

Unsteady Metric Based Grid Adaptation using Koopman Expansion

Cherith Lavisetty

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Aerospace Engineering

Luca Massa, Co-chair
Rakesh K. Kapania, Co-chair
Danesh Tafti

May 8, 2024
Blacksburg, Virginia

Keywords: Computational Fluid Dynamics, Anisotropic Grid Adaptation, Dynamic Mode
Decomposition, Koopman Operator, Unsteady Flows.

Copyright 2024, Cherith Lavisetty

Unsteady Metric Based Grid Adaptation using Koopman Expansion

Cherith Lavisetty

ABSTRACT

Unsteady flowfields are integral to high-speed applications, demanding precise modelling to characterize their unsteady features accurately. The simulation of unsteady supersonic and hypersonic flows is inherently computationally expensive, requiring a highly refined mesh to capture these unsteady effects. While anisotropic metric-based adaptive mesh refinement has proven effective in achieving accuracy with much less complexity, current algorithms are primarily tailored for steady flow fields. This thesis presents a novel approach to address the challenges of anisotropic grid adaptation of unsteady flows by leveraging a data-driven technique called Dynamic Mode Decomposition (DMD). DMD has proven to be a powerful tool to model complex nonlinear flows, given its links to the Koopman operator, and also its easy mathematical implementation. This research proposes the integration of DMD into the process of anisotropic grid adaptation to dynamically adjust the mesh in response to evolving flow features. The effectiveness of the proposed approach is demonstrated through numerical experiments on representative unsteady flow configurations, such as a cylinder in a subsonic flow and a cylinder in a supersonic channel flow. Results indicate that the incorporation of DMD enables an accurate representation of unsteady flow dynamics. Overall, this thesis contributes to making advances in the adaptation of unsteady flows. The novel framework proposed makes it computationally tractable to track the evolution of the main coherent features of the flowfield without losing out on accuracy by using a data-driven method.

Unsteady Metric Based Grid Adaptation using Koopman Expansion

Cherith Lavisetty

GENERAL AUDIENCE ABSTRACT

Simulating unsteady, high-speed fluid flows around objects like aircraft and rockets poses a significant computational challenge. These flows exhibit rapidly evolving, intricate pattern structures that demand highly refined computational meshes to capture accurately. However, using a statically refined mesh for the entire simulation is computationally prohibitive. This research proposes a novel data-driven approach to enable efficient anisotropic mesh adaptation for such unsteady flow simulations. It leverages a technique called Dynamic Mode Decomposition (DMD) to model the dominant coherent structures and their evolution from snapshot flow field data. DMD has shown powerful capabilities in identifying the most energetic flow features and their time dynamics from numerical or experimental data. By integrating DMD into the anisotropic mesh adaptation process, the computational mesh can be dynamically refined anisotropically just in regions containing the critical time-varying flow structures. The efficacy of this DMD-driven anisotropic adaptation framework is demonstrated in representative test cases - an unsteady subsonic flow over a circular cylinder and a supersonic channel flow over a cylinder. Results indicate that it enables accurate tracking and resolution of the key unsteady flow phenomena like vortex shedding using far fewer computational cells compared to static mesh simulations. In summary, this work makes anisotropic mesh adaptation computationally tractable for unsteady flow simulations by leveraging data-driven DMD modelling of the evolving coherent structures. The developed techniques pave the way for more accurate yet efficient unsteady CFD simulations.

Dedicated to my family.

Acknowledgments

I would like to sincerely thank my advisor Dr. Luca Massa for taking a chance on me with the complex research projects and for having the belief in my abilities, which was incredibly encouraging for me to believe in myself. I truly admire your hard-working nature and promise to try to live up to your work ethic.

I would also like to take this opportunity to thank Dr. Rakesh K. Kapania for his valuable guidance related to my presentations to NASA and professional conduct in general. Your insights and advice have been invaluable throughout my research journey so far.

Additionally, I extend my heartfelt gratitude to Dr. Danesh Tafti for his course on Computational Fluid Dynamics (CFD), which provided me with a fundamental base in numerical fluid mechanics to pursue deeper studies. I also appreciate your willingness to join my advisory committee.

I would also like to acknowledge the support from NASA through the STTR project, which has been instrumental in advancing my research.

Special thanks go to Edward Szwabowski of M4 Engineering and Dr. Gabriel Nastac from NASA for their help and support throughout this journey.

Lastly, to my roommates and friends in Blacksburg who resonated with my goofiness and shared our love for football and movies, your companionship and support have been invaluable, and I am grateful to have you all in my life.

Contents

List of Figures	ix
List of Tables	xi
List of Abbreviations	xii
1 Introduction	1
1.1 Motivation for the Adaptation	1
1.2 Anisotropic mesh adaptation on unsteady flows	2
1.3 Data driven methods for feature extraction	3
1.4 Outline	4
2 Mesh Adaptation	6
2.1 Overview	6
2.2 Metric Spaces	7
2.2.1 Euclidean Metric Spaces	8
2.2.2 Riemannian Metric Space	11
2.3 Interpretation of Unit Mesh	12
2.4 Metric Interpolation	14
2.5 Control of Interpolation Error	16
2.6 Adaptation Mechanics	17
2.6.1 Full-Adaptation	18
2.6.2 r-Adaptation	19

3	Dynamic Mode Decomposition	21
3.1	Overview	21
3.1.1	Koopman theory	21
3.1.2	Triple Decomposition using DMD	22
3.1.3	Computation of Koopman Modes	23
3.1.4	Amplitudes of the DMD modes	25
3.1.5	Choice of Observables	25
3.2	Validation of DMD	26
3.2.1	DMD on data from Linear Dynamical system	26
3.2.2	DMD Employed on Nonlinear Data	28
4	Unsteady Mesh Adaptation	32
4.1	Outline	32
4.2	DMD on moving meshes	34
4.3	Algorithm Implementation	35
5	Results and Discussion	36
5.1	CFD SOLVER	36
5.2	DMD reconstruction	36
5.2.1	Cylinder in a Subsonic flow	36
5.2.2	Cylinder in a Supersonic flow	40
5.3	Grid Adaptation using DMD	45
6	Conclusions and Future Work	50
	Appendices	52

A Appendix I	52
A.1 Proof for linearization in 3.2.2	52
A.2 r-adapted grids for the Subsonic Case	54
Bibliography	55

List of Figures

2.1	Linear interpolate of u on uniform and optimal mesh	7
2.2	Unit balls associated with metric $\mathcal{M} = \mathcal{R}\mathcal{A}\mathcal{R}^T$ in 3D and 2D	10
2.3	Mapping between unit balls in computational space and reference space	10
2.4	Parameterized curve on Riemannian manifold	11
2.5	Adaptation operators	18
2.6	Full adaptation	19
2.7	R-Adaptation	20
3.1	Pressure Distribution of the flow field	27
3.2	Spatial Modes (left singular vectors) and Temporal Modes (right singular vectors) obtained from SVD are plotted. Each colour represents a distinct vector. The x-axis of the top plot denotes the spatial coordinates, while the x-axis of the bottom plot corresponds to the time instant or time coordinate. The y-axis of both plots corresponds to the mode value at the respective space and time coordinates respectively.	28
3.3	Spatial Modes obtained from DMD	29
5.1	Subsonic flow around cylinder	37
5.2	Amplitudes distribution and eigenvalues plot	38
5.3	Reconstruction and Prediction Error(%)	38
5.4	3990-20000 predictions using DMD on 300 snapshots	39
5.5	600 th prediction using DMD on one shedding cycle	39
5.6	Feature fields at an arbitrary time instant	41
5.7	Time evolution of flow-field	41

5.8	Amplitudes distribution and eigenvalues plot	42
5.9	Eigenvalue distribution from DMD using 255 and 260 Snapshots	42
5.10	multi-resolution map of both test cases	43
5.11	DMD reconstruction using 250 Snapshots	43
5.12	Error w.r.t true snapshot using DMD prediction and Mean flow	44
5.13	Input Mesh of Subsonic Case	45
5.14	Grids adapted with actual solution and DMD feature field	46
5.15	Region Division	46
5.16	Input Mesh of Supersonic Case	47
5.17	Grids adapted for Supersonic case	48
A.1	R-adapted grids	54

List of Tables

5.1	Reconstructions using DMD on actual flowfield and FFT filtered flow . . .	45
5.2	Element Distribution	47
5.3	Element distribution for supersonic case	49

List of Abbreviations

D_α	Diagonal Matrix
α	Amplitudes
$\gamma(t)$	Parameterized curve with parameter t
Λ	Eigenvalues of \tilde{A}
\mathcal{M}	Metric tensor
ω	Frequencies of temporal modes
Ψ	Spatial Modes of Koopman Operator
Σ	Singular Values Matrix
\tilde{A}	Koopman Operator in Lower Dimensional Subspace
A	Koopman Matrix
F	Frobenius Norm
$H_u(\mathbf{x})$	Hessian of a solution u
$J(\alpha)$	Objective Function
$l_{\mathcal{M}}(\mathbf{e})$	length of an edge w.r.t Riemannian metric space
P_i	Linear interpolate
Re_D	Reynolds number based based on a reference length D
St	Strouhal number
U	Left Singular Vectors Matrix
V^*	Right Singular Vectors Matrix
V_{and}	Vandermonde Matrix

- X Data Matrix of first N snapshots
- X' Data Matrix of last N snapshots
- Y Eigenvectors of \tilde{A}

Chapter 1

Introduction

1.1 Motivation for the Adaptation

Partial Differential Equations (PDEs) hold significant interest for the scientific community as they encapsulate fundamental laws governing various physical phenomena. However, their nonlinear structure and limited analytical tools pose challenges to finding exact solutions. Thus, numerical techniques such as Finite Element Methods, Finite Difference Methods, Finite Volume Methods, and Discontinuous Galerkin methods are employed to obtain approximate solutions. These numerical techniques play a vital role in solving PDEs by discretizing continuous domains into finite-dimensional spaces through meshing. Discretizing the continuous domain into a discrete representation in terms of nodes inherently introduces discretization errors in the solutions obtained by these numerical methods. Achieving higher accuracy in numerical solutions hinges on mesh resolution. Finer meshes closely approximate the continuous domain, reducing discretization errors compared to coarser meshes. This finer resolution is particularly crucial in capturing regions with high gradients in flow states, such as boundary layers, shocks, shear layers, and separated flows.

In the context of high-speed flows, such as those encountered in Entry, Descent, and Landing (EDL) systems, accurate modelling is paramount. Unsteady flows and regions with high gradients in flow states add complexity to simulations, necessitating highly refined meshes for accurate representation. To mitigate computational overhead while preserving solution accuracy, mesh adaptation advocates employing finer mesh elements or denser node distributions only in regions with pronounced spatial variations. This optimizes computational resources and focuses refinement where it is most needed.

Mesh adaptation mainly consists of identifying large variations of flowfield or detection of crucial flow features by monitoring the solution errors and then modifying the mesh elements accordingly to ensure a uniform distribution of these errors in all the mesh elements by the

well-known equidistribution principle as explained in [1]. The majority of the researchers have worked on isotropic mesh adaptation where they perform refinement at respective regions in a uniform manner, dividing the mesh element into equal proportions. The problem with such an approach is that often these techniques tend to accumulate way too many cells in regions with large solution gradients or large solution errors, thus increasing the computational expense significantly. Although we achieve a reduction in computational intensity as compared to completely uniform meshes to achieve a given accuracy in the solution, a more effective representation of meshes is needed.

To that end, a directional property of the interpolation error was introduced in [2] which provided measurement of errors involving directions which gave rise to the idea of generating elements with high aspect ratios using local mapping procedures by reduction of the interpolation errors distributed in a directional sense. The key aspect of mesh adaptation is to control the size, shape, and orientation of mesh elements in the computational domain. Later, Ref.[3] introduced the concept of metric-based adaptation. The specifics of the current state-of-the-art metric-based anisotropic adaptation can be found in Ref.[4].

1.2 Anisotropic mesh adaptation on unsteady flows

However, the existing Metric-based Grid Adaptation algorithms have been typically designed and performed on steady-state simulations which precludes their use in the unsteady EDL applications. The unsteadiness in this type of flow requires a time-accurate approach for modelling because in some cases a steady solution may not even exist. In short, a time-accurate solution with an adequate level of mesh resolution without the need for any extreme refinement is needed for an accurate and efficient simulation of these types of problems.

To consider the progression of the flow in the mesh adaptation process a transient fixed-point algorithm has been proposed in [5], which consists of an inner loop which ensures the convergence of the initial solution at a transient fixed point in an iterative manner to ensure all the potential dynamic regions are resolved in an average sense over the time step duration. The main idea behind the above approach is that for “*explicit algorithms under the CFD conditions the error in space controls the error in time, hence to control the solution accuracy we have to control the error in space throughout a given time frame of the computation*”. Briefly, we evolve the solution from a time instant t to $t + \Delta t$ and then this solution is iterated

inside an internal loop to ensure its convergence at a fixed point of time and parallelly adapt the meshes taking into account a notion of metric intersection in time to capture all the important time scales over the whole time step.

The algorithm proposed in [5] is only valid when the space-time interpolation error is controlled in L^∞ norm. Later an L^p norm variant has been presented in [6] where the minimisation of space-time interpolation error on a continuous mesh is solved in a decoupled manner: first the instantaneous interpolation error in space is evaluated, then a temporal minimization is performed for an optimal space-time mesh. A notion of an unsteady metric has been derived in this work which has a dependence on the instantaneous sub-interval of time. But to obtain the normalization factor of this metric first we need to evolve the solution at least once over the whole time interval to obtain all the Hessians $H_{\mathbf{u}}(\mathbf{x}, t)$ at sub time intervals which is not practical for a large unsteady simulation. To that end, we are proposing to use data-driven methods to find the key features of the flow to capture their evolution in time which can help us perform the grid adaptation for unsteady flows in a more simple and robust way.

1.3 Data driven methods for feature extraction

While an unsteady flow can appear chaotic and random, it often conceals hidden patterns and structures which can be uncovered through careful analysis and understanding. Data-driven techniques, such as Proper Orthogonal Decomposition (POD)[7, 8, 9], and Dynamic Mode Decomposition (DMD) have garnered significant attention for analyzing unsteady flows due to their ability to extract coherent structures and dominant modes from high-dimensional datasets. POD is a widely used technique that provides a lower-dimensional representation of the flow field based on the Singular Value Decomposition (SVD) [10] of the snapshot matrix, which decomposes the flow field into a set of orthogonal spatial modes and their corresponding time coefficients. Although POD is efficient in capturing the dominant spatial structures, it lacks temporal resolution, making it challenging to identify the underlying frequencies associated with the modes.

This limitation can be overcome by employing the Spectral Proper Orthogonal Decomposition (SPOD)[11], which incorporates frequency information by applying the SVD to the cross-spectral density tensor of the flow field. An alternative approach to obtain modal

decompositions with both spatial and temporal information is the Dynamic Mode Decomposition (DMD) [12]. DMD approximates the flow dynamics by a linear operator, allowing the extraction of dynamically relevant modes and their corresponding frequencies. Unlike SPOD, DMD does not require the calculation of spatial correlations, making it computationally more efficient for large datasets. The connection between the DMD and Koopman theory is established in [13], where the Koopman theory follows the process of representing a lower-dimensional nonlinear system in state space to a higher dimensional linear system in the observable space that leads us to estimate a Koopman operator that would enable us to project the observables to a future state in a linear sense.

To this end, the DMD tries to approximate the infinite-dimensional Koopman operator as a linear model advancing our spatial measurements in time. Often the non-linearities are too high in fluid flows which makes it very difficult for DMD to resolve all the non-linearities due to the absence of a rich set of observables. This leads to much more robust algorithms like Extended Dynamic Mode Decomposition (EDMD) as introduced in [14] where one performs a data-driven analysis of selecting a rich set of dictionary functions to best represent our nonlinear system in a linear fashion. Although the EDMD approach represents the Koopman operator better than the standard DMD, an optimal set of dictionary functions is not known *a priori* to us making it computationally extensive to optimise the best set of observable functions in a rather extensive library of functions. To that end, a simple DMD analysis has been used to capture the evolution of the unsteady features of a flow field in this thesis. A brief outline of the thesis is given below.

1.4 Outline

Chapter 2 provides a basic review of some mathematical preliminaries and context required for the anisotropic mesh adaptation, namely introducing the concepts of Riemannian metric spaces, the properties of a metric tensor and its ability to convey the directional properties of the interpolation error as well as introducing the notions of unit mesh. This chapter also discussed our perspective of a full adaptation and r-adaption which will be utilised in a sequential manner in our unsteady adaptation algorithm described in Chapter 4.

Chapter 3 provides a very brief review of the Dynamic Mode Decomposition algorithm and shows a simple application using two example problems to demonstrate its working

methodology of extracting coherent spatial and temporal dynamics.

Chapter 4 introduces our approach to perform the grid adaption for unsteady flows using full-adaptation and r-adaptation which leverages dynamic mode decomposition for feature tracking.

Chapter 5 presents the results of using the approach developed in this thesis to solve a couple of examples of problems and explores a few factors that might possibly be crucial factors for the applicability of this approach.

Chapter 6 finally concludes this thesis with closing remarks and a brief discussion about future work that we plan to accomplish and suggesting possible extensions of the approach taken to the grid adaptation of unsteady flows.

Chapter 2

Mesh Adaptation

2.1 Overview

Before analysing mesh adaptation let us first understand the meaning of linear interpolate. Let Ω be the computational domain of \mathbb{R}^n we are interested in and \mathcal{H} be the mesh of Ω with \mathbf{N} vertices. Assume that the exact solution of the PDE is a function of space $g : \mathbb{R}^n \rightarrow \mathbb{R}$, then the linear interpolate of g on \mathcal{H} , written as $\Pi_{\mathcal{H}}g$, which is the piece-wise linear representation of g on \mathcal{H} . The difference between these two quantities is hereby denoted as the **interpolation error** $e(g)$ in this thesis, which in a geometric sense is the gap between the actual function and the piece-wise linear function; $e(g)$ can thus be understood as a measure of the accuracy of the solution.

$$e(g) = \|g - \Pi_{\mathcal{H}}g\|_{L^p(\Omega)} \tag{2.1}$$

So having defined the interpolation error of the numerical solution, the problem for us to solve now is to find an optimal mesh with a given number of vertices which reduces the interpolation error. So essentially we need to find an optimal mesh that best represents the solution while maintaining a linear variation between the nodal points. An example of what optimal nodal positions look like and how well such meshes perform compared to a uniform mesh with the same number of nodes can be seen in Fig 2.1. One of the widely known principles that guides this adaptation process is the equidistribution principle [1] which essentially searches for a mesh in which the error density is distributed evenly among the elements of the mesh instead of any accumulation of error in specific regions. Most of the initial approaches as explained earlier have been focused on isotropic adaptation where the regions of high errors are divided isotropically in all directions uniformly which does distribute the error and improves the accuracy although comes with an increased computational expense. However as discussed in the previous chapter a directional property of the interpolation error was

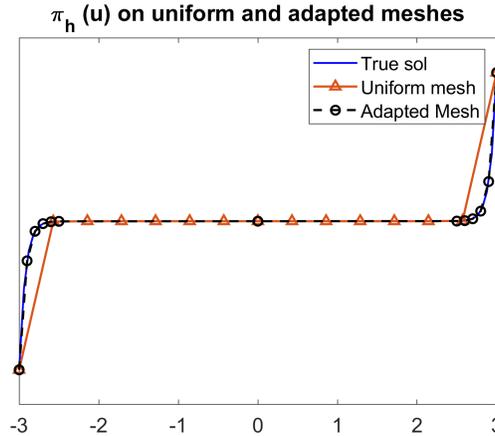


Figure 2.1: Linear interpolate of u on uniform and optimal mesh

observed in the 80s, which led to the idea to generate stretched elements with high aspect ratios. Along came the idea of using a Riemannian metric to dictate the size, shape and orientation of the mesh that can be communicated to the automatic mesher in a highly elegant manner. The equidistribution of the errors to reduce and achieve better interpolation is achieved by converging towards this notion of a quasi-unit mesh with respect to a Riemann metric space. Hence, a brief review of the mathematical concepts related to metric spaces and the notion of unit mesh will be discussed in this chapter which will help us understand more about converging towards optimal anisotropic meshes.

2.2 Metric Spaces

A basic idea of metric spaces is essential for us to understand about anisotropic meshing, hence we shall start with a few known definitions and further discuss the key ideas that are essential for us. A closed set \mathbb{K}^n of n -tuples $k = (k_1, k_2, \dots, k_n)$, termed vectors, forms a **Vector space** if these vectors can be added or multiplied by scalars, preserving the properties of commutativity, associativity, and distributivity in ordinary arithmetic. An inner product $\langle \cdot, \cdot \rangle$ is a way to multiply vectors in a vector space with the result being a scalar quantity. Finally, a set (X) is associated with a distance function (d) which allows us to evaluate the distance between any two arbitrary points in the set then it is termed as a **Metric space** (X, d) .

2.2.1 Euclidean Metric Spaces

An Euclidean metric space is a finite vector space where the inner product is defined by means of a symmetric definite positive tensor \mathcal{M} :

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{M}} = \langle \mathbf{u}, \mathcal{M}\mathbf{v} \rangle = \mathbf{u}^t \mathcal{M}\mathbf{v}, \quad \text{for } (\mathbf{u}, \mathbf{v}) \in \mathbb{R}^3 \times \mathbb{R}^3 \quad (2.2)$$

And the matrix \mathcal{M} is the metric tensor which is the metric that we were referring to in this thesis so far. The Euclidean space (\mathbb{R}^3) which we extensively use for most of our physical analyses is a special case of this Euclidean metric space when the metric tensor is the Identity matrix, hence resulting in the familiar dot product $\langle \mathbf{u}, \mathcal{M}\mathbf{v} \rangle = \langle \mathbf{u}, \mathbf{v} \rangle$.

In such Euclidean metric spaces, the norm of a vector as well as distance between two points are defined as:

- $\forall \mathbf{u} \in \mathbb{R}^3, \quad \|\mathbf{u}\| = \sqrt{\langle \mathbf{u}^T, \mathcal{M}\mathbf{u} \rangle}$
- $\forall (P, Q) \in \mathbb{R}^3 \times \mathbb{R}^3, \quad d_{\mathcal{M}}(P, Q) = \sqrt{(\mathbf{PQ})^T \mathcal{M}(\mathbf{PQ})}$

Hence, the lengths of the segments are defined as the distance between its extremities as:

$$l_{\mathcal{M}}(\mathbf{pq}) = d_{\mathcal{M}}(\mathbf{p}, \mathbf{q}) = \|\mathbf{pq}\|_{\mathcal{M}} = \sqrt{(\mathbf{q} - \mathbf{p})^T \mathcal{M}(\mathbf{q} - \mathbf{p})}. \quad (2.3)$$

In Euclidean metric spaces the volumes and angles are well defined which are also of interest for us in the context of meshing.

The angle between two vectors \mathbf{u} and \mathbf{v} can be computed as follows:

$$\text{Cos}(\theta_{\mathcal{M}}) = \frac{\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{M}}}{\|\mathbf{u}\|_{\mathcal{M}} \|\mathbf{v}\|_{\mathcal{M}}}. \quad (2.4)$$

This is the angle between two vectors in 2D space and the dihedral angle between two surfaces in a 3D space. Also, given a bounded space K , with a Euclidean volume $|K|_{I_3}$, the volume K with respect to \mathcal{M} is computed as:

$$|K|_{\mathcal{M}} = \sqrt{\det \mathcal{M}} |K|_{I_3} \quad (2.5)$$

where I_3 signifies metric tensor in Euclidean space. As the metric tensor \mathcal{M} is a SPD matrix, it can be diagonalized, and a spectral decomposition of the tensor can be written as follows:

$$\mathcal{M} = \mathcal{R}\Lambda\mathcal{R}^T \quad (2.6)$$

where \mathcal{R} are the eigenvectors ($\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$) that are orthonormal and Λ diagonal matrix of eigenvalues of \mathcal{M} denoted by $(\lambda_i)_{i=1,2,3}$, which are strictly positive and are arranged in descending order. The geometric interpretation of these eigenvalues and eigenvectors are key to understanding the mapping between reference and our computational domains. To understand this mapping, let us explain the notion of unit ball. First consider that set $\Phi_{\mathcal{M}}(\epsilon)$ represents all the points that are in the vicinity of point \mathbf{p} denoted as $\mathcal{V}(\mathbf{p})$, that are a distance ϵ , then this set can be represented as:

$$\Phi_{\mathcal{M}}(\epsilon) = \{x \in \mathcal{V}(\mathbf{p}) \mid \sqrt{(\mathbf{x} - \mathbf{a})^T \mathcal{M} (\mathbf{x} - \mathbf{a})} = \epsilon\}. \quad (2.7)$$

The distance can be replaced by a unit distance which gives us the set of all the points that are at a unit distance from the point \mathbf{p} and the region enclosing all the subset of points is called as the unit ball $\mathcal{B}_{\mathcal{M}}$ of \mathbf{p} with respect to \mathcal{M} . Rewriting the above equation for unit distance gives us:

$$\Phi_{\mathcal{M}}(1) = \{x \in \mathcal{V}(\mathbf{p}) \mid \sum_{i=1}^n (x_i - p_i)^T \mathcal{M} (x_i - p_i) = 1\}. \quad (2.8)$$

The eigenvectors of \mathcal{M} which are the column vectors of \mathcal{R} are the orthonormal basis which tells us about the local orientation of the computational domain relative to the reference domain. Hence, let us transform our domain into the eigenvectors frame and assume that coordinates in the transformed bases are $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{p}}$. Consequently, we simplify the above equation into:

$$\Phi_{\mathcal{M}}(1) = \{\tilde{x} \in \mathcal{V}(\tilde{\mathbf{p}}) \mid \sum_{i=1}^n \lambda_i^2 (\tilde{x}_i - \tilde{p}_i)^2 = 1\} \quad (2.9)$$

$$= \{\tilde{x} \in \mathcal{V}(\tilde{\mathbf{p}}) \mid \sum_{i=1}^n \left(\frac{\tilde{x}_i - \tilde{p}_i}{h_i}\right)^2 = 1\} \quad \text{where, } h_i = \frac{1}{\lambda_i} \quad (2.10)$$

The simplified result above, when $n = 3$ is an equation of an ellipsoid, centered at \mathbf{p} with the principal directions along the eigen directions of the metric tensor \mathcal{M} and an ellipse in the case of 2D space. The sizes of the principal axes of an ellipse or ellipsoid from the above equation can be deduced as h_i , corresponding the respective eigen directions. Hence $\mathcal{B}_{\mathcal{M}}$, unit ball of \mathcal{M} is depicted in the Fig 2.2 .

Also note that the stretching factors, which are the sizes of major and minor axes are related

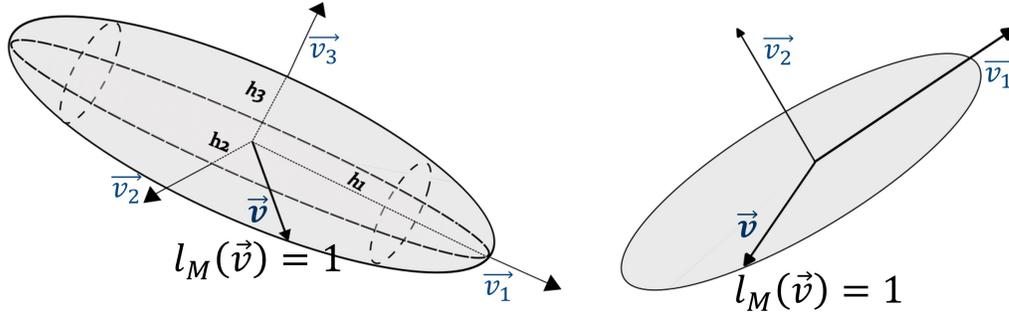


Figure 2.2: Unit balls associated with metric $\mathcal{M} = \mathcal{R}\Lambda\mathcal{R}^T$ in 3D and 2D respectively

to the eigenvalues of \mathcal{M} in an inverse squared sense, hence it is obvious that the mapping from the reference space (spherical ball, \mathcal{B}_{I^3}) to the computational space (unit ball $\mathcal{B}_{\mathcal{M}}$, of \mathcal{M}) is also inverse squared of the tensor $\mathcal{M}^{\frac{1}{2}}$ as shown below:

$$\mathcal{M}^{-\frac{1}{2}} : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \quad (2.11)$$

$$\mathbf{x} \rightarrow \mathcal{M}^{-\frac{1}{2}} \mathbf{x} \quad (2.12)$$

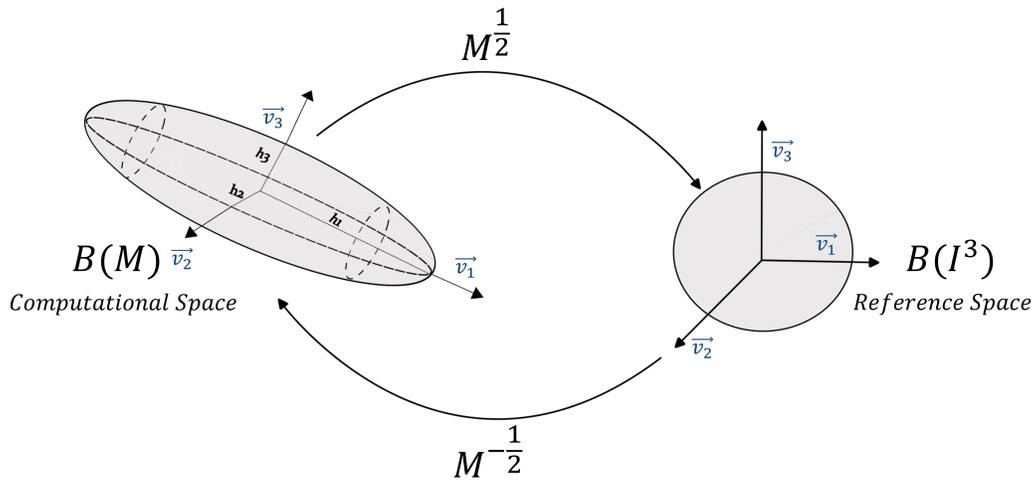


Figure 2.3: Mapping between unit balls in computational space and reference space

2.2.2 Riemannian Metric Space

For a smooth manifold M , if there exists a scalar product defined by a metric tensor \mathcal{M} on the tangent space of the manifold then it is called a Riemannian manifold or Riemannian space (M, \mathcal{M}) . Another way to look at it is, although a Riemannian manifold is curved non-Euclidean manifold, locally at each point \mathbf{p} the tangent space $T_{\mathbf{p}}M$ can be visualised as an Euclidean metric space $(T_{\mathbf{p}}M, \mathcal{M}(\mathbf{p}))$ and hence the Riemannian metric varies smoothly over the manifold. Another way to look at the Riemannian metric space is in the form of a function associated with a higher dimensional Cartesian surface which is useful for visualization. Hence, this Cartesian surface \mathcal{S} of \mathbb{R}^{n+1} with n independent spatial dimensions and a \mathcal{C}^2 scalar function can be defined by:

$$\mathcal{S} = (x_1, x_2, \dots, x_n, \rho(x_1, x_2, \dots, x_n)) \in \mathbb{R}^{n+1} \quad (2.13)$$

Although there is no notion of a global scalar product over the computational space, the Riemannian metric tensor available at a point characterizes the local geometry of the manifold, providing information about distances, angles, and shapes in the vicinity of that point. Let us assume that our computational domain is $\Omega \subset \mathbb{R}^2$, a flat space and each point in this domain are associated with a metric, we can assume the equivalent higher-order Cartesian surface as a curved surface $\mathcal{S} \in \mathbb{R}^3$ as shown in the Fig 2.4 below.

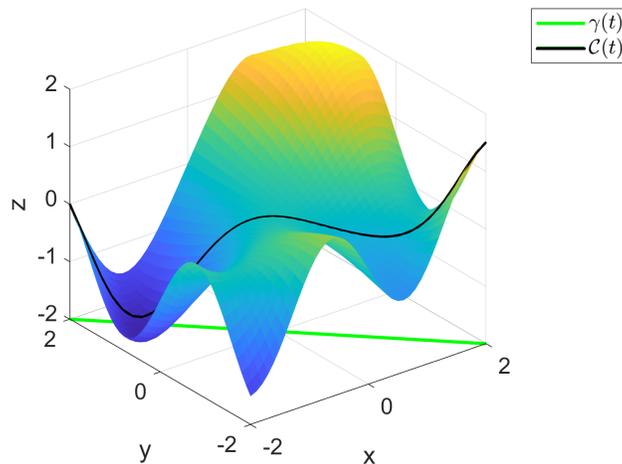


Figure 2.4: Parameterized curve on Riemannian manifold

Now that we have defined this interpretation of embedding a parameterized Riemann surface in higher dimensional space, let us describe finding the distance between two points along a

parameterized curve that is of interest to us. Assume a line γ defined in the computational space Ω just like the edges of our mesh elements, connecting two points \mathbf{p} and \mathbf{q} parameterized by $t \in [0, 1]$ as follows:

$$\gamma(t) = \mathbf{p} + t(\mathbf{q} - \mathbf{p}) \quad (2.14)$$

Now that we are using this notion of constructing a higher dimensional ($S \in \mathbb{R}^3$) space corresponding to this flat 2D space, for each point on the 2D space, there exists a corresponding image on the 3D curved surface. Hence, we can also trace a curve \mathcal{C} connecting all these images on the curved surface \mathcal{S} which is the geometrical representation of the parameterized curve γ on the flat space. Now that we have defined the geometrical interpretation let us find the length of this curve which is essentially the length of the curve mapped by the images on the curved space which can be mathematically calculated by our general definition of distance calculated on a parameterized spaces:

$$\begin{aligned} l_{\mathcal{M}}(\mathbf{p}\mathbf{q}) &= \int_0^1 \|\gamma'(t)\|_{\mathcal{M}} dt \\ &= \int_0^1 \sqrt{\mathbf{p}\mathbf{q}^T \mathcal{M}(\mathbf{p} + t\mathbf{p}\mathbf{q}) \mathbf{p}\mathbf{q}} dt \end{aligned} \quad (2.15)$$

And similarly for a given subset of a region, K of Ω , the volume of K can be computed with respect to Riemannian metric space $(\mathcal{M}(\mathbf{x}))_{\mathbf{x} \in \Omega}$ as an integral over small subregions of K as:

$$|K|_{\mathcal{M}} = \int_K \sqrt{\det \mathcal{M}(\mathbf{x})} d\mathbf{x} \quad (2.16)$$

2.3 Interpretation of Unit Mesh

To generate anisotropic meshes, we need to specify both the size and orientation of elements at each point in the domain. The Riemannian metric tensor provides an elegant solution for this task, allowing us to define local orientation and geometry stretching. This information is crucial for communicating with the mesher to achieve anisotropic grid adaptation. Before discussing how this process optimizes grids to reduce interpolation error, let us introduce a few more key notions related to the adaptation process, namely the concepts of the unit element and unit mesh.

Let us recall that unlike the calculation of distances and volumes in Euclidean spaces, in metric spaces, these quantities are evaluated with respect to a metric tensor. The definitions of lengths of parameterized curves and volumes have been discussed in previous sections. To extend these concepts of distances and volumes to an element K of a mesh, if the length of all the edges of this element is unit w.r.t the prescribed metric \mathcal{M} then the element K is said to be **unit** with respect to the metric \mathcal{M} . Hence for an n -sided element, defined by its list of edges $(\mathbf{e}_i)_{i=1\dots n}$, (where $n = 3$, if it is a 2D triangular element or $n = 6$, if it is a tetrahedron), the element \mathbf{K} is unit with respect to \mathcal{M} if:

$$\forall i = 1, \dots, n, l_{\mathcal{M}}(\mathbf{e}_i) = 1, \quad (2.17)$$

and its volume equal to: $|K|_{\mathcal{M}} = \frac{\sqrt{3}}{4}$ for a triangle (Area in the case of 2D) and $|K|_{\mathcal{M}} = \frac{\sqrt{2}}{12}$ in the case of tetrahedron.

Similarly, the definition of **Unit mesh** is that all the elements of the mesh must be unit with respect to the prescribed metric \mathcal{M} . However, it is often difficult to have all the elements in the discrete mesh be unit since most of the time the Riemannian metric space is not compatible with the domain size most of the times. Hence, the definition of a unit mesh has to be relaxed as follows:

“A discret mesh \mathcal{H} of a domain $\Omega \subset \mathbb{R}^n$ is **unit** for Riemannian metric space $(\mathcal{M}(x))_{\mathbf{x} \in \Omega}$ if all its elements are quasi-unit. [15]”

As the name suggests, an element is said to be a quasi-unit when the length of the edges may not be equal to 1 but are close to the unit with an admissible range. *i.e.*, $\forall i, l_{\mathcal{M}}(\mathbf{e}_i) \in [\frac{1}{\sqrt{2}}, \sqrt{2}]$. Often checking the lengths wouldn't be sufficient to generate good elements, we can often see the formation of flat elements with null volume. Hence, we need to monitor the volume of each element in the case of 3D and the area of the element in 2D to control the notion of the unit element. This is generally monitored using a quality function given below:

$$Q_{\mathcal{M}} = \frac{36}{\sqrt{3}} \frac{|K|_{\mathcal{M}}^{\frac{2}{3}}}{\sum_{i=1}^6 l_{\mathcal{M}}^2(\mathbf{e}_i)} \in [\alpha, 1] \text{ with } \alpha > 0 \quad (2.18)$$

2.4 Metric Interpolation

From a mesh perspective, the two main things we care about are measuring the lengths of edges and the volumes of elements in the mesh. When dealing with a Riemannian metric space, we previously discussed how to find the length of a parameterized curve using continuous integration. However, in a discrete mesh, we don't have a continuous metric tensor; it is only defined at the vertices. Also, instead of using continuous integration, we use more practical discrete methods like quadrature formulas. These formulas require defining the metric at quadrature points, which we don't have readily available.

One of the main benefits of dealing with metric spaces is the clear definition and well-posed nature of operations involving metric tensors. Among these operations, metric intersection and metric interpolation are notably valuable. However, for the scope of this thesis, we will not explore the topic of metric intersection in detail since we are not dealing with multi-metric adaptation. Where the metric interpolation is a key aspect as discussed above, it is then mandatory for us to be able to compute the metric at any point in the domain. Some of the interpolation schemes that were suggested in the past had some issues with the operations being not commutative and often had issues depending on the order of these operations when multiple metric tensors are involved. Hence a log-Euclidean framework was suggested in [16] which solves these issues. Before defining the operations between two metric tensors in this framework, let us first define what a metric logarithm and matrix exponential is: For a metric tensor $\mathcal{M} = \mathcal{R}\Lambda\mathcal{R}^T$ the metric logarithm is given by:

$$\ln(\mathcal{M}) := \mathcal{R} \ln(\Lambda) \mathcal{R}^T \quad (2.19)$$

and a matrix exponential is given by:

$$\exp^{\mathcal{M}} := \mathcal{R} \exp(\Lambda) \mathcal{R}^T \quad (2.20)$$

where $\ln(\Lambda) = \text{diag}(\ln(\lambda_i))$ and $\exp(\Lambda) = \text{diag}(\exp(\lambda_i))$. Hence from the above two definitions, we can now define logarithmic addition \oplus and the logarithmic scalar multiplication \odot as follows:

$$\mathcal{M}_1 \oplus \mathcal{M}_2 := \exp(\ln(\mathcal{M}_1) + \ln(\mathcal{M}_2)) \quad (2.21)$$

$$\alpha \odot \mathcal{M} := \exp(\alpha \ln(\mathcal{M})) = \mathcal{M}^\alpha \quad (2.22)$$

Now the metric interpolation can be done using a linear interpolation operator derived from the above framework. Consider an element K with vertices defined by $(\mathbf{x}_i)_{i=1\dots k}$ such that a metric $(\mathcal{M}(\mathbf{x}_i))_{i=1,\dots,k}$ is defined on each of these vertices. Then for an arbitrary point \mathbf{x} inside the element defined by:

$$\mathbf{x} = \sum_{i=1}^k \alpha_i \cdot \mathbf{x}_i \quad \text{with} \quad \sum_{i=1}^k \alpha_i = 1 \quad (2.23)$$

where the α_i are the barycentric coordinates of the point \mathbf{x} with respect to the element. Then the interpolated metric is defined by:

$$\mathcal{M}(\mathbf{x}) = \bigoplus_{i=1}^k \alpha_i \odot \mathcal{M}(\mathbf{x}_i) = \exp\left(\sum_{i=1}^k \alpha_i \ln(\mathcal{M}(\mathbf{x}_i))\right). \quad (2.24)$$

which upon further simplifications, using the operations discussed above, boils down to:

$$\mathcal{M}(\mathbf{x}) = \prod_{i=1}^k \mathcal{M}(\mathbf{x}_i)^{\alpha_i} \quad (2.25)$$

Now that we have established how to obtain the metric at any point in the domain given a discrete metric field, let us extend these concepts to determine the lengths of edges of a mesh prescribed by a Riemannian metric field. The integral defined in the 2.26 can be approximated numerically using an m points Gaussian quadrature with the weights $(\omega_i)_{i=1\dots m}$ and barycentric coefficients $(\alpha_i)_{i=1\dots m}$:

$$l_{\mathcal{M}}(\mathbf{ab}) = \int_0^1 \sqrt{\mathbf{ab}^T \mathcal{M}(\mathbf{a} + t\mathbf{ab})\mathbf{ab}} dt \approx \sum_{i=1}^m \omega_i \sqrt{\mathbf{ab}^T \mathcal{M}(\mathbf{a} + \alpha_i \mathbf{ab})\mathbf{ab}} \quad (2.26)$$

where $\mathcal{M}(\mathbf{a} + \alpha_i \mathbf{ab})$ is the metric at the i^{th} Gauss point. We have established previously that the edge lengths of the mesh elements are one of the main things that we would be monitoring hence these computations are performed in a recurrent manner, hence further optimization of the calculation would be more advantageous. The key point to focus on is that these lengths are computed at an edge level and the metric is known at the edge points, which are the vertices. Assuming that the metric follows a variation law along the edge, lengths can be computed analytically using the metric at the endpoints. Let us consider an

edge $\mathbf{e} = \mathbf{p}_1\mathbf{p}_2$ to be an edge of the mesh of Euclidean length $\|\mathbf{e}\|_2$, and $\mathcal{M}(\mathbf{p}_1)$ and $\mathcal{M}(\mathbf{p}_2)$ be the metrics at the vertices respectively. Now let us denote $l_i(\mathbf{e}) = \sqrt{\mathbf{e}^T \mathcal{M} \mathbf{e}}$ as the length of the edge in the metric $\mathcal{M}(\mathbf{p}_i)$. And for an arbitrary case consider $l_1(\mathbf{e}) > l_2(\mathbf{e})$ and set $a = \frac{l_1(\mathbf{e})}{l_2(\mathbf{e})}$. Now assuming that the metric varies linearly in a Log-Euclidean way along the edge then the eigenvalues and the stretching factors can be written as follows:

$$\lambda(t) = \exp((1-t)\ln(\lambda_1) + t\ln(\lambda_2)) = \lambda_1^{(1-t)} \lambda_2^t \quad (2.27)$$

$$h(t) = h_1^{1-t} h_2^t \text{ where } h(t) = \sqrt{\frac{1}{\lambda(t)}}, \quad h_1 = \sqrt{\frac{1}{\lambda_1}} \text{ and } h_2 = \sqrt{\frac{1}{\lambda_2}} \quad (2.28)$$

Hence, by assuming that the metric field follows this geometric variation law shown above, upon further simplification we have:

$$l_{\mathcal{M}}(\mathbf{e}) = l_1(\mathbf{e}) \frac{a-1}{a \ln(a)} \quad (2.29)$$

where $l_1(\mathbf{e})$ can also be written as $\frac{\|\mathbf{e}\|_2}{h_1(\mathbf{e})}$. Similarly, the volumes can be computed by using an m -point Gaussian quadrature approximations.

2.5 Control of Interpolation Error

As discussed at the beginning of mesh adaptation, the ideal mesh is the one that approximates the solution smoothly, hence we focus on finding the mesh that is optimal for the interpolation error. The traditional metric-based error minimises the interpolation error in L^∞ norm that tends to equidistribute the spatial error, which gives us the most intuitive understanding of adapting to a unit mesh w.r.t Riemannian metric defined by the bounds on the interpolation error in L^∞ norm as given in [17]. Some of the small-scale features that might be crucial to the flow are not captured by the metric obtained from the L^∞ norm, hence better estimates based on L^1 norm have been proposed in [15], and those based on L^p norm in [18], both of which provides us with a way to resolve multi-scale features of the flow. Hence, the problem that we are solving is to “*find an optimal mesh \mathcal{H} of Ω such that for a given physical quantity u and for a given number of vertices N , the following norm is*

minimized”[18]:

$$\mathbf{E}_{L^p}^h(\mathcal{H}_{L^p}) = \min_{\mathcal{H}} \left(\int_{\Omega_h} |u(\mathbf{x}) - \Pi_h u(\mathbf{x})|^p d\mathbf{x} \right)^{\frac{1}{p}} \quad (2.30)$$

where Π_h is the discrete linear interpolate on the mesh \mathcal{H} . Performing mesh operations to find the optimal topology that minimizes the above error function is a quite difficult task, hence the following discrete problem is recast into a continuous mesh framework suggested in [15], wherein the whole mesh is assumed to be continuous space defined by metric tensor $\mathbf{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x} \in \Omega}$ at each point \mathbf{x} which prescribes the local stretching and orientation. The discrete problem above is recast into the continuous optimization problem as :

$$\text{Find } \mathbf{M}_{L^p} \text{ such that } \epsilon_{L^p}(\mathbf{M}_{L^p}) = \min_{\mathbf{M}} \left(\int_{\Omega} \|u(\mathbf{x}) - \pi_{\mathcal{M}} u(\mathbf{x})\|^p d\mathbf{x} \right)^{\frac{1}{p}} \quad (2.31)$$

where $\pi_{\mathcal{M}}$ is the continuous linear interpolate. The metric which minimizes the above error in the L^p norm is given as:

$$\mathcal{M}_{L^p}(\mathbf{x}) = D_{L^p} (\det |H_u(\mathbf{x})|)^{\frac{-1}{2p+d}} \|H_u(\mathbf{x})\| \quad \text{with} \quad D_{L^p} = \mathcal{N}^{\frac{2}{d}} \left(\int_{\Omega} (\det \|H_u(\mathbf{x})\|)^{\frac{p}{2p+d}} d\mathbf{x} \right)^{\frac{-2}{d}} \quad (2.32)$$

where d is the dimension of the mesh, \mathcal{N} is the spatial complexity which is a constraint set for the 2.31 given as $C(\mathbf{M}) = \int_{\Omega} \sqrt{\det \mathcal{M}(\mathbf{x})} d\mathbf{x} = \mathcal{N}$ which is the analogous to number of vertices in the discrete mesh, and $H_u(\mathbf{x})$ is the hessian of the solution u .

2.6 Adaptation Mechanics

In transitioning from an initial mesh to an optimal configuration with respect to the Riemannian metric, we confront the inherent complexity arising from the continuous variation of the metric tensor across the domain. Unlike in scenarios with a constant metric, each point exhibits distinct stretching and orientation characteristics. Consequently, a direct, single-step mapping between grid configurations becomes infeasible. Instead, the adaptation process involves iteratively adjusting nodal positions to ensure that each mesh element achieves the unit status relative to the Riemannian metric. In the following section, we will briefly outline key operations conducted by the mesher, focusing on achieving a quasi-unit mesh within the Riemannian metric space, and we will discuss two approaches: Full-adaptation

and R-adaptation.

2.6.1 Full-Adaptation

In the context of adaptive meshing techniques used in numerical simulations, a standard adaptive mesher employs fundamental operations such as point insertion, edge removal, edge swapping, and point smoothing, which can be seen in the Fig 2.5 all guided by metric-based quality functions. These operations are executed sequentially within each iteration of the adaptation cycle, and the selection of nodes or edges to be modified is determined by the quality of the elements, ensuring stability and numerical accuracy. Anisotropy considerations play a crucial role, with adjustments made to meet prescribed criteria dictated by the metric field.

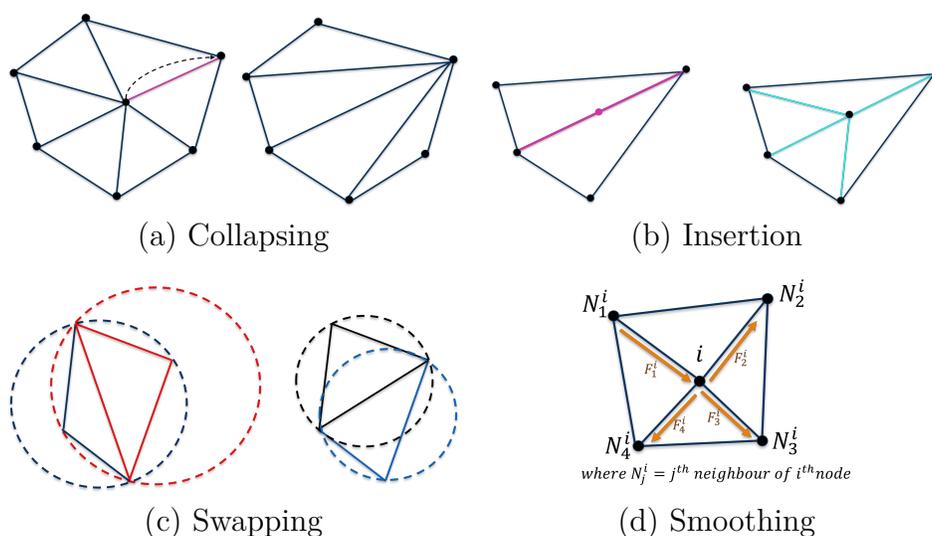


Figure 2.5: Adaptation operators

Mesh smoothing, a key strategy for enhancing mesh quality, involves adjusting node positions to optimize Riemannian lengths, often achieved through solving static equilibrium at each node. However, challenges arise due to the mutual linkage of nodes and elements, wherein modifying nodes to satisfy the anisotropy prescription of the metric tensor for certain elements may impact adjacent elements, potentially leading to saturation of anisotropy levels beyond a certain point. Hence it requires us to add more nodes by point insertion to achieve more freedom to achieve higher aspect ratio elements or removal of nodes by

collapsing edges when some of the edges are too small and the edges after each of these operations edge swapping is also performed to maintain the quality of the meshes. In the context of this thesis, we are terming the approach where all the operations are performed in an adaptive iteration loop as the full adaptation since it uses the full suite of operators to move towards the unit mesh. A scatter plot can be seen below in Fig 2.6(a), where each point corresponds to an edge in the mesh and the y location of each point representing their length w.r.t Riemannian metric space and one can see that all the edges in the mesh satisfy the quasi-unit mesh criteria.

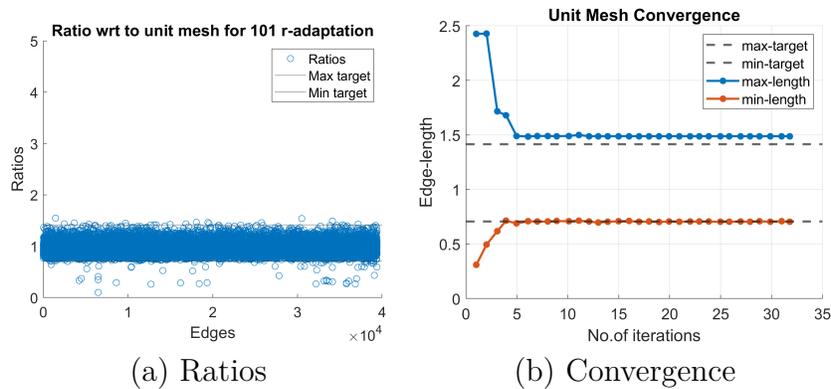


Figure 2.6: Full adaptation

This full-adaptation process is quite computationally intensive for large-scale simulations with very high degrees of freedom. A convergence plot can be seen above in Fig 2.6 where we can see the maximum and minimum of the edge lengths in the mesh to monitor their movement towards the unit mesh per iteration. One can clearly see that as we increase the number of nodes, the number of iterations it takes to converge increases greatly as well and one has to take into account the fact that each of these iterations takes significant time compared to an adaptation of mesh with a lesser number of target nodes.

2.6.2 r-Adaptation

The r-adaptation technique consists of a rearrangement of the nodes based on a pseudo-elastic system of equations. The most widely used r-adaptation techniques are based on some spring analogies where each link between the nodes is modelled as a spring with its stiffness prescribed by the geometrical quantities such as the length of the edge [19] or based

on the value of an actual physical variable [20] distribution on the mesh. So essentially based on the position of the nodes as well as the solution variation (depending on the approach employed), we will have a force imbalance at each of the nodes in an abstract sense, and the nodes are moved in an iterative fashion to achieve a static equilibrium. To define these forces, we are using a smoothing function suggested in [21], although the suggested method is to generate a graded isotropic mesh. We are changing the definition of the distance defined in the paper by the length of an edge computed with respect to the Riemannian metric and the formulation of the force is given in Eq 2.33.

$$f(d) = (1 - d^4).exp(-d^4) \quad \text{where } d(\mathbf{p}_i^{old}, \mathbf{p}_j^{old}) = l_{\mathcal{M}}(\mathbf{p}_i^{old}, \mathbf{p}_j^{old})$$

Hence, if the length of the edge (d) is not unity and ($d < 1$) then there is a repulsive force where as if ($d > 1$) we get an attractive force. And the nodes are adjusted from an initial position \mathbf{p}_i^{old} to a new position \mathbf{p}_i^{new} as given in Eq 2.33.

$$\mathbf{p}_i^{new} = \mathbf{p}_i^{old} + \alpha_i \sum_{j \in N_i} f(d(\mathbf{p}_i^{old}, \mathbf{p}_j^{old})) \mathbf{u}_{ij} \quad (2.33)$$

where α_i is a smoothing factor, \mathbf{u}_{ij} are the unit vectors joining the point \mathbf{p}_i^{old} and the points in the neighbourhood N_i . While relying solely on smoothing may not achieve a perfectly uniform mesh, as discussed earlier, there is a noticeable enhancement in mesh representation, in the sense that the number of edges conforming to the unit size w.r.t the Riemannian metric has increased after smoothing as depicted in Fig 2.7.

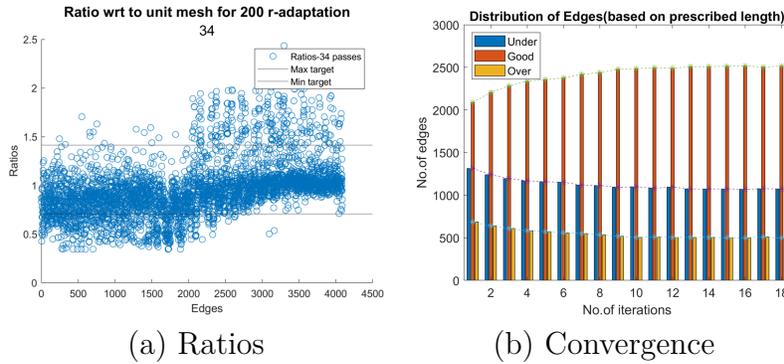


Figure 2.7: R-Adaptation

Chapter 3

Dynamic Mode Decomposition

3.1 Overview

Dynamic Mode Decomposition (DMD) is essentially a data-driven feature extraction technique which can be applied to any form of data which is evolving in time. Unlike other feature extraction techniques such as POD which gives us modes that are spatially orthogonal and multi-frequential temporal content, the DMD modes on the other hand may be non-orthogonal but each distinct spatial mode is associated with a specific temporal frequency. Before defining DMD, it is essential to understand the Koopman theory, which demonstrates a way for us to represent a nonlinear dynamical system in terms of an infinite dimensional linear operator acting on functions spanning Hilbert space.

3.1.1 Koopman theory

According to the Koopman theory, given a rich set of observables, one can find a linear operator acting on the observable space that can help us emulate the non-linear dynamical evolution of the data. The Koopman theory is based on the fact that a nonlinear dynamical system can be represented as a higher dimensional linear system after selecting a right set of observables that can help us achieve this linear formulation [22, 23], and the nonlinear dynamics are coded into the Koopman Operator itself. It is not always possible, and feasible to calculate the infinite-dimensional Koopman operator, hence we resort to its approximate representation. The process of approximating this operator and its eigenfunctions from the time series data is called Koopman Analysis. The DMD is one such data-driven technique that does not exactly evaluate the Koopman Operator but instead calculates the eigenvalues and eigenvectors of the Koopman Operator, which gives us enough information that we need to know about the evolution of the nonlinear dynamical system in hand. In short, the

DMD can be understood as an approximate eigendecomposition of the best-fit operator in a least-squares sense relating two data matrices [24].

3.1.2 Triple Decomposition using DMD

To linearise our dynamical system, instead of actually computing the Koopman Operator, we can achieve the end goal by estimating the spectral properties of the operator alone [25, 26]. It has to be noted that the data matrices for the DMD analysis are formed by taking the measurements which are often referred to as observables of our dynamical system at discrete time instances and each time instance is referred to as a snapshot, and the consecutive snapshots are equidistant in time. Using DMD, we can obtain the Triple Decomposition of this data matrix into Spatial Modes (DMD Modes), Temporal Modes (Eigenvalues) and their corresponding amplitudes [27]. This triple decomposition allows us to interpret the evolution of our dynamical system as a linear combination of spatial modes oscillating in time with distinct frequencies, hence it is obvious to note that the spatial modes and their corresponding amplitudes of each snapshot are the same, but each snapshot varies due to their oscillating component which keeps changing periodically as we progress forward in time. As mentioned earlier, the Koopman Analysis lets us evolve our dynamical system linearly in higher dimensions, so each distinct spatial mode of our decomposition can be evolved linearly with their corresponding oscillating frequency. Hence, we can obtain the state of our dynamical system after a time δt by multiplying our current snapshot by the corresponding eigenvalues of each mode, and these eigenvalues represent the Euler form of the corresponding frequency components ($\lambda_i = \exp(\omega_i \delta t)$). So based on the discussion above we can write the general form of the observables at each snapshot as given in 3.1.

$$\begin{aligned}
 x_0 &= \Phi D_\alpha \\
 x_1 &= \Phi \Lambda D_\alpha \\
 &\vdots \\
 x_{n-1} &= \Phi \Lambda^{n-1} D_\alpha
 \end{aligned} \tag{3.1}$$

The Triple Decomposition in matrix form can be written in the following form 3.2 where \otimes

represents the Kronecker Product:

$$\begin{bmatrix} | & | & & | \\ x_0 & x_1 & \dots & x_{n-1} \\ | & | & & | \end{bmatrix} = \Phi \begin{bmatrix} I & \Lambda & \dots & \Lambda^{n-1} \end{bmatrix} (I \otimes \alpha) = \Phi \text{diag}(\alpha) C \quad (3.2)$$

To obtain the mentioned triple decomposition it is important to form the Vandermonde matrix V_{and} which contains the discrete temporal dynamics, that uniquely identify the partitioning of the data sequence into a triple factorization with a diagonal amplitude matrix. The matrix V_{and} as given by [28] takes the following structure:

$$V_{and} = \begin{bmatrix} 1 & \lambda_1 & \lambda_1^2 & \dots & \lambda_1^{n-1} \\ 1 & \lambda_2 & \lambda_2^2 & \dots & \lambda_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \lambda_n & \lambda_n^2 & \dots & \lambda_n^{n-1} \end{bmatrix} \quad (3.3)$$

3.1.3 Computation of Koopman Modes

In DMD analysis, we have " $n + 1$ " snapshots at discrete points of time with each snapshot consisting of " m " observations or measurements obtained from an experiment or a numerical simulation. In the case of our grid adaptation, we make measurements of several physical quantities such as the pressure, the density or the velocities at each grid point and reshape them into a long column vector that represents the state of that particular snapshot. The data matrix, " $X_{m \times n}$ ", consists of the data from first n snapshots, and Matrix " $X'_{m \times n}$ " consists of last n snapshots arranged sequentially. Ideally, each consecutive snapshot in X must be equidistant in time as mentioned earlier, but even if they are not equidistant we have to make sure that each column in the matrix " X' " is displaced from the corresponding column in X by the same time step.

$$X = \begin{bmatrix} | & | & & | \\ x_0 & x_1 & \dots & x_{n-1} \\ | & | & & | \end{bmatrix} \quad X' = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \dots & x_n \\ | & | & & | \end{bmatrix}$$

The reason for forming the data matrices in the explained manner is to write our system evolution in terms of a Koopman operator as given in Eq 3.4. By definition, if we multiply

the Koopman Operator onto a snapshot at time t , we get a snapshot that is displaced by a time step δt hence the structure of the data matrices.

$$AX = X' \quad (3.4)$$

To find the complete Koopman operator, we can take the inverse of the data matrix X and multiply it by X' . The important thing to note is that the matrix X is not always square, e.g., in our case, the number of measurements per snapshot is more than the number of snapshots hence it is a long and thin matrix, i.e. more columns than rows. Hence, we need to find a pseudo-inverse of our data matrix which can be evaluated using a widely used technique called Singular Value Decomposition (SVD)[29]. Once we find the SVD of the data matrix as shown in 3.5 we can find the inverse of the components and multiply by the data matrix X' as shown in 3.6 to obtain the Koopman operator, which results in a huge matrix of the order of $(m \times m)$ which is not feasible.

$$X = U\Sigma V^* \quad (3.5)$$

$$A_n = X'V\Sigma_n^{-1}U^* \quad (3.6)$$

Hence, before even evaluating this Koopman Operator in 3.6 we can project it onto a lower dimensional subspace as shown in 3.7 to obtain $\tilde{A}_{n \times n}$ which after simplification gives us the following expression given by 3.8.

$$\tilde{A} = U^*AU \quad (3.7)$$

$$\tilde{A} = U^*X'V_n\Sigma_n^{-1} \quad (3.8)$$

We can then find out the eigenvalues and eigenvectors of \tilde{A} by performing its eigendecomposition.

$$\tilde{A}Y = Y\Lambda \quad (3.9)$$

The eigenvalues of the \tilde{A} are the diagonal elements of Λ which are the same as the eigenvalues of the original Koopman operator, which gives us information about the temporal dynamics of our system. And the eigenvectors Y of \tilde{A} can be projected back onto our Koopman space by using 3.10 to get the exact DMD modes Ψ as proven in [24].

$$\Psi = UY \quad (3.10)$$

3.1.4 Amplitudes of the DMD modes

After obtaining the spatial and temporal modes from the lower dimensional representation of the Koopman Operator, the only thing left to evaluate are the amplitudes of the corresponding DMD modes. We can find an optimal set of amplitudes for these modes by finding the least square fit between the data matrix and a linear combination of the DMD modes. So essentially, we need to find the optimal amplitudes that minimise the following Frobenius norm given in 3.11, see [30]

$$\underset{\alpha}{\text{minimize}} \|X - \Phi D_{\alpha} V_{and}\|_F^2 \quad (3.11)$$

and we know that $X = U\Sigma V^*$ and $\Phi = UY$, hence we can use a lower dimensional representation of the objective functions as:

$$\underset{\alpha}{\text{minimize}} \|\Sigma V^* - Y D_{\alpha} V_{and}\|_F^2 \quad (3.12)$$

which can be equivalently represented as shown:

$$J(\alpha) = \alpha^* P \alpha - q^* \alpha - \alpha^* q + s, \quad (3.13)$$

where the matrices P , q and s are given in 3.14, where $*$ and overline represents the Hermitian (conjugate transpose) operation and \circ implies Hadamard multiplication:

$$P = (Y^* Y) \circ (\overline{V_{and} V_{and}^*}), q = \overline{\text{diag}(V_{and} V \Sigma^* Y)}, s = \text{trace}(\Sigma^* \Sigma) \quad (3.14)$$

The optimal amplitudes obtained from solving the optimization problem 3.12 is as follows:

$$\alpha_{dmd} = P^{-1} q \quad (3.15)$$

3.1.5 Choice of Observables

As we have stated before, the DMD is used to find the spectral properties of the approximate Koopman Operator, the power of DMD lies in the fact that our observable space or the observable functional space should be rich enough to represent the non-linear system in a linear sense. Hence, to obtain an accurate decomposition, we need to make appropriate choices of observables for the specific problem at hand. To appropriately capture the nonlinearities in

the flow, we might have to introduce functions of our initial observable space and find the best combination of these non-linear functions. An example problem to demonstrate this choice of observables will be discussed in the following section where we had to include the time derivatives of the initial observations itself in order to obtain the accurate temporal modes.

3.2 Validation of DMD

In this section, the approximation of Koopman modes using DMD is validated presenting a couple of test cases wherein we generate a set of snapshots with known dynamics of evolution, and then perform DMD on these generated snapshots to obtain the DMD modes. For both the test cases, we have set the same mean pressure field distribution, which you can see in the Fig 3.1, and we have added unsteady pressure terms that are oscillating with some frequency in time, and the unsteady terms are also varied according to Gaussian distribution in space. The frequency of the unsteady terms is assigned according to their respective regions as indicated in Fig 3.1. The frequencies with which we are forcing the pressure field are given as follows:

$$\omega_1 = 6.28, \quad \omega_2 = 12.56, \quad \omega_3 = 18.84 \quad \text{and} \quad \omega_4 = 25.13 \quad (3.16)$$

So the goal is to generate these snapshots, and perform DMD analysis on this data set in an attempt to obtain these frequencies and also their spatial distributions.

3.2.1 DMD on data from Linear Dynamical system

For the first test case, we are assuming a linear dynamical system which can be written as

$$\frac{dy}{dt} = cy \quad \text{where } c \text{ is a constant} \quad (3.17)$$

which upon solving by separation of variables gives $y(t) = \exp(ct)$. Hence, the corresponding spatiotemporal data is generated as follows:

$$P_{unsteady}(x, y, t) = P_{steady}(x, y) + \epsilon_j e^{(-x^2-y^2)} e^{i\omega_j t} \quad (3.18)$$

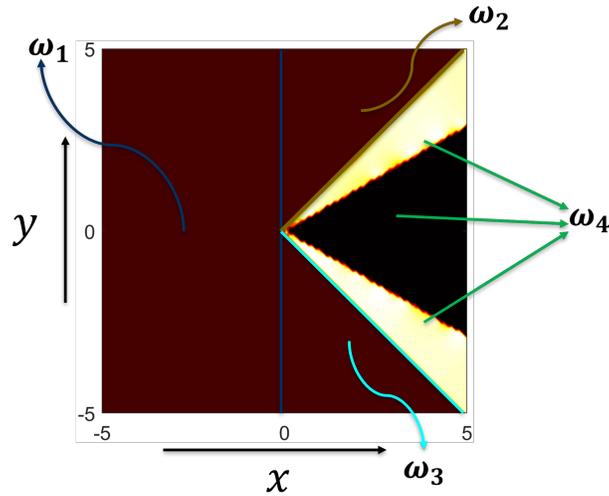


Figure 3.1: Pressure Distribution of the flow field

As the dynamics of the problem are already linear, we don't have to go to a higher dimension to linearise our system. Hence, the pressure measurements at each snapshot alone span our observable space, and the representation of the application of the Koopman operator on a current state to construct the dynamics at a future state is an exact representation. After decomposing the data matrix using SVD, we get the following spatial and temporal modes as shown in 3.2.

From Fig 3.2, we can say that the spatial modes rightly represent the Gaussian distribution in space and we can also clearly see the distinct maximum values of these Gaussian distributions which characterizes the exact solution of the amplitudes of the modes in this test case. As for the time dynamics is concerned, we can clearly see four distinct temporal modes although a couple of them look like they are intertwined with each other, indicating incomplete distinction of the temporal modes. These modes can be disjointed using the DMD method to obtain a clear distinction of temporal modes and also obtain their spatial distributions.

Upon performing DMD we can find the frequencies of each distinct mode by taking the logarithm of the eigenvalues divided by the time duration between the successive snapshots. As discussed in the previous section, the eigenvalues obtained from DMD are the Euler form

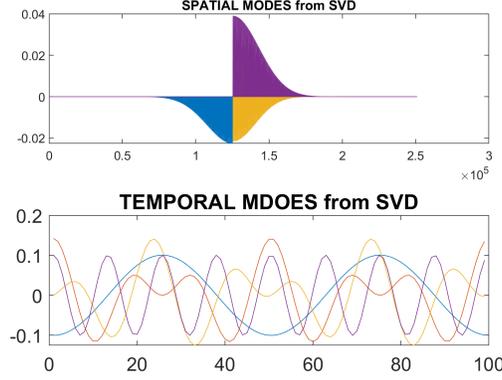


Figure 3.2: Spatial Modes (left singular vectors) and Temporal Modes (right singular vectors) obtained from SVD are plotted. Each colour represents a distinct vector. The x-axis of the top plot denotes the spatial coordinates, while the x-axis of the bottom plot corresponds to the time instant or time coordinate. The y-axis of both plots corresponds to the mode value at the respective space and time coordinates respectively.

of the frequency of our temporal modes which is described by 3.19.

$$\lambda_i = e^{i\omega_j dt} \quad (3.19)$$

$$\text{hence } \omega_j = \frac{\ln \lambda_j}{i\delta t}. \quad (3.20)$$

The frequencies that we obtained from performing DMD on the data snapshots are given in Eq 3.21 :

$$\omega_1 = 6.28 \quad \omega_2 = 12.56 \quad \omega_3 = 18.84 \quad \text{and} \quad \omega_4 = 25.13 \quad (3.21)$$

which rightly matches with the exact frequencies we used to construct the snapshots given in Fig 3.16. Moreover, the corresponding eigenvectors which represent the spatial distribution of these temporal modes are shown in Fig 3.3.

From the above visual representation of the obtained DMD modes, we can say that the spatial distribution of the temporal modes are perfectly matching with our initial construction.

3.2.2 DMD Employed on Nonlinear Data

For the next test case, we are constructing the data set for a non-linear dynamical system as given in Eq 3.22:

$$\dot{y} = \sqrt{1 - y^2} \quad (3.22)$$

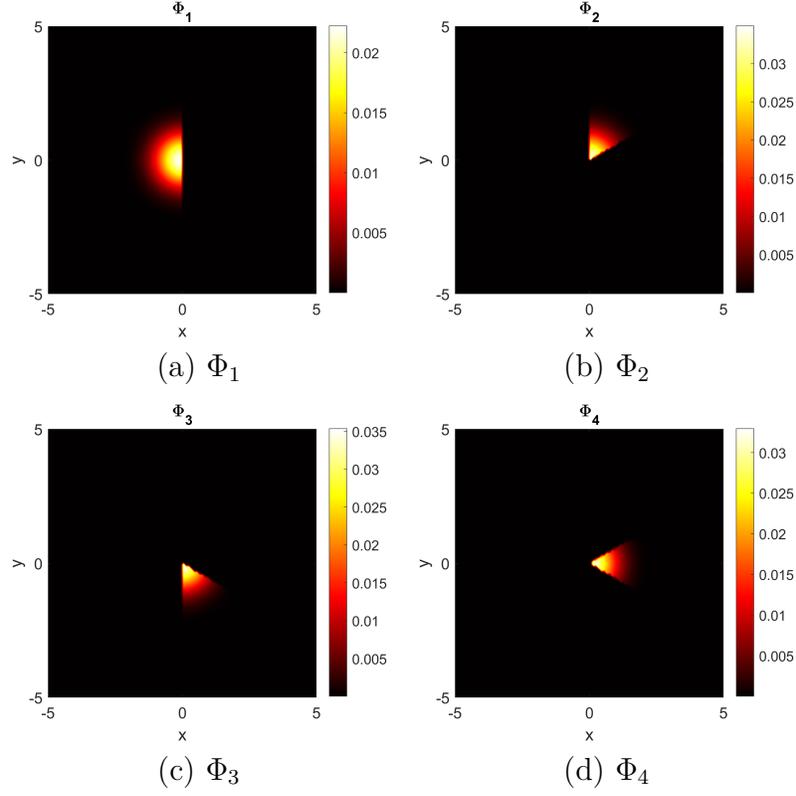


Figure 3.3: Spatial Modes obtained from DMD

The analytical solution for the above ODE is well known to be a sinusoidal function hence, we have constructed our snapshots at regular intervals of time using a sinusoidal forcing as given in Eq 3.23.

$$P_{unsteady}(x, y, t) = P_{steady}(x, y) + \epsilon_j e^{(-x^2 - y^2)} \sin \omega_j t \quad (3.23)$$

The reason for choosing the sinusoidal variation is to test the application of DMD on a flow or system that is nonlinear. As we know the system is nonlinear in this case, according to Koopman's theory, we have to increase our dimensions to linearise the system. To demonstrate the nonlinear nature of the data, we have performed the same analysis as the last test case wherein we use the pressure measurements alone as our observables and frequencies obtained from DMD are given in Eq. 3.24, which does not match our original forcing frequencies given in Eq. 3.16.

$$\omega_1 = 0.3998 \quad \omega_2 = 1.6125 \quad \omega_3 = 3.6798 \quad \text{and} \quad \omega_4 = 6.6749 \quad (3.24)$$

This is an indication that the way we have set our observables hasn't rendered our nonlinear system to a linear setup adequately, hence we have to add more appropriate observables to make it linear in a higher dimension.

To illustrate the idea of applying Koopman theory on the given nonlinear dynamical system in Eq 3.22, let us try to linearise the system by choosing the appropriate observables. This is a non-linear ODE in one dimension (only one variable to define the state) and the solution for this equation is in sinusoidal form. To linearise this ODE use y and \dot{y} as two observables:

$$y_1 = y, \quad y_2 = \dot{y} \quad (3.25)$$

By algebraic manipulation, we can write the new evolution law in higher dimension (y_1, y_2) as:

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \quad (3.26)$$

So by including the time derivatives of the dependent variable, we are able to linearise the system in a higher dimension which suggests us to use the time derivatives of our unsteady field in the corresponding problem to extract the DMD modes. We have used a forward difference scheme to calculate the time derivative of the snapshots and have increased our observable space to both the pressure and the time derivatives of pressure. After performing DMD on this new observable space we obtained the following complex conjugate frequencies ($\ln \frac{\lambda_i}{\delta t}$), whose magnitudes exactly match with our exact frequencies given in 3.16.

$$i\omega_1 = -i6.28 \quad i\omega_2 = +i6.28$$

$$i\omega_3 = -i12.56 \quad i\omega_4 = +i12.56$$

$$i\omega_5 = -i18.84 \quad i\omega_6 = +i18.84$$

$$i\omega_7 = -i25.13 \quad i\omega_8 = +i25.13$$

To fortify the importance of the choice of observables, from this simple demonstration, we can say that the convergence towards the Koopman operator relies on judicious choice of the observables; incomplete or poor choice of observables may yield inaccurate results. Although we don't explore this angle of analysis in this thesis, it is going to be very crucial to find

the right observable spaces specific to the problem in hand going forward to be able to use DMD effectively.

Chapter 4

Unsteady Mesh Adaptation

4.1 Outline

We have previously discussed that achieving a uniform mesh greatly reduces interpolation errors when provided with a prescribed metric field, proving to be a powerful tool for steady flows. Therefore, for any given metric field, we can generate a unit mesh using full-adaptation or approach a unit mesh using an r-adaptation approach. Consequently, for an unsteady simulation, we need to predict the flow at a future instant to build a metric tailored to that flow instant, enabling us to deform the mesh accordingly before evaluating the solution using our numerical techniques. For example, in unsteady high-speed flow problems, accurately capturing shocks is of utmost importance. This requires us to use shock-fitted meshes where the refined meshes align with the shocks to minimize numerical dissipation and improve convergence, as flow solutions with shocks often exhibit numerical oscillations. In certain cases, like a transonic buffet or the shock structures behind a cylinder in hypersonic flow, we observed shock oscillations which required us to find a way to deform the refined regions during the simulations to align with the moving shocks. This is where DMD comes in handy to predict the position of such unsteady features and helps us to form metric tensors catering to such dynamic flow fields, which in turn enables us to adapt the mesh before the actual CFD time iteration. Hence, our first priority is to accurately capture these unsteady features using DMD and to be able to predict the flow in the near future with enough accuracy to monitor these dynamic flow features. As discussed in the previous chapter, even though we can predict the flowfield to some extent using DMD, it is not feasible to use mesh adaptation at each time instant. Besides, the full adaptation is not very useful in the context of generating dynamic meshes. Hence, our approach combines all these steps performed in sequence to optimise our computational effort while not diminishing the simulation accuracy. The basic structure of the algorithm can be seen in the algorithm given in [1](#).

Algorithm 1 Unsteady Mesh Adaptation Algorithm

Assume n snapshots between two consecutive full adaptation cycles.

Main Function: $\text{UnsteadyMeshAdaptation}(\mathcal{H}_{j-1}^{i=1,2,\dots,n}, S_{j-1}^{i=1,2,\dots,n}, n)$

Inputs: n meshes \mathcal{H}_i and n solutions S_i , where $i = 1, 2, \dots, n$

Output: Adapted meshes $\mathcal{H}_j^{i=1,2,\dots,n}$.

Step1:- Interpolation for Snapshot matrix generation:

1. Generate a base mesh \mathcal{H}_0 from the latest available solution.
2. Interpolate previous n solutions on the r-adapted grids onto the base mesh for consistency of spatial locations in DMD snapshot matrix.

Step2:- DMD Module:

1. Perform Dynamic Mode Decomposition (DMD) on the snapshot matrix and predict m snapshots in future for r-adaptation purposes, and the time difference between each snapshot depending on some fundamental frequency of the flow.

Step3:- r-adaptation:

1. Generate the metric fields using the generated feature fields and utilize `r_adapt` to generate “m” r-adapted grids.
2. Perform polynomial fitting in time for the location of each node for the above generated grids to be able to interpolate their trajectories during other time-steps between them.
3. Generate the meshes for the following p time steps before the next full-adaptation cycle. ($p > n$, assuming snapshots are generated every for every q time iterations $n \times q = p$)

Step4:- Time-advancement:

1. Evaluate the unsteady CFD solution on the adapted grids.
 2. Perform an error analysis between the computed CFD solutions and the DMD predictions to evaluate the accuracy of DMD, based on a tolerance value either decrease or increase the time interval between successive snapshot generations.
-

To give a brief outline of the algorithm, we perform Dynamic Mode Decomposition on a set of unsteady CFD solutions to extract the dynamics of the flow in question to predict the flow features at a future time instant. This enables us to adapt the meshes accordingly using the combination of the r-adaptation and the full-adaptation in a sequential manner. To understand the idea, consider an unsteady CFD simulation at k^{th} time iteration. To best represent the solution, next consider performing a full adaptation at the current time iteration. Our objective now is to generate a dynamic mesh that optimises the nodal positions before the next full-adaptation step to best track the unsteady features in the intermediate time iterations. To prescribe the motion of the nodes we need to predict the flow beforehand. Hence, we generate a data matrix like in Ch 3 using the previous N flow snapshots equally spaced in time with adequate sampling of snapshots and the time duration of sampling dependent on the convective scales of the problem to obtain an optimal DMD prediction of flow fields. Assuming we predict the flow features accurately for a future time-instant, to prescribe the velocity to each node we need a minimum of one r-adapted mesh at the end of the cycle just before the next full-adaptation time iteration to compute a finite difference in time. But keep in mind that this approach approximates the motion of the nodes in a linear fashion which might be adequate to certain simulations with rather simple unsteadiness, but when in need of a more detailed prescription of the movement we can use multiple r-adapted meshes during the cycle to get a better representation, and can trace the trajectory of the nodes using higher order polynomial fitting, hence in a sense improving the order of tracking these unsteady features.

4.2 DMD on moving meshes

One of the challenges to performing DMD in this approach is that the formation of a data matrix for DMD assumes that we are recording the time variation of an observable at a fixed location, but as discussed above we perform r-adaptation which results in a change of position of the nodes hence the formation of the data matrix from the nodal values is not straightforward. Hence we are opting to interpolate the feature field onto a static grid to obtain temporal data which is consistent in space. In our algorithm, we have decided to interpolate the previous snapshots onto the latest fully adapted grid. Hence, the accuracy of this interpolation step is also crucial for the satisfactory performance of the algorithm because any one of such interpolation errors introduced in the construction of the data

matrix can alter the dynamics captured by DMD if the interpolation errors are significant in magnitude.

4.3 Algorithm Implementation

A brief discussion about the actual implementation of this technique for an actual simulation case would be helpful since there are multiple moving parts in this approach to mesh adaptation. Firstly, we have already created a simple algorithm that does all the tasks described previously in this chapter automatically. Efforts are being currently directed towards integrating this module into the FUN3D software of NASA Langley to perform this operation in real-time along with the actual CFD simulations.

To start with some key specifics about the algorithm, the anisotropic mesh adaptation is handled by the code *refine* of FUN3D [31] which becomes very helpful with *refine*'s rich suite of operations with meshes ranging from the computation of the metric tensor from the prescribed feature field to the complete suite of adaptation mechanics that is necessary to modify the mesh with respect to the metric field and also accurately interpolates solutions between meshes, each used extensively in our approach. Additionally, *refine* supports parallel execution which is a huge benefit. We have added an r-adaptation module in *refine* using the smoothing function already available inside the software that also assigns the same forces as discussed in 2. The DMD module that performs this spatio-temporal decomposition has been executed in C++ and to perform the matrix operations and eigen-decomposition on large data matrices we have used the state-of-the-art LAPACK [32] and LBLAS [33] libraries which are highly optimised for efficiency and numerical accuracy. Finally, we have written a module to perform the interpolations necessary for DMD on adaptive meshes as discussed in 4.2 to form the data matrix in an automated fashion.

Since we are proposing repeated application of this approach, as the simulation proceeds forward in time, we have to look at what might be the most computationally extensive parts of the algorithm. The two main aspects that comprise (a) the computational load are the computation of Singular Value Decomposition (SVD) and (b) the full-adaptation and r-adaptation steps. The adaptation functions are already parallelised, thanks to *refine* [34, 35]. To compute the continuous flow of data matrices we are proposing to use an on-the-fly DMD approach, wherein we perform a streaming SVD as suggested in [36].

Chapter 5

Results and Discussion

5.1 CFD SOLVER

The snapshots utilized in this study were generated using the FUN3D CFD solver, developed by NASA. FUN3D is an advanced, fully unstructured Navier-Stokes solver renowned for its robustness in simulating complex aerodynamic flows across a wide range of regimes, from subsonic to hypersonic. FUN3D employs a finite-volume method for spatial discretization, ensuring second-order accuracy in both spatial and temporal dimensions. Additionally, FUN3D incorporates advanced mesh adaptation and refinement techniques, allowing for enhanced accuracy without a prohibitive increase in computational cost. For time integration, FUN3D provides both implicit and explicit schemes, facilitating efficient simulations of large-scale, time-dependent problems. This dual approach ensures the generation of high-fidelity snapshots that capture the temporal dynamics of fluid flows accurately.

5.2 DMD reconstruction

5.2.1 Cylinder in a Subsonic flow

$$St = 0.212\left(1 - \frac{21.2}{Re_D}\right) \quad (5.1)$$

for Re_D between 15 and 150 [37]. With known Strouhal number, the shedding frequency, f_s , can be calculated as follows:

$$f_s = \frac{StU_\infty}{Re_D} \quad (5.2)$$

where U_∞ and D are the freestream velocity and cylinder diameter, respectively. For this case, the Strouhal number was calculated to be 0.167 which gives a shedding frequency of 1.25 Hz. Figure 5.1 shows a sample snapshot of the flowfield at a particular instant in time

where the “vortex street” can clearly be seen.

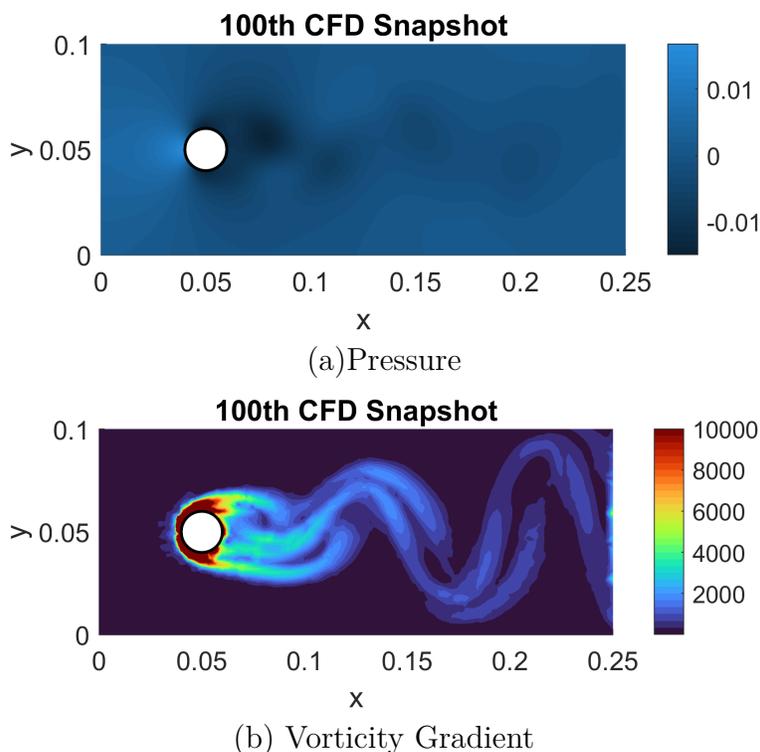


Figure 5.1: Subsonic flow around cylinder

To perform DMD analysis on this test case, we have generated 309 snapshots at a regular interval of $0.0333s$, with each snapshot containing the pressure, velocity, and vorticity gradient at each grid point. We have initially used pressure as our observable since we have observed very periodic variation, but the downside is that the pressure is very uniform across the space since it is an incompressible flow hence, we used vorticity gradient as the observable for this test case, which is a very indicative metric of the wake as seen in 5.1(b).

We have used 250 snapshots per Data Matrix hence we would get a total of 249 distinct DMD modes. After performing triple decomposition of this data matrix using DMD, we have obtained the following distribution of amplitudes and the eigenvalues as shown in Fig. 5.2.

Also one can see using 200 Snapshots, in Fig 5.3(a), we have obtained exact reconstruction for the 100^{th} snapshot, which was already included in the training set. Also, one can see that the error for predicting the feature field in Fig 5.3(b), which is at the time instant of 100 times the time difference in the future and is about 1 percent. Hence, in this case, we can say

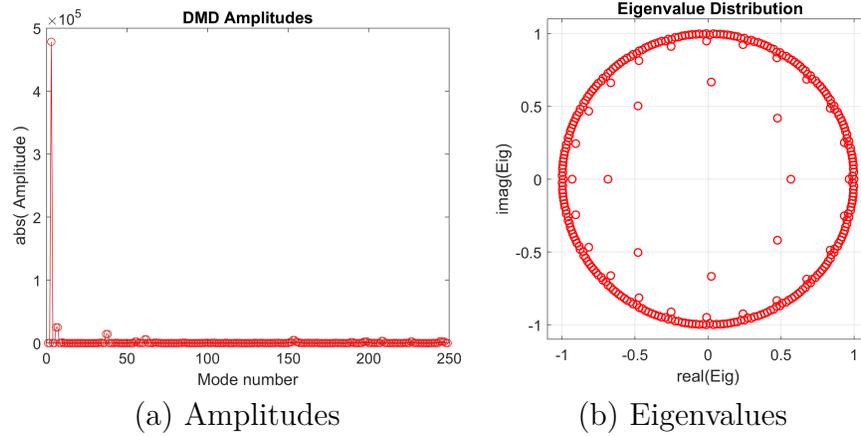


Figure 5.2: Amplitudes distribution and eigenvalues plot

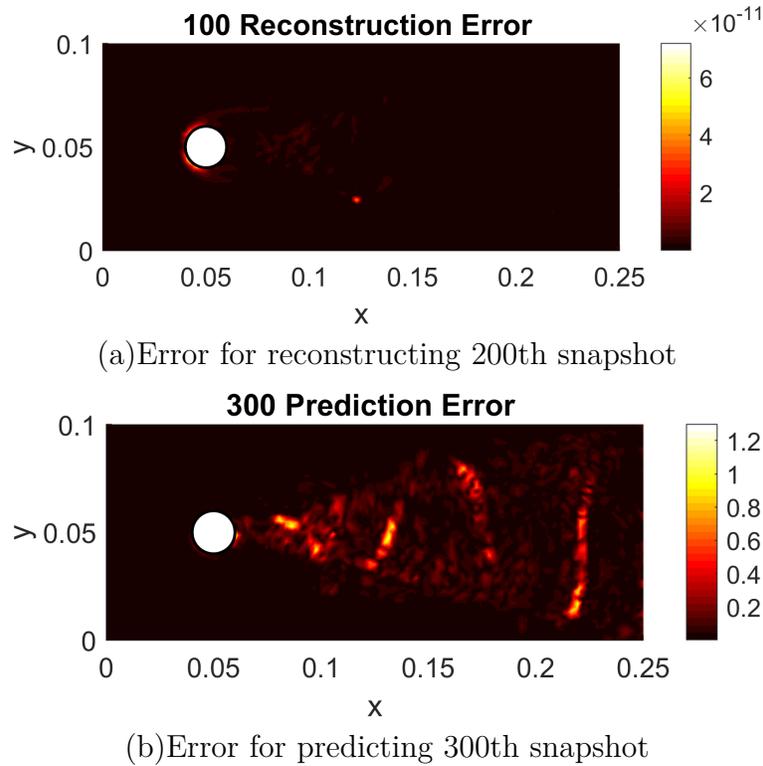


Figure 5.3: Reconstruction and Prediction Error(%)

that we were able to resolve the features of the flow into linear modes really well. We have also experimented with reconstructing the flowfield with the dominant modes which come out to be around 20-30 modes that have significantly greater amplitudes than the others and we have achieved a prediction of 300th snapshot with less than 15%.

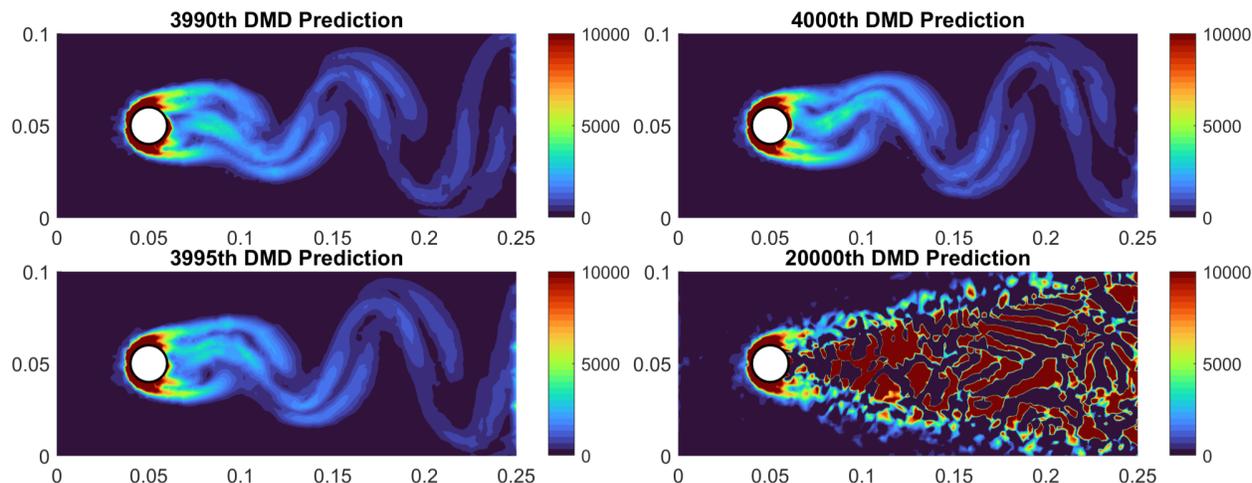


Figure 5.4: 3990-20000 predictions using DMD on 300 snapshots

Also, we have shown a few feature field predictions in Fig. 5.4 predicted using DMD on 300 snapshots, where we can see some periodic activity predicted even at a time-instant more than 10 times the time interval of the data set, and finally the modes decay a lot which is expected. Hence the question arises as to how many snapshots are necessary to obtain this sort of linear representation to be able to predict the flowfield in future time instances consistently, and the answer for this particular problem is only one shedding cycle is necessary and we have seen great reconstructions with just 25 snapshots. The only difference in using 25 snapshots is capturing intricate details of small-scale structures, but the overall shedding of vortices, which is our key target in this problem is captured remarkably well with exploring DMD on one shedding cycle alone as can be seen in Fig 5.5.

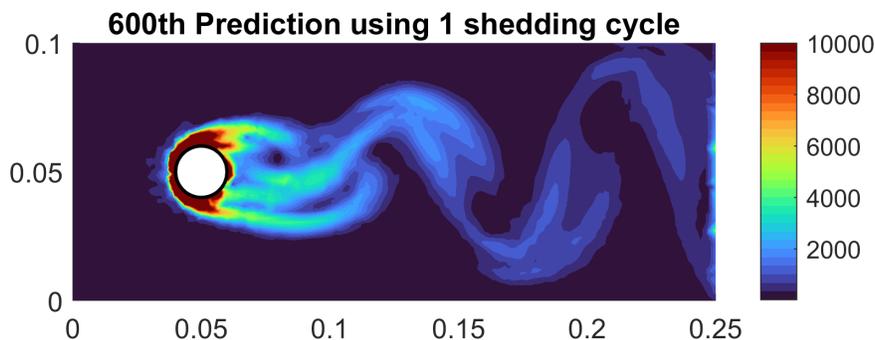


Figure 5.5: 600th prediction using DMD on one shedding cycle

5.2.2 Cylinder in a Supersonic flow

The second test case is a cylinder in a highly supersonic or weakly hypersonic flow of $Ma = 5.0$, where you can see very complex unsteady patterns in the wake of the cylinder as displayed in Fig 5.6. The fluid is modelled as an ideal gas with freestream values of pressure as 1 atm, and Temperature 300K and the time step of the simulation is 6×10^{-5} s. The convective time scale of this flow can be determined by measuring the time it takes for the flow to traverse the cylinder, which is approximately five times the duration between two consecutive snapshots. This observation underscores the chaotic behaviour of the flow. As you can see in Fig. 5.6 I have shown the snapshots of local Mach number, non-dimensional density, temperature and entropy, each of them detects distinct features of the flow like the density field captures the shocks pretty well and remains relatively constant in the wake behind the cylinder, but by using Temperature or Mach number, we can see these complex vortex-shock interactions that characterize the wake in a much clearer manner and the entropy captures the moving vortices since by Crocco's theorem we know the inherent relation between vorticity and entropy. The point being made is that we need to carefully choose what property of the field to use for adaptation depending upon what we want to capture. In this thesis, we have decided to use the Mach number as our observable to test how well the DMD expansion captures these vortex interactions and we will also show some reconstructions of the coefficient of pressure in the following section.

Unlike the first test case where there is relatively periodic vortex shedding over a wide period of time, we can clearly see that there are dynamic interactions between vortices and the shocks behind the cylinder which is evolving over time and completely changing its nature in very short periods of time as shown in Fig 5.7. We can clearly see that there are large amplitude oscillations in the 100th snapshot, which sort of calms down in the 250th snapshot but again there is this breakdown of the vortices and they start to become more dynamic during 300-320 snapshots which indicates the large variability of flowfield over small time durations proving that the time resolution of the data is very low.

Even though a flow might be nonlinear in the way the state variables are coupled theoretically, DMD will be able to linearize the nonlinearities given the flow is periodic in some sense. But as explained in [9] “*DMD does not typically work well for systems with highly intermittent dynamics*” which is the case in this example. The convective time scale is about the duration of the time between 3-4 snapshots approximately, making it difficult for long-term accurate

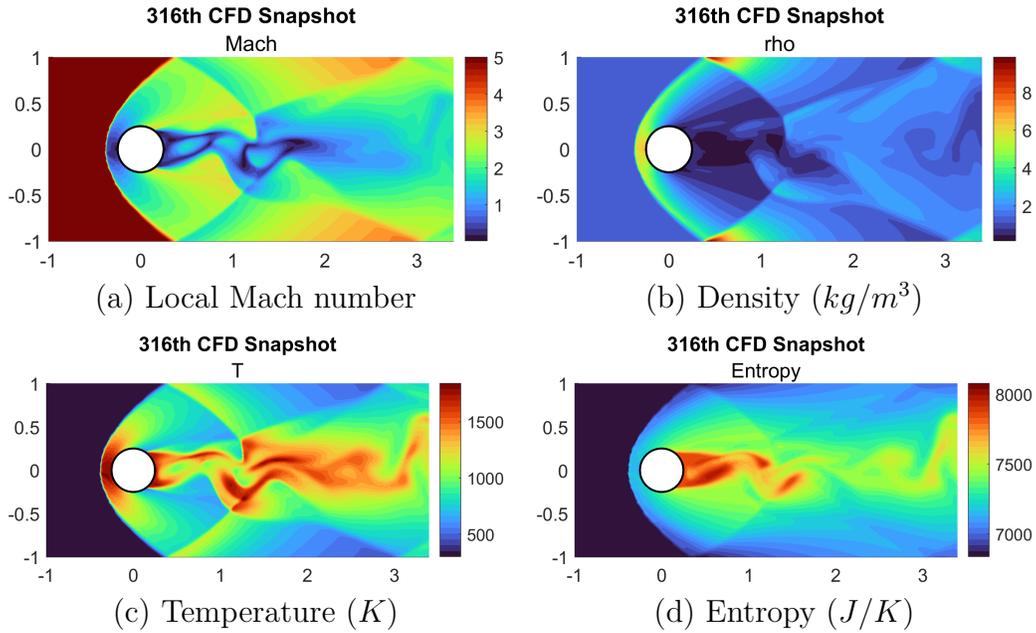


Figure 5.6: Feature fields at an arbitrary time instant

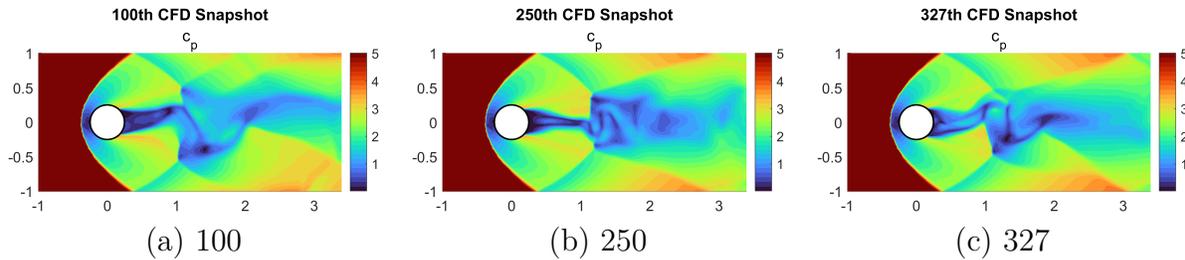


Figure 5.7: Time evolution of flow-field

DMD predictions.

Using the local Mach number as our observable, we obtain the modes with the amplitudes and eigenvalues as shown in Fig. 5.8. As evident from the figure, the DMD gives us some high amplitude modes at the end of the spectrum that are inside the unit circle, which is often what DMD does when some intermittent dynamics is present that is very local to a very short period of time. The DMD tries to fit each unsteady feature hence it includes a high amplitude decaying term for those fast-changing unsteady features. Due to this property of DMD, there is a discrepancy in the way the energy is generally distributed among the DMD modes, which affects the DMD future predictions.

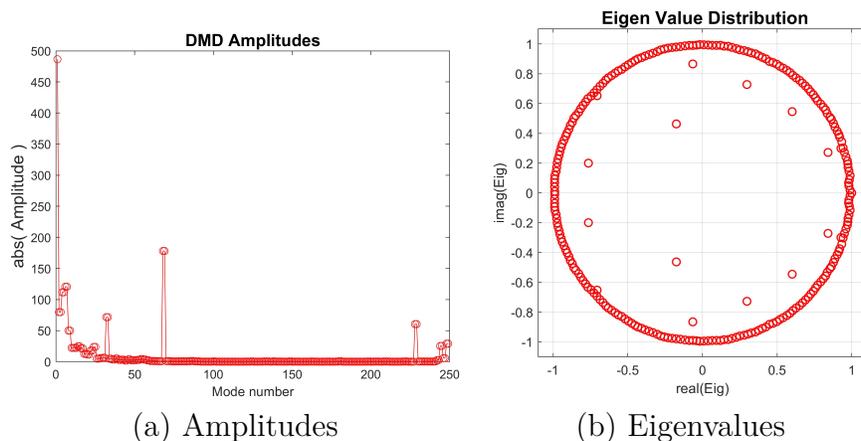


Figure 5.8: Amplitudes distribution and eigenvalues plot

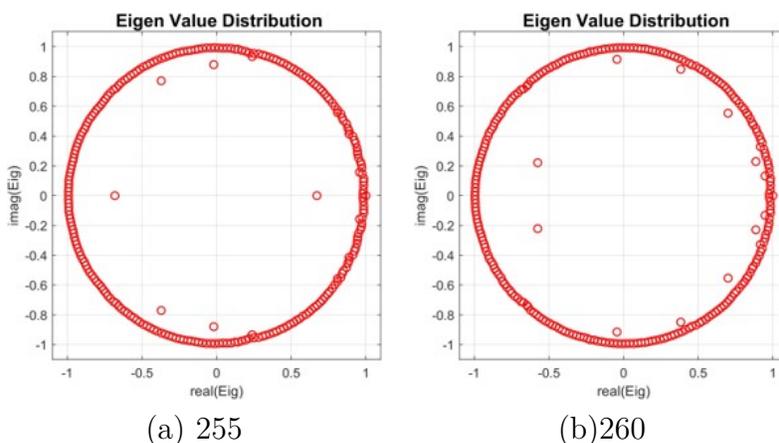


Figure 5.9: Eigenvalue distribution from DMD using 255 and 260 Snapshots

Similarly, more of such intermittent modes are seen when decomposing 255 snapshot data set and 260 snapshot data set, each decomposition showing some new decaying mode that is introduced locally by the additional snapshots as shown in 5.9. To analyze the reason for these decaying modes, we have utilized Multi-resolution Dynamic Mode Decomposition (mrDMD) as introduced in [38], wherein we divide the data into multiple temporal scales or levels. This decomposition allows us to capture and isolate dynamic behaviours at different resolutions, providing a more detailed understanding of the temporal evolution and decay characteristics of the modes present in the system. Similarly, we have also performed mrDMD on the subsonic test case. In Fig. 5.10, we compare the decompositions of both the subsonic and supersonic cases. For the subsonic case, the majority of the temporal content is

concentrated around a frequency of 1.25 Hz, and the temporal variation remains consistent across the bins at that level. In contrast, the supersonic case exhibits significant variation within the individual bins, highlighting the level of intermittency in the flowfield. This comparison reveals the distinct dynamic behaviours between the subsonic and supersonic regimes.

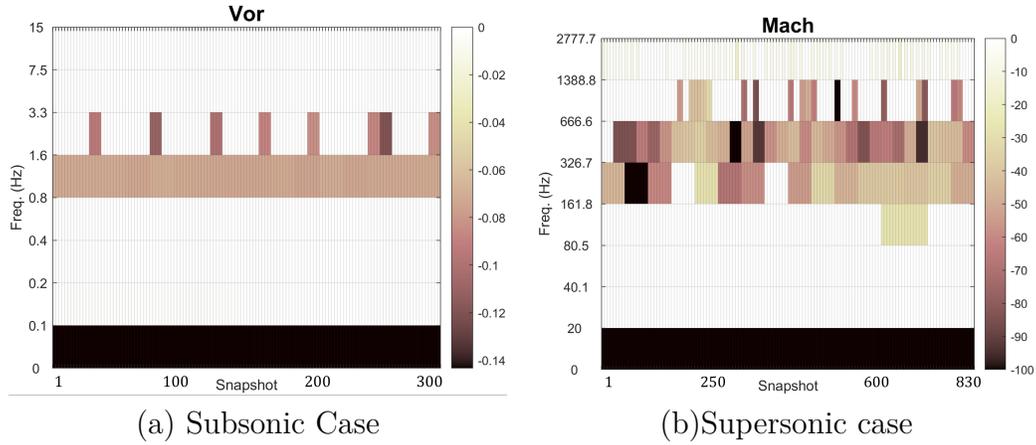


Figure 5.10: multi-resolution map of both test cases

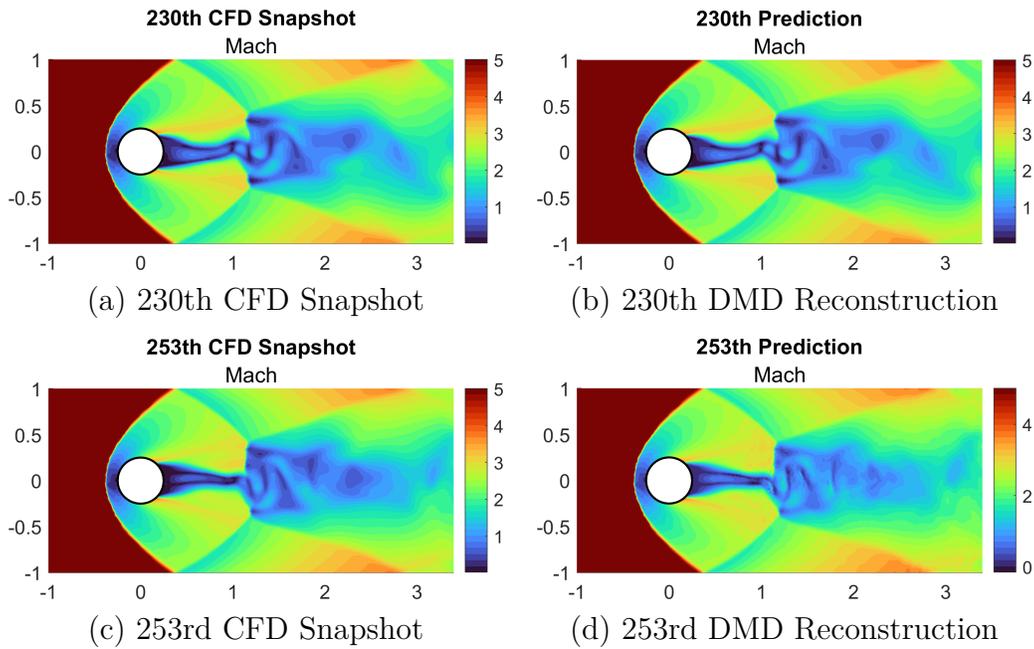


Figure 5.11: DMD reconstruction using 250 Snapshots

We have performed DMD analysis on the first 250 data snapshots, and the true solutions

and the corresponding reconstruction and predictions obtained from the DMD are shown in 5.11. We have obtained an exact reconstruction at the 230th snapshot, and the maximum absolute relative error for the 253rd snapshot is about 29.3% which might be due to little displacement of the shock related to the true snapshot data which has a significant jump in Mach number across it. Although, the rms error is only about 1.3% which says that in aggregate there are only a few regions where the reconstruction accuracy has been hampered but in an overall sense the reconstruction holds close to the actual snapshot. The method currently used by NASA to simulate unsteady flow fields with adaptive mesh refinement takes an averaged flowfield over a wide range of timescales and performs the grid adaptation on the mean flowfield; hence, we compared how our DMD prediction holds up against the mean flow field as shown in 5.12, and the RMS error when the Mean flow is compared to the actual snapshot came out to be 3.97%. Although our DMD reconstruction performs well over very short-term predictions, it doesn't evolve the flowfield well for a longer time because of the imperfect reconstruction due to the temporal resolution of the snapshots.

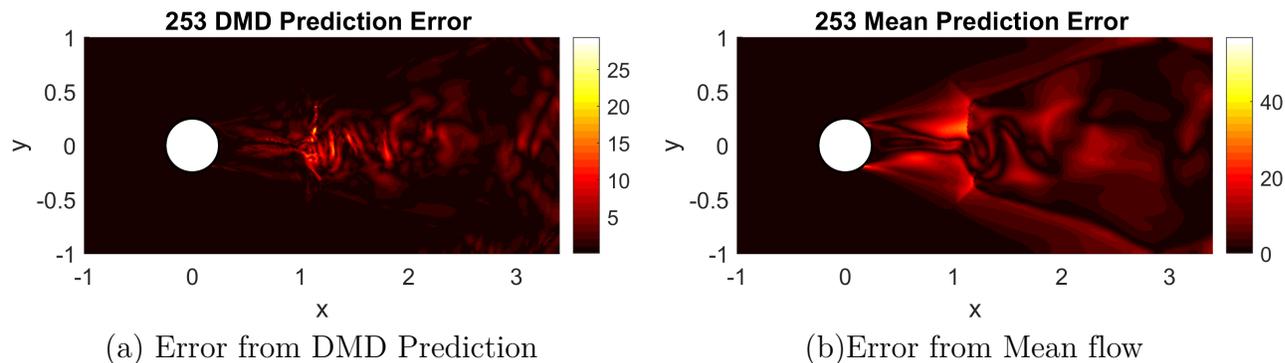


Figure 5.12: Error w.r.t true snapshot using DMD prediction and Mean flow

Since we have observed several of these spurious modes changing rapidly overtime we tried to perform a Fourier transform in time at each spatial coordinate to filter out the high-frequency components using a threshold frequency to be defined by the user, and the results are given in Table 5.1.

We can clearly see an improvement in predictions using this filtered fictitious field, we also see that the predictions moreover fail immediately after that, due to the fact that, by removing the high-frequency content, we not only remove the local contents but flow features that vary with the same frequency but are global in nature. Also, the whole flowfield is filtered which makes it difficult to extend to a realistic scenario and the threshold value is not known a

Table 5.1: Reconstructions using DMD on actual flowfield and FFT filtered flow

Predicted Snapshot	rms Error(%)	max Error(%)	Error with FFT flowfield(%) ($\omega_{threshold} = \frac{0.25}{Timestep}$)
251	0.58	12.68	1.27
253	1.39	29.30	2
255	1.95	29.35	1.75
260	2.70	40.35	52.73
265	3.17	44.12	303.97

priori. To resolve all the local features and the global background flow, we would adopt the Multiresolution DMD [38] approach which is the MRA combined with DMD that separates the slow modes from fast modes from the windowing approach.

5.3 Grid Adaptation using DMD

The mesh used for the subsonic cylinder case is shown in Fig 5.13 which is an isotropic mesh.

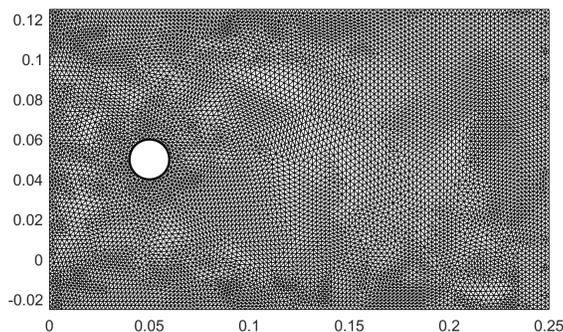


Figure 5.13: Input Mesh of Subsonic Case

The meshes obtained from performing a full adaptation on an actual solution and a DMD predicted feature field of a 100th snapshot can be seen in 5.14.

To compare the meshes in Fig. 5.14, one can divide the whole mesh into discrete regions and compare the element distributions or density in each of these regions. The region division is shown in Fig. 5.15, for a demonstration we have divided the mesh into 10 sub-regions and the element density is shown in 5.2.

Qualitatively looking at the grids generated by the actual flowfield and the DMD prediction

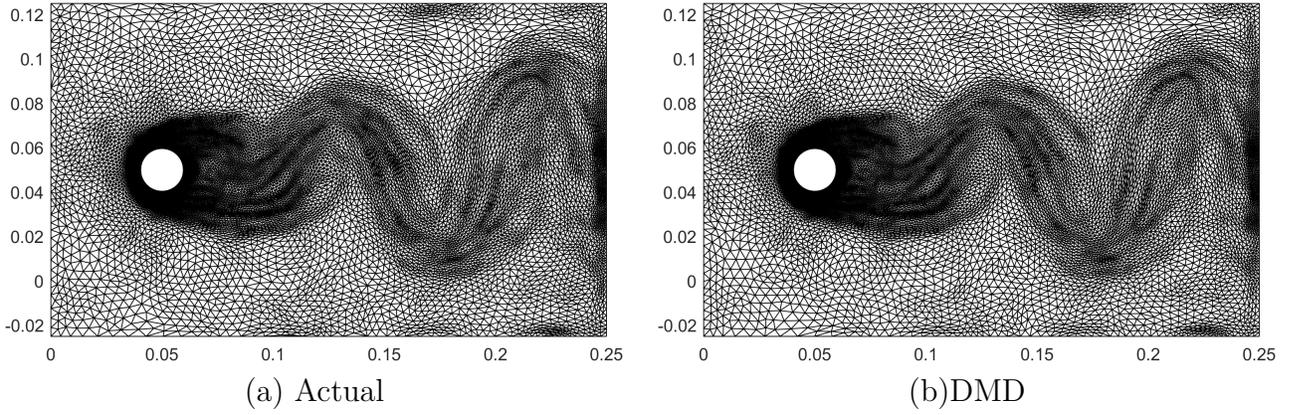


Figure 5.14: Grids adapted with actual solution and DMD feature field

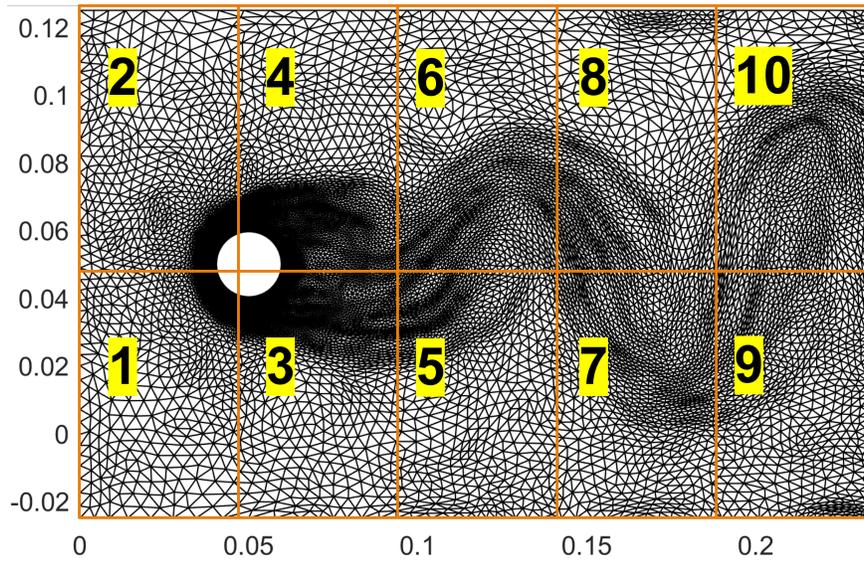


Figure 5.15: Region Division

in Fig. 5.14, we can say that the grids look nearly identical, which can be seen in Table 5.2.

Table 5.2: Element Distribution

Region	Initial Mesh	Adapted using actual Sol	Adapted using DMD Prediction
1	1007	2070	2050
2	1021	2007	2031
3	955	1781	1803
4	935	1636	1682
5	871	774	779
6	822	961	975
7	868	1013	991
8	811	591	612
9	959	987	972
10	953	1159	1164

Similarly for the supersonic case the input mesh can be seen in [5.16](#).

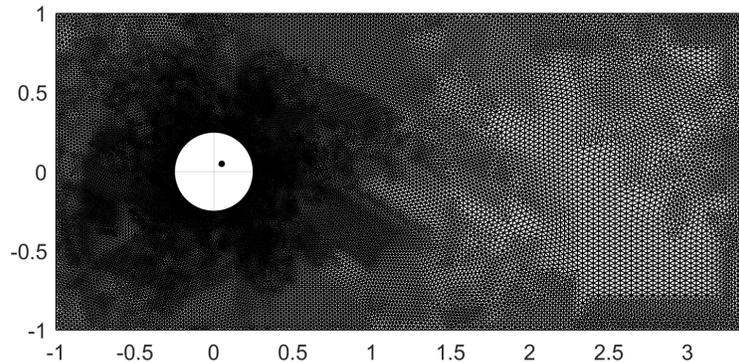
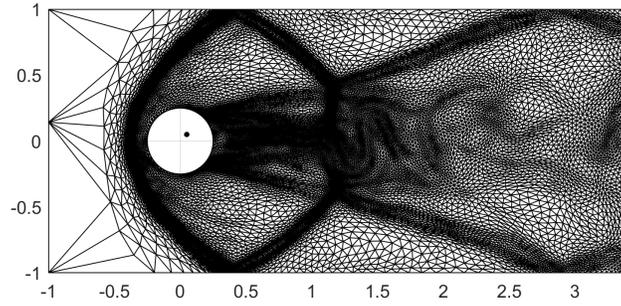
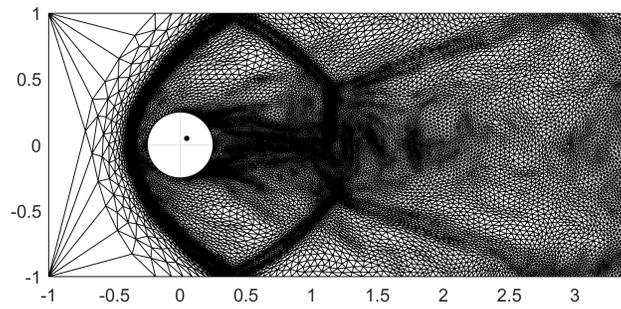


Figure 5.16: Input Mesh of Supersonic Case

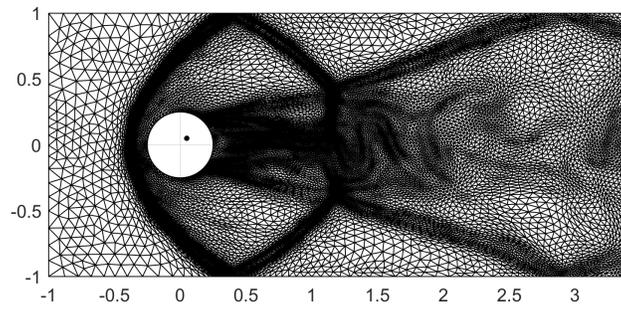
For the supersonic case as shown in [5.17](#), we have constructed grids based on the (a) actual 255th snapshot, (b) 255th prediction obtained from DMD on 250 snapshots, (c) prediction using DMD performed on FFT filtered flowfield and (d) the mean flow of the data set. From the figures, we can clearly see that the grid obtained from FFT-filtered data is very close to the actual grid. Whereas the grid obtained from a direct DMD-generated snapshot does capture a good amount of unsteady features such as the wake and the initial oblique shock the reflected shock seems to be a little less developed than the other two grids. But using the mean flow seems to be the most detrimental as the oblique shock in the wake is very diffused due to the averaging of the oscillatory data as well and the wake is also not captured at all. Similarly, the Mach stem in the wake is decently captured in the DMD-generated



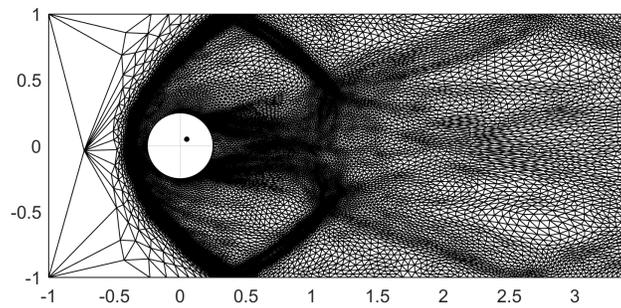
(a) Actual flowfield



(b) DMD



(c) DMD on FFT filtered flow



(d) Mean flow

Figure 5.17: Grids adapted for Supersonic case

Table 5.3: Element distribution for supersonic case

Region	Initial Mesh	Actual grid	DMD generated	FFT generated	Mean flow
1	3536	1108	915	1200	1604
2	3294	1033	867	1102	1461
3	3414	1838	1653	1812	2226
4	3494	1970	1653	1924	2306
5	2054	1773	1883	1776	1679
6	2197	1957	2039	1935	1658
7	1314	901	1046	912	602
8	1357	1039	1034	1027	599
9	1294	857	1159	845	501
10	1453	825	1089	814	539

grid whereas the Mean flow doesn't do a good job. Similar trends can be seen in Table 5.3, where the grids from the filtered solution match pretty well with the actual grid other than in the first two regions which was generated due to precision errors obtained from the FFT which created a minute numerical difference.

Chapter 6

Conclusions and Future Work

Conclusions

Anisotropic grid adaptation stands as a pivotal tool for accurately modelling high-speed flows in steady-state conditions. However, its seamless extension to the domain of unsteady flows, which holds immense scientific interest, has proven to be a formidable challenge. Despite numerous proposed methodologies aiming to capture the evolving dynamics of unsteady flow fields, none have thus far achieved computational tractability for practical applications. This underscores the pressing need for innovative approaches that balance accuracy with computational efficiency in addressing this critical demand. To that end this study leverages a data-driven technique namely Dynamic Mode Decomposition (DMD), to model these unsteady flows in conjunction with anisotropic grid adaptation. DMD has proven to be an effective tool for extracting coherent structures and has shown its ability to make short-time predictions. We have studied two different unsteady flow regimes, one being a cylinder in a subsonic flow that exhibits periodic vortex shedding and another a cylinder in a supersonic flow with complex vortex-shock interactions. In the subsonic case, our study demonstrates DMD's effectiveness in accurately forecasting the flowfield over extended durations. While in the supersonic case, we were not able to make long-term predictions we could, however, successfully make acceptable short-term predictions which seems promising. The DMD has shown a promise to aid in grid adaptation to capture the essential unsteady features even though the data set lacked good temporal resolution unlike using a mean flow which diffuses out a lot of unsteadiness of the flow features. One of the main lessons learnt after performing this study is that an accurate DMD prediction is crucial for us to be able to adapt our grids accurately. Moreover, it is advisable to have snapshots with good temporal resolution, without which the flow can transition into different regimes in an intermittent fashion which can prove challenging for DMD to capture all the dynamics accurately. These findings underscore the potential of leveraging DMD in conjunction with anisotropic grid

adaptation to advance the modelling of unsteady flows, paving the way for enhanced accuracy and computational efficiency in practical applications. The method proposed in this thesis surpasses the conventional approach of dealing with unsteady flows, which typically involves using a mean flow. This conventional method tends to smooth out the unsteady characteristics of the flow, thereby missing crucial features. A modular unsteady grid adaptation routine has been developed at Virginia Tech, which has been tested over the discussed cases in the thesis and the efforts are currently directed towards integration of this module with FUN3D in collaboration with M4 Engineering, a small business in Long Beach, CA, and NASA Langley Research Center.

Future Work

This work can easily be extended to more complex test cases to test the applicability of DMD, such as a flow around an airfoil rather than a simple cylinder, transonic buffet over an airfoil, 3D test cases, and also turbulent flows. Extended DMD (EDMD) is a potent tool for modelling complex nonlinear flows. However, its application in grid adaptation requires further exploration due to computational demands. A study is needed to evaluate the balance between accuracy and efficiency. When dealing with highly nonlinear flows, linearizing with base state variables may not suffice, necessitating the search for nonlinear embeddings. This issue is particularly relevant in the broader context of applying Koopman's theory to fluid dynamics. As discussed in the supersonic case, often we can encounter a case with bad temporal resolution because of a lack of knowledge about the flowfield *a priori*, hence resulting in poor construction of DMD modes, which leads to bad predictions. As a result, unresolved temporal simulations can heavily affect the grid adaptation process's ability to capture important features of the flow, which in turn affects the accuracy of the solution. The multi-resolution DMD (mrDMD) approach has demonstrated considerable effectiveness in managing flows characterized by high intermittent phenomena. Therefore, it's logical to extend the investigation in this thesis to include the supersonic case and assess whether leveraging mrDMD can enhance prediction accuracy. One other aspect to note is that the smoothing function used for r-adaptation in this study is too weak to completely depend upon to converge towards a unit mesh. It would be wise to test out more appropriate functions which cluster the nodes more in the required regions without having any concerns about having a unit mesh at the regions which is beyond our interest.

Appendix A

Appendix I

A.1 Proof for linearization in 3.2.2

The proof for the linearization of the nonlinear equation will be discussed in this section. As suggested earlier, let us use the dependent variable as well as its time derivative as our observables to linearize the equation given in 3.22. Let us write these two observables as two other new dimensions y_1 and y_2 as follows:

$$y_1 = y \tag{A.1}$$

$$y_2 = \dot{y} \tag{A.2}$$

Upon differentiating the A.1 w.r.t time and using relation in A.2 we get the following representation in A.4.

$$\dot{y}_1 = \dot{y} \tag{A.3}$$

$$\dot{y}_1 = y_2 \tag{A.4}$$

We can rewrite the 3.22 as given in A.5, which upon differentiating with time we get A.6

$$\dot{y}^2 = 1 - y^2 \tag{A.5}$$

$$2\dot{y}\ddot{y} = -2y\dot{y} \tag{A.6}$$

Differentiating A.2 w.r.t time we get $y_2 = \ddot{y}$, using this result and equations A.1, A.4 in eq A.6 we get the following result in A.8

$$2y_1\dot{y}_2 = -2y_1y_2 \tag{A.7}$$

$$\dot{y}_2 = -y_1 \tag{A.8}$$

Hence, rewriting the two equations A.4 and A.8 in a matrix form, we can write the equation of state as given in 3.26

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

A.2 r-adapted grids for the Subsonic Case

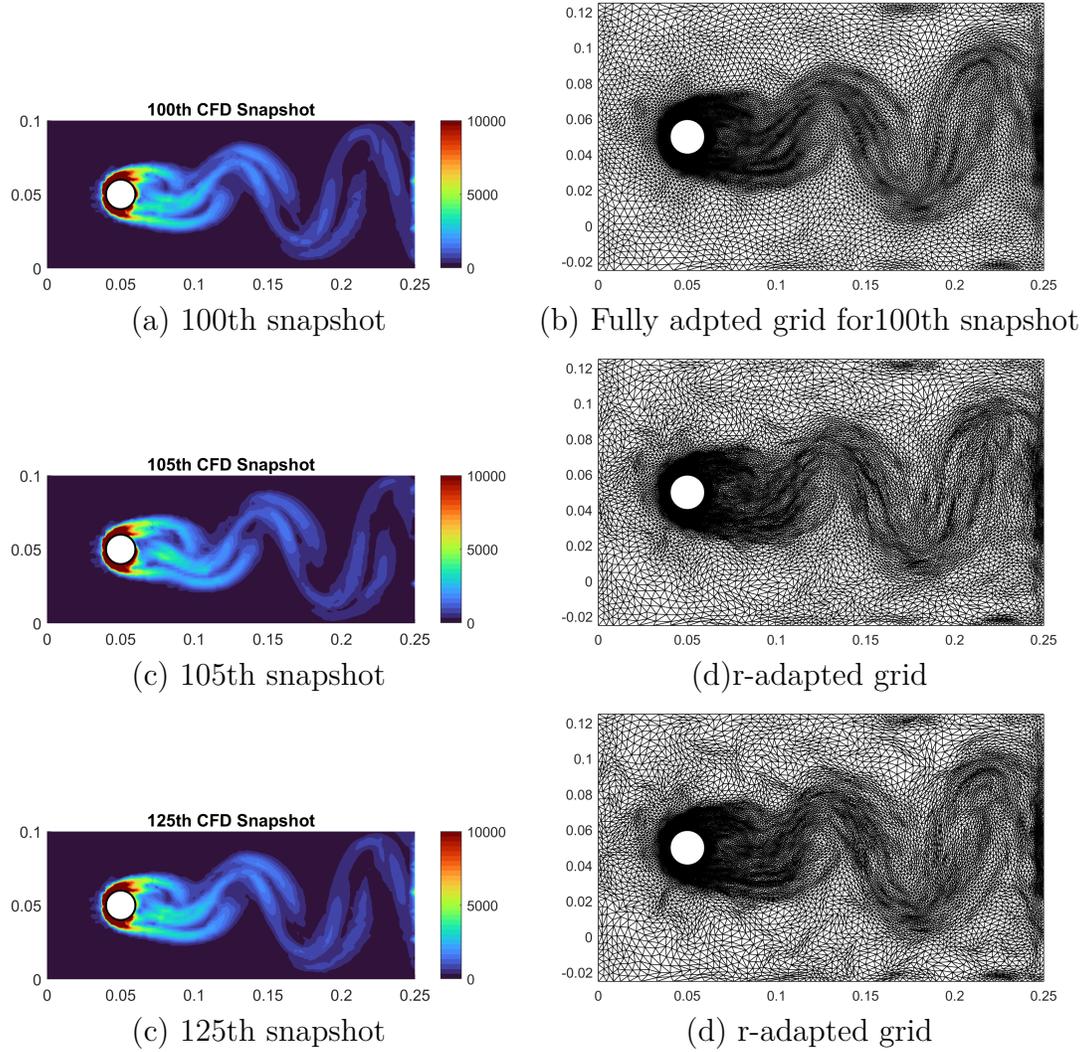


Figure A.1: R-adapted grids

Bibliography

- [1] W. Huang, Y. Ren, and R. D. Russell, “Moving mesh partial differential equations (mm-pdes) based on the equidistribution principle,” *SIAM Journal on Numerical Analysis*, vol. 31, no. 3, pp. 709–730, 1994.
- [2] J. Peraire, M. Vahdati, K. Morgan, and O. Zienkiewicz, “Adaptive remeshing for compressible flow computations,” *Journal of Computational Physics*, vol. 72, no. 2, pp. 449–466, 1987.
- [3] P. George, F. Hecht, and M. Vallet, “Creation of internal points in voronoi’s type method. control adaptation,” *Advances in Engineering Software and Workstations*, vol. 13, no. 5, pp. 303–312, 1991.
- [4] F. Alauzet and A. Loseille, “A decade of progress on anisotropic mesh adaptation for computational fluid dynamics,” *Computer-Aided Design*, vol. 72, pp. 13–39, 2016.
- [5] F. Alauzet, P. Frey, P. George, and B. Mohammadi, “3d transient fixed point mesh adaptation for time-dependent problems: Application to cfd simulations,” *Journal of Computational Physics*, vol. 222, no. 2, pp. 592–623, 2007.
- [6] F. Alauzet, A. Loseille, and G. Olivier, “Time-accurate multi-scale anisotropic mesh adaptation for unsteady flows in cfd,” *Journal of Computational Physics*, vol. 373, pp. 28–63, 2018.
- [7] G. Berkooz, P. Holmes, and J. L. Lumley, “The proper orthogonal decomposition in the analysis of turbulent flows,” *Annual Review of Fluid Mechanics*, vol. 25, no. 1, pp. 539–575, 1993.
- [8] A. Chatterjee, “An introduction to the proper orthogonal decomposition,” *Current science*, pp. 808–817, 2000.
- [9] K. Taira, S. L. Brunton, S. T. Dawson, C. W. Rowley, T. Colonius, B. J. McKeon, O. T. Schmidt, S. Gordeyev, V. Theofilis, and L. S. Ukeiley, “Modal analysis of fluid flows: An overview,” *Aiaa Journal*, vol. 55, no. 12, pp. 4013–4041, 2017.

- [10] L. SIROVICH, “Turbulence and the dynamics of coherent structures part i: Coherent structures,” *Quarterly of Applied Mathematics*, vol. 45, no. 3, pp. 561–571, 1987.
- [11] M. Sieber, C. O. Paschereit, and K. Oberleithner, “Spectral proper orthogonal decomposition,” *Journal of Fluid Mechanics*, vol. 792, p. 798–828, 2016.
- [12] P. J. Schmid, “Dynamic mode decomposition of numerical and experimental data,” *Journal of fluid mechanics*, vol. 656, pp. 5–28, 2010.
- [13] C. W. ROWLEY, I. MEZIĆ, S. BAGHERI, P. SCHLATTER, and D. S. HENNINGSON, “Spectral analysis of nonlinear flows,” *Journal of Fluid Mechanics*, vol. 641, p. 115–127, 2009.
- [14] A. Fidziańska and I. Hausmanowa-Petrusewicz, “Architectural abnormalities in muscle nuclei. ultrastructural differences between x-linked and autosomal dominant forms of edmd,” *Journal of the neurological sciences*, vol. 210, no. 1-2, pp. 47–51, 2003.
- [15] A. Loseille and F. Alauzet, “Continuous mesh framework part i: well-posed continuous interpolation error,” *SIAM Journal on Numerical Analysis*, vol. 49, no. 1, pp. 38–60, 2011.
- [16] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, “Log-euclidean metrics for fast and simple calculus on diffusion tensors,” *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, vol. 56, no. 2, pp. 411–421, 2006.
- [17] M. Castro-Díaz, F. Hecht, B. Mohammadi, and O. Pironneau, “Anisotropic unstructured mesh adaption for flow simulations,” *International Journal for Numerical Methods in Fluids*, vol. 25, no. 4, pp. 475–491, 1997.
- [18] F. Alauzet, A. Loseille, and G. Olivier, *Multi-scale anisotropic mesh adaptation for time-dependent problems*. PhD thesis, INRIA Saclay-Ile-de-France, 2016.
- [19] J. T. Batina, “Unsteady euler airfoil solutions using unstructured dynamic meshes,” *AIAA journal*, vol. 28, no. 8, pp. 1381–1388, 1990.
- [20] F. B. Ameer and A. Lani, “Physics-based r-adaptive algorithms for high-speed flows and plasma simulations,” *arXiv preprint arXiv:1808.09445*, 2018.

- [21] F. J. Bossen and P. S. Heckbert, “A pliant method for anisotropic mesh generation,” in *5th Intl. Meshing Roundtable*, vol. 63, p. 76, Citeseer, 1996.
- [22] B. O. Koopman, “Hamiltonian systems and transformation in hilbert space,” *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.
- [23] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, “Modern koopman theory for dynamical systems,” *SIAM Review*, vol. 64, no. 2, pp. 229–340, 2022.
- [24] J. H. Tu, *Dynamic mode decomposition: Theory and applications*. PhD thesis, Princeton University, 2013.
- [25] I. Mezic, “Spectral properties of dynamical systems, model reduction and decompositions,” *Nonlinear Dyn*, vol. 41, pp. 309–325, 2005.
- [26] I. Mezić, “Analysis of fluid flows via spectral properties of the koopman operator,” *Annual Review of Fluid Mechanics*, vol. 45, no. 1, pp. 357–378, 2013.
- [27] P. J. Schmid, “Dynamic mode decomposition and its variants,” *Annual Review of Fluid Mechanics*, vol. 54, no. 1, pp. 225–254, 2022.
- [28] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, “Spectral analysis of nonlinear flows,” *Journal of fluid mechanics*, vol. 641, pp. 115–127, 2009.
- [29] L. N. Trefethen and I. D. Bau, *Numerical Linear Algebra*. SIAM, 1997.
- [30] M. R. Jovanović, P. J. Schmid, and J. W. Nichols, “Sparsity-promoting dynamic mode decomposition,” *Physics of Fluids*, vol. 26, no. 2, 2014.
- [31] R. T. Biedron, J.-R. Carlson, J. M. Derlaga, P. A. Gnoffo, D. P. Hammond, K. E. Jacobson, W. T. Jones, B. Kleb, E. M. Lee-Rausch, E. J. Nielsen, *et al.*, “Fun3d manual: 13.7,” tech. rep., NASA Langley, 2020.
- [32] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, *et al.*, *LAPACK users’ guide*. SIAM, 1999.

- [33] L. S. Blackford, A. Petitet, R. Pozo, K. Remington, R. C. Whaley, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, *et al.*, “An updated set of basic linear algebra subprograms (blas),” *ACM Transactions on Mathematical Software*, vol. 28, no. 2, pp. 135–151, 2002.
- [34] C. Tsolakis, N. Chrisochoides, M. A. Park, A. Loseille, and T. Michal, “Parallel anisotropic unstructured grid adaptation,” *AIAA Journal*, vol. 59, no. 11, pp. 4764–4776, 2021.
- [35] M. Park and D. Darmofal, “Parallel anisotropic tetrahedral adaptation,” in *46th AIAA Aerospace Sciences Meeting and Exhibit*, p. 917, 2008.
- [36] M. S. Hemati, M. O. Williams, and C. W. Rowley, “Dynamic mode decomposition for large and streaming datasets,” *Physics of Fluids*, vol. 26, no. 11, 2014.
- [37] A. Roshko and U. S. N. A. C. for Aeronautics, *On the Development of Turbulent Wakes from Vortex Streets*. NACA R-1191, National Advisory Committee for Aeronautics, 1953.
- [38] J. N. Kutz, X. Fu, and S. L. Brunton, “Multiresolution dynamic mode decomposition,” *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 2, pp. 713–735, 2016.