,nv2,curdir)
!
 implicit none
 character*50 filename
 integer iseq, nv1, nv2, nn
 character*30 curdir
!
 if (nv.le.4) then
 !filename1='..../data_dij/@'
 !filename2='..../data_dij/@'
 !filename3='..../data_dij/@'
 filename='../../../scratch/data_dij//trim(curdir)//'@'  !
 directory where DNS results are stored.
 nn=index(filename,'@')
 write(unit=filename(nn:],fmt='(bn,i1.1)') nv1
 write(unit=filename(nn+1:],fmt='(bn,i1.1)') nv2
 write(unit=filename(nn+2:],fmt='(bn,a1)') '.'
 write(unit=filename(nn+3:],fmt='(bn,i7.7)') iseq
end subroutine get_filename_dij(filename,iseq,nv1,nv2,curdir)
!endif

return
end