Delft University of Technology

&

NLR - Royal Netherlands Aerospace Centre

MSc Thesis Aerospace Engineering

Reduced-Order Modeling of Wing Surface Pressure Distribution in Transonic Flow

A Machine Learning-Based Approach

Lars Boaz Compagne

Reduced-Order Modeling of Wing Surface Pressure Distribution in Transonic Flow

A Machine Learning-Based Approach

by

Lars Boaz Compagne

A thesis submitted to the Delft University of Technology in
partial fulfillment of the requirements for the degree of
Master of Science in Aerospace Engineering,
to be defended publicly on Wednesday, September 10, 2025, at 14:00.

The work in this thesis was carried out at NLR - Royal Netherlands Aerospace Centre.

Student Number:	4998537	
Supervisors:	Dr. S.J. Hulshoff	Delft University of Technology (TU Delft)
	Dr.ir. M.P.C. van Rooij	Royal Netherlands Aerospace Centre (NLR)
	ir. P. Blom	Royal Netherlands Aerospace Centre (NLR)
Thesis commitee:	Prof.dr.ir. L.L.M. Veldhuis	Aerospace Engineering – Flight Performance
	Dr.ir. F. de Prenter	Aerospace Engineering – Wind Energy
	Dr. S.J. Hulshoff	Aerospace Engineering – Aerodynamics
Project Duration:	Jan, 2025 - Sep, 2025	
Faculty:	Aerospace Engineering	

Preface

This thesis is the result of my research to complete the MSc in Aerospace Engineering specializing in Flight Performance and Propulsion at the Delft University of Technology. I conducted this research at NLR - Royal Netherlands Aerospace Centre in Amsterdam from January 2025 to August 2025. As a leader for aerospace research in the Netherlands, NLR offered the perfect environment for my research, for which I am very grateful.

The research objective of this thesis includes two of my main interests, which are flight physics and machine learning. Being able to work on efficiency improvements of flight physics modeling with novel machine learning techniques made me very enthusiastic to pursue this research. Hence, the results of this work are especially rewarding.

I want to thank my supervisors Steven Hulshoff, Michel van Rooij, and Peter Blom for their intensive support. Their guidance and feedback during this thesis have been invaluable in shaping and deepening the research. Furthermore, I would like to thank Yoeri Ton for his efforts in connecting me with NLR.

Lars Compagne
Amsterdam, August 2025

Abstract

Modern aircraft design requires efficient prediction of complex, time-dependent flow phenomena. While high-fidelity datasets offer accuracy, they come with high computational costs. To address this, NATO's research group AVT-351 is investigating reduced-order models (ROMs). Proper Orthogonal Decomposition (POD) is widely used in ROMs, but truncated POD bases struggle with sharp gradients and shocks. The enriched POD-LSTM neural network ROM (ePOD-LSTM-ROM) has demonstrated improved accuracy for airfoil pressure distributions and was extended to full wing surfaces. This approach was effective for sharp discontinuities moving significantly in time. For realistic cases, such as the ONERA M6 wing with multiple shocks, domain decomposition allowed for multiple enrichments in separate subdomains, but at the cost of rapidly increasing degrees of freedom. To improve LSTM-NN accuracy, enrichments were optimized using mutual information, filtering, and error trade-offs, reducing the total ePOD-LSTM-ROM error by 23.8%. Finally, a goal-oriented reduced-order model (GOROM) was developed to alter POD modes optimized for a specific goal function, improving shock accuracy by 5.0% with minor losses elsewhere.

Contents

List of Figures

List of Tables

Nomenclature

Symbols

Symbol	Definition	Unit
C_p	Pressure coefficient	–
f	Frequency	Hz
L	Length	m
M_∞	Free-stream Mach number	–
M	Number of snapshots	–
N	Number of POD modes	–
p	Static pressure	Pa
q	Pitch rate	Deg/s
Re	Reynolds number	–
St	Strouhal number	–
t	Time	s
U_∞	Free-stream velocity	m/s
x, y, z	Spatial coordinates	m
$\mathbf{a}(t)$	POD time coefficients	–
$\mathbf{C}$	Weight matrix	–
$\mathbf{f}$	Nonlinear operator / governing equations	–
$\mathbf{q}$	Flow state vector	–
$\mathbf{Q}$	Covariance matrix	–
$\mathbf{u}$	Velocity vector	m/s
α	Angle of attack	deg
γ	Ratio of specific heats	–
ϵ	Error metric	–
λ	Eigenvector	–
ρ	Density	kg/m^3
σ	Standard deviation / singular value	–
ϕ	POD mode	–
Δ	Difference	–
Φ	POD basis matrix	–
Ψ	Secondary basis	–

Abbreviations

Abbreviation	Definition
AE/AD	Auto-Encoder/Decoder
ANN	Artificial Neural Network
AVT	Applied Vehicle Technology
CFD	Computational Fluid Dynamics
DD	Domain Decomposition
DLR	German Aerospace Center
DL	Deep Learning
DNN	Deep Neural Network
DNS	Direct Numerical Simulations
ePOD	Enriched Proper Orthogonal Decomposition
FOM	Full-Order Model
GRU	Gated Recurrent Units
LSTM	Long Short-Term Memory
MAC	Mean Aerodynamic Chord
MI	Mutual Information
ML	Machine Learning
MSE	Mean Squared Error
NATO	North Atlantic Treaty Organization
NLR	Royal Netherlands Aerospace Centre
NN	Neural Network
POD	Proper Orthogonal Decomposition
RANS	Reynolds-Averaged Navier-Stokes
RNN	Recurrent Neural Network
RMS	Root Mean Square
ROM	Reduced-Order Model
S&C	Stability and Control
SBLI	Shock Wave-Boundary layer Interaction
STO	Science and Technology Organization
VAE	Variational Autoencoder
VLM	Vortex Lattice Method

1 Introduction

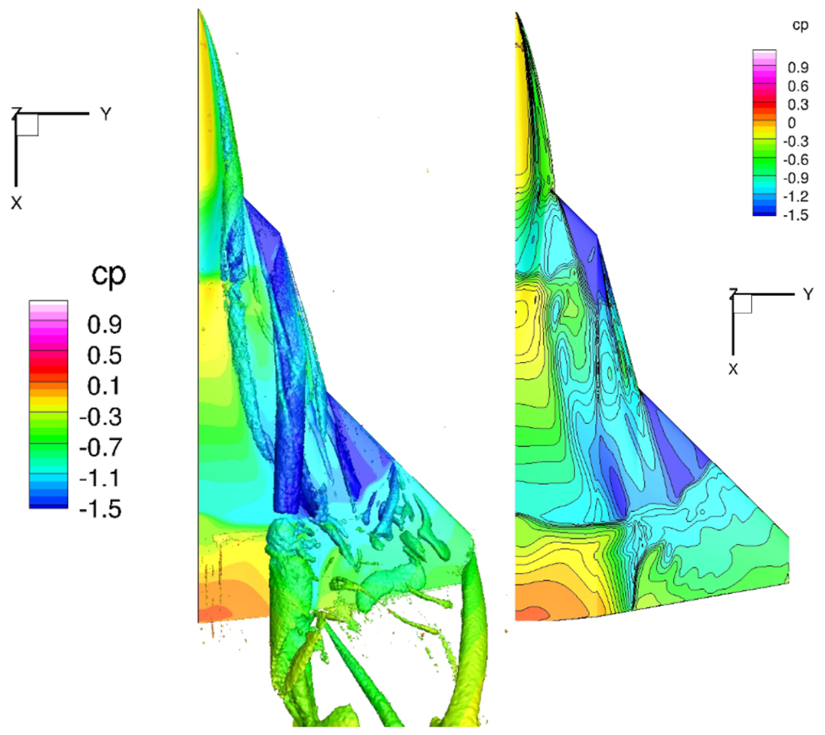
Despite designers committing many resources to the extensive design of highly agile aircraft, many of them showed unexpected aeroelastic problems during flight tests [1], [2]. During a flight test in 2001 for instance, a high angle-of-attack manoeuvre of Lockheed Martin's X-35 (later JSF) had to be aborted prematurely due to buffeting tails [1]. Moreover, predicting performance as well as Stability and Control (S&C) characteristics in the early design phase requires extensive high-fidelity datasets of unsteady non-linear aerodynamics [3]–[5]. With increasing agility and flight conditions of modern aircraft, the size of high-fidelity datasets, and thus computational costs, only increases. However, the initial design process is iterative, where fast turnaround times are needed to quickly develop a design that meets all the requirements [4]. As a result, creating high-fidelity datasets is not feasible due to its high computational cost. However, discovering these unfavourable phenomena only in the test phase can seriously impact either the performance or the design of the aircraft. Since changing the design late in the design cycle is very costly, methods to accurately predict aeroelastic problems early in the design phase are needed [3].

AVT-351 research task group, part of the Applied Vehicle Technology (AVT) panel, focuses on this specific issue [3], [6]. This group belongs to the Science and Technology Organization (STO) of the the North Atlantic Treaty Organization (NATO), who conducts and coordinates scientific research to support state-of-the-art defence and security technology [7]. The objective of AVT-351 is to "investigate and compare various numerical methods to create accurate performance and S&C datasets at feasible cost" [8]. The current approach under development is the application of reduced-order models (ROMs) to predict time-dependent manoeuvres, thereby reproducing high-fidelty datasets at low cost [3].

ROMs are mathematical representations of the results of Full Order Models (FOMs), using a limited number of spatial and temporal modes. This means that they have less degrees of freedom (DOF), making them less computationally heavy. In a typical CFD simulation, this reduction can be from 5-50 million or more grid points in the FOM to less than one hundred spatial/temporal modes in the ROM [9]. This leads to a significant faster response prediction, allowing for faster turnaround times during the initial design process. This research builds further on previous studies of an enriched Proper Orthogonal Decomposition (ePOD)-based ROM that employs a neural network to predict temporal coefficients [10]–[13]. Specifically, the ePOD-ROM will be extended from predicting the pressure distribution over airfoil sections [13] to covering the entire wing surface.

This thesis will consider the construction of a ROM suitable for the DLR-F22 ONERA model, also called Future Fighter Demonstrator (FFD). This is the use case for all of research task group AVT-351 [5]. FFD is a generic model of a three-delta wing fighter aircraft used in wind tunnel research [4]. The ROM will be constructed to model transonic flow fields at different angles of attack and pitch angles sequences. The geometry, vortex structure, and pressure distribution of the FOM at an arbitrary time instance is shown in Figure 1.1. This figure shows the complex flow characteristics influencing the surface pressure distribution, like shocks, and interaction of vortices.

In order to generate the models, the United States Air force Academy (USAFA) produced high fidelity datasets with CFD for the Pseudorandom Binary Sequence (PRBS) motion and Schroeder manoeuvrer input signals [13]. These manoeuvrers consist of periodic changes in angle of attack and pitch rate at 0.85 Mach. Different ROM techniques to approximate the flow field under these input signals are investigated in research task group AVT-351. The USAFA uses regression and a feed-forward neural network, DLR a surrogate-based recurrence framework, and NLR a Long Short-Term Memory (LSTM) neural network model in combination with enriched Proper Orthogonal Decomposition (ePOD) [5].



(a) (b)

Figure 1.1: FFD (a) vortex structure and (b) pressure distribution at $M_\infty = 0.85$ and $\alpha = 20$ deg [4]

1.1. Problem Statement

Making a Full-Order Model (FOM) of an unsteady non-linear airflow, especially in three dimensions, requires a huge amount of computational power. That is why less accurate models are sometimes used if the application allows for it. To be more precise, four levels of fidelity exist [14], [15]. Level 0 consists of typical initial aircraft design relations and empirical data. In level 1 fidelity, only linear effects are captured and the physical relations are simplified. In this level, mainly the Vortex Lattice Method (VLM) is used. Level 2 refers to a detailed representation of the aircraft components, where non-linear phenomena are captured. It is mainly based on Euler equations where viscosity is not considered. Lastly, level 3 fidelity performs viscous simulations with non-linear effects. These simulations are typically based on the Reynolds-averaged Navier–Stokes (RANS) equations which require a turbulence model. While level 1 fidelity simulations usually take minutes, a level 2 takes hours and a level 3 can take days.

A FOM refers to the level 3 fidelity, thereby either by solving the RANS/LES (Large Eddy Simulation) equations or by means of experimental (wind tunnel) data [3]. Simulations with this level of fidelity are able to capture aerodynamic effects impacting aeroelastic design and fatigue analysis caused by turbulent flow, separation, and shock waves [5]. However, due to their high computational cost, these simulations are only used to determine aerodynamic coefficients at a few important design flight conditions. Moreover, experimental data is not available to a useful extent, since modern wind tunnels are not always able to measure flow characteristics at high manoeuvrability and high angles of attack [16]. In addition, constructing and testing wind tunnel models in the early design phase is not feasible since many different configurations are to be tested [4].

Furthermore, it is important to know what type of high fidelity data is required for a proper analysis. In addition to the standard force and moment coefficients and derivatives, the occurrence of aeroelastic issues influence the design as well. Transonic control surface buzz phenomenon for example, a self-excited limit-cycle oscillation of control surfaces in the transonic flow regime [2], is a common aeroelastic issue. Aeroelastic problems come in two types, aeroelastic stability problems and and dynamic response problems [2]. The former is caused by fluid structure interaction with coupled feedback, usually in stable free-stream flow. The latter is caused by unsteady flows like the shock wave–boundary layer interaction (SBLI), creating aerodynamic force variations.

SBLIs are often the cause of buffet and flow unsteadiness in transonic flows [17]. When this is the case, shock waves are formed at the downstream edge of a supersonic region which causes a sudden pressure jump and increase in adverse pressure gradient in the boundary layer, pushing the shock forward. This increases the relative Mach number seen by the shock, eventually increasing its strength and causing separation of the boundary layer [17], [18]. As separation decreases the effective airfoil curvature, the shock wave weakens and eventually makes the boundary layer reattach again. Now, the shock wave moves back and sees a lower relative Mach Number, so no separation occurs. As the shock wave stops moving, separation occurs and the cycle starts over again [17], [18].

This showcases the importance and need for high-fidelity simulations during the initial design phase, thereby fully representing the effects of the viscous boundary layer and accurately predicting separation. Moreover, since dynamic response problems are often independent of the motion of the structure (except in the case of high-aspect-ratio sailplanes), the analysis can be done independently as well, making it a decoupled and purely aerodynamic problem [2]. This means that by reproducing high-fidelity aerodynamic data in an efficient manner, like with ROMs, one can make a huge impact in tackling design problems in the very early design phase.

In a previous study, the ePOD-ROM with a long short-term memory Neural Network (LSTM-NN) showed promising results for airfoil section data [13]. The pressure coefficient along the chord length, including shock(s), was accurately predicted based on the angle of attack, pitch rate, and their derivatives. This method makes use of a piecewise linear sawtooth enrichment function at the shock region to smoothen the pressure distribution seen by the POD. Doing so allows for less POD modes to accurately define the pressure distribution. However, extending this method to the full wing surface still has foreseeable problems. These are listed below:

- Since discontinuities in the pressure distribution can be oriented in different directions, a different method to detect them is needed. This also leads to different enrichment modes.
- When multiple discontinuities are present, multiple enrichment domains are needed. The wing surface should be divided such that each subdomain contains only one discontinuity, otherwise enriching becomes too complex. However, this leads to a non-straightforward surface decomposition since the amount and location of discontinuities changes over time. So, an automated method to create subdomains is needed.
- Multiple discontinuities in the pressure distribution can intersect, leading to a complex enrichment domain. This means that strict rules should be employed to decide on how the subdomains should be made in case of an intersection point.
- The interface between adjacent subdomains should not introduce irregularities in the flow field.
- Different discontinuities could require different enrichment modes. This means that the enrichment modes of each subdomains need to be predicted.
- The order of the ROM should be substantially less than the FOM.

Addressing these issues is a necessary step toward developing a robust method for constructing an ePOD-LSTM-ROM capable of predicting the aerodynamics of agile aircraft wing surfaces. Previous research dedicated to the development of this method [10]–[13] is used for the LSTM-NN architecture and the ePOD decomposition method.

In addition to the challenges associated with extending the ePOD-LSTM-ROM from an airfoil section to a complete wing surface, increased prediction accuracy is needed. For an airfoil section, the LSTM-NN error was demonstrated to be an order of magnitude higher than the error of the reduced-basis, or projection error [13]. This means that although the reduced basis allows for an accurate reconstruction of the flow field, the LSTM-NN is not able to predict the time coefficients accurately. As a result, the method seems to be very accurate when only considering the projection error, while in reality the total error is a lot higher. So, a method has to be found that considers this error already early on in constructing the reduced-order basis, by designing the enrichments such that they are suitable for regression.

Feature selection techniques, like mutual information (MI) or Pearson's correlation coefficient [19]–[21], could provide a solution. In processes that employ large input datasets, feature selection is used to identify non-redundant subsets that capture the most relevant information. Applying feature selection techniques showed to boost neural network prediction accuracy substantially [22], [23]. Therefore, it should be investigated if a feature selection technique could be employed to improve the correlation between input signals and time coefficients which boosts neural network performance.

Lastly, methods other than POD in combination with enrichments should be considered. Previous research showed that improvements in numerical accuracy compared to POD can also be achieved by means of a goal-oriented reduced-order model (GOROM) [24], [25]. As POD constructs modes based on energy content, flow features with low-energy content could be truncated while it could be a very important feature. In GOROM, modes are found with numerical optimization in a user-defined norm, which could lead to modes that represent particular flow features more accurately. Previous research demonstrated improvements in reconstructing a two-dimensional transitional boundary layer over a flat plate [24]. This indicates that applying the GOROM method to a wing surface with flow discontinuities could be promising.

1.2. Research Objectives

Following up on the problem statement, the research objective of this MSc Thesis can be summarized as follows:

"The research objective is to develop a method to construct a Reduced-Order Model (ROM) that can accurately predict the unsteady surface pressure distribution in the shock region of a wing planform in the transonic flow regime, by creating modes that allow for a decrease in the number of temporal coefficients of the reduced order basis to be evaluated by a Long Short-Term Memory (LSTM) neural network"

Leading to the following research questions:

1. How well does an ePOD-LSTM-ROM perform in multidimensional space?
    (a) How to implement an ePOD-LSTM-ROM in multidimensional space?
    (b) What are the limiting factors of an ePOD-LSTM-ROM in multidimensional space?
2. To what extent can feature selection integrated into the enrichment design improve the predictive accuracy of neural networks?
3. To what extent can goal-oriented mode optimization improve projection accuracy in discontinuous flow field domains?

1.3. Report Outline

The report is structured as follows: chapter 2 presents previous work conducted needed for the research in this thesis. In chapter 3, the development of the ePOD-ROM method in multidimensional space is explained together with a performance evaluation. Thereafter, chapter 4 explores a method to anticipate the neural network performance before training the neural network. Subsequently, chapter 5 explains a method to construct modes purely through optimization. The conclusions and recommendations are given in chapter 6. Lastly, Appendix A provides additional information about deep learning and Appendix B additional results.

2 Literature Review

In this chapter, the theoretical basis for this research is laid. The first section discusses three methods to construct a reduced order model, that of proper orthogonal decomposition, neural networks, and a hybrid method. Subsequently, the method of domain decomposition is explained. Lastly, transonic buffet onset characteristics in POD are explained in detail. Moreover, an additional chapter on the principles of deep learning and different neural network architectures to predict parameters is available in Appendix A.

2.1. Reduced Order Models

Present-day CFD simulations are getting more and more accurate but still have trouble with separation and turbulence. Experiments however have a limit on test conditions. This means that often CFD simulations are used to create high fidelity datasets, doing so by numerically solving the Navier-Stokes equations with turbulence models for millions of grid points, leading to very high computational cost and long simulation time [26]–[29]. Moreover, the high-fidelity data utilizes a lot of memory, making regressions in time inefficient. That is why Reduced Order Models (ROMs) that can produce accurate results at a lower cost are needed, especially in the early design phase where many iterations are required [4].

In general, a model that reduces the degrees of freedom can be called a ROM. This can be achieved by using less nodes, bigger time steps, or simplified equations that represent the physical behaviour [30]. However, since the accuracy should not be affected, a projection-based method is often used to construct the ROM. This method uses a reduced, or low-dimensional, basis of vectors or equations to define the full order model (FOM) [26], [29], [30]. This reduced basis is also called the latent space, since it contains the most important information about the FOM in a very efficient and compressed form. The reasoning comes from the study of turbulent flow structures, where it was found that also in these non-linear flows, patterns exist [30], [31].

A ROM approximates a function f, shown in Equation 2.1, with a surrogate function $\hat{f}$, such that the FOM solution y based on parameters p is approximated as accurately as possible [32]. The surrogate function $\hat{f}$ generally produces the solutions much faster than the FOM. The construction of $\hat{f}$ is called the offline step, which is the most computationally heavy step but only has to be done once. Computing the result with Equation 2.1 is called the online step [30], [33]. The online step is generally extremely efficient making it very fast and convenient to generate parameters compared to the FOM. The difference between the computational phases is highlighted below.

$$\hat{\mathbf{y}} = \hat{f}(\mathbf{p}) \approx f(\mathbf{p}) \tag{2.1}$$

1. **Offline Step -** Establishing the reduced basis $\hat{f}$ of the FOM (ROM training)
2. **Online Step -** Resolving the reduced basis for new parameter values (ROM approximation)

The methods used in the offline-online phases distinguish ROMs from surrogate models, which are similar but not the same. A surrogate model does not construct an accessible reduced basis but rather searches for direct relations between input and output data, which are usually unknown. This means that a surrogate model can be regarded as a black-box model [26]. Furthermore, the method of a ROM in the online step can be classified as intrusive or non-intrusive. The key differences between surrogate models and ROMs are explained below [26], [30]:

- **Surrogate Model -** Regression is performed on the original dataset. Original input and output data is used to approximate relationships, of which the result is a black-box. This model is purely data-driven. *Example: Neural network that maps input to output*
- **ROM -** Regression is performed on a lower dimensional basis, or latent space, which contains compressed information of the original dataset. *Example: POD*
  - **Intrusive -** Physics-based model where the FOM PDE's describing the physical behaviour are needed to project the high-fidelity dataset onto the latent space. *Example: Galerkin projection*
  - **Non-Intrusive -** Data-driven regression where no information is provided about the original equations describing the physical behaviour. *Example: Neural network that performs regression on parameters of ROM*

2.1.1. Proper Orthogonal Decomposition

The concept of POD was introduced in the turbulent fluid dynamics field by Lumley in 1967 first [34]. In different fields, POD is also referred to as Karhunen-Loève decomposition or principal components analysis (PCA) [31], [35], [36]. Nowadays, POD is the preferred modal decomposition method in many fields since it's a linear process and very robust [31]. In the field of fluid dynamics, a POD decomposes the flow field into base functions, also called POD modes, which all capture a part of the kinetic energy [35], [37]. The term 'proper' in POD comes from the fact that the first n POD modes give the lowest projection, or L_2, error. This means that the most dominant flow characteristics are kept in the first modes. Moreover, 'orthogonal' indicates that all modes are orthogonal to each other, where each mode represents different flow characteristics without overlapping. This means that the preserved energy fraction of the flow is maximum for a given number of modes [37].

First of all, the high-fidelity dataset is constructed from experiments or CFD simulations. The latter is based on the Navier-Stokes equations (NSE), which for an incompressible flow and non-dimensionalised for the Reynolds Number can be written as follows [38]:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\frac{1}{\rho}\nabla p + \frac{1}{Re}\nabla^2 \mathbf{u}, \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned} \tag{2.2}$$

Here, ($\mathbf{u}$) is the velocity, (p) the pressure, (ρ) the density, and (Re) the Reynolds Number. The first equation represents the momentum equation, while the second equation corresponds

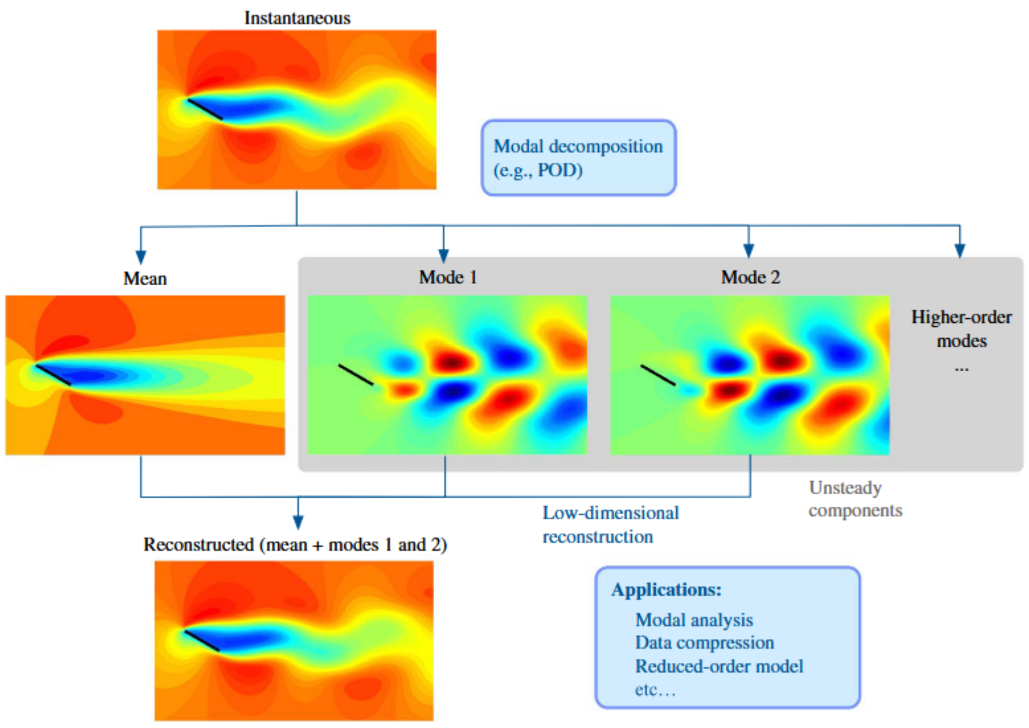


Figure 2.1: A two-dimensional POD of the u component of the velocity over a flat-plate [35]

to the continuity equation. The high-fidelity dataset, in turn, can be used to construct the POD modes. The POD procedure described above is visualized for the incompressible flow over a two-dimensional flat plate in Figure 2.1. As seen, only two POD modes describe this relatively simple flow field already pretty accurate. The mathematical procedure of the modal decomposition is explained below, based on the methods of [35]–[37], [39].

Spatial (Classical) POD Method:
A quantity (e.g., pressure, velocity) in a flow field at a certain time instance, or snapshot, is denoted by $q\left(\mathbf{x},t\right)$. Here, $\mathbf{x} = (x, y, z)$ denotes the spatial coordinates in three-dimensions and (t) the time step. The first step is to obtain the fluctuations of the quantity by subtracting the temporal mean, this is shown in Equation 2.3. Here $\mathbf{q}'\left(\mathbf{x},t\right)$ denotes the fluctuations vector and $\overline{\mathbf{q}}\left(\mathbf{x}\right)$ the temporal mean.

$$\mathbf{q}'\left(\mathbf{x},t\right) = \mathbf{q}\left(\mathbf{x},t\right) - \overline{\mathbf{q}}\left(\mathbf{x}\right) \tag{2.3}$$

The principle of POD is to decompose the flow field as shown in Equation 2.4. In this equation, $\phi_{\mathbf{k}}(\mathbf{x})$ are the POD modes and $a_k(t)$ are the corresponding time coefficients.

$$\mathbf{q}'\left(\mathbf{x},t\right) = \sum_{k=1}^{\infty} a_k(t)\phi_{\mathbf{k}}(\mathbf{x}) \tag{2.4}$$

The earlier mentioned property of orthogonality comes from the relation in Equation 2.5. This indicates that each time coefficient only depends on the corresponding spatial mode, and not on other modes. This means that Equation 2.6 is true as well.

$$\iiint_x \phi_{k_1}(x)\phi_{k_2}(x)\,dx = \begin{cases} 1 & \text{if } k_1 = k_2 \\ 0 & \text{if } k_1 \neq k_2 \end{cases} \tag{2.5}$$

$$a_k(t) = \iiint_x \mathbf{q}'(\mathbf{x},t)\ \phi_{\mathbf{k}}(\mathbf{x})\,dx \tag{2.6}$$

To determine the POD modes of Equation 2.4, the following procedure is needed. First of all, the high fidelity dataset must be ordered in matrix **Q**. As shown in Equation 2.7, this matrix contains m columns for each individual snapshot. The size, n, of the snapshot matrix comes from the number of spatial nodes. Subsequently, the POD modes can be found by solving the eigenvalue problem of the covariance matrix of **Q**. The covariance matrix and eigenvalue problem are shown in Equation 2.8 and Equation 2.9 respectively.

$$\mathbf{Q} = [\mathbf{q}'(t_1)\,\mathbf{q}'(t_2)\ldots\mathbf{q}'(t_m)] \in \mathbb{R}^{m\times n} \tag{2.7}$$

$$\mathbf{C} = \frac{\mathbf{Q}\mathbf{Q}^T}{m-1} \in \mathbb{R}^{n\times n} \tag{2.8}$$

$$\mathbf{C}\phi_{\mathbf{k}} = \lambda_k\phi_{\mathbf{k}}, \qquad \phi_{\mathbf{k}} \in \mathbb{R}^n, \qquad \lambda_1 \geq \ldots \geq \lambda_k \geq 0 \tag{2.9}$$

Here, $\mathbf{Q}^T$ denotes the transpose of matrix **Q**, $\phi_{\mathbf{k}}$ the eigenvectors of C or POD modes, and λ_k the eigenvalues of C. The eigenvalues denote the amount of kinetic energy captured by each POD mode, so in a decreasing order they structure these modes in order of importance. To determine the number of POD modes needed, the portion of the total amount of kinetic energy in the flow field captured with r modes can be calculated with Equation 2.10. This must be close to 1 to conserve the most important flow features.

$$I = \frac{\sum_{k=1}^{r}\lambda_k}{\sum_{k=1}^{n}\lambda_k} \tag{2.10}$$

The temporal coefficients can de determined by using Equation 2.11. Moreover, with the first r POD modes considered, the flow field can be approximated by rewriting Equation 2.12 to Equation 2.11. Now, the order of the high-dimensional flow field is reduced from n to r.

$$a_k(t) = \sum_{k=1}^{\infty}\mathbf{q}'(\mathbf{x},t)\,\phi_{\mathbf{k}}(\mathbf{x}) \tag{2.11}$$

$$\mathbf{q}'(\mathbf{x},t) \approx \sum_{k=1}^{r} a_k(t)\phi_{\mathbf{k}}(\mathbf{x}) \tag{2.12}$$

Subsequently, the reconstruction, or projection, error of the POD method can be calculated according to Equation 2.13 [40]. This error is the average root mean square error of all snapshots m in the dataset. In addition, this equation can be used to determine the amount of POD modes needed to obtain a certain accuracy ϵ.

$$\text{err} = \frac{1}{m}\sum_{j=1}^{m}\left\| q' - \sum_{k=1}^{r} a_k(t)\phi_{\mathbf{k}}(\mathbf{x}) \right\|_j \leq \epsilon \tag{2.13}$$

Method of Snapshots:
In the application of fluid dynamics, the number of spatial nodes n is generally very large. This means that the covariance matrix C of size (n× n) gets very large, making it very difficult to solve the eigenvalue problem. It was found, however, that the eigenvalue problem of the covariance matrix for temporal modes with size (m× m) results in the same spatial eigenvectors [35]. Since usually the number of snapshots is smaller than the number of spatial nodes ($m \ll n$), this covariance matrix is easier to solve. The equation is shown in Equation 2.14.

$$\mathbf{Q}^T\mathbf{Q}\psi_{\mathbf{k}} = \lambda_k\psi_{\mathbf{k}}, \qquad \psi_k \in \mathbb{R}^m, \qquad m \ll n \tag{2.14}$$

Since by solving Equation 2.14 one obtains the eigenvectors for the temporal covariance matrix, a conversion to spatial eigenvectors has to be made. This is shown in Equation 2.15.

$$\phi_{\mathbf{k}} = \frac{\mathbf{Q}\psi_{\mathbf{k}}}{\sqrt{\lambda_k}} \in \mathbb{R}^n; \qquad k = 1, 2, \ldots, m \tag{2.15}$$

Singular Value Decomposition (SVD):
A variant on POD is singular value decomposition (SVD) [35]. In SVD, the eigenvectors can be extracted from a rectangular matrix directly, so without the conversions to the square covariance matrix first. The decomposition of data matrix **Q** according to SVD is shown in Equation 2.16.

$$\mathbf{Q} = \mathbf{\Phi}\mathbf{\Sigma}\mathbf{\Psi}^T \tag{2.16}$$

In this equation, $\Phi = [\phi_1\ \phi_2\ \ldots\ \phi_m] \in \mathbb{R}^{n\times m}$, $\Psi = [\psi_1\ \psi_2\ \ldots\ \psi_m] \in \mathbb{R}^{m\times m}$, and Σ is a diagonal matrix $\mathbb{R}^{n\times m}$ with singular values $[\sigma_1\ \sigma_2\ \ldots\ \sigma_m]$ on the diagonal. The singular values squared equal the eigenvalues of the temporal covariance matrix, $\sigma_k^2 = \lambda_k$. The property of SVD is that the columns of matrices Φ and Ψ hold the eigenvectors of $\mathbf{Q}\mathbf{Q}^T$ and $\mathbf{Q}^T\mathbf{Q}$ respectively [35]. These are called the left and right singular vectors, which are the orthonormal output and input bases respectively. SVD can be seen as a method to obtain the solution of a POD, however for large datasets, the method of snapshots is preferred [35].

Intrusive Prediction with POD

Predictions of unseen outputs from unseen input parameters can be done in an intrusive manner by projecting the Navier-Stokes on the truncated POD modes. A common technique for doing this is that of Galerkin projection [41], which rewrites the Navier-Stokes equations such that they describe the temporal POD modes evolution. Recall from Equation 2.12 that r POD modes form the basis to describe the zero-mean parameter, say $u(x,t)$ in case of velocity.

The Navier-Stokes equation of Equation 2.2, but also other partial differential equations (PDE), can be written in a general form as [41], [42]:

$$\frac{\partial u}{\partial t} = \frac{1}{Re}\nabla^2\mathbf{u} - \mathbf{u}\cdot\nabla\mathbf{u} = L[u] + N[u;u] \tag{2.17}$$

where, $u(x,t)$ is the velocity, $L[u]$ representing the linear terms, and $N[u;u]$ representing the non-linear terms. To project the Navier-Stoke equation on the POD base, the inner product with each POD mode $\phi_k(x)$ is taken:

$$\left\langle \frac{\partial u}{\partial t}, \phi_k \right\rangle = \langle L[u], \phi_k\rangle + \langle N[u;u], \phi_k\rangle, for \quad k = 1, ..., r \tag{2.18}$$

By substituting the POD expression for u in this equation, defined in Equation 2.12, the following expression is obtained.

$$\sum_{i=1}^{N} \dot{a}_i(t) \left\langle \phi_i, \phi_k \right\rangle = \frac{1}{Re} \sum_{i=1}^{N} a_i(t) \left\langle \nabla^2 \phi_i, \phi_k \right\rangle - \sum_{i=1,j}^{N} a_i(t) a_j(t) \langle \phi_i \cdot \nabla \phi_j, \phi_k \rangle, for \quad k = 1, ..., r \tag{2.19}$$

Recall from Equation 2.5 that the POD modes are orthonormal, meaning that the inner product on the left hand side is zero if i≠k and 1 if i=k. As a result, the time derivative of a at i=k is considered only. This leads to the following system of ordinary differential equations (ODEs) for the time coefficients $a_k(t)$:

$$\dot{a}_k(t) = \frac{1}{Re} \sum_{i=1}^{N} \mathcal{L}_{ik} a_i(t) - \sum_{i=1,j}^{N} \mathcal{N}_{ijk} a_i(t) a_j(t), for \quad k = 1, ..., r \tag{2.20}$$

Where the terms $\mathcal{L}_{ik}$ and $\mathcal{N}_{ijk}$ are the inner product of the linear and non-linear terms with the POD modes respectively as explained below.

$$\mathcal{L}_{ik} = \left\langle \nabla^2 \phi_i, \phi_k \right\rangle \tag{2.21}$$

$$\mathcal{N}_{ijk} = \langle \phi_i \cdot \nabla \phi_j, \phi_k \rangle \tag{2.22}$$

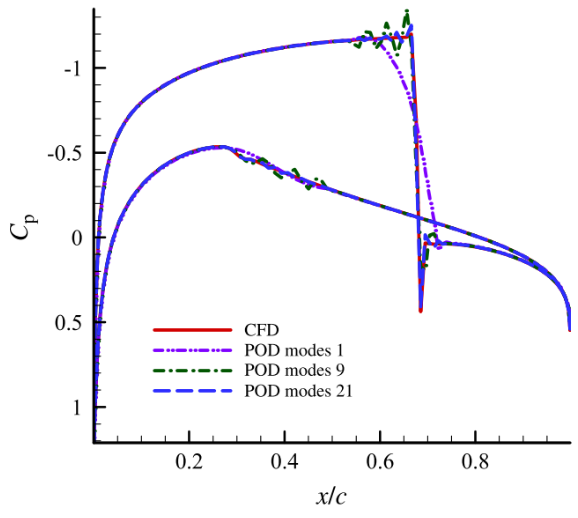
With the temporal evolution of the time coefficients and an initial condition, one can calculate the flow field properties of a future time instance. Equation 2.11 can be used to project the initial flow field on the POD modes to calculate the initial time coefficients to be used in Equation 2.20.

2.1.2. Enriched Proper Orthogonal Decomposition

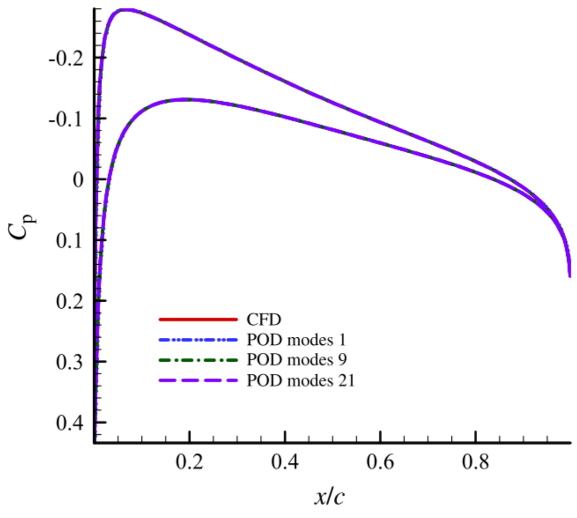
A truncated basis that consists of a low number of POD modes can be used to accurately reproduce laminar and turbulent flows [13], [35]. However, inaccurate results are achieved for flow fields that have a lot of variance and convection [29]. To be more precise, flow fields that have sharp gradients, shocks, or interfaces moving in time, remain very difficult to model with a truncated POD basis [43]. This difficulty comes from the fact that moving discontinuities alter the flow field significantly locally, so many globally defined POD modes are needed to account for these local effects [43]. In this case, too many modes are required to reconstruct the discontinuity accurately, causing an increase in the number of DOF of the ROM.

The problem is shown in Figure 2.2 for a NACA 0012 airfoil. As can be seen, in the subsonic case, 1 POD mode is enough to accurately model the pressure distribution. However, for the transonic case, 9 POD modes are needed to match the FOM with a low error. In this example, 1 POD mode at M = 0.5 captures 99.9% of the total kinetic energy while 9 POD modes are needed for 99.0% and 21 POD modes for 99.9% at M = 0.8 [44]. As can be seen, by using more POD modes, the FOM in the shock region is approached with higher frequency and lower amplitude oscillations. To use this many POD modes however, increases the order and computational cost of the ROM by a lot.

A solution to this problem is that of enriched-POD (ePOD) [13], [29], [45], which results in a reduction in the number of DOF while the same accuracy is maintained for flow fields containing discontinuities. In ePOD, the latent basis is enriched with one or more numerical functions, which can be arbitrary or learned from local spatial subdomains [29]. Since they support the solution in small spatial subdomains only, they are a lot more effective in capturing discontinuities. This means that less modes are needed and the DOF and cost are kept low. Two methods for ePOD are discussed, one with a designed enrichment function in one dimension and one with a locally learned mode in two dimensions.



(a) $M = 0.8$.



(b) $M = 0.5$.

Figure 2.2: NACA 0012 pressure distribution at $\alpha = 1.65$ deg and (a) M = 0.8, (b) M = 0.5 comparison between FOM and three different amounts of POD modes [44]

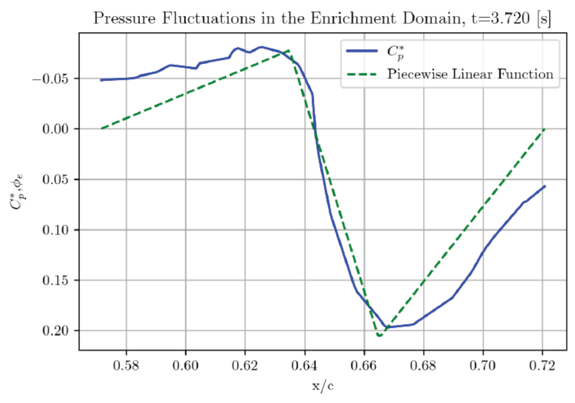
One-dimensional enrichment function:
Panagiotopoulos [13] proposed an ePOD method specifically designed to account for discontinuities in the pressure distribution caused by shock waves. The governing equation is shown in Equation 2.23.

$$C_{pk} = \bar{C}_p + \underbrace{\sum_{k=1}^{r} a_k(t)\phi_k(x)^T}_{\text{Standard Basis}} + \overbrace{\sum_{s=1}^{i} \phi_{e_s}(x, p(t))}^{\text{Enrichment Basis}} \tag{2.23}$$

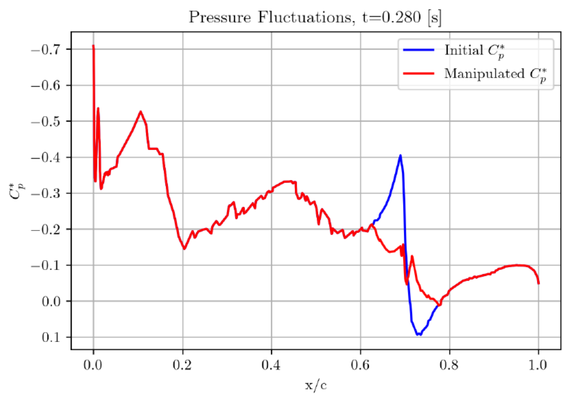
In the equation above, r defines the amount of truncated spatial modes, $\phi_k(x)$ and $a_k(t)$ the spatial modes and time coefficients respectively, i the amount of shocks, and ϕ_{e_s} the enrichment functions defined with p(t). The piecewise linear sawtooth function was the selected enrichment function, of which the mathematical formulation is defined in Equation 2.24. The parameters a, b, x_1, and x_2 define the shape of this function, which are amplitudes and locations of the two control points respectively. These parameters are embedded in the time-dependent p(t) of ϕ_{e_s}. Moreover, this function ensures that the Dirichlet Boundary Condition is met, meaning that its value is zero at the boundary of the enrichment domain to omit induced discontinuities. The fitted sawtooth enrichment function to the pressure fluctuation plot of the DLR-F22 Onera model is shown in Figure 2.3a.

$$\phi_e(x, p(t)) = \begin{cases} \frac{ax}{x_1} & \text{for } x < x_1 \\ \frac{(b-a)x}{x_2-x_1} + \frac{ax_2-bx_1}{x_2-x_1} & \text{for } x_1 < x < x_2 \\ \frac{-bx}{1-x_2} + \frac{b}{1-x_2} & \text{for } x > x_2 \end{cases} \tag{2.24}$$

Since the enrichment function should only be active locally at the shock location, a shock sensor was employed. This shock sensor detected steep pressure gradients and market them as a shock location. Subsequently, the enrichment domain was defined with a certain chord wise distance from this shock location. To obtain the pressure fluctuation distribution to be reconstructed by the general POD modes, the enrichment function was subtracted from the data. As seen in Figure 2.3b, the manipulated pressure fluctuation plot is smoothed out very effectively.



(a) Sawtooth enrichment function fitted to the pressure distribution



(b) Sawtooth enrichment function subtracted from pressure fluctuation plot

Figure 2.3: C_p smoothing of section 1 of the DLR-F22 Onera model with linear sawtooth enrichment under the Schroeder manoeuvre [13]

As a result of adopting this enriched basis, the projection error of the ePOD modes is a lot lower with less DoF compared to POD, as shown in Figure 2.4. To be more precise, the time-averaged projection error of POD was 17.75% higher than that of ePOD with the same number of DoF. Moreover, the projection error was 58.64% higher in the enrichment domain only. This significant improvement with the same amount of DoF leads to a more accurate result without increasing the cost.

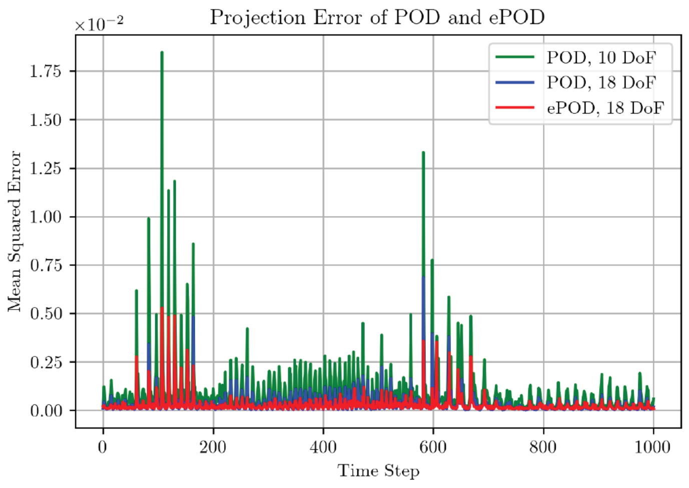


Figure 2.4: Projection error in Section 1 of the DLR-F22 model [13]

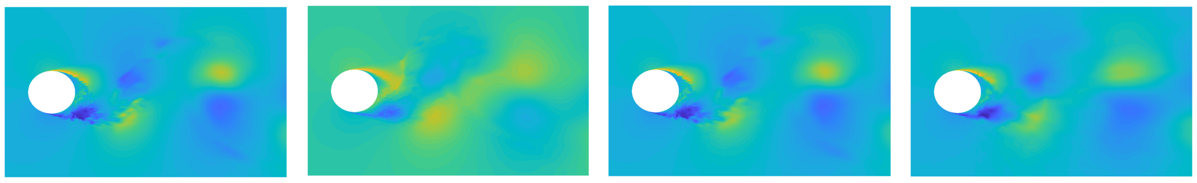
Two-dimensional generalized finite element:
Another method, presented by Deshmukh [29], makes use of support functions applied locally in each node. These support functions consist of FEM base functions and generalized finite element (GFEM) enrichment functions of a higher order. The FEM functions of a lower order

should reconstruct the large-scale structures while the high-order enrichment functions capture the repetitive small-scale structures. The governing equation is shown in Equation 2.25.

$$u_h(x,t) = \sum_{\alpha=1}^{n} \hat{u}_\alpha(t)\phi_\alpha(x) + \sum_{\alpha=1}^{n} \phi_\alpha(x) \sum_{i=1}^{m} \tilde{u}_{\alpha i}(t) L_i(x) \tag{2.25}$$

In this equation, $u_h(x,t)$ is the reconstructed solution at a certain spatial location and time step, n is the number of spatial nodes, m is the number of enrichment functions applied to each node in the domain, ϕ_α is the FEM shape base of node α, L_i is the GFEM enrichment function, and $\hat{u}_\alpha$ and $\tilde{u}_{\alpha i}$ are the time coefficients.

Subsequently, next to the general POD procedure, the local enrichment enrichment functions are learned. A key difference is that the enrichment functions are learned from a single snapshot, instead of all the snapshots in the dataset. To construct a local data matrix, p random patches of equal size are selected from the snapshot. This data matrix is used to construct the GFEM enrichment functions, of which the first m functions are selected to use. Patches p have size $(2nskip + 1)^3$, which is based on a pre-set grid coarsening of nskip number of nodes skipped in between each new node spacing. An example of the GFEM method for the flow over a cylinder is shown in Figure 2.5. A total of 3000 snapshots were used to construct the global POD modes.



(a) Test snapshot

(b) POD with 50 modes, NRMSE = 59.81%, CR = 60×

(c) GFEM, nskip = 9, 10 enrichments, NRMSE = 3.25%, CR = 60×

(d) GFEM, nskip =24, 5 enrichments, NRMSE = 23.50%, CR = 1760×

Figure 2.5: u' velocity component of 3D flow past a cylinder at Re=100,000 [29]

As seen in Figure 2.5b, 50 POD modes are needed for a compression ratio of only 60. Meanwhile, the normalized root mean square error (NRMSE) is still very high with 59.81%. For the GFEM approach, the same compression ratio is obtained with 10 enrichment modes and a node coarsening of 9, while the NRMSE is only 3.25%. Moreover, a compression ratio of 1760 can be obtained with 5 enrichment modes and a node coarsening of 24. However, this implies a higher NRMSE of 23.50%. This highlights the power of enrichment modes in efficiently constructing data snapshots.

2.1.3. Artificial Neural Networks

Next to POD and ePOD, an artificial neural network (ANN) can be used for model order reduction. ANNs are used in machine learning (ML), or deep learning (DL) to be more precise, for their ability to learn complex patterns [26]. Moreover, ANNs are data-driven or non-intrusive in their nature, meaning that the problem is represented by a black-box model which results in easy implementation and fast turnaround times [32]. There is no physical reasoning behind a black-box model, since the PDE's are not accessed. This section introduces ANNs in the use case of a reduced order modeling technique, while Appendix A, explains the working principles and different types of ANNs.

A widely used DL technique making use of ANNs to transform a FOM to a latent space is that of autoencoders (AE) [26], [32]. An AE is a NN that reconstructs high dimensional data by first transferring it to a low dimensional bottleneck and subsequently restoring the dimension, and by doing so learns the most important low-dimensional features of the data [32]. This is a form of unsupervised learning since these features are extracted without labels assigned to the data, [30]. Essentially. this is achieved by employing two NNs, one being the encoder and one the decoder, connected with a latent layer. The encoder, typically referred to as h, reduces the high-dimensional data to a low dimensional latent base, following $\mathbb{R}^n \rightarrow \mathbb{R}^d$ [32]. Each data point of the input layer is connected to the elements in the first encoder layer, also called neurons [26]. These neurons are in turn connected to the next layer of neurons, and together, they form the hidden layers L of the encoder. If a NN has more than one hidden layer, it is called a deep neural network (DNN) [30]. This is usually the case for autoencoders since they have to recognize complex, non-linear patterns. The dimension of each layer gets lower with a downsampling constant r, until dimension d of the latent layer is reached. Elements z in the latent layer contain the information of the input data in a very compressed manner. Subsequently, the decoder g reconstructs the solution from the latent representation with an equal amount of hidden layers L, thereby increasing the dimension to match the original data $\mathbb{R}^d \rightarrow \mathbb{R}^n$ [32]. An example of an AE used for modal decomposition of two-dimensional flow field snapshots is shown in Figure 2.6. In this example, high-fidelity data snapshots are fed into the encoder which transforms the data into a low-dimensional latent vector. Subsequently, the decoder increases the dimensionality to create high-fidelity flow field modes, which are combined to reconstruct the input flow field.

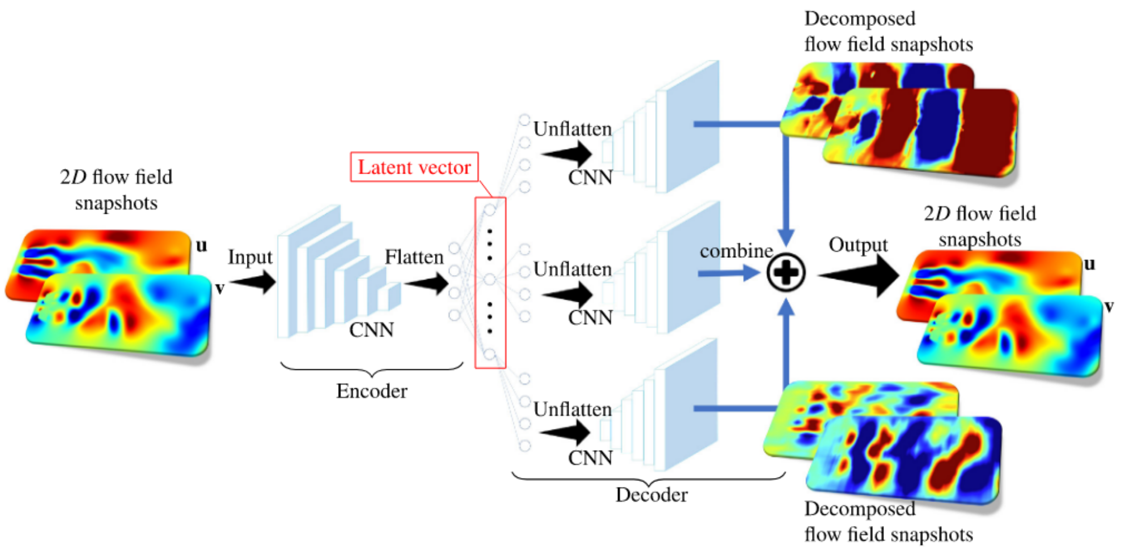


Figure 2.6: Concept of an autoencoder for flow field order reduction [46]

The working principle of an AE can be compared to POD or ePOD in the following way. First of all, the time coefficients in POD would be equivalent to the latent space vector of the AE since both of them are a set of weights defining the most important features of the solution. Secondly, the multiplication of the time coefficients with the POD modes would be equivalent to the decoder since this reconstructs the data [26]. However, as opposed to POD, which is a linear method, neural networks are non-linear, enabling them to capture complex flow field characteristics more accurately [46]. Moreover, non-linear reduced spaces for ROMs obtained with neural networks are generally more compact than obtained with POD. As a result, the regression step of this reduces space in the ROM is better manageable [26]. However, it is important to note that an AE with only one hidden layer that employs a linear activation function, produces similar results as POD/SVD [30].

The downside of using an AE however, is that a lot of training data is needed. In the application of flow field reconstruction, this would be high-fidelity snapshots of different flow fields. Moreover, the training process of the AE using these data-sets can take very long. For example, the training of an AE for transonic flow data with 7 encoder and decoder layers where the amount of neurons ranges from 512 to 7 takes 100.2 minutes for 20000 epochs [26]. Considering that POD time is almost zero, this is a considerable duration. Moreover, since the latent space is created with a non-linear data-driven process, it is very hard or even impossible to interpret it.

Non-Intrusive Prediction with neural networks

Besides its use in autoencoders, neural networks can be used to predict parameters values based on different input parameters. In its simplest form, this can be done by a multilayer perceptron (MLP), which is a type of artificial neural network (ANN) that consists neurons structured in multiple layers [47]. The MLP can be used to perform supervised learning, to understand the relations between inputs θ and outputs $\mathbf{u}$ in a non-intrusive manner as shown in Equation 2.26. The model then delivers outputs $\tilde{\mathbf{u}}$ which are estimations for the true high fidelity parameters $\mathbf{u}$.

$$\mathbb{M} : \theta \to \tilde{\mathbf{u}} \tag{2.26}$$

In its simplest form, the values of θ are used to predict $\mathbf{u}$ using a feed-forward neural network (FNN) [48]. In these networks, information flows in one direction only. However, when $\mathbf{u}$ depends also on the order of inputs θ, there has to be a loop of information in the network. When this is the case, a recurrent neural network (RNN) has to be used [26]. More details of these networks is provided in Appendix A.

2.1.4. Hybrid models

A special method is that of the so-called hybrid models, making use of both POD and a NN [5], [28], [33], [49]. In a hybrid model, high-fidelity data is first decomposed in a low-order basis using POD whereafter a NN is used to perform regression on the POD time-coefficients. The hybrid approach takes advantages of the benefits of both methods, reducing the drawbacks of the overall method.

The idea of the ROMs proposed in subsection 2.1.1-subsection 2.1.3 is that a computationally heavy offline stage is used to build the reduced-order base, which only has to be done once. After that, the extremely efficient online stage is used to reconstruct the solution [42]. In the case of a NN however, very high-dimensional FOMs require a more complex NN architecture. This leads to more parameters to be estimated and more training data to accurately establish these parameters. This means that the offline stage can get too computationally heavy, ensuring that the construction of the NN becomes infeasible [33]. Moreover, in the case of POD, very high-dimensional FOMs with complex non-linear characteristics could lead to the need of many POD modes. This comes from the fact that at high Reynolds Numbers, more POD modes are needed to obtain a certain energy content portion [50]. When the order of the dimension of the latent space gets close to that of the FOM, the computational cost in the online stage has not improved [51].

To improve the computationally heavy offline stage experienced with NNs, POD is applied to reduce the dimensionality of the FOM first. This makes training less computationally heavy since there are less parameters describing the flow field, allowing for a smaller NN architecture [33]. Moreover, the cost of the online stage is improved compared to POD alone due to the non-intrusive nature of the NN, which is faster than an intrusive Galerkin projection [41]. Also, the non-linearity introduced by the NN improves the efficient representation of complex flow fields.

Another advantage, different from computational cost, is that the NN creates a linear combination of POD modes, which are constructed from real datasets. This prevents the NN from producing unphysical flow fields, which makes POD-NN a semi-intrusive method. Moreover, this makes it possible to trace the source of errors [5]. Since a NN produces its results in a black box, the origin of errors is unknown. However, in the hybrid approach, projection errors and NN errors can be distinguished, as stated below [28].

1. **Projection error -** The MSE between the high-fidelity data and the projection of this data on the truncated POD modes. So this is the maximum achievable accuracy with the selected POD modes.
2. **NN error -** The MSE between the time coefficients predicted by the NN and the time coefficients obtained by projecting the high-fidelity data on the POD modes. Results from insufficient training data or design of the NN.

Figure 2.7 shows the workflow diagram of a type of POD-NN method [52]. In this example, spectral proper orthogonal decomposition (SPOD) is employed to filter the covariance matrix if there is much noise in the data, caused by high-frequency flow structures. By doing so, the obtained temporal coefficients and modes from SVD contain less noise, making the regression task of the NN more convenient. Subsequently, the POD modes and temporal coefficients are obtained using SVD. Moreover, the time derivates of the temporal coefficients are obtained using Galerkin Projection. These values are fed to the NN which performs regression between the time coefficients and their derivatives. Lastly, the trained model can be used to reconstruct flow fields by outputting the temporal evolution of the time coefficients.

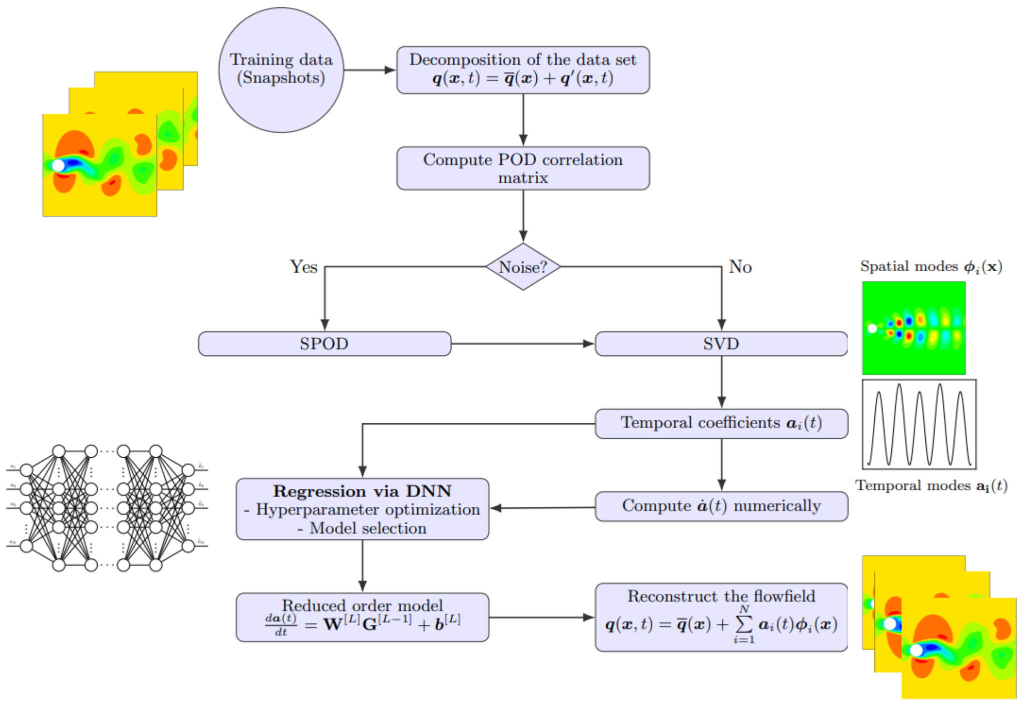

Training data (Snapshots)
Decomposition of the data set $q(x,t) = \overline{q}(x) + q'(x,t)$
Compute POD correlation matrix
Noise?
Yes
No
SPOD
SVD
Spatial modes $\phi_i(\mathbf{x})$
Temporal coefficients $a_i(t)$
Temporal modes $\mathbf{a_i}(t)$
Regression via DNN
- Hyperparameter optimization
- Model selection
Compute $\dot{a}(t)$ numerically
Reduced order model $\frac{da(t)}{dt} = \mathbf{W}^{[L]}\mathbf{G}^{[L-1]} + b^{[L]}$
Reconstruct the flowfield $q(x,t) = \overline{q}(x) + \sum_{i=1}^{N} a_i(t)\phi_i(x)$

Figure 2.7: Workflow diagram of the POD-NN method for flow field reconstruction [52]

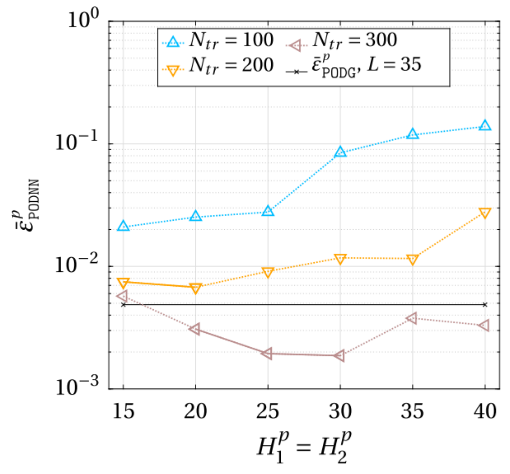
Non-Intrusive Prediction with POD-NN

Next to simply reconstructing flow fields, POD-NN can be used to predict flow fields for unseen input parameters that fall within the range of input parameters used to create the training snapshots. As opposed to POD, where only the time derivates of the temporal coefficients can be used to make predictions of the near future, NNs can be employed to predict the time coefficients based on different input parameters θ as shown in Equation 2.26. Algorithm 1 shows the steps used to train the NN to predict the time coefficients of the POD modes for inputs θ [47], [53].

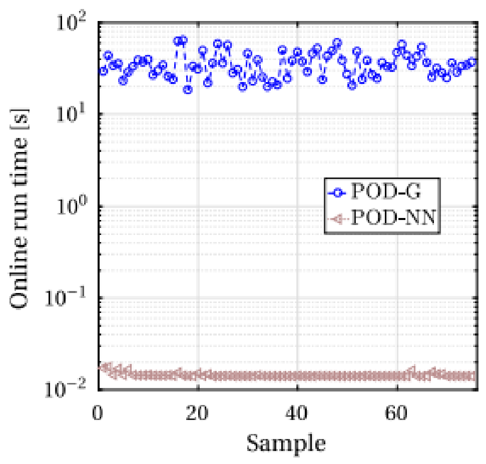
Algorithm 1 Machine learning pressure field POD time coefficients

```
   Input: High-fidelity snapshots at different $\theta$
   Output: Trained NN $\mathbb{M}$
   Offline Phase:
1: Compute POD modes using SVD
2: Compute time coefficients by projecting snapshots on truncated POD modes
3: Combine input parameters $\theta$ and time coefficients as training dataset
4: Train NN by minimizing the error between $\mathbf{a_i(t)}$ and $\mathbf{\tilde{a}_i(t)}$
   Online Phase:
5: Verify accuracy of trained NN $\mathbb{M}$ output $\mathbf{\tilde{a}_i(t)}$ for unseen inputs $\theta$
```

To assess the accuracy of this method, a comparison between POD-Garlekin and POD-NN was made by Hesthaven [51]. The use case was the 2-dimensional lid-driven cavity problem for viscous flow at Reynolds Numbers 200 and 400. The NN consists of two hidden layers with an equal amount of neurons and the number of POD modes used is 35. Figure 2.8a shows the relative error of the pressure distribution for different amounts of training samples used to train the NN and neurons present in each layer. As seen, the error decreases when more training samples are used. However, the error increases when the amount of neurons increases and the amount of training samples is insufficient. Only for 300 training samples, the error decreases up to 30 neurons per layer. Moreover, only when sufficient training samples and neurons are used, the error is lower than the POD-Galerkin method, which is the case for 300 training samples. POD-Galerkin can however, be used to make predictions outside of the range of input parameters used to create the FOM snapshots.



(a) Relative pressure error for training samples N_{tr}, amount of neurons H, and 35 POD modes L



(b) Online run time for different samples

Figure 2.8: Lid-driven cavity problem for Re=200 comparison of POD-Galerkin and POD-NN [51]

These results verify the statements made about the hybrid method in the beginning of this section. However, it also becomes clear that the accuracy is very sensitive to the design of the NN architecture and training samples used. This means that an extensive NN design process and thorough sensitively analysis is needed.

Figure 2.8b shows the online run time for each method. As can be seen, for all 75 samples tested, the order of magnitude is around 3 times more for POD-Galerkin. However, the offline runtime should be taken into account as well, which is a lot higher for the NN. The NN training phase takes about 8 hours, of which one hour is used to generate snapshots, but computing the POD basis takes around half an hour [51].

Another investigation for the flow over a NACA 4412 airfoil was made to compare the offline and online run times between different ROM models [54]. In the test case, the angle of attack was fixed at 5°, and the Reynolds number varied between 10^5 and 10^6. The CPU time to run each FOM ranged from 2,5 to 7 hours, dependent on Reynolds number. Two types of ROMs were made, 1- and 2-dimensional, the former simulating the airfoil wall parameters and the latter simulating the flow field parameters around the airfoil. To train the ROMs, 70 snapshots were used. The DOF of these snapshots is 965 for the 1-dimensional case and $\sim 2.5 \times 10^5$ for the 2-dimensional case. The number of latent parameters used in the ROMs on the other hand, is 3. Table 2.1 shows the simulation times for both the offline and online stages of the different types of ROMs.

As seen in Table 2.1, the time to construct the reduced bases with POD and train the ANN is more for the 2-dimensional case, which comes from the difference in initial dimension of the FOM. Moreover, POD has the shortest offline runtime with milliseconds to seconds while the ANN has the longest runtime with tens of seconds. It can also be seen that by combining POD with an AE, the training time almost halves. Next to this, the online reconstruction times are all less than a second, with POD-ANN the fastest method. This stresses the effectiveness of reduced order modeling and that of combining POD and ANNs.

Table 2.1: Offline and online simulation times for different combinations of ROM types [54]

Model	**Field**	**Offline time [s]**
POD	1-Dimensional	$\sim 4 \times 10^{-3}$
POD	2-Dimensional	~ 2.18
AE	1-Dimensional	$\sim 40 - 50$
PODAE	2-Dimensional	$\sim 25 - 35$
ANN	1-Dimensional	$\sim 7 - 10$
ANN	2-Dimensional	$\sim 80 - 100$
Model	**Field**	**Online time [s]**
POD-ANN	1-Dimensional	$\sim 2 \times 10^{-4}$
POD-ANN	2-Dimensional	$\sim 2 \times 10^{-3}$
AE-ANN	1-Dimensional	$\sim 4 \times 10^{-4}$
PODAE-ANN	2-Dimensional	$\sim 4.3 \times 10^{-2}$

2.2. Domain Decomposition

The POD-NN method for model order reduction is a promising technique for accurately reconstructing flow fields against highly reduced computational times. However, for the design of unconventional or highly agile aircraft, ROMs often can't accurately represent complex shock interactions and other non-linear features that are needed to determine its performance [55]. As seen in subsection 2.1.2, these highly discontinuous flow fields proved to be difficult to represent accurately with a low number of truncated modes, and a significant increase in dimensionality was needed. The method of adding enrichment modes that are locally active has shown to increase the accuracy of the ROM with still a low number of truncated modes. However, deciding where to add enrichments and how to ensure that they do not introduce irregularities in the flow field still poses a challenge. In this section, methods to decompose the domain in subdomains where enrichment modes are active are discussed.

One of the challenges of enriching is to ensure that there are no discontinuities introduced on the boundary between two adjacent subdomains. Possible solutions include adding additional constraints to the domain, for example ensuring that the enrichment is zero at the domain boundaries or ensuring that the parameter of interest is equal at both sides of the domain boundary. It is also possible to incorporate the time coefficients of neighbouring domains [56]. Another solution is to use the gappy-POD approach for the interface region, which was introduced first by R. Everson and L. Sirovich [57] to reconstruct images with missing pixels. With this method, the values in the interface region are hidden and the values of surrounding nodes are used to construct global POD modes which are used to approximate the missing values.

The domain decomposition (DD) step is performed after the high-fidelity data is produced. After that, the domain should be divided such that each subdomain contains only one discontinuity for all the snapshots. Subsequently, the high-fidelity data points are separated to create individual datasets for each subdomain. The shape of the subdomains can be triangles, quadrilaterals, and rectangles since the method is robust enough [55]. The decomposition of the domain can be automatized by using shock sensors based on pressure gradients, by minimizing the total error of the ROM, or by using arbitrary domains that capture individual discontinuities for all time instances.

Constructing the POD modes for each subdomain can be done according to localized global POD (LG-POD) or local POD (L-POD) [58]. In the first approach, the POD modes are constructed from the monolithic solution, which is the global domain, whereafter these modes are spatially cut to obtain separate modes for each subdomain. In the second approach, the solution is first separated into its subdomains, after which the POD modes are created separately for each subdomain. Iyengar et al. [55] present different methods to create these subdomains.

1D case - DD of converging–diverging nozzle:
The first case considered is that of the flow through a converging–diverging nozzle with a moving shock [55], [59]. The goal is to create subdomains that contain shocks, for which independent ROMs are developed. Subsequently, the global solution is reconstructed by combining the subdomains through a connection of their boundaries. In this case, a shock sensor that uses a first-order, forward finite difference scheme to measures the pressure gradient at every consecutive node is employed to detect the shock locations for every snapshot. The node with the highest gradient is selected as the shock location. Subsequently, the most forward and aft shock locations of all snapshots are selected as forward and aft bound of the subdomain respectively [55]. This ensures that the shock is located in the subdomain for all the conditions encountered. The DD of the converging-diverging nozzle is shown in Figure 2.9.

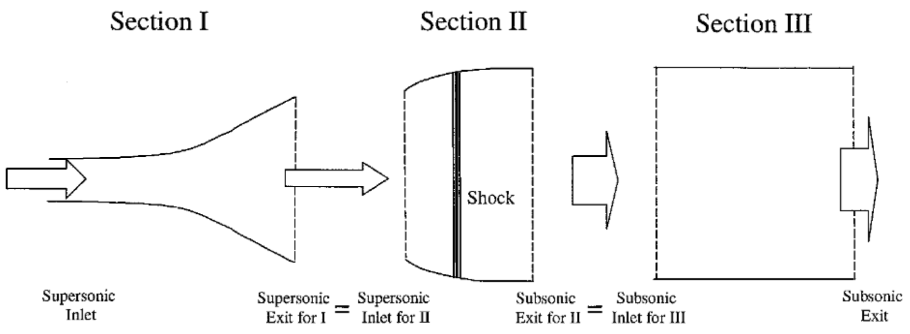

Section I
Section II
Section III
Shock
Supersonic Inlet
Supersonic Exit for I = Supersonic Inlet for II
Subsonic Exit for II = Subsonic Inlet for III
Subsonic Exit

Figure 2.9: Domain decomposition of the one-dimensional nozzle, the shock is located in subdomain 2 for every snapshot [59]

The results of the study, indicated by the relative error between the pressure variation of the FOM and the ROM are shown in Table 2.2. The global ROMs are constructed from the full domain pressure data while the DD ROMs are constructed separately for each subdomain. For the DD ROMs, the subdomains are linked according to gappy-POD, ensuring that there are no discontinuities introduced at the subdomain boundaries. Moreover, the non-linear ROM method used in this example is that of isometric feature mapping (ISOMAP), which structures and maps data points by obtaining the shortest path between them using non-linear algorithms [60]. However, for the non-linear ROM, a NN could be used as well. The last ROM type uses POD in the subdomains without shocks (i.e. subdomains 1 and 3), while using ISOMAP in subdomain 2 which contains the shock.

As seen in Table 2.2, DD decreases the error for POD in subdomains 2 and 3, which are highly dependent on the shock. However, the total error is slightly higher. For the non-linear ROM on the other hand, DD leads to a lower error in subdomains 1 and 3, leading to a slightly lower error in the full domain. Lastly, the combined DD method results in a lower error compared to global POD and global non-linear ROMs. The non-linear DD ROM results in the lowest full domain error, however, this introduces the highest cost as well. The combined POD and ISOMAP with DD leads to a substantial decreased error compared to global POD, mitigating the computational cost.

Table 2.2: Prediction error of the pressure variation through the converging-diverging nozzle with DD and ROM types [55]

ROM type	**Subdomain 1** $[10^{-4}]$	**Subdomain 2** $[10^{-4}]$	**Subdomain 3** $[10^{-4}]$	**Full Domain** $[10^{-4}]$
Global POD	1.63	220.0	10.3	29.5
Global non-linear	3.33	153.0	10.7	22.1
DD POD	2.54	218.0	9.25	29.8
DD non-linear	1.35	154.0	9.84	20.7
DD POD + non-linear	2.55	154.0	9.24	21.4

2D case - DD of flow over wedge and airfoil:
To understand the DD in two dimensions, two cases are considered: supersonic flow over a wedge and transonic flow over an airfoil [55]. The domain in the first case is decomposed based on the position of the attached oblique shock. Again, the target of the DD is to isolate the shock for all instances. By analysing the oblique shock for different input parameters in the dataset, the minimum and maximum angle could be found. The domain was divided into three overlapping subdomains, one in front and behind the shock, and one containing the shock, shown in Figure 2.10. The boundaries of subdomain 2 which contains the shock are defined with a margin of five nodes away from the minimum and maximum oblique shock position. In the overlap between subdomains, which is 10 nodes wide, gappy-POD is applied. For this case, DD reduced the error with 9% with POD and 6% with ISOMAP. The POD + ISOMAP DD ROM even reduced the error with 46% compared to global POD [55].

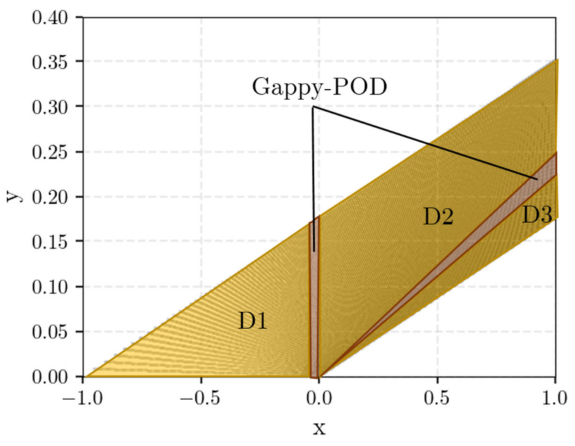


Figure 2.10: DD of transonic flow over two-dimensional wedge, leading edge at (0,0) [55]

For the transonic flow over an airfoil case, a two-step approach was used to define the subdomain containing the shock. First of all, the most forward and aft shock positions were evaluated based on the FOM dataset. Again, an area bounded by these minimum and maximum shock positions is dedicated as one subdomain on the airfoil upper surface. In this case, the rest of the domain makes up the other subdomain, which means that there are two subdomains in total. The second step entails a compass search optimization algorithm, which is a gradient-free search optimization were both subdomain boundaries are moved further away from each other until the lowest DD ROM error is found [55]. This research however, calls for more research into automating the domain decomposition procedure based on shock-sensors, automatic clustering, and gradient-based optimizers for example.

To analyse this method, first the number of modes required to capture 99.99% of the total energy are evaluated for the linear POD method. Where 21 POD modes were needed for the global POD, 16 and 12 modes were needed for the DD POD in subdomain 1 and 2 respectively for 3 design variables. When more design variables are used, which leads to more variation in the flow field, the DD POD proves to be even more effective in reconstructing the flow field. For 15 design variables, the truncated POD base reduces from 54 modes to 37 in subdomain 1 and 27 in subdomain 2. This means that the POD modes represent the flow better in each domain. As for the error, a 5% decrease was seen in the shock region on average and a decrease of 11% on average in the non-shock area. For the non-linear ROMs on the other hand, an average decrease in error of 29% was observed for the non-shock area while the average error in the shock area didn't improve. However, the overall error decreased for both ROMs [55].

Optimization of DD interface:
Prusak et al. [61] presented a similar DD approach, however, instead of using Gappy POD for the interface between subdomains Ω_1 and Ω_2, an optimization problem to minimize the normal flux was introduced. To do so, Equation 2.27 is used, where $\mathcal{J}_\gamma$ is the cost function that has to be minimized. This equation consists of two terms, the continuity on the boundary Γ_0 connecting two subdomains and a regularization term that keeps the normal stress g on the boundary low [61]. γ denotes the regularization parameter which puts more or less importance on keeping this normal stress low. Moreover, n represents the time step.

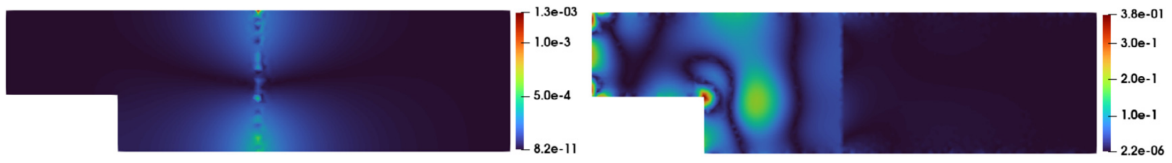
Equation 2.28 shows the equation for the modified normal stress acting on the boundary between subdomains. Here, ν denotes the viscosity, u_i^n the velocity vector, n_i the normal vector, p_i the pressure, and i the subdomain Ω_i. The additional term $\frac{1}{2}(u_i^n \cdot \mathbf{n_i})u_i^n$ accounts for convection due to the fluid flow motion and is subtracted to obtain the fluctuation of the pressure, as is usually done in ROMs.

$$\min \mathcal{J}_\gamma(u_1^n, u_2^n; g) := \frac{1}{2}\int_{\Gamma_0} |u_1^n - u_2^n|^2 \, d\Gamma + \frac{\gamma}{2}\int_{\Gamma_0} |g|^2 \, d\Gamma \tag{2.27}$$

$$g \approx \nu \frac{\partial u_i^n}{\partial \mathbf{n_i}} - p_i \mathbf{n_i} - \frac{1}{2}(u_i^n \cdot \mathbf{n_i})u_i^n, \quad i = 1, 2. \tag{2.28}$$

This optimization problem can be solved in an iterative manner by means of gradient-based algorithms. During this iterative process, the flow fields of both subdomains can be solved independently for a certain value of g. Thereafter, g is updated with the gradient-based algorithm until the loss function $\mathcal{J}_\gamma$ converges [61]. This leads to a solution where the difference in velocity vectors on both sides of the subdomain boundary and the interface normal stress are at a minimum.

Figure 2.11 shows the absolute error of the reconstructed pressure field with the DD method described above compared to the monolithic solution for the backward-facing step test case. In this case, the domain is split into a forward and aft subdomain. As can be seen, the error of the FOM is mainly concentrated around the interface while the ROM displays errors in the whole first subdomain. The latter being caused by the truncated POD base. The magnitudes of the absolute errors at the interface however, are in the order of 1.0e-2 or lower meaning that there are no large discontinuities introduced at the interface. This shows that POD in combination with domain decomposition is possible without introducing discontinuities at boundaries of adjacent subdomains.



(a) DD with FOM **(b)** DD with POD-Galerkin ROM

Figure 2.11: Absolute pressure errors compared to the monolithic solution for the backward–facing step test case at the final time step. [61]

2.3. Transonic Buffet Analysis with ROMs

An important analysis made with CFD simulations is that of transonic buffet behaviour. When a ROM is used to generate the aerodynamic data, these types of analyses should still be possible. That is why this chapter discusses the capability to identify transonic buffet based on POD modes. Specifically, POD modes with a specific Strouhall Number bandwidth are presented to identify transonic buffet.

Transonic wing buffeting, which is also called wing shock buffeting, occurs when the wing contains an oscillating shock wave and boundary layer separation [62]. This forward and aft movement of the shock in chord wise direction usually creates in peak in the two-dimensional airfoil pressure spectrum of rectangular wings at a frequency of St = 0.06-0.07 [63]. The definition of the Strouhal number is shown in Equation 2.29, where f is the frequency, L is the characteristic length, in this case the MAC, and U_∞ the freestream velocity. For three-dimensional swept wings, however, transonic buffeting is very different compared to two-dimensional rectangular wings and still not fully understood [63]–[65]. But, since the vibration of the wing is dependent on the pressure distribution, it is important to obtain very accurate data. This section will highlight the extent to which this is possible with POD.

$$\mathrm{St} = \frac{fL}{U_\infty} \tag{2.29}$$

Figure 2.12 shows the pressure distribution of the Airbus XRF-1 swept wing in transonic flow ($M_\infty = 0.9$) for different angles of attack. As can be seen, a lambda shock is formed with a strong shock branch on the aft part of the wing chord from root to tip. A second, weaker shock branch, emerges from the root leading edge and coincides with the stronger shock at about 40% of the half span. For larger angles of attack, the shock gets more intense and shifts forward. Moreover, the weaker shock moves inboard.

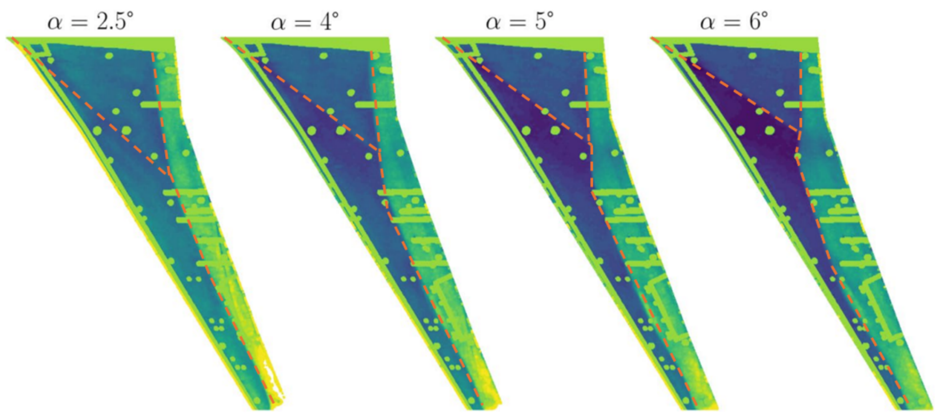


Figure 2.12: Pressure distribution of buffet flow of Airbus XRF-1 wing, $M_\infty = 0.9$, Re = $25{\times}10^6$ [62]

As opposed to the pressure spectrum peak at a frequency range of St = 0.06-0.07 in two dimensions, an additional peak in range St = 0.2-0.6 is seen in three dimensional swept wings [63]. Within this frequency range, it was shown with experimental data obtained with unsteady pressure measurements and pressure-sensitive paint, and subsequently a decomposition with POD, that the pressure fluctuations propagate from the root in spanwise direction. The flow structures resulting from this pressure propagation are called buffet cells [66]. To understand which three-dimensional coherent structures contribute to this behaviour, zonal detached-eddy simulations (ZDES), validated with experimental data, followed by POD can be used. This method is applied to the NASA CRM swept wing at M=0.85, Re=1.516×10^6, and $\alpha = 4.87$ since experiments showed buffetting for these conditions.

After obtaining the POD base, Dynamic Mode Decomposition (DMD) can be applied to obtain dominant temporal modes, and thus the structures and their frequencies with which they evolve in time. Equation 2.30 shows the linear operator A which contains the time evolution of each consecutive mode. $\tilde{\Psi}_0 = [\tilde{\psi}_1, \ldots, \tilde{\psi}_{N-1}]$ and $\tilde{\Psi}_1 = [\tilde{\psi}_2, \ldots, \tilde{\psi}_N]$, where $\tilde{\psi}_i$ is the time coefficient vector for the truncated POD base of length r at timestep i. By calculating the eigenvalues λ and eigenvectors ϕ of matrix A, one obtains the DMD modes [63].

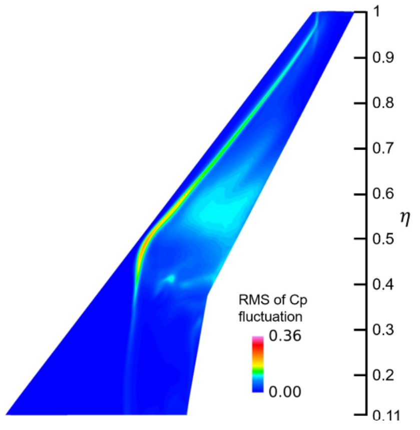
$$\tilde{\Psi}_1 \approx A\tilde{\Psi}_0 \tag{2.30}$$

The rate at which each mode growths in amplitude σ and frequency St in which each mode oscillates can be calculated from eigenvalue λ with Equation 2.31 and Equation 2.32, where Δt denotes the time interval between each snapshot and $\arg(\lambda)$ the phase angle of complex eigenvalue λ [63]. As can bee seen, each POD mode is associated with an amplitude and frequency.

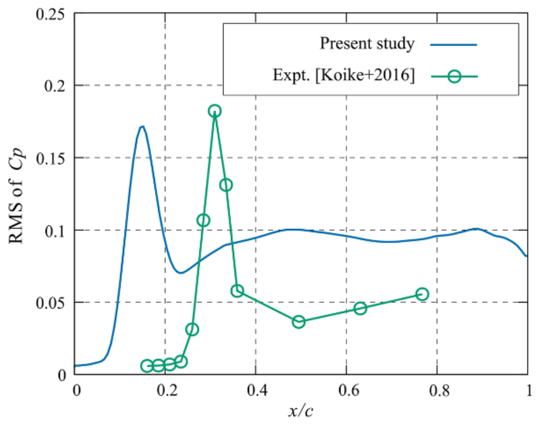
$$\sigma = \frac{\log|\lambda|}{\Delta t} \tag{2.31}$$

$$St = \frac{\arg(\lambda)}{2\pi\Delta t} \tag{2.32}$$

To evaluate the accuracy of this method, its results are presented below. First of all, the ZDES simulation results used to build the POD modes are compared to the experimental results. Figure 2.13a shows the rms of the pressure coefficient fluctuation for each time step. As can be seen, at the position of the shock foot at spanswise positions larger than $\eta \geq 0.4$. there exists large variation in the pressure coefficient. This means that there exists oscillation of the shock wave at these positions. Figure 2.13b shows a comparison with experimental pressure variation data at the spanwise section $\eta = 0.6$. This shows that the numerical result places the shock wave oscillation too far upstream. However, the range and amplitude of the shock wave oscillation is more accurate. Lastly, the vortices induced by the shock wave oscillation seem to propagate further downstream in the numerical simulation. This highlights the importance of accurate high-fidelity data to utilize for the POD. However, exploring methods to obtain more accurate data is outside the scope of this study.



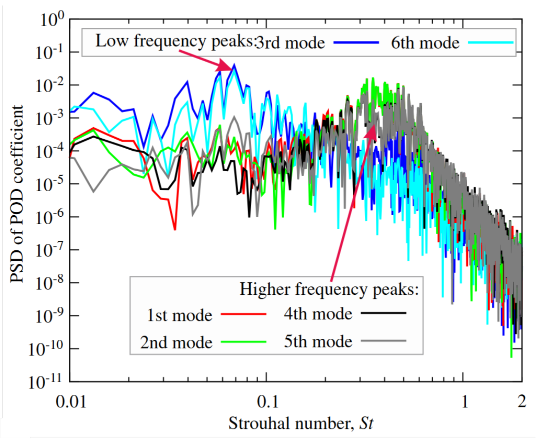
(a) Simulated rms pressure coefficient **variation**



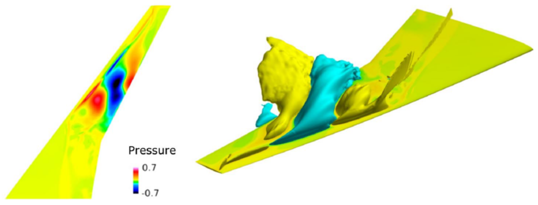
(b) Comparison experiment and simulation of the rms pressure coefficient **variation** at $\eta = 0.6$

Figure 2.13: Pressure coefficient **variation** of NASA CRM swept wing at M=0.85, Re=1.516×10^6, and $\alpha = 4.87$ [63]

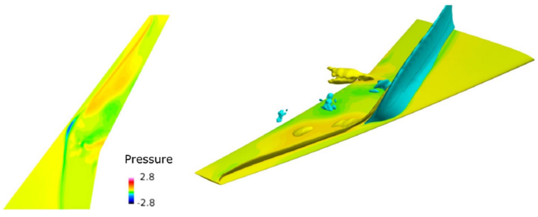
After the high-fidelity was generated by means of the numerical simulation, POD was applied to the data and the first 51 modes were used. Figure 2.14a shows the power spectral density (PSD) spectrum of the POD time coefficients of the first six modes. As can be seen, the 3rd and 6th mode display a peak at St = 0.06 while the 1st, 2nd, 4th, and 5th mode show a peak at St = 0.4. This means that the 3rd and 6th mode cause two-dimensional buffet while the other modes cause three-dimensional buffet [63]. This can also be observed by plotting these modes, as done in Figure 2.14b and Figure 2.14c for the first and third mode respectively. As seen in Figure 2.14b, the first mode is indeed associated with three-dimensional buffet since this mode displays pressure fluctuations in lateral directions. Moreover, the shock wave position is modelled very accurate and also displays pressure fluctuations. The pressure variation from vortices behind the shock wave as observed in Figure 2.13a are also induced by this mode. Figure 2.14c shows the third mode which is responsible for two-dimensional buffet. As seen, the shock itself shows a large amplitude causing pressure variation at the shock location. This indicates that the third mode causes shock position oscillation.



(a) PSD of POD time coefficients



(b) Pressure component of first POD mode



(c) Pressure component of third POD mode

Figure 2.14: PSD of POD coefficients and POD modes [63]

Next to regular POD, DMD can be applied to obtain modes with a constant frequency. Figure 2.15 shows the reconstruction error based on the number of selected DMD modes. As can be seen, the error decreases quickly by selecting more modes and each mode corresponds to a constant frequency. Based on the aforementioned frequency ranges, it is easy to observe that the first mode corresponds to three-dimensional buffet while the second mode corresponds to two-dimensional buffet. Plotting these modes show similar patterns as seen in Figure 2.14b and Figure 2.14c. Another benefit of this method is that, since each mode has a constant frequency, one can plot the phase of each mode. By doing so, the propagation path of fluctuating pressure can be observed by marking the direction that shows a decreasing phase.

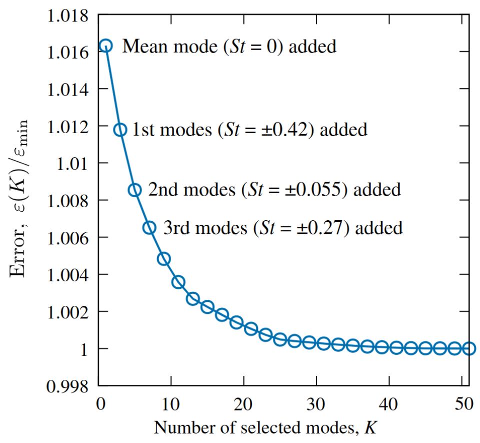


Figure 2.15: MSE for the number of DMD modes [63]

The method presented above shows that the modes capturing frequencies in range St = 0.06-0.07 and St = 0.2-0.6 must be present in order to model transonic buffeting well. By utilizing accurate datasets with sufficient oscillation cycles, these modes are usually part of the first few modes. This means that a ROM based on a truncated POD base is able to accurately capture buffet of swept wings.

3 Initial Model Development: Multidimensional ePOD

In this chapter, the method developed to enrich a two-dimensional transonic flow field is explained. First of all, artificial flow field snapshots were created to serve as controlled datasets for the model development in section 3.1. Moreover, the ePOD efficiency is evaluated for different artificial datasets in order to determine when enrichments are profitable and what the limitations are. Subsequently, in section 3.2, the enrichment methodology is tested on a real dataset, that of the ONERA M6 wing upper surface pressure distribution. The ONERA M6 is a well-known wing often used to validate CFD models. First, the data acquisition is explained. Thereafter, the domain decomposition methodology of the computational field is explained. Lastly, the results are presented and the performance of the model is discussed.

3.1. Two-dimensional ePOD for Artificial Flow Fields

In this section, an artificial flow field is created to develop an enrichment method in two-dimensions, i.e. for a surface pressure distribution. In subsection 3.1.1, the construction of the artificial flow fields is explained. Subsequently, subsection 3.1.2 explains how two-dimensional enrichments are made. Lastly, subsection 3.1.3 explains the performance of the enrichment method for different datasets.

3.1.1. Flow Field Configurations

In this subsection, the method used to create the artificial flow fields is explained. For the flow field domain, an area of 800mm$\times$1200mm with a rectangular grid of size 100×200 was used. The main flow field was chosen to consist of one full sine wave that completes 7 periods in 320 time steps. Moreover, the flow field has a mean and amplitude of -0.2 and 0.3 respectively. In addition, the sine shows a phase shift to the right over the span, with a maximum value of 0.2π at the tip. This main flow field is the same for every dataset.

Subsequently, distinct datasets were created by adding shocks to the main flow. datasets with a different variety in shock positions were created, based on the number of grid points where a shock could be located. To allow for an enrichment range of 100mm on both sides of the shock, 130 out of the 200 nodes in each row of the grid were allowed to contain a shock. So, for the bottom row, this comes down to nodes 25 to 155, which gives a shock position range of 100mm to 620mm. Within this range of possible shock positions, a jumping shock wave was placed based on the percentage of nodes allowed for the shock to be located. This percentage ranged from 2% for only the two outer positions, to 100% for every 130 of the shock nodes. The shock wave consists of a step function with an amplitude of 0.8.

Figure 3.1 shows the first five time steps of a dataset where the number of shock positions is five. As can be observed, the shock travels 130 mm per time step while the sine travels 17.5 mm per time step. Moreover, it can be observed that the pressure coefficient increases very rapidly at the shock position, causing a discontinuity in the flow field.

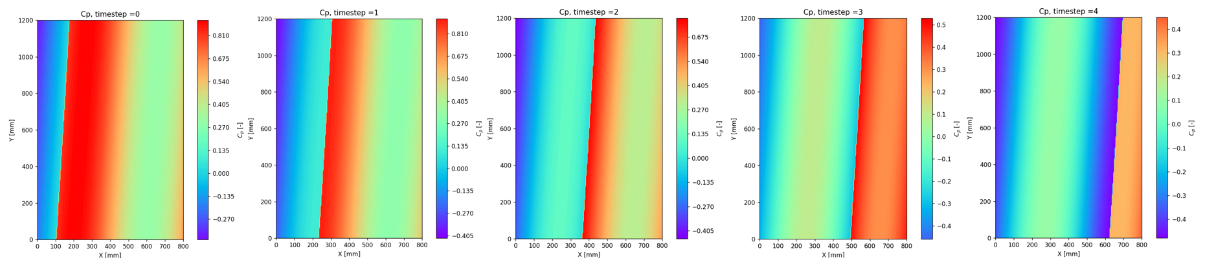


Figure 3.1: Artificial pressure distributions with 5 different shock positions, i.e. $p_{variety} = 4\%$

Table 3.1 summarizes the parameters of the flow field. Here, x_{shock} is the x-position of the shock at every row and $p_{variety}$ is the percentage of shock position variation. In total, six different datasets were created, one for every percentage of shock variety shown in the table. Moreover, it can be observed that the main flow field completes 7 periods during the 320 time steps in the dataset. The number of periods of the discontinuity depends on the number positions, where more positions means less periods.

Table 3.1: Artificial flow field parameters

	Main flow field	**Discontinuity**
Signal	$-0.2 + 0.3\sin(2\pi(\frac{x}{800} - \frac{7}{320}t) - \frac{0.2\pi}{1200} \cdot y)$	$\begin{cases} 0 & \text{if } x < x_{shock} \\ 0.8 & \text{if } x \geq x_{shock} \end{cases}$
$\mathbf{p_{variety}}$ **[%]**	-	[2,14,26,51,75,100]
Period [time steps]	45.71	$(p_{\text{variety}}/100) \cdot 130$
N periods [-]	7	320/period

3.1.2. Constructing Two-dimensional Enrichments

In this section, it is explained how the one-dimensional enrichment method developed in [13] was used to construct two-dimensional enrichments. Here, the artificial flow field datasets from the previous section were used to construct and test the method. The enrichments were first constructed in 1D for both boundary edges of the domain containing a discontinuity. Thereafter, the two 1D enrichments were used to create a surface enrichment by means of linear interpolation between both edges.

First of all, a shock sensor based on the spatial pressure gradients of every flow field node was developed to locate the discontinuities on both outer edges. This was done by making use of the function `numpy.gradient`[1] from the NumPy library in Python. This function calculates the gradient of a node by applying second order central differences, as shown in Equation 3.1

[1]`https://numpy.org/doc/stable/reference/generated/numpy.gradient.html`

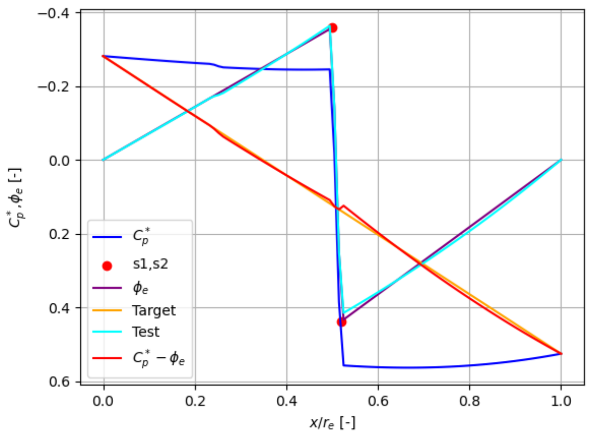
for the gradient in x-direction.

$$\frac{\partial f}{\partial x}\bigg|_{i,j} \approx \frac{f_{i+1,j} - f_{i-1,j}}{2h_x} \tag{3.1}$$

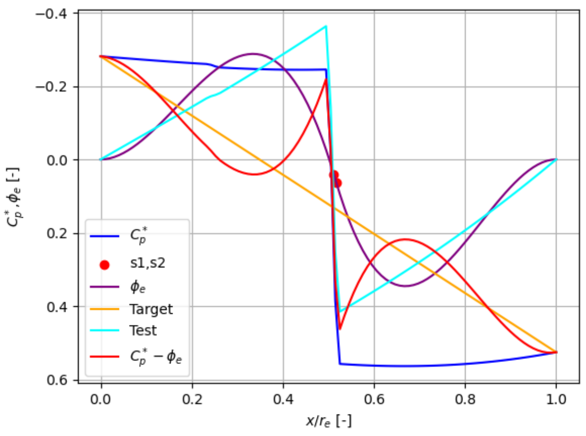
Thereafter, one-dimensional enrichment functions were created. Since POD will be applied on the pressure distribution fluctuation, so after the mean flow field is subtracted, it was needed to subtract the mean flow field first. Thereafter, an enrichment function was fitted to the sectional pressure distribution at the particular edge. An enrichment range of 100mm was used on both sides of the enrichment node for this curve fit. The enrichment function can be a piecewise linear sawtooth in the case of a very sharp discontinuity, or a polynomial in the case of a more gradual discontinuity. Figure 3.2 shows an example of both curve fits. Here, the dark blue line is the initial one-dimensional pressure distribution containing the discontinuity in the centre and 100mm on both sides of it.

To smooth out the discontinuity, a linear line between the pressure coefficient values at the end points of the enrichment range was used as target function. As a result, the Dirichlet boundary condition is met for the enrichment domain, resulting in a smooth transition to outside the enrichment domain. This target function is shown by the yellow line. Moreover, the cyan curve is the difference between the original pressure distribution and the yellow target curve. This means that by subtracting this curve from the original pressure distribution, the target function is made. So, this is the enrichment shape function needed. However, the enrichment should approximate this curve with a low order representation, showed by the purple curve. This low order enrichment function can be defined by two control points only. As a result, each enrichment function has five DOF. One for its position and four for the internal control points defining the amplitude.

The red curve is the manipulated pressure distribution that results from subtracting the low order enrichment function from the initial pressure distribution. As seen in Figure 3.2a, in this case, the piecewise linear sawtooth enrichment function results in a smooth pressure distribution. The piecewise cubic spline on the other hand, is not effective in this case, as seen in Figure 3.2b.



(a) Piecewise linear sawtooth enrichment



(b) Piecewise cubic spline enrichment

Figure 3.2: Difference between two types of enrichment shape functions

The `scipy.optimize.curve_fit`[2] from the SciPy library in Python was used to calculate the two control points that leads to the best fit. To be more precise, this function minimizes the

[2]`https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html`

residuals of the non-linear least-squares fitting of the shape function. Each shape function is defined by the following four control points:

$$\begin{aligned} P &= [(x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3)] \\ &= [(0,0), (s_1, a), (s_2, b), (1,0)] \end{aligned}$$

The boundary points are fixed, which means that only the inner two points are free to optimize. Moreover, Dirichlet's boundary condition is satisfied by setting the end points equal to zero. This ensures that the enrichment does not introduce irregularities where the enrichment domain ends. The optimization of the control points was done by fitting the sawtooth on the pressure distribution as follows:

$$\phi_e(x, p(t)) = \begin{cases} \frac{ax}{s_1}, & \text{for } 0 \le x < s_1 \\ \frac{(b-a)x}{s_2-s_1} + \frac{as_2-bs_1}{s_2-s_1}, & \text{for } s_1 \le x \le s_2 \\ \frac{-bx}{1-s_2} + \frac{b}{1-s_2}, & \text{for } s_2 < x \le 1 \end{cases} \tag{3.2}$$

where p(t) defines the two control points (s1, a) and (s2, b). In some cases however, the cubic spline is the best shape function. In that case, `scipy.interpolate.CubicSpline_fit`[3] from the SciPy library in Python is used. This function creates a smooth cubic spline $S_i(x)$ between every control point $x_i \le x \le x_{i+1}$. Each segment is defined by:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{3.3}$$

where a_i, b_i, c_i, d_i are the spline coefficients for segment i. The function computes these coefficients such that the following conditions are satisfied:

$$\begin{aligned} S_i(x_i) &= y_i \\ S_i(x_{i+1}) &= y_{i+1} \\ S_i'(x_{i+1}) &= S_{i+1}'(x_{i+1}) \\ S_i''(x_{i+1}) &= S_{i+1}''(x_{i+1}) \end{aligned}$$

These conditions ensure that neighbouring segments have the same value, first derivative, and second derivative at their connecting control point. Moreover, the boundary conditions are clamped for the best fit, such that $S'(x_0)$ and $S'(x_n)$ are zero, satisfying Neumann's boundary condition.

Subsequently, the enrichment curves were transformed from one dimension to two dimensions. This was done by connecting the curves constructed on both enrichment nodes by means of linear interpolation. The enrichment shape surface is shown in three-dimensions in Figure 3.3. Since each one-dimensional enrichment function has five DOF, the total number of DOF is ten for the two-dimensional enrichment.

Now, the two-dimensional enrichments were used to modify the pressure distributions. Figure 3.4 shows the initial and modified pressure distribution at time step 10. As observed in Figure 3.4a, the initial flow field consists of a sine wave with sweep angle and a strong shock that causes an increase in pressure at 180mm of the root chord. Since the shock variation is 51% in this example, the shock has not travelled far yet at time step 10.

[3]`https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.CubicSpline.html`

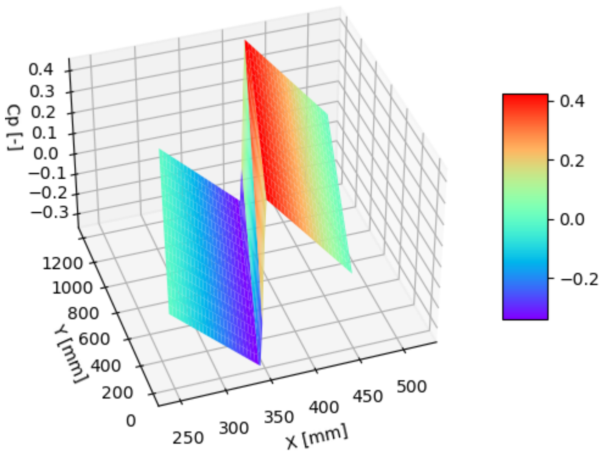
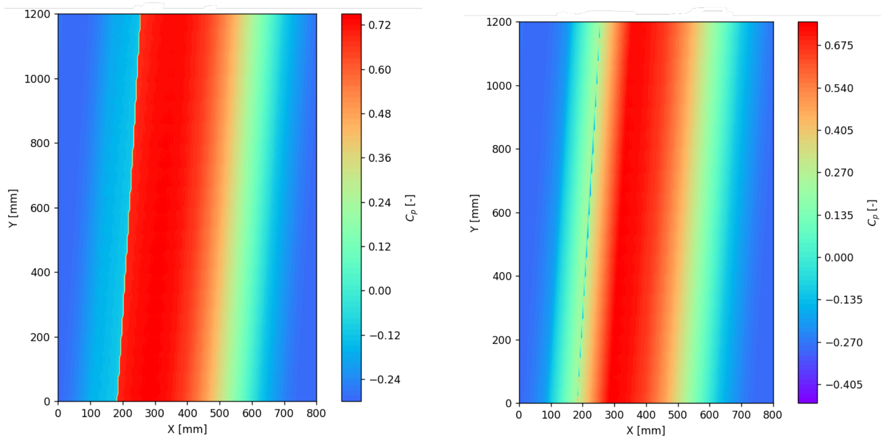


Figure 3.3: Enrichment shape surface in 3D for an arbitrary time step

Figure 3.4b shows the enriched pressure distribution, which was created by subtracting the enrichment surface for time step 10 from the initial pressure distribution fluctuation. As can be seen, the shock has been smoothed out very well. In addition, the maximum pressure coefficient has also decreased, which allows for a shallower slope of the pressure distribution in the enrichment domain. Moreover, the enrichment does not introduce additional irregularities, the transition to the enrichment domain is very smooth.



(a) Initial flow field **(b)** Enriched flow field

Figure 3.4: Artificial pressure distribution fluctuation at time step 10, shock variation 51%

3.1.3. Model Performance

In this section, the performance of the model on the different types of artificial datasets is evaluated. The main measure is the projection, or L_2 error, since the objective is to assess for which kind of flow fields the ePOD method is capable of decreasing this error with less DOF compared to regular POD. To do so, a comparison is made where the shock variation is a metric to quantify the type of flow field.

For each of the artificial flow fields constructed in subsection 3.1.1, the POD modes were retrieved my means of SVD, as explained in subsection 2.1.1. Subsequently, the flow field snapshots were reconstructed with an increasing number of POD modes. Thereafter, the projection error, which can be calculated with the L_2 norm, was calculated for each dataset.

The relative L_2 error for a certain POD reconstruction of the pressure distribution can be calculated with Equation 3.4, where C_p is the true solution, $\hat{C_p}$ the prediction, $\|\cdot\|$ the L_2, or Euclidean, norm, and N the number of nodes.

$$\varepsilon_{L^2} = 100 \cdot \frac{\left\| C_p - \hat{C_p} \right\|_2}{\left\| C_p \right\|_2} = 100 \cdot \frac{\sqrt{\sum_{i=1}^{N} (C_{p_i} - \hat{C_{p_i}})^2}}{\sqrt{\sum_{i=1}^{N} C_{p_i}^2}} \tag{3.4}$$

The result is shown in Figure 3.5, where the solid lines denote the regular POD reconstructions, and the dashed lines the equivalent ePOD reconstructions. Note that the ePOD reconstruction starts at 11 DOFs, which is due to the enrichment which always requires 10 DOFs.

It can be seen that for 2% shock variation, only 4 POD modes are needed to get an L_2 error close to zero. This means that ePOD can never be more efficient than POD for this percentage of shock variation. For 14% however, it can be seen that at 15 DOFs, which are 15 POD modes and 5 ePOD modes, both methods are equally efficient and accurate but ePOD remains more inaccurate for different amounts of DOFs.

However, for 26% shock variation, there exists a region were ePOD is more efficient compared to POD. From 13 to 32 DOFs, ePOD is more efficient with a maximum decrease in L_2 error of 39.0% at 15 DOFs. As the shock variation percentage increases, this range where ePOD is more efficient gets larger and the number of modes required for POD to get more efficient as well. For 51% shock variation, ePOD is more efficient up to a number of DOFs of even 60, with a maximum of 40.9% more accurate than POD. This proves that if the variety of the discontinuity is large enough, ePOD is more efficient than regular POD in terms of projection error.

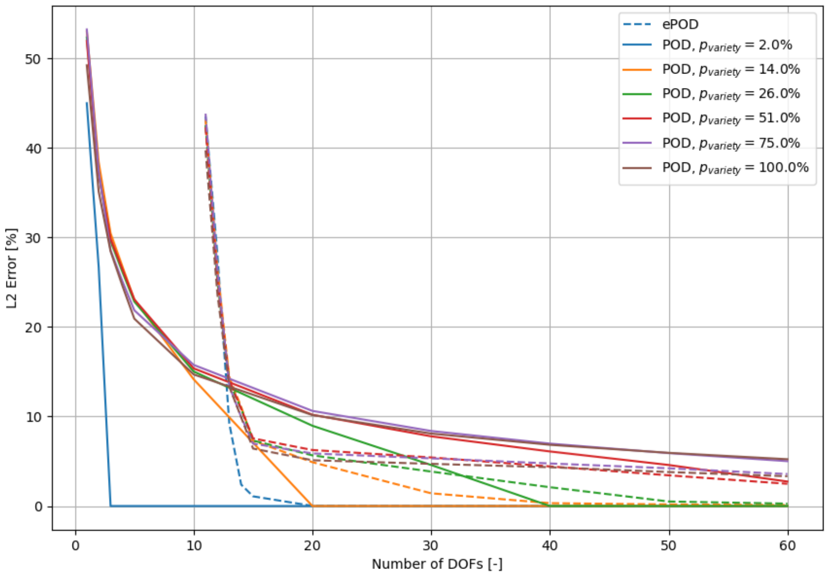


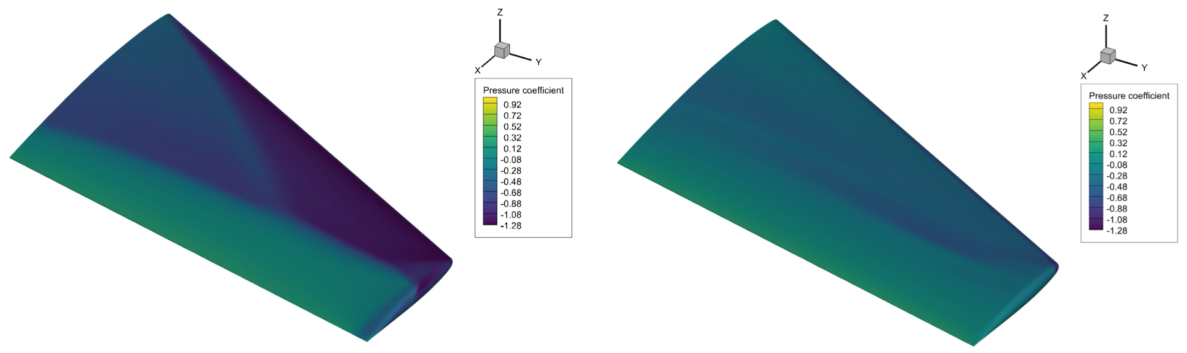
Figure 3.5: L_2 error against number of modes for different flow fields

3.2. Two-dimensional ePOD for the ONERA M6 Wing

In this section, the enrichment method developed for an individual discontinuity will be used to develop an enrichment mode for the ONERA M6 wing, which contains multiple discontinuities. These discontinuities are, however, relatively smooth and show only limited variation in time. In section 3.1, it was already shown that ePOD is beneficial only for strongly varying discontinuities with sharp gradients. Therefore, the method is expected to be less efficient in this case, since POD already achieves a low global L_2 error with relatively few DOF. Instead, this section will focus on evaluating how domain decomposition and different refinement strategies affect the shock-specific L_2 error.

The ONERA M6 wing is often used for CFD validation due to its relatively simple geometry but complex flow behaviour in transonic flow [67]. It is a semi-span wing with a 30.0° leading edge sweep angle and no twist. Moreover, the half-span is 1.1963 m, the MAC is 0.64607 m, the aspect ratio is 3.8 and the taper ratio is 0.562. The airfoil is the ONERA D section which is symmetric and has a thickness-to-chord ratio of 0.10482. High fidelity pressure distributions were generated with RANS, $k-\omega$, steady-state simulations using NLR's CFD solver ENSOLV [68]. The wing was subjected to a sinusoidal change in angle of attack, with a mean and amplitude of 3° and a period of 32 time steps. A total of 10 periods were simulated for a Mach and Reynolds Number of 0.85 and 11×10^6 respectively.

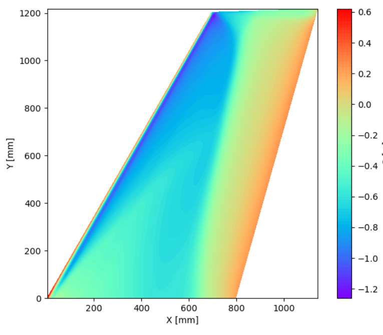
Two snapshots of the wing upper surface are shown in Figure 3.6, one for the maximum angle of attack (Figure 3.6a) and one for the minimum angle of attack (Figure 3.6b). As seen, for the maximum angle of attack, the pressure distribution shows a normal shock wave from 80% root chord to 30% tip chord. Moreover, an oblique shock wave is present originating from the root leading edge and intersecting the normal shock at 60% of the semi-span. Also, a discontinuity at the tip chord stemming from the tip vortex can be observed. For zero angle of attack, however, only a very weak normal shock can be observed.



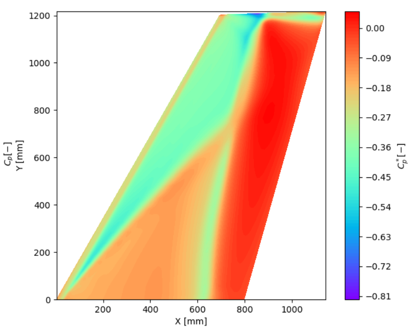
(a) timestep = 8, $\alpha = 6.0$ **(b)** timestep = 24, $\alpha = 0.0$

Figure 3.6: ONERA M6 pressure distributions obtained with ENSOLV, M=0.85

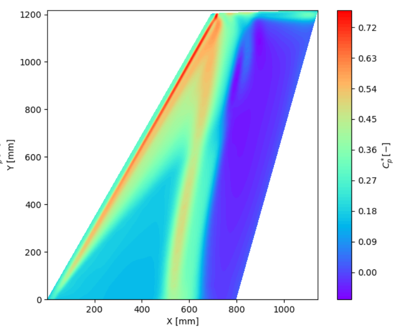
Figure 3.7 shows two-dimensional pressure distributions of the wing upper surface. Since the POD will be performed on the pressure distribution minus the time-averaged pressure distribution, i.e. the pressure fluctuation distribution, the time-averaged pressure distribution was calculated first. The result is shown in Figure 3.7a. As seen, the normal shock position does not vary very much over the full timescale, the oblique shock does however. Figure 3.7b shows the pressure fluctuation distribution for the maximum angle of attack. As illustrated, the normal and oblique shock are clearly present, just like the tip vortex. For zero angle of attack, the shocks are weaker and there is a drop in pressure due to the average as observed in Figure 3.7c.



(a) Time-averaged pressure distribution over all snapshots



(b) Pressure fluctuation distribution, timestep = 8, α = 6.0



(c) Pressure fluctuation distribution, timestep = 24, α = 0.0

Figure 3.7: Two-dimensional ONERA M6 pressure distributions

3.2.1. Domain Decomposition

Since the enrichment method in section 3.1 was developed for single linear discontinuities, the flow field was divided into subdomains that each isolate one discontinuity. A triangulation was defined based on the detected shock and vortex intersections, as shown in Figure 3.8. This allows for the same enrichment methodology per discontinuity. However, while this makes the method scalable to more complex flow fields, it also increases the number of DOF.

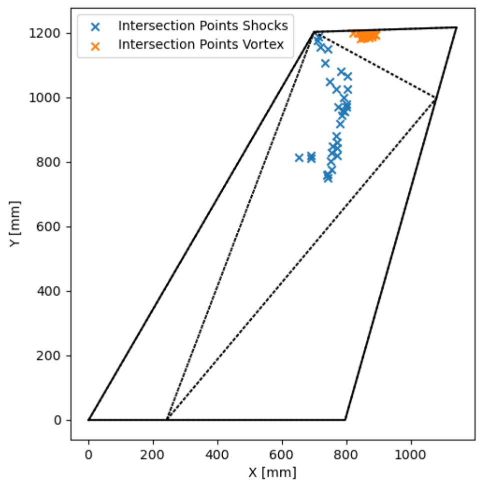
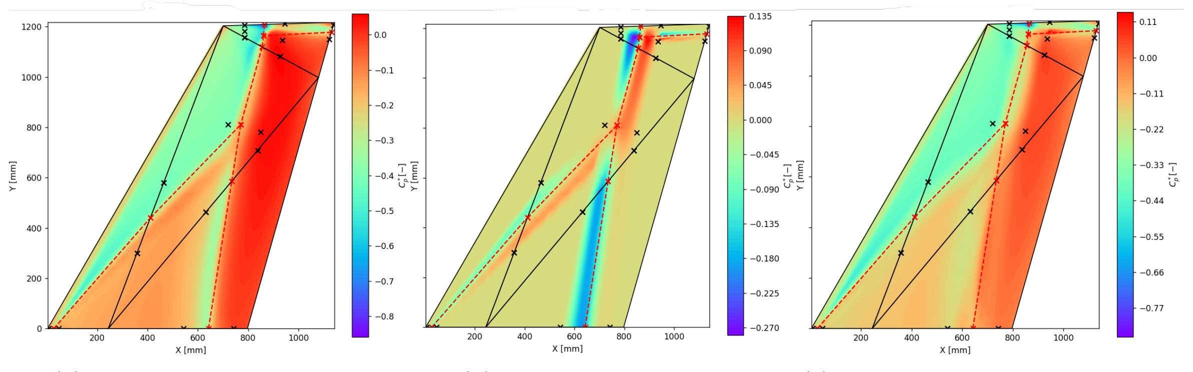


Figure 3.8: Intersections points of shocks and vortexes, and resulting domain decomposition

Within each subdomain, enrichment functions were defined along the boundary intersection points. For subdomains with intersecting discontinuities, an additional node was placed at the intersection point. Moreover, the enrichment functions were implemented using cubic splines, as the discontinuities in this dataset are relatively smooth. Figure 3.9 shows how the enrichments of all subdomains are combined into one mode stretching the entire domain. The red and black crosses define the enrichment nodes and range respectively. As shown in Figure 3.9b, the entire enrichment mode covers both straight-line and Y-shaped discontinuities. Figure 3.9c shows the manipulated pressure distribution, which is the result of the initial pressure distribution minus the enrichment mode.



(a) Initial pressure distribution **(b)** Entire enrichment mode **(c)** Manipulated pressure distribution

Figure 3.9: ONERA M6 pressure distribution enrichment process at time step 10

3.2.2. Results and Performance Evaluation

This section evaluates how domain decomposition and different refinement strategies affect the ePOD modes and shock-specific L_2 error. To investigate how a decrease in amount of DOF influences the shock accuracy, the following refinement strategies were tested:

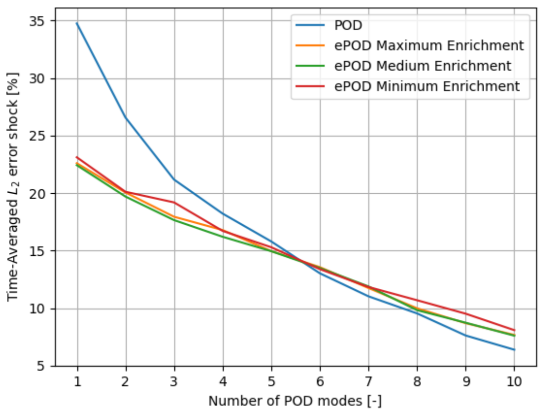
- **H-refinement -** 'H', element size. Subdividing triangles to obtain more subdomains
- **P-refinement -** 'P', polynomial degree. Adding parameters to the enrichment functions

As a first analysis, a sensitivity analysis of the L_2 error in the shock domain was made based on H-refinement. The following configurations were created:

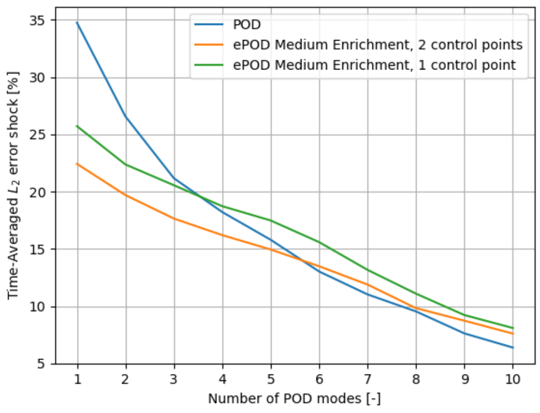
- **Maximum enrichment -** two Y-shaped enrichments, shock intersection and tip vortex
- **Medium enrichment -** one Y-shaped enrichment, shock intersection
- **Minimum enrichment -** no Y-shaped enrichments, only normal shock enriched

The enrichment configuration as shown in Figure 3.9 is referred to as maximum enrichment as it captures all discontinuities. The first 8 POD and ePOD modes of maximum enrichment used to reconstruct the pressure distributions are shown in Figure B.1 and Figure B.2, respectively. It is very clear that the dominant shock features present in the POD modes appear less dominantly in the ePOD modes. Moreover, it can be observed that the energy content of every mode decays rapidly. This is confirmed by Figure B.3, where it can be seen that the first POD modes captures already more than 80% of the total energy.

The shock L_2 error as a result of H-refinement is shown in Figure 3.10a. It can be observed that the medium enrichment results in the lowest error compared to the other enrichments, however the difference is very minor. The minimum enrichment results in the highest shock error, as in this configuration the oblique shock is not enriched. It can also be observed that even for a shock-based L_2 error, POD becomes already more accurate than all enrichment configurations at five POD modes. In addition to H-refinement, a P-refinement was performed. Since the lowest L_2 error was achieved for the medium enrichment, this analysis was made for this enrichment grid only. For this analysis, a comparison between enrichment shape functions with two control points and one control point was made, which decreases the amount of DOF from five to three for every node. For the medium enrichment, this results in a decrease from 36 to 22 DOF, a decrease of 38.9%. However, as seen in Figure 3.10b, this refinement results in a projection error that is only between 6.0% and 12.8% higher.



(a) H-refinement influence on shock L_2 error



(b) P-refinement influence on shock L_2 error

Figure 3.10: Shock L_2 error against number of POD modes

Figure 3.10 confirms the hypothesis stated at the start of this section. Due to the relatively smooth and constant character of the flow field, enrichment is not effective in decreasing the number of DOF while maintaining the same accuracy, even in a shock-based norm. This chapter showed however, that the enrichment method can be applied to real wing pressure distributions, even if it contains multiple discontinuities, making POD capture the the resulting flow field more effectively. This is shown in Figure 3.11, where the reconstructed pressure distribution with medium enrichment and two control points per node with only one POD mode is shown. It can be observed that after enrichment, one POD mode allows for a very accurate shock reconstruction. Similar figures showing the reconstruction for more POD modes at the same time step are shown in Figure B.4.

To conclude, these findings highlight the practical limits of enrichment for smooth flows. They also demonstrate that it provides a systematic way to construct a dynamic mode capable of capturing varying discontinuities with a limited number of DOF, potentially becoming beneficial when discontinuities vary strongly.

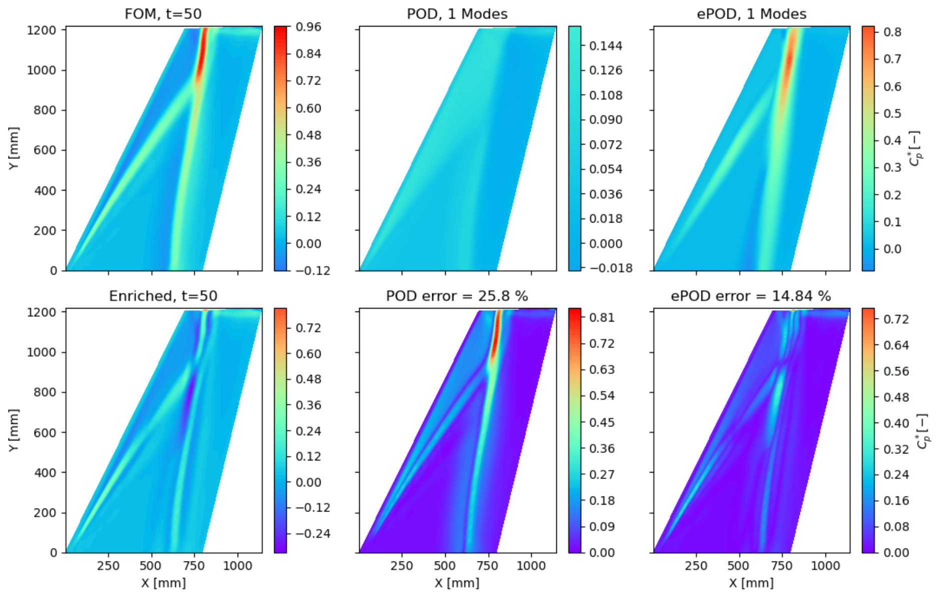


Figure 3.11: Reconstruction at time step 50 with one (e)POD mode. TL=initial flow field, BL=enriched flow field, TC=POD reconstructed, TR=ePOD reconstructed, BC=POD error, BR=ePOD error.

4 Integrated Enrichment Design: Enhancing Predictability

In specific conditions, the enrichment methodology developed in chapter 3 demonstrates higher projection accuracy with fewer DOFs compared to regular POD, as shown in section 3.1. However, in the (e)POD-LSTM ROM, an additional error exists, which is the neural network (NN) regression error. This means that an increased accuracy with a fixed number of DOF can be achieved by considering this error early on in the enrichment design process. Whereas regular POD optimizes for L2 error, enrichments can incorporate predictability of its parameters, reducing the NN regression error significantly. This integrated approach leads to a lower total ROM error, enhancing the accuracy in unseen scenarios.

Time coefficients for higher order POD modes are very irregular over time, hence difficult to predict. When these modes are replaced by enrichment modes that are more suitable for regression, a similar number of DOFs should results in a lower NN regression error. This means that the projection error could be higher than for POD, while the total ROM error is lower. In this chapter, the strategy to design these enrichments as well as an evaluation of the results is presented.

4.1. Quantifying Intrinsic Predictability

In this section, a method to quantify the intrinsic predictability of datasets is found. To anticipate how well the NN can learn the mapping between input and output parameters, a low-cost surrogate model that estimates this ability was used. By doing so, the predictability of the enrichment parameters can be evaluated without the pain of training a NN at every iteration. The input signal together with the first 50 POD time coefficients of the Onera M6 case were used to find the best surrogate. First, different intrinsic predictability scores were calculated for each time coefficient. Thereafter, the realized predictability of these coefficients was determined by calculating the MSE of the predictions of a trained LSTM-NN. This was used to determine which intrinsic predictability measure is the most accurate.

Previous studies for time series have proved that relationships exist between certain intrinsic predictability measures and realized predictability of specific forecasting models. In [69], the LSTM forecasting with RMSE for realized predictability showed a good relationship with SVD entropy for the intrinsic predicability. The equation for SVD entropy is shown in Equation 4.1. Here, x is the time series, $\overline{\sigma}_j^2$ are the normalized singular values of embedded matrix Y created with slices of x, and N is the number of singular values [70].

$$H(X) = \sum_{j=1}^{N} \overline{\sigma}_j \log(\overline{\sigma}_j^2) \tag{4.1}$$

A large SVD entropy means that the data is complex, and there are many modes needed to describe it, hence difficult to predict. SVD entropy, however, is not very suitable for input-output regression since it measures the dimensionality of the one-dimensional time series. To find the best surrogate for the input-output regression, the relationship between other intrinsic predictability measures and RMSE of the LSTM-NN is assessed.

The first measure tested is the Pearson Coefficient, which serves as a relatively simple estimate for linear correlation between two parameters [71]. The equation is given in Equation 4.2.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4.2}$$

Moreover, the Spearman Coefficient which is less strict towards non-linear data was also used. This measure is based on the rank of the values and assigns high scores to monotonic relations, which can also be non-linear, thereby being less sensitive to extreme values. The equation is given in Equation 4.3[71]. Here, a_i and b_i are the rank of x_i and y_i respectively.

$$r_s = 1 - \frac{6\sum_{i=1}^{n}(a_i - b_i)^2}{n(n^2 - 1)} \tag{4.3}$$

Another good method is that of mutual information (MI), which is widely used in the medical world for feature selection [20][21]. In feature selection, the input variables that influence the outcome the most are selected to be used from a dataset. MI assigns a score to a set of two variables, which indicates the dependence of one of the variables on the other variable. A high score means that one of the variables gives a lot of information about the other variable.

The equation for MI is given in Equation 4.4 [21], where H(X) is the entropy of input vector X and $H(X \mid Y)$ the conditional entropy of input vector X in presence of output vector Y.

$$I(X;Y) = H(X) - H(X \mid Y) \tag{4.4}$$

The entropy of X is defined by Equation 4.5, where $p(x_i)$ is the probability mass function of discrete random variable x_i, which gives the probability that this variable is equal to some value. Moreover, the conditional entropy of X in presence of Y is given by Equation 4.6, where $p(x_i, y_j)$ is the joint probability of x_i and y_j.

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \tag{4.5}$$

$$H(X \mid Y) = -\sum_{i=1}^{n}\sum_{j=1}^{n} p(x_i, y_j) \log \frac{p(x_i)}{p(x_i, y_j)} \tag{4.6}$$

The MI scores were normalized to be able to compare the MI scores of different parameters, which is done by dividing the score with the square root the product of both entropies, as shown in Equation 4.7.

$$NMI(X;Y) = \frac{I(X;Y)}{\sqrt{H(X)H(Y)}} \tag{4.7}$$

To determine the realized predictability, measured by the RMSE of the predicted time coefficients, the Long Short-Term Memory Neural Network (LSTM-NN) used in the ROM was used. An LSTM-NN is a type of Recurrent Neural Network (RNN) which employs forget, input, and output gates that maintains both long- and short-term dependencies without exploding or vanishing gradients. More details on the technicalities of LSTM cells are given in Appendix A.

Previous studies at NLR have investigated the performance of different LSTM-NN architectures for the regression of (e)POD time coefficients [11][12]. The results of these studies were used to construct an LSTM-NN with similar hyperparameters. When tuning these parameters, a trade-off exists between prediction accuracy and cost. However, the risk of under- and overfitting is also an important consideration. Making the network very large leads to better accuracy on the training data, but requiring more computational power as a consequence. Moreover, this increases the risk of overfitting. A small network on the other hand, leads to less computational power but also less accuracy. By understanding the meaning of the hyperparameters, an efficient and accurate network can be made. The following hyperparameters can be changed depending on the use case:

- **Number of LSTM layers -** The number of stacked LSTM sequences in the network, where the length of each sequence is equal to the number of time steps considered. Each LSTM layer has an attached drop-out layer
- **Number LSTM units per cell -** The memory of each LSTM cell, also called the hidden size. Determines the complexity of relations that can be made, equivalent to the number of neurons in a DNN.
- **LSTM activation -** The type of non-linearity introduced in LSTM layers.
- **Number of dense layers -** The number of layers with regular nodes used after the LSTM layers.
- **Number of dense units per layer -** The number of nodes used in a dense layer.
- **Dense activation -** The type of non-linearity introduced in the dense layers.
- **Drop-out rate -** The percentage of randomly de-activated nodes in each training iteration. Used to make the network robust and not rely on individual nodes.
- **Batch size -** The number of training sets used in each training iteration.
- **Time steps -** The number of previous time steps considered by the network in predicting the output at the current time step.
- **Optimizer -** The algorithm used to update weights and biases during training.
- **Loss function -** The function used to determine the prediction error during training, which should be minimized.
- **Epochs -** The number of passes trough the entire training data during training.

The initial hyperparameters used for the network are summarized in Table 4.1. As can be seen, the network has two LSTM layers and one dense layer, both with 128 units. This means that the network is quite complicated. Moreover, the drop-out rate is 20% which is common for datasets of this size. Lastly, the number of time steps considered by the LSTM is eight, and the batch size 32. For a detailed discussion on the impact of changing these parameters, one could consult a previous study conducted at NLR [11].

Table 4.1: LSTM-NN Hyperparameters

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Number of LSTM layers	2	Number LSTM units per cell	128
Number of dense layers	1	Number of dense units per layer	128
LSTM activation	tanh	Dense activation	tanh
Drop-out rate	0.2	Batch size	32
Time steps	8	Optimizer	ADAM
Loss function	mse	Epochs	2000

The input of the LSTM-NN is the motion of the Onera M6 wing as described in section 3.2. Since this motion can be described by the angle of attack, its first derivative, and its second derivative, the input consists of a vector with three features. Since the number of time steps considered is eight, the total input is a matrix of shape (8,3). Moreover, since the batch size is 32, the input matrix for every iteration is (32,8,3). To get a good understanding of the network, a visualization of the network architecture is provided in Figure 4.1.

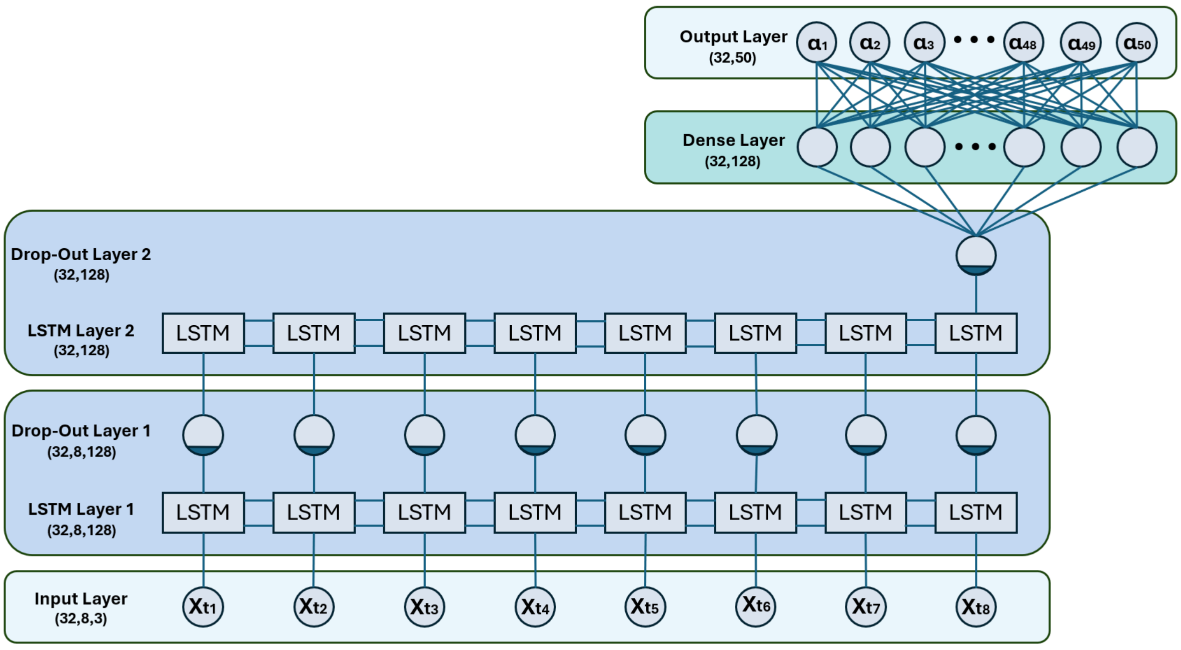
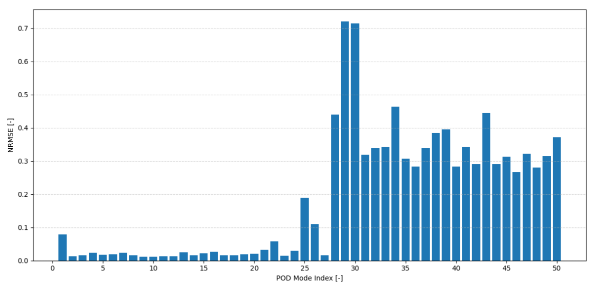
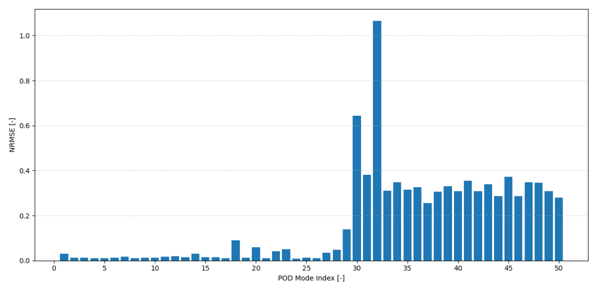

Output Layer (32,50)
α_1 α_2 α_3 α_{48} α_{49} α_{50}
Dense Layer (32,128)
Drop-Out Layer 2 (32,128)
LSTM Layer 2 (32,128)
LSTM
Drop-Out Layer 1 (32,8,128)
LSTM Layer 1 (32,8,128)
Input Layer (32,8,3)
X_{t1} X_{t2} X_{t3} X_{t4} X_{t5} X_{t6} X_{t7} X_{t8}

Figure 4.1: LSTM-NN architecture employed in the ROM for time coefficient prediction

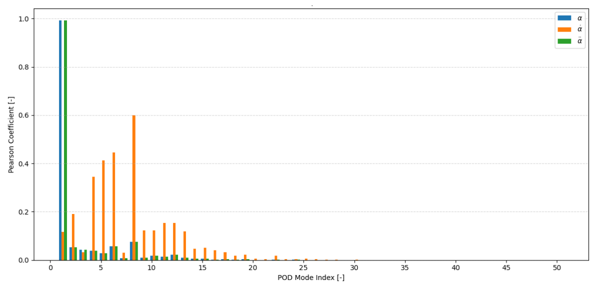
The results of the intrinsic and realized predictability are shown in Figure 4.2. The figures on the left hand side are for the first 50 POD time coefficients, while the figures on the right hand side are for the ePOD time coefficients, so after the enrichments were subtracted. As can be observed in Figure 4.2a, the normalized root mean square error (NRMSE) of the LSTM-NN prediction gets very high from POD mode 28 onwards. For ePOD, this is the case for ePOD mode 30 and higher, as seen in Figure 4.2b. Figure 4.2c-Figure 4.2j show the intrinsic predictability scores for the different measures. What stands out is that the Pearson Coefficient is detecting some correlation between the first derivative of alpha and the first 20



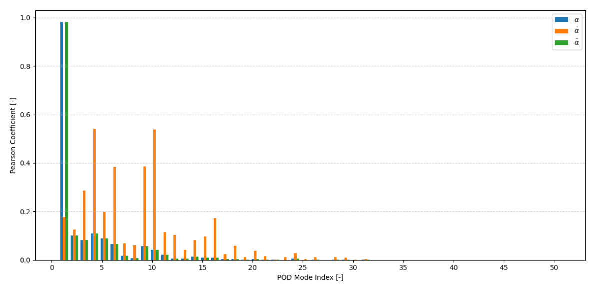
(a) NRMSE of first 50 POD modes predicted by the LSTM-NN



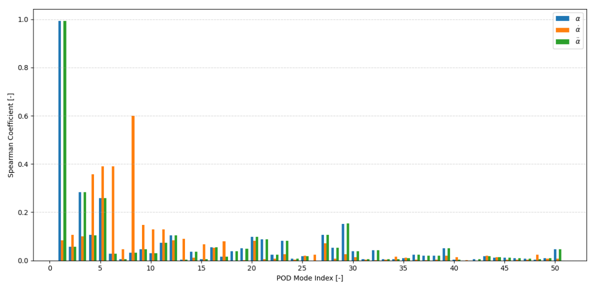
(b) NRMSE of first 50 ePOD modes predicted by the LSTM-NN



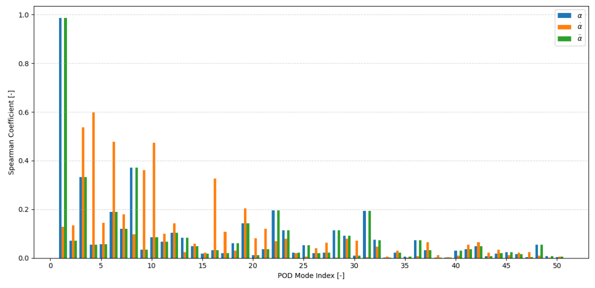
(c) Pearson correlation between input variables and first 50 POD modes



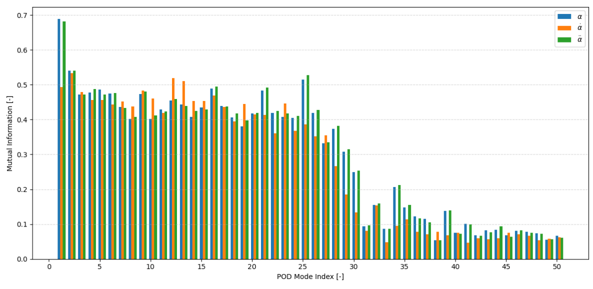
(d) Pearson correlation between input variables and first 50 ePOD modes



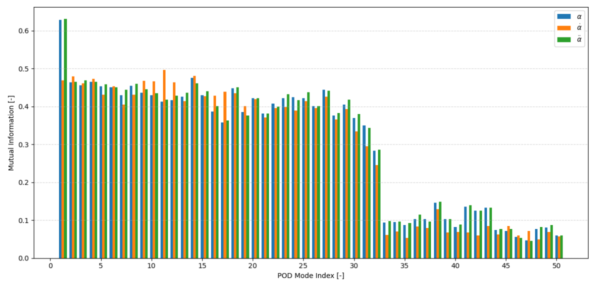
(e) Spearman correlation between input variables and first 50 POD modes



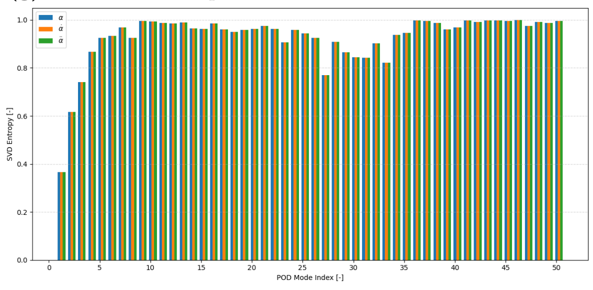
(f) Spearman correlation between input variables and first 50 ePOD modes



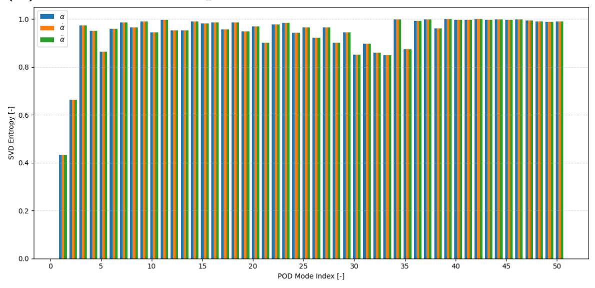
(g) NMI between input variables and first 50 POD modes



(h) NMI between input variables and first 50 ePOD modes



(i) SVD entropy between input variables and first 50 POD modes



(j) SVD entropy between input variables and first 50 ePOD modes

Figure 4.2: Comparison of LSTM-NN prediction NRMSE with different input-output regression scoring methods

time coefficients. However, for higher coefficients, there is almost no intrinsic predictability. The Spearman Coefficient on the other hand picks up correlation up until time coefficient 50. However, the correlation is very low and the increase in realized predictability is not clearly visible. The mutual information however resembles the realized predictability very well. It is clearly visible that before the 30th time coefficient, the mutual information between al input features is high. After the 30th time coefficient, there is a drop in mutual information, just as was observed in the realized predictability. Lastly, SVD entropy is not a good predictability measure in this use case. It can be observed that the scores are high for almost all time coefficients. As a result, mutual information was selected as the intrinsic predictability measure to act as a surrogate model for the LSTM-NN predictability.

4.2. Integrated Enrichment Strategy

In this section, the strategy to design enrichments that have strong predictive properties while not increasing the projection error is explained. An artificial dataset similar to the dataset used in section 3.1 will be used to test the method and assess the change in total error. As an addition, input variables will be used to construct the pressure distribution.

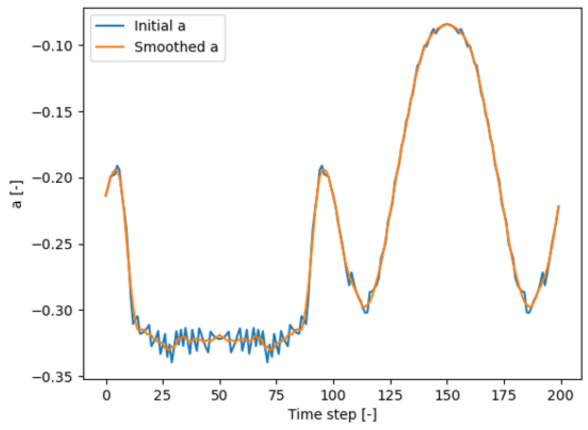
Table 4.2 shows the equations used to construct the pressure distribution dataset. The angle of attack α, varies with a sinusoidal motion that has a period of 200 time steps, and an amplitude and mean of three degrees. Subsequently, the first and second derivative of the angle of attack are straightforward to compute and utilized in the optimization as well. The main flow field is constructed by using a sinusoidal signal that varies with angle of attack, which ensures that the main flow field completes a full period of variation for the full range of angles of attack. Moreover, the sinusoidal completes one full period over the chord length and experiences a phase shift which is maximum 0.2π at the tip. The shock position varies between 500 mm and 200 mm at the root for the minimum and maximum angle of attack respectively, but follows the same angle as the main flow field towards the tip. Lastly, the strength of the shock wave increases from 0.2 to 0.8 for the minimum and maximum angle of attack respectively.

Table 4.2: Artificial flow field parameters

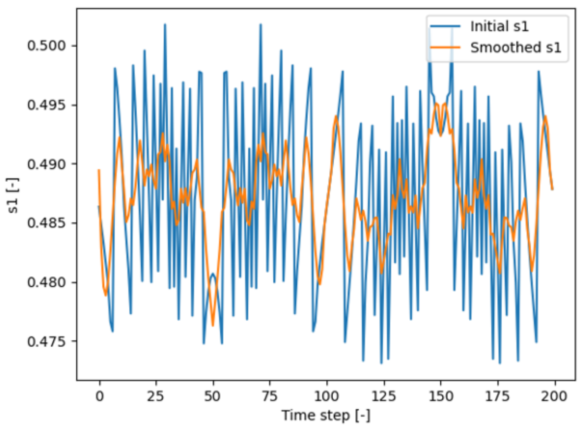
Variable	**Equation**
Flow field dimensions	$(L_{chord} \times L_{span})$=$(800mm \times 1200mm)$
Grid size	grid$(X \times Y)$=(160×100)
Time steps	$N_{timesteps}$=1000
Angle of attack	$\alpha(t)$=$3 + 3\sin(\frac{2\pi t}{200})$
Angle of attack derivative	$\dot{\alpha}(t)$=$3(\frac{2\pi}{200})\cos(\frac{2\pi t}{200})$
Angle of attack 2nd derivative	$\ddot{\alpha}(t)$=$-3(\frac{2\pi}{200})^2\sin(\frac{2\pi t}{200})$
Main flow field	$C_p(\alpha, x, y) = -0.2 - 0.3\sin(2\pi(\frac{x}{800} - \frac{\alpha}{6}) - \frac{0.2\pi}{1200} \cdot y)$
Shock position	$x_{shock}(\alpha, x, y) = 200 + 300 \cdot \frac{6-\alpha}{6} + 120\sin(\frac{0.2\pi}{1200} \cdot y)$
Shock strength	$\Delta C_p(\alpha, x_{shock}) = \begin{cases} 0 & \text{if } x < x_{shock} \\ 0.2 + 0.1\alpha & \text{if } x \geq x_{shock} \end{cases}$

To create enrichments with the intrinsic predictably taken into account, a two step process is needed. First of all, the regular enrichment process from section 3.1 is followed to calculate the enrichments that result in the best approach to the linear target function in the pressure distribution. Secondly, these enrichments are adjusted through an optimization that enhances the intrinsic predictability of each parameter without allowing the projection error to exceed a predefined threshold. This two step process is needed, because to enhance the intrinsic predictability of each enrichment parameter, the relation with neighbouring time steps is needed. This means that the initial course of each enrichment parameter over the whole time span is required. Moreover, since the regular enrichment process results in the best approach to the linear target function, this is a good starting point to obtain a low projection error. Creating enrichments with good intrinsic predictability but very different from these initial parameters will likely result in very ill filtering of the discontinuities, so also in large projection errors. The second step of this integrated enrichment design process is explained below.

To increase the predictability of an enrichment parameter, its value over time must be smooth and its trajectory should be similar to one of the input variables. To adjust the initial enrichment parameters accordingly, the Savitzky–Golay (Savgol) filter [72] is applied first for smoothing. This filter works by means of convolution, which fits a low order polynomial to small sub-sets of the data. In this case, a window with the size of eight data points was used to fit a third degree polynomial. Figure 4.3 shows the manipulated time coefficients of two enrichment parameters. As can be observed, high frequency noise is filtered out and the result is a much smoother curve.



(a) Filtered enrichment time coefficient a



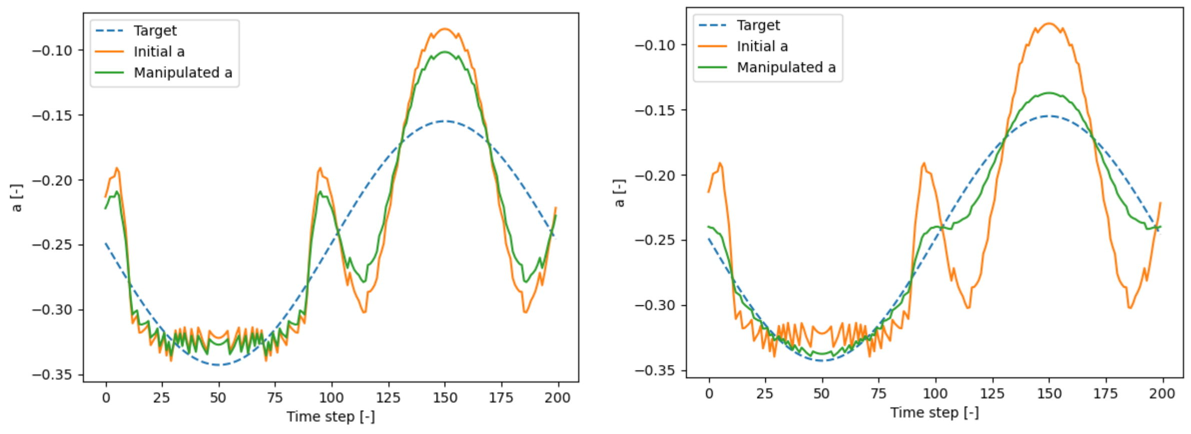
(b) Filtered enrichment time coefficient s1

Figure 4.3: Savitzky–Golay filter with W=8 and p=3 applied to enrichment time coefficients

Subsequently, a target function which has similar characteristics as the input variables was identified. This was achieved by fitting the time series of each input variable to each enrichment parameter with least squares and selecting the variable with the best fit for each enrichment parameter. Subsequently, the difference between both curves at each time step was evaluated and a fraction K of this difference was added to the original parameter curve, as described by Equation 4.8, where $\mathcal{T}_t$ is the target function. The parameter K is to be optimized for each enrichment parameter in order to obtain the best trade-off between predictability and projection error.

$$f_{t,manipulated} = f_{t,initial} + K \cdot (\mathcal{T}_t - f_{t,initial}) \tag{4.8}$$

Figure 4.4 shows the effect of different magnitudes of K on enrichment parameter a, which is the amplitude of the first control point. The target function is the angle of attack, fitted by means of least squares to the curve. As shown in Figure 4.4a, a small K value of 0.25 results in a slightly altered curve. As calculated with Equation 4.7, the normalized mutual information (NMI) between the manipulated a and the angle of attack is 0.64. However, as seen in Figure 4.4b, a higher K value of 0.75 results in a very smooth curve that resembles the input variable very well. This in turn, leads to a NMI score of 0.83, showing that the predictability has increased substantially. This illustrates that K can be used to tune the NMI, and an optimization can be made.



(a) Manipulated a with K=0.25, NMI=0.64 **(b)** Manipulated a with K=0.75, NMI=0.83

Figure 4.4: Increasing the MI for enrichment parameter a

Since the ePOD-LSTM-ROM error consists of the projection error and the neural network error, both should be accounted for in designing the enrichments. The L_2 error accounts for the projection error, while the aforementioned NMI accounts for the neural network error. Since it is not possible to minimize both errors independently, they are included in a norm to be minimized, as shown in Equation 4.9. Here, W is a weighting factor used to adjust the relative importance of both errors in the norm. Moreover, N is the number of parameters included in the ePOD-LSTM-ROM including the ePOD time coefficients.

$$\text{Norm} = W \cdot \epsilon_{L2} + \left(1 - \frac{\sum_{i=1}^{N} \text{NMI}}{N}\right) \tag{4.9}$$

The procedure is summarized in Algorithm 2. In line 2, the initial enrichment parameters are computed from snapshot matrix Q of the flow field. Thereafter, the input parameter that corresponds the best is fitted to the time series of each enrichment parameter. After K is initialized, the enrichment parameters are updated and the first N POD modes are computed. With these POD modes, the average L_2 error can be computed. Subsequently, the NMI for all coefficients can be computed. Lastly, the norm is computed until convergence.

4.3. Results of the Improved Enrichment Method

In this section, the results of the enrichment method explained in the previous section are provided. First the effect on the shape of the enrichment is explained. Thereafter, the changed plots of the coefficients over time and the effect on the predictability are presented. Lastly, the total error composed of the projection and neural network error is compared with the initial enrichments and regular POD.

The integrated enrichment method as summarized in algorithm 2 was used to create improved enrichments for the artificial dataset. Since the shock position is directly dependent on the angle of attack, this parameter was not used in the optimization process. Moreover, since the flow field is very similar at the root and tip chord, the same set of K values is used at both edges. This means that the optimization can be simplified from eight enrichment parameters to four, each with its associated K value.

Algorithm 2 Integrated Enrichment Design Algorithm

```
1: procedure EnrichmentOptimization(initial_enrichment_design, input_parameters)
2:     $P \leftarrow$ INITIAL_ENRICHMENT_DESIGN(Q)                    ▷ Initialize enrichment matrix
3:     for all enrichment parameters $p_i(t) \in P$ do
4:         for all input parameters $x_j(t)(t) \in X$ do
5:             Filter $p_i(t)$ using Savgol filter
6:             Fit $x_j(t)$ to $p_i(t)$ using least squares
7:         end for
8:         Select $x_j(t)$ with best fit for $p_i(t)$
9:     end for
10:    Initialize $K^{(0)} = [K_1^{(0)}, K_2^{(0)}, \ldots, K_N^{(0)}]$          ▷ Initial guess for K
11:    while stopping criteria not met do
12:        for all enrichment parameters $p_i(t) \in P$ do
13:            Calculate difference between $x_j(t)$ and $p_i(t)$
14:            Update enrichment parameter $p_i(t)$ with current $K_i$
15:        end for
16:        Compute enrichments
17:        Subtract enrichments from flow field
18:        Compute POD modes of the flow field
19:        Compute average $L_2$ error using first N modes
20:        for all enrichment parameters $p_i(t) \in P$ do
21:            Compute NMI between $p_i(t)$ and $x_j(t)$
22:        end for
23:        for all POD time coefficients $a_i(t) \in A$ do
24:            for all input parameters $x_j(t) \in X$ do
25:                Compute NMI between $a_i(t)$ and $x_j(t)$
26:            end for
27:            Select maximum NMI
28:        end for
29:        Calculate norm combining projection error and NMI
30:        Use scipy.optimize.minimize to minimize the norm by adjusting $K^{(i)}$
31:    end while
32:    return $K = [K_1, K_2, \ldots, K_N]$
33: end procedure
```

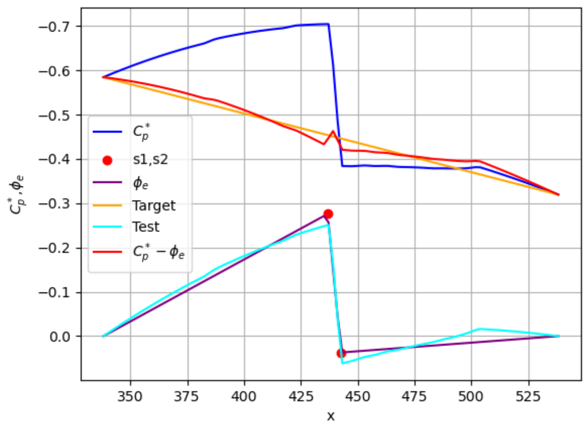
The weight factor in the norm to be minimized was set to 100, which means that a NMI increase of 0.1 is allowed to result in a maximum increase of 0.1% in projection error. To calculate the projection error, five POD modes were used to reconstruct the pressure distribution. Subsequently, the `scipy.optimize.minimize`[1] package from the SciPy library in Python was used to find the values for K that result in the lowest norm by making use of a gradient-descent search. The following values were found, where the order of associated enrichment parameters is a, b, s1, and s2:

$$K = [0.99799248, 0.95361772, 0.99973638, 0.99667716]$$

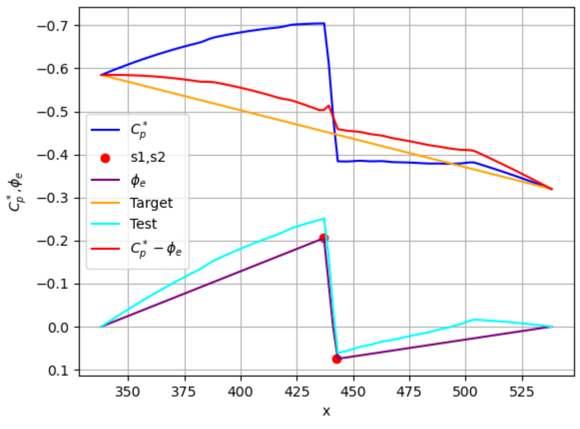
As can be observed, the enrichment control points parameters are modified substantially. The effect of these changes is illustrated in Figure 4.5 for an arbitrary time step. As can be observed, the modified enrichment shape has control points with larger amplitudes, resulting in

[1]`https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html`

a pressure distribution that is approaching the linear target function less accurately. However, the discontinuity is still filtered out very well, and no additional discontinuities are introduced. This indicates that enrichment shapes that are not optimal in L_2 sense can still be very good filters. This illustrates that there is some freedom around constructing the enrichments in favour of predictability. Plots of the enrichment parameters values over time are provided in Figure B.5, where it can be observed that these curves became less irregular and noisy.



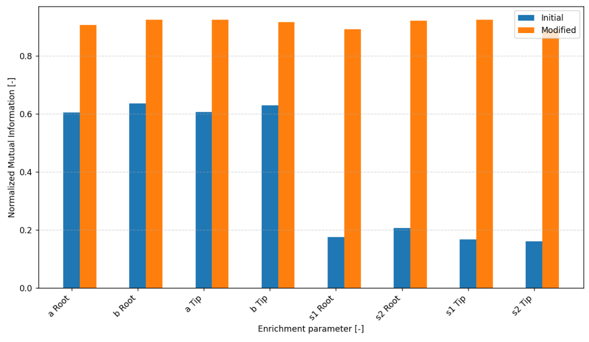
(a) Initial enrichment fitting



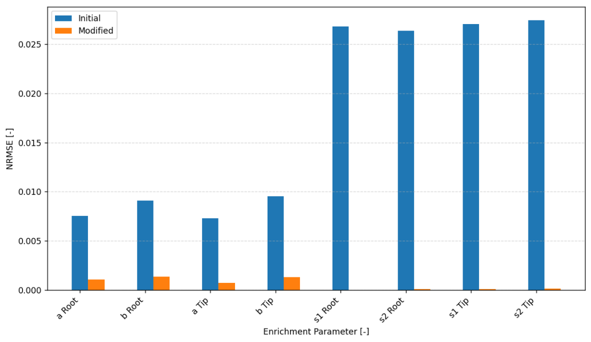
(b) MI optimized enrichment fitting

Figure 4.5: Enrichment fitting at t=120 before and after optimization

Subsequently, the enrichments were, together with five POD modes, used in the ROM. This means that the LSTM-NN with parameters as given in Table 4.1 was trained for these 15 parameters. For training, the complete snapshot matrix was split in a training, validation, and testing set of size 600, 200, and 200 respectively. The network was trained three times, for the initial ePOD, the optimized ePOD, and the regular POD without enrichments but 15 POD modes. To evaluate the MI optimized enrichments, a comparison between the intrinsic and realized predictability of the enrichment parameters is made in Figure 4.6. As can be observed in Figure 4.6a, the NMI of the control points amplitudes increased by about 50%. For the position of the control points, this is even 350%. Moreover, Figure 4.6b shows that the NRMSE of the LSTM-NN predictions decreased for all parameters. The NRMSE even became close to zero for most parameters.



(a) NMI scores of initial and MI optimized enrichment coefficients



(b) NRMSE of LSTM prediction of initial and MI optimized enrichment coefficients

Figure 4.6: Intrinsic and realized predictability of enrichment coefficients

To calculate the actual accuracy of both methods, the test snapshots had to be reconstructed from the predicted coefficients and compared to the ground truth. To do so, the relative L_2 error is used, which measures the size of the error relative to the exact solution. Equation 4.10 can be used to calculate this error, where y denotes the exact solution and $\hat{y}$ the approximate solution. To make a significant comparison, both the projection and neural network relative L_2 errors are computed. The projection error is the maximum achievable accuracy of the ROM, since this is the error when all coefficients are predicted perfectly. The neural network error is the error in de pressure distribution due to incorrectly predicted coefficients. Equation 4.11 represents the projection error, where $\mathbf{x} = (x, y)$, C_p is the exact pressure coefficient, $\overline{C_p}$ the average pressure coefficient, a_k the projected time coefficient, ϕ_k the POD mode, and ϕ_{e_s} the enrichment mode. Equation 4.12 gives the neural network error, where a_k^{NN} and p^{NN} denote the predicted time coefficients and enrichment parameters.

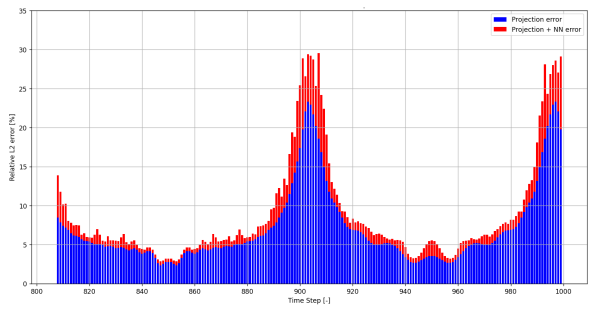
$$\text{Relative L2 error} = \frac{\|y - \hat{y}\|_2}{\|y\|_2} = \frac{\sqrt{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}}{\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{4.10}$$

$$\epsilon_{\text{P}} = \frac{\left\|(C_p(\mathbf{x},t)) - \left(\overline{C_p}(\mathbf{x}) + \sum_{k=1}^{r} a_k(t)\phi_k(\mathbf{x}) + \sum_{s=1}^{i} \phi_{e_s}(\mathbf{x}, p(t))\right)\right\|_2}{\|C_p(\mathbf{x},t)\|_2} \tag{4.11}$$

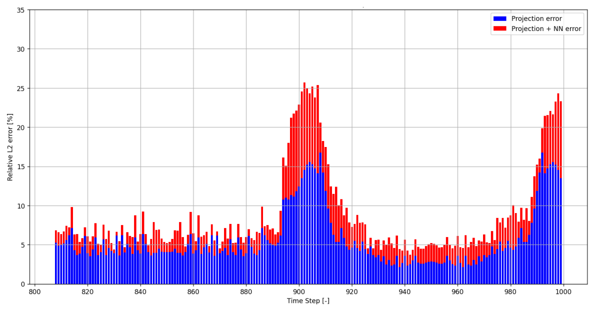
$$\epsilon_{\text{NN}} = \frac{\left\|\left(\sum_{k=1}^{r} a_k(t)\phi_k(\mathbf{x}) + \sum_{s=1}^{i} \phi_{e_s}(\mathbf{x}, p(t))\right) - \left(\sum_{k=1}^{r} a_k^{\text{NN}}(t)\phi_k(\mathbf{x}) + \sum_{s=1}^{i} \phi_{e_s}(\mathbf{x}, p^{\text{NN}}(t))\right)\right\|_2}{\left\|\overline{C_p}(\mathbf{x}) + \sum_{k=1}^{r} a_k(t)\phi_k(\mathbf{x}) + \sum_{s=1}^{i} \phi_{e_s}(\mathbf{x}, p(t))\right\|_2} \tag{4.12}$$

Figure 4.7 shows the relative L_2 errors originating from the reduced order basis and neural network accuracies of the predicted pressure distribution for all time steps in the test set. The blue, or bottom part of the bar, shows the projection error, while the red, or top part of the bar, shows the neural network error. As shown in Figure 4.7b, the projection error for the initial ePOD model is lower for most time steps compared to POD (Figure 4.7a). However, the neural network error is relatively high, reducing the accuracy of the model a lot.

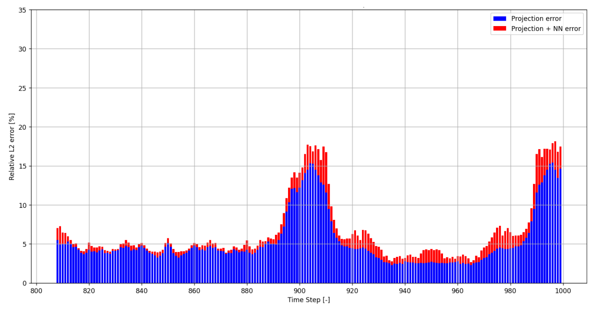
The effect of smoothing the time coefficients is shown in Figure 4.7c. As can be observed, the improved predictions of the enrichment parameters result in lower neural network errors, while the projection error stayed more or less the same. After manipulating the time coefficients such that the mutual information increased as well, the neural network error decreased even more (Figure 4.7d). Whereas the neural network error comprised roughly 40% of the total error in the initial ePOD-ROM, as shown in Figure 4.7b, this error makes up roughly 5% of the total error in the improved ePOD-ROM. In addition, the projection error did not increase but stayed constant and even decreased slightly for some time steps.



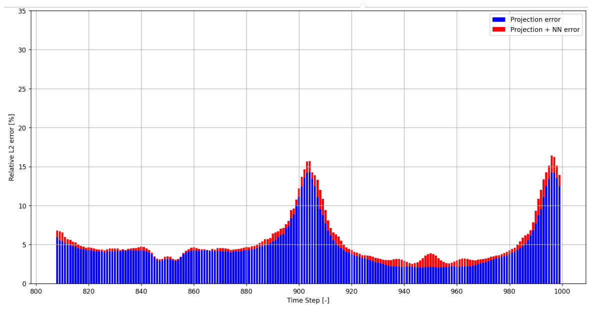
(a) L_2 error of POD predictions



(b) L_2 error of initial ePOD predictions



(c) L_2 error of smoothed ePOD predictions



(d) L_2 error of smoothed+MI optimized ePOD predictions

Figure 4.7: L_2 error comparison of neural network and projection error for ePOD-LSTM-ROM predictions on the test set

Lastly, the total L_2 error was compared for all ROM's. By means of Equation 4.13, the relative L_2 error of the predicted pressure distributions for the POD, intial ePOD, and optimized ePOD ROM's was calculated. The result is shown in Figure 4.8. As can be observed, where the initial ePOD performs equally or worse than regular POD with 15 DOF's at all time steps, the optimized ePOD performs better than regular POD for most time steps. This improvement, mainly due to the reduction in neural network error, is a result of the integrated enrichment strategy where predictability of the coefficients was taken into account. A visualization of the error of the predicted flow fields is provided in Figure B.6-Figure B.8.

$$\epsilon_{\mathrm{T}} = \frac{\left\| C_p(\mathbf{x},t) - \left(\overline{C_p}(\mathbf{x}) + \sum_{k=1}^{r} a_k^{\mathrm{NN}}(t)\phi_k(\mathbf{x})^T + \sum_{s=1}^{i} \phi_{e_s}(\mathbf{x}, p^{\mathrm{NN}}(t)) \right) \right\|_2}{\|C_p(\mathbf{x},t)\|_2} \tag{4.13}$$

The time averaged relative L_2 errors are summarized in Table 4.3 for each model. As can be observed, the L_2 error originating from the NN is reduced by 76.7% compared to the initial ePOD model. This proves that the enrichment strategy really improved the predictability of the time coefficients. Moreover, the projection error decreased with 17.5%, which is also a result of the smoothness of the enrichment time coefficients over time. It was found that POD was able to capture the enriched flow field more accurately and that the POD time coefficients were also smooth in time. As a result, the total L_2 error was reduced by 23.8%.

Table 4.3: Relative L_2 errors summarized for each source of error

	Projection error [%]	**NN error** [%]	**Total error** [%]
POD	7.0	1.9	7.3
ePOD initial	5.7	3.0	6.3
ePOD smoothed	5.3	1.2	5.5
ePOD smoothed + MI optimized	4.7	0.7	4.8

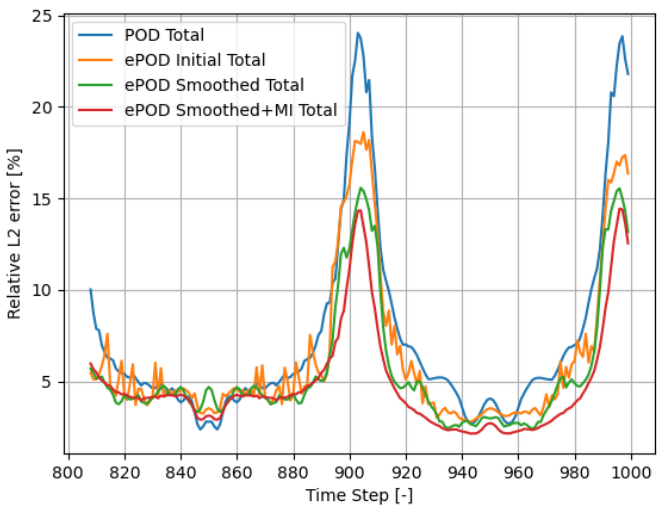


Figure 4.8: Comparison of relative L_2 error of (e)POD-LSTM-ROM predictions with 15 DOF

5 Robust Shock Prediction: Goal-Oriented Reduced-Order Model

In the previous chapters, enrichment methods have been explored to increase the ROM accurately in the shock region of discontinuous pressure distributions. It was shown however, that the number of DOF's increase rapidly with enrichments, and that the overall efficiency increased only in specific conditions. As a result, a more robust method called a Goal-Oriented Reduced-Order Model (GOROM) was developed. Instead of minimizing the L_2 error of the entire computational domain, the reduced basis of a GOROM is optimal for a specific goal function.

In this chapter, the newly developed GOROM is explained. In section 5.1, the method to construct the GOROM is explained. Subsequently, the results and comparisons with POD are presented in section 5.2. Lastly, section 5.3 presents a model that combines POD and GOROM.

5.1. Construction of GOROM with Secondary POD Basis

In this section, the method for constructing a GOROM that aims at predicting the shock accurately is explained. In regular POD, many high order modes are required to model the shock accurately, which leads to many DOF's. To prevent this, a new set of modes will be derived from a combination of these high order POD modes. This allows the GOROM to access high order modes while keeping the number of DOF's low. Ultimately, this leads to more information accessible to the GOROM needed to model its goal function accurately. The ONERA M6 dataset from chapter 3 will be used to develop and test the method.

The set of POD modes used to construct the new GOROM modes is referred to as the secondary base Ψ, and consists of M POD modes. Weight matrix C with shape $(N \times M)$ can then be used to create N GOROM modes that consist of different linear combinations of each secondary base mode Ψ_j. Note that $M > N$, which means that the number of GOROM modes is lower than the amount of modes in the secondary base. Equation 5.1 shows how the inner product between C and Ψ leads to the new GOROM modes.

$$\Phi = \sum_{i=1}^{N} \sum_{j=1}^{M} C_{ij} \Psi_j \tag{5.1}$$

Here,

$$\Phi = \begin{bmatrix} \phi_1^{GOROM} & \phi_2^{GOROM} & \cdots & \phi_n^{GOROM} \end{bmatrix}^T,$$

$$C = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1,m-1} & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2,m-1} & w_{2m} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{n,m-1} & w_{nm} \end{bmatrix},$$

and

$$\Psi = \begin{bmatrix} \phi_1^{POD} & \phi_2^{POD} & \cdots & \phi_{m-1}^{POD} & \phi_m^{POD} \end{bmatrix}^T.$$

To establish C, an optimization was needed that finds C resulting in the most accurate shock parameter of interest for the whole dataset. For this, the `scipy.optimize.minimize`[1] package was used again. The Sequential Least Squares Programming (SLSQP) optimization method was used, which is a gradient-based method that allows for bounds and constraints. The initial C is an identity matrix for the first set of columns forming a square block, while the remaining columns were filled with zeros. This means that the original POD modes are being used as the initial guess, which results in the lowest overall L_2 error, making it a suitable starting point. An example for a (3×5) initial weight matrix C_0 can be illustrated as,

$$C_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

Subsequently, the time coefficients of the GOROM modes had to be found. This can be done by means of a least-squares fit between the GOROM modes and the pressure coefficients in the shock region. Now, these time coefficients can be used to reconstruct the pressure distribution from the GOROM modes at every time step.

Based on the reconstructed pressure distributions, a user-defined parameter of interest characterising the shock was calculated. In this case, the shock gradient error was used since the adverse pressure gradient is an important parameter for flow separation. However, the shock L_2 error, error in pressure jump, or another parameter could be used as well.

To calculate the shock gradient, the ONERA M6 surface was split in chord wise sections corresponding to the number of sections present in the dataset. At each section, the location of the maximum pressure gradient was determined to find the strongest shock. Thereafter, the shock was defined by two nodes in front and aft of the maximum gradient point. Subsequently, the shock pressure gradient was calculated based on the pressure jump and distance between these two outer points. This is illustrated in Figure 5.1.

[1] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html`

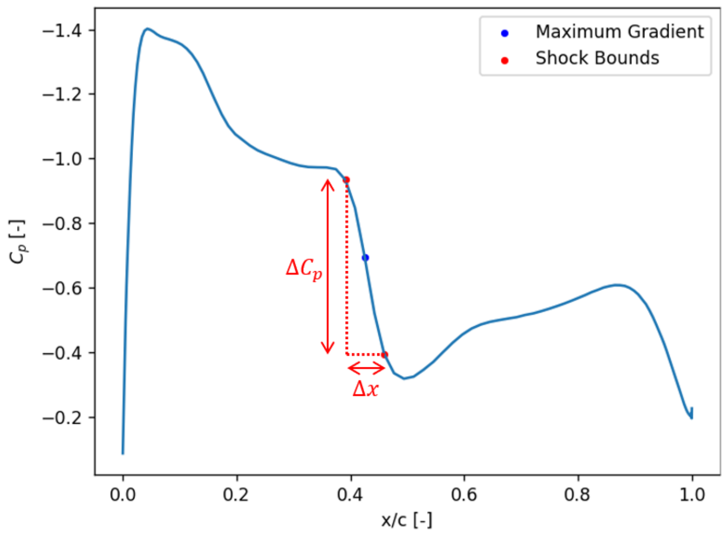


Figure 5.1: Section wise shock gradient calculation method

Subsequently, the shock gradient was calculated for each section at every time step in the dataset and the RMSE was outputted. Based on this value, the optimizer generates a new weight matrix C that should result in a lower RMSE.

In addition to the steps explained before, an additional intermediate step can be added. Since chapter 4 showed that smoothing the time coefficients resulted in a lower neural network error, this step can be added to the optimization process as well if the neural network error turns out to be high. When the time coefficients are calculated by means of least-squares, the Savgol filter can be applied to the resulting time coefficients. This means that non-smooth time coefficients are modified substantially, while the filter has little effect on smooth time coefficients. This means that the Savgol filter acts like a penalty for non-smooth time coefficients, since the reconstructed pressure distributions will be very different from the least-squares. As a result, the optimization disregards these options and the result should be modes that have smooth time coefficients.

The GOROM method is summarized in Algorithm 3.

5.2. GOROM Results and Efficiency

In this section, the results of the GOROM method are discussed. As explained in the previous section, the ONERA M6 dataset from chapter 3 is used to assess the efficiency of the method. There, it was found that by using 10 POD modes, the average L_2 error was 4.4% in the shock domain. To be able to obtain a higher accuracy, the secondary base should consist of more modes. When using 30 POD modes, the average L_2 error in the shock domain was found to be 0.009%. So, it was decided to use 30 POD modes in the secondary base. As a consequence, the weight matrix C has 30 columns.

Since the number of DOFs in the GOROM depends on N, the number of modes constructed from the secondary basis, a study was conducted to evaluate its effect on the shock gradient RMSE. The accuracy was determined for up to 10 modes, for which 10 different weight matrices had to be determined with Algorithm 3. The result is shown in Figure 5.2, where the POD accuracy is shown as well.

Algorithm 3 GOROM for Shock Gradient Algorithm

```
1: procedure GOROM(Ψ, Q)
2:     $shock\_gradients, shock\_area \leftarrow$ CALCULATE_GRADIENTS(Q)    ▷ True shock gradients
3:     Initialize $C^{(0)}$    ▷ Initial guess for $C$
4:     while stopping criteria not met do
5:         $N_\phi \leftarrow$ C.shape[0]
6:         $N_\psi \leftarrow$ C.shape[1]
7:         Compute $\Phi = \langle C, \Psi \rangle$    ▷ Construct GOROM modes
8:         for all $t \in T$ do
9:             Compute $Q[shock\_area][t]$ and $\Phi[shock\_area]$
10:            Compute A with least-squares between $Q[shock\_area][t]$ and $\Phi[shock\_area]$
11:        end for
12:        Optional: Apply Savgol filter to time coefficients
13:        Reconstruct pressure distributions via $\langle \Phi, A \rangle$
14:        for all $t \in T$ do
15:            Compute shock gradient error at each section
16:        end for
17:        Compute RMSE of shock gradient errors
18:        Optimize RMSE with scipy.optimize.minimize by adjusting $C^{(i)}$
19:    end while
20:    return $C$
21: end procedure
```

As shown in Figure 5.2, the GOROM model shows an initial offset compared to POD, resulting in a lower shock gradient RMSE with just one mode. Furthermore, the error decreases linearly for POD, while the decreases is exponential for GOROM. The largest difference can be seen at five modes, where the error decrease is almost 80%. To reach this efficiency, the POD needs 13 modes, which is more than 2.5 times more. Because of this, additional results will be presented in this section for five modes.

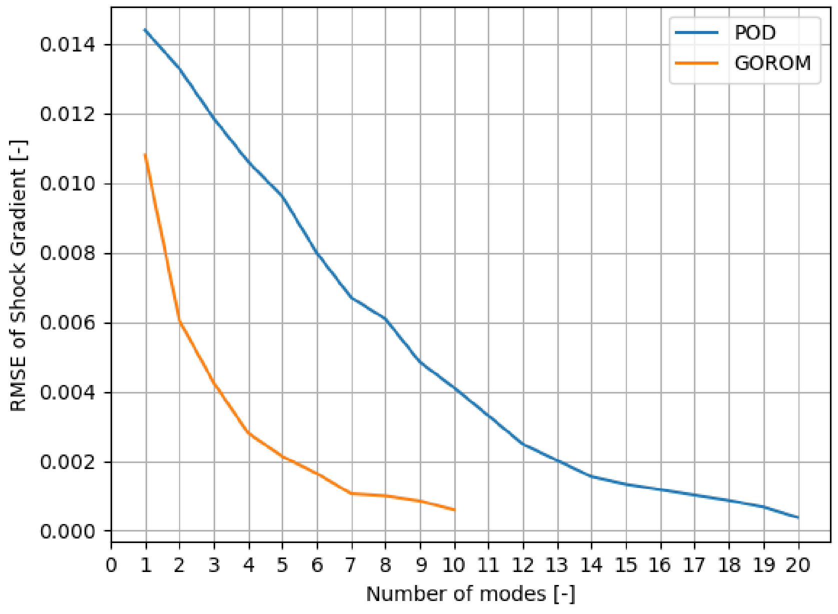


Figure 5.2: RMSE of shock gradient against number of modes for POD and GOROM

Figure 5.3 shows weight matrix C visualized as a heatmap. As can be observed, each GOROM mode has the strongest contribution from its original corresponding POD mode. However, each mode also receives strong contributions from other secondary modes up to number 15. Secondary modes 15 to 30 also contribute, but to a smaller extent. It can also be observed that secondary mode 12 contributes a lot to the first 4 GOROM modes. Lastly, even the last secondary mode contributes slightly, which makes clear that the size of the secondary base is not excessive.

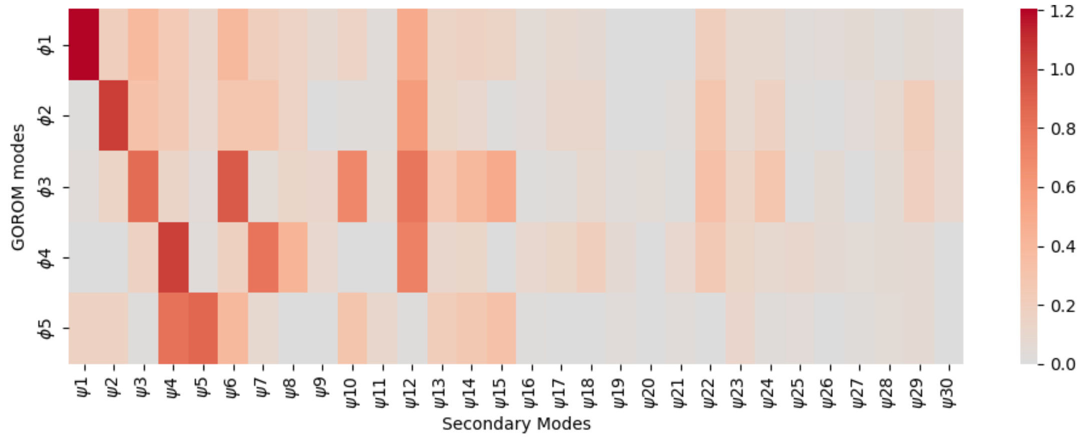


Figure 5.3: Heatmap of weight matrix C with five GOROM modes

To visualize the effect of the weights from Figure 5.3, the resulting GOROM modes are plotted together with the initial POD modes in Figure 5.4. As can be observed, the GOROM modes define the shocks a lot better. Especially the first mode is manipulated such that it is able to capture the shock effectively. Moreover, the POD mode content in each subsequent mode increases faster, which allows for a more accurate shock representation. It can also be observed that the shock/vortex at the tip region is defined a lot more at all GOROM modes. All together, this shows that the GOROM method is capable of manipulating the modes such that they are effectively capturing a user defined property.

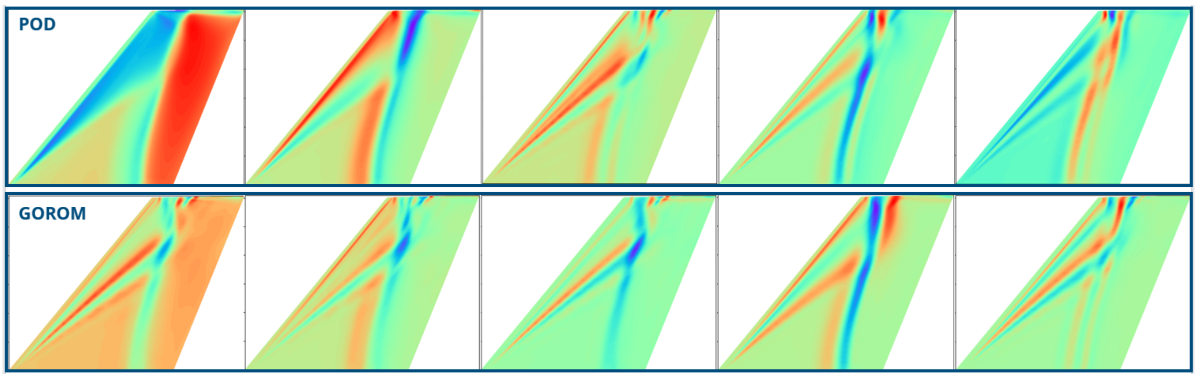
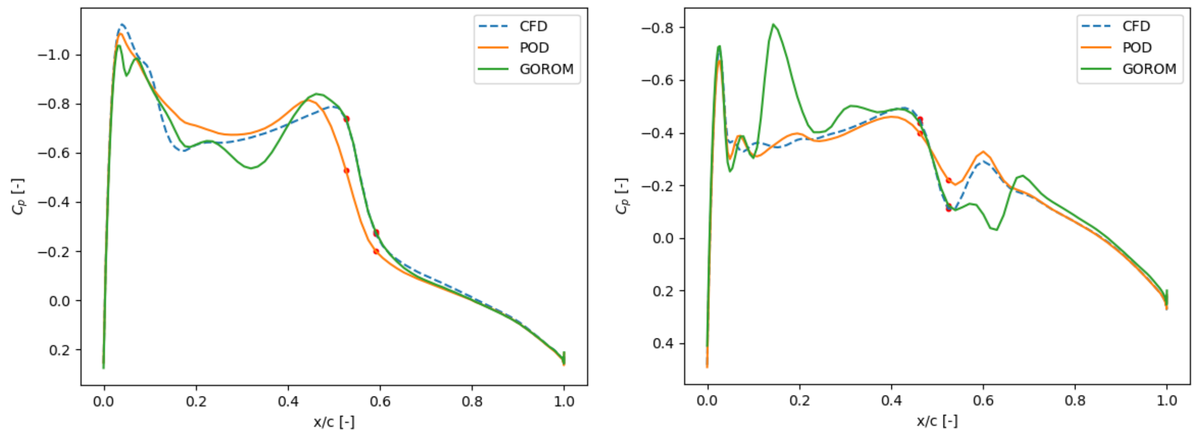


Figure 5.4: POD and GOROM modes with increasing order from left to right

By means of the five GOROM modes showed in Figure 5.4, the pressure distributions of all 320 snapshots in the ONERA M6 dataset were reconstructed. Subsequently, these pressure distributions were used to analyse the pressure gradient of the shock at every section of the wing. Figure 5.5 shows the one-dimensional pressure distributions of two arbitrary wing sections at time step 0 (Figure 5.5a) and time step 20 (Figure 5.5b).

It can be observed in Figure 5.5a that the POD solution not only underestimates the pressure gradient, its location is also off. The GOROM solution on the other hand, snaps onto the true pressure distribution at the shock location. Figure 5.5b for time step 20 shows the same behaviour. Again, the POD underestimates the pressure gradient, a well known issue, while the GOROM reconstructs the shock very accurately.

One could observe that while the shock is reconstructed very accurately, the rest of the pressure distribution is far from accurate. Outside of the shock area, POD is more accurate than GOROM. However, since this is a goal-oriented model, the sole purpose is to model the shock accurately. It is demonstrated that by recombining existing modes, an accurate solution can be realized in certain finite areas of the solution domain. By squeezing these modes together in the interest areas however, a less accurate solution is created outside these areas.



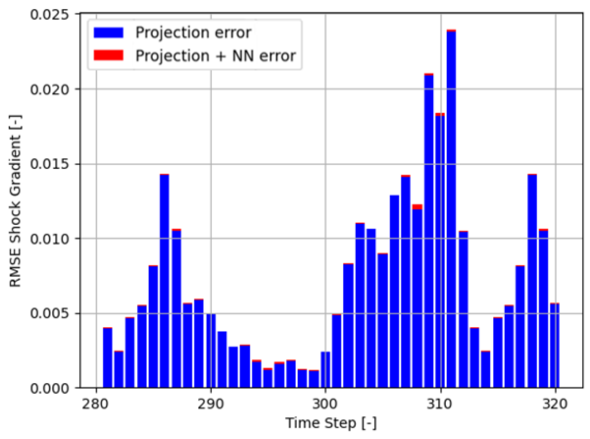
(a) Sectional pressure distribution at t=0 **(b)** Sectional pressure distribution at t=20

Figure 5.5: Sectional pressure distributions constructed with POD and GOROM compared with CFD

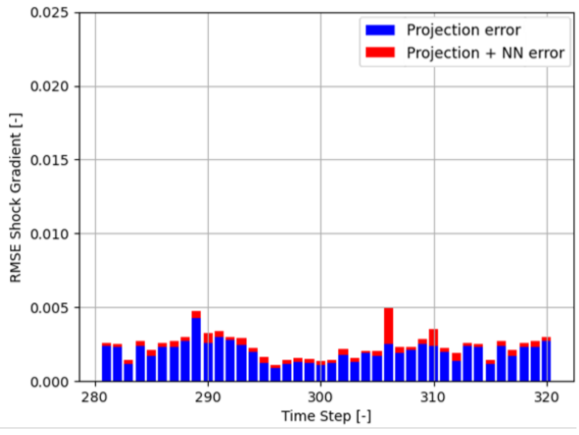
The next step is to train the LSTM-NN presented in section 4.1 to predict the time coefficients corresponding to each of the GOROM modes based on input variables defining the orientation of the ONERA M6 wing. The results of the prediction accuracy for each time step in the test set are shown in Figure 5.6. From Figure 5.2, it could already be concluded that the projection error is lower for GOROM, which is confirmed by the blue part of the bars. An additional insight however, is that this is true for each individual time step.

Another important result however, is that the additional error introduced by the neural network is low. Defined by the red part of the bars, it can be observed that the neural network only adds a very small error to the GOROM shock gradient. Whereas the neural network error for the POD is almost negligible, this additional error for the GOROM does not make the GOROM less effective. This means that the time coefficients associated with the GOROM modes are well predictable, making the modes useful.

A remarkable result is that the shock gradient RMSE stays below 0.005 for each time step. This means that the GOROM accuracy can be specified as 0.005. For POD, a time step with a 0.024 RMSE can be observed, making the uncertainty almost five times higher. The total RMSE for the POD was found to be 0.0096, while this is 0.0021 for the GOROM, a decrease of 77.9%.



(a) Shock gradient RMSE POD



(b) Shock gradient RMSE GOROM

Figure 5.6: RMSE of the shock gradient originating from the reduced order basis and neural network for each time step in the test set

Since the neural network error is very small, it is not needed to apply the Savgol filter in the optimization process. However, to investigate the effect of the filter, GOROM modes with the optional Savgol filter enabled were constructed as well. Figure 5.7 shows the resulting time coefficients for all five GOROM modes. To compare these, the prior found non-smoothed time coefficients are shown as well. It can be observed that the smoothed time coefficients are different in nature, and not just a smooth version of the initially found time coefficients. This means that a different solution is found when non-smooth time coefficients are penalized, resulting in different modes with different behaviour over time.

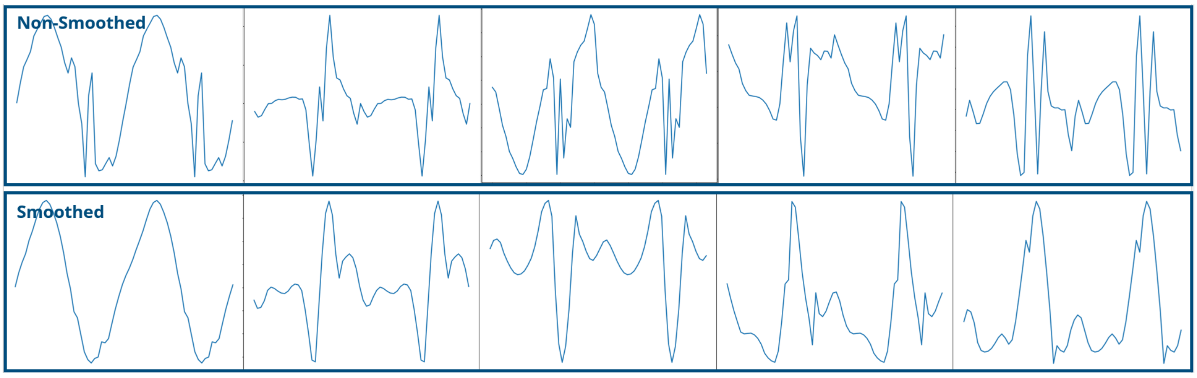


Figure 5.7: Time coefficients of GOROM modes with and without smoothing in optimization process

The RMSE originating from the reduced order basis, neural network, and the complete error are summarized for all models in Table 5.1. As can be observed, creating GOROM modes that behave smoothly over time, comes at the cost of accuracy. Whereas the reduced order basis of the non-smoothed GOROM yields a 77% decrease in RMSE compared to POD, only a 45% decrease can be realized with the smoothed GOROM. As a result, the projection error is almost 2.5 times larger by introducing smoothing. When this increase in projection error is cancelled out by the decrease in neural network error however, smoothing can still be beneficial. However, it was found that this is not the case since the neural network error is very small for all models. As a result, for this application, the total error was higher for the GOROM with smooth time coefficients.

Table 5.1: GOROM RMSE of shock gradients compared

Model	Projection error [-]	NN error [-]	Total error [-]
POD	9.25×10^{-3}	9.77×10^{-5}	9.27×10^{-3}
GOROM (non-smoothed)	2.14×10^{-3}	5.34×10^{-4}	2.22×10^{-3}
GOROM (smoothed)	5.05×10^{-3}	1.34×10^{-3}	5.32×10^{-3}

A GOROM model was also developed with a different objective: minimizing the L_2 error in the shock domain. Here, similar results were found compared to the shock gradient objective. This highlights the flexibility of the method and confirms that smoothness comes at the cost of higher projection errors.

5.3. Combining GOROM with POD

The previous section showed that through optimization, modes can be created that perform very well in a specific norm. However, the presented results are a rather extreme case, in which most of the reconstructed pressure distribution is inaccurate. GOROM can be used however, to alter the POD modes with a certain amount toward a more accurate shock representation. In this section, it is investigated to what extent the modified POD modes improve the shock accuracy.

In the previous section, two extremes of the spectrum have been presented. The first extreme is POD, which results in the lowest overall L_2 error but does not specifically aim for accuracy in the shock region. The other extreme is a GOROM that only aims for an accurate shock representation, disregarding the remaining areas. To make GOROM practically useful, it should be applied in an optimization loop that balances both the overall L_2 error and the shock-domain L_2 error.

A slightly adjusted version of algorithm 3 was used to allow for different norms. In this version, instead of the RMSE of the pressure gradient, the L_2 error in the shock domain is calculated. Time coefficients are obtained via least-squares over the entire domain. Finally, the norm to be minimized is defined as a weighted combination of the overall L_2 error and the shock-domain L_2 error. This allows the resulting modes to lie between the two extremes, where they are accurate overall while placing increased focus on the shock domain. It was confirmed that by using a norm that consists entirely of the overall L_2 error, the identity matrix was found for C, which gives the original POD modes.

Norms with varying fractions of shock-domain L_2 error were used to compute new GOROM modes. Figure 5.8 shows the relative L_2 error in the entire and in the shock domain as a function of the shock-domain error fraction in the norm. A rapid decrease in shock-domain L_2 error is observed when increasing its weight up to 10%, after which the error decreases more gradually to a value of 14.2%. The overall L_2 error increases gradually from 6.5% to 7.2%. This demonstrates that POD modes can be adjusted to create GOROM modes that are more shock-focused without causing an extreme increase in overall L_2 error. A small shock weighted norm of 10% reduces the shock-domain L_2 error by 5.0% while increasing total L_2 error by only 0.7%. A 100% shock weighted norm on the other hand decreases shock-domain L_2 error by 10.3% but increases total L_2 error by 12.4%. This shows that the overall L_2 error increases progressively for more shock accurate modes.

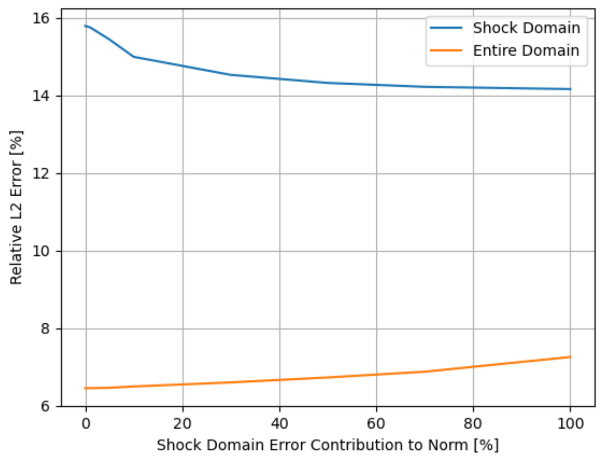
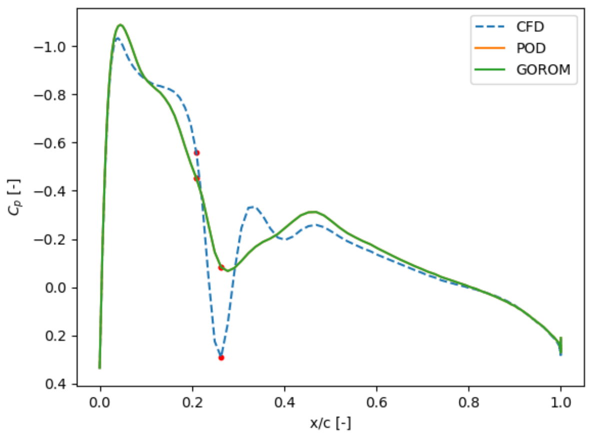


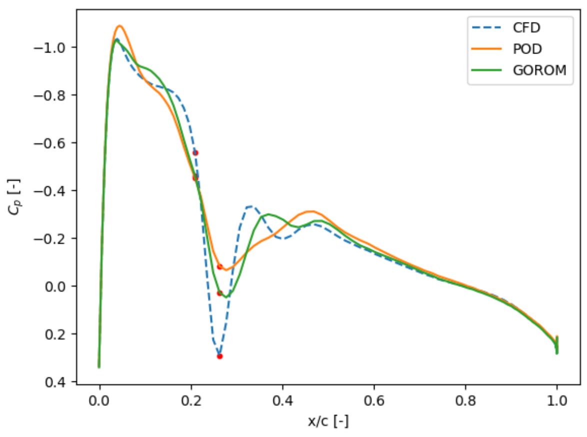
Figure 5.8: Comparison of the relative L_2 error in the entire and shock domain as a result of the optimization norm composition

It is interesting to observe in Figure 5.8 that the relative L_2 error in the shock domain stays relatively high, even for norms that are mainly composed of the shock domain error. Clearly, the shock domain can not be reconstructed as accurately as in a pure GOROM, since this leads to an overall L_2 error that is too high. Therefore, any improvement in shock accuracy comes at the cost of reduced accuracy elsewhere.

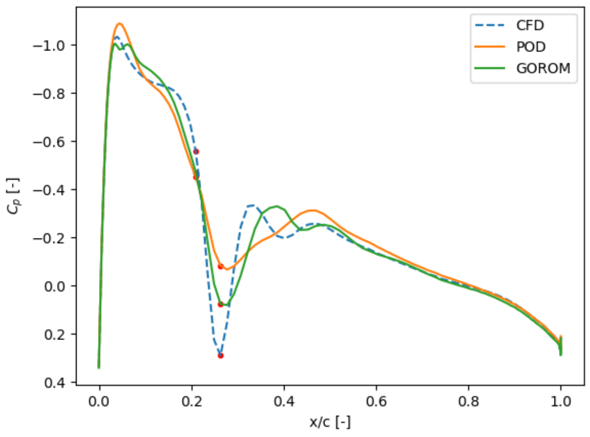
To illustrate this, four different sectional pressure distributions at time step 20 constructed with five GOROM modes are shown Figure 5.9. The GOROM basis used in each figure were constructed with different norms. In Figure 5.9a, the norm consisted entirely of the overall L_2 error. As a result, the constructed GOROM modes are equal to the POD modes. In Figure 5.9b, the norm consists for 30% of the shock domain L_2 error. As can be observed, GOROM captures the shock more accurately than POD as the pressure peak is approached more. Moreover, Figure 5.9c shows the pressure distribution for 70% shock domain L_2 error in the norm. Here, GOROM reconstructs the shock even more accurately. Lastly, for 100% in Figure 5.9d, the shock is almost entirely captured, while the rest of the pressure distribution is a bit less accurate.



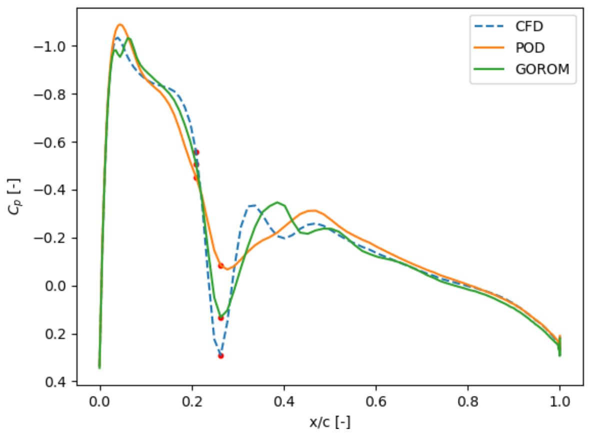
(a) GOROM norm consists 0% of shock domain L_2 error



(b) GOROM norm consists 30% of shock domain L_2 error



(c) GOROM norm consists 70% of shock domain L_2 error



(d) GOROM norm consists 100% of shock domain L_2 error

Figure 5.9: Sectional pressure distributions at t=20 constructed with POD and GOROM with different norms compared with CFD

These results highlight a clear trade-off between global accuracy and shock-domain accuracy. By systematically adjusting the POD modes using GOROM, it is possible to create modes that balance overall reconstruction accuracy with the goal-function. Low shock-weighted norms can substantially improve the shock region while barely affecting the total L_2 error, whereas extreme shock weighting achieves maximum shock accuracy at the cost of global accuracy. This demonstrates that GOROM provides a flexible and robust approach to improve POD for flows with discontinuities, with the possibility to tune the method to match specific accuracy requirements.

6 Conclusion and Recommendations

This chapter presents the most important findings of the research done in this thesis on the basis of the research questions given in section 1.2. Moreover, recommendations for future research and applications are given.

6.1. Conclusion

The work in this thesis aimed at developing a method for constructing a Reduced-Order-Model (ROM) that accurately predicts the unsteady surface pressure distribution in the shock region of a wing planform in the transonic flow regime. The method should, in particular, create a reduced-order basis of modes that model the shock region more accurately. This should lead to a decrease in the number of temporal coefficients to be evaluated by a Long Short-Term Memory (LSTM) neural network. To fulfil this objective, three research questions were explored.

First of all, it was investigated how well an ePOD-LSTM-ROM performs in multidimensional space. To do so, a method was developed to implement enrichment modes in reconstructing a wing pressure distribution. Enrichment surfaces, created by linear interpolation between two one-dimensional enrichment functions, proved very effective in smoothing out flow discontinuities. Moreover, by making use of domain decomposition, a flow field containing multiple discontinuities could be enriched with the same method. This showed the scalability of the method, as it provides a systematic way to construct a dynamic mode capable of capturing varying discontinuities with a limited number of DOF. For more discontinuous flow fields, an increasingly large number of POD modes is required for an accurate reconstruction. Enrichment on the other hand, relies on a fixed number of DOF, which makes it more advantageous for strongly varying discontinuities. It was found that a flow field containing a sharp discontinuity that moves more than 26% has a 39% decreased overall L_2 error with the same number of DOF.

Secondly, it was investigated if it is possible to decrease the total model error by anticipating the neural network performance already in the enrichment design. Mutual Information was found to be a good intrinsic predictability measure, as it showed a correlation with the RMSE of the LSTM-NN predictions. Hence, smoothing and MI were used to manipulate the time coefficients of the enrichment parameters. This reduced the L_2 error originating from the LSTM-NN by 76.7% compared to the initial ePOD model. As a result, the total L_2 was reduced by 23.8%. This improvement was the result of more accurate enrichment time coefficient predictions, as well as more accurate POD time coefficients predictions. It was observed that enrichment time coefficients that are smooth in time result in smooth POD time coefficients over time as well, as the enriched flow field evolves smoothly..

Lastly, the possibility of constructing modes through goal-oriented optimization was explored. A secondary base consisting of the first 30 POD modes was used to construct five new modes by minimizing a norm describing the shock projection error. This goal-oriented reduced-order model (GOROM) aimed at a low shock projection error demonstrated to have a lot of potential, since the RMSE of the shock gradient was decreased by 80% compared to POD. However, in this case, the region outside the shock region was disregarded. By combining the total L_2 error and L_2 error of the shock region in a norm, it was found that the modes can be manipulated to be more shock focussed. For a small shock weighted norm, a decrease of 5.0% percent of shock domain L_2 error results in an increase of 0.7% of total L_2 error. The method showed to be very robust and convenient, since the modes are constructed through an optimization loop only.

6.2. Recommendations

Based on the work presented in this thesis, some recommendations can be made for future work. The methods developed have limitations, and not all aspects of each method have been explored due to the scope of this thesis. As a result, improvements can still be made.

First of all, the domain decomposition used in section 3.2 was based on a geometric perspective, aimed at isolating discontinuities and positioning subdomain boundaries as far away as possible. However, additional options should be explored, potentially through an optimization that identifies the most effective domain decomposition. Moreover, a real dataset with sharp discontinuities moving significantly in time should be enriched with domain decomposition to determine if ePOD is beneficial for such a dataset. Furthermore, it is recommended to use the enrichment method only for highly discontinuous flow fields.

Secondly, it is recommended to apply smoothing and MI optimization to the enrichment time coefficients. The results in chapter 4 showed that the NN error decreases drastically while the projection error did not increase. However, the effectiveness on real datasets still has to be assessed. It is expected however, that also there the method will lead to accuracy improvements.

Lastly, the GOROM method developed in chapter 5 is very promising. It should be explored however, if goal functions augmented with MI are beneficial for problems with large NN errors. Moreover, it should be investigated if GOROM modes can be reconstructed for highly discontinuous datasets as well, as the ONERA M6 dataset is relatively smooth.

References

[1] A. S. Pototzky and R. W. Moses, “A Method to Analyze Tail Buffet Loads of Aircraft,” en, in *Flow Induced Unsteady Loads and the Impact on Military Applications*, Neuilly-sur-Seine, France, 2005, pp. 19-1 –19-16.

[2] C. Gao and W. Zhang, “Transonic aeroelasticity: A new perspective from the fluid mode,” en, *Progress in Aerospace Sciences*, vol. 113, p. 100 596, Feb. 2020, ISSN: 03760421. DOI: 10.1016/j.paerosci.2019.100596.

[3] M. Stradtner, D. Drazen, and M. Van Rooij, “Introduction to AVT-351: Enhanced Computational Performance and Stability & Control Prediction for NATO Military Vehicles,” en, in *AIAA SCITECH 2023 Forum*, National Harbor, MD & Online: American Institute of Aeronautics and Astronautics, Jan. 2023, ISBN: 978-1-62410-699-6. DOI: 10.2514/6.2023-0820.

[4] M. Widhalm, M. Stradtner, A. Schuette, M. Ghoreyshi, A. Jirasek, and J. Seidel, “Comparison of Reduced Order Models for Evaluating Stability Derivatives for the DLR-F22 ONERA model,” en, in *AIAA AVIATION 2023 Forum*, San Diego, CA and Online: American Institute of Aeronautics and Astronautics, Jun. 2023, ISBN: 978-1-62410-704-7. DOI: 10.2514/6.2023-4199.

[5] M. Ghoreyshi, P. Aref, M. Stradtner, *et al.*, “Evaluation of Reduced Order Aerodynamic Models for Transonic Flow Over a Multiple-Swept Wing Configuration,” en, in *AIAA AVIATION FORUM AND ASCEND 2024*, Las Vegas, Nevada: American Institute of Aeronautics and Astronautics, Jul. 2024, ISBN: 978-1-62410-716-0. DOI: 10.2514/6.2024-4158.

[6] D. Massegur Sampietro and A. Da Ronch, “Recurrent Geometric Deep Learning for Aerodynamic Prediction of the Future Fighter Demonstrator in Dynamic Manoeuvres,” en, in *AIAA AVIATION FORUM AND ASCEND 2024*, Las Vegas, Nevada: American Institute of Aeronautics and Astronautics, Jul. 2024, ISBN: 978-1-62410-716-0. DOI: 10.2514/6.2024-4067.

[7] NATO Science & Technology Organization. “About the STO.” (2025), [Online]. Available: https://www.sto.nato.int/Pages/organization.aspx (visited on 01/24/2025).

[8] NATO Science & Technology Organization. “Enhanced Computational Performance and Stability & Control Prediction for NATO Military Vehicles.” (2025), [Online]. Available: https://www.sto.nato.int/Lists/test1/activitydetails.aspx?ID=16796 (visited on 01/24/2025).

[9] M. Ghoreyshi, A. Jirásek, and R. M. Cummings, “Reduced order unsteady aerodynamic modeling for stability and control analysis using computational fluid dynamics,” en, *Progress in Aerospace Sciences*, vol. 71, pp. 167–217, Nov. 2014, ISSN: 03760421. DOI: 10.1016/j.paerosci.2014.09.001.

[10] D. Papp, “Prediction of unsteady nonlinear aerodynamic loads using deep convolutional neural networks,” *TU Delft Repository*, 2018.

[11] S. J. L. B. Bourier, “Development of a CFD data-driven surrogate model using the neural network approach for prediction of aircraft performance characteristics,” *TU Delft Repository*, 2021.

[12] G. Catalani, “Machine Learning Based Local Reduced Order Modeling for the prediction of unsteady aerodynamic loads,” *TU Delft Repository*, 2022.

[13] A. Panagiotopoulos, “Machine Learning Based Reduced-Order Modeling for the Prediction of Pressure Distribution in the Transonic Flow Regime,” *TU Delft Repository*, 2024.

[14] P. D. Ciampa, B. Nagel, and L. R. Gianfranco, “Preliminary Design for Flexible Aircraft in a Collaborative Environment,” en, in *The International Conference of the European Aerospace Societies (CEAS)*, Linköping, Sweden, 2013.

[15] X. Gu, P. D. Ciampa, and B. Nagel, “An automated CFD analysis workflow in overall aircraft design applications,” en, *CEAS Aeronautical Journal*, vol. 9, no. 1, pp. 3–13, Mar. 2018, ISSN: 1869-5582, 1869-5590. DOI: 10.1007/s13272-017-0264-1.

[16] D. Greenwell, “A Review of Unsteady Aerodynamic Modelling for Flight Dynamics of Manoeuvrable Aircraft,” en, in *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, Providence, Rhode Island: American Institute of Aeronautics and Astronautics, Aug. 2004, ISBN: 978-1-62410-072-7. DOI: 10.2514/6.2004-5276.

[17] H. Babinsky and J. K. Harvey, Eds., *Shock Wave-Boundary-Layer Interactions* (Cambridge Aerospace Series). New York, USA: Cambridge University Press, 2011, vol. 32, ISBN: 978-0-521-84852-7. DOI: 10.1017/CBO9780511842757.

[18] B. Lee, “Self-sustained shock oscillations on airfoils at transonic speeds,” en, *Progress in Aerospace Sciences*, vol. 37, no. 2, pp. 147–196, Feb. 2001, ISSN: 03760421. DOI: 10.1016/S0376-0421(01)00003-3.

[19] M. Shahryari, A. Yazdani, M. Khalighi, and M.-R. Salimi, “Mutual information-based feature selection for estimating an appropriate vector-valued seismic intensity measure,” *Soil Dynamics and Earthquake Engineering*, vol. 199, 2025, Cited by: 0. DOI: 10.1016/j.soildyn.2025.109676.

[20] T. Elemam and M. Elshrkawey, “A highly discriminative hybrid feature selection algorithm for cancer diagnosis,” *The Scientific World Journal*, vol. 2022, no. 1, p. 1 056 490, 2022. DOI: https://doi.org/10.1155/2022/1056490.

[21] K. Dissanayake and M. G. Md Johar, “Comparative study on heart disease prediction using feature selection techniques on classification algorithms,” *Applied Computational Intelligence and Soft Computing*, vol. 2021, no. 1, p. 5 581 806, 2021. DOI: https://doi.org/10.1155/2021/5581806.

[22] W. Wang, “High-accuracy PM2.5 prediction via mutual information filtering and Bayesian-Optimized Spatio-Temporal Convolutional Networks,” *Scientific Reports*, vol. 15, no. 1, p. 21 718, Jul. 2025, ISSN: 2045-2322. DOI: 10.1038/s41598-025-08896-1.

[23] U. Ahmed, A. Mahmood, A. R. Khan, *et al.*, “Parallel boosting neural network with mutual information for day-ahead solar irradiance forecasting,” *Scientific Reports*, vol. 15, no. 1, 2025, Cited by: 0. DOI: 10.1038/s41598-025-95891-1.

[24] P. Blom, “Goal-oriented reduced-order modeling of the incompressible navier-stokes equations: Application to a 2d transitional boundary layer,” *TU Delft Repository*, 2021.

[25] T. Bui-Thanh, K. Willcox, O. Ghattas, and B. van Bloemen Waanders, “Goal-oriented, model-constrained optimization for reduction of large-scale systems,” *Journal of Computational Physics*, vol. 224, no. 2, pp. 880–896, 2007, ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2006.10.026.

[26] M. Dias Ribeiro, M. Stradtner, and P. Bekemeyer, “Unsteady reduced order model with neural networks and flight-physics-based regularization for aerodynamic applications,” en, *Computers & Fluids*, vol. 264, p. 105 949, Oct. 2023, ISSN: 00457930. DOI: 10.1016/j.compfluid.2023.105949.

[27] K. Balla, R. Sevilla, O. Hassan, and K. Morgan, “An application of neural networks to the prediction of aerodynamic coefficients of aerofoils and wings,” en, *Applied Mathematical Modelling*, vol. 96, pp. 456–479, Aug. 2021, ISSN: 0307904X. DOI: 10.1016/j.apm.2021.03.019.

[28] J.-H. Shin and K.-W. Cho, “Comparative study on reduced models of unsteady aerodynamics using proper orthogonal decomposition and deep neural network,” en, *Journal of Mechanical Science and Technology*, vol. 36, no. 9, pp. 4491–4499, Sep. 2022, ISSN: 1738-494X, 1976-3824. DOI: 10.1007/s12206-022-0813-3.

[29] R. Deshmukh, T. Shilt, and J. J. McNamara, “Efficient representation of turbulent flows using data-enriched finite elements,” en, *International Journal for Numerical Methods in Engineering*, vol. 121, no. 15, pp. 3397–3416, Aug. 2020, ISSN: 0029-5981, 1097-0207. DOI: 10.1002/nme.6364.

[30] T. Rabczuk and K.-J. Bathe, Eds., *Machine Learning in Modeling and Simulation: Methods and Applications* (Computational Methods in Engineering & the Sciences), en. Cham: Springer International Publishing, 2023. DOI: 10.1007/978-3-031-36644-4.

[31] G. Berkooz, P. Holmes, and J. L. Lumley, “The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows,” en, *Annual Review of Fluid Mechanics*, vol. 25, no. 1, pp. 539–575, Jan. 1993, ISSN: 0066-4189, 1545-4479. DOI: 10.1146/annurev.fl.25.010193.002543.

[32] D. Hines Chaves and P. Bekemeyer, “Data-Driven Reduced Order Modeling for Aerodynamic Flow Predictions,” en, in *8th European Congress on Computational Methods in Applied Sciences and Engineering*, CIMNE, 2022. DOI: 10.23967/eccomas.2022.077.

[33] S. Fresca and A. Manzoni, “Pod-dl-rom: Enhancing deep learning-based reduced order models for nonlinear parametrized pdes by proper orthogonal decomposition,” *Computer Methods in Applied Mechanics and Engineering*, vol. 388, p. 114 181, 2022, ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2021.114181.

[34] J. L. Lumley, “The structure of inhomogeneous turbulent flows,” en, *Atmospheric turbulence and radio wave propagation*, pp. 166–178, 1967.

[35] K. Taira, S. L. Brunton, S. T. M. Dawson, *et al.*, “Modal Analysis of Fluid Flows: An Overview,” en, *AIAA Journal*, vol. 55, no. 12, pp. 4013–4041, Dec. 2017, ISSN: 0001-1452, 1533-385X. DOI: 10.2514/1.J056060.

[36] A. Chatterjee, “An introduction to the proper orthogonal decomposition,” *Current Science*, vol. 78, no. 7, pp. 808–817, 2000, ISSN: 00113891.

[37] J. Weiss, “A Tutorial on the Proper Orthogonal Decomposition,” en, in *AIAA Aviation 2019 Forum*, Dallas, Texas: American Institute of Aeronautics and Astronautics, Jun. 2019, ISBN: 978-1-62410-589-0. DOI: 10.2514/6.2019-3333.

[38] T. R. Smith, J. Moehlis, and P. Holmes, “Low-Dimensional Modelling of Turbulence Using the Proper Orthogonal Decomposition: A Tutorial,” en, *Nonlinear Dynamics*, vol. 41, no. 1-3, pp. 275–307, Aug. 2005, ISSN: 0924-090X, 1573-269X. DOI: 10.1007/s11071-005-2823-y.

[39] O. T. Schmidt and T. Colonius, "Guide to Spectral Proper Orthogonal Decomposition," en, *AIAA Journal*, vol. 58, no. 3, pp. 1023–1033, Mar. 2020, ISSN: 0001-1452, 1533-385X. DOI: 10.2514/1.J058809.

[40] H. Zhou, M. Nie, M. Qin, and G. Wang, "An unsteady aerodynamic reduced-order modelling framework for shock-dominated flow with application on shock-induced panel flutter prediction," en, *Journal of Fluids and Structures*, vol. 133, p. 104 251, Mar. 2025, ISSN: 08899746. DOI: 10.1016/j.jfluidstructs.2024.104251.

[41] O. San, R. Maulik, and M. Ahmed, "An artificial neural network framework for reduced order modeling of transient flows," *Communications in Nonlinear Science and Numerical Simulation*, vol. 77, pp. 271–287, 2019, ISSN: 1007-5704. DOI: https://doi.org/10.1016/j.cnsns.2019.04.025.

[42] G. Padula, M. Girfoglio, and G. Rozza, *A brief review of reduced order models using intrusive and non-intrusive techniques*, 2024. arXiv: 2406.00559 [math.NA].

[43] R. Mojgani and M. Balajewicz, *Lagrangian basis method for dimensionality reduction of convection dominated nonlinear flows*, 2017. eprint: 1701.04343.

[44] J. Li and W. Zhang, "The performance of proper orthogonal decomposition in discontinuous flows," *Theoretical and Applied Mechanics Letters*, vol. 6, no. 5, pp. 236–243, 2016, ISSN: 2095-0349. DOI: https://doi.org/10.1016/j.taml.2016.08.008.

[45] W. Aquino, J. C. Brigham, C. J. Earls, and N. Sukumar, "Generalized finite element method using proper orthogonal decomposition," *International Journal for Numerical Methods in Engineering*, vol. 79, no. 7, pp. 887–906, 2009. DOI: https://doi.org/10.1002/nme.2604.

[46] A. Higashida, K. Ando, M. Rüttgers, A. Lintermann, and M. Tsubokura, "Robustness evaluation of large-scale machine learning-based reduced order models for reproducing flow fields," en, *Future Generation Computer Systems*, vol. 159, pp. 243–254, Oct. 2024, ISSN: 0167739X. DOI: 10.1016/j.future.2024.05.005.

[47] S. A. Renganathan, R. Maulik, and V. Rao, *Machine-learning for nonintrusive model order reduction of the parametric inviscid transonic flow past an airfoil*, 2020. arXiv: 1911.07943 [physics.flu-dyn].

[48] B. A. Roccia, M. Ruiz, and C. G. Gebhardt, "On the use of feed-forward neural networks in the context of surrogate aeroelastic simulations: On the use of feed-forward neural networks: B. a. roccia et al.," *Acta Mechanica*, 2024. DOI: 10.1007/s00707-024-04165-w.

[49] A. T. Mohan and D. V. Gaitonde, *A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks*, 2018. arXiv: 1804.09269 [physics.comp-ph].

[50] M. Milano and P. Koumoutsakos, "Neural network modeling for near wall turbulent flow," *Journal of Computational Physics*, vol. 182, no. 1, pp. 1–26, 2002, ISSN: 0021-9991. DOI: https://doi.org/10.1006/jcph.2002.7146.

[51] J. Hesthaven and S. Ubbiali, "Non-intrusive reduced order modeling of nonlinear problems using neural networks," *Journal of Computational Physics*, vol. 363, pp. 55–78, 2018, ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.02.037.

[52] H. F. S. Lui and W. R. Wolf, "Construction of reduced-order models for fluid flows using deep feedforward neural networks," *Journal of Fluid Mechanics*, vol. 872, pp. 963–994, Jun. 2019, ISSN: 1469-7645. DOI: 10.1017/jfm.2019.358.

[53] M. Salvador, L. Dedè, and A. Manzoni, "Non intrusive reduced order modeling of parametrized pdes by kernel pod and neural networks," *Computers & Mathematics with Applications*, vol. 104, pp. 1–13, 2021, ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2021.11.001.

[54] A. Ivagnes, N. Tonicello, P. Cinnella, and G. Rozza, "Enhancing non-intrusive reduced-order models with space-dependent aggregation methods," English, *Acta Mechanica*, 2024, ISSN: 00015970. DOI: 10.1007/s00707-024-04007-9.

[55] N. Iyengar, D. Rajaram, and D. Mavris, "Domain Decomposition Strategy for Combining Nonlinear and Linear Reduced-Order Models," en, *AIAA Journal*, vol. 62, no. 4, pp. 1375–1389, Apr. 2024, ISSN: 0001-1452, 1533-385X. DOI: 10.2514/1.J063361.

[56] D. Xiao, F. Fang, C. Heaney, I. Navon, and C. Pain, "A domain decomposition method for the non-intrusive reduced order modelling of fluid flow," *Computer Methods in Applied Mechanics and Engineering*, vol. 354, pp. 307–330, 2019, ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2019.05.039.

[57] R. Everson and L. Sirovich, "Karhunen–loève procedure for gappy data," *J. Opt. Soc. Am. A*, vol. 12, no. 8, pp. 1657–1664, Aug. 1995. DOI: 10.1364/JOSAA.12.001657.

[58] J. Baiges, R. Codina, and S. Idelsohn, "A domain decomposition strategy for reduced order models. application to the incompressible navier–stokes equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 267, pp. 23–42, 2013, ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2013.08.001.

[59] D. J. Lucia, P. I. King, and P. S. Beran, "Domain decomposition for reduced-order modeling of a flow with moving shocks," *AIAA Journal*, vol. 40, no. 11, pp. 2360–2362, 2002. DOI: 10.2514/2.1576.

[60] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000. DOI: 10.1126/science.290.5500.2319.

[61] I. Prusak, D. Torlo, M. Nonino, and G. Rozza, "An optimisation–based domain–decomposition reduced order model for parameter–dependent non–stationary fluid dynamics problems," en, *Computers & Mathematics with Applications*, vol. 166, pp. 253–268, Jul. 2024, ISSN: 08981221. DOI: 10.1016/j.camwa.2024.05.004.

[62] R. Zahn, A. Weiner, and C. Breitsamter, "Prediction of wing buffet pressure loads using a convolutional and recurrent neural network framework," *CEAS Aeronautical Journal*, vol. 15, no. 1, pp. 61–77, Jan. 2024, ISSN: 1869-5590. DOI: 10.1007/s13272-023-00641-6.

[63] Y. Ohmichi, T. Ishida, and A. Hashimoto, "Modal decomposition analysis of three-dimensional transonic buffet phenomenon on a swept wing," *AIAA Journal*, vol. 56, no. 10, pp. 3938–3950, 2018. DOI: 10.2514/1.J056855.

[64] Y. Ohmichi, T. Ishida, and A. Hashimoto, "Numerical investigation of transonic buffet on a three-dimensional wing using incremental mode decomposition," *55th AIAA Aerospace Sciences Meeting*, Jan. 2017. DOI: 10.2514/6.2017-1436.

[65] S. Timme, "Global instability of wing shock-buffet onset," *Journal of Fluid Mechanics*, vol. 885, A37, 2020. DOI: 10.1017/jfm.2019.1001.

[66] H. Asada and S. Kawai, "Pod and dmd of three-dimensional transonic aircraft buffet using large-scale les data," *AIAA SCITECH 2024 Forum*, Jan. 2024. DOI: 10.2514/6.2024-0494.

[67] J. W. Slater, *ONERA M6 Wing*, https://www.grc.nasa.gov/www/wind/valid/m6wing/m6wing.html, Accessed: 2025-04-23, 2021.

[68] J. Kok, *User guide of ENSOLV/ENSENS: A flow and adjoint solver for aerodynamic, aeroelastic, and aeroacoustic applications using 3D multi-block structured grids*, Version 9.11, NLR - Netherlands Aerospace Centre, Apr. 2021.

[69] A. Bezbochina, E. Stavinova, A. Kovantsev, and P. Chunaev, "Enhancing predictability assessment: An overview and analysis of predictability measures for time series and network links," *Entropy*, vol. 25, no. 11, 2023, ISSN: 1099-4300. DOI: 10.3390/e25111542.

[70] M. Banerjee and N. R. Pal, "Feature selection with svd entropy: Some modification and extension," *Information Sciences*, vol. 264, pp. 118–134, 2014, Serious Games, ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2013.12.029.

[71] H. Yu and A. D. H. and, "A robust spearman correlation coefficient permutation test," *Communications in Statistics - Theory and Methods*, vol. 53, no. 6, pp. 2141–2153, 2024. DOI: 10.1080/03610926.2022.2121144.

[72] N. B. Gallagher, "Savitzky-golay smoothing and differentiation filter," *Eigenvector Research Incorporated*, 2020.

[73] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[74] S. Kohli, S. Miglani, and R. Rapariya, "Basics of artificial neural network," *International Journal of Computer Science and Mobile Computing*, vol. 3, no. 9, pp. 745–751, 2014.

[75] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," *Annual Review of Fluid Mechanics*, vol. 52, no. Volume 52, 2020, pp. 477–508, 2020, ISSN: 1545-4479. DOI: https://doi.org/10.1146/annurev-fluid-010719-060214.

[76] AFIT Data Science Lab R Programming Guide, *Feedforward deep learning models*, 2018.

[77] B. Genç and H. Tunc, "Optimal training and test sets design for machine learning," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 27, 2019. DOI: https://doi.org/10.3906/elk-1807-212.

[78] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2022. arXiv: 1312.6114 [stat.ML].

[79] J. Lu, C. Zhang, B. Li, Y. Zhao, R. Choudhary, and M. Langtry, "Self-attention variational autoencoder-based method for incomplete model parameter imputation of digital twin building energy systems," *Energy and Buildings*, vol. 328, 2025, ISSN: 03787788. DOI: 10.1016/j.enbuild.2024.115162.

[80] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994. DOI: 10.1109/72.279181.

[81] A. Graves, "Long short-term memory," in *Supervised Sequence Labelling with Recurrent Neural Networks.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 37–45, ISBN: 978-3-642-24797-2. DOI: 10.1007/978-3-642-24797-2_4.

[82] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019, ISSN: 0899-7667. DOI: 10.1162/neco_a_01199.

A

Additional Literature: Deep Learning

Deep learning (DL) is a subset of artificial intelligence (AI) and machine learning (ML) that focuses on developing algorithms that learn patterns and representations directly from data, instead of using mathematical or physical models [13]. This is used to solve tasks that are impossible to solve with fixed programs based on predefined functions [73].

As mentioned before, artificial neural networks are widely used in the field of deep learning. They are inspired on the neurons in the human brain, which have properties than can be applied for artificial simulations. The basic working principle of a biological neuron is to perform a non-linear operation on its combined inputs and transfers the result to other neurons [74], [75]. In the human mind, these neurons are all clustered together to form a three-dimensional neural network with interconnections and information sharing. Artificial neural networks on the other hand, are two-dimensional with a finite number of layers and neurons.

Figure A.1 shows a detailed deep feedforward neural network that maps y based on input values x. With fully connected (FC) or dense layers, like in this NN, each individual neuron is connected to all the neurons in the next layer making it a deep neural network (DNN). This creates a highly interconnected network making information between al the neurons possible. Each neuron in the NN applies a linear transformation and a non-linear activation to the inputs from the previous layer. Function z, shown in Equation A.1 [32], shows the linear transformation that the neurons in layer l apply to the outputs of the neurons in the previous layer. In this function, z is the weighted sum, W is the weight matrix, a is the output vector of the previous layer, and b is the bias vector. Subsequently, a non-linear activation function f is applied to z, which can be a different function depending on the use case. This non-linear activation function must be applied in every layer because otherwise the whole system of layers is linear, meaning that it could be represented by a single equivalent layer [73]. As a result, the NN would not be able to learn complex non-linear patterns.

$$\begin{aligned} \mathbf{z}^{[l]} &= \mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}, \\ \mathbf{a}^{[l]} &= f^{[l]}(\mathbf{z}^{[l]}) = \begin{cases} \mathrm{ReLU}(z) = \max(0, z), & \text{or} \\ \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, & \text{or} \\ \sigma(z) = \frac{1}{1+e^{-z}}. \end{cases} \end{aligned} \tag{A.1}$$

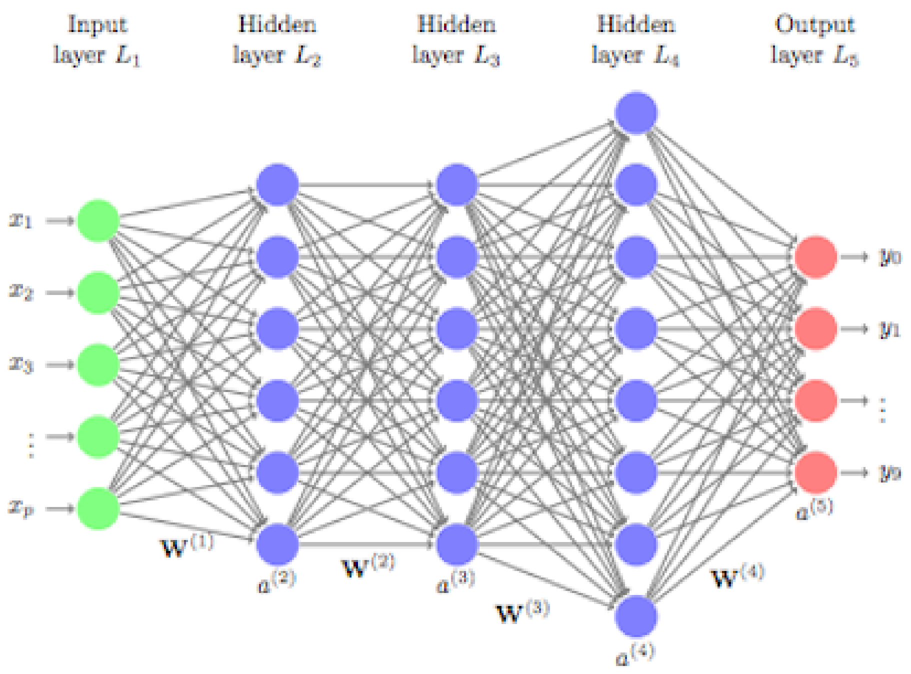


Figure A.1: Detailed structure of a deep feedforward neural network [76]

The weights w and biases b are initially defined with random values, but optimized in an iteratively during the training process. The error, or loss, which is the difference between high-fidelity solution and NN output is evaluated at every iteration, or epochs, whereafter the weights and biases are updated. Equation A.2 shows the mean-square-error (MSE) loss function which can be used to calculate this error [73] with m snapshots. Subsequently, the weights and biases can be updated in different ways, the steepest descent method being one of them. With this method, the parameters are changed in the opposite direction of the maximum error gradient [26]. Equation A.3 describes the steepest descent method. In this equation, $\nabla F(\mathbf{a_n})$ represents the NN loss gradient with parameters $\mathbf{a_n}$. Moreover, γ denotes the learning rate.

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(\hat{\mathbf{y}} - \mathbf{y})_i^2 \tag{A.2}$$

$$\mathbf{a_{n+1}} = \mathbf{a_n} - \gamma \nabla F(\mathbf{a_n}) \tag{A.3}$$

When the ReLU, or rectified linear unit, function is chosen as activation function, negative values become zero and positive values stay the same [73]. By using this function, the system remains piecewise linear, ensuring that the system generalizes well and its optimization can be done with gradient based methods. On the other hand, the tanh function squeezes the output between -1 and 1, and the sigmoid function between 0 and 1. As a result, the output of both functions is not sensitive to changes in very small or large values, but only with input values close to 0. However, because of this property, gradients are vanishing for large values making gradient based learning methods difficult [73].

In addition, the training process of a DNN must be designed with care, since it determines how well the DNN performs with constructing outputs for unseen data. First of all, it is important to introduce two type of errors that need to be taken into account, the training and generalization (or test) error [73]. The generalization error must be low in order to generate highly accurate results for unseen inputs, but this is not always achieved by minimizing the training error. To determine both errors, a dataset is divided into the following categories, which can be done randomly or regularized [77]:

- **Training set -** Largest part of the dataset, used to train the model by updating the weights and biases. Usually 70% of dataset.
- **validation set -** Small part of the dataset, used to assess the generalization error by reconstructing the data for unseen inputs. If error is too large, model parameters are revised once again. Usually 15% of dataset.
- **Test set -** Small part of the dataset, used to assess the revised generalization error by reconstructing the data for unseen inputs. Can't be used to revise the parameters. Usually 15% of dataset.

Dividing the dataset over these categories determines how well the model is trained for unseen data. When not all characteristics are present in the training set, this could lead to under-representation and a model that does not perform well for unseen data [77]. This performance of the network can be evaluated by evaluating the training error of the training set and the generalization error of the test set. Two factors determine this performance. First of all, the training error must be small. Secondly, the difference between training and generalization error must be small [73].

Figure A.2 shows the model's performance for different combinations of these two factors. If the first factor is not obeyed, the training set is not reconstructed accurately, as shown on the left. If the second factor is not true, the model does not generalize well and the error for unseen data will be too large, which is shown on the right. These concepts are called under- and overfitting, which are important training considerations. A model can be adapted to reduce the risk for under- or overfitting by changing its capacity, which is its capability to model complex tasks. If the capacity is too high for its use case, the model will have a bigger risk over overfitting. A lower capacity on the other hand will increase the risk of underfitting. In general, the training error will keep decreasing for a higher capacity. However, this is undesirable since the generalization error has a minimum for a moderate capacity. This means that a training error needs to be found that leads to the smallest difference from the generalization error.

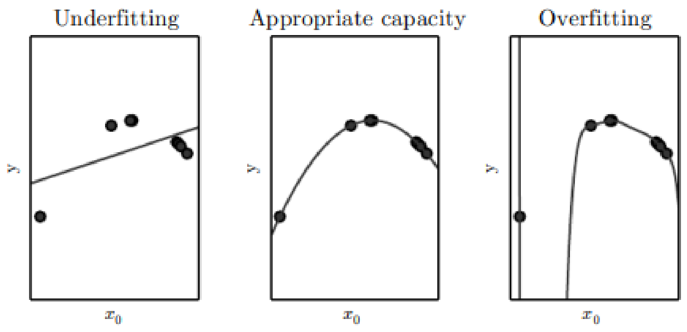


Figure A.2: Three different models fitted to an example training set [73]

Variational Autoencoder:
Another important consideration in the use of an AE is its architecture, which is to be determined beforehand by the user [30]. However, there is no systematic way to determine the number of layers, neurons, and dimension of the latent space, unlike POD where the conserved

energy portion denotes the number of modes. Moreover, the dimension of the latent space cannot be reduced afterwards since the distribution of information over these nodes is very irregular and not hierarchical, meaning that they are all needed to reproduce the solution [30]. As an improvement to trial and error to determining the dimension of the latent space, variational autoencoders (VAE) were introduced [78]. Moreover, they create a latent space that is structured which is more robust and allows for better interpolation.

A VAE creates a latent space that consists of probability distributions instead of true values [78], [79]. The encoder is manipulated such that the probability of latent parameter z based on input data x follows a probability distribution. Calculating the actual probability distribution, also called the true posterior $p_\phi(z \mid x)$, is hard since this would require taking into account all the possible latent parameters z. That is why an approximate posterior $q_\phi(z \mid x)$ is used which follows a normal, or Gaussian distribution. Note that ϕ are the biases and weights of the encoder. Equation A.4 [78] displays the multivariate Gaussian with a diagonal covariance structure which is typically used for the approximate posterior. Here, μ and σ are the mean and standard deviation (SD) of the Gaussian which are outputs of the encoder [79]. Moreover, the latent variables z can be sampled from the Gaussians with Equation A.5, where $\mathcal{N}(0, I)$ is a standard normal distribution.

$$\log q_\phi(z \mid x^{(i)}) = \log \mathcal{N}(z; \mu^{(i)}, \sigma^{2(i)} I) \tag{A.4}$$

$$z^{(i,l)} = \mu^{(i)} + \sigma^{(i)} \cdot \epsilon^{(l)}, \quad \epsilon^{(l)} \sim \mathcal{N}(0, I) \tag{A.5}$$

The decoder of a VAE is given by $p_\theta(x \mid z)$, which is trained to maximize the likelihood of x for a latent space z. In this expression, θ denotes the biases and weights of the decoder. As opposed to the regular autoencoder, the VAE has two loss terms as shown in Equation A.6 [78], also called the evidence lower bound (ELBO) derivation [79]. In this equation, the reconstruction term is similar to that of a regular AE and denotes the difference between the input data x and reconstructed output data $\hat{x}$. Here, $\mathbb{E}$ denotes the expectation that $\hat{x}$ is equal to x. The Kullback-Leibler (KL) divergence loss term measures the difference between the pre-set Gaussian approximate posterior and the true probability distribution of the latent parameters [79]. The objective of a VAE is to maximize this function, such that the lower bound $\mathcal{L}$ is maximized [73].

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(\theta, \phi; x^{(i)}) = \underbrace{-D_{\mathrm{KL}}(q_\phi(z \mid x^{(i)}) \parallel p_\theta(z))}_{\text{KL divergence term}} + \underbrace{\mathbb{E}_{q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)} \mid z)\right]}_{\text{reconstruction term}} \tag{A.6}$$

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are NNs with a more complex architecture, allowing them to learn recurrent patterns in the data over time [26]. This creates the capability to construct the solution based on the order of inputs in previous time steps. This capability is needed since, especially in transonic flow, the flow field is highly depended on previous time instances. For example in transonic buffeting resulting from shock wave–boundary layer interaction (SBLI), where the shock position is time dependent [2].

A commonly used type of RNN is that of Long Short-Term Memory (LSTM) due to its handling of instabilities introduced in regular RNNs [49]. When parameters of the network are trained through back propagation, the gradients of long-term dependencies between input and output can vanish or explode over time due to the many multiplication operations of new inputs

[80], [81]. This means that long-term decencies are not taken into account well due to the strong influence of short-term inputs. The proposed LSTM network solves this problem by incorporating a gating mechanism that maintains both long- and short-term memory [26]. This is used to decide which characteristics of previous time steps are relevant for constructing the solution for the current time step.

In a LSTM network, the hidden layers of a FNN are replaced by LSTM memory cells. A LSTM cell has three gates: the input, output, and forget gate. The training information flows through these gates to control it and maintain long-term dependencies [49]. The input gate regulates the new information stored in the cell, the output gate the part of the memory that is passed on to the next cell, and the forget gate the information that is removed. Figure A.3 shows the architecture of this LSTM cell, where the input, output, and forget gates are denoted by i(t), o(t), and f(t) respectively. Moreover, the cell state is denoted by C(t) and the updated cell state by $\tilde{C}(t)$. Lastly, the cell output is denoted by h(t) and the cell input by x(t). Lastly, the half circles denote the bias added to the signals.

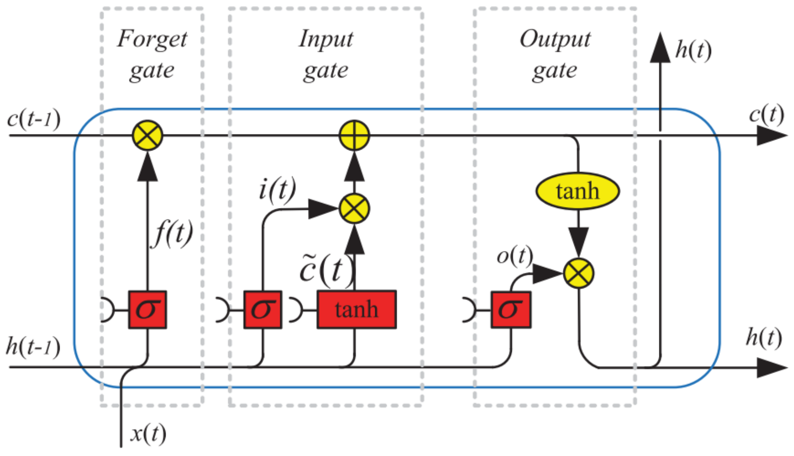


Figure A.3: Single LSTM cell architecture. c(t−1) and c(t) are the previous and current cell's memory respectively, h(t−1) and h(t) are the previous and current cell's output respectively, and x(t) is the input vector [82]

In order to calculate gate values and cell state and output, Equation A.7 till Equation A.12 can be used [49]. In these equations, W_f denote the weight of each gate and b_f the bias. As can be seen in this set of equations, the values of the gates are calculated by applying an activation function σ to the weighted and biased input vector, just like in regular neurons. However, the cell's output of a previous time step is added to the input as well.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{A.7}$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{A.8}$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{A.9}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{A.10}$$

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{A.11}$$

$$h_t = o_t * \tanh(C_t) \tag{A.12}$$

B

Additional Results

POD modes with and without enrichment

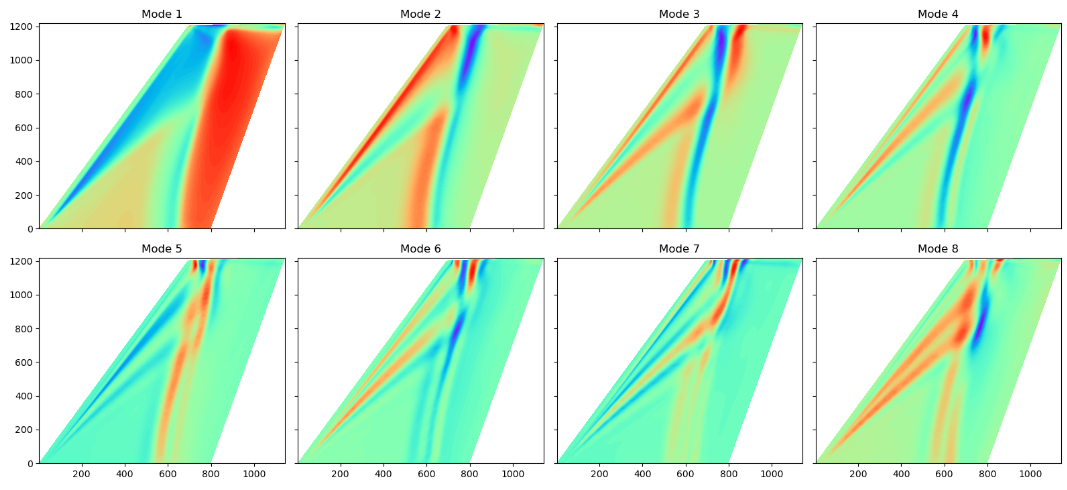


Figure B.1: First 8 POD modes

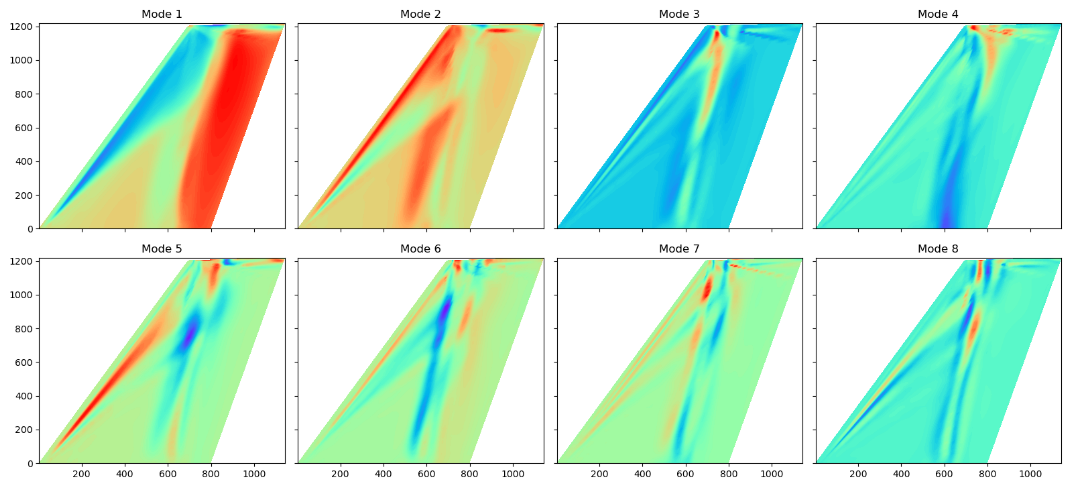


Figure B.2: First 8 ePOD modes maximum enrichment

Energy fraction of POD modes

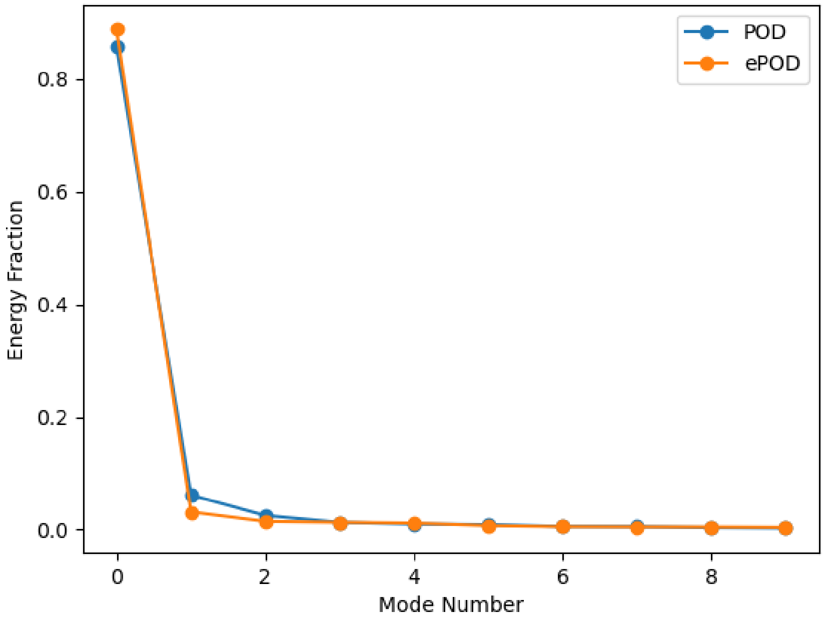
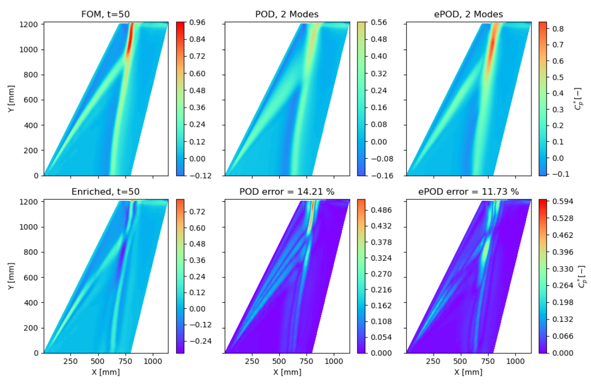
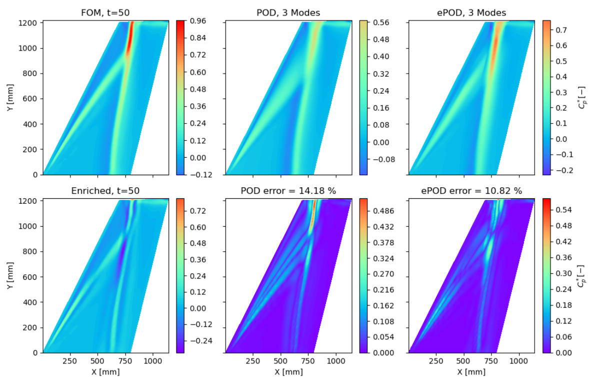


Figure B.3: Energy fraction of first 10 POD modes, maximum enrichment

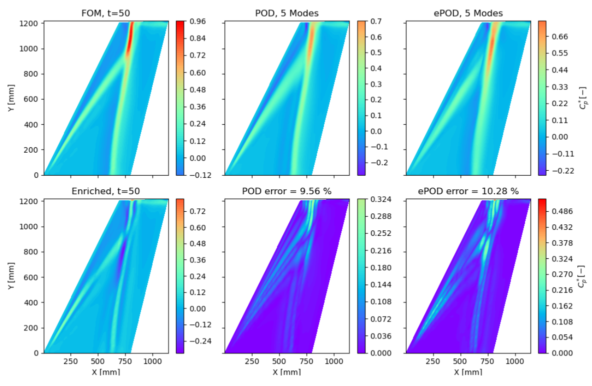
Projection error visualized



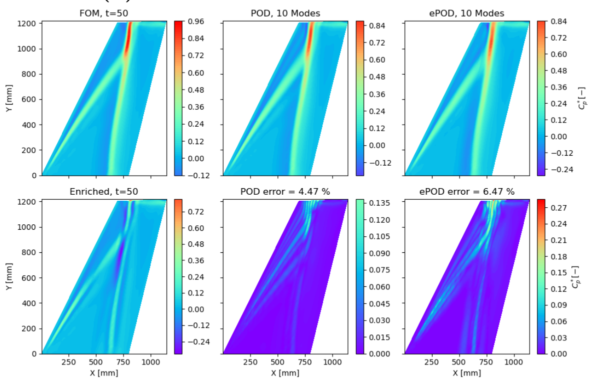
(a) Reconstruction with 2 POD modes



(b) Reconstruction with 3 POD modes



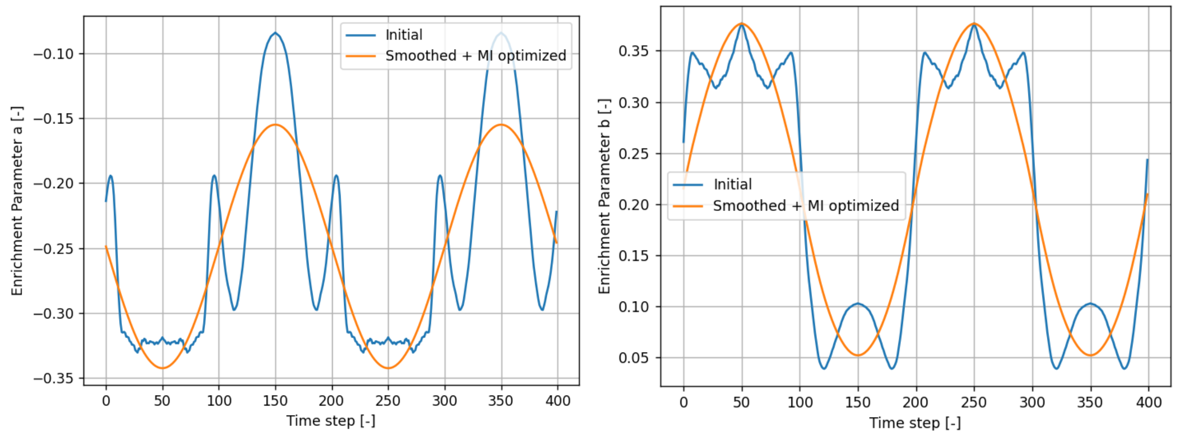
(c) Reconstruction with 5 POD modes



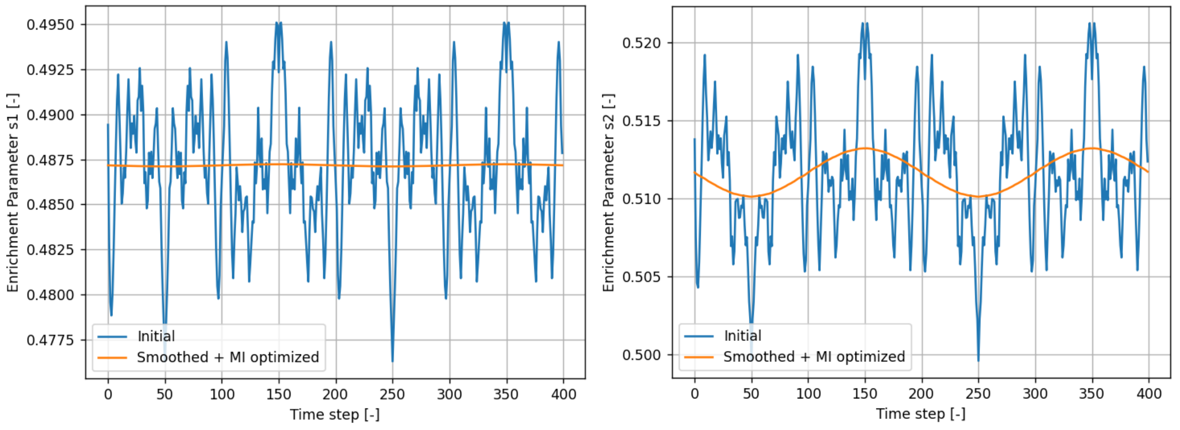
(d) Reconstruction with 10 POD modes

Figure B.4: Flow field reconstruction for a variety of POD modes, time step = 50

Optimized enrichment coefficients



(a) Enrichment parameter a **(b)** Enrichment parameter b



(c) Enrichment parameter s1 **(d)** Enrichment parameter s2

Figure B.5: Optimization of enrichment time coefficients

Optimized Enrichment Reconstruction Error

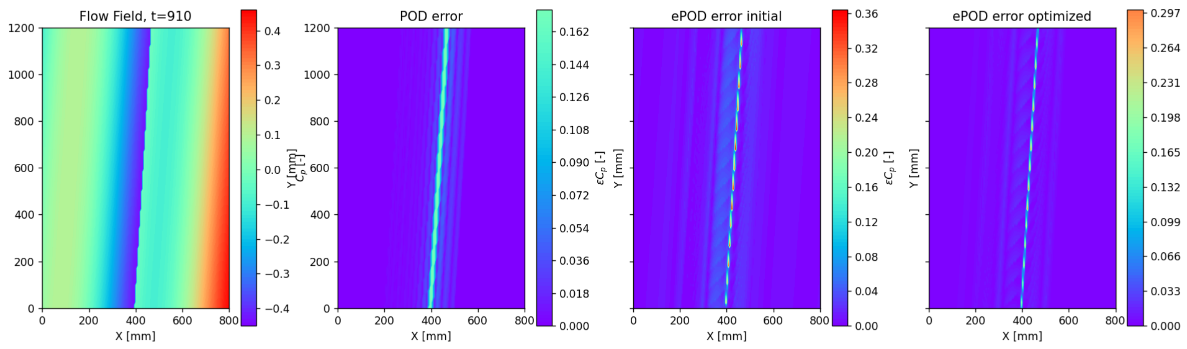


Figure B.6: Total error of predicted pressure distribution at t=910

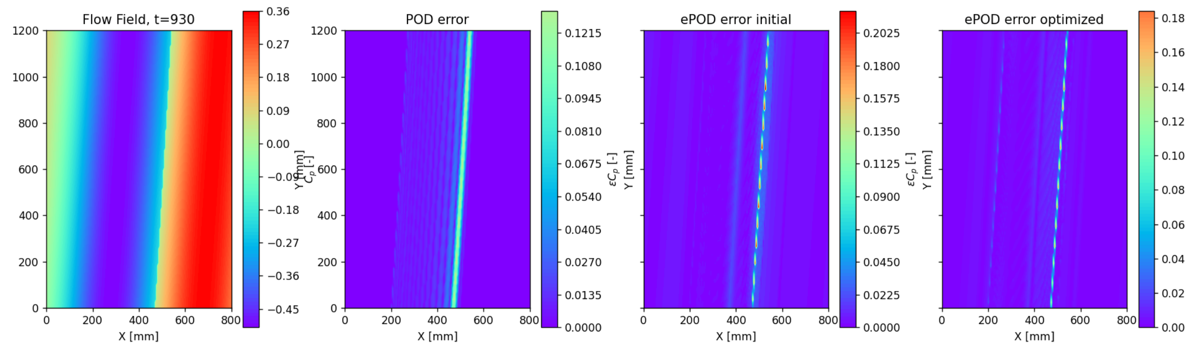


Figure B.7: Total error of predicted pressure distribution at t=930

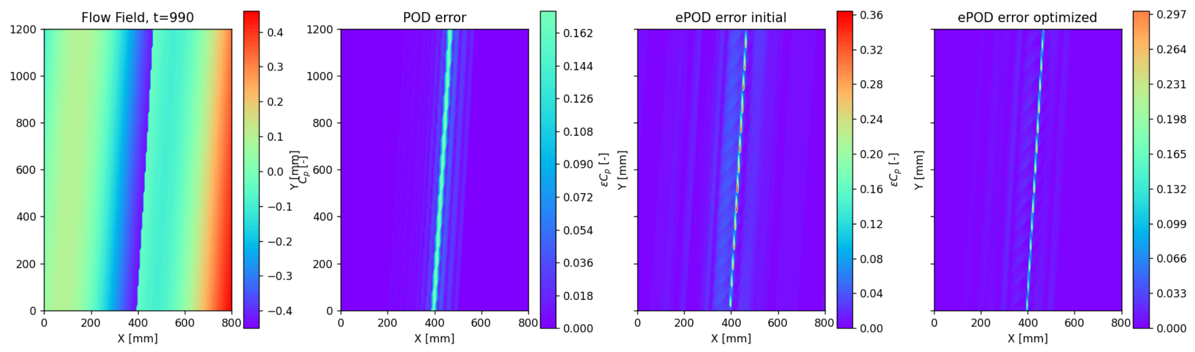


Figure B.8: Total error of predicted pressure distribution at t=990