# Simulation and modal analysis of instability and transition in a steady eccentric stenotic flow

Master Thesis
January 2013

**Zeinab Moradi Nour**

Department of Mechanics
KTH Stockhom

Written with MiKTEX2.9, a pc implementation of LATEX

# Abstract

Direct numerical simulation (DNS) of steady flow, with $Re = 750$ at inlet, through stenosed pipe has been done to study transition and turbulence of the flow in the post-stenosis area. The pipe has 75% constriction by area reduction and 5% eccentricity of the main pipe diameter at the throat. A sinusoidal Gaussian force is implemented to keep turbulent in the domain. The result shows acceptable agreement with previous study has been done by Fischer et al. [4]. We simulated the case by Nek5000 which benefits from the spectral element method (SEM) as a higher order accurate method. To have a better understanding of the turbulent flow, we have done the modal decomposition to obtain coherent structures. Among several methods for modal decomposition, we considered Proper Orthogonal Decomposition (POD) and Dynamic Mode Decomposition (DMD) for current study. The methods have been implemented in Fortran, accelerated using OpenMP and is potentially settled for computation of large data sets. DMD implementation shows 2.5 speed up. The structures correspond to the implemented force are extracted by POD however they have not been recognized by dynamic decomposition.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background

Stenosis pipe flow, as a model for arterial stenosis, was the subject for many experimental ([3]) and computational studies with different hypothesis. The local restriction of the artery (heart vessels) which is a result of the accumulation of fatty-plaque in the walls of the large arteries, is called *Atherosclerosis* which can be a reason for Artery Disease, thrombosis, heart attack and stroke. Flow separation in combination with loosing pressure, can reduce the flow rate and cause localized absence of oxygen which is an indicator for surgical interferences. Stenoses are commonly characterized as a percentage reduction in diameter or area of the host vessel and are clinically considered significant when the area reduction is greater than 75%. In most cases, eccentricity is a characteristic of the arterial stenosis.

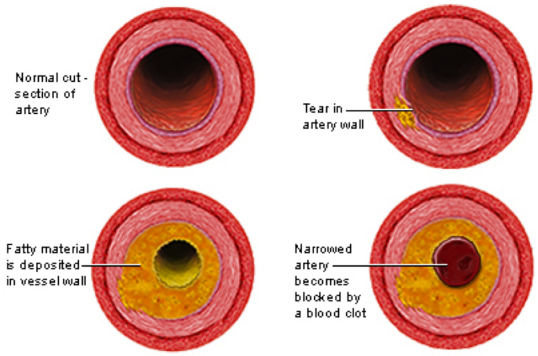



Figure 1.1: The local restriction of the artery. Borrowed from U.S. National Library of Medicine website.

The pulsatile (caused by cyclic pumping motion of the heart), low Reynolds number (relevant to human physiology) flow in combination with strong shear layer can cause transition of flow to turbulence in post-stenotic area which leads to heart attack or if it happens in one of the major vessels supplying the brain can lead to a cerebral stroke [2]. Therefore, studying stenosis artery has an important role to prevent progression of diseases and cure.

## 1.2 Present thesis

A complex turbulent flow is superposition of spatially coherent, temporally evolving vortical motions which is called coherent structures[19]. The study of dominant and coherent structures in turbulent flows was always challenging due to unsteadiness, three dimensionality and stochastic characteristics of turbulent flows. These coherent structures are responsible for the bulk mass, momentum and energy transfer [11]. Analysing these structures by decomposing the flow into the modes is a common approach. Among several decomposition methods, *Proper Orthogonal Decomposition (POD)* and *Dynamic Mode Decomposition (DMD)* which have been applied to study the physics of the dynamics of the flows are widely used in different applications. Developing a package to implement the mentioned methods on the cases that contain turbulent flow, was the aim of the current study. As a test case for the code a pipe with rigid wall and constriction in upstream has been chosen. The Direct Numerical Simulation of the flows in a 75% stenotic pipe flow (by area reduction) with 5% eccentricity position of the constriction has been studied previously by [4]. These conditions cause transition a laminar jet-like to a (quasi-)turbulent state in post-stenosis area depending on the Reynolds number which in most cases are dominated by a clear shedding frequency. The proper prediction of this changeover is crucial for an improved understanding. The physical goal of the study is the numerical investigation of the turbulence breakdown downstream of the stenosis area.

In order to simulate and discretize the flow numerically, the *Nek5000* is used to apply *Direct Numerical Simulation (DNS)* and *Spectral Element Method (SEM)*. Nek5000 is a parallel, modular, open source code, written in Fortran 77, and is able to scale up to 250000 processors. In Nek5000, there is an interface to visIt for visualization and post-processing, which is used also during the present study [18].

To have a better understanding of the POD and DMD theory, a comparison between some more common algorithms, introduced in previous papers, has been done and is presented in §2. The explanation about simulation process and tools is given in §3. In §4, there is description about the code development. The results and conclusions will be discusses in §5 and §6 respectively.

# Chapter 2

# Theory and formulation

Study transition of jet flow to turbulence in stenosis pipe flow needs initial knowledge about instability and turbulence. Therefore in the following sections some definition related to the subject will be mentioned. Furthermore, there will be a discussion about theory of different methods of POD and DMD.

## 2.1 Instability and Turbulence

### 2.1.1 Instability

The *hydroynamic stability* theory deals with this problem that how the laminar flow responses to forces or disturbances with small or moderate amplitudes. In this context, a flow is called stable if it turns back to the original laminar state, otherwise it goes to unstable state and often there is a changeover to turbulence [15]. According to the definition of the stability, if there is a decreasing in energy of the perturbation as the time proceeds, the flow is going to be stable, otherwise depending on the initial perturbation, it can be stable or unstable. There are three types of evolution of disturbances; evolution of disturbances that are localized in space, evolution of an unstable wavelike disturbance in time, and the oscillatory disturbances localized in space. In a case that the source of disturbance is fixed (first type of evolution), then the temporal growth is not an appropriate characteristic to define stability with. In absence of bound for disturbance growth, as the stream-wise distance goes to infinity, the flow becomes unstable. For stenosis pipe flow as we waited till the turbulent flow reaches near fully developed situation (statistically steady state), therefore we can assume that there is no temporal growth in time and instead there could be a growth in space.

### 2.1.2 Turbulence

For all flows, there is a critical Reynolds number where below this critical point, the flow is laminar and above this point there is a transition to truculence. Receiving energy from an external energy source (e.g. the mean flow) causes the turbulent flow to be sustained. Interaction between large scale and small scale is known as the cascade process. In the cascade process, the energy transfers from the larger scales (eddies) to the smaller scales. This means that the viscosity takes progressively important role up to final stage that kinetic energy dissipates to heat (by viscous dissipation, $\varepsilon$). The largest structure in a turbulence flow

is limited by the geometry i.e. the large vortical structure in a pipe is equal to its diameter while smallest scale has no dependency to the geometry. The viscosity and viscous dissipation are the only important factors in smallest scales or Kolmogorov scales. The quasi-equilibrium is a situation when transforming energy into heat in small scales is in balance with the energy that transfers from large scale to small scale. There are some characteristics of the turbulent flows that other approaches, that can be used for other problems in fluid mechanics, become useless [17]. These characteristics can be listed as below:

- *Stochastic behaviour: there is still no applicable simplification in statistical mechanics.*
- *Strong dependency in space and time: It is not easy to be modelled.*
- *Non-linearity: linearisation destroys the problem.*
- *Presence of vorticity (rationality).*
- *Three dimensionality.*

All turbulent flows are described by statistical fluctuations of all variables (e.g. velocity, pressure, density, temperature, etc.) around mean flow. For instance the perturbed values for velocity and pressure can be defined as $U_i + u_i'$ and $P + p'$ where the $(U_i, P)$ denote the basic state [15]. Therefore by applying the perturbed variables $(U)$ on Navier-Stokes equation and subtracting it from the basic state, we get the disturbance non-linear equations for incompressible flows as,

$$\begin{cases} \dfrac{\partial u_i'}{\partial t} = -\dfrac{\partial p}{\partial x_i} - U_j \dfrac{\partial u_i'}{\partial x_j} - u_j' \dfrac{\partial U_i}{\partial x_j} + \dfrac{1}{Re}\nabla^2 u_i' - u_j' \dfrac{\partial u_i'}{\partial x_j}, \\ \\ \nabla u = 0 \end{cases} \tag{2.1.1}$$

which is the equation for evolving system with initial disturbance $u_i^0 = u_i(t = 0)$ and will be complete by appropriate boundary conditions.

### 2.1.3 Statistical Analysis of Turbulent Flows

As it is mentioned in the last section, the Navier-Stokes equation can be used to study turbulent flows. However, since $u_i(u, v, w)$ is a random variable (it is the summation of the mean flow and fluctuations), to study turbulent flow one needs to define the concepts which cover its properties. To have a better understanding of a turbulent flow, we need to analyse development of some statistical quantities. First, we introduce two types of averages; averaging over time and ensemble average. These averages are equivalent for steady mean flow. The averaged velocity is commonly shown by $u_i(\underline{x}, t)$ where $\underline{x} = (x, y, z)$. The general definition of the average over time is,

$$\overline{u}_i(\underline{x}) = \lim_{T\to\infty} \frac{1}{T} \int_0^T u_i(\underline{x}, T_0 + t)dt \tag{2.1.2}$$

and the ensemble average over number of observation or snapshots of the flow which are obtained from experiments or numerical simulations is,

$$\overline{u}_i(\underline{x}) = \lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} u_i(\underline{x}, t_j). \tag{2.1.3}$$

The velocity deviation from mean value is called fluctuation,

$$u_i' = u_i - \overline{u_i}. \tag{2.1.4}$$

Also, the rms-value of $u$ can be defined by,

$$u_{rms} = \overline{u'u'}^{1/2} \tag{2.1.5}$$

As $\overline{u'u'}$, $\overline{v'v'}$ and $\overline{w'w'}$ introduce the kinetic energy per unit mass of the velocity fluctuations in each direction, therefore the *Turbulent Kinetic Energy* is defined as,

$$K = \frac{1}{2}\overline{u_i'u_i'} \tag{2.1.6}$$

that is presented in Einstein summation convention notation ($\overline{u_i'u_i'}+\overline{v_i'v_i'}+\overline{w_i'w_i'}$). If $\overline{u}_i' = 0$, 2.1.6 becomes,

$$K = \frac{1}{2}(\overline{u_i^2} - \overline{u}_i^2) \tag{2.1.7}$$

### 2.1.4 Simulation of Turbulent Flows

*Direct Numerical Simulation (DNS)* and *Large Eddy Simulation* (LES, used at a lower level of approximation) are two methods that can be used to model turbulent flows. The most important currents in analytical studies of turbulence are *statistical* and *deterministic*. However, there are some disadvantages with these approaches such as closure problem and limited criterion in studying and analysing only transition and pre-turbulence. Solving the Navier-Stoke equation numerically, without turbulence modelling is an important advantage of DNS [6]. As well as having a higher order of accuracy, it gives possibility to do the computation on complex geometries with non-conformal meshes in local refinements. However, increasing the Reynolds number make this method challenging and expensive (according to the reciprocal proportion of the smallest turbulent eddies to $Re^{\frac{3}{4}}$) specially in the flows with inhomogeneous turbulence. Therefore it is limited to the cases with simple geometries and low Reynolds number.

### 2.1.5 Discretization method

Spectral Element Method (SEM), a higher-order accuracy method, is a common method to apply to the DNS of the fluid flow with moderate Reynolds number. This method provides both efficiency of global spectral method and geometry flexibility of the finite elements method, provided by non-conforming deformed hexahedral elements [4]. Initial step to solve the Navier-Stokes equation is to decouple the equation into two elliptic sub-problems [16]. The spectral element spatial discretization is based on representing the weak form of the Helmholtz

equations on domain $\Omega$. In the next step the domain is subdivided to $K$ macro-(spectral)elements and is expanded the solution, data and test function within each sub-domain, in terms of $Nth$-order tensor product polynomials. The algorithm proceeds as the Legendre spectral element method uses the mapped Lagrangian interpolant bases on $-1$ to 1 interval (depend on dimension). The Gauss-Lobatto quadrature assure the accurate approximation to apply the obtained ansatz for $u$ (velocity variable) into weak integral. The result of this discretization is a linear system of equation that the rest of the process concerns to solve this system.

## 2.2 Modal decomposition

As the simulation of the flow does not give understanding about the underlying dynamics in it, there is a need to perform an analysis on the result of the simulation. On the other hand coherent structures approach to turbulence gives different picture, based on this idea that some turbulent are not so complicated that they look like at first sight. In another word, such flows are assumed to be composed of the set of organized motions with simple spatial structures. Complexity of these kind of turbulent flows generated by the superposition in space and time of these organized, so-called coherent structures[21]. Applying modal decomposition to identify coherent structures and dynamics of the evolving flow field is a solution to overcome the difficulties of the study of turbulent flows. The set of the modes obtained by modal decomposition can be represented as the reduced-order basis that is a approximation of the original flow. There are different decomposition techniques derived previously, but in the current study two of them will be considered; Proper Orthogonal Decomposition (POD) and Dynamic mode decomposition (DMD).

### 2.2.1 Proper Orthogonal Decomposition (POD)

The *Proper Orthogonal Decomposition* (POD) is originally proposed by Lumley in 1967. POD is a method to extract a basis for a modal decomposition from a set of snapshots and rank them by their energy. The energetic structures can be captured in transitional, weak turbulent flows and in a case that large scale dynamic (hairpin vortices) exist in turbulent flows [8]. It is possible to show that the POD decomposition is optimal for modelling and reconstructing a signal $u(x,t)$ (see more in [7]). The following explanation is according to the procedure that presented by Manhart et al. [7] about the theory of the POD. Assume $\overline{u}(\overline{x},t)$ is a turbulent flow field and the velocity vector $\overline{u}=(u,v,w)$, is defined in physical space $\overline{x}=(x,y,z)$ and time $t$. The data for velocity can be obtained either by measurement as a result of the experiment or by numerical simulation in discrete points and stored in matrix $U$. The time-space dependent velocity field can be expanded into orthogonal basis functions $\overline{\phi}^n(\overline{x})=(\phi_1^n,\phi_2^n,\phi_3^n)(\overline{x})$ in space, and orthogonal time coefficients $a^n(t)$ (see more about orthogonality in [7]). Therefore, if $N_M$ is number of the modes that we want to extract, we have,

$$u_i=(\overline{x},t)\rightarrow U=\begin{bmatrix}\overline{u}(\overline{x}_1,t_1) & \dots & \overline{u}(\overline{x}_{N_p},t_1)\\ \vdots & \ddots & \vdots\\ \overline{u}(\overline{x}_1,t_{N_T}) & \dots & \overline{u}(\overline{x}_{N_p},t_{N_T})\end{bmatrix} \tag{2.2.1}$$

$$a^n(t) \rightarrow A = \begin{bmatrix} a^1(t_1) & \dots & a^{N_M(t_1)} \\ \vdots & \ddots & \vdots \\ a^1(t_{N_T}) & \dots & a^{N_M}(t_{N_T}) \end{bmatrix} \tag{2.2.2}$$

$$\phi_i^n(\overline{x}) \rightarrow \Phi = \begin{bmatrix} \overline{\phi}^1(\overline{x}_1) & \dots & \overline{\phi}^1(\overline{x}_{N_p}) \\ \vdots & \ddots & \vdots \\ \overline{\phi}^{N_M}(\overline{x_1}) & \dots & \overline{\phi}^{N_M}(\overline{x}_{N_p}) \end{bmatrix} \tag{2.2.3}$$

in which, $N_p = N_x.N_y.N_z$ is the number of grid points, $N_T$ is the number of samples or snapshots that are taken in time and $N_M$ is the number of modes. The idea behind POD is to find a basis function $\phi_i(\overline{x})$ and coefficient $a$ to decompose the velocity field $\overline{u}$ such that,

$$u_i(\overline{x}, t) \approx \sum_{n=1}^{N_M} a^n(t)\phi_i^n(\overline{x}) \tag{2.2.4}$$

Therefore, one can say that the goal of POD is to find the proper orthogonal basis $\Phi$ by solving the minimization problem,

$$min\ R\ for\ a\ and\ \phi, \quad where \quad R = \int_T \int \int \int_v (u_i(\overline{x}, t) - \sum_{n=1}^{N_M} a^n(t)\phi_i^n(\overline{x}))^2 d\overline{x}dt \tag{2.2.5}$$

Solving the mentioned minimization problem leads us to two equivalent eigenvalue problems for $A$ and $\Phi$. The first one is original eigenvalue problem of the spatial modes, proposed by Lumley (1967)[7],

$$\frac{1}{N_T} U^T U G \Phi^T = \Phi^T \Lambda, \qquad \iiint_v R_{ij}(\overline{x}, \overline{x}')d\overline{x}' = \lambda_n \phi_i^n(\overline{x}). \tag{2.2.6}$$

in which, $G$ is a diagonal matrix and contains the coefficients of the discrete quadrature use to approximate the integral. The second one eigenvalue problem for temporal modes that is used for the number of snapshots are sampled in time,

$$\frac{1}{N_T} U G U^T A = A\Lambda, \qquad \int_T C(t, t')a^n(t')dt' = \lambda^n a^n(t) \tag{2.2.7}$$

These eigenvalue problems were called "direct method" and "methods of snapshots" by Sirovich in 1987 [7]. The spatial correlation tensor $R_{ij}(\overline{x}\overline{x}')$ and its temporal counterpart $C(t, t')$ in these equations can be defined as,

$$R = \frac{1}{N_T} U^T U G, \quad R_{ij}(\overline{x}, \overline{x}') = \frac{1}{T} \int_T u_i(\overline{x}, t) u_j(\overline{x}', t)dt \tag{2.2.8}$$

$$C = \frac{1}{N_T} U G U^T, \quad C(t, t') = \frac{1}{T} \iiint_v u_i(\overline{x, t}) u_i(\overline{x}, t')d\overline{x} \tag{2.2.9}$$

Because of the weighting function G, the matrix $U^T U G$ is not symmetrical. But for numerical purpose, it is possible to make it symmetrical by using a similarity transformation with $G^{1/2}$ [7]. As it is mentioned before, each eigenvalue represent

the energy content correspond to its mode and the total energy can be obtained by summation of these eigenvalues.

Instead of solving eigenvalue problem, one can use SVD for POD. Therefore, applying SVD on matrix $U$ we have,

$$U = \Phi \Sigma V^T \tag{2.2.10}$$

where, $\Phi$ is matrix of spatial modes, $V$ is matrix of temporal modes and $\Sigma$ contains sorted singular values on its diagonal.

As we discussed above, the method of the snapshots is another method to implement POD on a data field, which solves eigenvalue problem or SVD on $U^T U$ (specially in case of rank deficiency). This could be a faster method if the size of $U^T U$ is much smaller than $U$. Therefore we have,

$$U^T U = (\Phi \Sigma V^T)^T (\Phi \Sigma V^T) = V \Sigma^2 V^T. \tag{2.2.11}$$

The non-zero singular values of matrix $U$ are the square roots of the non-zero eigenvalues of $UU^T$ and $U^T U$ and can be used to obtain the energy. Therefore, the energy is the square root of diagonal elements of $\Sigma$. As we have the right singular vectors matrix then the left singular vectors of matrix $U$, and pod modes can be calculated as,

$$\Phi = UV\Sigma^{-1} \tag{2.2.12}$$

Computing energy either by solving an eigenvalue problem or by applying SVD one can recognize the most energetic structures of the system. It is possible to reconstruct the flow by these energetic structures according to the equation 2.2.4.

**Reconstructing travelling structures**

As POD separates the flow in space and time, therefore one can reconstruct the travelling structures according to the coherence structures in the flow filed. To find these coherence structures, understanding characteristic of the flow is the initial step.

Assume the flow contains a pure travelling mode, therefore the equation of the wave can be defined as,

$$f(x,t) = e^{i(\alpha x - \omega t)} = \sin(\alpha x - \omega t). \tag{2.2.13}$$

By implementing relation of difference of trigonometric functions and rewrite the wave equation we have,

$$f(x,t) = \sin(\alpha x - \omega t) = \sin(\alpha x).\cos(\omega t) - \cos(\alpha x).\sin(\omega t) \tag{2.2.14}$$

where, $\alpha$ and $\omega$ are real. The right hand side of this equation shows that this travelling structure contains two modes. The $\sin(\alpha x)$ and $\cos(\alpha x)$ denote the

Topo-modes (spatial modes) and the $\cos(\omega t)$ and $\sin(\omega t)$ denote the Chrono-modes (temporal modes). Therefore, reconstructing a pure travelling requires two modes. For the case that the flow has growth in space, $\alpha$ has both real and imaginary parts. In that case the equation 2.2.13 can be separated into real and imaginary parts,

$$f(x,t) = e^{\alpha_i x}.e^{i(\alpha_r x - \omega_r t)} \tag{2.2.15}$$

The only difference between last equation and 2.2.13 is an extra coefficient. Therefore, it shows that to reconstruct coherent structures only two modes are sufficient.

In real flow, it is not easy to recognize structures that a wave is composed by, therefore one solution is to look at the phase portrait of each two similar modes. This is possible by analysing parametric plot as a function of time. The parametric plot represents the periodic behaviour of related modes.

### 2.2.2 Dynamic Mode Decomposition (DMD)

However, POD is a method that can be applied for experimental or numerical snapshots to obtain the energetic structures, but there are some drawbacks which may not give the correct understanding of the dynamics in the flow. For instance, ranking only by energy may not cover all circumstances that the flow structures can be ranked by [9]. Therefore those modes captured by POD contain large range of frequencies. There is another decomposition method according to the Koopman analysis which can be applied on dynamic systems and is called Dynamic Decomposition. The method finds the eigenvalues and eigenvectors of non-linear dynamics of the flow that are approximated according to the eigenvalues and eigenvectors of the linear model [13]. The frequencies and growth rate of each mode can be obtained by magnitude and phase of its corresponding eigenvalue. In the following, there is a brief explanation about the theory of Koopman operator and DMD and also a comparison between different approachs to implement them.

- **Koopman Operator**

  Rowley et al. [13] define *Koopman operator* as below,

  *Definition:* Consider a dynamical system evolving on a manifold M such that, for all $x_i \in M$

  $$x_{i+1} = f(x_i) \tag{2.2.16}$$

  where f is a map from M to itself and i is an integer index. The Koopman operator is a linear operator U that acts on scalar-valued functions on M in the following manner: for any scalar-valued function $g : M \to \mathbb{R}$, U maps g into a new function Ug given by

  $$Ug(x) = g(f(x)). \tag{2.2.17}$$

  Then, there is a unique expansion [10] that expands each snapshot in terms

of $\phi_j$ or *Koopman eigenfunctions* and vector coefficients $v_j$ which are called *Koopman modes*, so

$$g(x) = \sum_{j=1}^{\infty} \phi_j(x) v_j \tag{2.2.18}$$

Now, iterates of $x_0$ are given by

$$g(x_k) = \sum_{j=1}^{\infty} \lambda_j^k \phi_j(x_0) v_j, \tag{2.2.19}$$

In this equation $\{\lambda_j\}_{j=1}^{\infty}$ or *Koopman eigenvalues* dictates the growth rate and frequency of each modes.

- **General description for DMD**

  The general idea of modal decomposition for non-linear systems can be defined as below.

  Suppose that $X_1^N$ is a collection of snapshots:

$$X_1^N = \{x_0, x_1, \ldots, x_N\} \tag{2.2.20}$$

which is collected in a constant sampling time $\Delta t$. Let, $\lambda_j$ and $v_j$ be the empirical Ritz values and vectors of this sequence. Assume $\lambda_j$'s are distinct. Then

$$x_k = \sum_{j=1}^{N} \lambda_j^k v_j, \qquad k = 0, \ldots, N-1 \tag{2.2.21}$$

where $v_j$ denotes modes and $\lambda_j$ has the same definition that is mentioned for Koopman expansion. One can write $v_j = v_j^{(0)} d_j$, where norm of $v_j$ is 1 and $d_j$ is called amplitudes of the modes [14]. Let, $V^{(0)}$ denotes normalized modes matrix and $V^{(1)}$ includes $v_j$. Therefore the matrix form of 2.2.21 is,

$$X_0^{N-1} = \{V^{(1)}\}_1^N S \quad \text{or} \quad X_0^{N-1} = \{V^{(0)}\}_1^N D^{(1)} S \tag{2.2.22}$$

where $D^{(1)}$ is a diagonal matrix containing amplitudes and is called scaling matrix $V^{(1)} = V^{(0)} D^{(1)}$. Also the matrix $S$ is the so-called *Vandermonde* matrix,

$$S = \begin{bmatrix} 1 & \lambda_1 & \lambda_1^2 & \ldots & \lambda_1^{N-1} \\ 1 & \lambda_2 & \lambda_2^2 & \ldots & \lambda_1^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_N & \lambda_N^2 & \ldots & \lambda_N^{N-1} \end{bmatrix}. \tag{2.2.23}$$

According to this expansion one can say that DMD algorithm approximates Koopman modes and eigenvalues from a finite set of data [10]. The idea

behind this decomposition is close to the Arnoldi method. Therefore, we are going to explain how the space that is constructed by linear mapping gives information about Ritz values and dynamic modes.

Assume $\{x_0, x_1, \ldots, x_n\}$ is a data sequence generated by a nonlinear system and also $f(x) = Ax$ where $A$ is a linear mapping that maps each snapshot to the next one, such that

$$x_{i+1} = Ax_i \tag{2.2.24}$$

this gives another representation of snapshots data set as below

$$X_0^{N-1} = \{x_0, Ax_0, A^2x_0, \ldots, A^{N-1}x_0\}. \tag{2.2.25}$$

We perform an analysis on the flow by extracting dynamic characteristics (eigenvalues, eigenvectors, energy amplification, etc.). Suppose that we can write last snapshot $x_N$ as a linear combination of previous $N-1$ vectors, for a sufficiently long sequence of the snapshots. Thus

$$x_N = c_0x_0 + c_1x_1 + \cdots + c_{N-1}x_{N-1} + r \tag{2.2.26}$$

or in the matrix form

$$x_N = X_0^{N-1}\overline{c} + r \tag{2.2.27}$$

in which $\overline{c}^T = \{c_0, c_1, \ldots, c_{N-1}\}$ and $r$ is the residual vector. Also we have the following relations,

$$A\{x_0, x_1, \ldots, x_{N-1}\} = \{x_1, x_2, \ldots, x_N\} = \{x_1, x_2, \ldots, X_0^{N-1}\overline{c}\} + re_{N-1}^T \tag{2.2.28}$$

or in the matrix form,

$$AX_0^{N-1} = X_1^N = X_0^{N-1}C + re_{N-1}^T \tag{2.2.29}$$

in which $C$ is known as the *Companion matrix*,

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & \ldots & 0 & c_0 \\ 1 & 0 & \ldots & 0 & c_1 \\ \vdots & \ddots & & \vdots & \vdots \\ 0 & \ldots & 1 & 0 & c_{N-2} \\ 0 & \ldots & 0 & 1 & c_{N-1} \end{bmatrix}. \tag{2.2.30}$$

The straight consequence of 2.2.29 is that decreasing the residual increases the overall convergence and therefore the eigenvalues of $C$ will converge

toward some eigenvalues of $A$. In a linear process this happens by increasing the number of flow fields. Therefore one way to monitor the convergence is evaluating the residual during the process and plotting its $L_2-$norm.

- **Computing the Companion matrix**

  Since, some of the eigenvalues of $C$ approximate some of the eigenvalues of $A$, and also corresponding eigenvectors are in proportion to the modes of $C$, we are looking for methods to compute the coefficients $c_1, c_2, \ldots, c_{N-1}$ (as the only unknowns of Companion matrix) such that $r \perp span(x_0, x_1, \ldots, x_{N-1})$. The eigenvalues of $C$ determine the temporal behaviour of the dynamic modes. Next, we will discuss about some methods to compute Companion matrix.

  1. *Pseudoinverse*

     As Rowley mentioned in [10], the easiest way is to obtain vector $\overline{c}$ from 2.2.27 such that,

     $$\overline{c} = (X_0^{N-1})^+ x_N \tag{2.2.31}$$

     in which, $(.)^+$ is *Moore-Penrose Pseudoinverse* and $x_N$ is the last snapshot. Therefore, to create the matrix $C$, one should set $\overline{c}$ as the last column and fill the rest with 1 and 0 as it represented in 2.2.30.

  2. *QR decomposition*

     A solution for the linear least-square problem obtained from 2.2.29, according to the economy size QR-decomposition of $X_0^{N-1} = QR$,

     $$C = R^{-1} Q^H X_1^N. \tag{2.2.32}$$

     This is an approximation for the Companion matrix.

     *Note*: Another approach [9] in this method is to substitute the QR-decomposition into 2.2.27,

     $$\overline{c} = R^{-1} Q^H x_N \tag{2.2.33}$$

     which gives the last column and only unknowns of the companion matrix, and the rest of the algorithm will be the same as the one for Pseudo-inverse. For more on this, see [11] and [12].

  3. *SVD*

     Instead of using QR-decomposition, one can also apply *SVD* on $X_0^{N-1}$,

     $$X_0^{N-1} = U \Sigma W^T \tag{2.2.34}$$

     Thus by applying the expression above on 2.2.29 we have

     $$X_1^N = U \Sigma W^T C \Rightarrow C = W \Sigma^{-1} U^T X_1^N \tag{2.2.35}$$

As it is mentioned in the POD section, it is possible to apply SVD on the matrix $(X_0^{N-1})^T(X_0^{N-1})$ and find the singular values and vectors, This is helpful when the matrix $X_0^{N-1}$ is rank deficient [9].

- **Calculating dynamic modes, scaling, reconstruction and energy analyzing**

  - *Diagonalizing by Vandermonde matrix*

    To initiate this method one needs to have Vandermonde matrix. To find Vandermonde matrix, first we need to find the eigenvalues of $C$ and then construct the $S$ matrix as it is mentioned in general description section. Now we take $S^{-1}$ to diagonalize Companion matrix such that:

$$C = S^{-1}\Lambda S \tag{2.2.36}$$

    Substituting this into 2.2.29 gives,

$$\begin{aligned} &AX_0^{N-1} = X_0^{N-1}S^{-1}\Lambda S \Rightarrow \\ &AX_0^{N-1}S^{-1} = X_0^{N-1}S^{-1}\Lambda \Rightarrow V^{(1)} = X_0^{N-1}S^{-1} \end{aligned} \tag{2.2.37}$$

    in which $V^{(1)}$ denotes eigenvectors of $A$ which is known as Dynamic modes. To obtain the energy, one can use the energy norm $\|V^{(1)}\|_2$ to normalize the columns of the dynamic modes matrix. So, normalizing the columns of $V^{(1)}$ has done by their 2-norm or global norm value. To see more in this, see [13] and [10].

  - *Diagonalizing via eigenvectors of Companion matrix*

    Another approach to find the dynamic modes is diagonalizing of Companion matrix by its eigenvectors [11]. By substituting $C = Y\Lambda Y^{-1}$ into 2.2.29, we have,

$$AX_0^{N-1} = X_0^{N-1}Y\Lambda Y^{-1} \Rightarrow AX_0^{N-1}Y = X_0^{N-1}Y\Lambda \Rightarrow V^{(2)} = X_0^{N-1}Y \tag{2.2.38}$$

    To reconstruct the snapshots by computed dynamic modes, there should be an appropriate scaling such that $V^{(1)} = V^{(2)}D_1$. To find the scaling matrix one can substitute the values for $V^{(1)}$ and $V^{(2)}$ from 2.2.37 and 2.2.38 which gives

$$X_0^{N-1}S^{-1} = X_0^{N-1}YD_1 \Rightarrow D_1^{-1} = SY \tag{2.2.39}$$

    where $D_1$ is a diagonal complex matrix. By following this method we obtain $V^{(1)}$ and then we are able to find the energy in the same way that is explained in the last session. But there is another way which also gives another scaling, reconstruction and energy. In this method the scaling matrix contains the coefficients which normalize the columns of $V^{(2)}$ and therefore if we denote the new scaling matrix as $D_2$ then $|D_2^{-1}|^2$ gives the energy.

- **POD-DMD**

  The mentioned methods so far, are appropriate when the matrix $X_0^{N-1}$ is full-rank. However, it is an ill-conditioned method in case of rank-deficiency or when the experimental data are contaminated by noise or other uncertainties. As a more robust approach, one can apply POD-DMD [9] to find dynamic modes. As a first step we apply $SVD$ on $X_0^{N-1}$ matrix, so

$$X_0^{N-1} = U\Sigma W^T \tag{2.2.40}$$

  Then by substituting last expression into 2.2.29 we have

$$X_1^N = AU\Sigma W^T \Rightarrow X_1^N W\Sigma^{-1} = AU \Rightarrow AU = UU^T X_1^N W\Sigma \tag{2.2.41}$$

  Next, we define $\widetilde{C}$ such that

$$\widetilde{C} = U^T X_1^N W\Sigma^{-1}. \tag{2.2.42}$$

  By diagonalizing $\widetilde{C}$ and substituting it into the last expression of 2.2.38 we have

$$AU = UY\Lambda Y^{-1} \Rightarrow AUY = UY\Lambda. \tag{2.2.43}$$

  Therefore, from the last equality spatial modes can be defined as,

$$V^{(3)} = UY \tag{2.2.44}$$

  in which $U$ is the right singular vector matrix of $A$ and $V^{(3)}$ contains the dynamic modes.

  The method so far gives the eigenvalues and the dynamic modes. Therefore, to reconstruct the snapshots from computed modes, we need to find a scaling matrix which can be defined as modes amplitude. Note that there is only a unique scaling matrix which relates $V^{(1)}$ to spatial modes $V^{(3)}$. Let $D_3$ denotes the scaling matrix. We have $V^{(1)} = V^{(3)}D_3$. To find the mentioned matrix we start from 2.2.22,

$$X_0^{N-1} = VS \tag{2.2.45}$$

  By applying SVD on $X_0^{N-1}$ and substituting the scaled $V^{(3)}$ we have,

$$U\Sigma W^T = V^{(3)}D_3S = UYD_3S \Rightarrow D_3^{-1} = SW\Sigma^{-1}Y \tag{2.2.46}$$

  which $D_3^{-1}$ is a diagonal complex matrix and if we normalize the eigenvectors of the transformed companion matrix $Y$, then the diagonal components of $|D_3^{-1}|^2$ is the amplitude $\phi$ or Energy [14]. Consequently, the dynamic

modes gives information about the energy or amplitude and the shape of each mode. This method is implemented for present thesis, but still there is some points to discuss about the scaling and energy ranking. Computing the Vandermonde matrix is a requirement to obtain the scaling matrix according to the mentioned method. This causes problems in case of applying the method for bigger number of snapshots, as one should power the eigenvalues by $N-1$. Therefore, there is another approach that may help to find more accurate and stable scaling method, however it did not test in current study. By applying SVD on $X_0^{N-1}$ we have,

$$X_0^{N-1} = U\Sigma W^T \Rightarrow X_0^{N-1} = UYY^{-1}\Sigma W^T \tag{2.2.47}$$

which one can recognize the $V^{(3)}$ from the last expression. Therefore if we define $W^{'} = Y^{-1}\Sigma W^T$ and do normalize its rows to unity then we have another diagonal scaling matrix $D_4$ which satisfies this expression,

$$X_1^{N-1} = V^{(3)} D_4 W^{'T} \tag{2.2.48}$$

# Chapter 3

# Simulation

In following chapter there are more explanation about the start-up case and previous studies and also about the Nek5000 and the way that it is applied for present study.

## 3.1 Simulation tool

Nek5000 is a computer package to simulate fluid flows and heat transfers for steady and unsteady, stationary or moving 2D or 3D cases [18]. The package includes three main parts, `prex` as a pre-processor, `nek5000` as solver, and `postx` as a post processor. The `prex` is an interactive menu-driven program which can be used to inter the information of the mesh, material properties, boundary conditions or other specification of the case. The `nek5000` contains the solvers for integrating Navier-Stocks equation or heat equations and the `postx` is an interactive graphical package for post-processing. The package is written in f77 and C and also is parallelized using MPI. There are some LAPACK routines which are applied in some modules. For the present study the `prenek` is used to generate the mesh for the straight pipe and the `nek5000` is used as a solver for the flow case.

### 3.1.1 Structure of each simulation

There are three files in Nek5000 that needs to be modified according to the specification of the case at the first step of simulation: `.rea, .usr`, and `SIZE`. The `.rea` brings the ability to control the runtime parameters which consists of viscosity/Reynolds, conductivity, number of steps, time-step size, order of time-stepping, frequency of output, filter strength, etc. . One of these parameters is related to define the output specification file. The output file depends on the value that is chosen for `parameter 66` in `.rea` file, can be varied. In the case of negative values for that parameter, the result will be in ASCII format with `.fld` output file, otherwise it will be a binary file with the name `.f%05d`. For the binary format all sections of `.rea` file that need to be stored (scaled with number of elements, i.e. boundary conditions, geometry) are moved to `.re2` file. For large cases (number of elements $> 100,000$) there is an option with the name `reatore2` (the `re2torea` to do inverse of this conversion) which gives `re2` file that keeps all data about the mesh and boundary conditions in binary format. Also, options to resolve or restart the simulation and to choose the required data

(coordinate, pressure, temperature, etc) to write in the outpost files, are part of the `.rea`.

In `SIZE` file there are variables to govern memory allocations. There are parameters to define dimensions and polynomial order for simulation. The `ldim` is dimension parameter and `lx1` controls the polynomial degree ($N = lx1 - 1$). The parameter `lxd` is used to control the polynomial order of the integration for convective terms and generally is defined as `lxd`$= 3*$`lx1`$/2$. Also, the upper limit for total number of processors and elements in the simulation and the maximum number of elements dedicated per each processor, `lelt`, are defined in this file.

The `.usr` file which concludes the subroutines allows to access all runtime variables. The subroutine `userchk` helps user to check the solution after each iteration for diagnostic purposes. The `userf` which is the forcing for the fluid can be defined in this part. To specify initial and boundary conditions, the `useric` and `userbc` subroutines can be used.

As initial step for simulation, one should generate the mesh for the case and run `genmap` for domain partitioning .The generated file by `.rea` after running `genmap` is called `.map` file. Compiling the code can be done by `makenek` which builds the `nek5000`. At the end one can run the code with running `nek5000` for demanding number of processors. The, snapshots can be visualized for analysing and post-processing by different visualization tools. Figure 3.1 is taken from [18], shows the steps of doing a simulation by Nek5000:

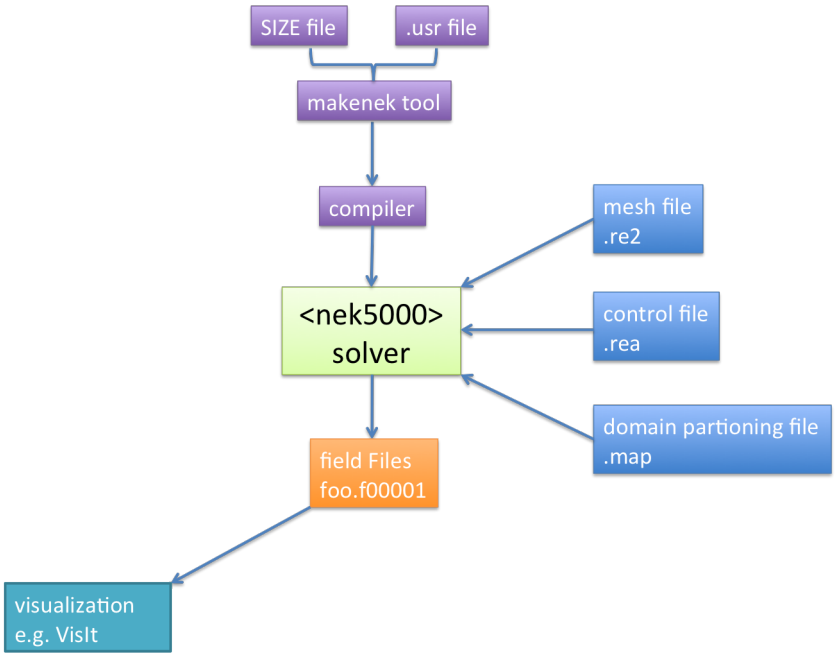



Figure 3.1: Working flow overview.

## 3.2 Previous studies

Stenotic flow has already been studied by many different groups, both experimentally and numerically. A number of these works focused on unsteady turbulent pipe flows, including periodic pulsating flows and non-periodic transient growth.

There are some discussion about the effect of amplitude and frequency of turbulent flows. However, there are some topics related to the steady flows. On 2007 Fischer et al. published two papers to study pulsatile and steady stenotic flows with the help of direct numerical simulations ([4] & [5]). In that case, the 3D solid pipe with maximum area reduction 75% for both, axi-symmetric and eccentric (5% perturbation at throat) of constriction is considered. The inlet Reynolds number has been taken 1000 and 500. The Direct Numerical Simulation predicted a laminar flow field downstream of the axi-symmetric stenosis model for both Reynolds number. Stenosis eccentricity causes the jet to deflect toward the side of eccentricity. For this case, flow stayed laminar at Reynolds 500 however a breakdown to turbulence has been observed for 1000 in post-stenosis area [4]. In the following sections, the process of simulations of the case with the same setting of previous study of eccentric stenosis pipe, is explained.

## 3.3 Formulation and simulation

As it is mentioned before, the following steps explain how the 3D stenosis pipe has generated according to [4]. The non-dimentionalized form of the incompressible Navier-Stokes eqution 3.3.1 in $R^d$ ($d = 3$ in our case) has been solved by Nek5000:

$$\begin{cases} \dfrac{Du}{Dt} = -\nabla p + \dfrac{1}{Re}\nabla^2 u \quad in\ \Omega, \\ \\ \nabla u = 0 \end{cases} \tag{3.3.1}$$

where $u = (u, v, w)$ is the velocity vector, $Du/Dt = \partial u/\partial t + u\nabla u$ is the material derivative of $u$ and $p$ is the pressure. The $Re = UD/\nu$ is the Reynolds number ,in which $U$ is the mean velocity, $D$ is the pipe diameter and $\nu$ is the kinematic viscosity. To impose inhomogeneous Dirichlet velocity on inlet, the parabolic velocity profile of the laminar fully developed Poiseuille flow is taken as,

$$\frac{w}{u_b} = 2(1 - 4r^2), \quad \frac{v}{u_b} = 0, \quad \frac{u}{u_b} \tag{3.3.2}$$

in which $u$, $v$ and $w$ are velocity components in $x$, $y$ and $z$ directions, respectively, $u_b$ is bulk velocity and $r = \sqrt{x^2 + y^2}$ is radial distance from pipe center line. However, stress-free boundary condition has been implemented for outlet, it means that $\partial u/\partial x - p = 0$ where p denotes pressure and is set to zero. The solid boundary (homogeneous Dirichlet) is taken for axial boundary as the pipe wall assumed to be rigid.

There are different rules for boundary conditions in Nek5000 but the general rule is that the capital letters indicate that the `.rea` file or solver provide the required details of the boundary conditions, on the other hand, lower case letters indicate that the solver should look in the `usr` file for required parameters. In latter case, the appropriate values, temperature, velocity, and flux boundary conditions must be specified in the `USERBC` subroutine in the `.usr` file [18]. To implement inlet boundary conditions that mentioned, the lower case letter `'v'` has been taken. Therefore, we implemented the required equations as initial and boundary conditions in `userbc` and `useric` subroutines of `.usr` file. Whereas for outflow boundary condition, capital letter `'O'` was chosen and applied the Dirichlet boundary condition at the outflow.

As it is mentioned before, the flow case, the same as [4], is an eccentric stenosis rigid pipe flow with 75% reduction of the area and 5% (of the pipe diameter) eccentricity. To generate the stenosis, a cosine function was applied on axial coordinate, $z$, and the function $S(z)$ computed cross-stream coordinates. Therefore, offset $E(z)$, for eccentricity $0.05D$, $S(z)$, $x$ and $y$ are defined as below,

$$\begin{cases} S(z) = \frac{1}{2}D[1 - s_0(1 + cos(2\pi(z - z_0)/L))], \\ E(z) = \frac{1}{10}s_0(1 + cos(2\pi(z - z_0)/L)), \\ x = S(z)cos\theta, \\ y = E(z) + S(z)sin\theta \end{cases} \tag{3.3.3}$$

in which, $D = 1$ is diameter of the non-stenosed pipe, $s_0 = 0.25$ is a coefficient to apply 75% area reduction, $L(= 2D)$ denotes the length of the stenosis and its center is located at $z_0 = 0$ ($z_0 - 1/2L \leq z \leq z_0 + 1/2L$). The command to implement the eccentric stenosis on geometry was added to the `usrdat2` subroutine in `.usr` file.

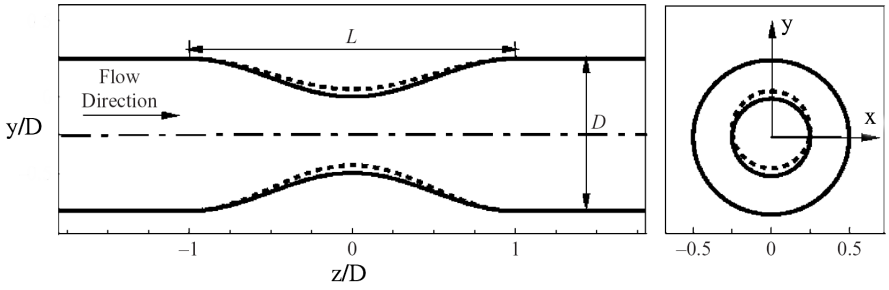
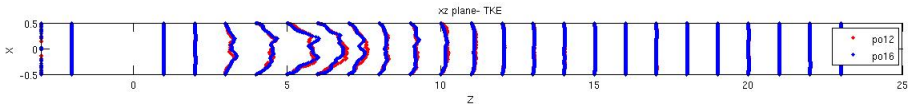



Figure 3.2: Side and front views of the stenosis geometry ($L = 2D$). The dashed line shows the 0.05 offset from the main axis of the pipe (borrowed from [4]).

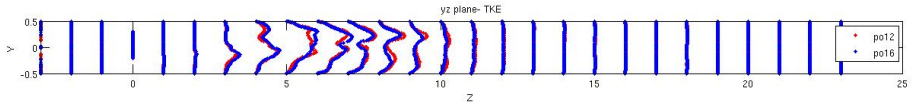
The spatial descritization, has done by Nek5000 that applies the $P_N - P_{(N-2)}$ spectral element method. Therefore, it represents the velocity ($N$) and the pressure ($N - 2$) at $N$th order tensor product Lagrange polynomials, in each of $K$ element of the mesh. For present $3D$ case, the number of elements $K = 11136$ was found sufficient. Furthermore, to find the suitable number of grid points, we did the convergence study on rms velocity (for axial and stream velocity) and turbulent kinetic energy (figure 3.3) for $N = 12$ and $N = 16$. The statistics for two number of grid points is matched together, therefore the simulation of snapshots has been done for $N = 12$.

The mesh in 2D ($xy - plane$) is generated by `Prenek` and then to generate 3D mesh it is extruded in $z$ direction, from $-3D$ to $24D$, by using `n2to3` which can be seen in figure 3.4.

We did the simulation for $Re = 750$ according to the pipe diameter, $D$, and $u_b$. The non-dimensional time step is taken $2.5 \times 10^{-4}$, according to the time step of the eccentric case, used in [4]. This keeps the CFL number less than 0.5. From now, all the variables we use for Reynolds number, velocity and time are




(a)




(b)

Figure 3.3: The turbulent kinetic energy, a)xz-plane b)yz-plane, comparison for two sets of snapshots for $N = 12$ and $N = 16$ shows a coincident, therefore $N = 12$ is suitable resolution.

Figure 3.4: Layout of spectral element mesh with $k = 11153$ hexahedral cells.

non-dimensionalized according to the pipe diameter $D$ and mean inlet velocity $u_b$.

To make sure that the initial transition had left the domain, the snapshots gathered when the turbulent flow has achieved statistically steady states. So, turbulent kinetic energy and root-mean-square velocities monitored and the snapshots gathered when the flow statistically converged. For the first set of the snapshots, the velocity profile monitored up to time 170 ($tu_b/D$). The figure 3.5 shows the center-line velocity contour plot for the snapshots gathered in time interval $100 - 170$. Those back and forth which are visible in time interval 110 to 145 are the result of not using full restart method which shows the sensitivity of the flow to the very small disturbances. The source of this error is related to the high-order semi-implicit temporal descritzation that Nek5000 implies for time stepping. This means that the restart process should not be only based on one previous time step. In full restart method, four snapshots are used as initial snapshots for the rest of the computation.

Furthermore, figure 3.5 shows that, since time 150 that the snapshots gathered in one restart, the plot shows that for this Reynolds number, when the time passes, there is an advection of the turbulence to the end of the pipe. Therefore, from $t = 170$, a sinusoidal Gaussian force has been implemented in the post stenosis area and in the span-wise direction to keep shedding at this region,

$$f_y = A \cdot \sin(\omega t) \cdot e^{-\frac{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2}{\sigma^2}} \tag{3.3.4}$$

where, $A = 0.1$ is amplitude, $(x_0, y_0, z_0) = (0, -0.25, 2)$ is coordinates of the center of the force, and $\sigma = 0.2$ is width of the Gaussian. The $\omega = 2\pi/0.5$ means that to see the first harmonic we need to have at least 4 snapshots during each time period of the force. The time period $T = 0.5$ is taken according to the frequency of the hairpin vortices, in $\lambda - 2$ plot [19], from the previous set of snapshots. The hairpin vortex is a stream-wise inclined $\Omega$-shaped vortex with a span-wise arch [4]. The negative $\lambda_2$ value as one of the two negative eigenvalues

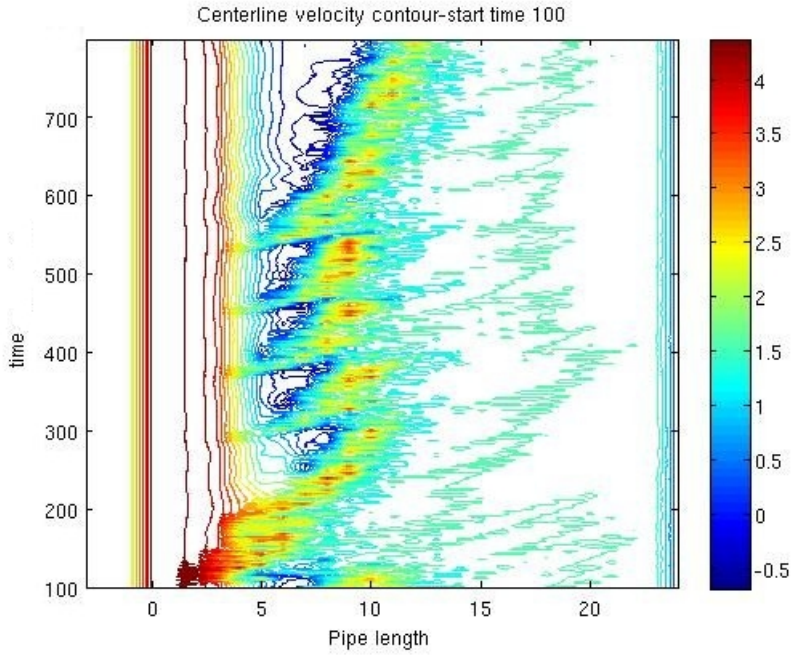



Figure 3.5: The center-line velocity contour for the case before applying the force.

of $S^2 + \Omega^2$ is introduced by Husaain and Jeong to identify the core of a vortex, where, the $S^2 + \Omega^2$ is the part of Navier-Stokes equations after applying gradient and neglecting the unsteady irrotational straining and viscous effects. Figure 3.6 shows a pseudo-color plot of the force generated by `Nek5000` and visualized by `visIt`.

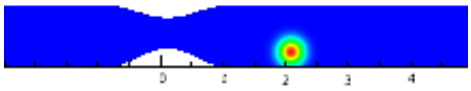


Figure 3.6: The position of the force in the pipe.

After applying the force, the center-line contour velocity shows the fixed position of turbulent region in the pipe 3.7.

After implementing the force the statistics of flow has been monitored from starting time 222 to make sure that the initial transient had left the domain. This study has was done for rms-velocity and turbulent kinetic energy values, according the formulas 2.1.6 and 2.1.5 for 800 snapshots from $t = 222$. To find the suitable starting time these snapshots were divided by four time intervals such that each interval contains 200 snapshots. Comparison of the statistics of these four time intervals showed acceptable agreement from $t = 247$ (see figure 3.8 and 3.9).

The visualization of generated snapshots by Nek5000 was done by `visIt 2.1.1`. The case has been run with 1000 processors on Lindgren, a cluster at

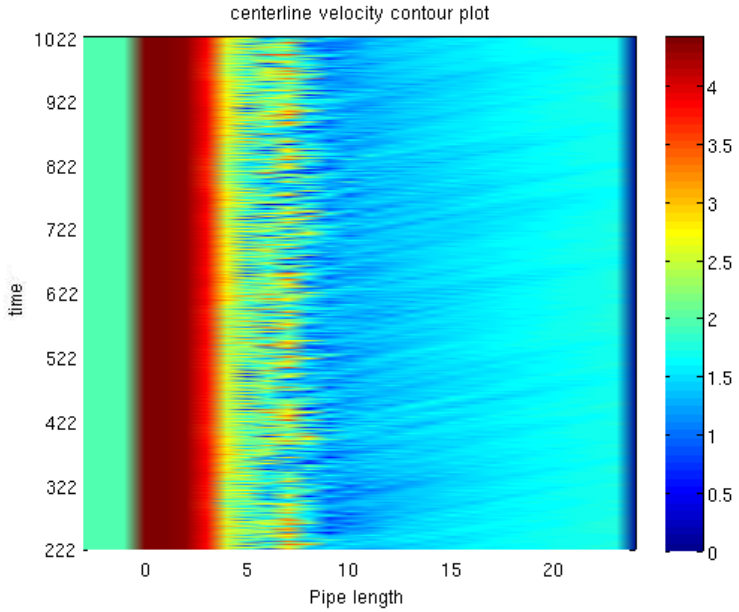
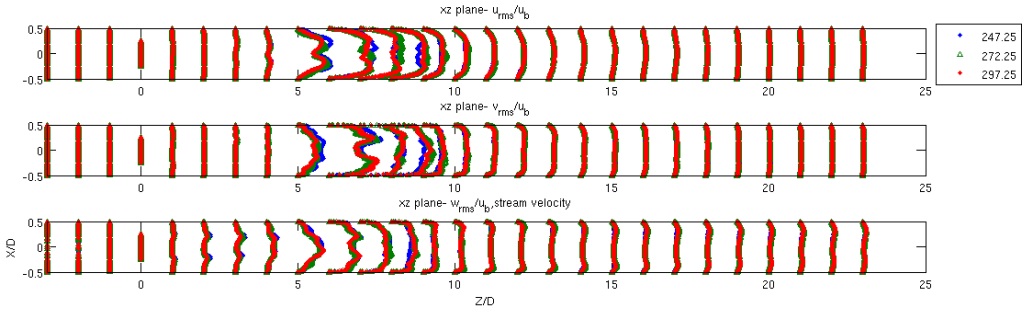



Figure 3.7: The center-line velocity contour for the case after applying the force starting from time unit 222.

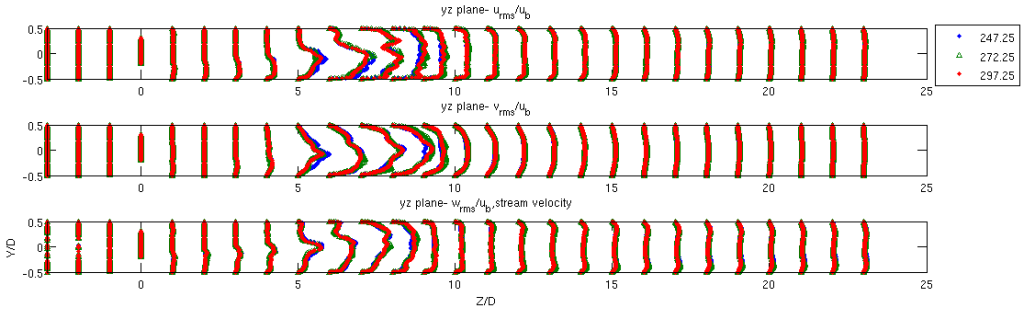
PDC center for hight performance computing at KTH. The required time to gather 600 snapshot, was about 3 wall-clock days.

### 3.3.1 Physical interpretation of the flow

The figure 3.10 presents the velocity magnitude contour of first snapshot captured at time 247 for $Re = 750$. The localized transition to turbulence can be seen approximately after $z = 4D$ which closely matches the previous study [4]. The break down to turbulence around $6D$ happens where the jet flow is deflected back toward the center. Deflection of the stream wise velocity toward the wall along the side of eccentricity, caused by geometry perturbation, can be seen in post-stenosis area that creates a distinct recirculation in opposite side (see the figure 3.11). The figure 3.12 that is the cross section cuts of the pipe shows the time-averaged velocity in which the mentioned process after stenosis can be observed. The negative values of the velocity shows the existence of the recirculation region. Furthermore, as the figure shows, the reattachment happens at $8D$ which is two pipe diameter earlier than [4] for the case with $Re = 1000$. The stream-wise vortices appear in the region $4D < z < 14D$. In figure 3.13, negative iso-contour values $\lambda - 2$, shows the transition and turbulent region. As it can be seen in this figure there are hairpin vortices appear after $4D$ which is two vessel diameter sooner than the previous study for $Re = 1000$ (distance from constriction is measured by vessel or pipe diameter). The study on these structures are the aim of modal decomposition which will be discussed in next sections. The figure 3.14, is the result of time-averaging is defined by 2.1.3. As it can be seen in this figure, biasing of the stenotic jet is apparent in yz-plane profile, however the xz-plane profile shows the symmetry.

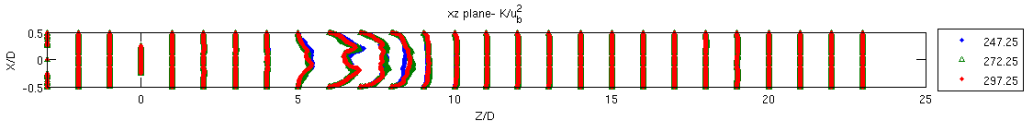



(a)

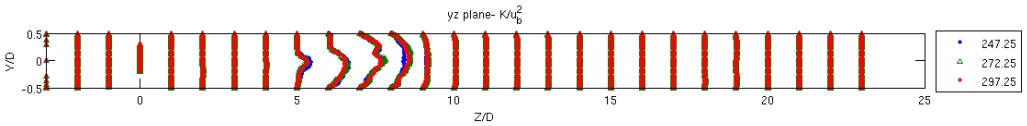



(b)

Figure 3.8: The rms-velocity shows acceptable agreement with the last three time intervals for xz-plane (y= 0) and yz-plane (x= 0). Each symbol represents the start point of these three intervals. It is assumed that the turbulent flow is reached to the statistically steady state. Furthermore, w_rms velocity profile in xz-plane shows existence of the hairpin vortices in the interval $0 < z/D < 4$.




(a)




(b)

Figure 3.9: The turbulent kinetic energy shows acceptable agreement with the last three time intervals for xz-plane (y= 0) and yz-plane (x= 0). Each symbol represents the start point of these three intervals. It is assumed that the turbulent flow is reached to the statistically steady state.

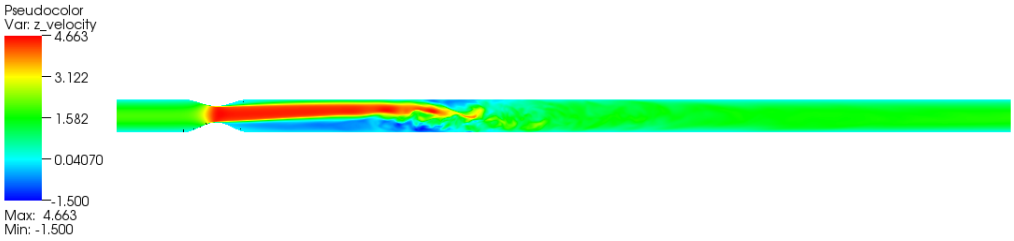



Figure 3.10: Stream-wise instantaneous velocity magnitude plot at 247, yz-plane cut.

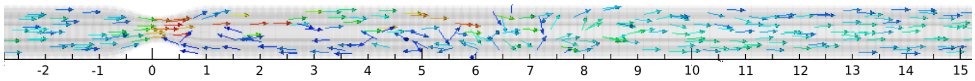



Figure 3.11: The negative velocity in post stenosis area.

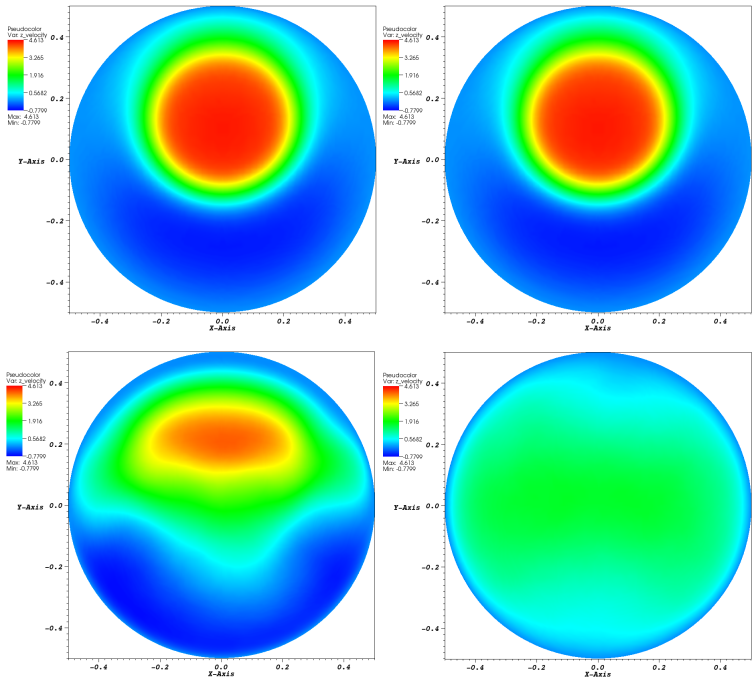
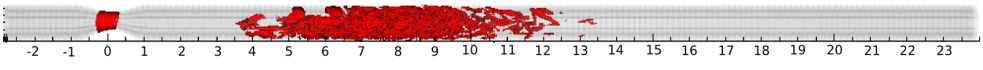
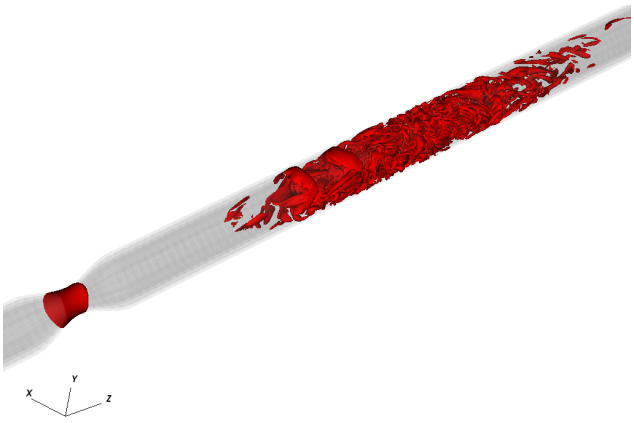



Figure 3.12: Stream-wise velocity cross sections: $2D$ (top left), $4D$ (top right), $6D$ (down left) and $8D$ (down right).




(a)




(b)

Figure 3.13: Negative contour -2.0 $\lambda - 2$ values represent instantaneous coherent structures in the turbulent region between $4D < z < 14D$

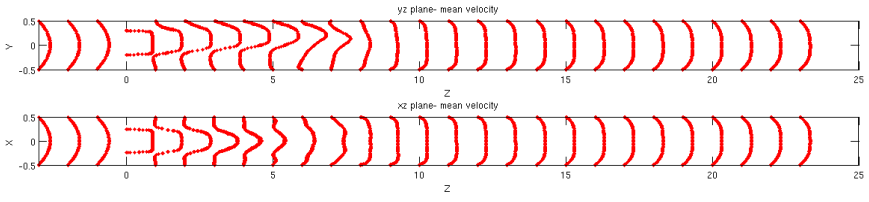



Figure 3.14: The mean velocity profile defined by ensemble average.

# Chapter 4

# Implementation

To implement modal decomposition on stenosis pipe flow, a modular code has developed for both POD and DMD according to the methods discussed before. Both snapshots and classical methods (applying SVD on the whole data set) has implemented for POD. For DMD the algorithm which is mentioned in [9] has been implemented.

## 4.1 The main structure

The code can be divided into three main parts, reading, modal decomposition and writing subroutines. The reading part contains required subroutines to read the snapshots and a module to get all specification related to the case. The second part that is computation of the modes, contains main program that calls the subroutines that the user chooses to implement for modal decomposition or reconstruction methods. The rest of the second part consist of all methods for decomposition or reconstruction and all subroutines that is required during the processes such as orthogonality checking of the POD modes. Finally, the third part contains a subroutine to write back the data to the file. There is a modular package to compute POD and DMD for generated data by `Nek5000` (with .fld format) and `SIMSON` that was developed by Milos Ilak in 2011. A point that make the current code different from the previous Mode package is, the ability of the code to read the file from the snapshots that are generated by `Nek5000` in '0.f' (binary) format and write the result to the same format. For this reason there are some subroutines and functions that are borrowed from `Nek.5000` and used in this package.

There is a script file (`nekbin.i`), to specify required input data, such that; name of the snapshots, the number of snapshots (to implement the method on), the number of the required modes, the number of the elements (to specify the domain of the interest), choosing the method, the option for generating coordinate in output, reducing mean velocity form the result in DMD and finally the input path and output path.

Each generated snapshot by `Nek5000` includes a header which contains information about the case and the data inside the file:

- single or double precision format of generated data
- polynomial degree (lx1,ly1,lz1)

- number of elements in the file, for parallel output the number of elements in one snapshot can be less than the total number of elements
- The time of the snapshot
- cycle
- number of the snapshots for each time interval (related to the parallel output) and the number of processors that generate the result in snapshot
- The variable indicator `XUPT`, shows the existence of different quantities in the output.

Reading these characteristics from the header is done by a subroutine inside the `i/o` module, which reads the header of the first snapshot.

There is a subroutine in the `readmat.f` is called `readmat` which gets the size of the matrix (that will be filled by required data for computation), index of the first matrix and the file path, as input. `Readsnap` is a subroutine which is called by `readmat` and gets the index of the snapshot to read as input argument. According to the variable indicator, the required memory will be allocated for coordination, velocity, pressure or temperature. The order of the loops to read data is set according to the subroutine `mfo_outfld` of the `Nek5000` in `prepost.f` as the code reads the snapshots in binary format ('0.f'). As Double precision data is required for computation, therefore there is possibility to read data in both double and single precision form and then store data in double precision in a matrix. Writing data in binary format is done by subroutine `writemat` in the writing part. The output matrix and the prefix of the output file name are the input arguments of this subroutine. There are some possible modification on header before it dumps in output file such as modification in number of elements or indicator. Also, to be able to visualize the result by `VisIt` the data is converted to the single precision format and written in an output file. Furthermore, generating coordination in output data is an option that user is able to specify that as an input condition.

The main part (`mode.f90`), contains the program `mode` to select between the exist methods. In following sections the subroutines for modal decomposition and reconstruction and some important issues e.g. solving the memory problem of large data sets will be discussed.

### 4.1.1 POD subroutines

#### Computing modes

As it is mentioned before the POD is implemented in two approaches, applying the SVD on complete data set which can be used for the case that there is enough space to load the input matrix into main memory. The subroutine that compute POD modes with this method is called `POD_orig`. This subroutine includes LAPACK routine `DGESVD` that gets the matrix of snapshots as input argument and returns $\Phi 1$, left singular vectors or POD modes as a matrix, a vector containing diagonal elements $\sigma$ as singular values or the energies, and $V^T$ where the temporal modes are stored on its rows. To optimize the memory usage, in all parts of the code `DGESVD` used with input character 'S' as `JOBU` and `JOBVT`, such that, for both matrices (the left singular vectors and right singular

vectors matrix) the first min(m,n) columns of them are returned in an array. This method is known as economy-size of SVD.

The required coefficients of the discrete quadrature are taken from `bm1` values of `Nek5000` which are used during the POD computation. The `bm1` is a mass matrix in which the 'm1' suffix refers to Mesh 1, that is where velocity and temperature are represented [18]. Reading `bm1` vector is done by reading subroutines and the snapshot number of `bm1` file name, assumed to be 1.

In `POD_snap` subroutine, the main concern is the memory space. As it mentioned in 2.2.11, the first step in this method is multiplying the snapshots matrix by its transpose. The result is a square matrix with the size of number of snapshots. Applying SVD on this matrix gives the temporal modes and the energies while the POD modes can be obtained according to the 2.2.12. However, reducing in size of the matrix solves the problem caused by applying SVD on large matrix, there are still some other problems related to the memory such as multiplying two large snapshots matrix by its transpose and finding the POD modes that has the same size as snapshots matrix. To multiply the snapshots matrix and its transpose and to avoid loading the whole matrix on memory at once, the snapshots will be read in chunks. The size of the chunks is computed by dividing number of snapshots by number of the chunks that could be specified in `nekbin.i` by user. Therefore, the size and the offset of the chunk is sent to `readmat.f`. As, in each step the new chunk is overwritten on last chunk, some chunks are read more than one time. The following pseudo-code and figure 4.1 explain about the process for this matrix multiplication.

```
for i=1:C
   load ch(i)
   multiply ch(i) by ch(i)
   fill chxTx(i,i)
   for j=i+1:C
       load ch(j)
       multiply ch(i) by ch(j)
       fill chxTx(i,j) and chxtx(j,i)
   end for
end for
```

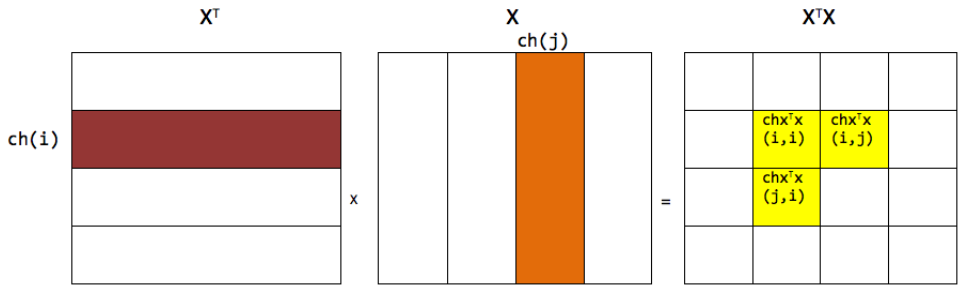



Figure 4.1: Matrix multiplication of $x^T x$ where $ch(i,j)$ denotes each chunk and $chx^T x(i,j)$ denotes one block in solution.

Since the output matrix is symmetric, by computing element $(i,j)$ the position

| # Snapshots | 200 | 400 | 600 |
|---|---|---|---|
| Matrix size | $5376 * 200$ | $5376 * 400$ | $5376 * 600$ |
| Memory size(GB) | 89 | 178 | 267 |
| CPU time(seconds) | 4100 | 10600 | 18000 |

Table 4.1: Memory requirement and sequential time of running POD_orig on Ellen for three sets of snapshots.

$(j, i)$ will be filled. Therefore, the number of operation is $N \cdot (\frac{C(C+1)}{2}) \cdot (2M - 1)$, where $N$ denotes the number of columns in each chunk, $M$ is the number of rows calculated in module `IO` (ntot) and $C$ is the number of chunks. To investigate the orthogonality of the computed POD modes (the columns of the left singular vector matrix), there is a subroutine, called `check_orthogonality`. In this subroutine the orthogonality of the columns of the mode matrix is checked within machine-accuracy.

For final 600 snapshots, the convergence of the result is checked for 200, 400 and 600 sets of snapshots. The result of the convergence will be discussed in chapter §5. We also report the required resources for these different of snapshots. Table 4.1 as a result of running `pod_orig` shows that increasing time has direct relation with number of the chunks. Also to compare the required resources for both `pod_orig` and `pod_snap`, the table 4.2 shows the result of different runs for these methods with 200 snapshots. The result clearly shows the difference between required memory and the runtime of these two methods. The maximum allocated memory for `pod_snap` method is less than 18 GB, however the runtime for 200 snapshots (which the snapshots are divided by 50 chunks) is more than runtime of `pod_orig` method for 600 snapshots. This is a direct consequence of the `pod_snap` algorithm. Therefore, having more chunks, simply means more reloading of data on memory (see figure 4.2).

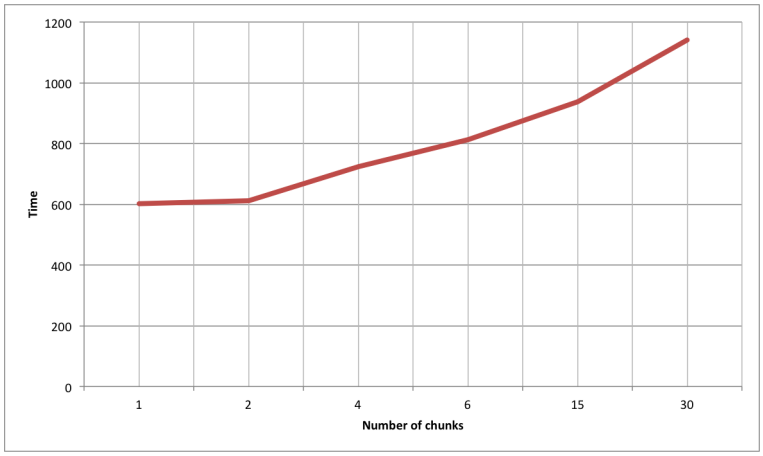



Figure 4.2: The number of chunks effects on cputime of running `pod_snap` for 60 snapshots.

The cputime for running 60 snapshots by `pod_snap` with one chunks is 602 seconds whereas running `pod_orig` for the same set of snapshots is 525 seconds. The required memory for 800 snapshots in running `pod_snap` is approximately 54(G).

| Method | pod_orig | pod_snap |
|---|---|---|
| Memory size (GB) | 89 | 18 |
| CPU time (seconds) | 4100 | 40000 |

Table 4.2: Comparison between running the code for two methods pod_orig and pod_snap (50 chunks) for 200 snapshots.

Another approach to manage the memory is that, in all subroutines, specially in `pod_snap`, allocating memory for matrices and array was done very carefully. For this reason after each step of the computation, all non-required memory spaces are freed. Also to control the problem caused by memory allocation after all allocation and releasing, the memory in use has been calculated and the result printed out to inform the user.

**reconstruction**

To reconstruct the travelling structures or to validate the modal decomposition result, there are two subroutines after each method, `pod_orig_reconstruct` and `pod_snap_reconstruct`. The difference between these subroutines is related to read and load of modes into memory. Apparently, to load the modes as a matrix, user should check required memory before calling `readmat`. To reconstruct requested modes together, there is a loop which will set the other singular values equal to zero. It means that the result shows the travelling structures of the modes correspond to the non-zero singular value.

### 4.1.2 DMD method

The `DMD` subroutine is implemented according to [9]. However, implementing scaling part was according to the method proposed by [14]. The mentioned subroutine gets the snapshots matrix as input and then returns the dynamic modes which are stored in different files as snapshots format and the amplitudes which are obtained from the inverted elements of diagonal scaling matrix. The LAPACK routines used in this subroutine are: `dgesvd`, `dgemm` and `zgemm`. As implementing DMD process was longer than the POD, many matrices should be loaded into memory for the computation. In case of large data set, keeping those matrices confronts the computation with memory problems even on Ellen. Ellen is a shared-memory multiprocessing (SMP) system, with total main memory 1 TB with 64 cores. The CPU's are of the model Xeon E7-4830 X with 8 cores 2.31 GHz. Therefore, instead of keeping the data in memory, we read the data more than once and this increases the computation time and decreases the performance. For instance, as `dgesvd` damage the data on original matrix, therefore for the next step of computation we should load the data into memory again. The CPU time for running `DMD` on 600 snapshots is about 25300 seconds and the maximum required memory is 267 (GB). There is also a reconstruction subroutine for this method which gets the dynamic modes as input and returns the coherent structure or complete flow field as output according to the 2.2.22.

### 4.1.3 Accelerating the code

As it is mentioned before, the main issue in writing the code the memory management, however increasing the performance is another issue in the second order of priority. Therefore, we used OpenMP pragmas for different loops, as much as possible and we run a multi-threaded code to accelerate the code. It is possible to set threads by using `MKL_NUM_THREADS` for LAPACK routines and `OMP_NUM_THREADS` for the rest of the code. However it is possible to set threads for both case only by using `OMP_NUM_THREADS` [20]. To find the optimum number of threads, the codes for all available methods ran for different number of threads. Due to the complexity of the algorithm in the `pod_snap`, we were not able to parallelize the algorithm significantly. This means that the speedup for the two mentioned subroutines are not noticeable. We have to emphasize that the multi-threaded SVD did not show a significant speedup. For `DMD`, the optimum number of threads is 8 which shows 2X speed up vs. the serial code (see figure 4.3). All computations for these codes have done on Ellen according to large size of data set.

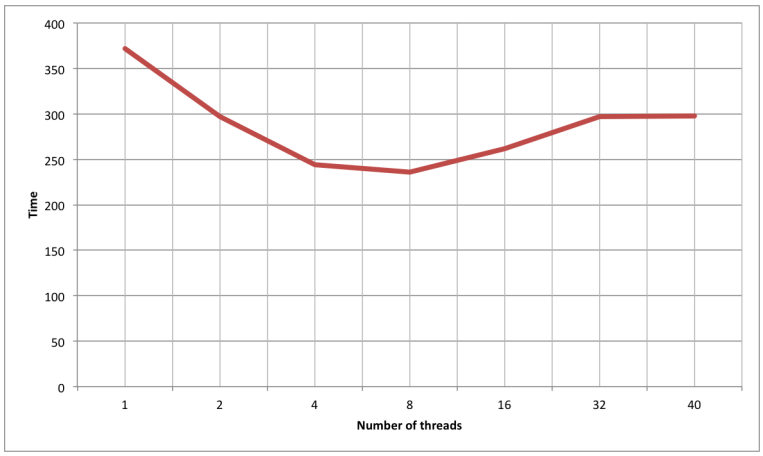



Figure 4.3: The optimum number of threads for `pod_snap` is 8.

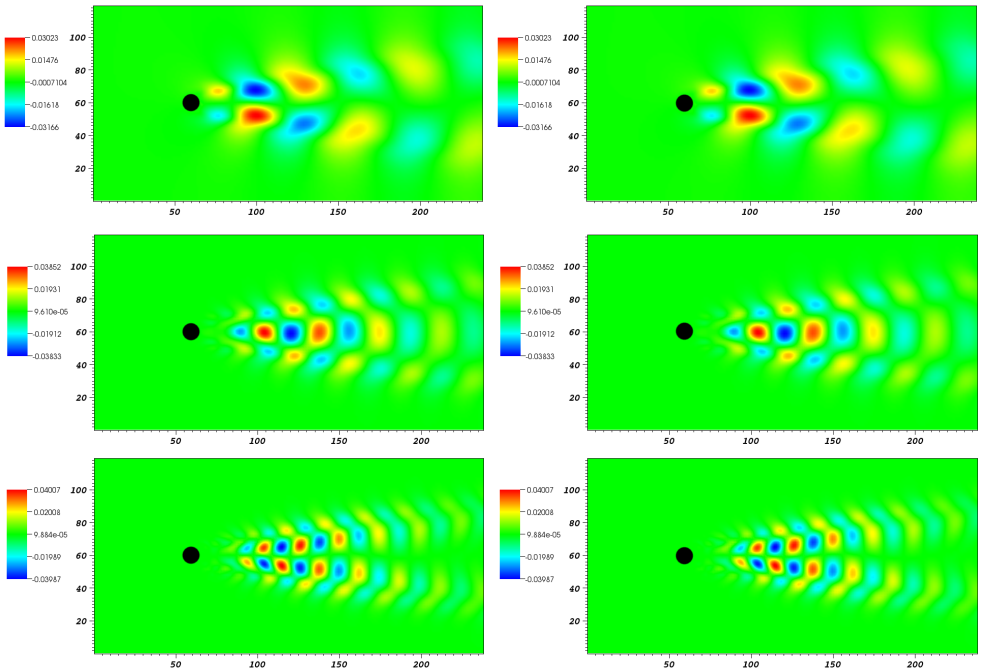
# Chapter 5

# Results

## 5.1 POD result

### 5.1.1 Validation

For validation of the code, the two-dimensional flow passing a cylinder has been considered. The computation has been done for $Re = 50$ and 50 snapshots to validate both `pod_snap` (for 10 chunks) and `pod_orig` codes. The figure 5.1 shows the spatial modes 1,3 and 5 computed by two methods and shows acceptable match in the results.



(a) (b)

Figure 5.1: Spatial modes 1, 3 and 5 (a) computed by `pod_orig` and (b) computed by `pod_snap` are match together.

This agreement can be seen in the temporal modes plot, presented in figure 5.3, and also reconstructing a travelling structure by spatial modes 1 and 2 (as coherent structure in the flow) showed in figure 5.5. The spectrum related to the

40 most energetic modes is represented in the figure 5.2. The figure 5.4 shows the orbit plot for mode 1 and 2. It shows periodic motion of these two modes discussed in §2.

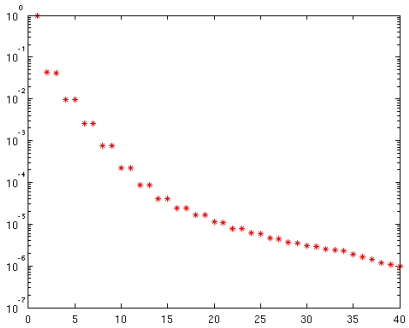
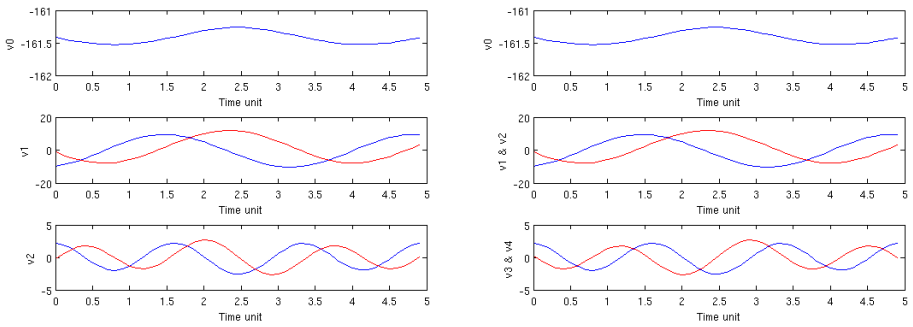
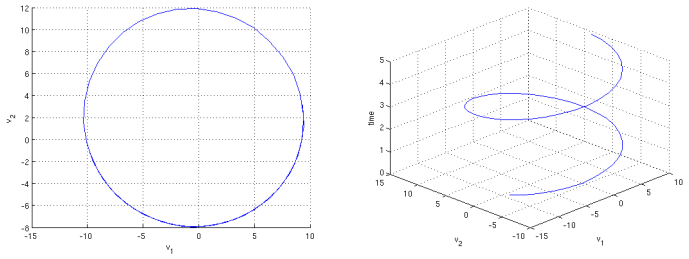


Figure 5.2: Logarithmic plot of energy spectrum correspond to most energetic POD modes.




(a) (b)

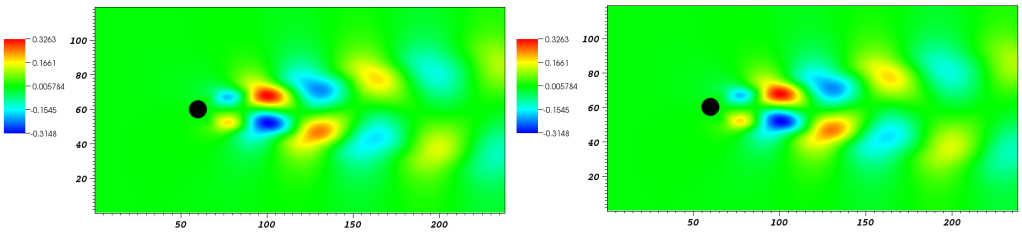
Figure 5.3: First 5 temporal modes (a) computed by `pod_orig` and (b) computed by `pod_snap` shows a complete agreement. Mode zero is almost constant and two pairs of next modes were plotted together, showing phase shift in time.




(a) (b)

Figure 5.4: Orbit plot of temporal modes 1 and 2 showing periodic motion of travelling structure correspond to these modes.

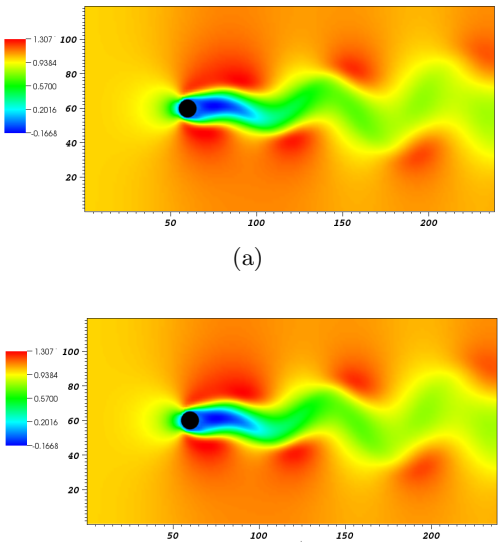
To validate the result, reconstruction has been done for all computed modes with both methods and the result compared with initial snapshots of the flow



(a) (b)

Figure 5.5: Reconstructed travelling structures by 1 and 2 spatial modes (a) computed by `pod_orig_reconstruct` and (b) computed by `pod_snap_reconstruct` shows perfectly match together.

(see figure 5.6).



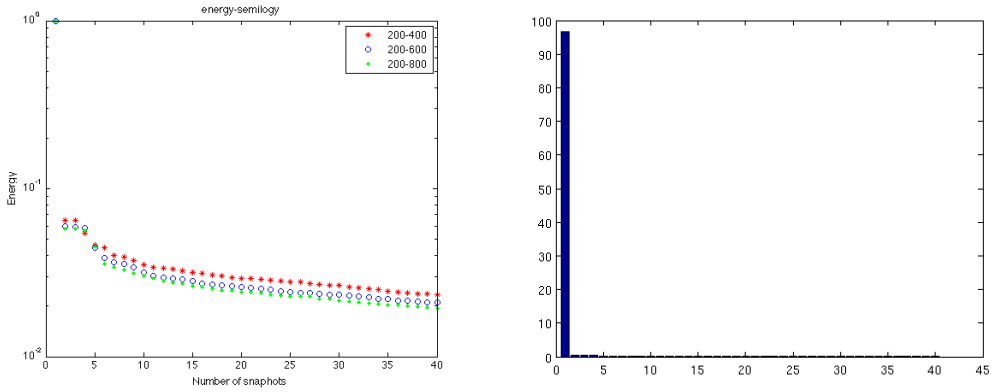
(a) (b)

Figure 5.6: (a) Initial snapshot from flow field (b) snapshot from reconstructed flow with `pod_snap_reconstruct` by all computed modes by `pod_snap` to validate the result.

### 5.1.2 POD result for stenosis pipe flow

To apply the POD, a subdomain is selected which contains the flow field with one inhomogeneous direction (because of the eccentricity of the stenosis). As mentioned in §3, the localized transition and breaking down to turbulence of the flow happens in post-stenosis area between $4D < z < 14D$. Therefore, to apply the POD, the interval $0 < z < 16D$ has been selected as domain of interest which contains 5376 elements. This domain is taken by removing elements from the first and last part of the pipe by using `reduced-geometry` which is an option in `nekbin.i`. In §3 and §4 the convergence of in hand snapshots regarding resolution and time step were analysed. Therefore POD has done on 600 gathered snapshots and the size of each snapshot is approximately 500 MB. The convergence of the POD modes is tested by analysing both energy cascade plot and the temporal

modes for 200, 400 and 600 set of snapshots, to take appropriate number of snapshots. The figures 5.7.a and 5.8 (which is a close up from the first and second modes) show acceptable agreement between the results. As it can be seen in figure 5.8 there is no phase-shifting in modes. This yields that POD modes does not dependent on the number of snapshots, however it is more apparent between 400 and 600. As the figure 5.7.b shows, the zero POD mode, correspond to the mean flow, contains 96% of the total kinetic energy which represents the mean flow.




(a) (b)

Figure 5.7: Energy spectrum (a) Semiology plot of energy spectrum of three sets of snapshots shows acceptable agreement. (b)The 96% of energy corresponds to the first mode.

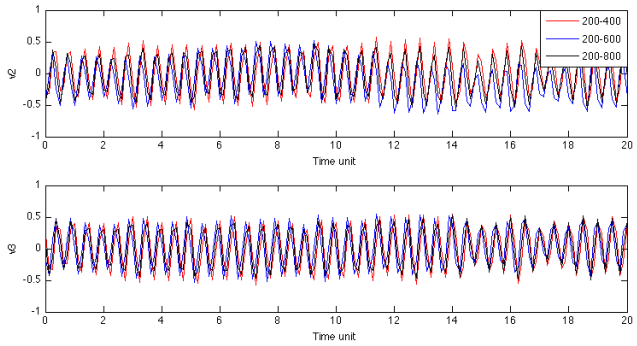
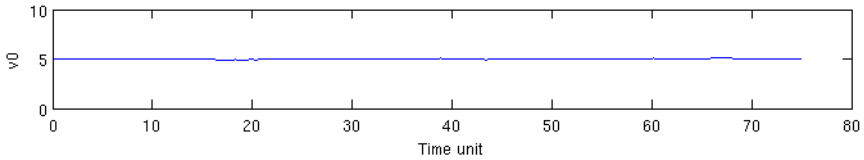



Figure 5.8: The second and third temporal modes for three sets of snapshots match together.

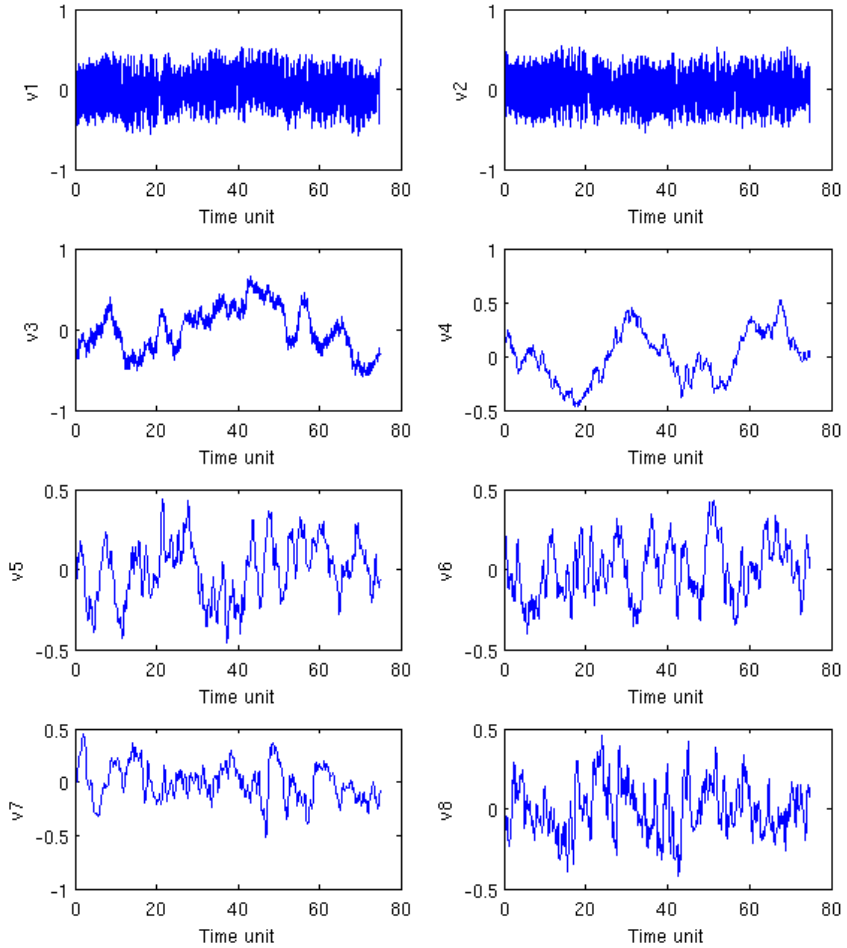
Furthermore, the orthogonality of POD modes has been checked and the result satisfied $\delta_{ij} < 10^{-14}$ for $i \neq j$ for all modes in both methods. From now on, all result are related to POD method with 600 snapshots.

The first 9 temporal plots in shown by figure 5.9. The constant value of the mode zero corresponds mean flow and the rest of the temporal modes present the fluctuation of the flow field. In the second and third modes, the effect of the implemented force is visible. The coherent structures, are represented in temporal modes such that, the second and third modes are related together. The figure 5.10 shows the temporal orbit correspond to 8 first modes. Each

plot includes a pair of the temporal modes. The figure 5.10.a corresponds to the temporal modes 1 and 2 and shows the periodic motion caused by force. However the figures 5.10.b-d which correspond to pairs (3,4), (5,6) and (7,8) temporal modes do not represent any periodicity.



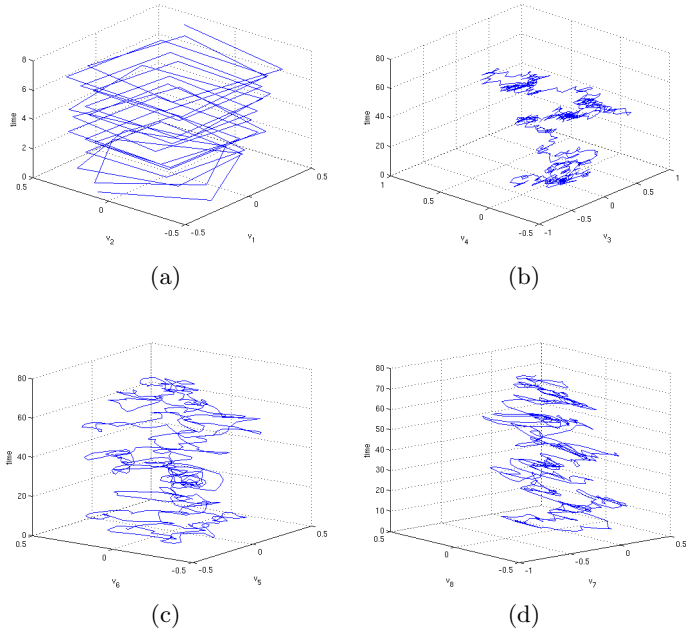

(a)




(b)

Figure 5.9: Temporal ,plot:(a) Mode zero corresponds mean flow, has constant value. (b) The first and second modes shows the effect of the force.

The figure 5.11.a represents the velocity iso-contours plot of reconstructed spatial modes correspond to (1,2) which shows travelling structure produced by these two modes. However, this result does not carry out from 5.11.b-c. Therefore, the superposition of the mentioned pair of modes only yields two large




(a) (b)

(c) (d)

Figure 5.10: Temporal orbit in subspace spanned by (a)$v_1$ and $v_2$ (b)$v_3$ and $v_4$ (c)$v_5$ and $v_6$ (d) $v_7$ and $v_8$ vs time.

structures that swap their positions, that is compatible with our understanding from phase portraits (5.10). The figure 5.11.d is the result of reconstructing modes with lower frequencies recognized in figure 5.12. The figure 5.12 shows distribution of energy among modes which the colors present the logarithm of the modal energy. The higher value in first modes corresponds to the frequency 2 is related to implemented force on the system. As it can be seen there is an accumulation of the energy in the lower frequencies related to the first 10 modes.

## 5.2 DMD result

### 5.2.1 Validation

To validate the `DMD` code, similar to POD, the code has been tested by 2D cylinder. The computed energy according to amplitude is shown by logarithmic scaled plot in figure 5.13.a (correspond to 5 first energetic modes). The figure 5.13.b shows position of Ritz values on unit circle. As it can be seen from figure, there are some values located inside the circle. It is possible due to the fact that the modes are not belong to a complete period. The computed dynamic modes correspond to the first mode, is presented in contour plot 5.14. The result is completely the same as the computed modes by an existing MATLAB code. Finally, the figure 5.15 shows a comparison between the reconstructed flow by all dynamic modes with the initial snapshot of the flow.

### 5.2.2 DMD result for stenosis pipe flow

The following results were obtained by computing dynamic modes for the stenosis flow. The figure 5.16 represents the Ritz values that are completely located on a unit circle. The computed energy of the modes obtained by diagonal elements of

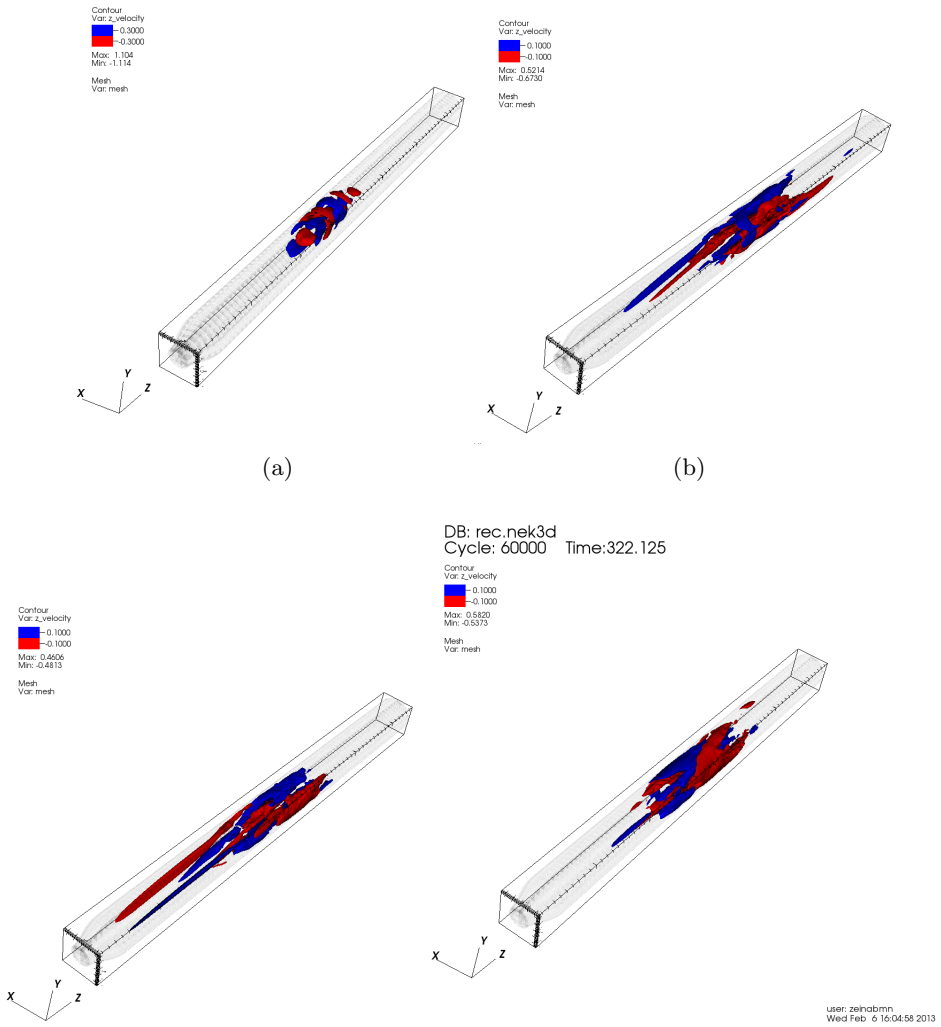



Figure 5.11: Velocity iso-contours of reconstructed travelling structure by (a) spatial modes 1 and 2 for $(-0.3, 0.3)$.(b) by spatial modes 5 and 6 for $(-0.1, 0.1)$. (c) spatial modes 7 and 8 for $(-0.1, 0.1)$ and (d) low frequnecy spatial modes 4-9 for $(-0.1, 0.1)$ found according to the figure 5.12

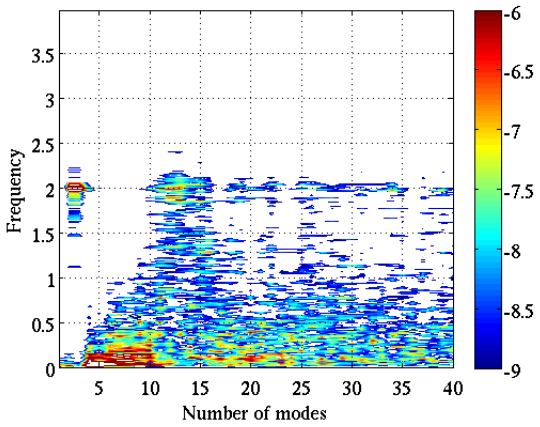
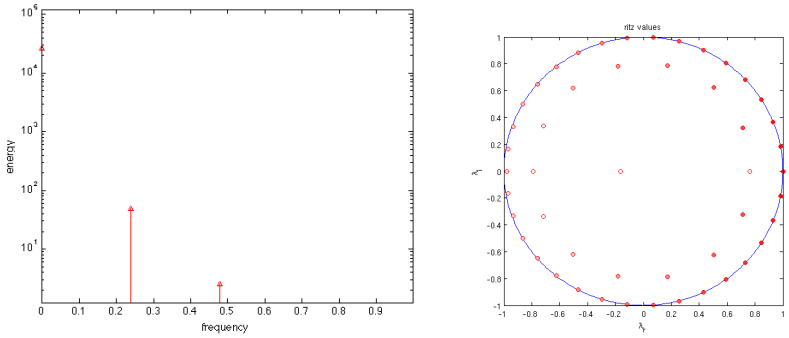
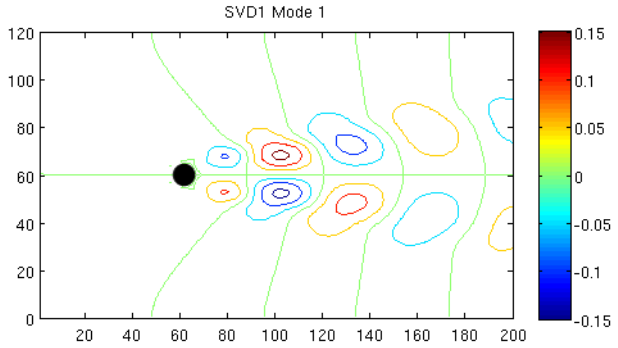



Figure 5.12: Distribution of energy among modes

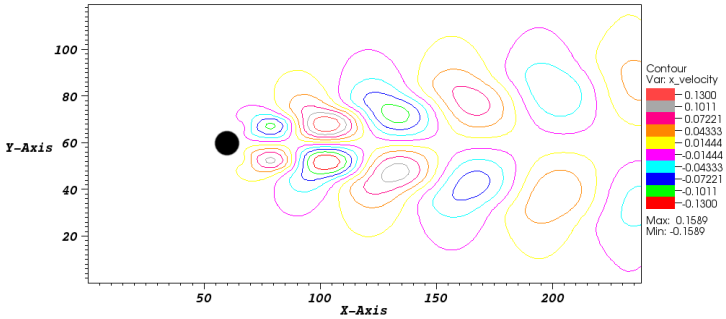



(a) (b)

Figure 5.13: Energy spectrum and Ritz values of 2D flow passing a cylinder (a) The energy computed by $|D^{-1}|^2$, shows only one energetic mode with low frequency. (b) Ritz values on unit circle



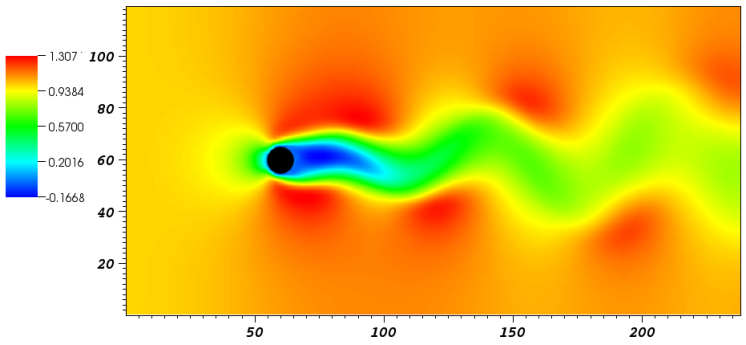

(a)



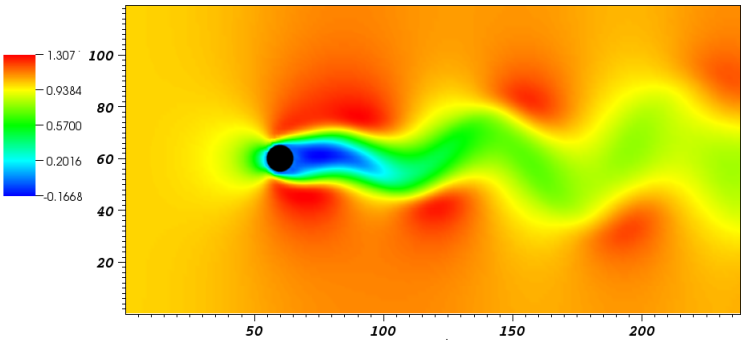

(b)

Figure 5.14: Comparison of first computed dynamic mode with (a) existing MATLAB code and (b)`mode.f90` for 2D flow passing a cylinder.

amplitude matrix $D^{-1}$ is shown in the figure 5.17 where 5.17.a includes the energy related to the force and mean flow and 5.17.b contains the energy of the existing structures in the flow. As it is apparent, the energetic modes corresponding to the




(a)




(b)

Figure 5.15: (a) Initial snapshot from flow fields (b) Reconstructed snapshot by all computed mode for 2D flow passing a cylinder

implemented force are not located on the expected frequency (as it is discussed in §5, the expected frequency of the implemented force is 2). It may caused by a small problem in the code that is remained unsolved. As we discussed in §2, the frequencies can be captured from imaginary part of the Ritz values. Also it can be seen that there are two energetic modes around the frequency 2.5. The contour plot of dynamic mode correspond to the implemented force is shown by figure 5.18. Finally the reconstructed flow field by all dynamic modes according to 2.2.22 is represented in figure 5.19 that shows the same result as initial flow field generated by Nek5000.

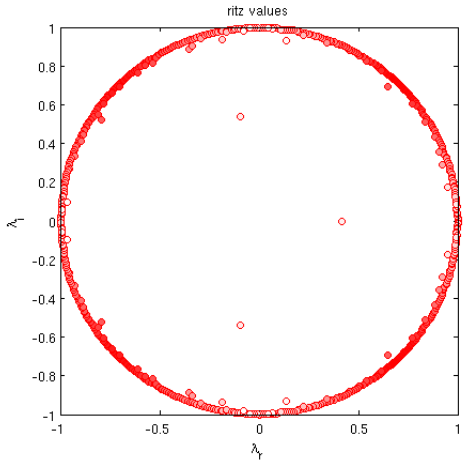
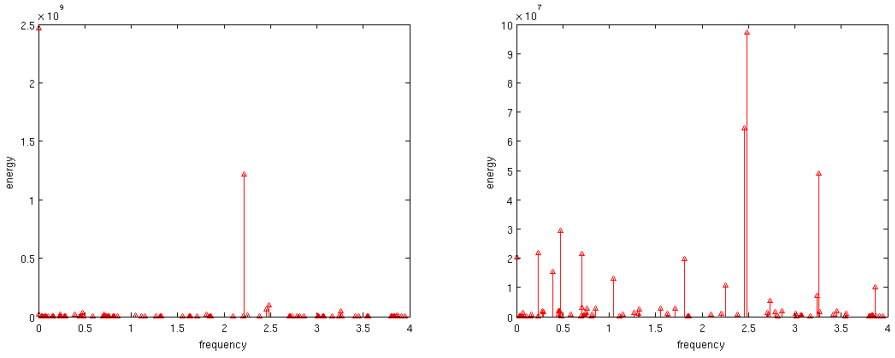



Figure 5.16: Ritz values located on unit circle, stenosis pipe flow in 3D.




(a) (b)

Figure 5.17: The energy of computed modes (a) consist of energy correspond to the mean flow and force. (b) The energy correspond to the mean flow and implemented force has been removed for 3D stenosis pipe flow.

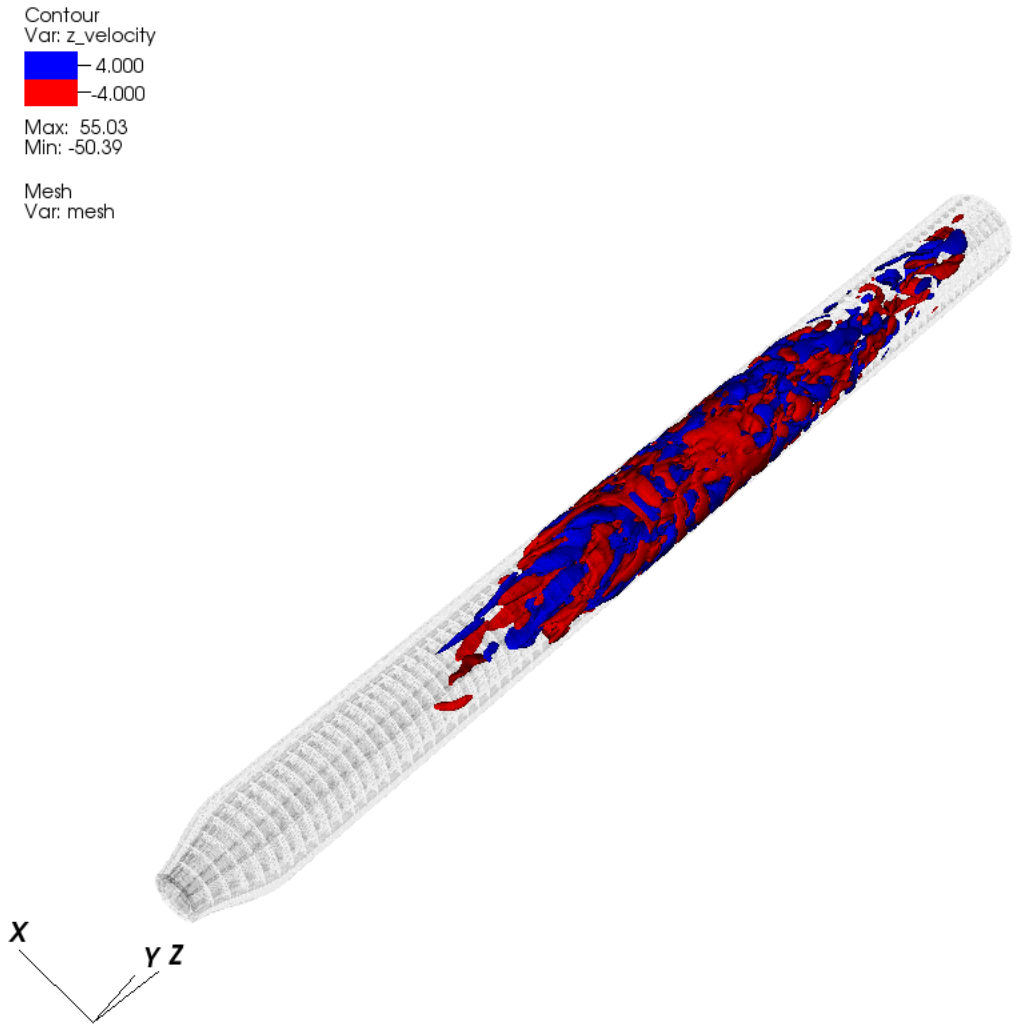



Figure 5.18: Velocity iso-surface $(-4, 4)$ plot of the first dynamic mode corresponds to the implemented force. The structures captured by POD for the same mode can not be recognized here.

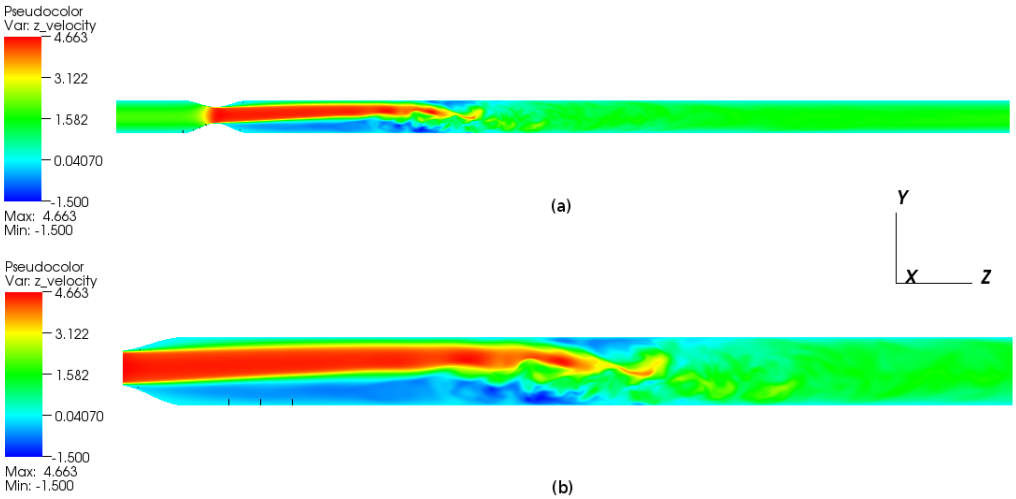



Figure 5.19: Stream wise velocity magnitude plot corresponds to the time 247. The reconstructed flow field presented in figure (b) is the same as the initial snapshot generated by Nek5000 in figure (a).

# Chapter 6

# Conclusion

The 3D eccentric stenosis pipe with a steady flow is chosen as a study case to analyse the coherent structures of the turbulence flow in post stenosis area. One method to study these structures is Modal Decomposition. To implement modal decomposition there was initial requirements such as gathering appropriate snapshots from the case. On the other hand, have a good understanding from the different existing algorithm for modal decomposition was important that two of them were considered for current study; Proper Orthogonal Decomposition(POD) and Dynamic Mode Decomposition(DMD). As a first step, a literature study has been done on previous papers that they introduced different methods. The following point was carried out from the literatures:

- There are different methods to implement the POD where the two most common among them are snapshot method and classical or direct method.
- The POD method still useful in many cases to study coherence structures of the flow with complexity e.g. turbulent flows. However, there are some issues that were discussed in previous studies. For instance ranking the structures according to the energy may not be a good measure. Because, maybe some dynamically highly relevant structures neglect as they have zero energy [9].
- These mentioned problems indicate that another method is required that could rank the flow according to the frequency of the structures that is called DMD. There are different way to implement this method which has been discussed in §2.
- The method proposed by [9] has been considered to use for current thesis. As it is discussed in §2 other methods face problem when the system contaminated by noise.

For simulation and gathering snapshots `Nek5000` has been used. To keep the turbulent structure in the domain an un-physical force has implemented on the system. The snapshots were gathered when statistics showed that the initial transition had left the domain. Therefore, after studying domain dependency, convergence and resolution study the final 600 snapshots were gathered. For the next step, there was a need to develop a code to implement modal decomposition on the set of snapshots. The code was implemented according to the three methods, snapshot POD, classical POD and DMD method of [9]. There was some bottle-neck during the implementation of the code. The most important

one was related to the lack of memory regarding to the loading large data set and keeping them in memory to do the computation. Another problem was related to read the snapshots from `Nek5000` output file format. Therefore, two subroutines has been developed to read from snapshots and write modes to the mentioned format. Finally there is a package called `mode.f90` that is available to apply POD and DMD which is accelerated by OpenMP.

Validation of the code is done by running both methods for the 2D flow passing a cylinder. As a final step, the code has been run for stenosis pipe flow for both methods. The coherent structure corresponds to the implemented force was captured by POD, however, the rest of the structures with low frequencies does not present any correlation. The energetic dynamic structure, related to the force, recognized by dynamic decomposition, does not show agreement with our expectation in sense of frequency of the force (see more in §5). The reason could be a small problem in the code that still remained unsolved.

To improve the code one can add the snapshot DMD method which makes the DMD code memory independent. Also to accelerate the code LAPACK routines can be substituted by ScaLAPACK ones. Also changing some part of the code to avoid reading unnecessary data from snapshots can effect the performance noticeably. As an improvement for the reconstruction codes, one can rewrite these subroutines in another way to reduce amount of operations or combine these two subroutines together.

# Acknowledgements

This essay is a Diploma Thesis for the master degree in Scientific Computing at KTH (Royal Institute of Technology) and is done at Mechanics departments at KTH.

This dissertation is done under the supervision of Dr. Philipp Schlatter. So my first and foremost thanks goes to him for his guidance and his helps.

I would like to show my gratitude to my dear colleagues Reza Dadfar, Azad Noorani, Sasan Sarmast, Adam Peplinski for their productive discussion.

# Bibliography


[1] Ku D. N., *Blood flow in arteries*, Ann. Rev. J. Fluid Mech. 29. 27-34, (1997)

[2] Young D. F., *Fluid mechanics of arterial stenosis*, J. Biomech. Eng. 101, (1979 )

[3] Ahmed S. A., Giddens, D. P. *Velocity measurements in steady flow through axi-symmetric stenoses at moderate Reynolds number* , J. Biomech. (1983)

[4] Varghese S., Frankel S. H., Fischer P. F., *Direct numerical simulation of stenotic flows. Part1: Steady flow*, J. Fluid Mech. (2007)

[5] Varghese S., Frankel S. H., Fischer P. F., *Direct numerical simulation of stenotic flows, Part2: Pulsatile flow*, J. Fluid Mech. (2007)

[6] Pope S. B., *Turbulent Flows*, Cambridge University Press (2010)

[7] Manhart M., Wngle H., *A spatiotemporal decomposition of a fully inhomogenous turbulent flow field*, Theoret. Comput. J. Fluid Dynamics (1993)

[8] Berkooz G., Holmes P., Lumley L., *The Proper orthogonal decomposition in the analysis of turbulent flows*, Annu. Rev. J. Fluid Mech. (1993)

[9] Shcmid P.J., *Dynamic mode decomposition of numerical and experimental data*, J. Fluid Mech. (2010)

[10] Chen K. K. , Tu J. H., Rowley C.W., *Variants of Dynamic Mode Decomposition: Boundary Condition, Koopman, and Fourier Analyses*, Springer Science+Business Media, LLC (2012)

[11] Schmid P. J., Violato D., Scarano F., *Decomposition of time-resolved tomographic PIV*, Springer-Verlag (2012)

[12] Schmid P. J., Li L., Juniper M. P., Pust O., *Application of the dynamic mode decomposition*, Springer-Verleg (2010)

[13] Rowley C. W., Mezic I., Bagheri S., Schlatter P., Henningson D. S., *Spectral analysis of nonlinear flows*, J. Fluid Mech. (2009)

[14] Sarmast S., Schlatter P., Ivanell S., Mikkelsen R. F., Henningson D. S., *Stability of the helical tip vortices behind a wind turbine*, Submitted to EUROMECH Colloquium 528 conference,(2012)

[15] Schmid, P. J., Hanningson, D. S., *Stability and Transition in Shear Flows*, Springer, First edition (2001)

[16] Fischer P.F., *Analysis and application of a parallel spectral element method*, Elsevier Science Publication,1990

[17] Holmes P., Lumeley J. L., Berkooz G., Rowley C. W., *Turbulence, coherent structures, dynamical systems and symmetry*, Cambridge, Second edition (2012)

[18] Fischer P. F., Lottes J. W., Kerkemeier S. G. (2008) nek5000 web page, http://nek5000.mcs.anl.gov

[19] Jeong J., Hussain F., *On the identification of a vortex*, J. Fluid Mech. (1995)

[20] netlib user guide, third edition (1999), http://http://www.netlib.org/lapack/lug

[21] Rempfer D., Fasel H., *Evolution of three dimensional coherent structures in a flat plate boundary layer*, J. Fluid Mech. (1994)