Isak August Solberg
Henrik Andreas Gusdal Wassertheurer

DMD as a substitute for computational fluid dynamics for wind parks

Bachelor's project in Renewable Energy
Supervisor: Tania Bracchi
Co-supervisor: Balram Panjwani & Benjamin Adrian
May 2021

Bachelor's project

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Energy and Process Engineering

NTNU
Kunnskap for en bedre verden

SINTEF

Isak August Solberg
Henrik Andreas Gusdal Wassertheurer

DMD as a substitute for computational fluid dynamics for wind parks

Bachelor’s project in Renewable Energy
Supervisor: Tania Bracchi
Co-supervisor: Balram Panjwani & Benjamin Adrian
May 2021

Norwegian University of Science and Technology
Faculty of Engineering
Department of Energy and Process Engineering

Preface

This dissertation is submitted as the final part of the 3 year Bachelor programme in Renewable Energy Engineering at the Norwegian University of Science and Technology. The Bachelor thesis is valued at 20 credits. The dissertation is a collaboration between two students, Isak August Solberg and Henrik Andreas Gusdal Wassertheurer.

The industry partner for this project is SINTEF, and the project statement was developed in collaboration with Balram Panjwani and Benjamin Adrian.

The controlled outlining of wind farm capability within the industry is something that at current date does not have a set standard. Contributing to the development of such a tool has been a strong motivation for writing this dissertation.

Being able to utilise a state of the art tool, based on modern advances in science, has been both a challenging and rewarding experience. Our primary challenge has been the fact that sources for the theory has been somewhat limited due to the program code used being written in 2019 based on PyDMD which was developed in 2017.

This dissertation is based on the experimental simulation data provided by Balram Panjwani through OpenFOAM. The DMD abbreviation code used in the thesis is created by Benjamin Adrian for the Upwards project. The layout of the wind turbines were decided upon by Isak August Solberg and Henrik Andreas Gusdal Wassertheurer in conjunction with the guidance councilors. Modifications to the original code were done by Isak August Solberg and Henrik Andreas Gusdal Wassertheurer.

The dissertation is meant to target renewable energy companies, research companies and universities that focus on the improvement of wind turbines and understanding the related physics.

We would like to give our sincerest gratitude to our internal supervisor, associate professor Tania Bracchi. We would also like to thank our external supervisors, Balram Panjwani senior scientist at SINTEF, and Dr Benjamin Adrian from Fraunhofer Institute for Industrial Mathematics.

Isak August Solberg

Isak August Solberg
Bærum, 22.04.2020

Henrik Wassertheurer

Henrik Andreas Gusdal Wassertheurer
Kristiansand, 22.04.20

Abstract

On a global scale, the demand for energy is rising. Combined with the issue of global warming the appeal of renewable energy sources increases. One such source is wind energy, but the state of the art is far from perfect. The efficiency of wind turbines could greatly benefit from an increase in the use of Computational Fluid Dynamics (CFD), though this is often forgone due to its time-consuming nature and the hardware requirements. In this thesis, a Dynamic Mode Decomposition (DMD) tool is applied to investigate its utility as an efficient substitute for the traditional CFD method using OpenFOAM simulation.

The examination was done by performing DMD on a total of six different CFD simulations. Three cases were analysed for two different velocities. The cases consists of a single turbine, two turbines in tandem and a three turbine approximation of the wind park Hywind Scotland. DMD was performed on the simulations and interpolated giving an assessment foundation to compare CFD simulations and DMD reconstruction.

The reconstructed systems were analysed finding the average and peak error rates compared to CFD simulations of equivalent wind velocities. Each reconstruction was examined thoroughly in the XY, XZ and YZ-plane at the time of its peak overall error rate. Wind velocities at turbines locations were scrutinised to investigate the ramifications of the velocity errors for power production.

The results obtained indicate a direct correlation between a simulation's complexity and the reconstruction's error rate, both regarding the velocities and power produced at the turbines. When a simulation of two turbines was analysed compared to a simulation with only one, the overall error rates grew from approximately 2% to 4%. The three turbine simulations had an overall error rate of approximately 2.5%, but this decrease was due to an increased simulated volume. A considerable amount of this volume consisted of relatively undisturbed flow.

The single most precise power prediction had an error rate of less than 1% at one turbine. The least accurate power prediction overestimated the power production at one of the turbines with an alarming error rate of 49.62%.

Given a reconstruction with an acceptable error rate, the time saves would be tremendous. The time spent creating a single OpenFOAM simulation ranged from 12 to 24 hours. While one OpenFOAM simulation is a necessity for DMD reconstructions, the time spent creating a DMD reconstruction from OpenFOAM flow field data was approximately 50 minutes for the most time-consuming system in this thesis. Furthermore, expansion of the DMD to another velocity was done in 19.31 seconds for the most comprehensive reconstruction.

The significance of these findings is not outright conclusive, as adjustments to the simulations are necessary to approach realistic conditions. Further testing is required to irrevocably deduce the accuracy boundaries of DMD and how to successfully implement DMD in the wind energy industry.

Sammendrag

I et globalt perspektiv øker etterspørselen etter energi, og kombinert med problemstillingen rundt global oppvarming er fornybare energikilder et attraktivt alternativ. Et eksempel på en slik kilde er vindkraft, men dagens teknologi er langt fra perfeksjonert. Effektiviteten til nye vindparker vil kunne bedres hvis Computational Fluid Dynamics (CFD) benyttes i større grad, men dette forbigås ofte grunnet hvor tidkrevende det er og kravene til maskinkraft. I denne tesen benyttes Dynamic Mode Decomposition (DMD) som et effektivt alternativ til CFD-analyseverktøyet OpenFOAM.

Undersøkelsen ble gjort ved å utføre DMD på totalt seks forskjellige CFD-simuleringer. Simuleringer ble utført på tre turbinoppsett ved to hastigheter: først en enkelt turbin, deretter to turbiner i tandem, og til slutt en approksimasjon av tre av turbinene i parken Hywind Scotland. DMD ble utført på simuleringene og interpolert for å danne et vurderingsgrunnlag for sammenlikning mellom DMD og CFD.

De rekonstruerte systemene ble analysert ved å finne den gjennomsnittlige- og høyeste feilmarginen. Hver rekonstruksjon ble undersøkt i XY-, XZ- og YZ-planet på tidspunktet med maksimal feilmargin. For å undersøke konsekvensene for produsert effekt ble vindhastigheter på turbinlokasjonene undersøkt.

Resultatene oppnådd indikerer en direkte sammenheng mellom en simulerings kompleksitet og rekonstruksjonens feilmargin. Dette var tilfellet både for vindhastigheter og produsert effekt ved turbinene. Simuleringene med to turbiner hadde en total feilmargin på omtrent 4%, sammenliknet med 2% for simuleringene som kun inkluderte en turbin. Simuleringene med tre turbiner hadde en total feilmargin på cirka 2.5%. Denne reduksjonen skyldes imidlertid en økning i simuleringsvolum hvor strømningen i et betydelig område er uforstyrret.

Rekonstruksjonen med høyest effekt-presisjon for èn turbin hadde en feilmargin på under 1%. Rekonstruksjonen med det høyeste avviket for produsert effekt hadde et avvik ved en av turbinene på alarmerende 49.62%.

Gitt en rekonstruksjon med akseptabel feilmargin blir tidsbesparelsene enorme. Tiden brukt på å lage en OpenFOAM-simulering var mellom 12 og 24 timer. En enkelt OpenFOAM-simulering er nødvendig for å produsere DMD-rekonstruksjoner. Prosessen å lage rekonstruksjonen fra strømningsdataen tok omtrent 50 minutter for den mest tidkrevende rekonstruksjonen i denne tesen. Å utvide DMD videre til enda en ny hastighet ble gjennomført på 19.31 sekunder i det mest omfattende tilfellet.

Betydningen av funnene i denne tesen er ikke kategorisk konklusive, ettersom det er nødvendig å gjennomføre justeringer av simuleringene for å tilnærme seg realistiske vindforhold. Ytterligere testing er nødvendig for å verifisere DMDs begrensninger ved implementering i vindenergisektoren.

List of symbols

A	Area	$[m^2]$
a	Axial induction factor	-
C_P	Power coefficient for wind turbines	-
E	Unitary square matrix	-
L^2	Matrix norm	-
P	Power	[W]
S	Rectangular diagonal matrix	-
U	Mean air flow velocity	[m/s]
v_{eq}	Equivalent velocity	[m/s]
V	Square complex unitary matrix	-
X	Complex matrix	-
α	Angle of attack	[°]
ρ	Density	$[kg/m^3]$

Glossary

Actuator disc	Theoretical turbine with an infinite number of blades
Ashes	Wind turbine simulation tool
Complex matrix	A matrix consisting of complex elements
Diagonal matrix	A matrix where elements not on the main diagonal are zero
Dynamic Mode Decomposition	A dimensional reduction algorithm
Eigenvalues	A scalar associated with linear equation systems
Flow tracer	Any tool that traces the flow in a fluid
High-dimensional data	Data where the number of dimensions are so large that calculations are impractical
OpenFOAM	Fluid flow simulation tool
Particle image velocimetry	A flow visualisation method
Snapshot	Visualisation at a single point in time
Square and rectangular matrix	Matrix shaped $\boldsymbol{m\ x\ m}$ if square and $\boldsymbol{m\ x\ n}$ if rectangular.
Unitary matrix	A complex square matrix with a conjugate transpose matching its inverse

Abbreviations

ABL	Atmospheric Boundary Layer
CFD	Computational Fluid Dynamics
CSV	Comma Separated Values
GHG	Greenhouse Gases
HAWT	Horizontal Axis Wind Turbine
HS	Hywind Scotland
NREL	National Renewable Energy Laboratory
PyDMD	Python Dynamic Mode Decomposition
REWS	Rotor Equivalent Wind Speed
SVD	Singular Value Decomposition

Contents

List of Figures

List of Tables

Introduction

Of the many challenges mankind will face in the future, there is an overwhelming probability that global warming and climate change will be of utmost importance. The main accelerator of this process is emissions of greenhouse gasses (GHG). To combat this development, more than 190 countries have signed the Paris agreement. The Paris agreement is a legally binding treaty, with the main goal being to keep global warming below 2 °C, compared to pre-industrial levels (European Union, 2021b).

The majority of man-made GHG emissions are derived from the energy sector (Ritchie & Roser, 2020). The demand for energy on a global scale is on a steady incline, so energy production must follow suit. Partially due to the pressure to reduce GHG emissions, wind energy extraction has on a global scale seen rapid growth in the last few decades (Weiss & Tsuchida, 2017, pp. 419–420). Despite this extraordinary increase in installed capacity, state of the art wind energy technologies is far from perfected.

The principle of extracting kinetic energy from the wind has been a well-known concept for centuries. However, the utilisation of offshore farms and their feasibility have so far been largely unexplored. An offshore wind park has several advantages compared to a traditional wind park. In general, the wind speeds are greater at sea than on land. Offshore wind parks are also less likely to be the source of noise and visual disturbance for the local population. It is also probable that offshore wind parks have less of a destructive impact on the local ecosystem. The major disadvantage for offshore wind parks is the increased cost associated with installation and maintenance.

Within the field of aerodynamics, proper understanding of fluid movements has long been something sought after. This can be observed through the continuous unsuccessful search for a general solution to the Navier-Stokes problem. Modern problems within aerodynamics are solved, not through a complete understanding of a general solution of the movement of fluids, but rather through using raw computer power to establish concurrent data, creating a realistic simulation. In this thesis an algorithmic reduction tool based on the Dynamic Mode Decomposition (DMD) was used to simulate fluid flow data. Furthermore, the results from DMD will be compared to data obtained by computational fluid dynamics (CFD), made with OpenFOAM. The desired end product is a baseline tool that reduces the simulation time and hardware requirements while retaining a reasonable degree of accuracy.

Part of the thesis work will be to determine the quantity of turbines for which the DMD simulation, at its current level, is valid. The wind turbines used were 5MW turbines as defined by National Renewable Energy Laboratory (NREL) (Jonkman et al., 2009). The thesis first introduces general aerodynamic concepts which are used in regards to wind energy. The utility value of DMD, its strengths and weaknesses were examined in comparison to the more traditional CFD approach. In the tools and methodology section the tools and hardware used are presented, as well as a general approach used for each case. A more in-depth explanation of the individual cases follows. Ultimately, the results are presented, discussed and conclusions based on the findings are drawn.

Commissioners

Upwards project

The work described in this report is a part of the EU project *Upwards*. The overall objective of this project is to improve the understanding of wind turbine-related physical phenomena through high-performance computing. This is deemed necessary in order to optimise the design of future turbines in line with technical, economic and societal demands. Three sub-goals are identified as key factors:

1. To advance the capacity and performance of existing wind turbines.
2. To accelerate the development, testing and implementation of newer turbines.
3. Consideration of feedback, both public and from stakeholders, in regards to the design of future wind turbines.

(European Union, 2021a)

SINTEF

SINTEF is a multidisciplinary research organisation. The institution has international top-level expertise in multiple fields, such as natural sciences, technology, social sciences and more. SINTEF is among the largest contract research institutions in Europe (SINTEF, 2021). In the *Upwards*-project, SINTEF acts as the coordinator, and has also been our method of ingress into this partnership. The group's contact in SINTEF is Balram Panjwani.

Fraunhofer-Gesellschaft

Fraunhofer-Gesellschaft was founded in 1949, and currently operates 74 institutes and research institutions throughout Germany. Fraunhofer plays an important role in innovation, by focusing on the development of technologies and enabling commercial exploitation of said work (Fraunhofer-Gesellschaft, 2021). The group's personal contact from Fraunhofer-Gesellschaft is Benjamin Adrian.

1 Theory

1.1 Wind energy

The fundamental principle for wind turbines is to utilise the kinetic energy in the wind. The kinetic energy is transformed into electrical energy by a turbine. The wind power P relies on the wind's velocity U, the surface area in question A, and the density of the air ρ, as given in equation 1.1 (Burton et al., 2011, p. 20).

$$P_{wind} = \frac{1}{2}\rho A U^3 \tag{1.1}$$

It is, however, impossible to harness all this power. The highest theoretical proportion obtainable by a wind turbine is known as Betz limit, attributed to Albert Betz (Betz, 2013). To calculate this limit, Betz employed the actuator disc model. The actuator disc is an ideal rotor with an infinite number of blades. The model is based on the Bernoulli equation, conservation of mass and linear momentum, as well as several assumptions, as listed in appendix A. The model is visualised in figure 1.1. Due to conservation of mass, U_2 must be equal to U_3. hence will the energy obtained by the rotor be equal to the difference in pressure over the rotor. (Manwell et al., 2009, pp. 92–96)

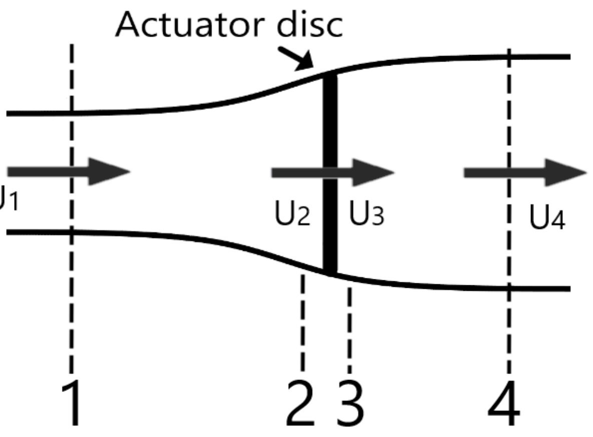


Figure 1.1: Actuator disc model.

Betz limit, equal to 16/27, is thoroughly derived in appendix A.

To compare the power in the wind and the power obtained by the rotor, the power coefficient C_P is introduced. The value of said coefficient cannot exceed Betz limit. As the value of C_P is dependent on the on-location conditions, a thorough analysis of the location's wind conditions is necessary to optimise the power production. Equation 1.2 shows the relationship between C_P, the rotor power and the power available in the wind (Burton et al., 2011, p. 43).

$$C_p = \frac{Rotor\ power}{Power\ in\ the\ wind} = \frac{P_{rotor}}{\frac{1}{2}\rho A U^3} \tag{1.2}$$

For optimal performance, C_P should be approaching Betz limit. However, there are several limiting factors. The most prominent are, wake rotation, the finite number of blades (with associated tip losses) and aerodynamic drag (Manwell et al., 2009, p. 96). Modern turbines typically have a peak C_P value between 0.4 and 0.5 (Hau & Renouard, 2013, p. 560). For the turbine examined in this report, the *NREL offshore 5-MW baseline wind turbine*, peak power coefficient was found to be 0.482 (Jonkman et al., 2009, p. 19). Combined with the unavoidable mechanical losses in the turbine, i.e., losses in the turbine generator and frictional losses, the turbine's overall efficiency will always be somewhat lower than the operating power coefficient at any given time.

The present industrial standard is a horizontal axis wind turbine (HAWT) with three blades. Among the deciding factors for this design choice are optimal operation speed (and the associated noise level), material costs, and the possibility to include pitch control. If the trend of increasingly larger wind turbines should continue, a return of a two-bladed standard is not unfathomable. (Hau & Renouard, 2013, pp. 69, 136)

1.2 Atmospheric Boundary Layer

The lowest part of the atmosphere is called the Atmospheric Boundary Layer (ABL). The behavior is heavily influenced by its proximity to the earth's surface (Pielke & Hayden, 2021). In this section of the atmosphere, physical characteristics such as temperature, humidity and velocity can change swiftly in space and time. In general, one can assume horizontal wind speed will increase as elevation increases. This is an important physical phenomenon for wind turbines, as wind velocity correlates with the kinetic energy in the wind. The ABL also has a tendency to resist vertical motion and suppress existing turbulence. (Manwell et al., 2009, pp. 36–37).

1.3 Wake

As a consequence of the interaction between the airflow and the wind turbine, a wake with decreased velocity and increased turbulence will be formed downwind of the turbine. The discrepancy in velocity between the wake and free flow results in additional shear-generated turbulence. (Burton et al., 2011, p. 34)

In the case of a rotating wind turbine rotor, the flow behind the rotor will be rotating in the opposite direction of the turbine's rotor. This is a reaction to the torque exerted by the flow on the rotor, according to Newton's third law of motion. The rotational kinetic energy in the wake is one of the major contributors to the deficit between Betz limit and real wind turbines' power coefficient. (Manwell et al., 2009, p. 96)

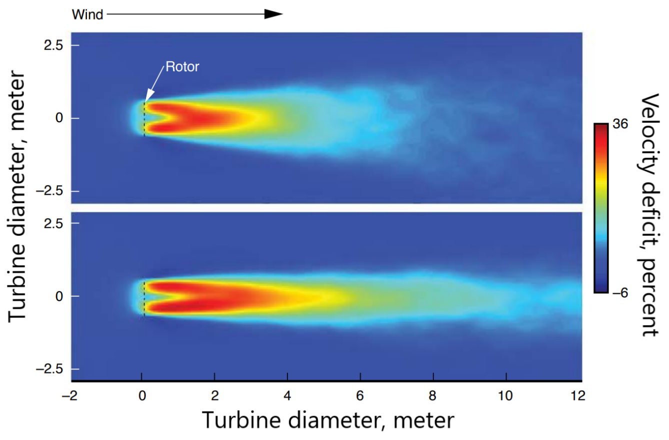


Figure 1.2: Turbine wakes visualised. Note the higher velocity loss at the perimeter of the turbines, and that an increase in atmospheric turbulence attenuates the turbine wake quicker. (Heller, 2014)

The most turbulent part of the wake is close to the turbine's edges, due to tip vortices, followed by the core of the wake. Disregarding the turbulent peaks generated by tip vortices, the turbulence is decreasing as the distance from the wake core increases. Over time, the wake and the surrounding flow mixes, and thus the velocity deficit in the wake erodes. Eventually, the airflow fully recovers, as can be observed in figure 1.2. The speed of which this occurs is dependent on the level of ambient turbulence (Burton et al., 2011, p. 34).

For wind turbines located in the wake of another turbine, there are several negative effects. First and foremost, due to a lower mean inflow velocity in the wake, the power output of the turbine in the wake will be reduced. Due to the increased turbulence, the downwind turbine experiences increased loads and fatigue. (Manwell et al., 2009, pp. 142, 428).

1.4 Wind energy infrastructure

Groups of concentrated wind turbines are a relatively new phenomenon, compared to wind turbines in a general sense. The first wind farms were developed in the United States in the late 1970s. The advantages are mainly economic, as it enables concentration of repair and maintenance equipment, as well as reducing the size of the electrical grid, roads etc. Favourable wind conditions are often located in restricted geographical areas, another argument in favour of wind parks. While the wind park is both cost-efficient and leads to better utilisation of the accessible wind, there are drawbacks to this installation method. Compared to singular turbines, a wind park demands more infrastructure, such as electrical grid, roads and data collection systems. A schematic example of a wind park layout is shown in figure 1.3. (Manwell et al., 2009, pp. 420–423)

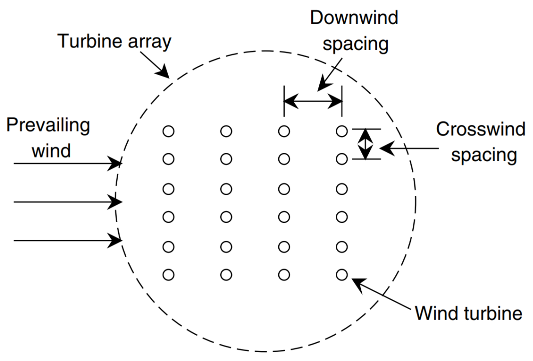


Figure 1.3: Wind farm layout exemplified. (Manwell et al., 2009, p. 421)

The electrical collection systems in wind parks generally operate at a higher voltage than the individual turbine's generators, as a means to decrease resistive losses in the grid. To accommodate this need, many modern turbines come with an installed transformer in the base of the turbine. When constructing roads in a wind park, grades and slopes must be gentle enough to allow transportation of turbine parts and heavy equipment to the turbine sites. The length of the rotor blades and/or tower sections are important to consider in this regard. (Manwell et al., 2009, p. 421)

1.5 Rotor Equivalent Wind Speed

Vertical wind shear across the rotor is known to have an impact on the rotor performance. To give more realistic power estimates than a calculation based on the wind speed at the hub, the Rotor Equivalent Wind Speed method (REWS) is utilised. In the REWS method, the swept rotor area is divided into sections, and the equivalent velocity is calculated as shown in equation 1.3,

$$v_{eq} = \left(\sum_{i=1}^{n_h} v_i^3 \frac{A_i}{A} \right)^{1/3} \tag{1.3}$$

where v_{eq} is the equivalent velocity, n_h is the number of height measurements, v_i is the average wind velocity over 10 minutes at height i, A is the complete area swept by the rotor and A_i is the area of i'th segment. An example of the segmentation, when used with five points, is illustrated in figure 1.4. (Wagner et al., 2014)

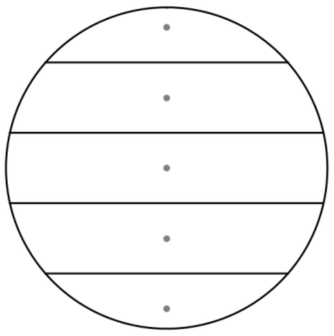


Figure 1.4: The segmentation used by the REWS method. (Wagner et al., 2014, p. 7)

1.6 Dynamic mode decomposition

Dynamic mode decomposition (DMD), created by Schmid (Schmid, 2010) is a decomposition method created to extract dynamic information from linear or non-linear flow fields through the analysis of either numerical data or visual experimentation such as flow tracers and particle image velocimetry. In general, the DMD attempts to simplify high-dimensional data in a way that limits the need for computational power while still providing data usable for short-term prediction of the flow changes in a system. When used on a linearised flow model the results are equivalent to the result of a global stability analysis. When used on a non-linear system the results are a string of linear tangent approximations which expresses the dominant tendencies in the flow. (Schmid, 2010)

1.6.1 L^2 norm

For an n-dimensional Euclidean space $\mathbb{R}$ the L^2 norm, also called the Euclidean norm or square norm, is the square root of the sum of the squared vector values. The L^2 norm is defined as shown in equation 1.4, where x is a point in a vector, matrix or flow field. (Li & Jain, 2009)

$$||x||_2 = \sqrt{x_1^2 + \cdots + x_n^2} \tag{1.4}$$

When used on a data set for wind speed the L^2 norm can be used to give a more accurate representation of the total wind speed in a system.

1.6.2 Singular value decomposition

Singular value decomposition (SVD) is in linear algebra a factorisation that generalises the eigendecomposition of a matrix. The SVD factorises a $\boldsymbol{m} \times \boldsymbol{n}$ complex matrix X as shown in equation 1.5.

$$X = ESV^* \tag{1.5}$$

Where E is a square unitary matrix, S is a rectangular diagonal matrix and V^* is a square complex unitary matrix. SVD has a wide variety of uses and abbreviations such as construction and modification of learning algorithms (Jankowski & Linowiecki, 2019), analysing the predictive power of the stock market (Gu et al., 2015), determining whether deep neural networks are viable for low-end smart devices (Astrid & Lee, 2018) and approximating a low-rank matrix.

1.6.3 Eckart-Young theorem

The approximation of a low-rank matrix using SVD is called the Eckart-Young theorem (Eckart & Young, 1936). The application lies in approximating a matrix X with a truncated matrix $\widetilde{X}$ with a given rank. By minimising the L^2 norm, while limiting the rank to a set one, the solution is given as can be observed in equation 1.6.

$$\widetilde{X} = E\widetilde{S}V^* \tag{1.6}$$

Here $\widetilde{S}$ only contains the r largest singular values, while the rest of the matrix is equal to zero. It is otherwise indistinguishable from S.

1.6.4 Eigenvalues

The eigenvalues of a system can be used to help describe the system. In figure 1.5 an example of the eigenvalue distribution of a dummy system that has had an SVD performed on it can be observed. Here the number of points is dictated by the SVD rank and the location gives information on the accuracy of the system. Although unrealistic an ideal DMD case would see all points stationed on the unit circle. Points located outside the unit circle means energy is being created, which is impossible, and indicates an error with the system. Eigenvalues stationed inside the unit circle indicate a dampening effect, which can skew the results, although the severity is dependant on the number of points and their proximity to the unit circle. In general, several low eigenvalues from real data can reasonably be ignored under the assumption that it is caused due to noise.

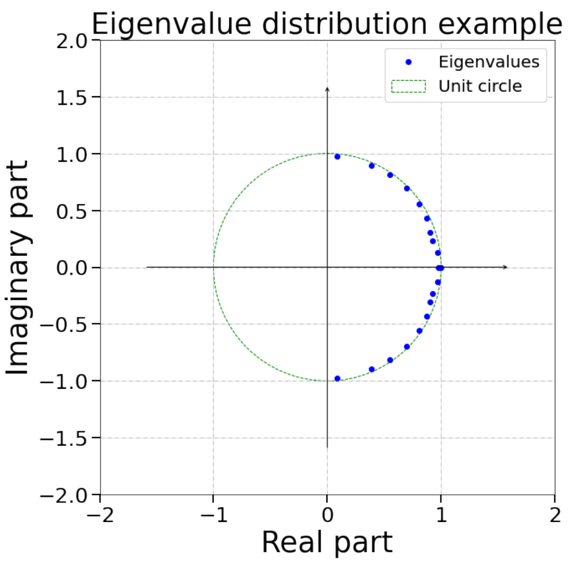


Figure 1.5: Eigenvalue example plot of a dummy-system with an SVD rank of 20.

2 Tools and Methodology

In the following section, the tools and methodology used for the thesis are presented and explained so as to give an insight into the approach and analysis performed. The section is split into multiple segments. First, a short introduction of the tools and, along with their gross properties is presented. Secondly, the tools used, including the python environment, Ashes and a description of OpenFOAM simulations converted is showcased. Further, a general approach for testing is set explaining the parts that are consistent throughout all tests. More in-depth explanations are then given, taking a closer look at each of the three individual cases with reasoning for the differing parameters in each case. For the sake of readability, the number of visualisations and plots shown in this section is severely limited compared to the ones used throughout the testing.

2.1 Tools and specifications

In this section, the tools and specifications are elaborated on. Wherever the NTNU and SINTEF hardware is mentioned, the Virtual machines (VM) that were used for testing are the ones described in section 2.1.3.

2.1.1 Paraview

Paraview is a tool used for the visualisation and simulation of data sets. It is open source and the standard OpenFOAM format is supported for easy visualisation. In this dissertation, Paraview is used for visualising each data set to ensure that the desired velocities and turbine locations are correct before proceeding with testing and conversion.

2.1.2 OpenFOAM

OpenFOAM is an open-source Computational fluid dynamics (CFD) software with users in a large number of engineering and science-oriented businesses and academic institutions. CFD is the use of numerical analysis to simulate and understand fluid flows. It is often used to get a better understanding of the unique interaction between fluid flows and defined surfaces. Complex simulations require a significant amount of computational power to achieve employable results with a high degree of accuracy. The results given from a CFD analysis are strongly dependant on various assumptions determined by the equations used. This establishes a large number of equations, like Navier-Stokes, laws of conservation, compression and Bernoulli, which in turn are used to give plausible simulations. The OpenFOAM simulations performed in this thesis were done using the URANS method.

2.1.3 Hardware

The hardware specifications for the two machines used for DMD testing, and OpenFOAM simulations, can be observed in table 2.1. The OpenFOAM simulations for this thesis are provided by SINTEF. During SINTEF's OpenFOAM simulations only a certain number of cores were used for each case. 12 cores were used for Case 1 and 2, and 24 cores for Case 3.

Table 2.1: Technical details for the SINTEF and NTNU hardware used.

	NTNU - DMD	SINTEF - OpenFOAM
Operating system	Ubuntu 20.04	Debian 4.9.258-1
RAM	64 GB	64 GB
CPU / VCPU speed	2.4 Ghz	2.1 Ghz
CPU's/ VCPU's	8	47

2.1.4 5-MW baseline wind turbine

The wind turbines used in the simulations in this thesis are of the model *NREL offshore 5MW baseline wind turbine* as defined by the U.S. Department of Energy (Jonkman et al., 2009).

The turbine has three key points with regard to the wind speed: cut-in speed, rated wind speed and cut-out wind speed. The Cut-in speed is the minimum wind speed at which the turbine will deliver a surplus of power. This is the lowest point of velocity where the turbine operates. The rated wind speed is the wind speed at which the rated power (usually the maximum power output) is reached. The cut-out speed is the maximum wind speed at which the turbine operates. This limitation exists to prevent damage to the turbine and due to safety constraints.

The power curve for the 5-MW NREL wind turbine can be observed in figure 2.1. As can be observed in the figure, the produced power is equal to 5MW for all wind speeds from the rated speed (11.4 m/s) to the cut-out speed (25 m/s). The cut-in speed is 3.0 m/s.

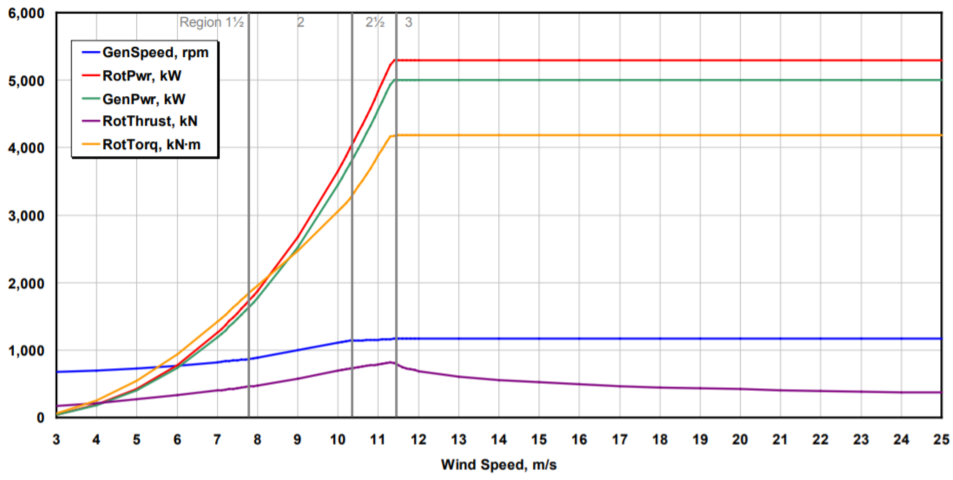


Figure 2.1: Power produced at various wind speeds by the 5MW NREL turbine (Jonkman et al., 2009).

2.1.5 Jupyter-notebook and associated packages

Jupyter notebook is a development environment, using the iPython kernel, focused on giving a flexible overview of data with the ability to simulate interactions between data sets. Within this dissertation, several python packages were used, all of which are freely available, except the ones directly created for the Upwards project. The full list can be observed in appendix E. Among the packages, the most notable ones are PyDMD, Numpy and Matplotlib. Matplotlib is used for plotting large quantities of data. Python Dynamic Mode Decomposition (PyDMD) is a code used to implement DMD in a Python environment. It was first introduced in 2018 (Demo et al., 2018), based on both the original work by Schmid and further expansions of the original work (Kutz et al., 2016). For the purpose of mathematical calculations done within Jupyter, the Numpy package is used for general data handling and the implementation of the SVD, Eckart-Young theorem and the L^2 norm.

2.1.6 Ashes

Ashes 13.16 is a wind turbine analysis program named as an acronym meaning aero-servo-hydro-elastic simulation. For this thesis, Ashes is used for power calculations. Elasticity within the turbine is ignored and power calculations are performed under the assumption that the only change in condition between tests is the uniform flow at each turbine location. The 5-MW NREL turbine is fully incorporated with the same specifications as in the OpenFOAM simulations shown in appendix B.

2.2 General approach

The tests are divided into 3 different cases. Each case is further divided into separate systems, where a system consists of an OpenFOAM simulation and a DMD reconstruction. Case 1 consists of Systems 1-4, Case 2 consist of System 5 and 6 and Case 3 consist of System 7 and 8. OpenFOAM visualisation, CSV conversion, error string calculations, timing, velocity extraction and power calculations are performed in each case as elaborated on below. The full code for Case 1 can be observed in appendix E. The code for Case 2 and Case 3 are not presented in the appendix. The reasoning for omitting these codes is the sheer size required to display it, in addition to the principle components being the same for each case. The code changes required to analyse a different data set are simply to update what simulation files to use and update the turbine locations for velocity extraction.

2.2.1 Simulation volume

For the OpenFOAM simulations, Euclidean spaces of two different proportions were used. This had a direct impact on the complexity of the system. The X and Z-axis were equal for all cases, but the y-axis was expanded in Case 3. For Case 1 and 2, the y-axis was simulated from -400 to 400, while for Case 3 the simulation area was from -600 to 1200.

2.2.2 SVD rank estimation and DMD training

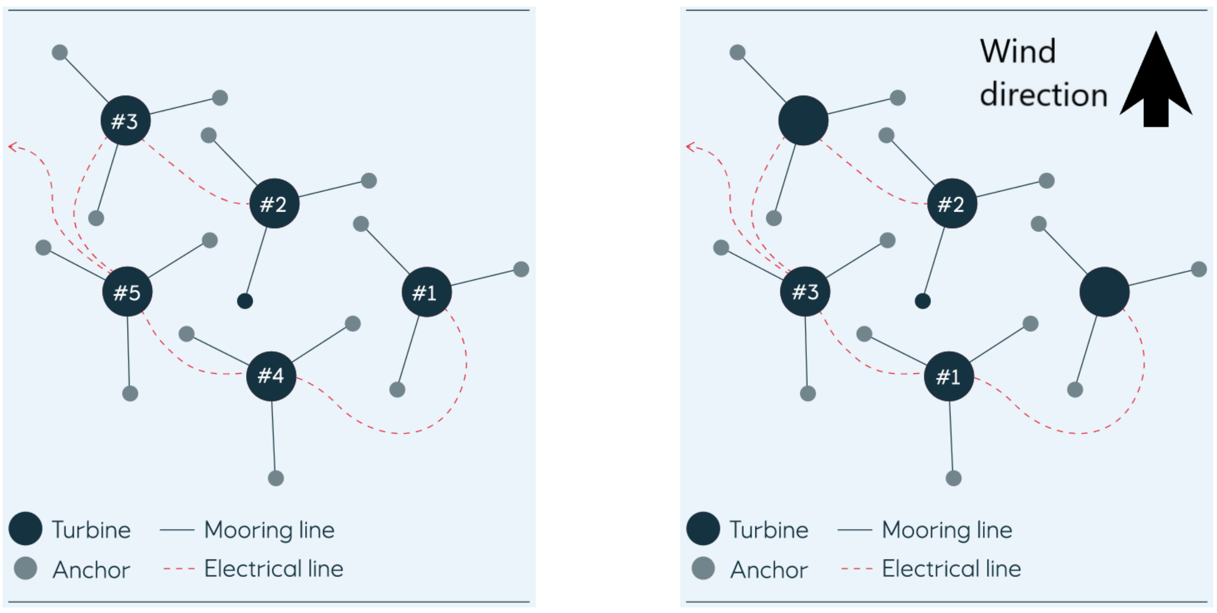
An SVD was run on the data dictionaries of the OpenFOAM simulations to gain a foundation to base the rank needed for the DMD on. The SVD rank estimation was evaluated by plotting the principle components. The rank was then estimated separately for each system within a case. A DMD was trained on the systems with the given rank and new DMD's were created through interpolation for the desired wind speeds. The eigenvalues for the systems as well as the modes and dynamics were displayed for analysis. The eigenvalues were observed to ensure that there were no severe dampening effects. Dynamics were observed to ensure that a sinusoidal pattern was observed at a higher SVD rank, as is expected. When the dampening effects of the eigenvalues were deemed too great, or dynamics didn't follow expectations, the SVD rank was reduced to accommodate. The eigenvalues will be displayed in the results section of the thesis.

2.2.3 OpenFOAM visualisation and CSV conversion

For each case, OpenFOAM simulations were performed for wind speeds of 6 and 8 m/s. This created the foundation on which the DMD's were created. Turbine specifications were implemented from OpenFOAM as shown in appendix B. To convert the OpenFOAM files to a *comma-separated values* (CSV) file format successfully, preliminary formatting was required within the OpenFOAM files. A folder named block must be created within the *constant/polyMesh* folder of each simulation, and the *points* and *faces* files from the simulation copied into it. The first time a CSV conversion was performed on simulation data an error consistently occurred. The issue was solved by executing each CSV conversion code a second time. The error details can be observed in appendix D. The error was believed to do with conversion issues and the time spent to circumvent this error for each case was not included in the results section. From there the conversion was performed as shown in appendix E converting wind speeds, rotor speed, rotor power, thrust, torque, yaw and pitch into data dictionaries ready for analysis in a python environment. The OpenFOAM files were visualised and analysed in Paraview to find the optimal time slice to analyse. The optimal time for cropping of the data sets was noted. The time was chosen based on what was deemed relevant. The reasoning for the restricted area will be elaborated on for each case. For easier comparison, the same amount of snapshots were used for all systems within each separate case. This means the number of data sets within Case 1 was equivalent for Systems 1-4. The dimensional properties of the OpenFOAM simulations were loaded as a grid as well as the previously mentioned CSV data dictionaries. The data dictionary shape for the CSV files of all systems within each case was restrained to the earlier noted area and displayed to ensure that proper conversion has occurred.

2.2.4 Turbine locations and wind directions

The turbine locations used in the OpenFOAM simulations were based on the layout used at Hywind Scotland's (HS) offshore wind farm, although with some notable changes. Both the original HS layout and this dissertation's adaptation of the layout can be observed in figure 2.2. In this thesis three of the turbines were imitated for the OpenFOAM simulations. The equivalent of turbine #4, #2 and #5 in the HS layout was referred to as Turbine 1, Turbine 2 and Turbine 3. A southern wind direction was used in the OpenFOAM simulations, as it is the most prominent wind direction at HS. The distance between turbines in HS is 9 times the rotor diameter, which was the standard used in this thesis. Due to a different turbine being used, the distance between the turbines equates to 1161m in this dissertation.



(a) (b)

Figure 2.2: (a) Hywind Scotland turbine layout (Equinor, 2021), (b) adaptation used in this thesis. Only the numbered turbines are used in simulation and the wind direction for all tests are as shown in the figure.

2.2.5 Error string analysis

Error strings were created by comparing every snapshot within the reconstructed system to every snapshot within the original OpenFOAM system at a given wind speed. The error strings were then compiled to obtain the snapshot with the average and peak error, along with the index number of the peak error. Slices of the XY, XZ, and YZ-planes at peak error-index were displayed. The visualisations displayed the L^2 norm as well as the individual components, meaning the wind speed, in x, y, and z-direction to check for irregularities. In the appendix code and visualisations, wind speed exclusively in x, y and z directions are classified as component 0, 1 and 2. The OpenFOAM simulations were plotted with the DMD reconstructed versions at all relevant turbine locations to get an insight into the reconstructions' accuracy in relation to velocity and power calculations. The complete error strings for the different DMD's were visualised to get an understanding of the error rate's evolution in time. The OpenFOAM simulations were plotted next to the reconstructed at varying locations depending on the case. The variations will be defined within each case.

2.2.6 Timing

Throughout testing, several timers were set to give a proper understanding of which aspects have the most significant impact compared to their CFD equivalent. ***Total time*** was defined as the total time used from CSV conversion starts until the DMD's were constructed and visualised for a case. ***DMD only*** was defined as the time used to create all DMD's, with interpolation and reconstruction for each case. ***dmdx only*** was defined as the time spent interpolating an existing DMD to a new velocity and reconstructing the snapshots at said velocity. Herein x is the DMD used for interpolation and reconstruction.

2.2.7 Velocity extraction and power calculations

The procedure for obtaining the power required velocity readings at each rotor. In accordance with REWS theory, velocity readings were required at five separate points. During this dissertation, an approximation to the point values was created for implementation in REWS theory. Rather than using the velocity of five single points, the average inlet flow for an area of 1x1 meter with origin dictated by REWS was used. Thus the wind speed for each single point reading used in REWS will for this thesis be an average of the velocities in an area around the chosen coordinates. The coordinates used are the same in relation to each turbine and are shown in table 2.2. Power calculations were performed in Ashes at the velocity calculated by the previously mentioned method.

Table 2.2: REWS weighting shown as used for each turbine in this thesis.

Segment height [m]	356.5	328.2	300.0	271.8	243.5
Segment weighting [%]	11.42	24.75	27.66	24.75	11.42

2.3 Case 1 - Proof of concept.

Case 1 was a baseline proof of concept test based on a single turbine. Four systems were created wherein the first two, System 1 and System 2, were fundamentally different from the other systems elaborated on in this thesis. The purpose of these two systems was to corroborate the validity of the DMD implementation, thereby authenticating more advanced DMD tests. First, the grid and data dictionaries were loaded from the OpenFOAM simulations. The OpenFOAM simulation had an initial duration of 2390 seconds for 6 m/s and 2310 seconds for 8 m/s. The simulations were then visualised and cropped to contain the snapshots from 500 to 2310 seconds. The crop timing was chosen to ensure that a steady wake had developed. A visualisation of the first snapshot after cropping can be observed in figure 2.3.

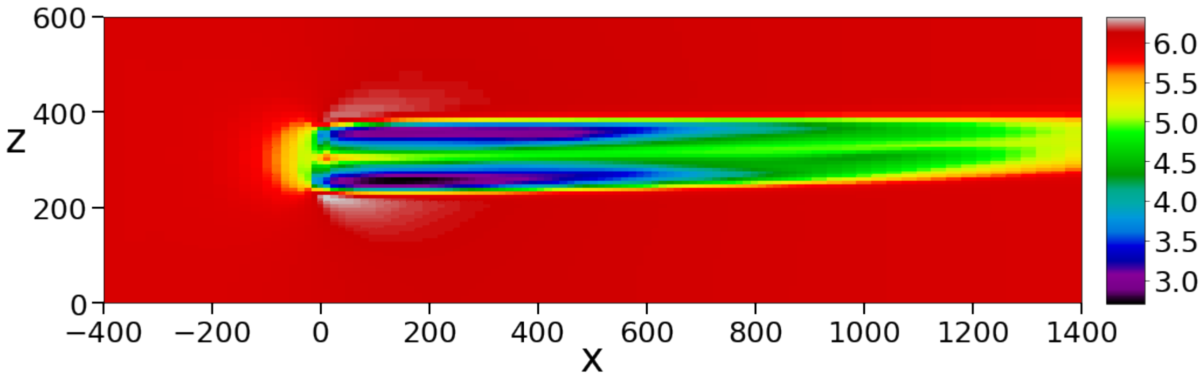


Figure 2.3: Visualisation of the first snapshot after cropping for the 6 m/s OpenFOAM simulation used in Case 1. Visualised using Jupyter notebook, with the colour bar units in m/s.

SVD analysis was performed on the dictionaries of the two wind speeds and the eigenvalues, modes and dynamics were evaluated. The modes and dynamics can be observed in appendix C.2. On inspection, a rank of 10 was determined to be appropriate for both data sets. A plot of the SVD's principle components can be observed in figure 2.4.

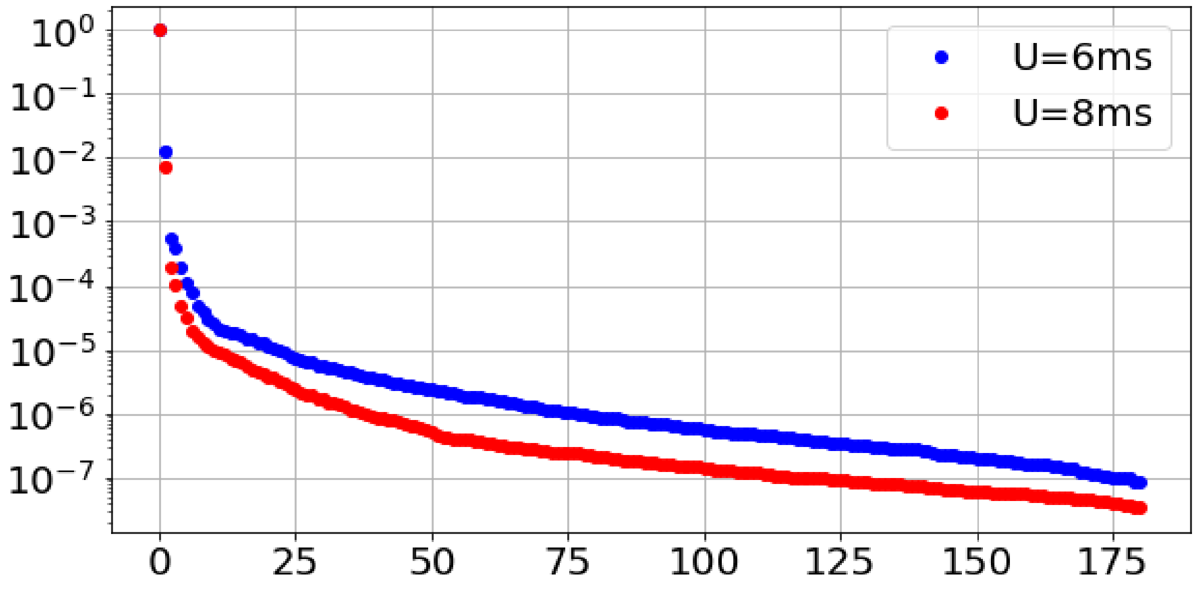


Figure 2.4: Plot of the principle components of the SVD performed on the 6 and 8 m/s OpenFOAM simulation data for Case 1. The plotted values are the SVD of a single rank divided by the sum of the SVD's within the given systems. The y-axis is shown in a logarithmic scale.

Two DMD's were then trained with an SVD rank of 10. System 1's DMD recreation was created from the 6 m/s simulation with a target speed of 6 m/s. System 2's recreation was created from the 8 m/s simulation with a target speed of 8 m/s. Two more DMD's were then created for System 3 and 4 through interpolation, giving them another target speed. System 3's DMD recreation was created and trained with a target speed of 8 m/s. System 4's recreation was created by interpolating dmd2's target speed to 6 m/s. The DMD's were then reconstructed with the same amount of snapshots as the post-crop, OpenFOAM simulations. Temporal error rates for the individual systems were calculated in comparison to their target system. This means System 1 and 4 were compared to the 6 m/s OpenFOAM simulation and System 2 and 3 to the 8 m/s. The 0, 1, 2 and norm component were then visualised at their peak temporal error rate for all planes at the turbine location and 1161m = 9 times rotor diameter downwind of Turbine 1. A summary of the information for all the systems in Case 1 can be observed in table 2.3.

Table 2.3: Case 1 specifications.

System	**1**	**2**	**3**	**4**
Trained wind speed [m/s]	6	8	6	8
Target Wind speed [m/s]	6	8	8	6
DMD	dmd1	dmd2	dmd3	dmd4
Wind direction	Due south	Due south	Due south	Due south
SVD rank	10	10	10	10
Crop time [s]	500	500	500	500

2.4 Case 2 - Two turbines in a row

Case 2 was a direct expansion of Case 1, where a second turbine was introduced. The second turbine was located 1161 meters downwind of Turbine 1. Due to the turbine locations, the wake from the first turbine had a direct impact on the wind speed affecting the second turbine. The grid and data dictionaries were loaded from the OpenFOAM simulations. Initially, the OpenFOAM simulation time was a total of 1160 seconds for 6 m/s and 1080 seconds for 8 m/s. The simulations were then visualised and cropped to contain the snapshots from 500 to 1080 seconds. The crop timing was chosen to ensure that the wake had started affecting the second turbine. A visualisation of the first snapshot after cropping can be observed in figure 2.5.

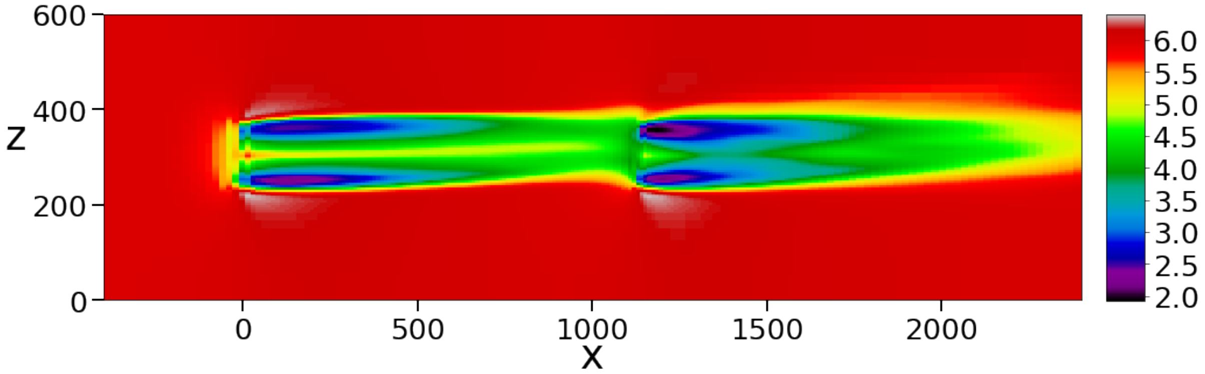


Figure 2.5: Visualisation of the first snapshot after cropping for the 6 m/s OpenFOAM simulation used in Case 2. Visualised using Jupyter notebook, with the color bar units in m/s.

SVD analysis was performed on the dictionaries of the two wind speeds and the eigenvalues, modes and dynamics were evaluated. On inspection, a rank of 10 was determined to be appropriate for both data sets. A plot of the SVD's principle components can be observed in figure 2.6.

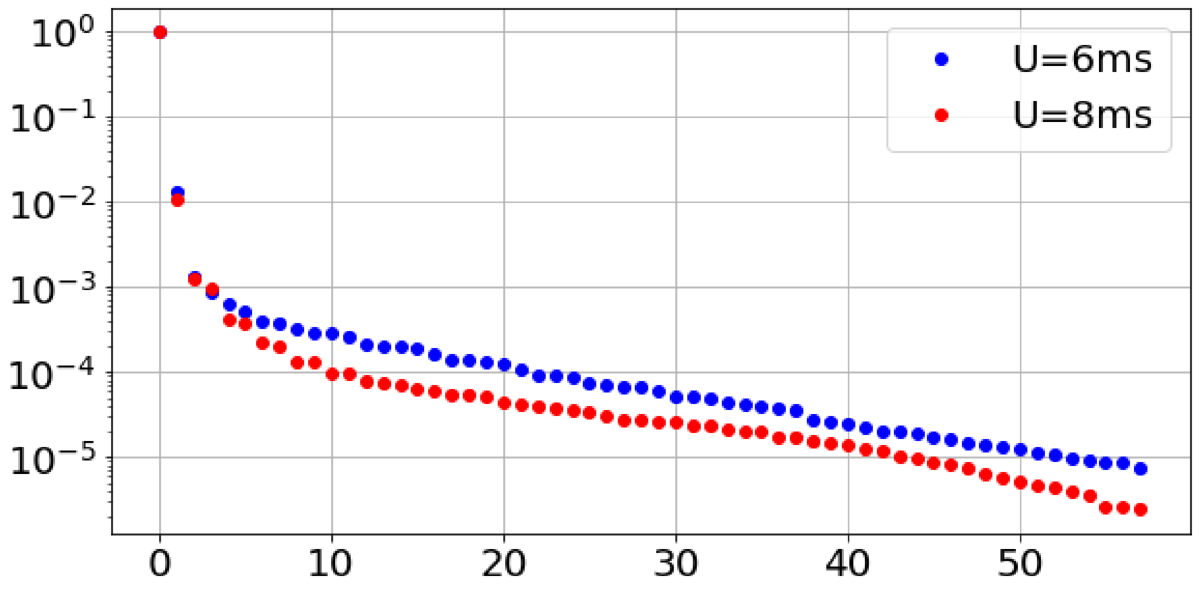


Figure 2.6: Plot of the principle components of the SVD performed on the 6 and 8 m/s OpenFOAM simulation data for Case 2. The plotted values are the SVD of a single rank divided by the sum of the SVD's within the given systems. The y-axis is shown in a logarithmic scale.

Two DMD's were then trained with an SVD rank of 10. System 5's DMD recreation was created and trained on the 6 m/s OpenFOAM simulation and interpolated to the target velocity of 8 m/s. System

6's DMD recreation was created and trained on the 8 m/s OpenFOAM simulation and interpolated to the target velocity of 6 m/s. The DMDs were then reconstructed with the same amount of snapshots as the post-crop OpenFOAM simulations. Temporal error rates for the individual systems were calculated in comparison to their target system. This means System 5 was compared to the 8 m/s OpenFOAM simulation and System 6 to the 6 m/s one. The 0, 1, 2 and norm component were then visualised at their peak temporal error rate for all planes at the turbine locations of x=0 and 1161. Dictionaries containing the deviance between the original simulations and the largest temporal error were then created and displayed for the 0, 1, 2 and norm component for all planes at the two turbine locations. A summary of the information for both systems in Case 2 can be observed in table 2.4.

Table 2.4: Case 2 specifications.

System	**5**	**6**
Trained wind speed [m/s]	6	8
Target Wind speed [m/s]	8	6
DMD	dmd5	dmd6
Wind direction	Due south	Due south
SVD rank	10	10
Crop time [s]	500	500

2.5 Case 3 - Hywind Scotland approximation

Case 3 was a further expansion of the previous cases. The turbines layout was formed as an equilateral triangle as previously explained in section 2.2.4. Due to the turbine locations, the wake from Turbine 1 had a direct impact on the wind speed affecting Turbine 2. Turbine 3 however would possibly have little or no effect on the other two turbines. The loading of the data dictionaries from OpenFOAM was slightly different in Case 3 when compared to the two previous cases. In contrast to Case 1 and 2, the data dictionaries from OpenFOAM were cropped prior to loading. The first 300 seconds of the simulation were removed from the dictionaries before loading. This was done to reduce loading time. Initially, the OpenFOAM simulation time was a total of 720 seconds for 6 m/s and 700 seconds for 8 m/s. After removal of the excess files, the 6 m/s simulation consist of 530 seconds and the 8 m/s of 510 seconds. The data dictionaries and grid was then loaded. The simulations were then visualised and cropped to contain the snapshots from 300 to 700 seconds. The crop timing was chosen to ensure that the wake from the first turbine affects the second one in a satisfactory manner. A visualisation of the first snapshot after cropping can be observed in figure 2.7.

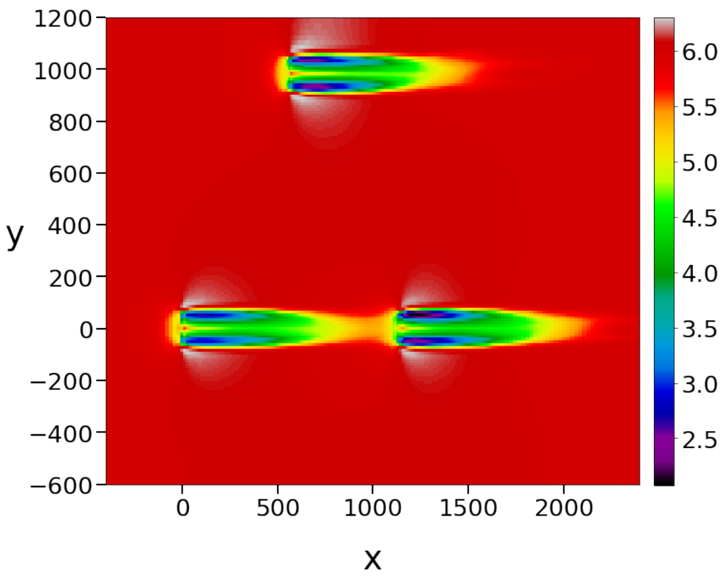


Figure 2.7: Visualisation of the first snapshot after cropping for the 6 m/s OpenFOAM simulation used in Case 3. Visualised using Jupyter notebook, with the colour bar units in m/s. Note the change in perspective from the visualisation presented in Cases 1 and 2.

SVD analysis was performed on the dictionaries of the two wind speeds and the eigenvalues, modes and dynamics were evaluated. On inspection, a rank of 5 and 6 was determined to be appropriate for dmd7 and dmd8. A plot of the SVD's principle components can be observed in figure 2.8.

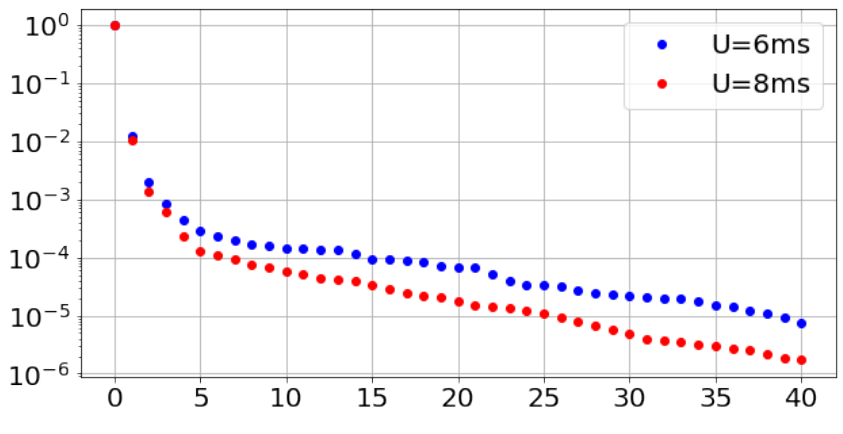


Figure 2.8: Plot of the principle components of the SVD performed on the 6 and 8 m/s OpenFOAM simulation data for Case 3. The plotted values are the SVD of a single rank divided by the sum of the SVD's within the given systems. The y-axis is shown in a logarithmic scale.

Two DMD's were then trained with the previously mentioned SVD rank. System 7's DMD recreation was created and trained on the 6 m/s OpenFOAM simulation and interpolated to the target velocity of 8 m/s. System 8's DMD was created and trained on the 8 m/s OpenFOAM simulation and interpolated to the target velocity of 6 m/s. The DMD's were then reconstructed with the same amount of snapshots as the post-crop OpenFOAM simulations, totalling 400 snapshots. Temporal error rates for the individual systems were calculated in comparison to their target system. This means System 7 was compared to the 8 m/s OpenFOAM simulation and System 8 to the 6 m/s one. The 0, 1, 2 and norm component were then visualised at their peak temporal error rate for all planes at the turbine locations. Dictionaries containing the deviance between the original simulations and the largest temporal error were then created and displayed for the 0, 1, 2 and norm component for all planes at the two turbine locations. A summary of the information for both systems in Case 2 can be observed in table 2.5.

Table 2.5: Case 3 specifications.

System	**7**	**8**
Trained wind speed [m/s]	6	8
Target Wind speed [m/s]	8	6
DMD	dmd7	dmd8
Wind direction	Due south	Due south
SVD rank	5	6
Crop time [s]	300	300

3 Results

The scale is always given in m/s for velocity visualisations. Visualisations for each system is displayed at the timestamp of the peak overall error rate in the system. All visualisations display the *norm* component. Visualisations exclusively in x, y and z-direction for Case 1 only can be found in appendix C.1.

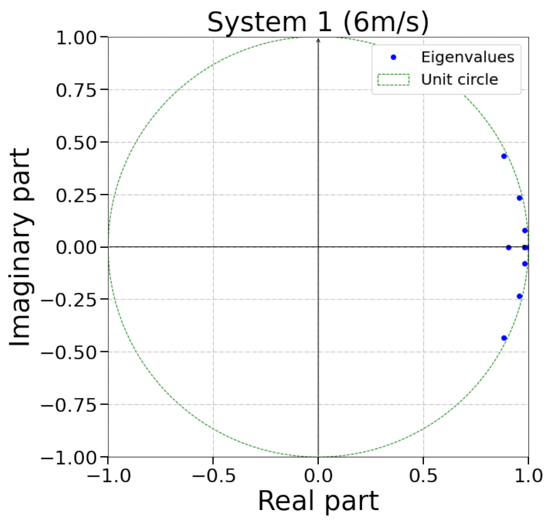
3.1 Case 1

To summarise from section 2.3: Case 1 consisted of two OpenFOAM simulations and four DMD reconstructions for a single turbine. Systems 1 and 2 were trained on OpenFOAM files with wind velocities of 6 and 8 m/s respectively. Both system's target velocities equated their training velocity before reconstruction was performed.

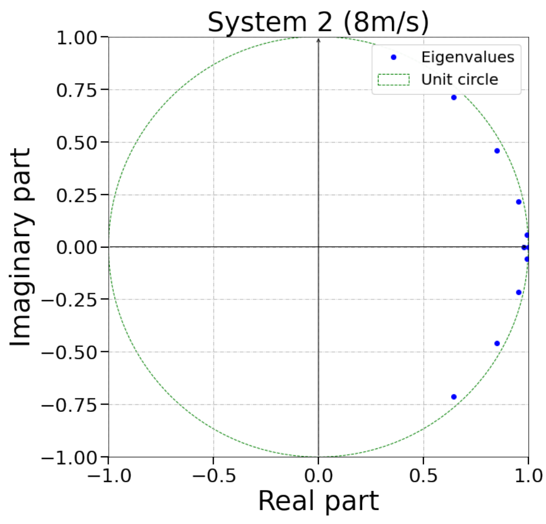
Systems 3 and 4 were trained on 6 and 8 m/s as well but were, unlike Systems 1 and 2, interpolated prior to reconstruction. System 3 interpolated with a target velocity of 8 m/s, and System 4 with a target velocity of 6 m/s. Note that all slices are referred to by their index number, after cropping as described in chapter 2.3.

3.1.1 System 1 and 2 - velocity precision

Eigenvalues for System 1 and 2 are shown in figure 3.1.



(a)



(b)

Figure 3.1: (a) Eigenvalues of System 1 and (b) System 2.

Furthermore, the error rate was calculated for each reconstructed snapshot. In table 3.1 the average and peak error rate can be observed, as well as when the peak error in each system occurs.

Table 3.1: Error rates for system 1 and 2.

System no.	Avg. error	Peak error	Peak error slice index	Last slice index
1	0.1034%	0.2104%	99	180
2	0.0578%	0.1017%	180	180

The systems were further examined by visualising the distribution of energy in the XY, XZ and YZ-plane respectively. Visualisations were made of both OpenFOAM simulations, the reconstructions made by dmd1 and dmd2, and ultimately the discrepancy within each system. The XY-plane was displayed first for System 1 and System 2 with the origin at the Turbine 1 location as shown in figure 3.2.

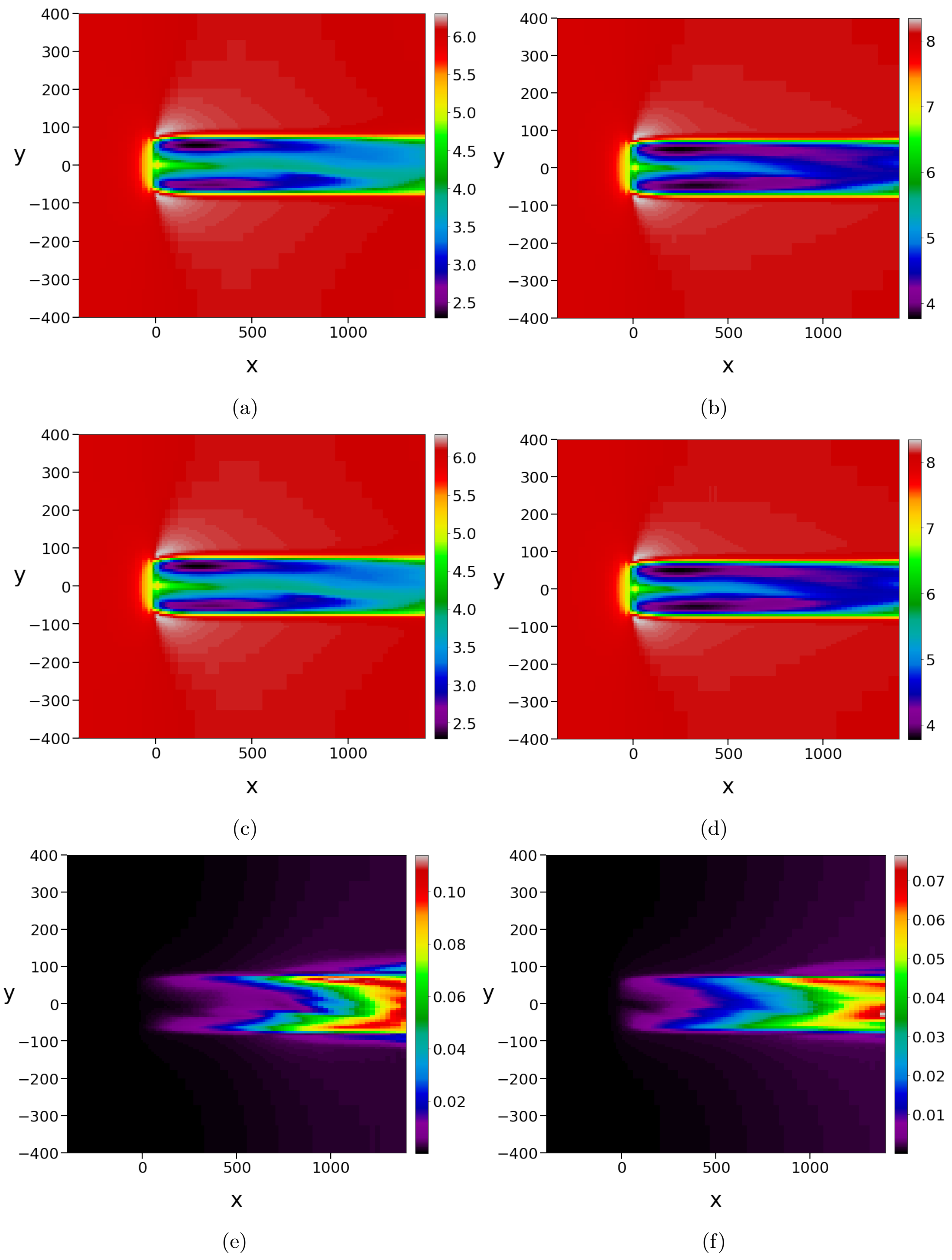


Figure 3.2: (a) OpenFOAM in System 1, (b) OpenFOAM in System 2, (c) dmd1, (d) dmd2, (e) discrepancy in system 1, (f) discrepancy in system 2. Visualised in the XY-plane.

In figure 3.3 the wake in the XZ-plane can be observed. Once again, CFD simulations, corresponding DMD reconstructions and the discrepancy in each system are included.

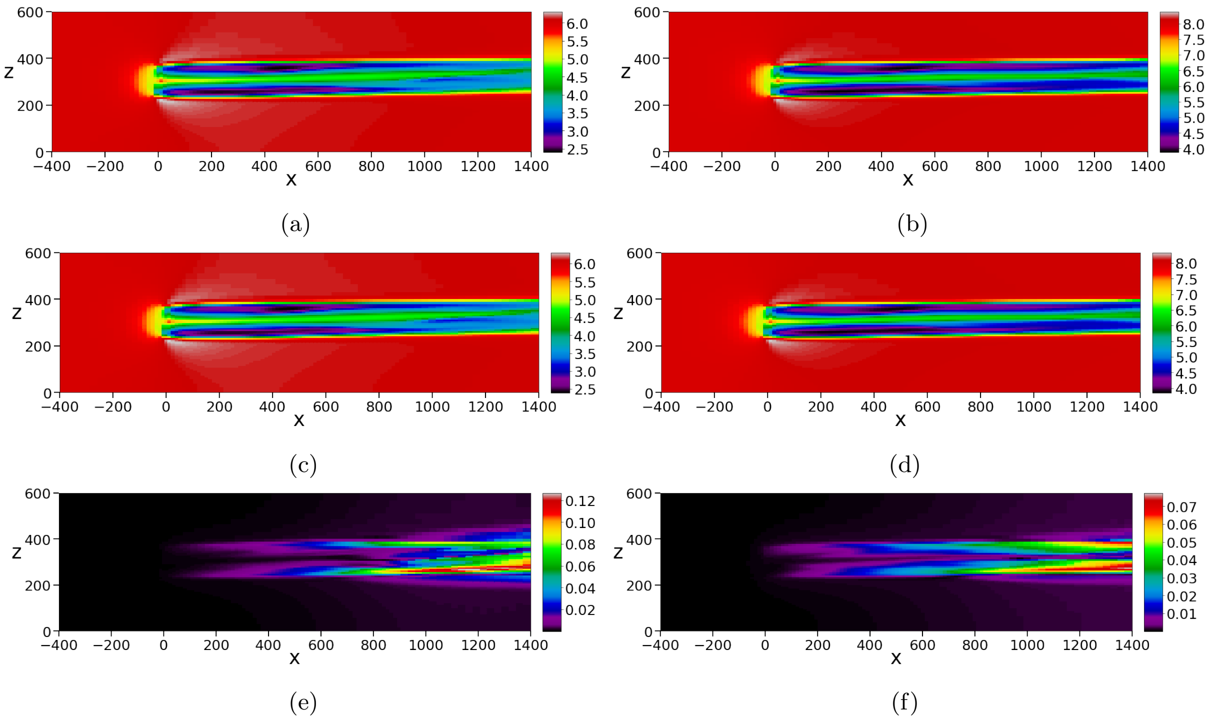


Figure 3.3: (a) OpenFOAM in System 1, (b) OpenFOAM in System 2, (c) dmd1, (d) dmd2, (e) discrepancy in system 1, (f) discrepancy in system 2. Visualised in the XZ-plane.

In figure 3.4 the YZ-plane at the location of Turbine 1 can be observed. Once again, CFD simulations, corresponding DMD reconstructions and the discrepancy in each system are included.

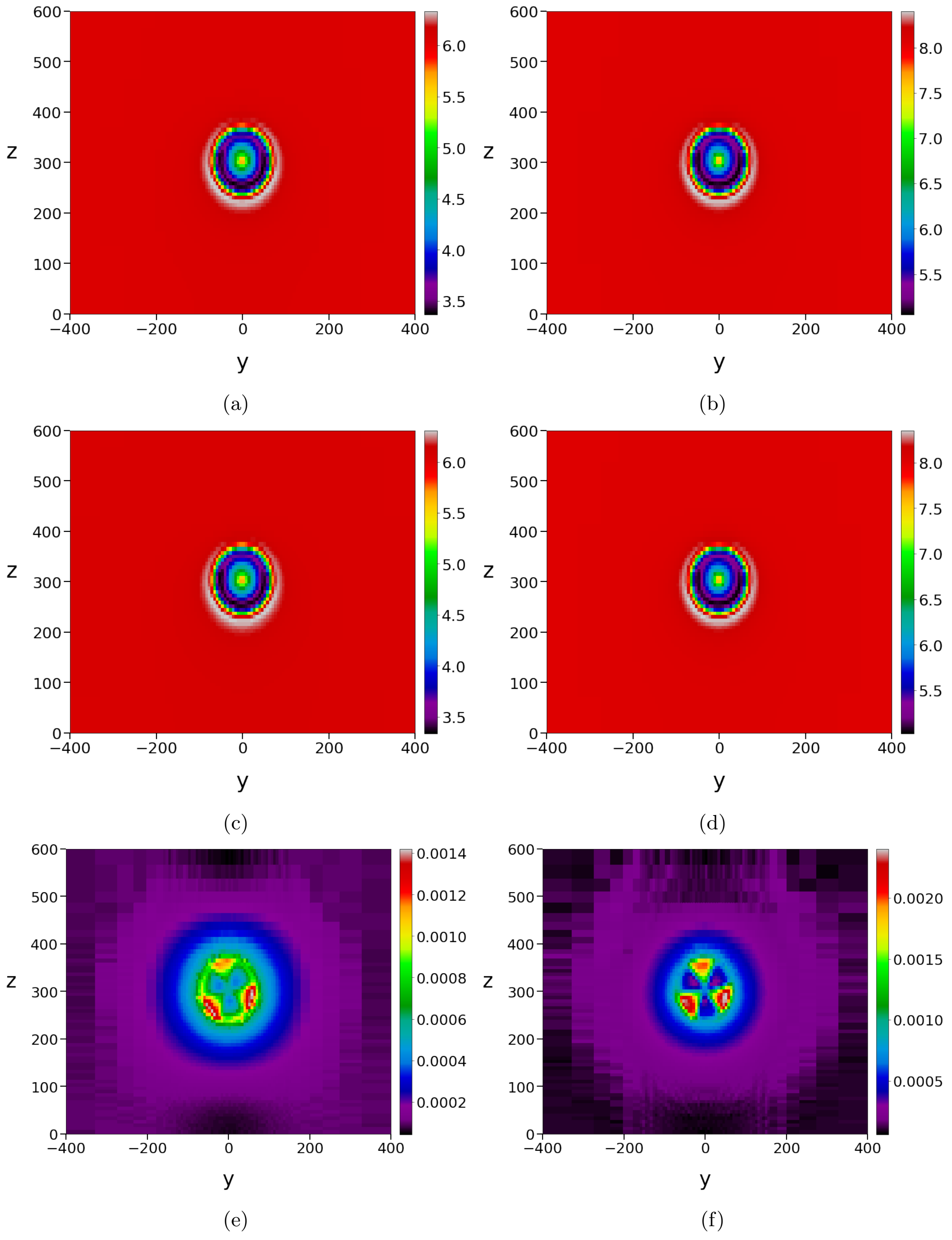


Figure 3.4: (a) OpenFOAM in System 1, (b) OpenFOAM in System 2, (c) dmd1, (d) dmd2, (e) discrepancy in system 1, (f) discrepancy in system 2. Visualised in the YZ-plane.

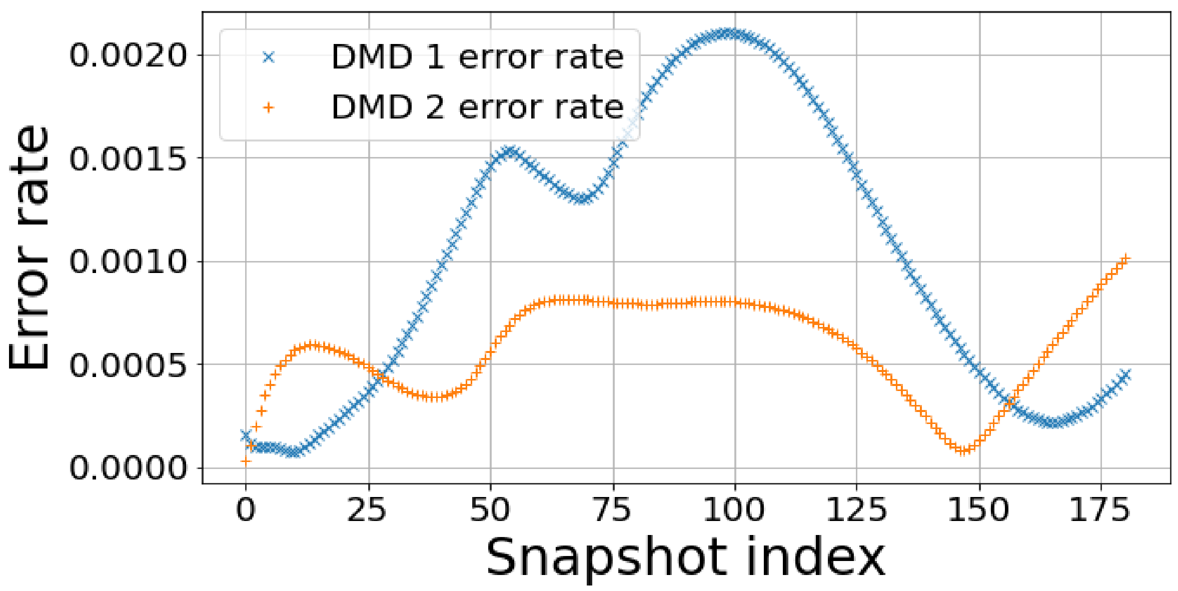
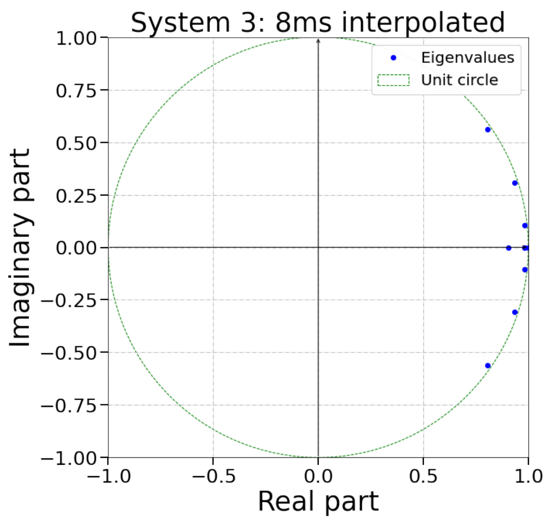


Figure 3.5: Error rates of each reconstructed slice for System 1 and 2.

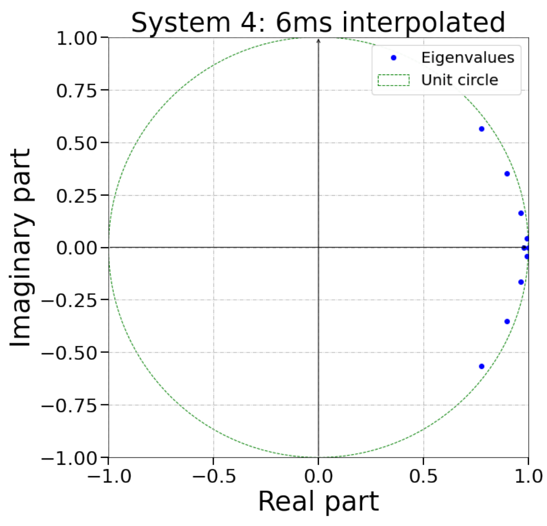
Additionally, the evolution of the overall error rate over time for both systems was examined. This can be observed in figure 3.5.

3.1.2 System 3 and 4 - velocity precision

Eigenvalues for System 3 and System 4 are visualised in figure 3.6.



(a)



(b)

Figure 3.6: Eigenvalues of (a) System 3 and (b) System 4.

Average and peak error for the two systems can be observed in table 3.2, as well as the peak error indices.

Table 3.2: Error rates for system 3.

System no.	**Avg. error**	**Peak error**	**Peak error slice index**	**Last slice index**
3	2.056%	2.355%	180	180
4	2.050%	2.417%	180	180

The systems were further examined by visualising the distribution of energy in the XY, XZ and YZ-plane respectively. Visualisations were made of both OpenFOAM simulations, the reconstructions made by dmd3 and dmd4, and ultimately the discrepancy within each system. The XY-plane was displayed first for System 3 and System 4 with the origin at the Turbine 1 location as shown in figure 3.7.

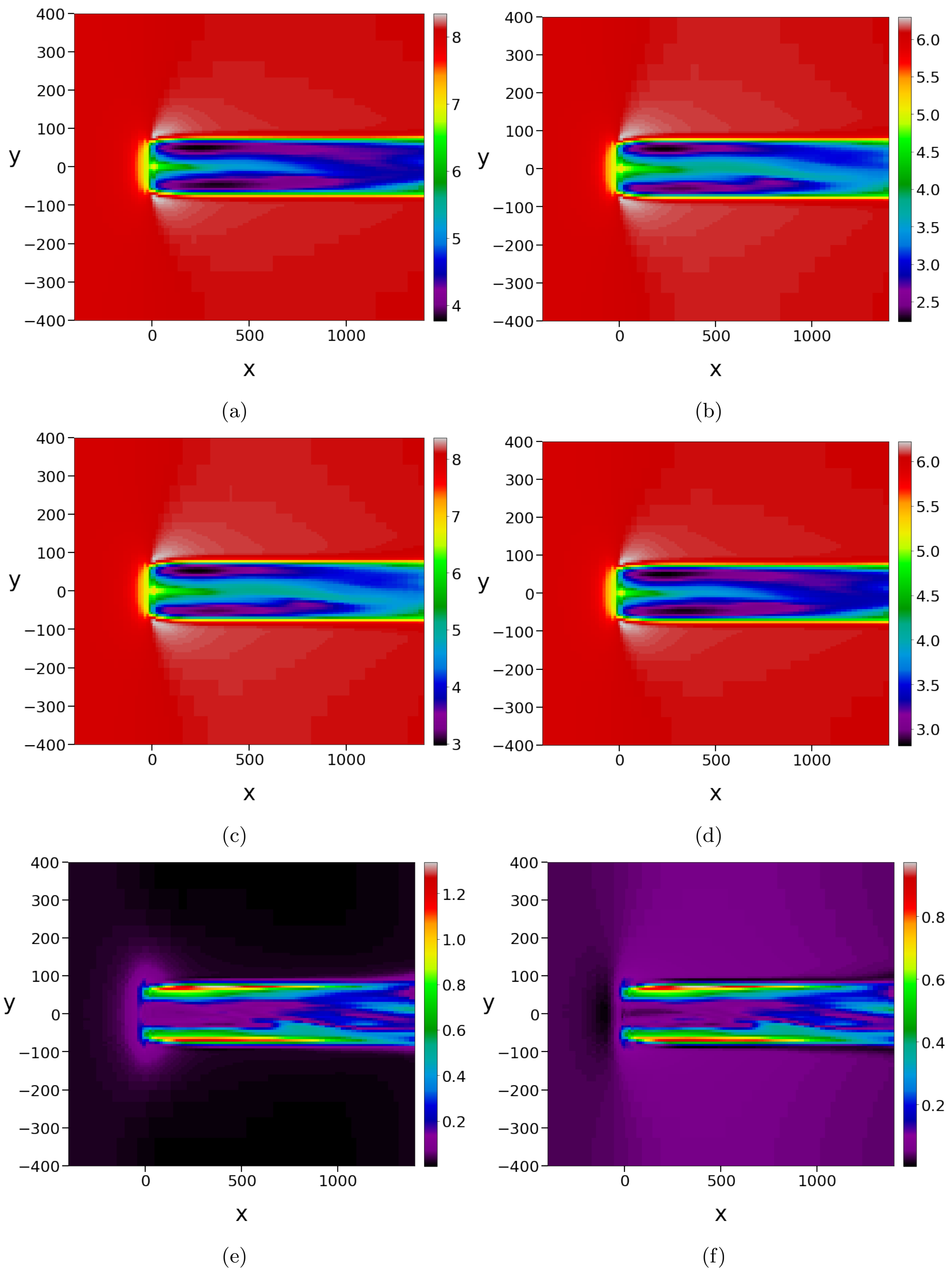


Figure 3.7: (a) OpenFOAM in System 3, (b) OpenFOAM in System 4, (c) dmd3, (d) dmd4, (e) discrepancy in system 3, (f) discrepancy in system 4. Visualised in the XY-plane.

In figure 3.8 the wake in the XZ-plane can be observed. CFD simulations, corresponding DMD reconstructions and the discrepancy in each system are included.

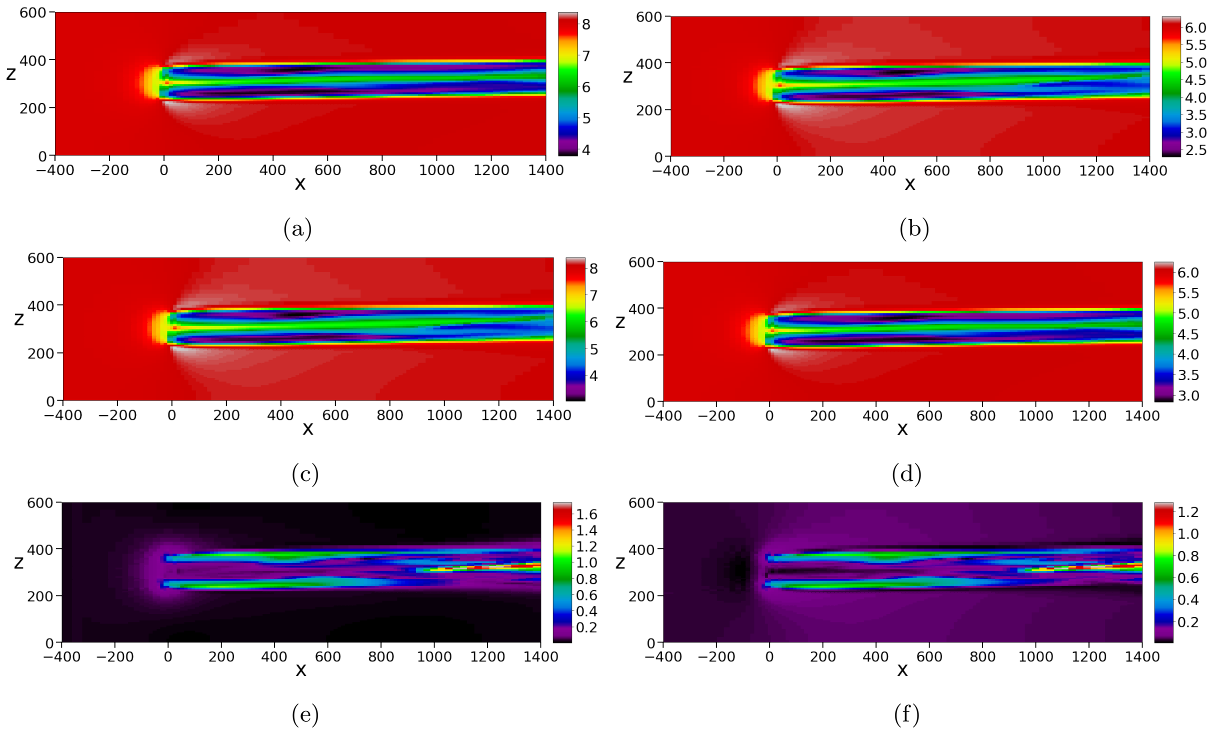


Figure 3.8: (a) OpenFOAM in System 3, (b) OpenFOAM in System 4, (c) dmd3, (d) dmd4, (e) discrepancy in system 3, (f) discrepancy in system 4. Visualised in the XZ-plane.

In figure 3.9 the YZ-plane at the location of Turbine 1 can be observed. Once again, CFD simulations, corresponding DMD reconstructions and the discrepancy in each system are included.

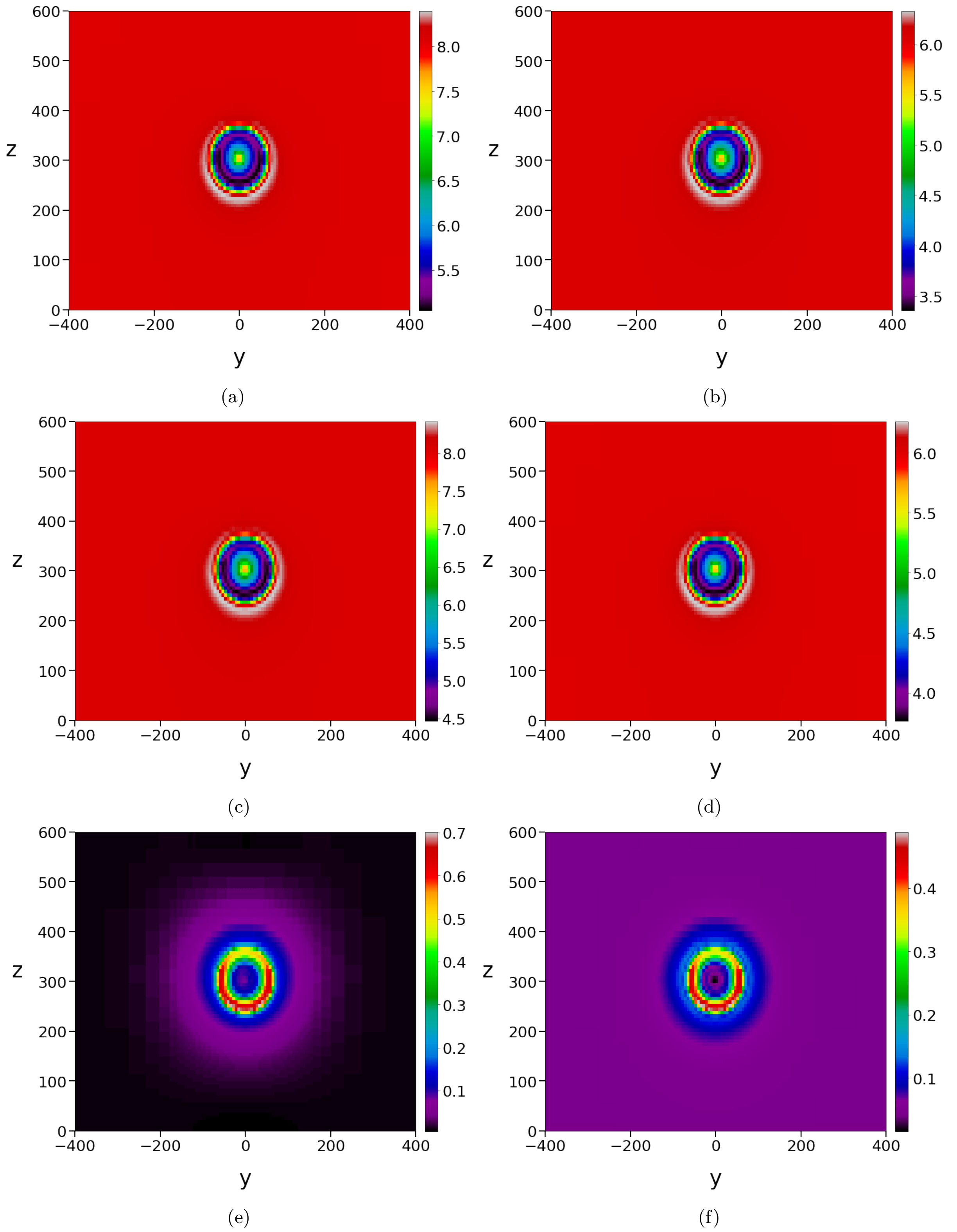


Figure 3.9: (a) OpenFOAM in System 3, (b) OpenFOAM in System 4, (c) dmd3, (d) dmd4, (e) discrepancy in system 3, (f) discrepancy in system 4. Visualised in the YZ-plane.

The distribution of energy was also examined parallel to the YZ-plane at x=1161, displaying the envisioned turbine located at (1161, 0, 300). CFD simulations, DMD reconstructions and the discrepancy in each system in this plane can be observed in figure 3.10.

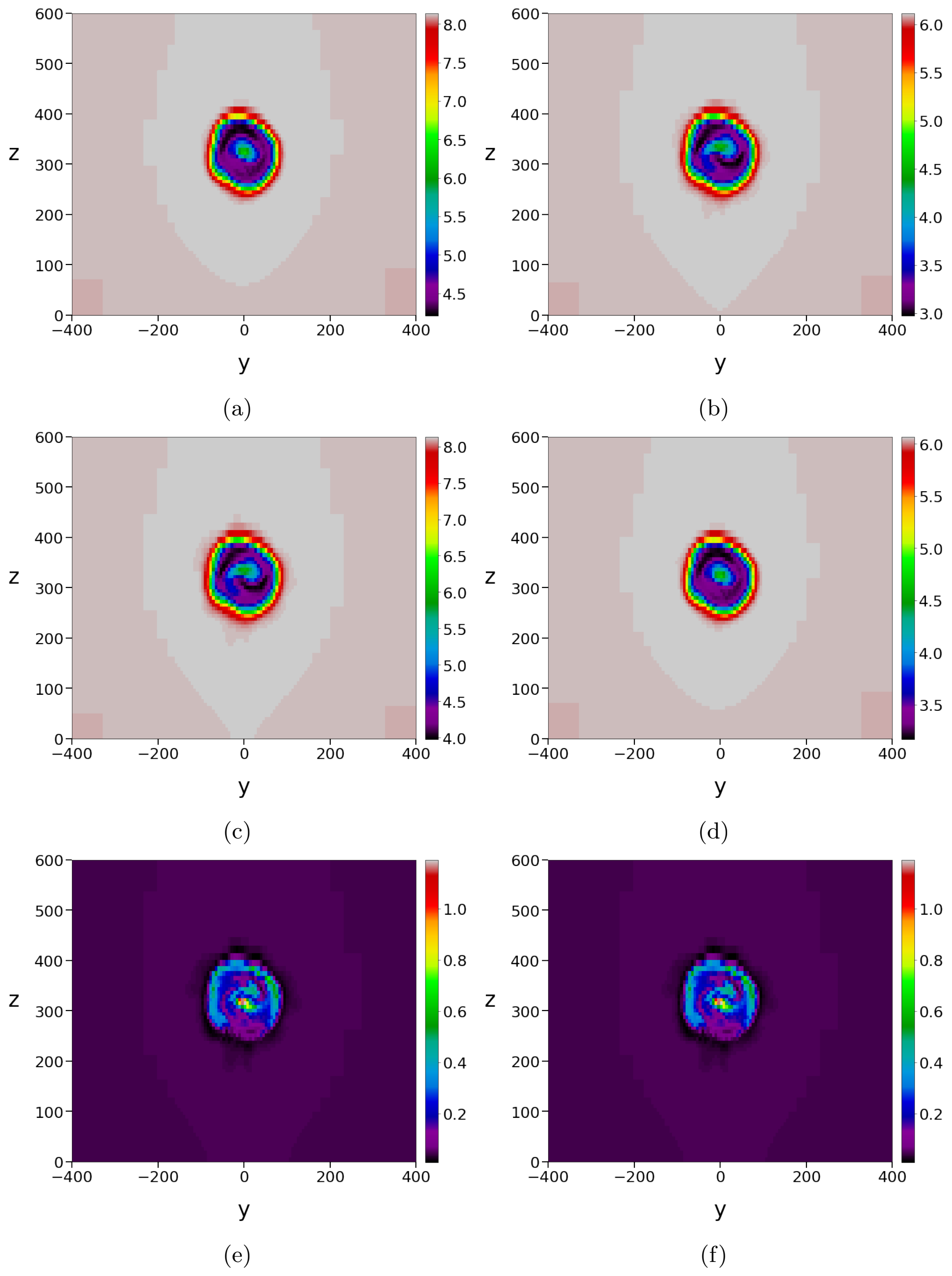


Figure 3.10: (a) OpenFOAM in System 3, (b) OpenFOAM in System 4, (c) dmd3, (d) dmd4, (e) discrepancy in system 3, (f) discrepancy in system 4. Visualised in the plane parallel to the YZ-plane at X = 1161 meters.

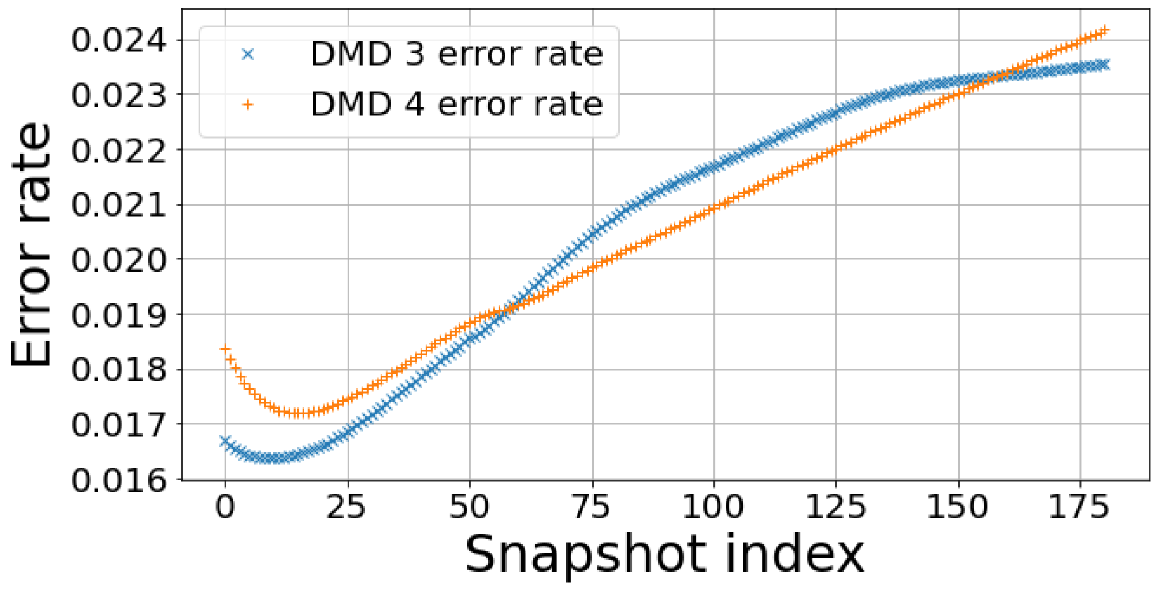


Figure 3.11: Error rates of each reconstructed slice for System 3 and 4.

The evolution of System 3 and System 4's error rates over time were also investigated, and can be observed in figure 3.11.

3.1.3 System 3 and 4 - power precision

Velocity readings were compiled, as explained in section 2.2.7, into a total of 40 data points: five locations at Turbine 1 and five at the envisioned turbine, once each for both CFD simulations and once each for both DMD reconstructions. The readings, as well as the REWS, can be observed in tables 3.3 and 3.4.

Table 3.3: Velocity readings for Systems 3 and 4 at Turbine 1.

	p1 [m/s]	**p2** [m/s]	**p3** [m/s]	**p4** [m/s]	**p5** [m/s]	**REWS** [m/s]
dmd3	6.126	5.553	6.560	5.670	5.264	**5.893**
OpenFOAM 8m/s	6.465	5.726	6.624	5.797	5.649	**6.068**
dmd4	4.821	4.270	4.940	4.323	4.213	**4.525**
OpenFOAM 6m/s	4.606	4.176	4.933	4.263	3.958	**4.431**

Table 3.4: Velocity readings for Systems 3 and 4 at x = 1161.

	p6 [m/s]	**p7** [m/s]	**p8** [m/s]	**p9** [m/s]	**p10** [m/s]	**REWS** [m/s]
dmd3	6.995	4.575	4.257	5.300	4.779	**4.966**
OpenFOAM 8m/s	7.082	4.639	4.777	5.737	4.665	**5.231**
dmd4	5.290	3.490	3.580	4.285	3.500	**3.918**
OpenFOAM 6m/s	5.259	3.416	3.188	3.990	3.590	**3.725**

The power production calculated for Case 1 is shown in table 3.5. The errors in each system can be observed in table 3.6.

Table 3.5: Power, predicted and actual

	Power, Turbine 1	**Power, Turbine 2**
System 3, dmd3	822.0 kW	489.6 kW
System 3, OpenFOAM, 8 m/s	899.3 kW	573.kW
System 4, dmd4	362.9 kW	226.1 kW
System 4, OpenFOAM, 6 m/s	340.1 kW	190.0 kW

Table 3.6: Error and error rates for predicted Power in Systems 3 and 4.

	Turbine 1		**Turbine 2**	
System	Error [kW]	Error rate	Error [kW]	Error rate
3	-77.30	8.596%	-83.70 kW	14.60%
4	22.80	6.704%	36.10 kW	19.00%

3.1.4 Time consumed

Total time: 1651 seconds, or 27 minutes and 31 seconds.

DMD only: 884.3 seconds, or approximately 14 minutes and 44 seconds.

dmd4 only: 13.55 seconds.

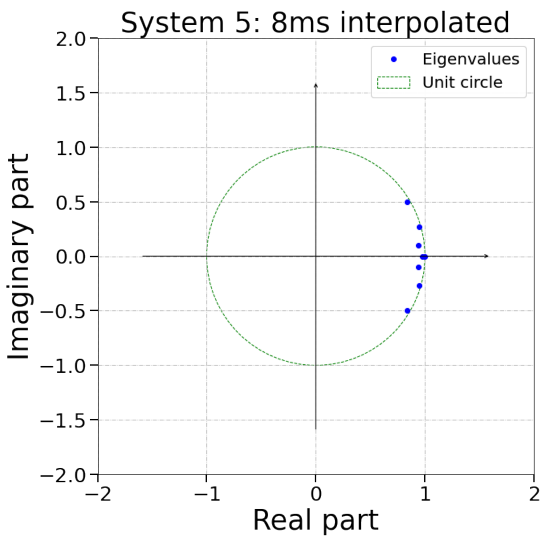
OpenFOAM simulation: approximately 12 hours per wind speed simulated.

3.2 Case 2

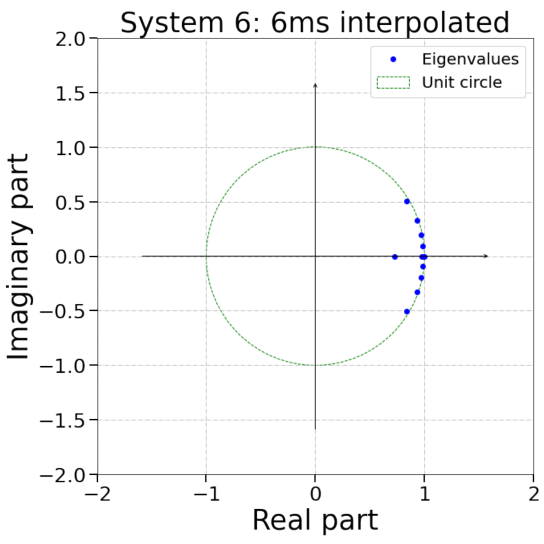
To summarise from section 2.4: Case 2 consisted of two OpenFOAM simulations and two DMD reconstructions for two turbines. Systems 5 and 6 were trained on 6 and 8 m/s respectively. Prior to reconstruction, they were interpolated to their target velocity. System 5's target velocity is 8 m/s, and System 6's target velocity is 6 m/s. Note that all slices are referred to by their index number, after cropping as described in chapter 2.4.

3.2.1 System 5 and 6 - velocity precision

Eigenvalues of System 5 and System 6 can be observed in figure 3.12.



(a)



(b)

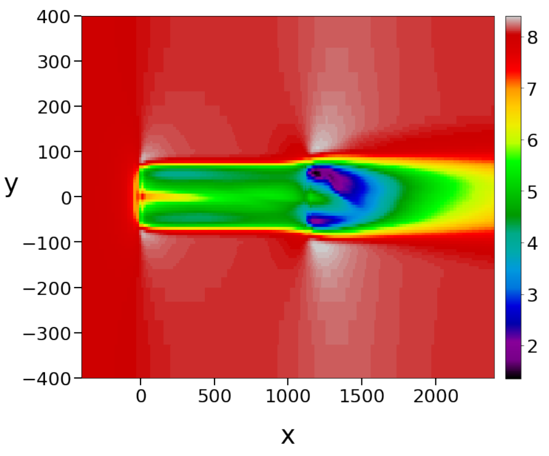
Figure 3.12: (a) Eigenvalues of System 5 and (b) System 6.

The error rates, both peak and average, for System 5 and System 6 are showed in table 3.7, as well as the index number for the slice with the highest spacial error.

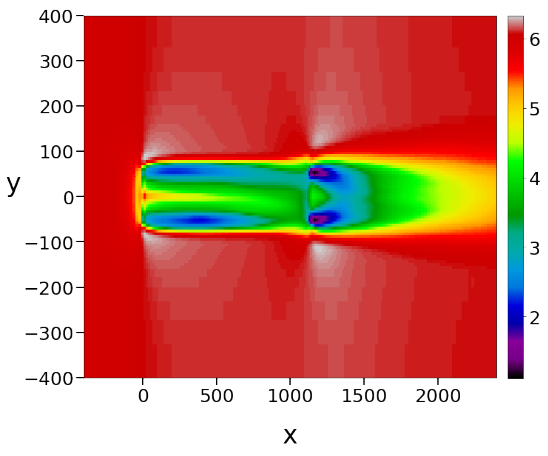
Table 3.7: Error rates for System 5 and System 6

System no.	Avg. error	Peak error	Peak error slice index	Last slice index
5	4.084%	4.737%	57	57
6	4.035%	4.653%	57	57

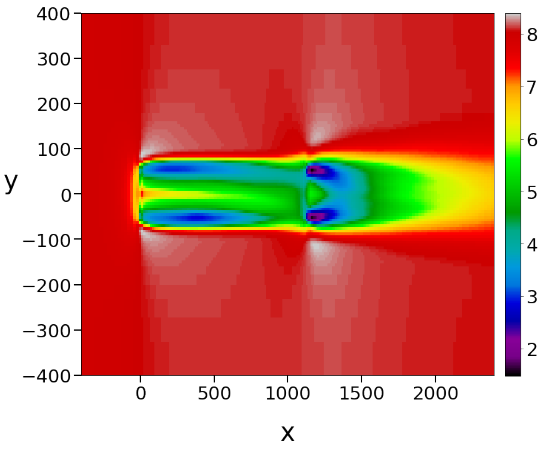
The systems were further examined by visualising the distribution of energy in the XY, XZ and YZ-plane respectively. Visualisations were made of both OpenFOAM simulations, the reconstructions made by dmd5 and dmd6, and ultimately the discrepancy within each system. The XY-plane was displayed first for System 5 and System 6 with the origin at the Turbine 1 location as shown in figure 3.13.



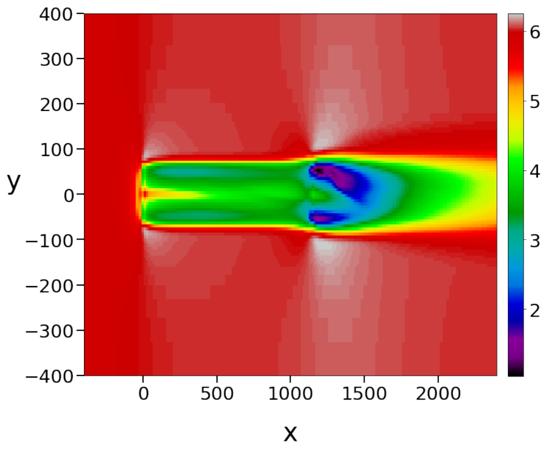
(a)



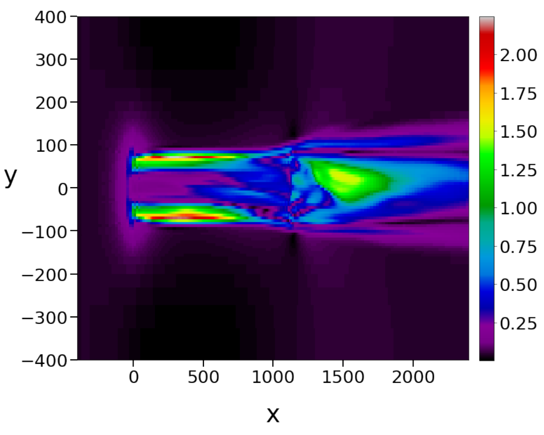
(b)



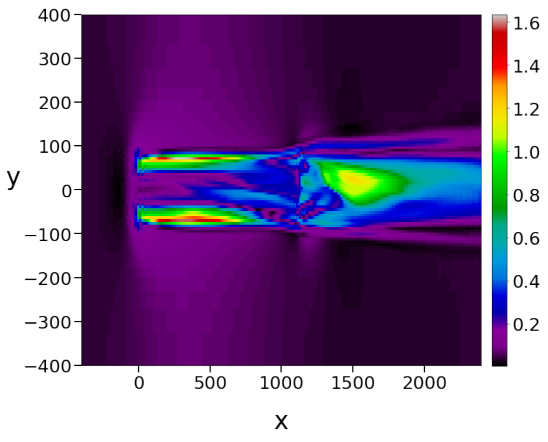
(c)



(d)



(e)



(f)

Figure 3.13: (a) OpenFOAM in System 5, (b) OpenFOAM in System 6, (c) dmd5, (d) dmd6, (e) discrepancy in system 5, (f) discrepancy in system 6. Visualised in the XY-plane.

In figure 3.14 the wake in the XZ-plane can be observed. CFD simulations, corresponding DMD reconstructions and the discrepancy in each system is included.

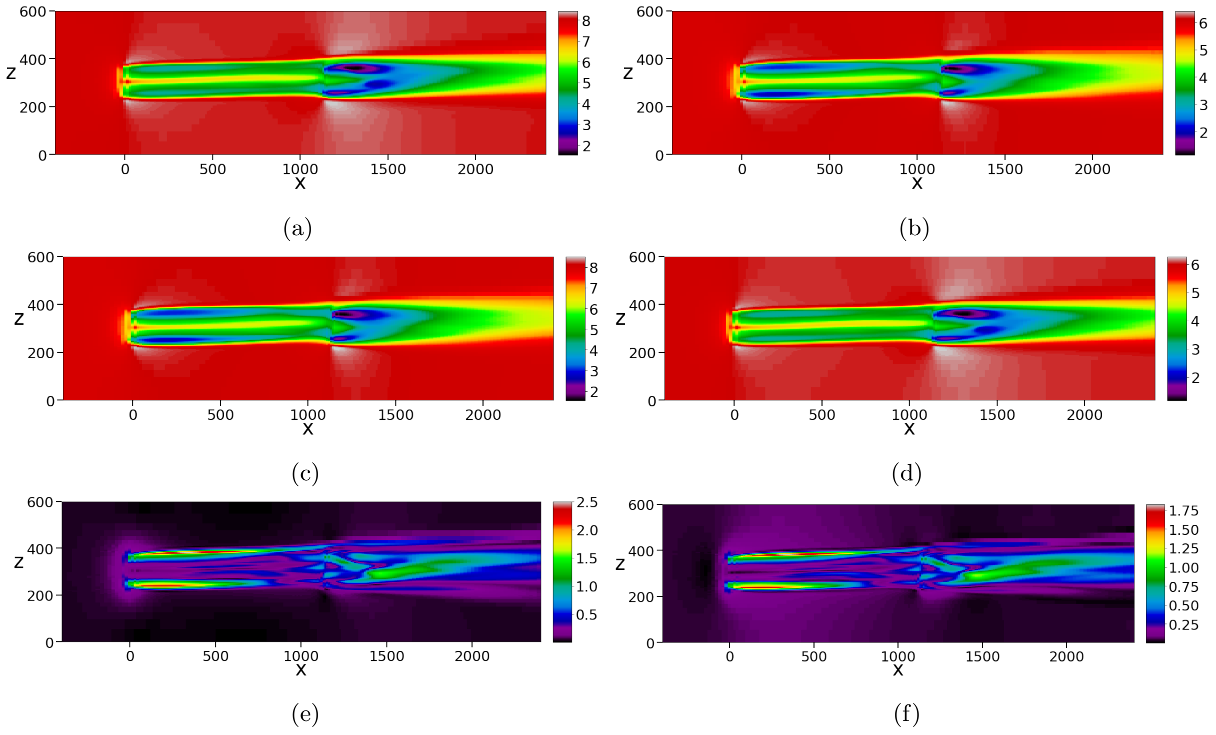


Figure 3.14: (a) OpenFOAM in System 5, (b) OpenFOAM in System 6, (c) dmd5, (d) dmd6, (e) discrepancy in system 5, (f) discrepancy in system 6. Visualised in the XZ-plane.

In figure 3.15 the wake in the YZ-plane can be observed. Once again, CFD simulations, corresponding DMD reconstructions and the discrepancy in each system are included.

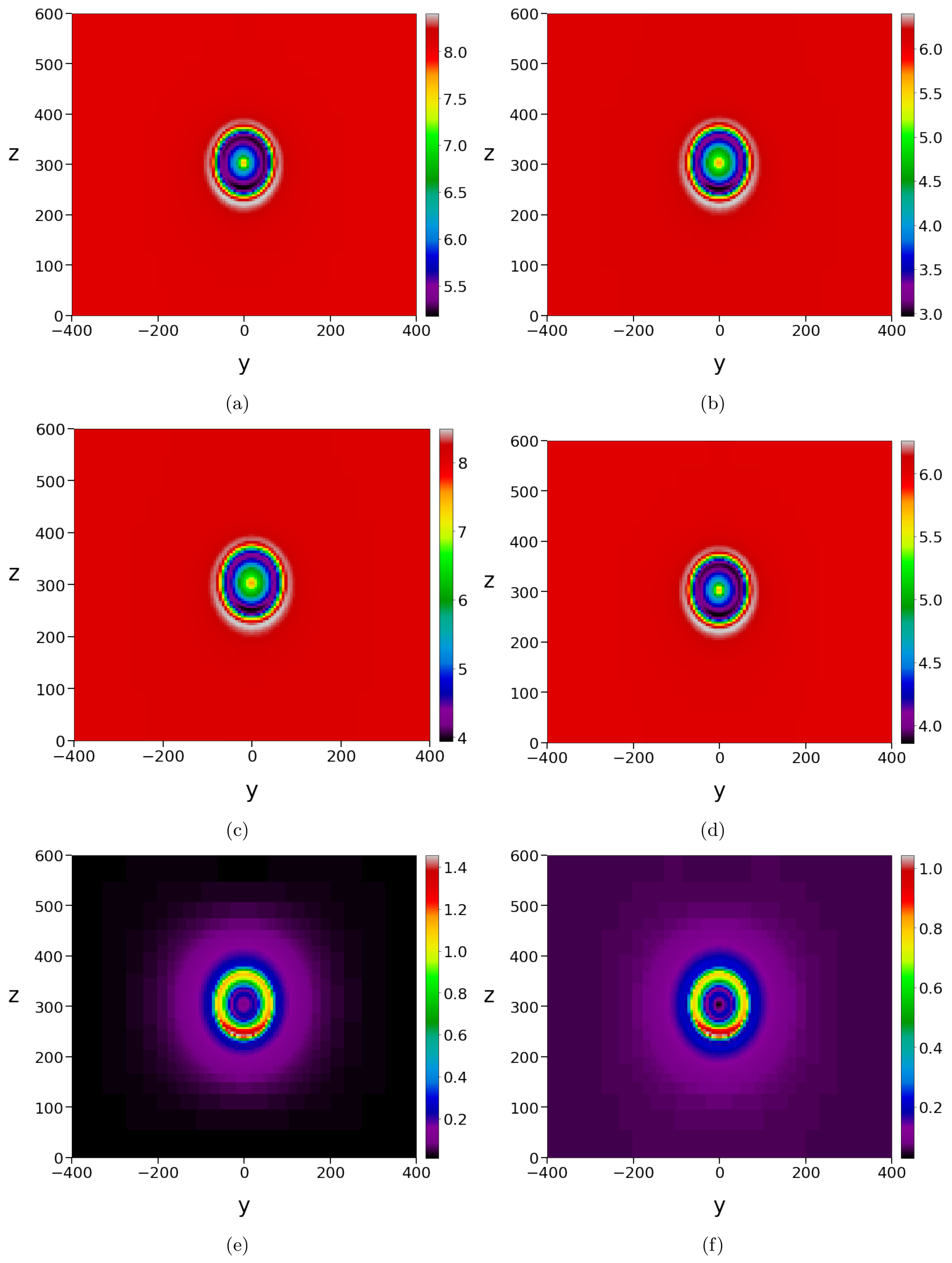


Figure 3.15: (a) OpenFOAM in System 5, (b) OpenFOAM in System 6, (c) dmd5, (d) dmd6, (e) discrepancy in system 5, (f) discrepancy in system 6. Visualised in the YZ-plane.

The distribution of energy was also examined parallel to the YZ-plane at x=1161, displaying the turbine located at (1161, 0, 300). CFD simulations, DMD reconstructions and the discrepancy in each system in this plane can be observed in figure 3.16.

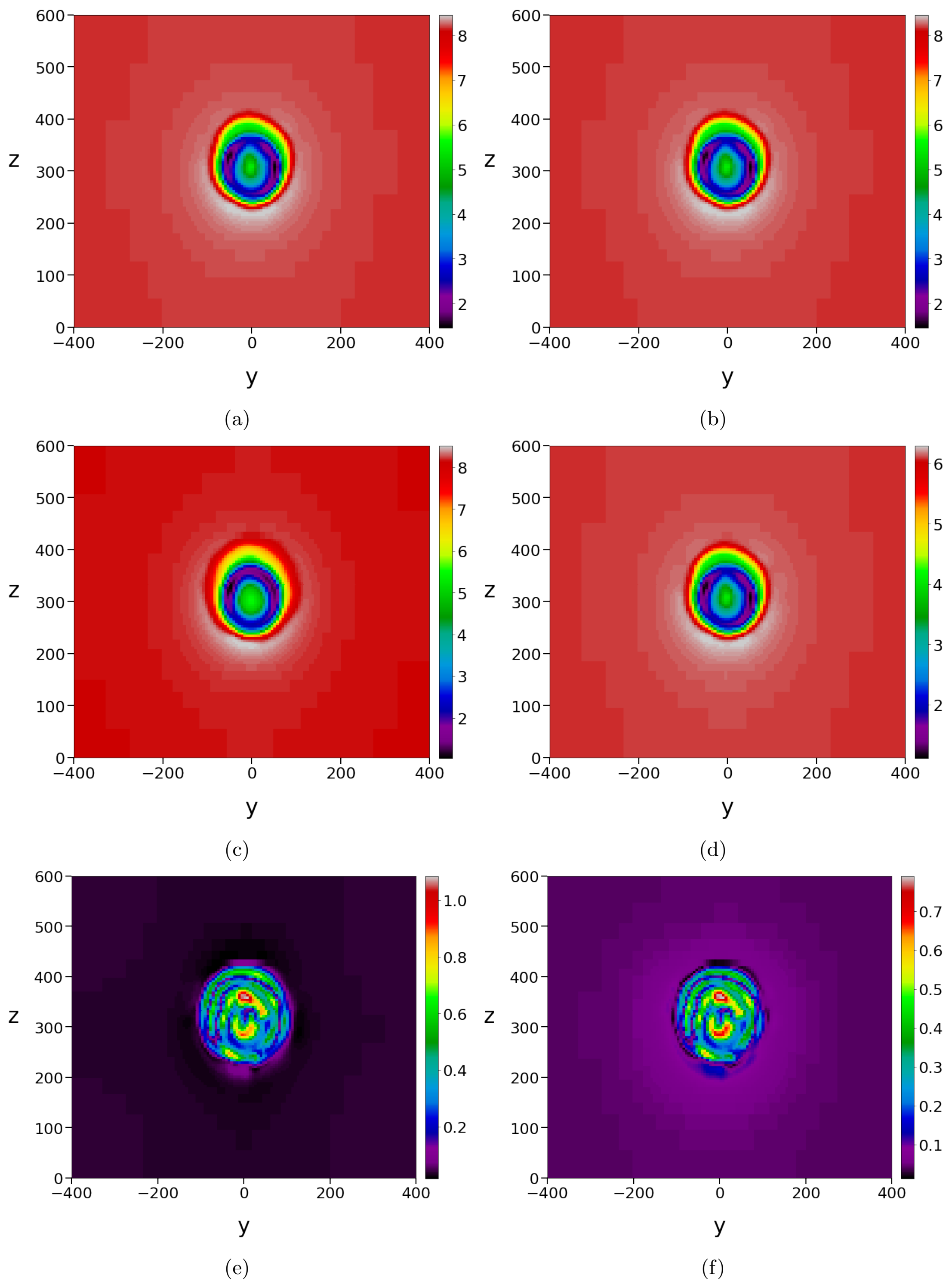


Figure 3.16: (a) OpenFOAM in System 5, (b) OpenFOAM in System 6, (c) dmd5, (d) dmd6, (e) discrepancy in system 5, (f) discrepancy in system 6. Visualised in the plane parallel to the YZ-plane at X = 1161 meters.

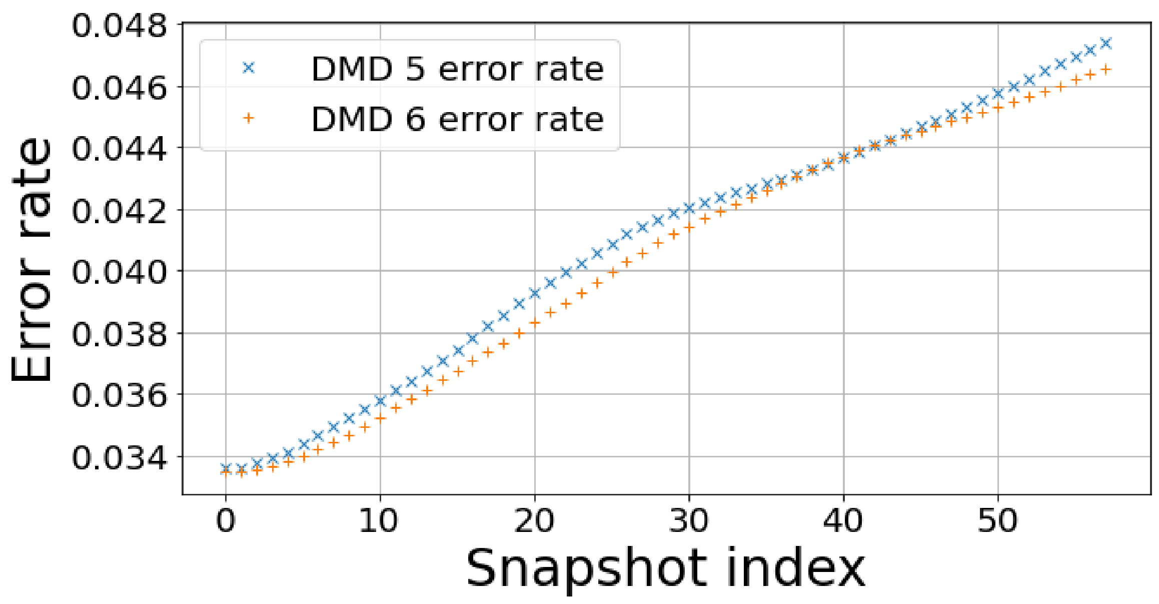


Figure 3.17: Error rates of each reconstructed slice for System 5 and 6.

The evolution of System 5 and System 6's error rates over time were also investigated, and can be observed in figure 3.17.

3.2.2 System 5 and 6 - power precision

Velocity readings were compiled, as explained in section 2.2.7, into a total of 40 data points: five locations at both turbines, once each for both CFD simulations and once each for both DMD reconstructions. The readings, as well as the REWS, can be observed in tables 3.8 and 3.9.

Table 3.8: Velocity readings for System 5 and 6 at Turbine 1.

	p1 [m/s]	**p2** [m/s]	**p3** [m/s]	**p4** [m/s]	**p5** [m/s]	**REWS** [m/s]
dmd5	5.060	5.235	6.498	5.422	4.512	**5.528**
OpenFOAM 8m/s	6.062	5.525	6.593	5.562	5.429	**5.880**
dmd6	4.517	4.116	4.913	4.144	4.045	**4.381**
OpenFOAM 6m/s	3.814	3.946	4.898	4.086	3.401	**4.167**

Table 3.9: Velocity readings for Systems 5 and 6 for Turbine 2.

	p6 [m/s]	**p7** [m/s]	**p8** [m/s]	**p9** [m/s]	**p10** [m/s]	**REWS** [m/s]
dmd5	3.821	3.794	5.275	4.328	1.999	**4.134**
OpenFOAM 8m/s	4.172	3.518	4.972	3.396	2.902	**4.142**
dmd6	3.116	2.625	3.703	3.273	2.161	**3.087**
OpenFOAM 6m/s	2.876	2.866	3.982	3.262	1.510	**3.119**

The power production calculated for Case 2 is shown in table 3.10. The errors in each system can be observed in table 3.11.

Table 3.10: Power, predicted and actual.

	Power, Turbine 1	**Power, Turbine 2**
System 5, dmd5	679.7 kW	270.0 kW
System 5, OpenFOAM, 8 m/s	817.8 kW	272.2 kW
System 6, dmd6	327.8 kW	91.71 kW
System 6, OpenFOAM, 6 m/s	278.9 kW	95.56 kW

Table 3.11: Error and error rates for predicted Power in Systems 5 and 6.

	Turbine 1		**Turbine 2**	
System	Error [kW]	Error rate	Error [kW]	Error rate
5	-138.1	16.89%	-2.200	0.8082%
6	48.90	17.53%	-3.850	4.029%

3.2.3 Time consumed

Total time: 1734 seconds, or 28 minutes and 54 seconds.

DMD only: 744.1 seconds, or approximately 12 minutes and 24 seconds.

dmd6 only: 11.85 seconds.

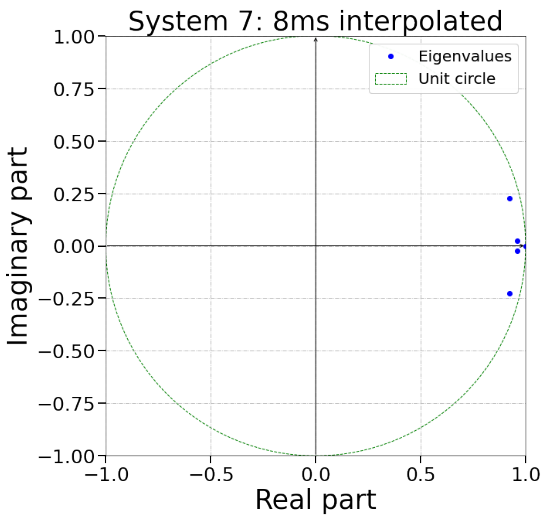
OpenFOAM simulation: approximately 24 hours per wind speed simulated.

3.3 Case 3

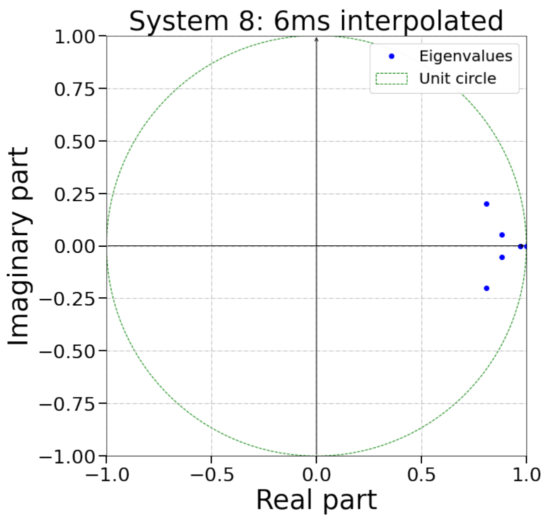
To summarise from section 2.5: Case 3 consisted of two OpenFOAM simulations and two DMD reconstructions for three turbines. Systems 7 and 8 were trained on 6 and 8 m/s respectively. Prior to reconstruction, they were interpolated to their target velocity. System 7's target velocity is 8 m/s, and System 8's target velocity is 6 m/s. Note that all slices are referred to by their index number, after cropping as described in chapter 2.5.

3.3.1 System 7 and 8 - velocity precision

Eigenvalues of System 7 and System 8 can be observed in figures 3.18.



(a)



(b)

Figure 3.18: (a) Eigenvalues of System 7 and (b) System 8.

The error rates, both peak and average, for System 7 and System 8 are showed in table 3.12, as well as the index number for the slice with the highest spacial error.

Table 3.12: Error rates for System 7 and System 8.

System no.	**Avg. error**	**Peak error**	**Peak error slice index**	**Last slice index**
7	2.493%	2.756%	40	40
8	2.444%	2.697%	40	40

The systems were further examined by visualising the distribution of energy in the XY, XZ and YZ-plane respectively. Visualisations were made of both OpenFOAM simulations, the reconstructions made by dmd7 and dmd8, and ultimately the discrepancy within each system. The XY-plane was displayed first for System 7 and System 8 with the origin at the Turbine 1 location as shown in figure 3.19.

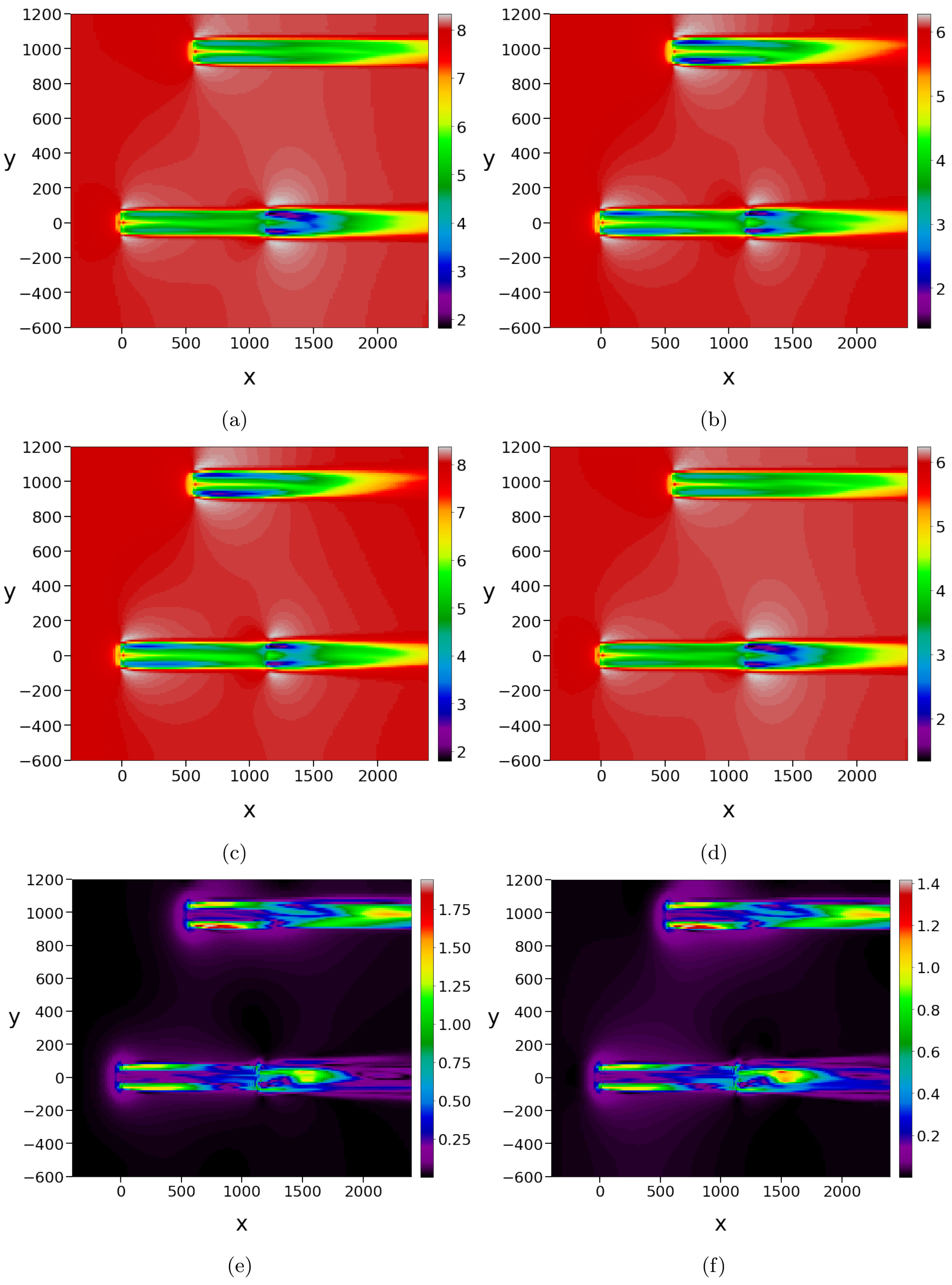


Figure 3.19: (a) OpenFOAM in System 7, (b) OpenFOAM in System 8, (c) dmd7, (d) dmd8, (e) discrepancy in System 7, (f) discrepancy in System 8. Visualised in the XY-plane.

In figure 3.20 the wake in the XZ-plane can be observed. CFD simulations, corresponding DMD reconstructions and the discrepancy in each system are included.

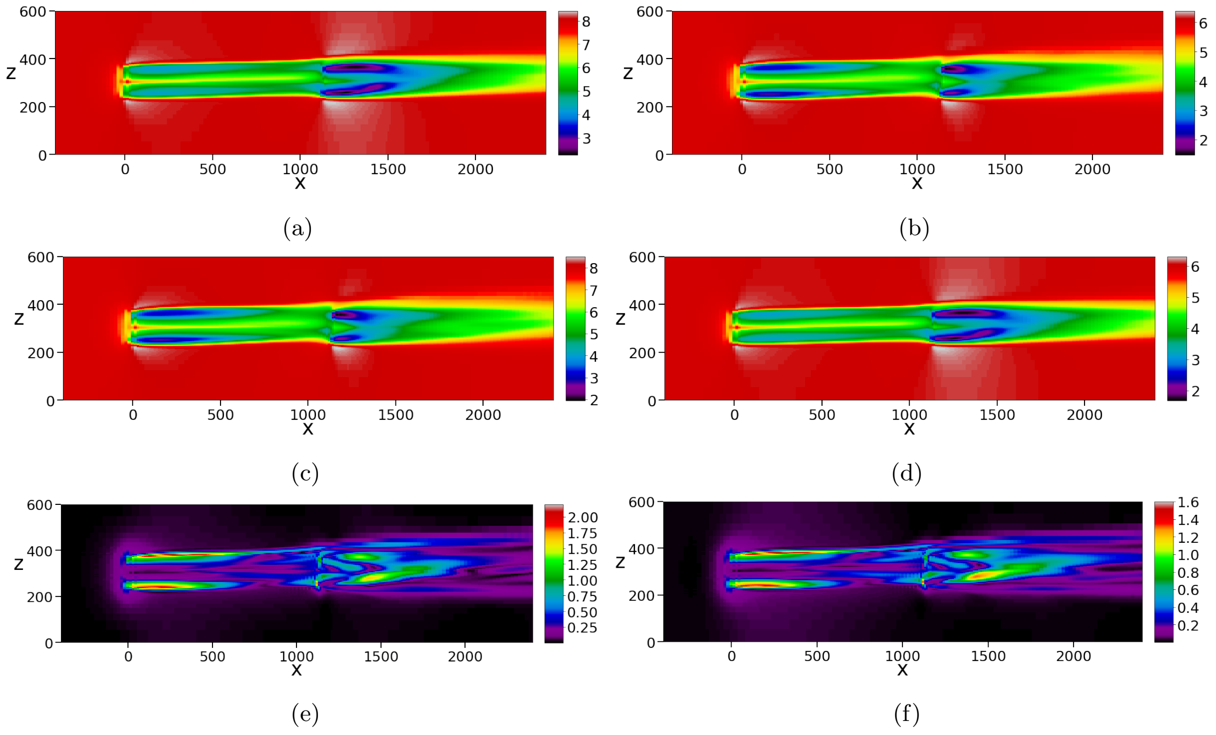
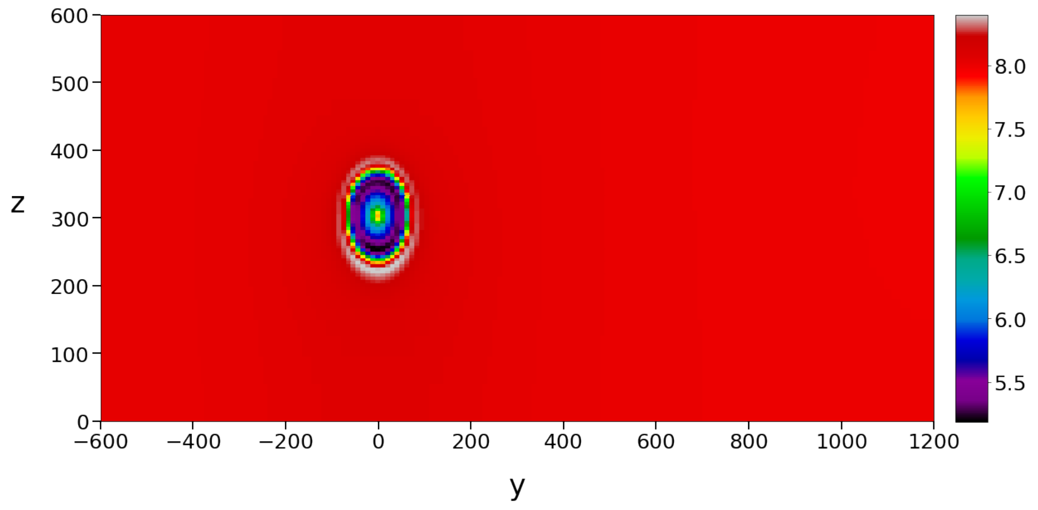
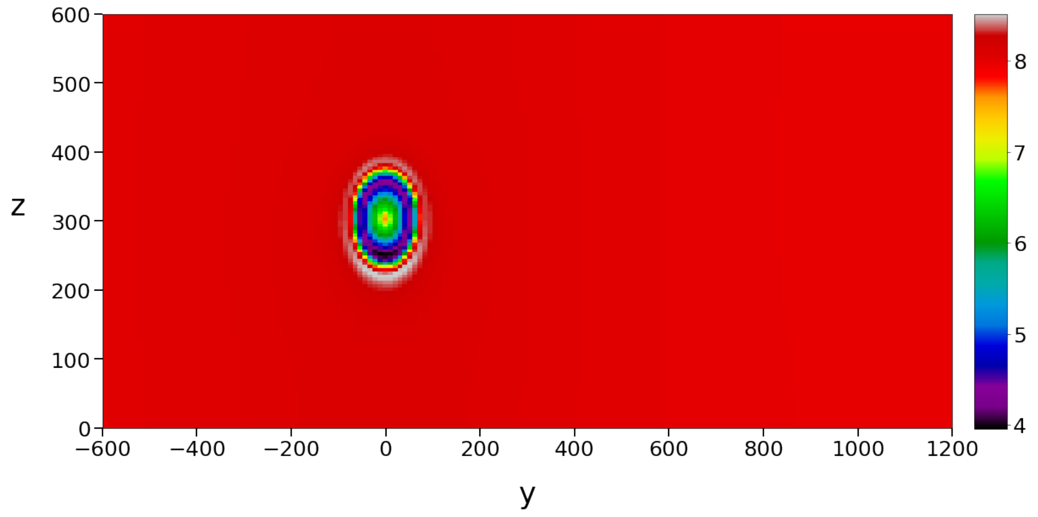


Figure 3.20: (a) OpenFOAM in System 7, (b) OpenFOAM in System 8, (c) dmd7, (d) dmd8, (e) discrepancy in System 7, (f) discrepancy in System 8. Visualised in the XZ-plane.

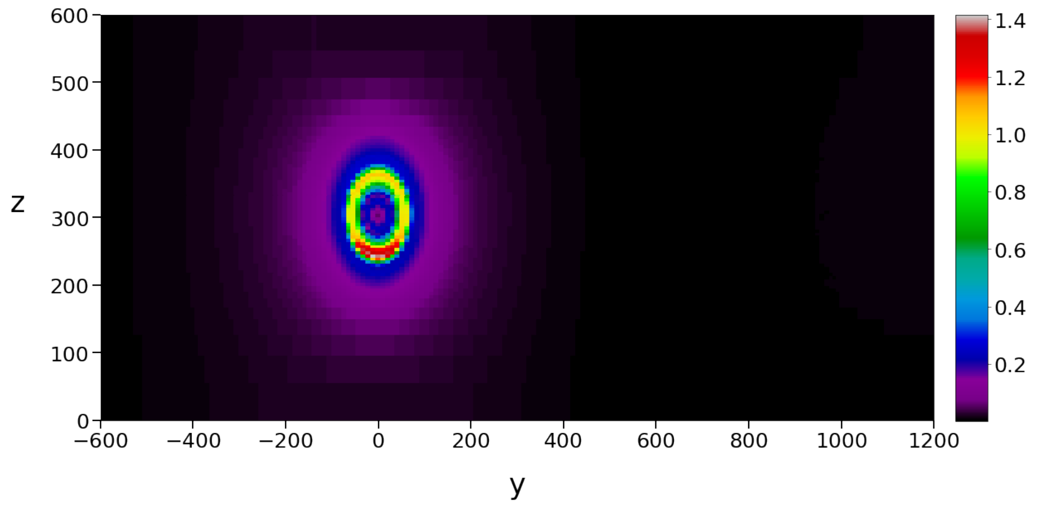
In figure 3.21 the wake in the YZ-plane for System 7 at X=0 can be observed. Once again, CFD simulations, corresponding DMD reconstructions and the discrepancy is included.



(a)



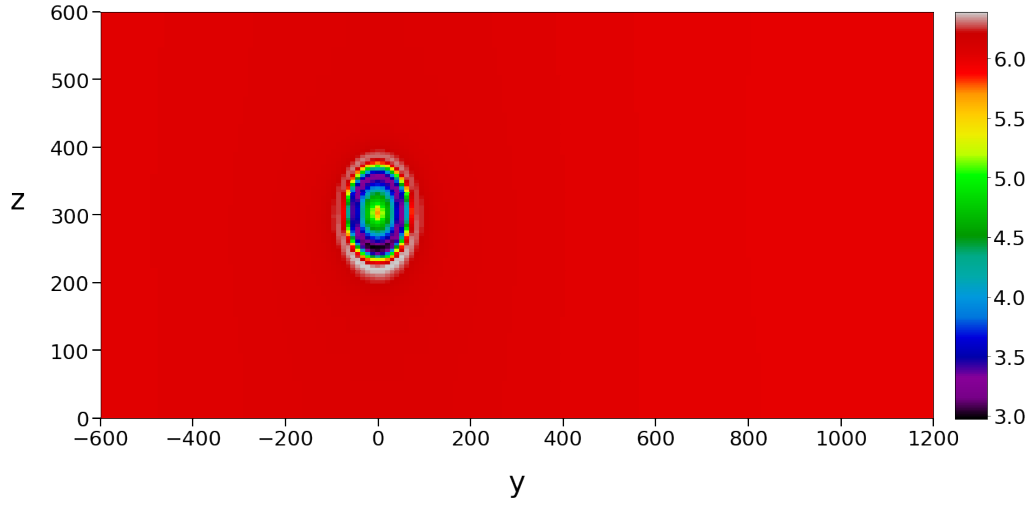
(b)



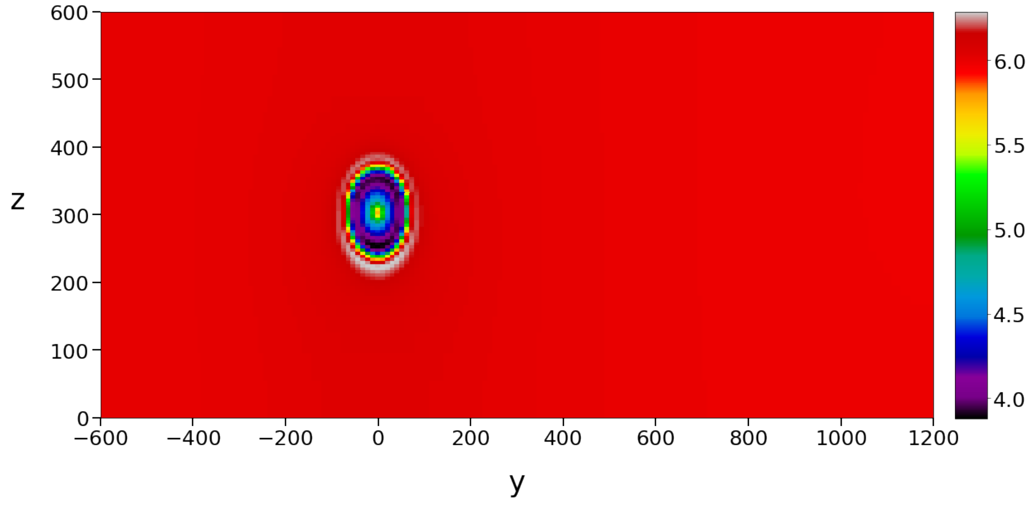
(c)

Figure 3.21: (a) OpenFOAM in System 7, (b) dmd7 (c) discrepancy in System 7, Visualised in the YZ-plane.

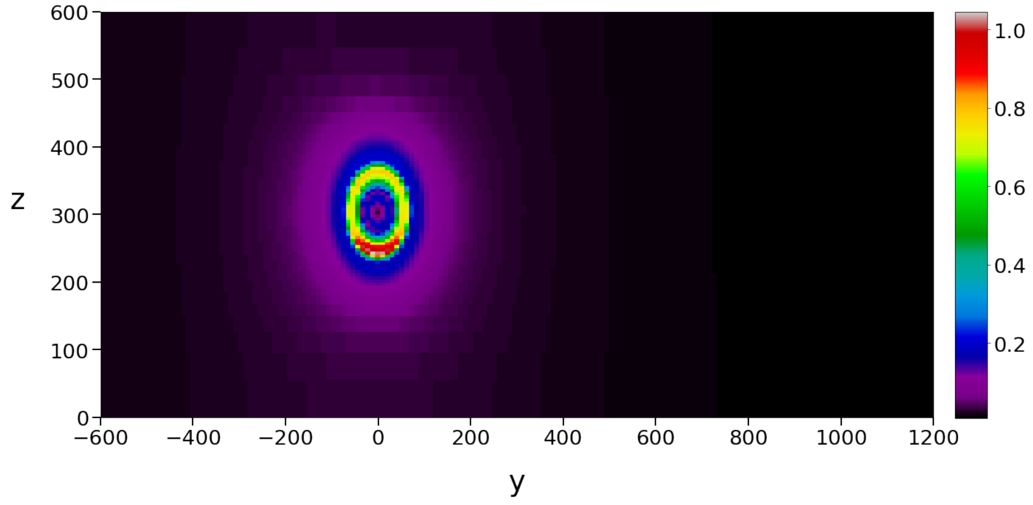
In figure 3.22 the wake in the YZ-plane for System 8 at X=0 can be observed. Once again, CFD simulations, corresponding DMD reconstructions and the discrepancy is included.



(a)



(b)



(c)

Figure 3.22: (a) OpenFOAM in System 8, (b) dmd8 (c) discrepancy in System 8, Visualised in the YZ-plane.

The distribution of energy was also examined parallel to the YZ-plane at x=1161, displaying the turbine located at (1161, 0, 300). CFD simulations, DMD reconstructions and the discrepancy for System 7 in this plane can be observed in figure 3.23.

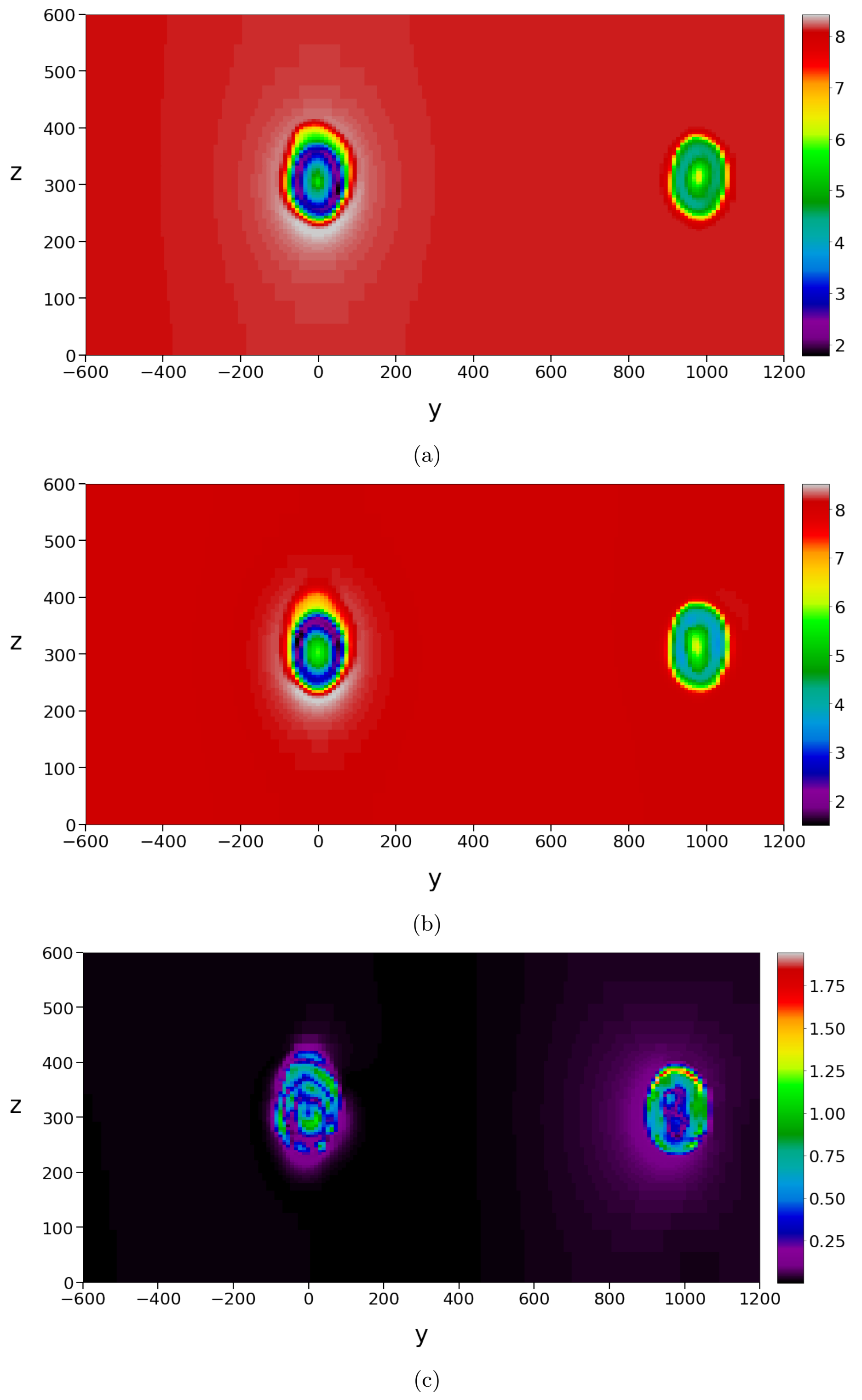
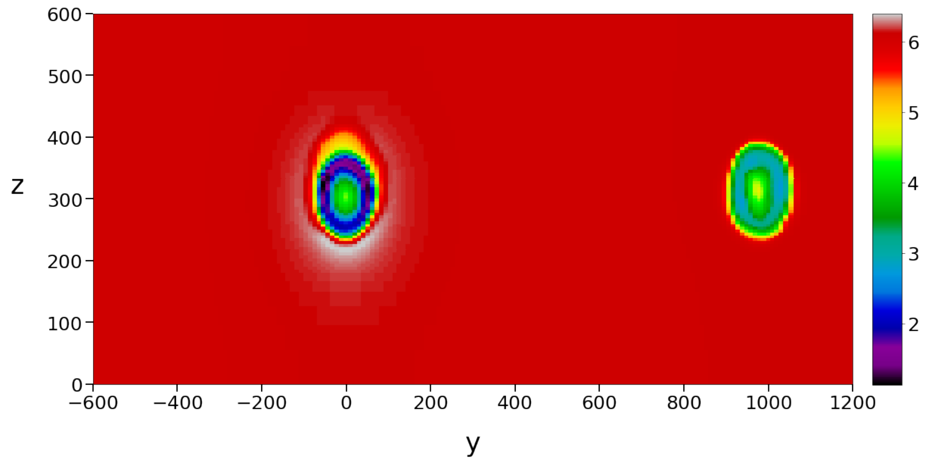
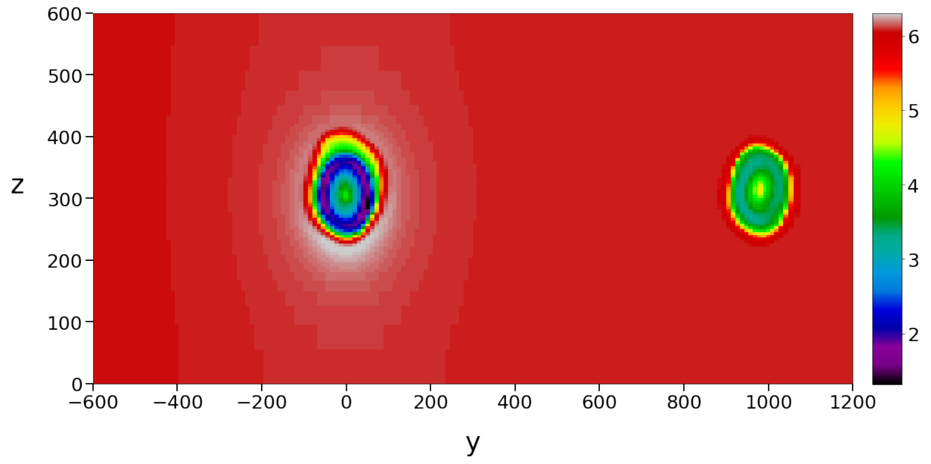


Figure 3.23: (a) OpenFOAM in System 7, (b) dmd7, (c) discrepancy in System 7. Visualised in the plane parallel to the YZ-plane at X = 1161 meters.

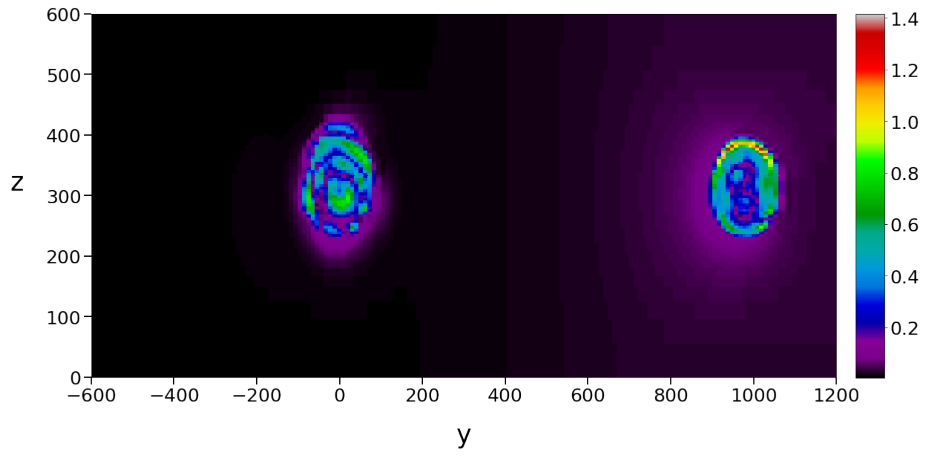
In figure 3.24 the wake for System 8 in the plane parallel to the YZ plane at X=1161 can be observed. Once again, CFD simulations, corresponding DMD reconstructions and the discrepancy is included.



(a)



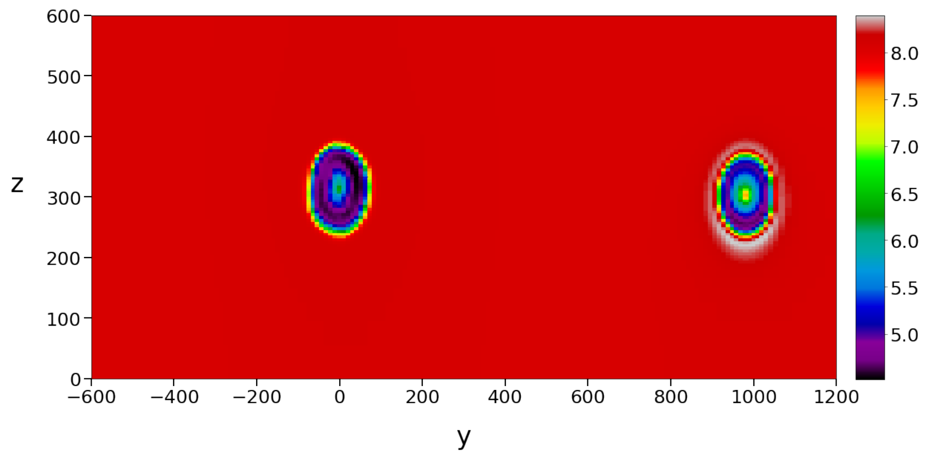
(b)



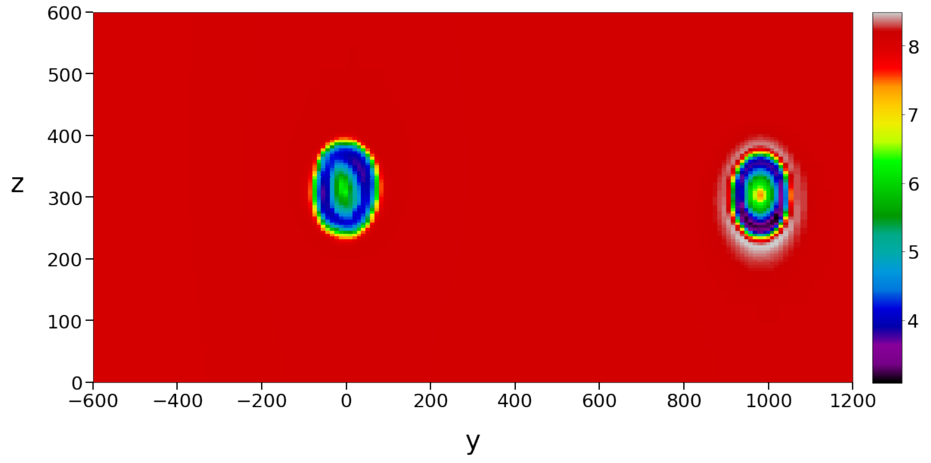
(c)

Figure 3.24: (a) OpenFOAM in System 8, (b) dmd8, (c) discrepancy in System 8. Visualised in the plane parallel to the YZ-plane at X = 1161 meters.

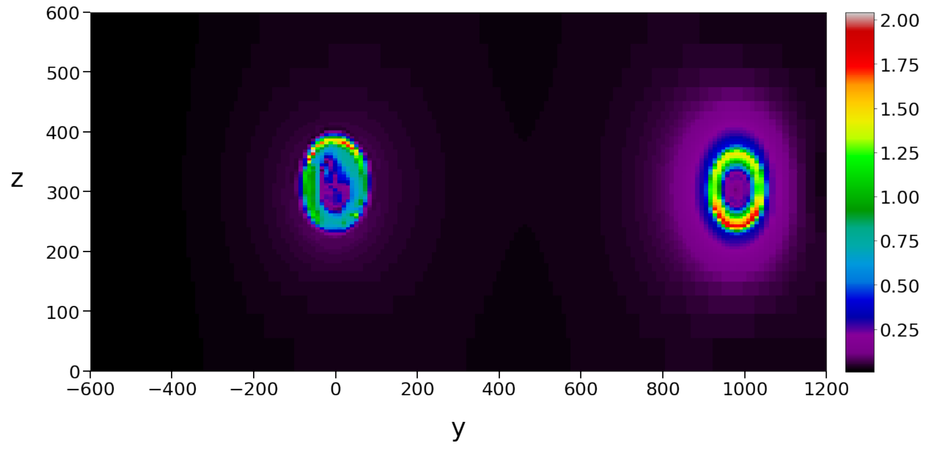
The distribution of energy was also examined parallel to the YZ-plane at X=580.5, displaying the turbine located at (580.50, 1005.46, 300). CFD simulations, DMD reconstructions and the discrepancy for System 7 can be observed in figure 3.25.



(a)



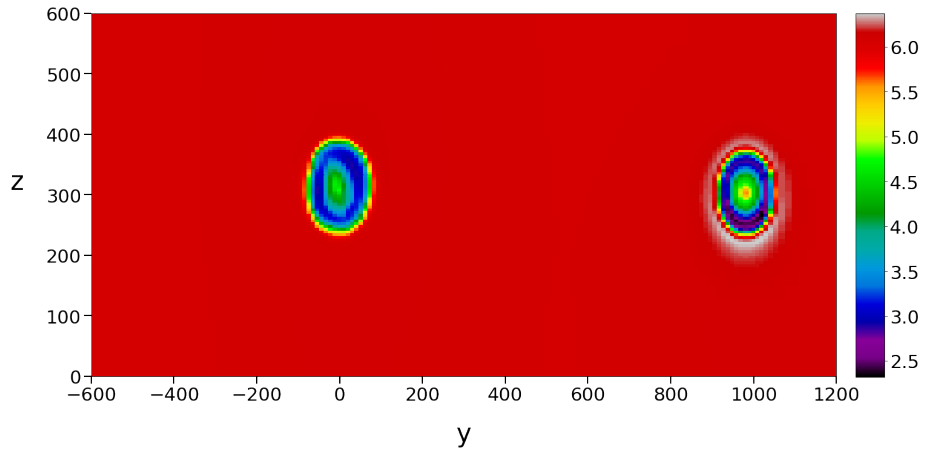
(b)



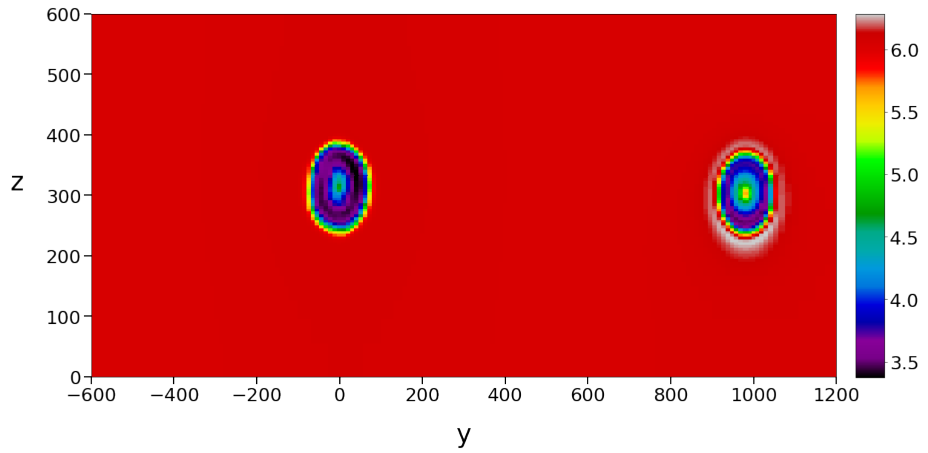
(c)

Figure 3.25: (a) OpenFOAM in System 7, (b) dmd7, (c) discrepancy in System 7. Visualised in the plane parallel to the YZ-plane at X = 580.5 meters.

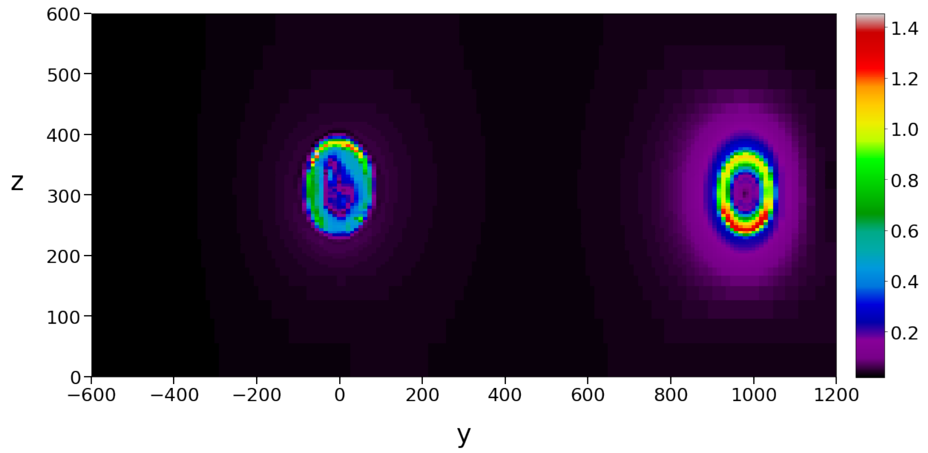
In figure 3.26 the wake for System 8 in the plane parallel to the YZ plane at X=580.5 can be observed. Once again, CFD simulations, corresponding DMD reconstructions and the discrepancy is included.



(a)



(b)



(c)

Figure 3.26: (a)OpenFOAM in System 8, (b) dmd8, (c) discrepancy in System 8. Visualised in the plane parallel to the YZ-plane at X = 580.5 meters.

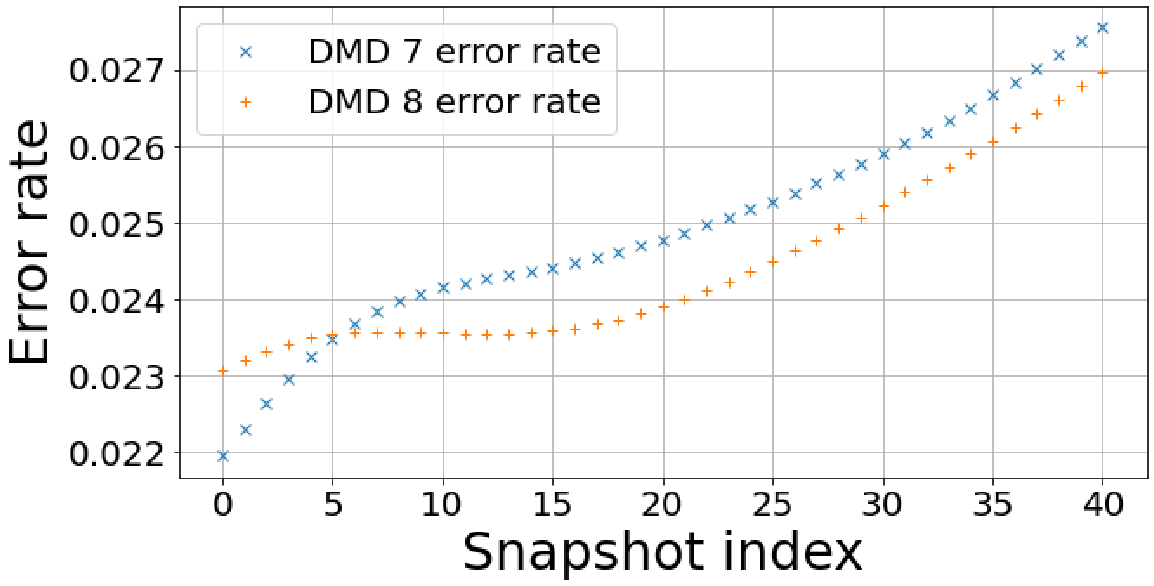


Figure 3.27: Error rates of each reconstructed slice for System 7 and 8.

Ultimately, the evolution of System 7 and System 8's error rates over time were also investigated, and can be observed in figure 3.27.

3.3.2 System 7 and 8 - power precision

Velocity readings were compiled, as shown in section 2.2.7, into a total of 60 data points: five locations at each turbine, once each for both CFD simulations, and once each for both DMD reconstructions. The readings, as well as the REWS, can be observed in tables 3.13, 3.14 and 3.15.

Table 3.13: Velocity readings for Systems 7 and 8 for Turbine 1.

	p1 [m/s]	**p2** [m/s]	**p3** [m/s]	**p4** [m/s]	**p5** [m/s]	**REWS [m/s]**
dmd7	5.114	5.231	6.417	5.416	4.555	**5.514**
OpenFOAM 8 m/s	6.087	5.514	6.467	5.540	5.448	**5.842**
dmd8	4.556	4.128	4.841	4.147	4.078	**4.373**
OpenFOAM 6 m/s	3.841	3.927	4.818	4.066	3.421	**4.140**

Table 3.14: Velocity readings for Systems 7 and 8 for Turbine 2.

	p6 [m/s]	**p7** [m/s]	**p8** [m/s]	**p9** [m/s]	**p10** [m/s]	**REWS [m/s]**
dmd7	4.155	4.085	5.488	4.457	2.435	**4.384**
OpenFOAM 8 m/s	4.308	3.646	4.948	4.325	3.004	**4.176**
dmd8	3.213	2.722	3.698	3.233	2.240	**3.119**
OpenFOAM 6 m/s	3.075	3.071	4.150	3.380	1.848	**3.307**

Table 3.15: Velocity readings for Systems 7 and 8 for Turbine 3.

	p11 [m/s]	**p12** [m/s]	**p13** [m/s]	**p14** [m/s]	**p15** [m/s]	**REWS [m/s]**
dmd7	5.036	4.267	5.613	4.921	4.776	**4.945**
OpenFOAM 8 m/s	6.279	5.039	5.782	5.396	5.912	**5.574**
dmd8	4.700	3.772	4.328	4.039	4.425	**4.172**
OpenFOAM 6 m/s	3.785	3.199	4.215	3.696	3.589	**3.715**

The power production calculated for Case 1 is displayed in table 3.16. The errors in each system can be observed in table 3.17.

Table 3.16: Power, predicted and actual for systems 7 and 8.

	Turbine 1	**Turbine 2**	**Turbine 3**
System 7, dmd7	672.3 kW	327.8 kW	483.5 kW
System 7, OpenFOAM, 8 m/s	801.2 kW	281.1 kW	694.7 kW
System 8, dmd8	325.4 kW	95.56 kW	278.9 kW
System 8, OpenFOAM, 6 m/s	272.2 kW	121.6 kW	186.4 kW

Table 3.17: Error and error rates for predicted Power in Systems 7 and 8.

	Turbine 1		**Turbine 2**		**Turbine 3**	
System	Error [kW]	Error rate	Error [kW]	Error rate	Error [kW]	Error rate
7	-128.9	16.10%	46.70	16.61%	-211.2	30.40%
8	53.20	19.54%	-26.04	21.41%	92.50	49.62%

3.3.3 Time consumed

Total time: 2972 seconds, or 49 minutes and 32 seconds.

DMD only: 899.9 seconds, or approximately 15 minutes and 0 seconds.

dmd6 only: 19.31 seconds.

OpenFOAM simulation: approximately 24 hours per wind speed simulated.

4 Discussion

The goal of this thesis was to investigate the possibilities of utilising Dynamic Mode Decomposition as an alternative method to the current industrial standard of computational fluid dynamics. This study indicates that the DMD has the potential for producing precise reconstructions, with the great benefits of sharply reducing simulation time and hardware requirements. Among the key findings in this study are the DMD reconstruction's low overall error rate in regards to wind velocity, a remarkable reduction in simulation time, and an error rate for produced power that varies from virtually negligible to quite significant. Although the findings give cause to be optimistic, a wider spectrum of simulations is necessary to further validate the versatility and test the limits of Dynamic Mode Decomposition.

To summarise the tests performed, three different turbine setups have been examined, labelled as Case 1, Case 2 and Case 3. The Case's number is also representative for the number of turbines simulated in each case. There are a total of 8 systems simulated, each system containing one OpenFOAM simulation and the corresponding DMD reconstruction.

4.1 Velocity precision analysis

The bedrock for evaluating the precision of the DMD in this thesis was the analysis of the wind flow's velocity. The DMD's impressive potential for accurate reconstruction was thoroughly demonstrated by Systems 1 and 2. However, with regards to creating a tool for simplified CFD analysis, Systems 3-8 were of much higher significance, as Systems 1 and 2 did not include an interpolated DMD reconstruction. System 1 and 2 are therefore excluded from this section of the discussion.

The data obtained from simulating Systems 3-8 indicated a direct correspondence between the complexity of the simulated system and the overall velocity error rate for the DMD reconstruction. Each case expanded by introducing one additional wind turbine. From Case 1 to Case 2 there was a noticeable escalation in the overall error rate. Case 3 was essentially the combination of Cases 1 and 2, and due to the increased simulated area, it can be bewildering to directly compare overall error rates between Case 3 and either Case 1 or 2. The simulated area in Case 3 was 2.25 times larger than in Case 1 and 2. One can observe each System's peak error rate, as well as its slice index, in table 4.1.

Table 4.1: Overall velocity error rates and more of Systems 3-8.

	System	Avg. error	Peak error	Peak error slice index	Last slice index
Case 1	3	2.056%	2.355%	180	180
	4	2.050%	2.417%	180	180
Case 2	5	4.084%	4.737%	57	57
	6	4.035%	4.653%	57	57
Case 3	7	2.493%	2.756%	40	40
	8	2.444%	2.697%	40	40

From table 4.1, it also becomes prevalent that the peak error occurred in the last slice for every single system except the excluded Systems 1 and 2. As further visualised in sections 3.1.2, 3.2.1 and 3.3.1, growing overall error rate was a trend for the majority of the time spent simulating, again disregarding Systems 1 and 2. This development alone gives cause to expand the duration of the simulations for further examination.

Turbine 1 was the only turbine included in all of Systems 3-8. The discrepancy within the systems in the YZ-plane at Turbine 1's coordinates varied between the systems. System 3 and 4, the only ones restricted to one turbine, had peak absolute errors of approximately 0.7 and 0.47 m/s. System 5 and 6, on the other hand, had peak absolute errors of roughly 1.4 and 1.0 m/s, and the errors in System 7 and 8 are practically identical to the ones found in System 5 and 6. All of these peak errors occurred in areas that were of high interest for wind power extraction. These results imply that when the complexity of the simulated system increases, one should expect a raise in error rate specifically at the first turbine that interacts with the wind flow. This was a somewhat unexpected result: an increase of errors propagated further downwind was the expected consequence of a more advanced simulated turbine layout.

When observing the wake in Systems 3-8 in either the XZ or XY-plane, it became apparent that there were errors in the system of greater amplitude than the ones appearing at the YZ-plane. System 5 from Case 2 exemplified this, experiencing error peaks that surpassed 2.0 m/s in the wake of Turbine 1, equating to more than 25% of the inlet velocity. This can be observed in figure 4.1 Interestingly, System 7 and 8 encountered smaller error rates than Systems 5 and 6 in the XY-plane. However, unlike the errors peaks in the YZ-plane, these errors were located at geographical points with limited relevance with respect to wind power extraction.

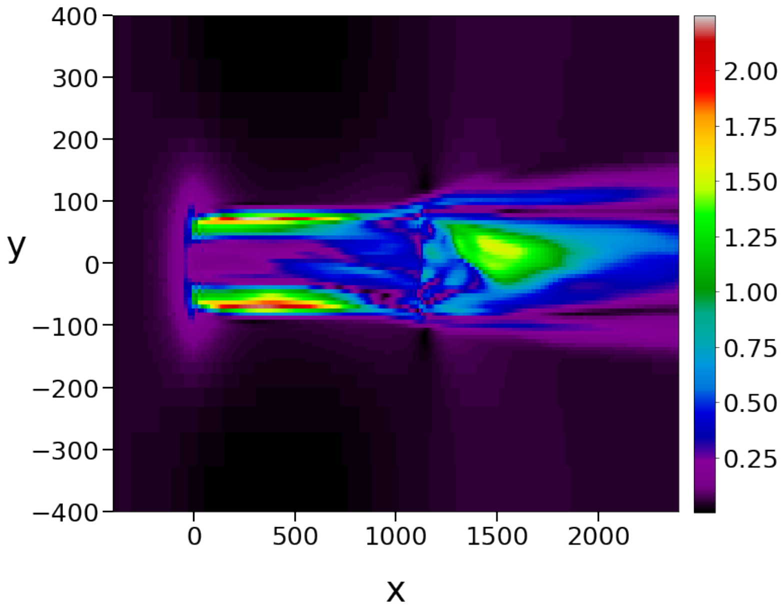


Figure 4.1: Errors in the wake of System 5, visualised in the XY-plane.

Shifting the focus to the plane across the face of Turbine 2, or in Case 1 the envisioned second turbine, the results when comparing the cases were surprising anew. The discrepancy in both systems in Case 1 was substantial, with peaks approaching 1.2 m/s. In Case 2, however, the peaks were reduced to barely eclipsing 1.0 and 0.7 m/s for System 5 and 6 respectively. With regards to power extraction, the location of the peak error were deemed favourable in Case 2 compared to Case 1, although the errors were not completely inconsequential in this regard. For System 7 and 8, disregarding readings at irrelevant Y-coordinates, the error peaks were approximately 1.0 and 0.8 m/s. These results suggest that, contrary to expectations, the error amplitudes were *not* worsened at the location of Turbine 2 as a consequence of the increased complexity of the simulated system from Case 1 to Case 2. It was therefore implied that Turbine 2's complex effects on the wind flow in between Turbine 1 and 2 were handled to a great extent by the DMD. The deviance between Case 2 and 3 seemed to be narrow, but given the change of colour palettes, it was more suitable to compare these cases by their predicted produced power, as examined in section 4.2.

Altering the focus once more to Turbine 3, there were only two systems to consider, System 7 and 8. The absolute error peaks, disregarding irrelevant Y-coordinates, eclipsed 2.0 and 1.4 m/s. These are worrisome amplitudes and might indicate that DMD's practicality is limited in regards to multiple columns of wind turbines perpendicular to the direction of the wind. The ramifications of these errors will be further demonstrated in section 4.2.

In Cases 1 and 2, there were no juxtaposed turbines. In Case 3, however, one such turbine was included giving a limited basis on which to evaluate how the DMD handles horizontal wake effects between rows of turbines. The examination of Case 3's projected velocities indicated that the errors obtained between the turbines were virtually unaffected by the introduction of juxtaposed turbines. This opens up the possibility of tinkering with the park's infrastructure layout, such as decreasing the distance between turbines perpendicular to the wind direction. This is only doable on sites with extremely predictable wind conditions.

There seemed to be a correlation between large variations in velocity over small areas and substantial errors in the DMD reconstruction. While the tests performed in this report are not tailored towards examining this issue, it is an interesting topic that could be elaborated on with specified tests in the future.

4.2 Power precision analysis

While the primary method of evaluating the precision of the DMD reconstructions in this thesis was analysing the velocity, the precision of predicted power produced is of higher importance with regards to wind farms. System 1 and 2 did not include predictions of power produced and are therefore deemed irrelevant for this section.

Overall, the DMD reconstructions varied from impressively precise to absolutely horrific when predicting the produced power for the three turbines. Concentrating initially on Turbine 1, all of Systems 3-8's total error and the error rate can be observed in table 4.2. All data is presented at the timestamp of the peak overall error rate for each individual system.

Table 4.2: Error and error rates for Systems 3-8 at Turbine 1.

	Case 1		**Case 2**		**Case 3**	
System	3	4	5	6	7	8
Error [kW]	-77.30	22.80	-138.1	48.90	-128.9	53.20
Error rate	8.596%	6.704%	16.89%	17.53%	16.10%	19.54%

As can be observed in table 4.2, the general trend was that an increase in the system's overall complexity was accompanied by an increase in error rate. Worth noting is also the pattern that dmd3, 5 and 7, which all had a greater target velocity than training velocity, estimated a lower power production than the 8 m/s OpenFOAM simulation. For dmd4, 6 and 8, the opposite was true; they all predicted a higher power production compared to the 6 m/s OpenFOAM simulation.

For Turbine 2, which in Case 1 was an envisioned turbine, all of Systems 3-8 total error and error rates are displayed in table 4.3.

Table 4.3: Error and error rates for Systems 3-8 at Turbine 2/envisioned turbine.

	Case 1		**Case 2**		**Case 3**	
System	3	4	5	6	7	8
Error [kW]	-83.70	36.10	-2.200	-3.850	46.70	-26.04
Error rate	14.60%	19.00%	0.8082%	4.029%	16.61%	21.41%

None of the patterns observed at Turbine 1 were present at Turbine 2. First, and most surprisingly, there seemed to be no correlation between the systems' overall complexity and error rate at Turbine 2. It was instead the two systems in Case 2 that obtained the greatest precision. As a matter of fact, System 5 and 6's error rates at Turbine 2 were the lowest error rates in regards to power produced in the entirety of this thesis. The two systems in Case 1 followed the pattern earlier proposed, where System 3 predicted too low power produced and System 4 predicted a value too high, but the systems in Case 2 and 3 broke away from this pattern.

Shifting focus to Turbine 3, the only results produced were from System 7 and 8. In both instances, the error rates were abysmal, at 30.40% and 49.62% respectively. Reminiscing briefly from section 4.1, the peak errors at this turbine were approximately 2.0 and 1.4 m/s. Having observed similar values in others systems earlier, it must be concluded that the errors at Turbine 3 were at geographical points of considerable importance for wind energy extraction. This result indicates that horizontal extension perpendicular to the direction of the wind is problematic for the Dynamic Mode Decomposition, and it might prove to be a focal point for further development and improvement of DMD as a substitute for traditional CFD methods.

The simulations performed in this dissertation indicate that there's a relationship between the complexity of the simulated system and peak power errors. While Case 1 had a higher error rate than Case 2 on

Turbine 2 (or the envisioned turbine, in Case 1), the Case 2 systems had a worse error rate on Turbine 1. One could argue that Case 1 experienced a higher total amount of errors, but for Case 1 it was Turbine 1 that was of paramount importance. Case 3, on the other hand, had at Turbine 2 a comparable error rate with Case 1, however it also included the horrific error rates obtained at Turbine 3.

As a closing note on the power analysis, an equal absolute error in reconstructed REWS would produce an unequal error with regard to power at other velocities. The power curve of the NREL offshore 5-MW baseline wind turbine is anything but linear. The precision of the produced power predicted by the DMD will therefore, given an equal absolute error, be immensely dependent on the absolute velocity of the wind. In the interval 8 - 10 m/s, for instance, the power curve is steeper than at the interval investigated in this thesis, and the error in produced power would be even greater. On the contrary, at the threshold of the rated power section, a reconstructed REWS with a positive error would have an error of zero in regards to power predictions.

4.3 Time saves

One of the most promising features of the DMD as a substitute for traditional CFD analysis is the potential for time-saving. To create a foundation on which to compare the DMD with the OpenFOAM simulations, three time intervals were defined: ***Total time***, ***DMD only*** and ***dmdx only***. For each case, these three values, as well as the time spent creating the necessary OpenFOAM files, can be observed in table 4.4.

Table 4.4: Time used and number of slices for each case.

	OpenFOAM sim. time	Total slices	Total time	DMD only	dmdx only
Case 1	24 hours	181	1651 sec.	884.3 sec.	13.55 sec.
Case 2	48 hours	58	1734 sec.	744.1 sec.	11.85 sec.
Case 3	48 hours	51	2972 sec.	899.9 sec.	19.31 sec.

The immense time savings were immediately obvious and remain the primary selling point of the DMD as a substitute for OpenFOAM simulations. Time saves surpassed 96% when comparing the OpenFOAM simulation time to the ***Total time*** for DMD in Case 3, the most time-consuming Case in this dissertation. This was a massive improvement and could, given an acceptable error rate, enable the implementation of CFD analysis in a wide variety of previously unconsidered cases.

Excluding hardware differences, which will be elaborated on in section 4.4, there were several factors influencing the time used in each case. The general complexity of the physical properties varied a lot between cases. Both Case 1 and Case 2 shared the same grid size, as mentioned in section 2.2.1, while the volume simulated in Case 3 was more than twice the size of the volume in Cases 1 and 2. Moreover, one additional turbine was introduced from each case to the next, which also greatly affected the systems' overall complexity. The total number of snapshots used for training and recreation differed as well. All available snapshots were utilised for training, with the consequence that Case 1 systems were trained on 181 snapshots, Case 2 systems were trained on 58 snapshots and Case 3 systems were trained on 51 snapshots. The same deviance between the systems was found in the number of snapshots recreated, which together with the simulated volume were the two main contributors to the duration of the dmdx only time.

Regarding the ***Total time***, Case 1's duration was artificially raised compared to the other cases. A lot of plotting was performed in Case 1 that was deemed unnecessary for the two other cases, as well as more finely detailed results produced for the extended results in appendixes C.1 and C.2. In contrast, Case 3's ***Total time*** was artificially lowered, due to how the systems were cropped. In Cases 1 and 2, all OpenFOAM files were initially utilised, and then cropping was performed based on visualisation in Paraview. In Case 3, however, the files deemed invalid for examination were removed beforehand. This had an impact on the time spent converting CSV files and loading data dictionaries from said CSV files. The cropping performed in Case 1 and 2 was almost instant, and could therefore be neglected.

When examining the ***DMD only*** column in table 4.4, there are some deviation in time spent for the various cases. This was mainly due to a deviation between the size of the data dictionaries, but as the DMD had already been performed, the differences were quite small. However, it is in the ***dmdx only*** column where the full potential for time savings are properly made apparent. Given a decent accuracy on the first established reconstruction, a completed DMD system could continuously produce high-quality reconstructions at an incredible rate. For the most complex systems simulated in this dissertation, Systems 7 and 8, the DMD is theoretically capable of producing reconstructions for five new velocities per minute. Producing the same amount of results with OpenFOAM would take upwards of 100 hours. This comparison is done with a notable deficit in hardware power. If this hardware deficit was equalised, the DMD time savings could be even greater.

4.4 Hardware

Compared to a traditional CFD approach, substituting in DMD will reduce the hardware requirements for calculation, thereby saving resources and attracting the relevant industries. When comparing the hardware of the two machines used in this thesis, as shown in table 2.1, there were a couple of factors that were considered negligible during comparison. Both of the operating systems used were Linux based systems and both machines had the same amount of RAM. Neither should have any major impact on the performance shown in this thesis.

The number of cores and the clock speed however directly corresponds to how fast calculations can be done, and will therefore inevitably have an impact on the results. When comparing the number of cores used the SINTEF machine has a clear advantage by having used (at most) 24 of its 47 cores during CFD simulations. In comparison, the NTNU VM used for DMD testing only has 8 cores available. The largest difference in cores can be noted in Case 3, where the DMD simulation used a total of 8 cores at 2.4GHz whilst the CFD simulations used 24 cores at 2.1GHz. The clock speed of the two machines are fairly similar but the difference in cores is notably larger. The storage used for the DMD tests was network storage-based and although there might have been some minor bottlenecks dependant on NTNU's internal network speed, the potential time losses were deemed negligible.

4.5 Limitations and sources of error

There were several possible limitations and sources of error present in this thesis. The velocity readings were obtained by averaging the velocities in a restricted area of 1x1 meters. Five such velocity averages were obtained and then weighted as described in section 2.2.7. This was a slight adjustment from the documented method utilised for REWS calculations. This was done since attempting to obtaining single-point readings for given Euclidean coordinates proved a fruitless pursuit. This approximation of the REWS method was deemed sensible by supervisors, but might still contribute to a small error in power calculations.

The inlet velocity used in Ashes was limited to two decimal digits. This gives reason to doubt the power calculations preciseness. Although the precision was under question it was only at a fraction of the minimum speed used in this thesis. This meant only wind speed variations below 0.00 were unaccounted for and from this we can infer that they had no major repercussions. The discrepancy is still worth being aware of.

The results concerning the time intervals ***Total time***, ***DMD only*** and ***dmdx only*** were defined after the code had been customised to fit each case. The time taken to customise each code was therefore not accounted for.

During the first CSV conversion of each OpenFOAM simulation an error occurred, as previously described in section 2 and further displayed in appendix D. Avoiding this error was not a focus for the group as it had no notable effect, and the CSV files were successfully converted upon re-executing the code. Regardless of this, the fact that the error occurred means that a delay was present during testing and unaccounted for in ***Total time***.

The results section displays snapshots at each individual system's peak overall error rate. The peak overall error rate did not necessarily correlate with the peak error at every point in the system. When examinations were performed in specified sections of the simulated volume, the time of the peak overall error was chosen. This has led to visualisations being presented at the peak error for the restricted volume, with a possibility of larger errors appearing at the given volume at a separate timestamp.

4.6 Further work

While the tests performed in this study shows great promise for the utilisation of DMD as a means to reduce simulation time and hardware requirements, further work is to be desired. In essence, the envisioned continuation of DMD testing and improvement can be sorted into three main categories:

1. Introducing a wider array of simulated conditions
2. Fundamental changes in the DMD code
3. DMD tests unrelated to wind energy extraction

First and foremost, a greater range of testing would strengthen the groundwork on which the DMD reconstructions are evaluated. Given Systems 3-8's increasing error rate over time, performing the same tests once more with an increased simulation duration would be logical. Introducing turbulence in the simulation would also be crucial as a means to achieve more realistic conditions, as turbulence is a fundamental concept in fluid mechanics. The ramifications of including turbulence in these tests are expected to be massive. The addition of turbulence would at least be partially beneficial, as turbulence is fundamental for wake suppression. However, by its very nature, turbulence is all but linear, and would almost certainly cause complications for the DMD reconstruction as well.

Expanding the variety of simulated velocities would also be favourable. As the power curve of the 5MW NREL turbine's rotor is not linear, an adjustment in the examined velocity interval could either improve or worsen the DMD's accuracy regarding power predictions.

Continuing towards the goal of achieving more realistic simulated conditions, the DMD should be faced with an increased amount of variance. In this thesis, the influence of the surrounding terrain was not explored, nor was the effects of an irregular inlet flow velocity or a fluctuating wind direction. The wind park infrastructure itself could also be tinkered with, including height difference between turbines, varying crosswind and downwind spacing or a fundamentally distinct turbine layout from the one utilised in the Hywind Scotland park. Given the results in this thesis, it would be especially interesting to examine whether obtaining precise power predictions are possible with more than one column of turbines, as well as examining three turbines in a single column.

Returning to the aforementioned changes in the code, it is probable that creating DMD reconstructions based on two OpenFOAM simulations instead of one could increase the overall accuracy. There are also several subcategories not utilised in this dissertation, such as Multiresolution DMD, Compressed DMD, Forward-Backward DMD, Higher Order DMD and DMD with control. Incorporating one or more of these might prove advantageous.

It could also prove interesting to examine the wake only instead of the whole simulated volume presented in this thesis. For this to be obtainable, code restricting the volume analysed must be developed.

In this dissertation, velocity calculations for power predictions was performed with a slight adjustment to the REWS method. Instead of obtaining five single data points, the average over small areas was utilised. Given an alternation in the code, single data point values could be obtained, and thus the REWS method would be followed more strictly. Whether this would contribute towards a more or less accurate power reading is something that has yet to be explored.

As previously mentioned in sector 4.5, the overall peak error for a system might not correspond with the largest error at the turbine locations. For the sake of finding the largest discrepancies in power produced, it would be beneficial to develop a code identifying the peak error timing in more restricted sections of the simulated volume. This analysis would give a more realistic representation of the DMD's largest power related errors, and thus be more applicable for the wind energy industry.

Lastly, this thesis was originally meant to include analysis of noise propagation, but it was deemed a too burdensome task early on in the project. While usually not a point of great interest when considering offshore parks, it is of great importance for onshore parks located in the vicinity of houses or other areas where the noise or visual pollution is unpalatable. Given the efficient nature of the DMD reconstruction, it might be a valuable tool for analysing noise propagation, further increasing the DMD's potential value for the wind energy industry. Utilising the DMD to examine noise propagation could also be of interest for industries outside the wind energy sector.

5 Conclusion

In this dissertation, the goal was to validate the Dynamic Mode Decomposition as a substitute for OpenFOAM simulations. Various simulations were produced in OpenFOAM, recreated by the DMD, and subsequent results were compared. A total of three different turbine setups were examined, herein labelled as Case 1, Case 2 and Case 3. There are a total of 8 systems simulated, each system containing one OpenFOAM simulation and the corresponding DMD reconstruction.

The DMD reconstruction can produce similar simulations as an OpenFOAM simulation, and depending on the required accuracy and the system's complexity, DMD can be a suitable substitute for traditional CFD methods. The overall error rate for all systems simulated in this dissertation never surpassed 5%. However, it is of paramount importance to expand the foundation on which the DMD is validated. Given the trend of growing error rates over time, as well as the correlation between system complexity and error rate, is necessary to investigate how the DMD responds to more diverse simulations.

The low overall error rate does not imply that there were no large errors present in the systems. The greatest velocity errors surpassed 2.0 m/s, equating more than 25% of the inlet velocity. In some instances, these large errors were largely irrelevant when considering power produced by the turbines, yet for others the impact was significant.

The error rate of the power predictions fluctuated greatly in the systems simulated in this thesis. On the one hand, Systems 5 and 6 managed to predict the produced power at Turbine 2 with a margin of error of 0.8082% and 4.029% respectively. Case 3 however, calculated power produced with an error rate of immense proportions at Turbine 3, erring as much as 49.62% in System 8.

The immense reduction in time spent simulating remains the main advantage of the DMD compared to OpenFOAM simulation. Even considering the best-case scenario for OpenFOAM, two simulations performed from OpenFOAM versus one by OpenFOAM and one through DMD with interpolation, the time saves amount to approximately 50%. This would increase rapidly as more DMD reconstructions are made at a rate of more than five each minute, contrasted with roughly 12-24 hours for each simulation in OpenFOAM, dependent on the complexity of the files simulated.

This dissertation gives reason to be optimistic for the implementation of DMD as a substitute for traditional CFD methods in the future. However, a widened array of simulations is necessary to improve the validation. The primary goal should be to achieve more realistic simulation conditions, mainly through the inclusion of turbulence.

References

Astrid, M., & Lee, S.-I. (2018). Deep compression of convolutional neural networks with low-rank approximation. *ETRI journal*, *40*(4), 421–434.

Betz, D. .-I. D. A. (2013). The maximum of the theoretically possible exploitation of wind by means of a wind motor. *Wind Engineering*, *37*(4), 441–446. https://doi.org/10.1260/0309-524X.37.4.441

Burton, T., Jenkins, N., Sharpe, D., & Bossanyi, E. (2011). *Wind energy handbook, second edition* (2nd ed.). John Wiley & Sons Ltd.

Demo, N., Tezzele, M., & Rozza, G. (2018). Pydmd: Python dynamic mode decomposition. *Journal of Open Source Software*, *3*(22), 530. https://doi.org/10.21105/joss.00530

Eckart, C., & Young, G. (1936). The approximation of one matrix by another of lower rank. *1*(3). https://doi.org/https://doi.org/10.1007%2FBF02288367

Equinor. (2021). Flytende havvind i Equinor - equinor.com. Retrieved May 13, 2021, from https://www.equinor.com/no/what-we-do/floating-wind.html

European Union. (2021a). Objectives — Upwards wind project. Retrieved February 11, 2021, from https://www.upwards-wind.eu/objectives/

European Union. (2021b). The Paris Agreement — UNFCCC. Retrieved April 3, 2021, from https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement

Fraunhofer-Gesellschaft. (2021). Profile / Structure. Retrieved February 11, 2021, from https://www.fraunhofer.de/en/about-fraunhofer/profile-structure.html

Gu, R., Xiong, W., & Li, X. (2015). Does the singular value decomposition entropy have predictive power for stock market? — evidence from the shenzhen stock market. *Physica A*, *439*, 103–113.

Hau, E., & Renouard, H. v. (2013). *Wind turbines: Fundamentals, technologies, application, economics* (3. Aufl.). Springer-Verlag.

Heller, A. (2014). Predicting Wind Power with Greater Accuracy. *Wind Energy*, 9.

Jankowski, N., & Linowiecki, R. (2019). A fast neural network learning algorithm with approximate singular value decomposition. *International journal of applied mathematics and computer science*, *29*(3), 581–594.

Jonkman, J., Butterfield, S., Musial, W., & Scott, G. (2009). *Definition of a 5-MW Reference Wind Turbine for Offshore System Development* (tech. rep. NREL/TP-500-38060, 947422). https://doi.org/10.2172/947422

Kutz, J. N., Brunton, S. L., Brunton, B. W., & Proctor, J. L. (2016). *Dynamic mode decomposition*. Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9781611974508

Li, S. Z., & Jain, A. (Eds.). (2009). L2 norm. In *Encyclopedia of biometrics* (pp. 883–883). Springer US. https://doi.org/10.1007/978-0-387-73003-5_1070

Manwell, J. F., McGowan, J. G., & Rogers, A. L. (2009). *Wind Energy Explained*. John Wiley & Sons.

Pielke, R. A., & Hayden, B. P. (2021). Planetary boundary layer — atmospheric science. Retrieved March 15, 2021, from https://www.britannica.com/science/planetary-boundary-layer

Ritchie, H., & Roser, M. (2020). Emissions by sector [https://ourworldindata.org/emissions-by-sector]. *Our World in Data*.

Schmid, P. J. (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, *656*, 5–28. https://doi.org/10.1017/S0022112010001217

SINTEF. (2021). APPLIED RESEARCH,TECHNOLOGY AND INNOVATION. Retrieved February 11, 2021, from https://www.sintef.no/en/sintef-group/this-is-sintef/

Wagner, R., Cañadillas, B., Clifton, A., Feeney, S., Nygaard, N., Poodt, M., Martin, C. S., Tüxen, E., & Wagenaar, J. W. (2014). Rotor equivalent wind speed for power curve measurement – comparative exercise for IEA wind annex 32. *Journal of Physics: Conference Series*, *524*, 012108. https://doi.org/10.1088/1742-6596/524/1/012108

Weiss, J., & Tsuchida, T. B. (2017). Chapter 20 - integration into national grids. In T. M. Letcher (Ed.), *Wind energy engineering* (pp. 419–436). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-809451-8.00020-5

A Betz limit derived

To explain Betz limit, the following assumptions are necessary:

- steady state fluid flow, which is homogenous and incompressible
- no frictional drag
- an actuator disc (the rotor) consisting of an infinite number of blades.
- uniform thrust over the rotor area
- a non-rotating wake
- static pressure far up- and downstream of the rotor, which is also equal to the undisturbed ambient static pressure

Given conservation of linear momentum, one can find the net force on the contents of the control volume. This force is of equal size and opposite to the thrust, T. This thrust is the force applied to the wind turbine by the flow. Given the aforementioned assumptions, the thrust is equal and opposite to the change of momentum of the air stream, as shown in equation A.1.

$$T = U_1(\rho AU)_1 - U_4(\rho AU)_4 \tag{A.1}$$

ρ is the density of the air, A is the cross-sectional area, U is the flow's velocity.
Since steady state flow is assumed, $(\rho AU)_1 = (\rho AU)_4 = \dot{m}$. Thus equation A.1 is can be simplified to equation A.2.

$$T = \dot{m}(U_1 - U_4) \tag{A.2}$$

Since the thrust is positive, the velocity behind the rotor, U_4 must be less than the free stream velocity, U_1. As no work is done on either side of the rotor, the Bernoulli function can be applied on two control volumes on either side of the rotor. Equation for the stream tube upstream of the rotor is shown in equation A.3, and the corresponding equation A.4 for the downstream section.

$$p_1 + \frac{1}{2}\rho U_1^2 = p_2 + \frac{1}{2}\rho U_2^2 \tag{A.3}$$

$$p_3 + \frac{1}{2}\rho U_3^2 = p_4 + \frac{1}{2}\rho U_4^2 \tag{A.4}$$

It is also assumed that the pressures far up- and downstream are equal ($p_1 = p_4$), and the velocity across the rotor is constant ($U_2 = U_3$). The thrust on the rotor can also be expressed by the forces on each side of the actuator disc, as can be seen in equation A.5.

$$T = A_2(p_2 - p_3) \tag{A.5}$$

By solving for $(p_2 - p_3)$ using the equations A.3 and A.4, equation A.6 is obtained.

$$T = \frac{1}{2}\rho A_2(U_1^2 - U_4^2) \tag{A.6}$$

Equating the thrust values from equations A.2 and A.6, while also recognizing that the mass flow rate could also be defined by $\rho A_2 U_2$, equation A.7 is obtained.

$$U_2 = \frac{U_1 + U_4}{2} \tag{A.7}$$

Given this simplistic model, the wind velocity at the rotor plane is the average of the undisturbed upstream wind speed and the downstream wind speed. Furthermore, the axial induction factor a is introduced, and defined as the fractional decrease in wind velocity between the undisturbed stream and at the rotor plane. This gives us the equations A.8, A.9 and A.10.

$$a = \frac{U_1 - U_2}{U_1} \tag{A.8}$$

$$U_2 = U_1(1 - a) \tag{A.9}$$

$$U_4 = U_1(1 - 2a) \tag{A.10}$$

As the axial induction factor increases from 0, the wind velocity behind the rotor is slowed down more and more. If a = 0.5, the wind has slowed to a stand still behind the rotor, and the simple theory is no longer applicable. The power obtained, P, is equal to the thrust multiplied by the velocity at the disc, as shown in equation A.11. Thrust is obtained from equation A.6.

$$P = \frac{1}{2}\rho A_2(U_1^2 - U_4^2)U_2 = \frac{1}{2}\rho A_2 U_2(U_1 + U_4)(U_1 - U_4). \tag{A.11}$$

Values for U_2 and U_4 are substituted according to equations A.9 and A.10, and equation A.12 is obtained.

$$P = \frac{1}{2}\rho A U^3 4a(1-a)^2 \tag{A.12}$$

.

The control volume area at the rotor A_2 and the free stream velocity U_1 are replaced by the rotor area A and U respectively. The performance of the wind turbine rotor is usually characterised by it's power coefficient, C_P. The power coefficient is equal to the fraction of the power in the wind that is obtained by the rotor, as showed in equation A.13

$$C_P = \frac{P}{\frac{1}{2}\rho U^3 A} = \frac{Rotor\ Power}{Power\ in\ the\ wind} \tag{A.13}$$

Combining equation A.13 and equation A.12, equation A.14 is obtained.

$$C_P = 4a(1-a)^2 \tag{A.14}$$

The highest possible value for C_P is determined by taking the derivative of the power coefficient from equation A.14 with respect to a and setting it equal to zero. This yields the value for $a = 1/3$. Thus the best possible value for C_P is defined, according to equation A.15.

$$C_{P,max} = 4 \cdot \frac{1}{3}\left(1 - \frac{1}{3}\right)^2 = 0.5926 \tag{A.15}$$

$C_{P,max}$ is the theoretical maximum value for the rotor power coefficient. However, certain effects makes this value unobtainable for real turbines. Three of these effects are at the core of the issue:

- the rotation of the wake behind the rotor
- the finite number of blades on the rotor, and the associated tip losses
- non-zero aerodynamic drag

This concludes the appendix concerning the rotor's efficiency. Note that the overall efficiency of the wind turbine is a function of both the rotor power coefficient and the mechanical efficiency of the turbine. (Manwell et al., 2009)

B OpenFOAM turbine properties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/

FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      turbineProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

NumBl                        3;
TipRad                      63.0;
HubRad                       1.5;
UndSling                     0.0;
OverHang                    -5.01910;
NacelleLength                8;
NacelleFrontalArea           2;
NacelleCd                    0.5;
TowerHt                     300.0;
Twr2Shft                     1.96256;
ShftTilt                    -5.0;
PreCone                    (-2.5 -2.5 -2.5);
GBRatio                     97.0;
GBEfficiency                 1.0;
GenEfficiency                0.944;
RatedRotSpeed               12.1;
GenIner                    534.116;
HubIner                    115.926E3;
BladeIner                   11.776047E6;
//GenTorqueControllerType    "fiveRegion";
//GenTorqueControllerType  "speedTorqueTable";
GenTorqueControllerType  "none";
//BladePitchControllerType   "PID";
BladePitchControllerType "none";
NacYawControllerType       "none";
RotSpeedLimiter             false;
GenTorqueRateLimiter        true;
NacYawRateLimiter           true;
BladePitchRateLimiter       true;
SpeedFilterCornerFrequency  2.0;


GenTorqueControllerParams
{
    RateLimitGenTorque       15.0E3;
    SpeedTorqueTable
    (
        //      gen speed (RPM) gen torque (N-m)
        (       670.00              0.0 )
        (       871.00          20000.0 )
        (       1161.96         32000.0 )
        (       1173.70         43093.6 )
    );
    CutInGenSpeed           670.0;
    Region2StartGenSpeed    871.0;
    Region2EndGenSpeed     1161.963;
    CutInGenTorque            0.0;
    RatedGenTorque           43.09355E3;
    KGen                      2.55764E-2;
}

BladePitchControllerParams
{
    RateLimitBladePitch       8.000;
    PitchMin                  0.000;
    PitchMax                 90.000;
    PitchK                    6.302336;
    PitchControlKP            1.82620057;
    PitchControlKI            0.78265750;
    PitchControlKD            0.000;
}

NacYawControllerParams
{
    RateLimitNacYaw           2.0;
}




Airfoils
(
    "Cylinder1"
    "Cylinder2"
    "DU40_A17"
    "DU35_A17"
    "DU30_A17"
    "DU25_A17"
    "DU21_A17"
    "NACA64_A17"
);



BladeData
(
//  radius(m)   c(m)     twist(deg) airfoil
```

```
    (2.8667     3.542    13.308     0)
    (5.6        3.854    13.308     0)
    (8.3333     4.167    13.308     1)
    (11.75      4.557    13.308     2)
    (15.85      4.652    11.48      3)
    (19.95      4.458    10.162     3)
    (24.05      4.249    9.011      4)
    (28.15      4.007    7.795      5)
    (32.25      3.748    6.544      5)
    (36.35      3.502    5.361      6)
    (40.45      3.256    4.188      6)
    (44.55      3.01     3.125      7)
    (48.65      2.764    2.319      7)
    (52.75      2.518    1.526      7)
    (56.1667    2.313    0.863      7)
    (58.9       2.086    0.37       7)
    (61.6333    1.419    0.106      7)
);
```

C   Case 1 - extended results

C.1   Systems 1, 2, 3, 4 - velocity in x, y and z direction

System 1

Results divided into wind directions (x, y and z-direction) were not shown in the results section, where only the norm value was included. This appendix gives an overview of the wind flow in each direction as reconstructed by all DMDs in Case 1, as well as the deviation in each system. The flows are visualised in the YZ, XZ and XY-planes one after another. In figure 3.4, one can observe System 1's extended results in the YZ-plane.

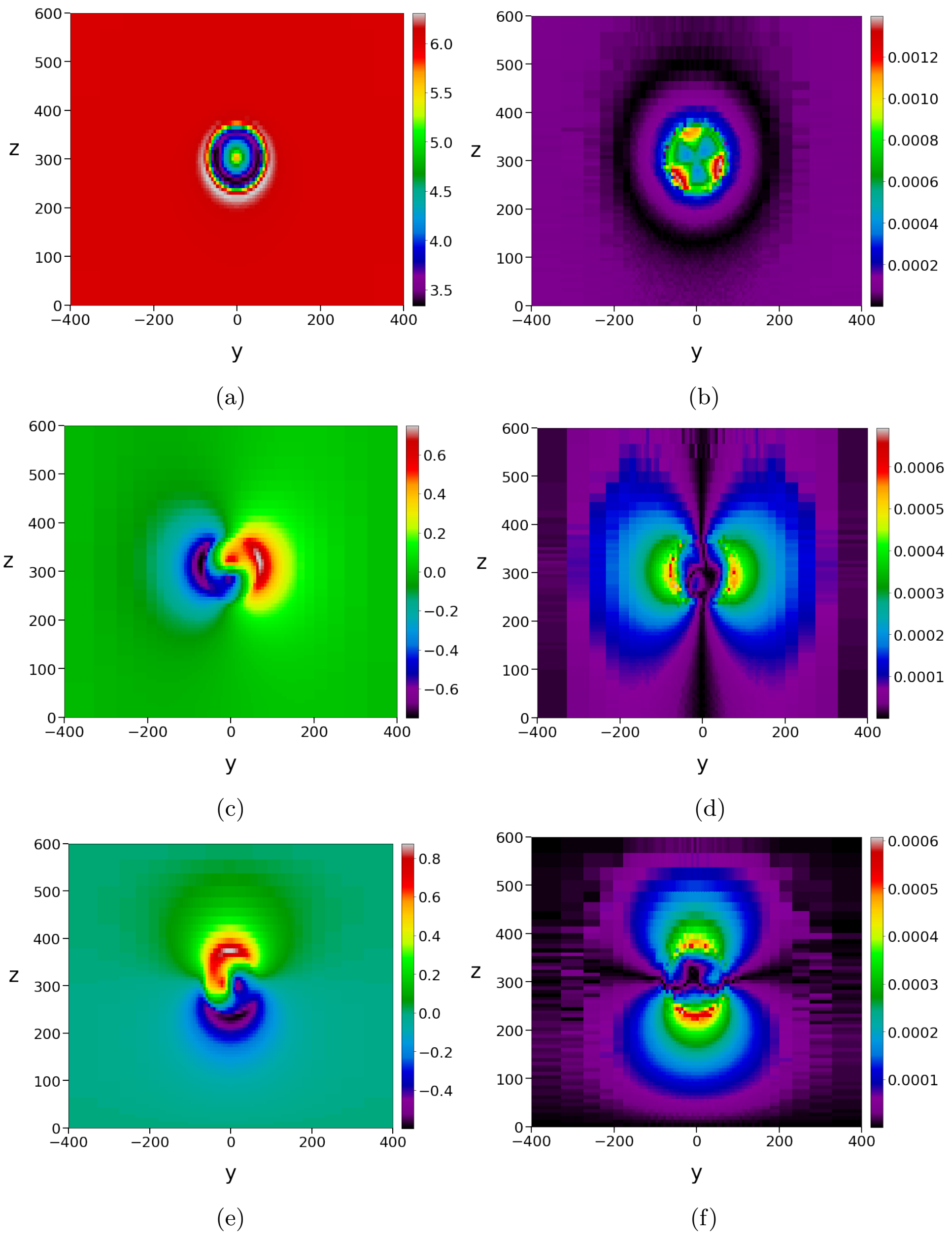
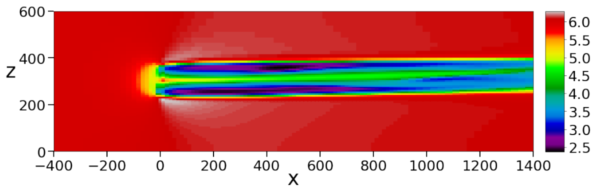
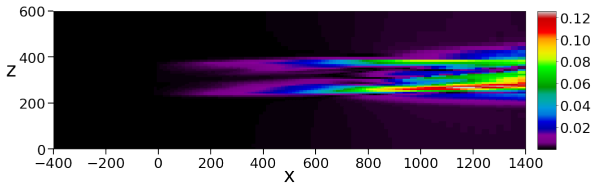


Figure C.1: (a) 0-component in dmd1, (b) 0-component deviance in System 1, (c) 1-component in dmd1, (d) 1-component deviance in System 1, (e) 2-component in dmd1, (f) 2-component deviance in System 2. Visualised in the YZ-plane.

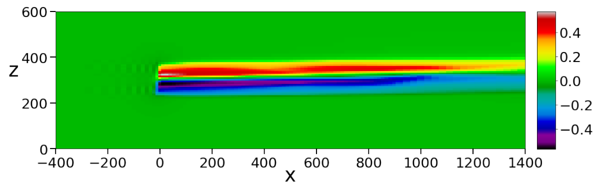
Figure 3.3 visualises the same extended results, now plotted in the XZ-plane.



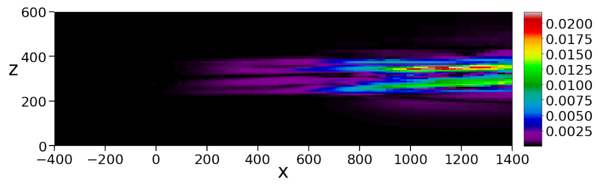
(a)



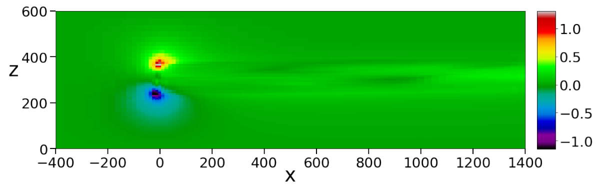
(b)



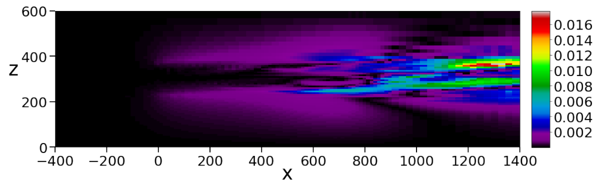
(c)



(d)



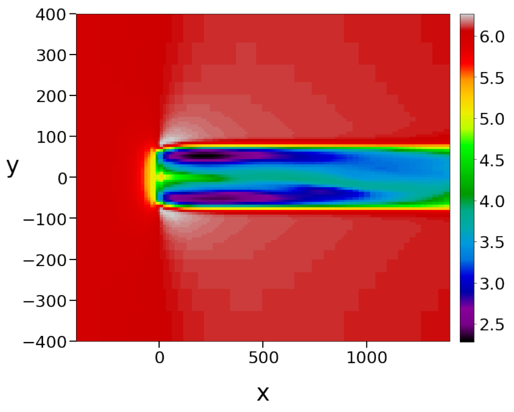
(e)



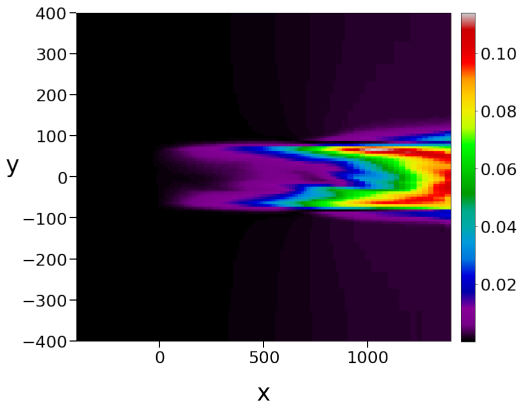
(f)

Figure C.2: (a) 0-component in dmd1, (b) 0-component deviance in System 1, (c) 1-component in dmd1, (d) 1-component deviance in System 1, (e) 2-component in dmd1, (f) 2-component deviance in System 1. Visualised in the XZ-plane.

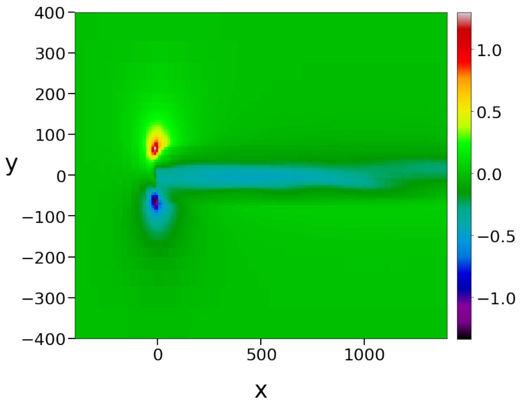
Moreover, figure 3.2 shows System 1's extended results in the XY-plane.



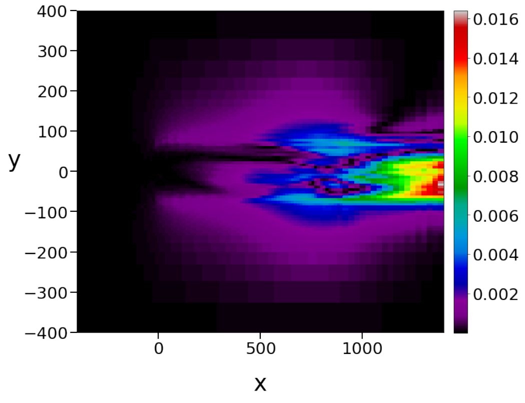
(a)



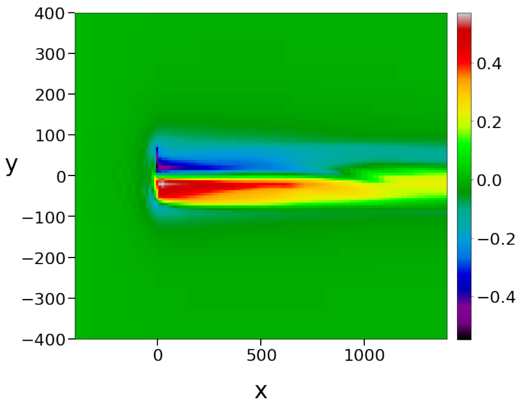
(b)



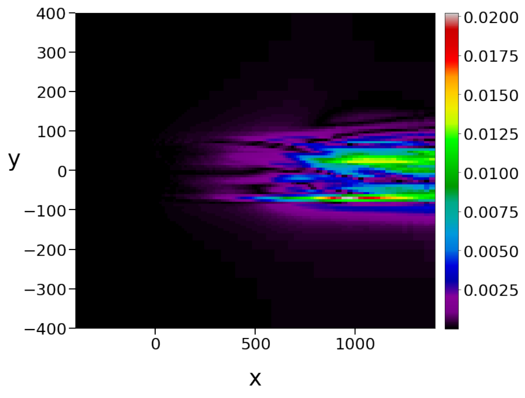
(c)



(d)



(e)

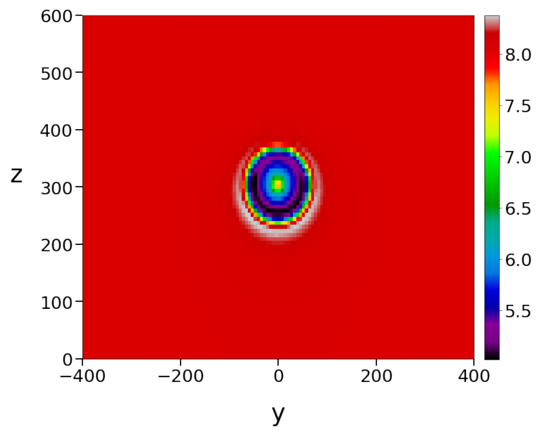


(f)

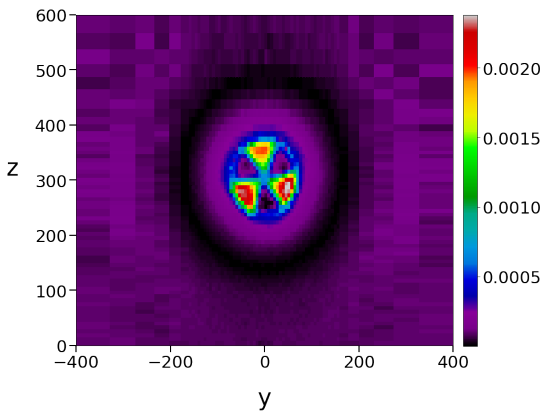
Figure C.3: (a) 0-component in dmd1, (b) 0-component deviance in System 1, (c) 1-component in dmd1, (d) 1-component deviance in System 1, (e) 2-component in dmd1, (f) 2-component deviance in System 1. Visualised in the XY-plane.

System 2

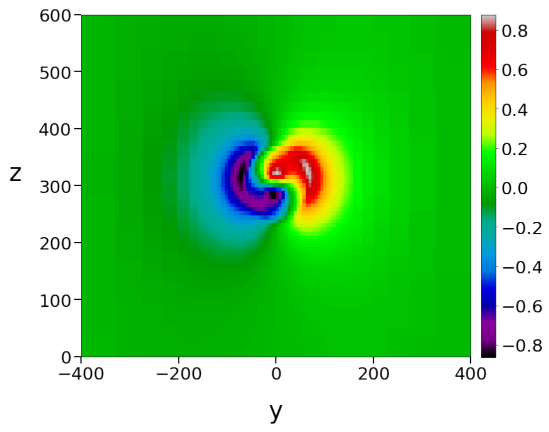
Extended results for System 2 are provided in this section. In figure C.4, one can observe the 0, 1 and 2 component plotted in the YZ-plane.



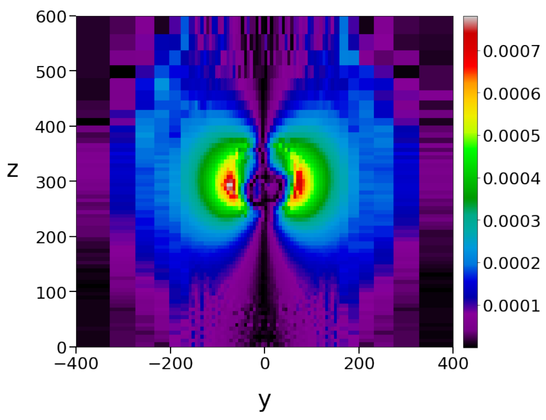
(a)



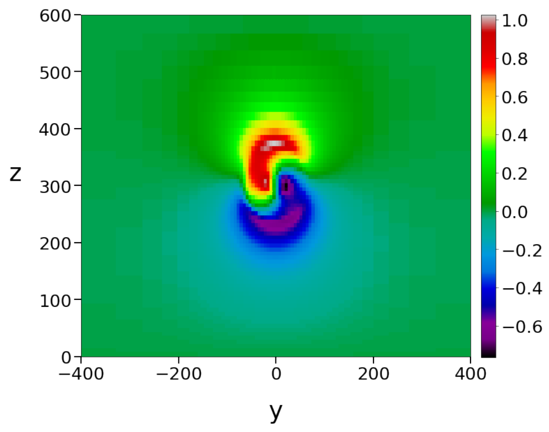
(b)



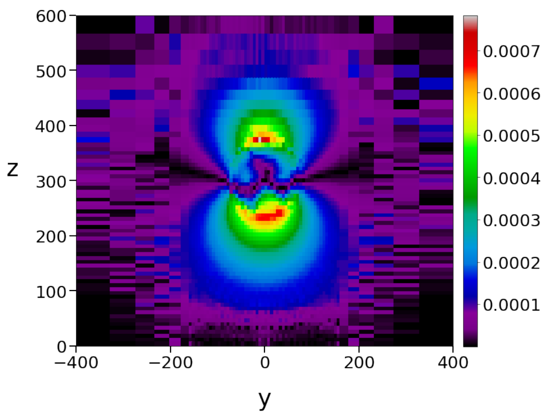
(c)



(d)



(e)



(f)

Figure C.4: (a) 0-component in dmd2, (b) 0-component deviance in System 2, (c) 1-component in dmd2, (d) 2-component deviance in System 2, (e) 2-component in dmd2, (f) 2-component deviance in System 2. Visualised in the YZ-plane.

The same results, now plotted in the XZ-plane, are shown in figure C.5.

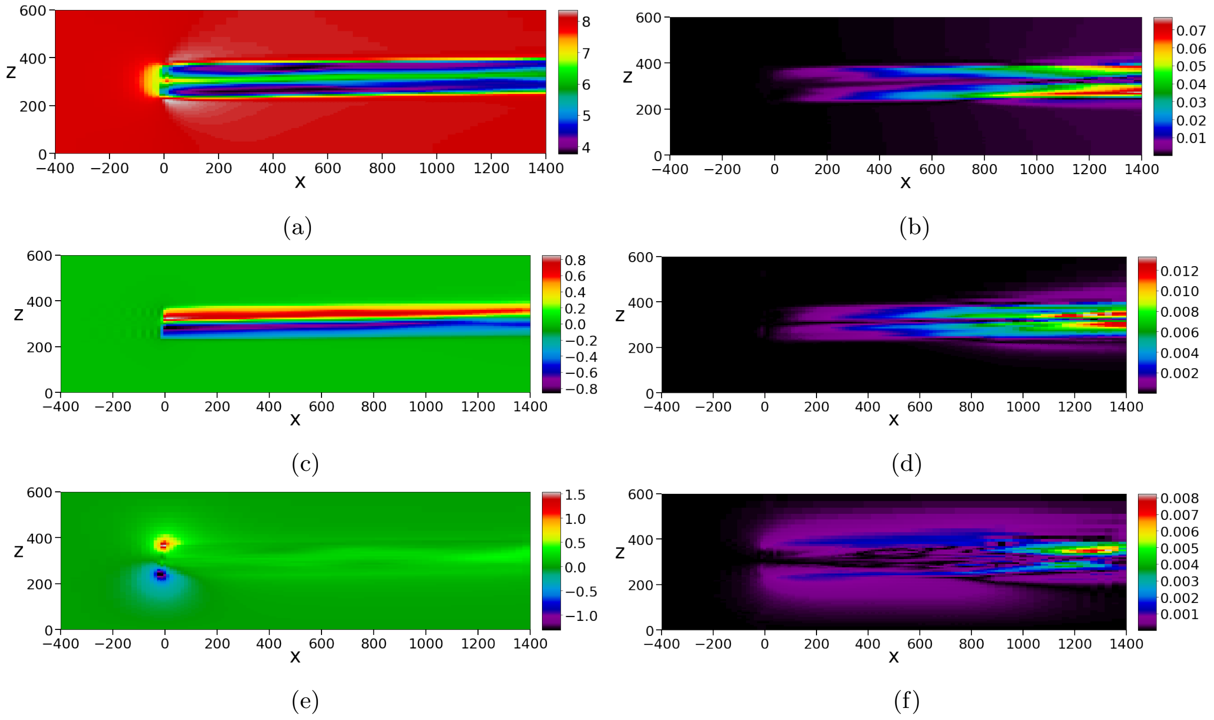


Figure C.5: (a) 0-component in dmd2, (b) 0-component deviance in System 2, (c) 1-component in dmd2, (d) 1-component deviance in System 2, (e) 2-component in dmd2, (f) 2-component deviance in System 2. Visualised in the XZ-plane.

Furthermore, figure C.6 shows System 2's extended results in the XY-plane.

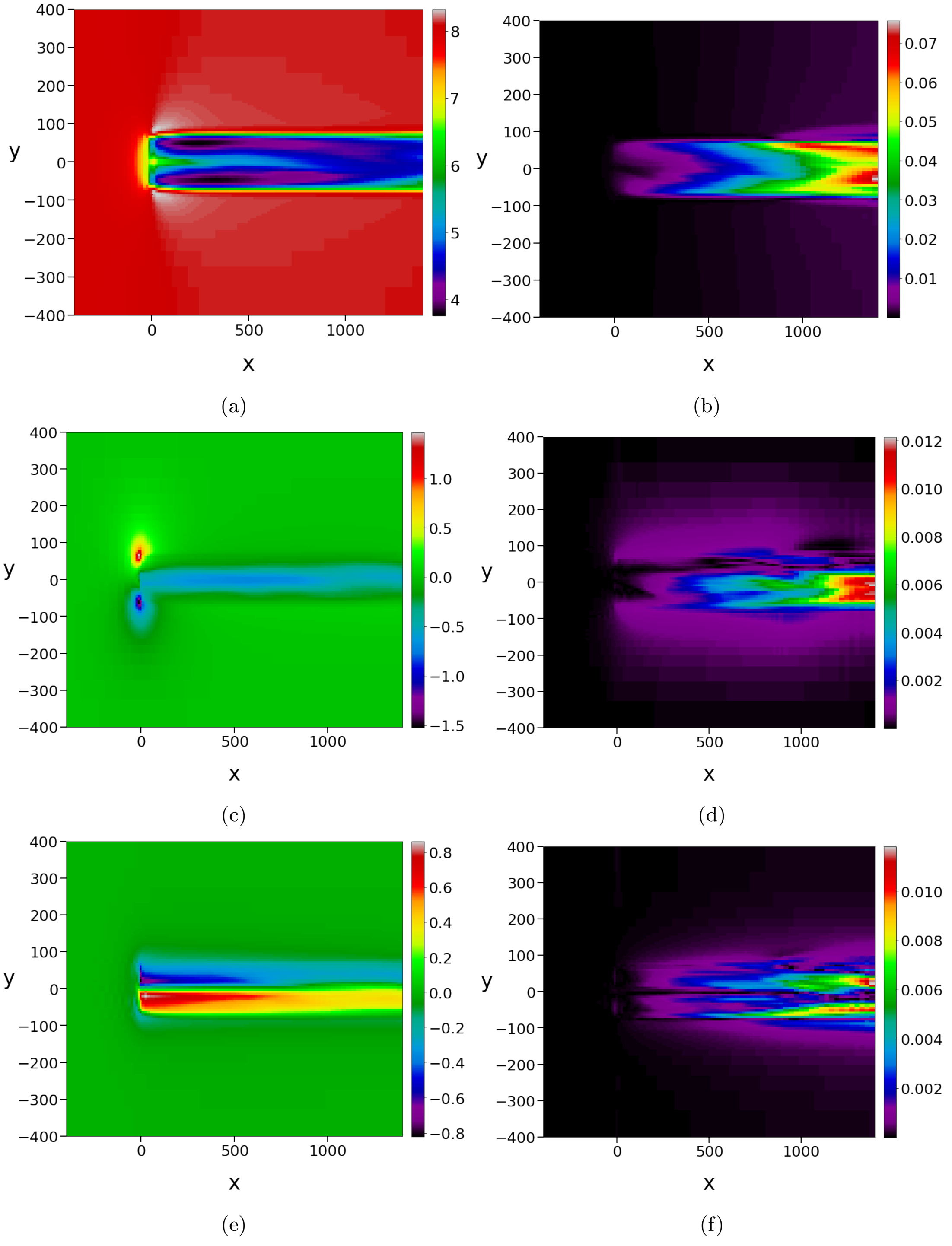
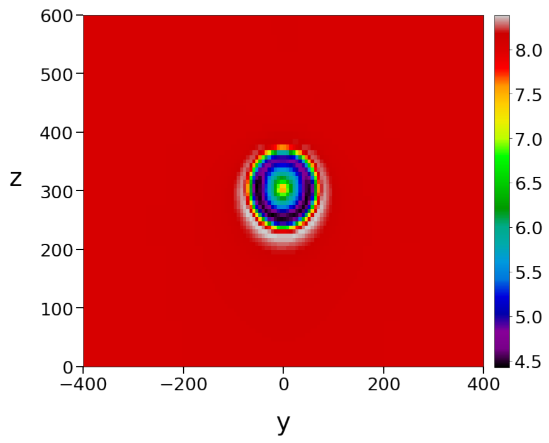


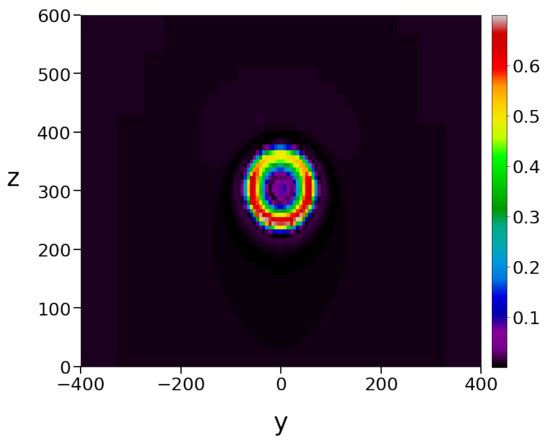
Figure C.6: (a) 0-component in dmd2, (b) 0-component deviance in System 2, (c) 1-component in dmd2, (d) 1-component deviance in System 2, (e) 2-component in dmd2, (f) 2-component deviance in System 2. Visualised in the XY-plane.

System 3

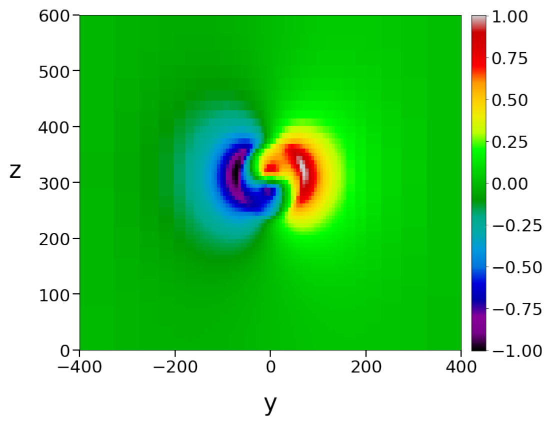
Extended results for System 3 are provided in this section. In figure C.7, one can observe the 0, 1 and 2 component plotted in the YZ-plane.



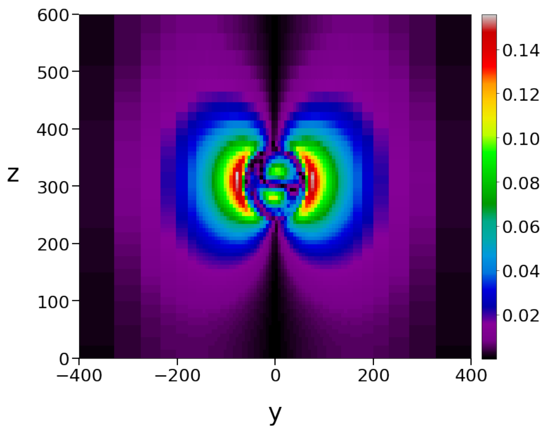
(a)



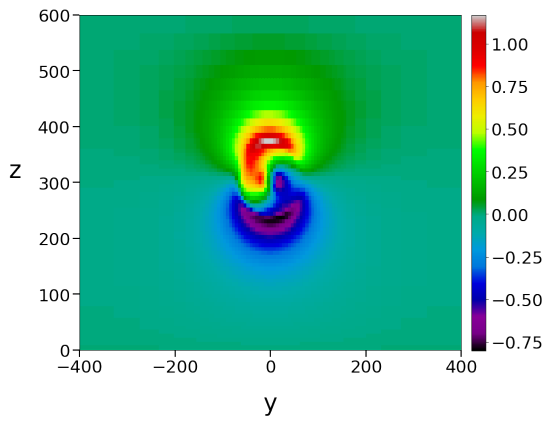
(b)



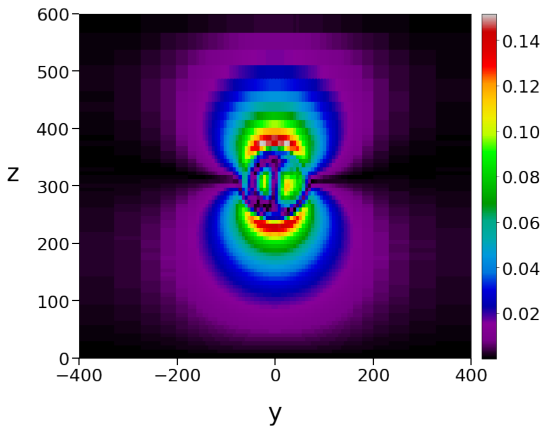
(c)



(d)



(e)



(f)

Figure C.7: (a) 0-component in dmd3, (b) 0-component deviance in System 3, (c) 1-component in dmd3, (d) 2-component deviance in System 3, (e) 2-component in dmd3, (f) 2-component deviance in System 3. Visualised in the YZ-plane.

The same results, now plotted in the XZ-plane, are shown in figure C.8.

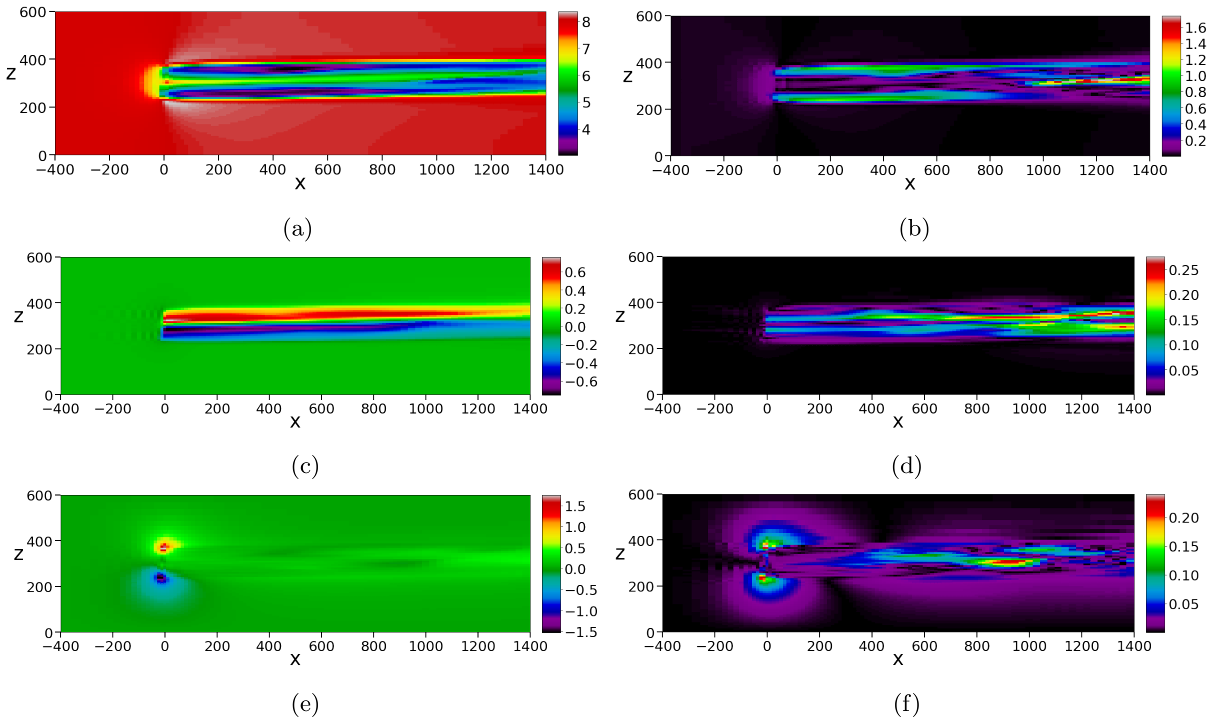
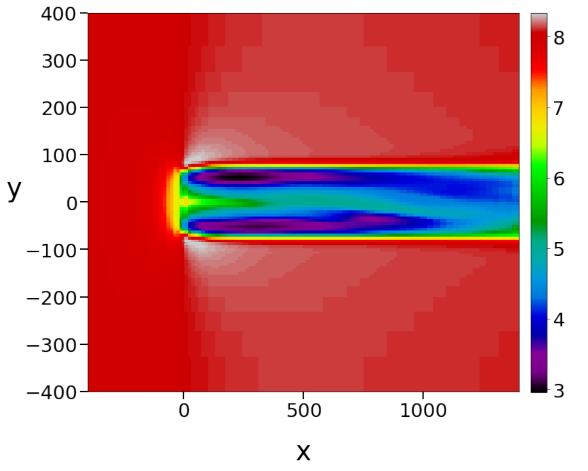
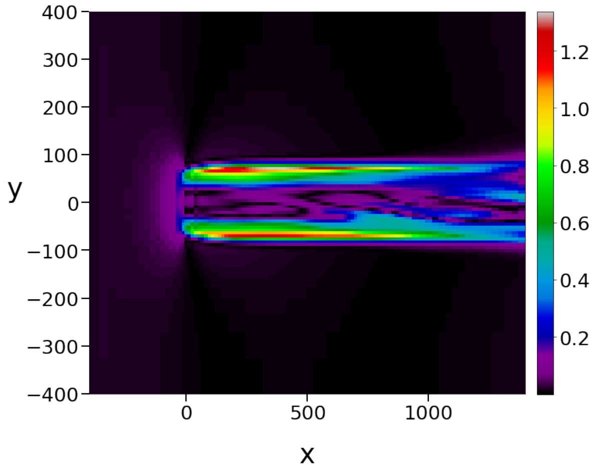
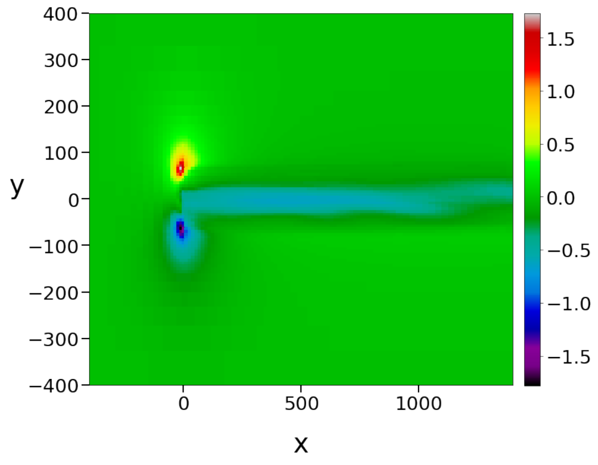


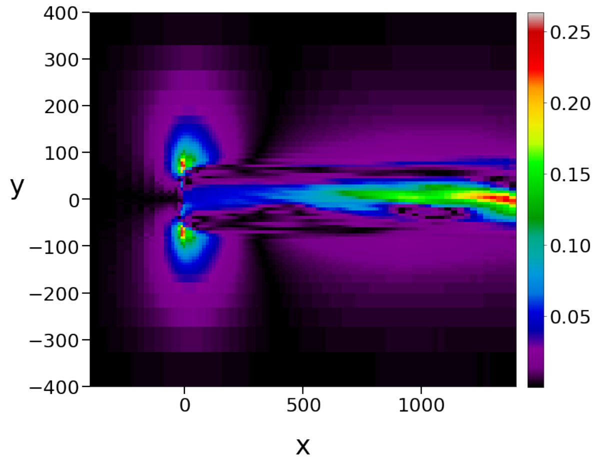
Figure C.8: (a) 0-component in dmd3, (b) 0-component deviance in System 3, (c) 1-component in dmd3, (d) 1-component deviance in System 3, (e) 2-component in dmd3, (f) 2-component deviance in System 3. Visualised in the XZ-plane.

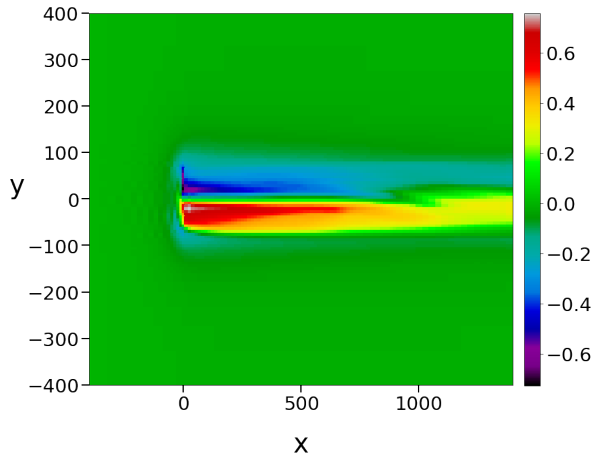
Furthermore, figure C.9 shows System 3's extended results in the XY-plane.


(a)


(b)


(c)


(d)


(e)

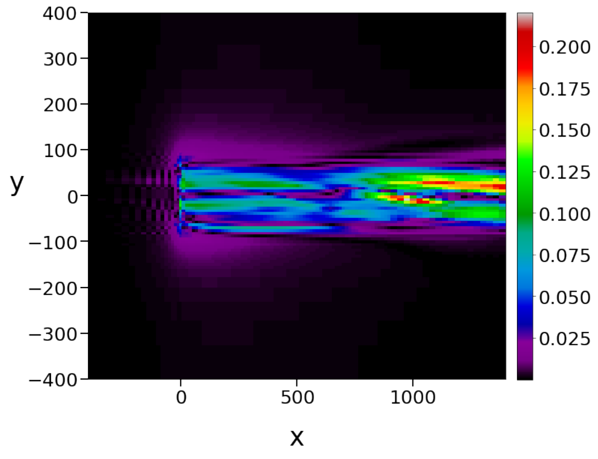

(f)

Figure C.9: (a) 0-component in dmd3, (b) 0-component deviance in System 3, (c) 1-component in dmd3, (d) 1-component deviance in System 3, (e) 2-component in dmd3, (f) 2-component deviance in System 3. Visualised in the XY-plane.

System 4

Extended results for System 4 are provided in this section. In figure C.10, one can observe the 0, 1 and 2 component plotted in the YZ-plane.

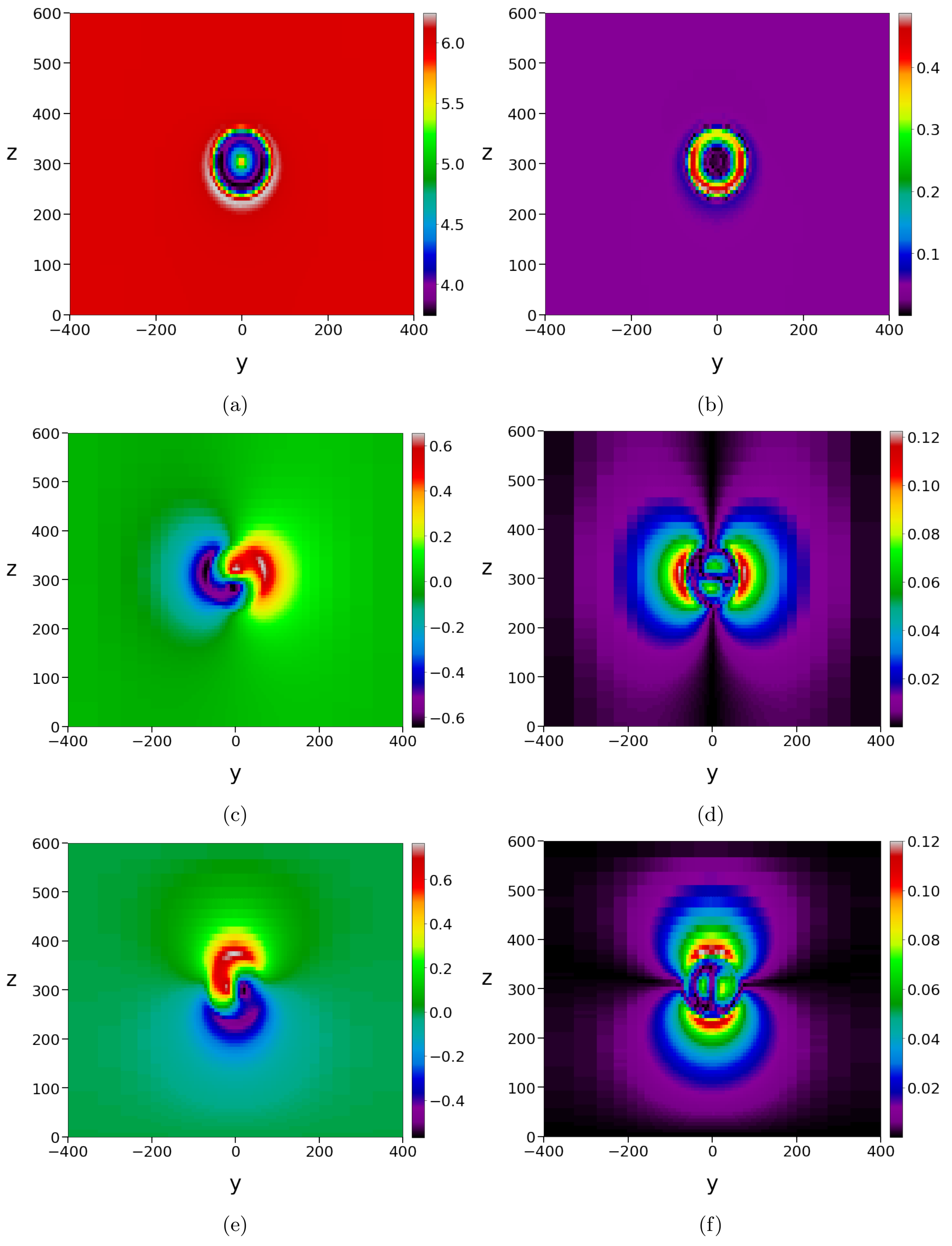


Figure C.10: (a) 0-component in dmd4, (b) 0-component deviance in System 4, (c) 1-component in dmd4, (d) 2-component deviance in System 4, (e) 2-component in dmd4, (f) 2-component deviance in System 4. Visualised in the YZ-plane.

The same results, now plotted in the XZ-plane, are shown in figure C.11.

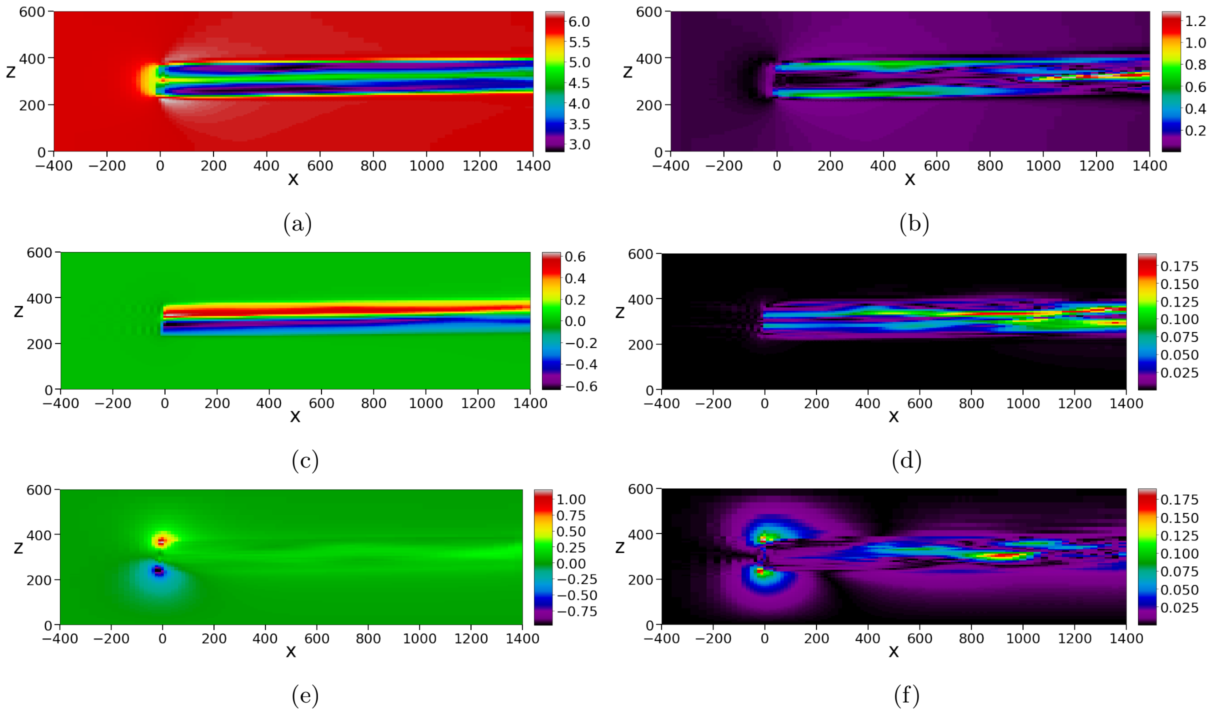
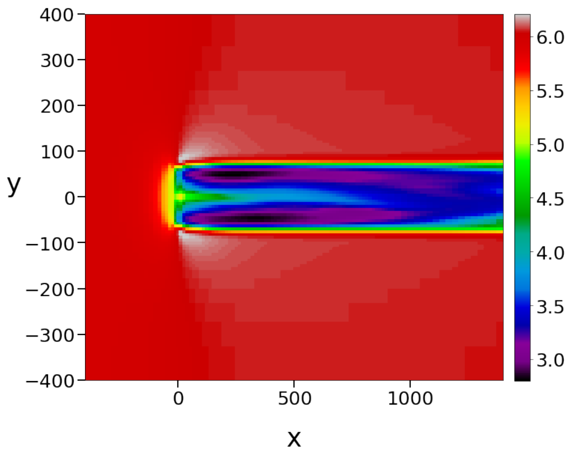
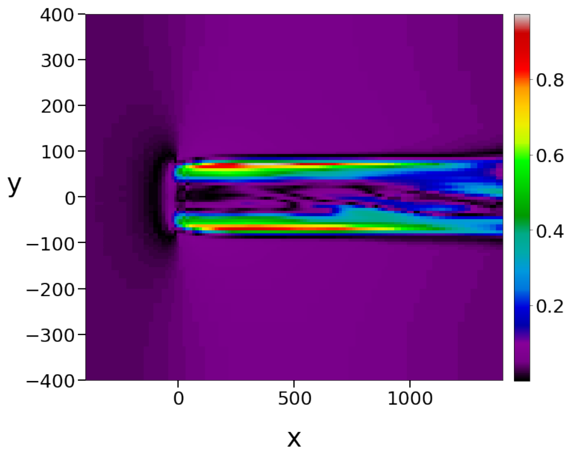


Figure C.11: (a) 0-component in dmd4, (b) 0-component deviance in System 4, (c) 1-component in dmd4, (d) 1-component deviance in System 4, (e) 2-component in dmd4, (f) 2-component deviance in System 4. Visualised in the XZ-plane.

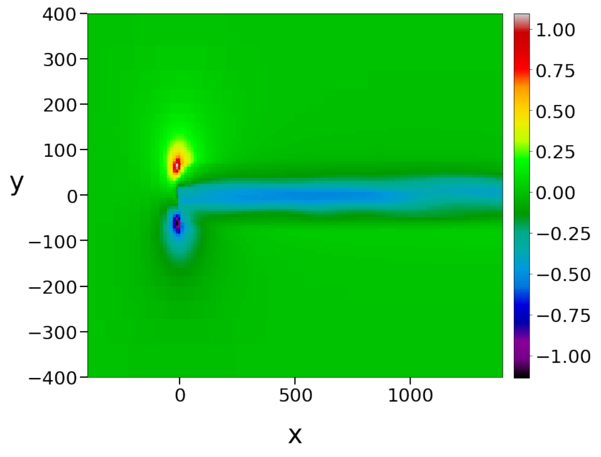
Furthermore, figure C.12 shows System 4's extended results in the XY-plane.



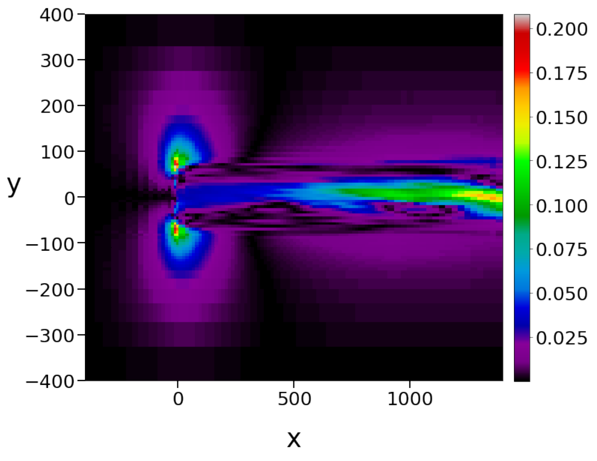
(a)



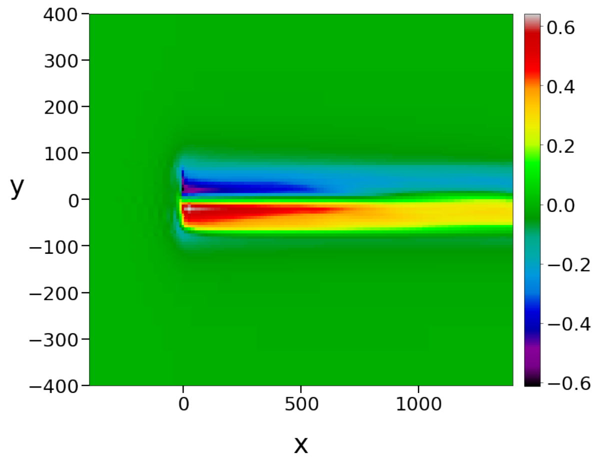
(b)



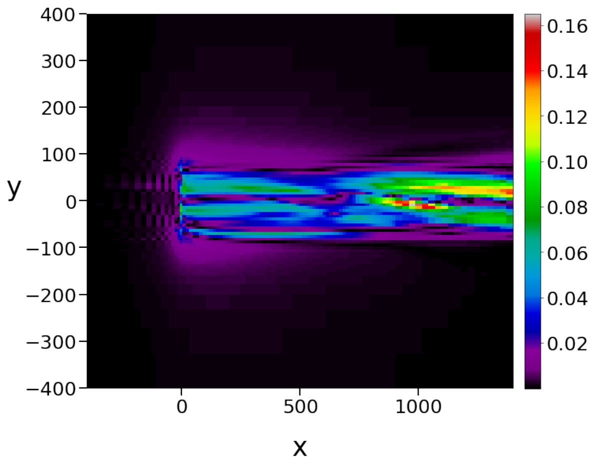
(c)



(d)



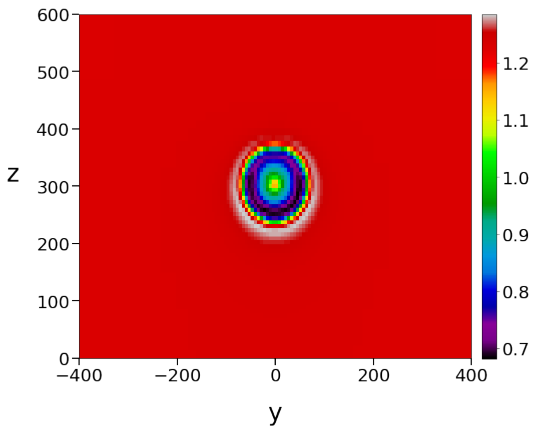
(e)



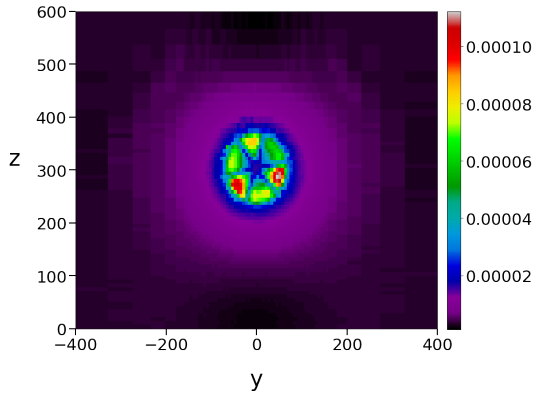
(f)

Figure C.12: (a) 0-component in dmd4, (b) 0-component deviance in System 4, (c) 1-component in dmd4, (d) 1-component deviance in System 4, (e) 2-component in dmd4, (f) 2-component deviance in System 4. Visualised in the XY-plane.

C.2 System 3 - modes and dynamics



(a)



(b)

Figure C.13: dmd3 modes for index 0 (a) and 1 (b).

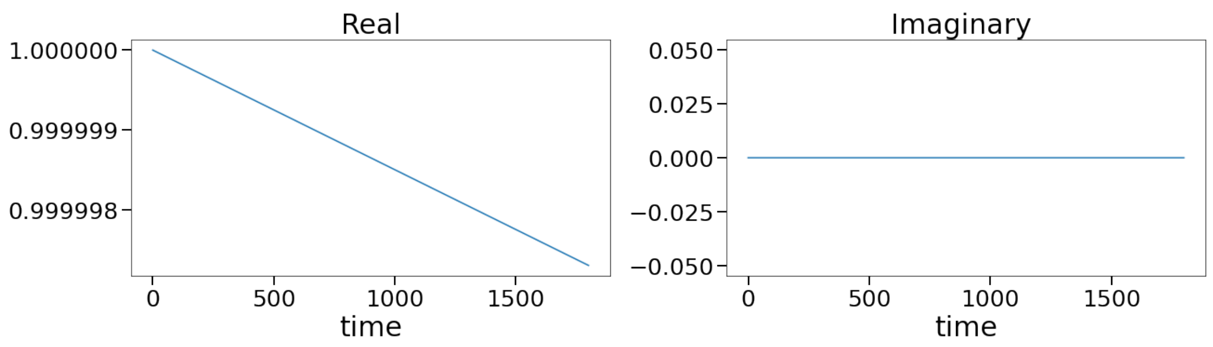


Figure C.14: dmd3 dynamics for index 0.

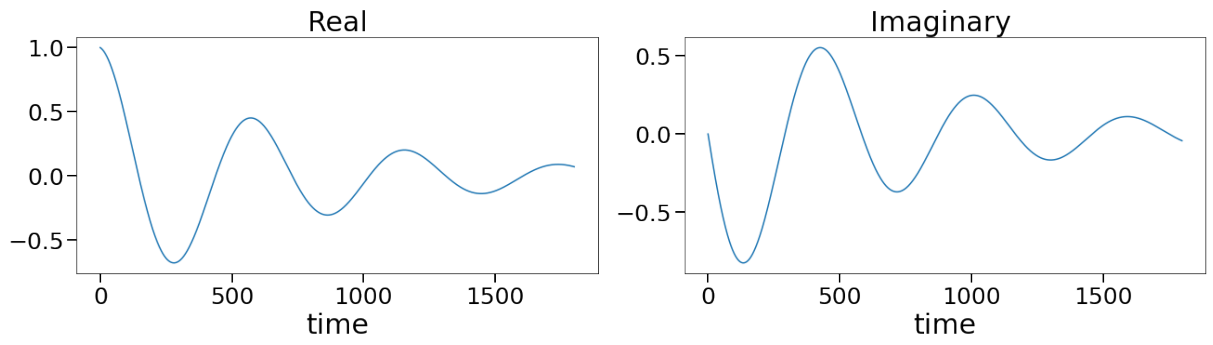
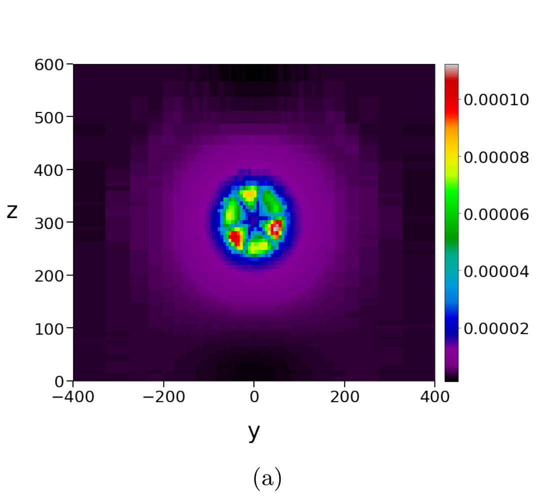


Figure C.15: dmd3 dynamics for index 1.



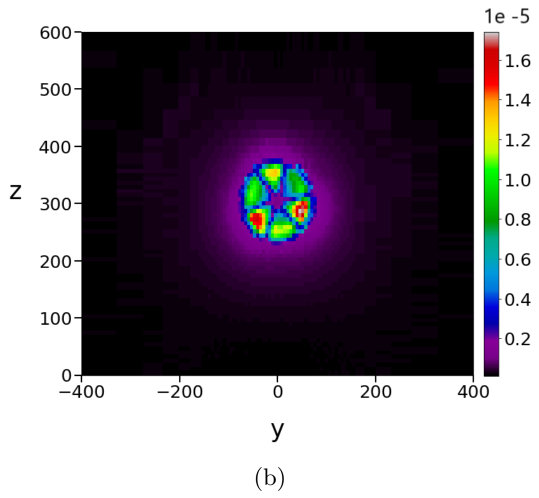


Figure C.16: dmd3 modes for index 2 (a) and 3 (b).

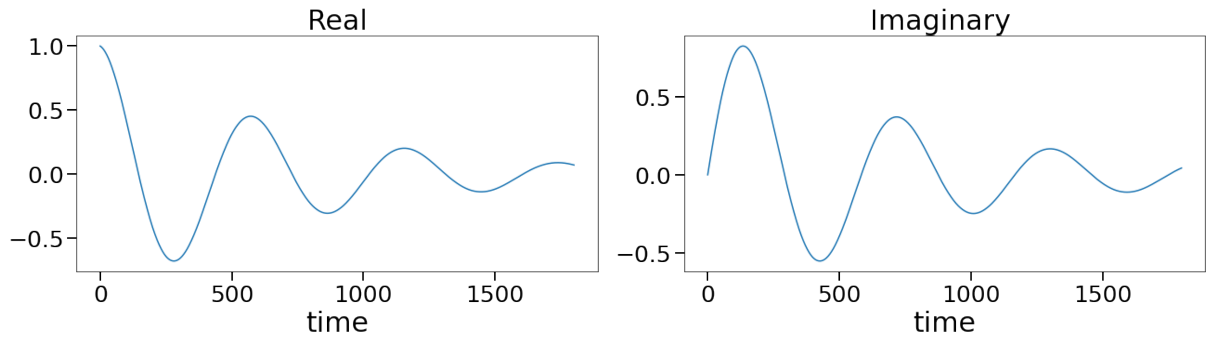


Figure C.17: dmd3 dynamics for index 2.

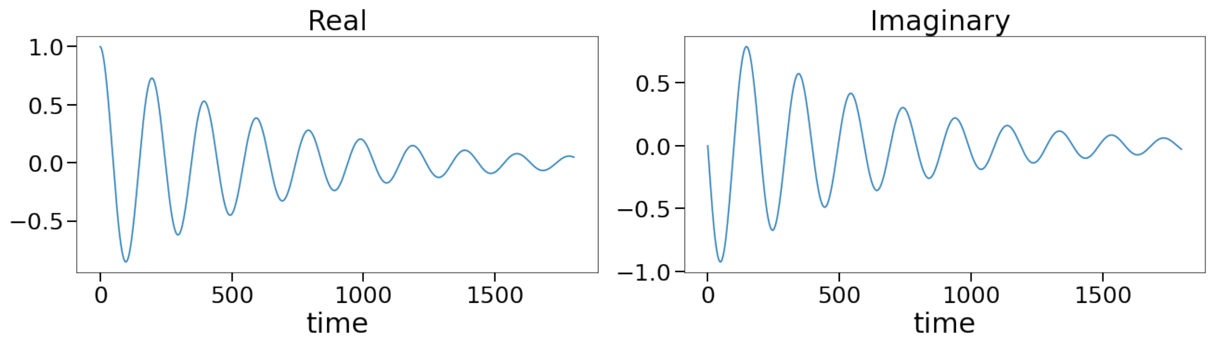
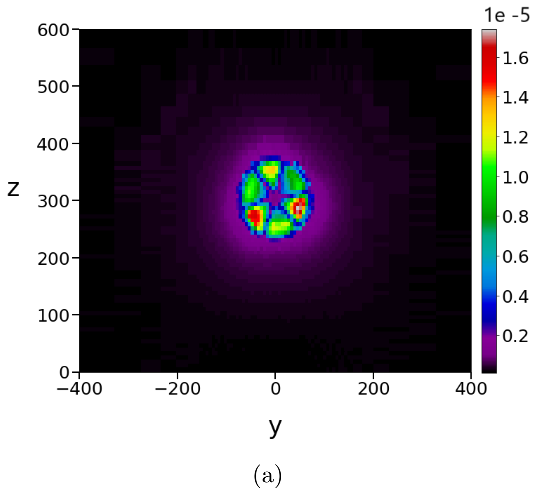


Figure C.18: dmd3 dynamics for index 3.



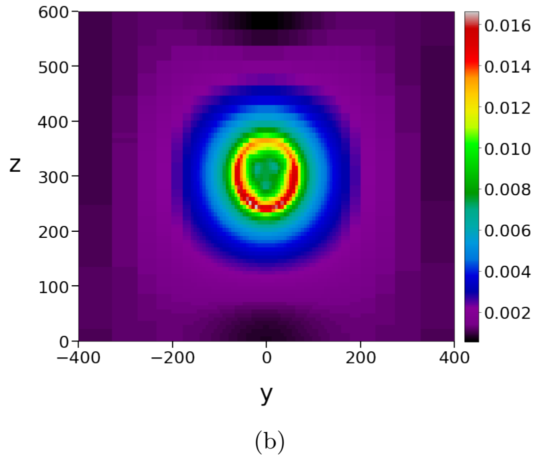


Figure C.19: dmd3 modes for index 4 (a) and 5 (b).

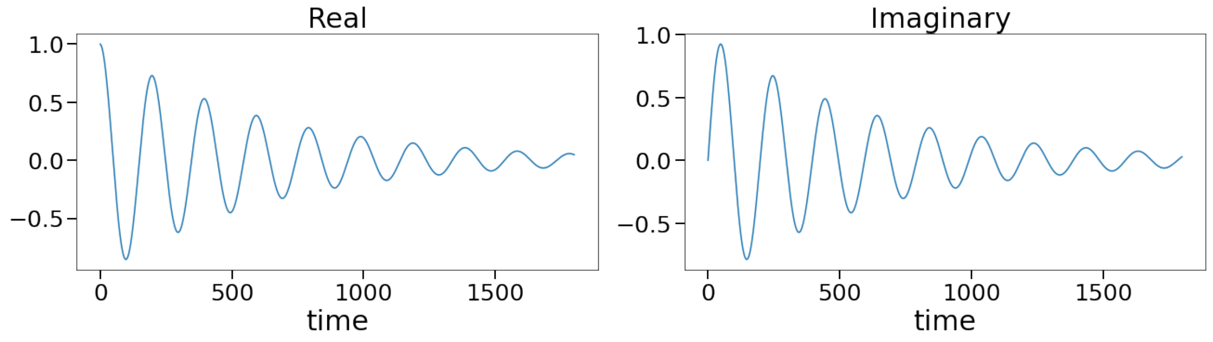


Figure C.20: dmd3 dynamics for index 4.

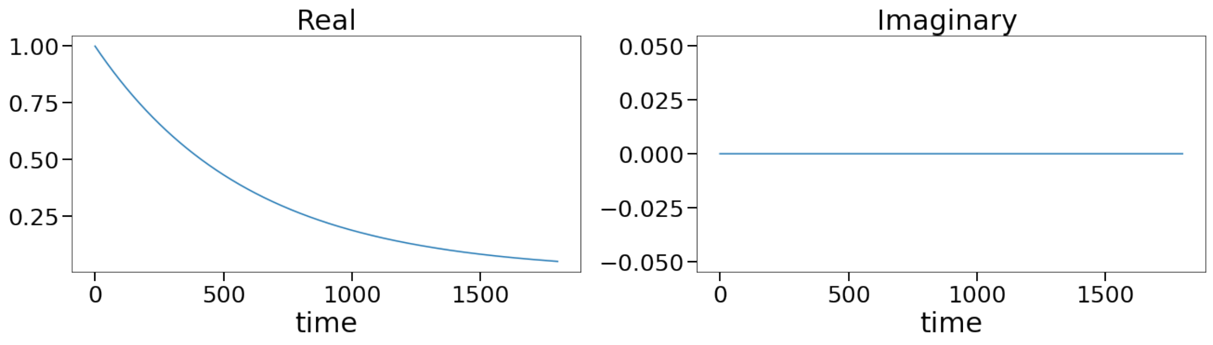
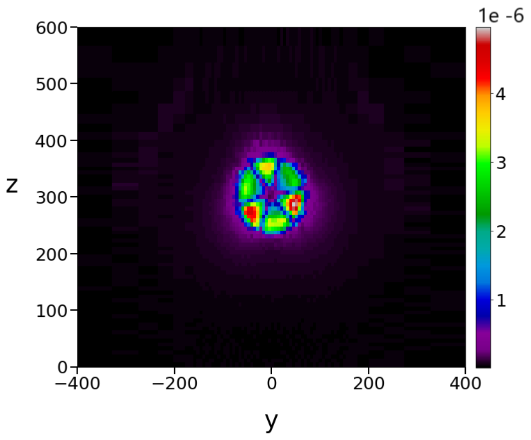
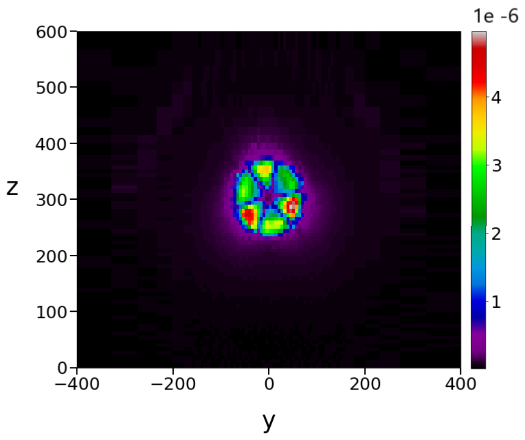


Figure C.21: dmd3 dynamics for index 5.



(a)



(b)

Figure C.22: dmd3 modes for index 6 (a) and 7 (b).

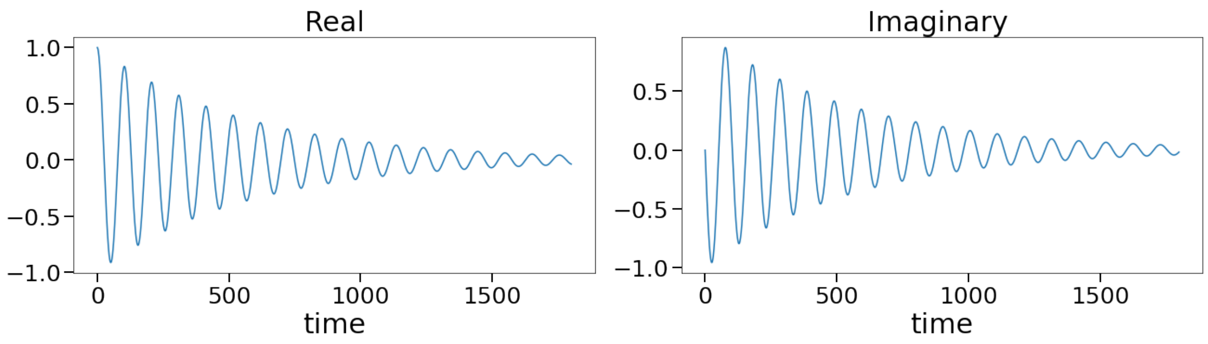


Figure C.23: dmd3 dynamics for index 6.

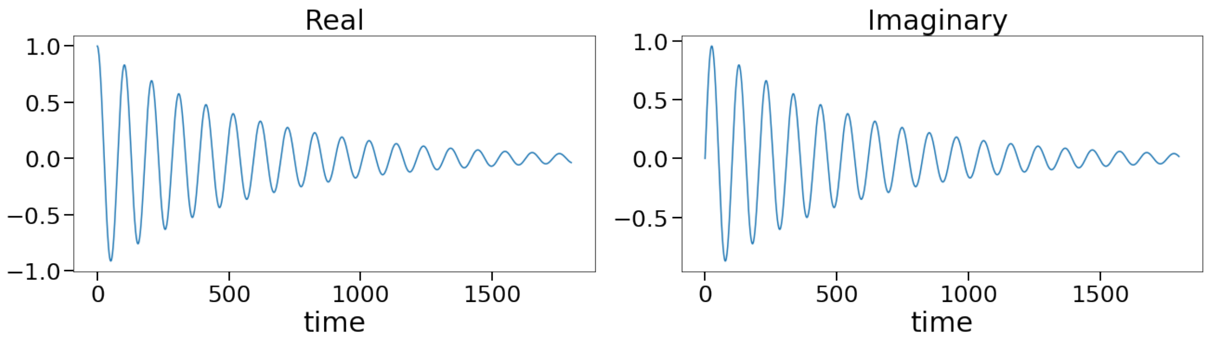
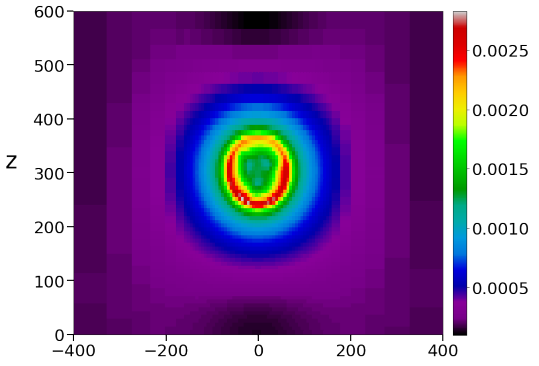
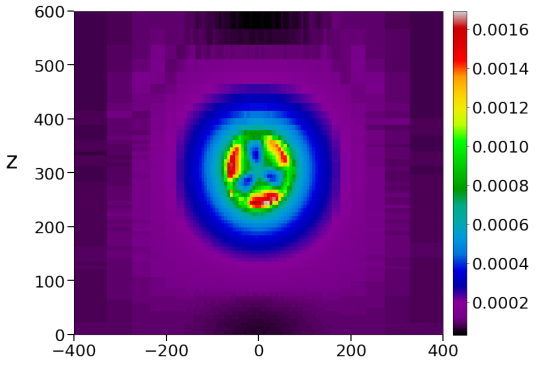


Figure C.24: dmd3 dynamics for index 7.



(a)



(b)

Figure C.25: dmd3 modes for index 8 (a) and 9 (b).

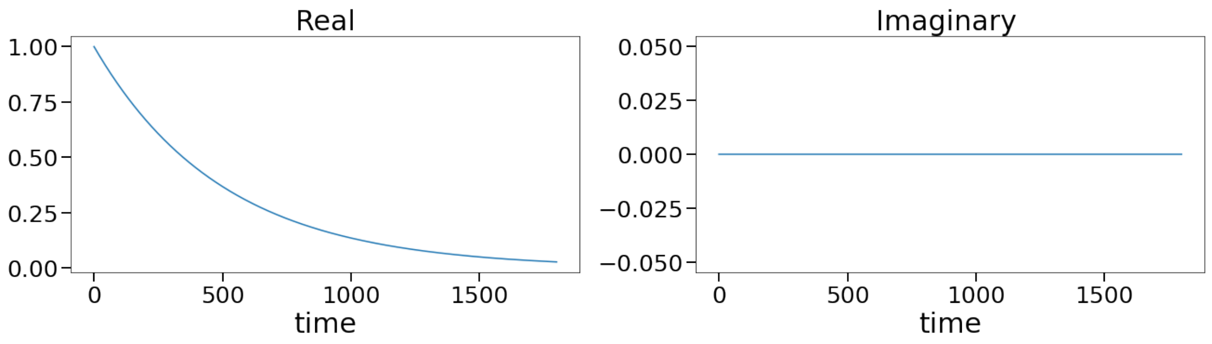


Figure C.26: dmd3 dynamics for index 8.

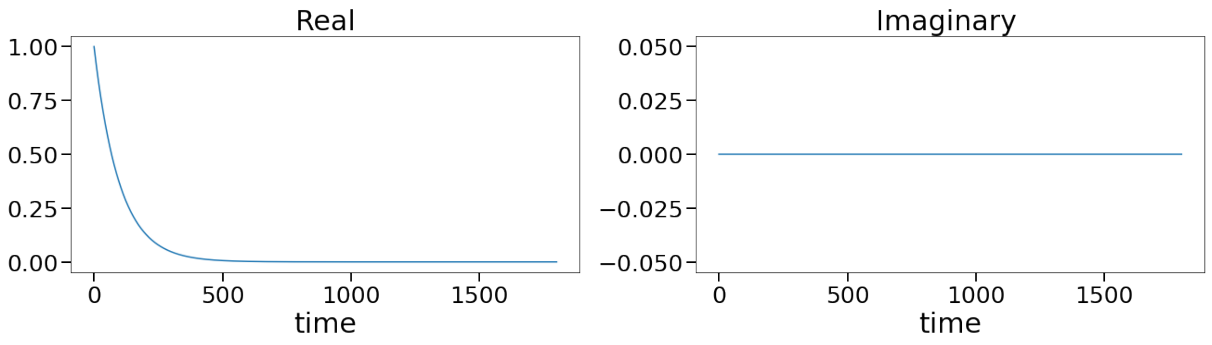


Figure C.27: dmd3 dynamics for index 9.

D CSV error

```
Load snapshot data
Path where data is stored
cf.convert_case_as_csv("/home/ubuntu/hawasser/Cases/Case_3/6MPS/", "/home/
    ubuntu/hawasser/Cases/Case_3/6MPS/" + "_csv",
                              convert_field_values=["U", "UMean"],
                              convert_turbine_values=["rotSpeed", "powerRotor", "
                                  thrust", "torqueGen", "torqueRotor", "azimuth", "
                                  nacYaw", "pitch"],
                              convert_grid=True, skew=True)

isfile
... points
      0              1  2
0  2400           -600  0
1  2400   -574.2737626  0
2  2400   -550.8185735  0
3  2400   -529.4339501  0
4  2400    -509.937108  0
... faces
       0      1      2      3
0      1    162  32523  32362
1    161  32522  32523    162
2  32361  32362  32523  32522
3      2    163  32524  32363
4    162  32523  32524    163
   0
0  0
1  1
2  2
3  3
4  4
... points_skew
      0              1  2
0  2400           -600  0
1  2400   -574.2737626  0
2  2400   -550.8185735  0
3  2400   -529.4339501  0
4  2400    -509.937108  0
... faces_skew
       0      1      2      3
0      1    162  32523  32362
1    161  32522  32523    162
2  32361  32362  32523  32522
3      2    163  32524  32363
4    162  32523  32524    163
   0
0  0
1  1
2  2
3  3
```

```
4  4
... owner_skew
   0
0  0
1  0
2  0
3  1
4  1
... neighbour_skew
        0
0       1
1     160
2   32000
3       2
4     161
sort_cells
project faces
phase 1
phase 2
phase 1
phase 2
phase 1
phase 2
... rotSpeed ... thrust ... powerRotor ... torqueRotor ... azimuth ... torqueGen


... pitch

... nacYaw



            0
0       280.4
1       280.4
2     280.877
3     281.353
4      281.83          0
0   7.942
1   7.942
2   7.942
3   7.942
4   7.942

               0
0   3.00845e+06
1   3.00843e+06
2   1.91635e+06
3   1.89165e+06
4   1.86749e+06
   0
0  0
1  0
2  0
```

```
3   0
4   0
          0
0   417493
1   417491
2   347485
3   345619
4   343753
                0
0    3.6173e+06
1   3.61728e+06
2   2.30418e+06
3   2.27448e+06
4   2.24543e+06
      0
0   270
1   270
2   270
3   270
4   270
```

```
RemoteTraceback                          Traceback (most recent call last)
RemoteTraceback:
"""
Traceback (most recent call last):
  File "/usr/lib/python3.8/multiprocessing/pool.py", line 125, in worker
    result = (True, func(*args, **kwds))
  File "/usr/lib/python3.8/multiprocessing/pool.py", line 51, in
      starmapstar
    return list(itertools.starmap(args[0], args[1]))
  File "/home/ubuntu/upwards/py_upwards/upwards/converter_functions.py",
      line 657, in convert_turbine_data
    os.mkdir(out_folder + '/turbineOutput')
FileExistsError: [Errno 17] File exists: '/home/ubuntu/hawasser/Cases/
    Case_3/6MPS/_csv/turbineOutput'
"""

The above exception was the direct cause of the following exception:

FileExistsError                          Traceback (most recent call last)
<ipython-input-4-836ce6c2dd9b> in <module>
----> 1 cf.convert_case_as_csv("/home/ubuntu/hawasser/Cases/Case_3/6MPS/",
    "/home/ubuntu/hawasser/Cases/Case_3/6MPS/" + "_csv",
      2                               convert_field_values=["U", "UMean"],
      3                               convert_turbine_values=["rotSpeed", "
          powerRotor", "thrust", "torqueGen", "torqueRotor", "azimuth", "
          nacYaw", "pitch"],
      4                               convert_grid=True, skew=True)

~/upwards/py_upwards/upwards/converter_functions.py in convert_case_as_csv(
    in_folder, out_folder, convert_field_values, convert_turbine_values,
    convert_grid, skew, process_count)
```

```
     83
     84     with Pool(processes=process_count) as pool:
---> 85         pool.starmap(convert_turbine_data,
     86                      [(in_folder, out_folder, f) for f in
        convert_turbine_values])
     87

/usr/lib/python3.8/multiprocessing/pool.py in starmap(self, func, iterable,
    chunksize)
    370         `func` and (a, b) becomes func(a, b).
    371         '''
--> 372         return self._map_async(func, iterable, starmapstar,
    chunksize).get()
    373
    374     def starmap_async(self, func, iterable, chunksize=None,
        callback=None,

/usr/lib/python3.8/multiprocessing/pool.py in get(self, timeout)
    769             return self._value
    770         else:
--> 771             raise self._value
    772
    773     def _set(self, i, obj):

FileExistsError: [Errno 17] File exists: '/home/ubuntu/hawasser/Cases/
    Case_3/6MPS/_csv/turbineOutput'
```

E Case 1 - Code

*Note that source code are slightly changed from the original to improve visualisation of plots.

```
#!/usr/bin/env python
# coding: utf-8

# # Train a DMD model

# In[1]:


cd /home/ubuntu/upwards/py_upwards/


# In[2]:


get_ipython().run_line_magic('load_ext', 'autoreload')
get_ipython().run_line_magic('autoreload', '2')

import time

import upwards.load_functions as load
import upwards.plot_functions as plot
import upwards.supporting_functions as support
import upwards.dmd_functions as dmd_func
from upwards import converter_functions as cf
import os

import numpy as np
import matplotlib.pyplot as plt
from pydmd import DMD


# In[3]:


t0 = time.time()


# ## Load snapshot data

# Path where data is stored

# In[4]:


cf.convert_case_as_csv("/home/ubuntu/hawasser/Cases/Case_1.1/constant_6mps/
    Data/", "/home/ubuntu/hawasser/Cases/Case_1.1/constant_6mps/Data/" + "
    _csv",
                              convert_field_values=["U", "UMean"],
                              convert_turbine_values=["rotSpeed", "powerRotor", "
                                  thrust", "torqueGen", "torqueRotor", "azimuth",
                                    "nacYaw", "pitch"],
                                convert_grid=True, skew=True)


# In[5]:


cf.convert_case_as_csv("/home/ubuntu/hawasser/Cases/Case_1.1/constant_8mps/
    Data/", "/home/ubuntu/hawasser/Cases/Case_1.1/constant_8mps/Data/" + "
    _csv",
                                convert_field_values=["U", "UMean"],
                                convert_turbine_values=["rotSpeed", "powerRotor", "
                                    thrust", "torqueGen", "torqueRotor", "azimuth",
                                    "nacYaw", "pitch"],
                                convert_grid=True, skew=True)


# In[6]:


t1 = time.time()


# In[7]:


sim_path1 = "/home/ubuntu/hawasser/Cases/Case_1.1/constant_6mps/Data/_csv/"
sim_path2 = "/home/ubuntu/hawasser/Cases/Case_1.1/constant_8mps/Data/_csv/"


# load finite volume grid

# In[8]:


grid1 = load.openfoam_grid(sim_path1, load = True, skew = True)
grid2 = load.openfoam_grid(sim_path2, load = True, skew = True)


# In[9]:


data_dict_U1 = load.load_Data("UMean", sim_path1, time = -1)
data_dict_U2 = load.load_Data("UMean", sim_path2, time = -1)


# In[10]:


data_dict_U1["data"].shape  #check all utilized datadicts, find the
    limiting one and proceeed with correct number of snapshots
```

```
# In[11]:


data_dict_U2["data"].shape


# ## Preprocess snapshot data
# Remove a sufficient number of snapshots to approach steady-state
    conditions.

# In[12]:


data_dict_U1["data"] = data_dict_U1["data"][:, 50:231]   #limit
    dictionaries to stable snapshots
data_dict_U2["data"] = data_dict_U2["data"][:, 50:231]


# In[13]:


used_snapshots = len(data_dict_U1["data"].T) #Defines the number of
    snapshots used for data_dict_U1 and data_dict_U2
print('Used snapshots: ', used_snapshots)


# ## Estimate suffient rank
# Analyse the distribution of variance represented by the principle
    components to determine the required SVD rank.

# In[14]:


S1 = np.linalg.svd(data_dict_U1['data'], full_matrices = False)[1]
S2 = np.linalg.svd(data_dict_U2['data'], full_matrices = False)[1]

plt.figure(figsize=(10,5))
plt.plot(S1/sum(S1), 'bo')
plt.plot(S2/sum(S2), 'ro')
plt.yscale("log")
plt.grid()
plt.legend(["U=6ms", "U=8ms"],fontsize=20)
plt.xlabel('', fontsize=30)
plt.ylabel('', fontsize=30)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)

plt.show()


# ## Train DMD

# In[15]:
```

```
wasted_snapshots = 0
svd_rank_6ms = 10
svd_rank_8ms = 10
exact = True

error1, dmd1, dmd_dict1 = dmd_func.DMD_execute(data_dict_U1, used_snapshots
    , used_snapshots, wasted_snapshots, svd_rank_6ms, exact)
error2, dmd2, dmd_dict2 = dmd_func.DMD_execute(data_dict_U2, used_snapshots
    , used_snapshots, wasted_snapshots, svd_rank_8ms, exact)


# ## Interpolate a DMD model

# In[16]:


v_orig = 6.0 #wind velocity of data_1


# In[17]:


v_inter = 8.0 #wind velocity of the interpolation data, data_3


# In[18]:


dmd3 = dmd_func.interpolate_dmd(v_orig, dmd1, dmd_dict1, v_inter,
    svd_rank_6ms)


# In[19]:


t_dmd = time.time()


# In[20]:


v_orig2 = 8.0


# In[21]:


v_inter2 = 6.0


# In[22]:
```

```
dmd4 = dmd_func.interpolate_dmd(v_orig2, dmd2, dmd_dict2, v_inter2,
    svd_rank_8ms)


# In[23]:


t2 = time.time()


# ## System analysis
#
# ### Visualize Eigenvalues
#

# In[24]:


dmd1.plot_eigs(figsize=(10,10), title="System 1 (6m/s)")  # Eigenvalues of
    system resulting from inlet surface of 6ms
print(dmd1.eigs)                   #values of the eigenvalues
dmd2.plot_eigs(figsize=(10,10), title="System 2 (8m/s)")  # Eigenvalues of
    system resulting from inlet surface of 8ms
print(dmd2.eigs)                   #values of the eigenvalues
dmd3.plot_eigs(figsize=(10,10), title="System 3: 8ms interpolated")  #
    Eigenvalues of system which we interpolated from inlet surface of 6ms to
     8ms
print(dmd3.eigs)                   #values of the eigenvalues
dmd4.plot_eigs(figsize=(10,10), title="System 4: 6ms interpolated")  #
    Eigenvalues of system which we interpolated from inlet surface of 6ms to
     8ms
print(dmd4.eigs)                   #values of the eigenvalues


# ## Visualize modes

# In[25]:


plot.plot_all_modes(dmd3, dmd_dict1, grid2, 'x', 500, component = "norm",
    part = 'real')
plot.plot_all_modes(dmd3, dmd_dict1, grid2, 'x', 500, component = "norm",
    part = 'imag')
plot.plot_all_modes(dmd4, dmd_dict1, grid1, 'x', 500, component = "norm",
    part = 'real')
plot.plot_all_modes(dmd4, dmd_dict1, grid1, 'x', 500, component = "norm",
    part = 'imag')


# In[26]:
```

```
plot.plot_all_modes(dmd3, dmd_dict1, grid1, 'z', 300, component = 0, part =
      'real')
plot.plot_all_modes(dmd3, dmd_dict1, grid1, 'z', 300, component = 0, part =
      'imag')
plot.plot_all_modes(dmd4, dmd_dict2, grid2, 'z', 300, component = 0, part =
      'real')
plot.plot_all_modes(dmd4, dmd_dict2, grid2, 'z', 300, component = 0, part =
      'imag')


# In[27]:


plot.plot_all_modes(dmd3, dmd_dict1, grid1, 'z', 300, component = 1, part =
      'real')
plot.plot_all_modes(dmd3, dmd_dict1, grid1, 'z', 300, component = 1, part =
      'imag')
plot.plot_all_modes(dmd4, dmd_dict2, grid2, 'z', 300, component = 1, part =
      'real')
plot.plot_all_modes(dmd4, dmd_dict2, grid2, 'z', 300, component = 1, part =
      'imag')


# In[28]:


plot.plot_all_modes(dmd3, dmd_dict1, grid1, 'z', 300, component = 2, part =
      'real')
plot.plot_all_modes(dmd3, dmd_dict1, grid1, 'z', 300, component = 2, part =
      'imag')
plot.plot_all_modes(dmd4, dmd_dict2, grid2, 'z', 300, component = 2, part =
      'real')
plot.plot_all_modes(dmd4, dmd_dict2, grid2, 'z', 300, component = 2, part =
      'imag')


# In[29]:


plot.plot_all_modes(dmd3, dmd_dict1, grid1, 'y', 0, component = "norm",
    part = 'abs')
plot.plot_all_modes(dmd4, dmd_dict2, grid2, 'y', 0, component = "norm",
    part = 'abs')


# ## Visualize dynamics
#

# In[30]:


plot.plot_all_dynamics(dmd3, dmd_dict1)
plot.plot_all_dynamics(dmd4, dmd_dict2)


# ## Visualize predictions/reconstructions
# ### 6ms

# In[31]:


t3 = time.time()


# In[32]:


print(data_dict_U1['data'][:,:].shape)   # shape shows last possible
    snapshot for comparison


# In[33]:


data_dict_dmd1_U = dict()
data_dict_dmd1_U["data"] = dmd_func.reconstruct_snapshots(dmd1, 1,
    used_snapshots).real
data_dict_dmd1_U["dim"] = data_dict_U1["dim"]
data_dict_dmd1_U["name"] = data_dict_U1["name"]
data_dict_dmd1_U["delta_t"] = data_dict_U1["delta_t"]


Snapshot_error_string_1 =[]
for i in range(0 , used_snapshots):
          Snapshot_error_string_1.append(np.linalg.norm(data_dict_U1['data
              '][:,i:i + 1] - data_dict_dmd1_U["data"][:,i:i + 1])/np.linalg.
              norm(data_dict_U1['data'][:,i:i + 1]))

print("Average error rate: ",sum(Snapshot_error_string_1)/len(
    Snapshot_error_string_1))
print("Max error rate: ", max(Snapshot_error_string_1))
print("Peak error slice: ", np.argmax(Snapshot_error_string_1))

String_1_peak = np.argmax(Snapshot_error_string_1)


# In[34]:


t4 = time.time()


# In[35]:
```

```
plot.plot_data(data_dict_U1, grid1, 0, String_1_peak, variable="x",
    component="norm")
plot.plot_data(data_dict_dmd1_U, grid1, 0, String_1_peak, variable="x",
    component="norm")
plot.plot_data(data_dict_dmd1_U, grid1, 0, String_1_peak, variable="x",
    component=0)
plot.plot_data(data_dict_dmd1_U, grid1, 0, String_1_peak, variable="x",
    component=1)
plot.plot_data(data_dict_dmd1_U, grid1, 0, String_1_peak, variable="x",
    component=2)


# In[36]:


plot.plot_data(data_dict_U1, grid1, 0, String_1_peak, variable="y",
    component="norm")
plot.plot_data(data_dict_dmd1_U, grid1, 0, String_1_peak, variable="y",
    component="norm")
plot.plot_data(data_dict_dmd1_U, grid1, 0, String_1_peak, variable="y",
    component=0)
plot.plot_data(data_dict_dmd1_U, grid1, 0, String_1_peak, variable="y",
    component=1)
plot.plot_data(data_dict_dmd1_U, grid1, 0, String_1_peak, variable="y",
    component=2)


# In[37]:


plot.plot_data(data_dict_U1,     grid1, 300, String_1_peak, variable="z",
    component="norm")
plot.plot_data(data_dict_dmd1_U, grid1, 300, String_1_peak, variable="z",
    component="norm")
plot.plot_data(data_dict_dmd1_U, grid1, 300, String_1_peak, variable="z",
    component=0)
plot.plot_data(data_dict_dmd1_U, grid1, 300, String_1_peak, variable="z",
    component=1)
plot.plot_data(data_dict_dmd1_U, grid1, 300, String_1_peak, variable="z",
    component=2)


# In[38]:


t5 = time.time()


# ### 8ms

# In[39]:
```

```
data_dict_dmd2_U = dict()
data_dict_dmd2_U["data"] = dmd_func.reconstruct_snapshots(dmd2, 1,
    used_snapshots).real
data_dict_dmd2_U["dim"] = data_dict_U2["dim"]
data_dict_dmd2_U["name"] = data_dict_U2["name"]
data_dict_dmd2_U["delta_t"] = data_dict_U2["delta_t"]




Snapshot_error_string_2 =[]
for i in range(0 , used_snapshots):
        Snapshot_error_string_2.append(np.linalg.norm(data_dict_U2['data
            '][:,i:i + 1] - data_dict_dmd2_U["data"][:,i:i + 1])/np.linalg.
            norm(data_dict_U2['data'][:,i:i + 1]))

print("Average error rate: ",sum(Snapshot_error_string_2)/len(
    Snapshot_error_string_2))
print("Max error rate: ", max(Snapshot_error_string_2))
print("Peak error slice: ", np.argmax(Snapshot_error_string_2))

String_2_peak =   np.argmax(Snapshot_error_string_2)


# In[40]:


t6 = time.time()


# In[41]:


plot.plot_data(data_dict_U2      , grid2, 0, String_2_peak, variable="x",
    component="norm")
plot.plot_data(data_dict_dmd2_U, grid2, 0, String_2_peak, variable="x",
    component="norm")
plot.plot_data(data_dict_dmd2_U, grid2, 0, String_2_peak, variable="x",
    component=0)
plot.plot_data(data_dict_dmd2_U, grid2, 0, String_2_peak, variable="x",
    component=1)
plot.plot_data(data_dict_dmd2_U, grid2, 0, String_2_peak, variable="x",
    component=2)


# In[42]:


plot.plot_data(data_dict_U2      , grid2, 0, String_2_peak, variable="y",
    component="norm")
plot.plot_data(data_dict_dmd2_U, grid2, 0, String_2_peak, variable="y",
    component="norm")
```

```
plot.plot_data(data_dict_dmd2_U, grid2, 0, String_2_peak, variable="y",
    component=0)
plot.plot_data(data_dict_dmd2_U, grid2, 0, String_2_peak, variable="y",
    component=1)
plot.plot_data(data_dict_dmd2_U, grid2, 0, String_2_peak, variable="y",
    component=2)


# In[43]:


plot.plot_data(data_dict_U2      , grid2, 300, String_2_peak, variable="z",
    component="norm")
plot.plot_data(data_dict_dmd2_U, grid2, 300, String_2_peak, variable="z",
    component="norm")
plot.plot_data(data_dict_dmd2_U, grid2, 300, String_2_peak, variable="z",
    component=0)
plot.plot_data(data_dict_dmd2_U, grid2, 300, String_2_peak, variable="z",
    component=1)
plot.plot_data(data_dict_dmd2_U, grid2, 300, String_2_peak, variable="z",
    component=2)


# In[44]:


plt.figure(figsize=(10,5))
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.plot(Snapshot_error_string_1, 'x', label='DMD 1 error rate')
plt.plot(Snapshot_error_string_2, '+', label='DMD 2 error rate')
plt.xlabel('Snapshot index', fontsize=30)
plt.ylabel('Error rate', fontsize=30)
plt.grid()
plt.legend(fontsize=20)
plt.show()


# ### 8ms interpolated

# In[45]:


t7 = time.time()


# In[46]:


data_dict_dmd3_U = dict()
data_dict_dmd3_U["data"] = dmd_func.reconstruct_snapshots(dmd3, 1,
    used_snapshots).real
data_dict_dmd3_U["dim"] = data_dict_U1["dim"]
```

```
data_dict_dmd3_U["name"] = data_dict_U1["name"]
data_dict_dmd3_U["delta_t"] = data_dict_U1["delta_t"]

data_dict_dmd3_U["data"] = data_dict_dmd3_U["data"] *(v_inter/
    data_dict_dmd3_U["data"][0][0])

Snapshot_error_string_3 =[]
for i in range(0 , used_snapshots):
        Snapshot_error_string_3.append(np.linalg.norm(data_dict_U2['data
            '][:,i:i + 1] - data_dict_dmd3_U["data"][:,i:i + 1])/np.linalg.
            norm(data_dict_U2['data'][:,i:i + 1]))

String_3_peak =   np.argmax(Snapshot_error_string_3)

print("Average error rate: ",sum(Snapshot_error_string_3)/len(
    Snapshot_error_string_3))
print("Max error rate: ", max(Snapshot_error_string_3))
print("Peak error slice: ", np.argmax(Snapshot_error_string_3))


# In[47]:


t8 = time.time()


# In[48]:


plot.plot_data(data_dict_U2      , grid2, 0, String_3_peak, variable="x",
    component="norm")
plot.plot_data(data_dict_dmd3_U, grid2, 0, String_3_peak, variable="x",
    component="norm")
plot.plot_data(data_dict_dmd3_U, grid1, 0, String_3_peak, variable="x",
    component=0)
plot.plot_data(data_dict_dmd3_U, grid1, 0, String_3_peak, variable="x",
    component=1)
plot.plot_data(data_dict_dmd3_U, grid1, 0, String_3_peak, variable="x",
    component=2)


# In[49]:


plot.plot_data(data_dict_U2      , grid2, 0, String_3_peak, variable="y",
    component="norm")
plot.plot_data(data_dict_dmd3_U, grid2, 0, String_3_peak, variable="y",
    component="norm")
plot.plot_data(data_dict_dmd3_U, grid1, 0, String_3_peak, variable="y",
    component=0)
plot.plot_data(data_dict_dmd3_U, grid1, 0, String_3_peak, variable="y",
    component=1)
```

```
plot.plot_data(data_dict_dmd3_U, grid1, 0, String_3_peak, variable="y",
    component=2)


# In[50]:


plot.plot_data(data_dict_U2       , grid2, 300, String_3_peak, variable="z",
    component="norm")
plot.plot_data(data_dict_dmd3_U, grid2, 300, String_3_peak, variable="z",
    component="norm")
plot.plot_data(data_dict_dmd3_U, grid1, 300, String_3_peak, variable="z",
    component=0)
plot.plot_data(data_dict_dmd3_U, grid1, 300, String_3_peak, variable="z",
    component=1)
plot.plot_data(data_dict_dmd3_U, grid1, 300, String_3_peak, variable="z",
    component=2)


# In[51]:


t9 = time.time()


# In[52]:


data_dict_dmd4_U = dict()
data_dict_dmd4_U["data"] = dmd_func.reconstruct_snapshots(dmd4, 1,
    used_snapshots).real
data_dict_dmd4_U["dim"] = data_dict_U1["dim"]
data_dict_dmd4_U["name"] = data_dict_U1["name"]
data_dict_dmd4_U["delta_t"] = data_dict_U1["delta_t"]

data_dict_dmd4_U["data"] = data_dict_dmd4_U["data"] *(v_inter2/
    data_dict_dmd4_U["data"][0][0])

Snapshot_error_string_4 =[]
for i in range(0 , used_snapshots):
        Snapshot_error_string_4.append(np.linalg.norm(data_dict_U1['data
            '][:,i:i + 1] - data_dict_dmd4_U["data"][:,i:i + 1])/np.linalg.
            norm(data_dict_U1['data'][:,i:i + 1]))

print("Average error rate: ",sum(Snapshot_error_string_4)/len(
    Snapshot_error_string_4))
print("Max error rate: ", max(Snapshot_error_string_4))
print("Peak error slice: ", np.argmax(Snapshot_error_string_4))


String_4_peak =   np.argmax(Snapshot_error_string_4)
```

```
# In[53]:


t10 = time.time()


# In[54]:


plot.plot_data(data_dict_U1      , grid1, 0, String_4_peak, variable="x",
    component="norm")
plot.plot_data(data_dict_dmd4_U, grid1, 0, String_4_peak, variable="x",
    component="norm")
plot.plot_data(data_dict_dmd4_U, grid1, 0, String_4_peak, variable="x",
    component=0)
plot.plot_data(data_dict_dmd4_U, grid1, 0, String_4_peak, variable="x",
    component=1)
plot.plot_data(data_dict_dmd4_U, grid1, 0, String_4_peak, variable="x",
    component=2)


# In[55]:


plot.plot_data(data_dict_U1      , grid1, 0, String_4_peak, variable="y",
    component="norm")
plot.plot_data(data_dict_dmd4_U, grid1, 0, String_4_peak, variable="y",
    component="norm")
plot.plot_data(data_dict_dmd4_U, grid1, 0, String_4_peak, variable="y",
    component=0)
plot.plot_data(data_dict_dmd4_U, grid1, 0, String_4_peak, variable="y",
    component=1)
plot.plot_data(data_dict_dmd4_U, grid1, 0, String_4_peak, variable="y",
    component=2)


# In[56]:


plot.plot_data(data_dict_U1      , grid1, 300, String_4_peak, variable="z",
    component="norm")
plot.plot_data(data_dict_dmd4_U, grid1, 300, String_4_peak, variable="z",
    component="norm")
plot.plot_data(data_dict_dmd4_U, grid1, 300, String_4_peak, variable="z",
    component=0)
plot.plot_data(data_dict_dmd4_U, grid1, 300, String_4_peak, variable="z",
    component=1)
plot.plot_data(data_dict_dmd4_U, grid1, 300, String_4_peak, variable="z",
    component=2)


# In[57]:
```

```
plot.plot_data(data_dict_U2      , grid1, 1161, String_3_peak, variable="x",
     component="norm")
plot.plot_data(data_dict_dmd3_U, grid1, 1161, String_3_peak, variable="x",
    component="norm")


# In[58]:


plot.plot_data(data_dict_U1      , grid1, 1161, String_4_peak, variable="x",
     component="norm")
plot.plot_data(data_dict_dmd4_U, grid1, 1161, String_4_peak, variable="x",
    component="norm")


# ## Compare interpolated and simulated snapshots

# In[59]:


data_dict_dmd1_error_U = dict()
data_dict_dmd1_error_U["data"] = np.absolute(data_dict_U1["data"][:,
    String_1_peak:String_1_peak + 1] - data_dict_dmd1_U["data"][:,
    String_1_peak:String_1_peak + 1])
data_dict_dmd1_error_U["dim"] = data_dict_U1["dim"]
data_dict_dmd1_error_U["name"] = data_dict_U1["name"]
data_dict_dmd1_error_U["delta_t"] = data_dict_U1["delta_t"]


# In[60]:


plot.plot_data(data_dict_dmd1_error_U, grid1, 0, 0, variable="x", component
    ="norm")
plot.plot_data(data_dict_dmd1_error_U, grid1, 0, 0, variable="x", component
    =0)
plot.plot_data(data_dict_dmd1_error_U, grid1, 0, 0, variable="x", component
    =1)
plot.plot_data(data_dict_dmd1_error_U, grid1, 0, 0, variable="x", component
    =2)


# In[61]:


plot.plot_data(data_dict_dmd1_error_U, grid1, 0, 0, variable="y", component
    ="norm")
plot.plot_data(data_dict_dmd1_error_U, grid1, 0, 0, variable="y", component
    =0)
plot.plot_data(data_dict_dmd1_error_U, grid1, 0, 0, variable="y", component
    =1)
```

```
plot.plot_data(data_dict_dmd1_error_U, grid1, 0, 0, variable="y", component
    =2)


# In[62]:


plot.plot_data(data_dict_dmd1_error_U, grid1, 300, 0, variable="z",
    component="norm")
plot.plot_data(data_dict_dmd1_error_U, grid1, 300, 0, variable="z",
    component=0)
plot.plot_data(data_dict_dmd1_error_U, grid1, 300, 0, variable="z",
    component=1)
plot.plot_data(data_dict_dmd1_error_U, grid1, 300, 0, variable="z",
    component=2)


# In[63]:


data_dict_dmd2_error_U = dict()
data_dict_dmd2_error_U["data"] = np.absolute(data_dict_U2["data"][:,
    String_2_peak:String_2_peak + 1] - data_dict_dmd2_U["data"][:,
    String_2_peak:String_2_peak + 1])
data_dict_dmd2_error_U["dim"] = data_dict_U2["dim"]
data_dict_dmd2_error_U["name"] = data_dict_U2["name"]
data_dict_dmd2_error_U["delta_t"] = data_dict_U2["delta_t"]


# In[64]:


plot.plot_data(data_dict_dmd2_error_U, grid2, 0, 0, variable="x", component
    ="norm")
plot.plot_data(data_dict_dmd2_error_U, grid2, 0, 0, variable="x", component
    =0)
plot.plot_data(data_dict_dmd2_error_U, grid2, 0, 0, variable="x", component
    =1)
plot.plot_data(data_dict_dmd2_error_U, grid2, 0, 0, variable="x", component
    =2)


# In[65]:


plot.plot_data(data_dict_dmd2_error_U, grid2, 0, 0, variable="y", component
    ="norm")
plot.plot_data(data_dict_dmd2_error_U, grid2, 0, 0, variable="y", component
    =0)
plot.plot_data(data_dict_dmd2_error_U, grid2, 0, 0, variable="y", component
    =1)
plot.plot_data(data_dict_dmd2_error_U, grid2, 0, 0, variable="y", component
    =2)
```

```
# In[66]:


plot.plot_data(data_dict_dmd2_error_U, grid2, 300, 0, variable="z",
    component="norm")
plot.plot_data(data_dict_dmd2_error_U, grid2, 300, 0, variable="z",
    component=0)
plot.plot_data(data_dict_dmd2_error_U, grid2, 300, 0, variable="z",
    component=1)
plot.plot_data(data_dict_dmd2_error_U, grid2, 300, 0, variable="z",
    component=2)


# In[67]:


data_dict_dmd3_error_U = dict()
data_dict_dmd3_error_U["data"] = np.absolute(data_dict_U2["data"][:,
    String_3_peak:String_3_peak + 1] - data_dict_dmd3_U["data"][:,
    String_3_peak:String_3_peak + 1])
data_dict_dmd3_error_U["dim"] = data_dict_U2["dim"]
data_dict_dmd3_error_U["name"] = data_dict_U2["name"]
data_dict_dmd3_error_U["delta_t"] = data_dict_U2["delta_t"]


# In[68]:


plot.plot_data(data_dict_dmd3_error_U, grid2, 0, 0, variable="x", component
    ="norm")
plot.plot_data(data_dict_dmd3_error_U, grid2, 0, 0, variable="x", component
    =0)
plot.plot_data(data_dict_dmd3_error_U, grid2, 0, 0, variable="x", component
    =1)
plot.plot_data(data_dict_dmd3_error_U, grid2, 0, 0, variable="x", component
    =2)


# In[69]:


plot.plot_data(data_dict_dmd3_error_U, grid2, 0, 0, variable="y", component
    ="norm")
plot.plot_data(data_dict_dmd3_error_U, grid2, 0, 0, variable="y", component
    =0)
plot.plot_data(data_dict_dmd3_error_U, grid2, 0, 0, variable="y", component
    =1)
plot.plot_data(data_dict_dmd3_error_U, grid2, 0, 0, variable="y", component
    =2)
```

```
# In[70]:


plot.plot_data(data_dict_dmd3_error_U, grid2, 300, 0, variable="z",
    component="norm")
plot.plot_data(data_dict_dmd3_error_U, grid2, 300, 0, variable="z",
    component=0)
plot.plot_data(data_dict_dmd3_error_U, grid2, 300, 0, variable="z",
    component=1)
plot.plot_data(data_dict_dmd3_error_U, grid2, 300, 0, variable="z",
    component=2)


# In[71]:


data_dict_dmd4_error_U = dict()
data_dict_dmd4_error_U["data"] = np.absolute(data_dict_U1["data"][:,
    String_4_peak:String_4_peak + 1] - data_dict_dmd4_U["data"][:,
    String_4_peak:String_4_peak + 1])
data_dict_dmd4_error_U["dim"] = data_dict_U1["dim"]
data_dict_dmd4_error_U["name"] = data_dict_U1["name"]
data_dict_dmd4_error_U["delta_t"] = data_dict_U1["delta_t"]


# In[72]:


plot.plot_data(data_dict_dmd4_error_U, grid1, 0, 0, variable="x", component
    ="norm")
plot.plot_data(data_dict_dmd4_error_U, grid1, 0, 0, variable="x", component
    =0)
plot.plot_data(data_dict_dmd4_error_U, grid1, 0, 0, variable="x", component
    =1)
plot.plot_data(data_dict_dmd4_error_U, grid1, 0, 0, variable="x", component
    =2)


# In[73]:


plot.plot_data(data_dict_dmd4_error_U, grid1, 0, 0, variable="y", component
    ="norm")
plot.plot_data(data_dict_dmd4_error_U, grid1, 0, 0, variable="y", component
    =0)
plot.plot_data(data_dict_dmd4_error_U, grid1, 0, 0, variable="y", component
    =1)
plot.plot_data(data_dict_dmd4_error_U, grid1, 0, 0, variable="y", component
    =2)


# In[74]:
```

```
plot.plot_data(data_dict_dmd4_error_U, grid1, 300, 0, variable="z",
    component="norm")
plot.plot_data(data_dict_dmd4_error_U, grid1, 300, 0, variable="z",
    component=0)
plot.plot_data(data_dict_dmd4_error_U, grid1, 300, 0, variable="z",
    component=1)
plot.plot_data(data_dict_dmd4_error_U, grid1, 300, 0, variable="z",
    component=2)


# In[75]:


plot.plot_data(data_dict_dmd1_error_U, grid1, 1161, 0, variable="x",
    component="norm")
plot.plot_data(data_dict_dmd2_error_U, grid2, 1161, 0, variable="x",
    component="norm")
plot.plot_data(data_dict_dmd3_error_U, grid2, 1161, 0, variable="x",
    component="norm")
plot.plot_data(data_dict_dmd4_error_U, grid1, 1161, 0, variable="x",
    component="norm")


# In[76]:


plt.figure(figsize=(10,5))
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.plot(Snapshot_error_string_3, 'x', label='DMD 3 error rate')
plt.plot(Snapshot_error_string_4, '+', label='DMD 4 error rate')
plt.xlabel('Snapshot index', fontsize=30)
plt.ylabel('Error rate', fontsize=30)
plt.grid()
plt.legend(fontsize=20)
plt.show()


# In[77]:


plt.figure(figsize=(10,5))
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.plot(Snapshot_error_string_1, label='DMD 1 error rate')
plt.plot(Snapshot_error_string_2, label='DMD 2 error rate')
plt.plot(Snapshot_error_string_3, label='DMD 3 error rate')
plt.plot(Snapshot_error_string_4, label='DMD 4 error rate')
plt.xlabel('Snapshot index', fontsize=30)
plt.ylabel('Error rate', fontsize=30)
plt.grid()
plt.legend(fontsize=20)
plt.show()


# In[78]:


t11 = time.time()
total = t11-t0
dmd_only = t10-t1-(t9-t8)-(t7-t6)-(t5-t4)-(t3-t2)


# In[79]:


total


# In[80]:


dmd_only


# In[81]:


DMD_interpolate = t2-t_dmd + t10 - t9
print('Time spent creating DMD4 only: ', DMD_interpolate)


# In[82]:


Peak_6ms = dict()
Peak_6ms['data']=          data_dict_U1['data'][:,np.argmax(
    Snapshot_error_string_4):np.argmax(Snapshot_error_string_4) + 1]
Peak_6ms["dim"] =          data_dict_U1["dim"]
Peak_6ms["name"] =         data_dict_U1["name"]
Peak_6ms["delta_t"] = data_dict_U1["delta_t"]


# In[83]:


Peak_8ms = dict()
Peak_8ms['data'] =         data_dict_U2['data'][:,np.argmax(
    Snapshot_error_string_3):np.argmax(Snapshot_error_string_3) + 1]
Peak_8ms["dim"] =          data_dict_U2["dim"]
Peak_8ms["name"] =         data_dict_U2["name"]
Peak_8ms["delta_t"] = data_dict_U2["delta_t"]


# In[84]:
```

```
Peak_dmd3 = dict()
Peak_dmd3['data']=        data_dict_dmd3_U['data'][:,np.argmax(
    Snapshot_error_string_3):np.argmax(Snapshot_error_string_3) + 1]
Peak_dmd3["dim"] =        data_dict_dmd3_U["dim"]
Peak_dmd3["name"] =       data_dict_dmd3_U["name"]
Peak_dmd3["delta_t"] = data_dict_dmd3_U["delta_t"]


# In[85]:


Peak_dmd4 = dict()
Peak_dmd4['data']=        data_dict_dmd4_U['data'][:,np.argmax(
    Snapshot_error_string_4):np.argmax(Snapshot_error_string_4) + 1]
Peak_dmd4["dim"] =        data_dict_dmd4_U["dim"]
Peak_dmd4["name"] =       data_dict_dmd4_U["name"]
Peak_dmd4["delta_t"] = data_dict_dmd4_U["delta_t"]


# In[86]:


inlet_u1_1 = support.extract_inlet(grid1, [0,0,33360/137   ], 1 , 1, 0, 270,
     0)
inlet_u1_2 = support.extract_inlet(grid1, [0,0,271.7518248], 1 , 1, 0, 270,
     0)
inlet_u1_3 = support.extract_inlet(grid1, [0,0,300         ], 1 , 1, 0, 270,
    0)
inlet_u1_4 = support.extract_inlet(grid1, [0,0,44970/137   ], 1 , 1, 0, 270,
     0)
inlet_u1_5 = support.extract_inlet(grid1, [0,0,48840/137   ], 1 , 1, 0, 270,
     0)

data_list_u1_1 , x_list , y_list , z_list = plot.extract_inlet_data('norm',
    Peak_6ms, inlet_u1_1 , 0)
data_list_u1_2 , x_list , y_list , z_list = plot.extract_inlet_data('norm',
    Peak_6ms, inlet_u1_2 , 0)
data_list_u1_3 , x_list , y_list , z_list = plot.extract_inlet_data('norm',
    Peak_6ms, inlet_u1_3 , 0)
data_list_u1_4 , x_list , y_list , z_list = plot.extract_inlet_data('norm',
    Peak_6ms, inlet_u1_4 , 0)
data_list_u1_5 , x_list , y_list , z_list = plot.extract_inlet_data('norm',
    Peak_6ms, inlet_u1_5 , 0)

u1_p1 = np.sum(data_list_u1_1)/len(data_list_u1_1)
u1_p2 = np.sum(data_list_u1_2)/len(data_list_u1_2)
u1_p3 = np.sum(data_list_u1_3)/len(data_list_u1_3)
u1_p4 = np.sum(data_list_u1_4)/len(data_list_u1_4)
u1_p5 = np.sum(data_list_u1_5)/len(data_list_u1_5)
```

```
print('Points: ', u1_p1, u1_p2, u1_p3, u1_p4, u1_p5)

REWS_U1=u1_p1*(0.1142)+u1_p2*(0.2475)+u1_p3*(0.2766)+u1_p4*(0.2475)+u1_p5
    *(0.1142)

print('REWS: ',REWS_U1)


# In[87]:


inlet_u2_1 = support.extract_inlet(grid2, [0,0,33360/137  ], 1 , 1, 0,270 ,
     0)
inlet_u2_2 = support.extract_inlet(grid2, [0,0,271.7518248], 1 , 1, 0,270 ,
     0)
inlet_u2_3 = support.extract_inlet(grid2, [0,0,300        ], 1 , 1, 0,270 ,
    0)
inlet_u2_4 = support.extract_inlet(grid2, [0,0,44970/137  ], 1 , 1, 0,270 ,
     0)
inlet_u2_5 = support.extract_inlet(grid2, [0,0,48840/137  ], 1 , 1, 0,270 ,
     0)

data_list_u2_1, x_list, y_list, z_list = plot.extract_inlet_data('norm',
    Peak_8ms, inlet_u2_1, 0)
data_list_u2_2, x_list, y_list, z_list = plot.extract_inlet_data('norm',
    Peak_8ms, inlet_u2_2, 0)
data_list_u2_3, x_list, y_list, z_list = plot.extract_inlet_data('norm',
    Peak_8ms, inlet_u2_3, 0)
data_list_u2_4, x_list, y_list, z_list = plot.extract_inlet_data('norm',
    Peak_8ms, inlet_u2_4, 0)
data_list_u2_5, x_list, y_list, z_list = plot.extract_inlet_data('norm',
    Peak_8ms, inlet_u2_5, 0)

u2_p1 = np.sum(data_list_u2_1)/len(data_list_u2_1)
u2_p2 = np.sum(data_list_u2_2)/len(data_list_u2_2)
u2_p3 = np.sum(data_list_u2_3)/len(data_list_u2_3)
u2_p4 = np.sum(data_list_u2_4)/len(data_list_u2_4)
u2_p5 = np.sum(data_list_u2_5)/len(data_list_u2_5)

print('Points: ', u2_p1, u2_p2, u2_p3, u2_p4, u2_p5)

REWS_U2=u2_p1*(0.1142)+u2_p2*(0.2475)+u2_p3*(0.2766)+u2_p4*(0.2475)+u2_p5
    *(0.1142)
print('REWS: ', REWS_U2)


# In[88]:


inlet_dmd3_1 = support.extract_inlet(grid2, [0,0,33360/137  ], 1 , 1, 0,
    270 , 0)
inlet_dmd3_2 = support.extract_inlet(grid2, [0,0,271.7518248], 1 , 1, 0,
    270 , 0)
```

```
inlet_dmd3_3 = support.extract_inlet(grid2, [0,0,300          ], 1 , 1,
    0, 270 , 0)
inlet_dmd3_4 = support.extract_inlet(grid2, [0,0,44970/137  ], 1 , 1, 0,
    270 , 0)
inlet_dmd3_5 = support.extract_inlet(grid2, [0,0,48840/137  ], 1 , 1, 0,
    270 , 0)

data_list_dmd3_1, x_list, y_list, z_list = plot.extract_inlet_data('norm',
    Peak_dmd3, inlet_dmd3_1, 0)
data_list_dmd3_2, x_list, y_list, z_list = plot.extract_inlet_data('norm',
    Peak_dmd3, inlet_dmd3_2, 0)
data_list_dmd3_3, x_list, y_list, z_list = plot.extract_inlet_data('norm',
    Peak_dmd3, inlet_dmd3_3, 0)
data_list_dmd3_4, x_list, y_list, z_list = plot.extract_inlet_data('norm',
    Peak_dmd3, inlet_dmd3_4, 0)
data_list_dmd3_5, x_list, y_list, z_list = plot.extract_inlet_data('norm',
    Peak_dmd3, inlet_dmd3_5, 0)

dmd3_p1 = np.sum(data_list_dmd3_1)/len(data_list_dmd3_1)
dmd3_p2 = np.sum(data_list_dmd3_2)/len(data_list_dmd3_2)
dmd3_p3 = np.sum(data_list_dmd3_3)/len(data_list_dmd3_3)
dmd3_p4 = np.sum(data_list_dmd3_4)/len(data_list_dmd3_4)
dmd3_p5 = np.sum(data_list_dmd3_5)/len(data_list_dmd3_5)

print('Points: ', dmd3_p1, dmd3_p2, dmd3_p3, dmd3_p4, dmd3_p5)

REWS_dmd3=dmd3_p1*(0.1142)+dmd3_p2*(0.2475)+dmd3_p3*(0.2766)+dmd3_p4
    *(0.2475)+dmd3_p5*(0.1142)
print('REWS: ', REWS_dmd3)


# In[89]:


inlet_dmd4_1 = support.extract_inlet(grid1, [0,0,33360/137  ], 1 , 1, 0,
    270 , 0)
inlet_dmd4_2 = support.extract_inlet(grid1, [0,0,271.7518248], 1 , 1, 0,
    270 , 0)
inlet_dmd4_3 = support.extract_inlet(grid1, [0,0,300          ], 1 , 1,
    0, 270 , 0)
inlet_dmd4_4 = support.extract_inlet(grid1, [0,0,44970/137  ], 1 , 1, 0,
    270 , 0)
inlet_dmd4_5 = support.extract_inlet(grid1, [0,0,48840/137  ], 1 , 1, 0,
    270 , 0)

data_list_dmd4_1, x_list, y_list, z_list = plot.extract_inlet_data('norm',
    Peak_dmd4, inlet_dmd4_1, 0)
data_list_dmd4_2, x_list, y_list, z_list = plot.extract_inlet_data('norm',
    Peak_dmd4, inlet_dmd4_2, 0)
data_list_dmd4_3, x_list, y_list, z_list = plot.extract_inlet_data('norm',
    Peak_dmd4, inlet_dmd4_3, 0)
data_list_dmd4_4, x_list, y_list, z_list = plot.extract_inlet_data('norm',
    Peak_dmd4, inlet_dmd4_4, 0)
```

```
data_list_dmd4_5 , x_list , y_list , z_list = plot.extract_inlet_data('norm',
    Peak_dmd4, inlet_dmd4_5 , 0)

dmd4_p1 = np.sum(data_list_dmd4_1)/len(data_list_dmd4_1)
dmd4_p2 = np.sum(data_list_dmd4_2)/len(data_list_dmd4_2)
dmd4_p3 = np.sum(data_list_dmd4_3)/len(data_list_dmd4_3)
dmd4_p4 = np.sum(data_list_dmd4_4)/len(data_list_dmd4_4)
dmd4_p5 = np.sum(data_list_dmd4_5)/len(data_list_dmd4_5)

print('Points: ', dmd4_p1, dmd4_p2, dmd4_p3, dmd4_p4, dmd4_p5)

REWS_dmd4=dmd4_p1*(0.1142)+dmd4_p2*(0.2475)+dmd4_p3*(0.2766)+dmd4_p4
    *(0.2475)+dmd4_p5*(0.1142)
print('REWS: ', REWS_dmd4)


# In[90]:


inlet_u1_6 = support.extract_inlet(grid1,  [1161,0,33360/137   ], 1 , 1, 0,
    270, 0)
inlet_u1_7 = support.extract_inlet(grid1,  [1161,0,271.7518248], 1 , 1, 0,
    270, 0)
inlet_u1_8 = support.extract_inlet(grid1,  [1161,0,300         ], 1 , 1,
    0, 270, 0)
inlet_u1_9 = support.extract_inlet(grid1,  [1161,0,44970/137   ], 1 , 1, 0,
    270, 0)
inlet_u1_10 = support.extract_inlet(grid1, [1161,0,48840/137   ], 1 , 1, 0,
    270, 0)

data_list_u1_6 , x_list , y_list , z_list   = plot.extract_inlet_data('norm',
    Peak_6ms, inlet_u1_6 , 0)
data_list_u1_7 , x_list , y_list , z_list   = plot.extract_inlet_data('norm',
    Peak_6ms, inlet_u1_7 , 0)
data_list_u1_8 , x_list , y_list , z_list   = plot.extract_inlet_data('norm',
    Peak_6ms, inlet_u1_8 , 0)
data_list_u1_9 , x_list , y_list , z_list   = plot.extract_inlet_data('norm',
    Peak_6ms, inlet_u1_9 , 0)
data_list_u1_10 , x_list , y_list , z_list  = plot.extract_inlet_data('norm',
    Peak_6ms, inlet_u1_10 , 0)

u1_p6  = np.sum(data_list_u1_6)/len(data_list_u1_6)
u1_p7  = np.sum(data_list_u1_7)/len(data_list_u1_7)
u1_p8  = np.sum(data_list_u1_8)/len(data_list_u1_8)
u1_p9  = np.sum(data_list_u1_9)/len(data_list_u1_9)
u1_p10 = np.sum(data_list_u1_10)/len(data_list_u1_10)


print('Points: ', u1_p6, u1_p7, u1_p8, u1_p9, u1_p10)

REWS_U1_1161=u1_p6*(0.1142)+u1_p7*(0.2475)+u1_p8*(0.2766)+u1_p9*(0.2475)+
    u1_p10*(0.1142)
```

```
print('REWS: ',REWS_U1_1161)


# In[91]:


inlet_u2_6 = support.extract_inlet(grid2,  [1161,0,33360/137  ], 1 , 1, 0,
    270, 0)
inlet_u2_7 = support.extract_inlet(grid2,  [1161,0,271.7518248], 1 , 1, 0,
    270, 0)
inlet_u2_8 = support.extract_inlet(grid2,  [1161,0,300        ], 1 , 1,
    0, 270, 0)
inlet_u2_9 = support.extract_inlet(grid2,  [1161,0,44970/137  ], 1 , 1, 0,
    270, 0)
inlet_u2_10 = support.extract_inlet(grid2, [1161,0,48840/137  ], 1 , 1, 0,
    270, 0)

data_list_u2_6 , x_list , y_list , z_list  = plot.extract_inlet_data('norm',
    Peak_8ms, inlet_u2_6 , 0)
data_list_u2_7 , x_list , y_list , z_list  = plot.extract_inlet_data('norm',
    Peak_8ms, inlet_u2_7 , 0)
data_list_u2_8 , x_list , y_list , z_list  = plot.extract_inlet_data('norm',
    Peak_8ms, inlet_u2_8 , 0)
data_list_u2_9 , x_list , y_list , z_list  = plot.extract_inlet_data('norm',
    Peak_8ms, inlet_u2_9 , 0)
data_list_u2_10 , x_list , y_list , z_list  = plot.extract_inlet_data('norm',
    Peak_8ms, inlet_u2_10 , 0)

u2_p6  = np.sum(data_list_u2_6)/len(data_list_u2_6)
u2_p7  = np.sum(data_list_u2_7)/len(data_list_u2_7)
u2_p8  = np.sum(data_list_u2_8)/len(data_list_u2_8)
u2_p9  = np.sum(data_list_u2_9)/len(data_list_u2_9)
u2_p10 = np.sum(data_list_u2_10)/len(data_list_u2_10)


print('Points: ', u2_p6, u2_p7, u2_p8, u2_p9, u2_p10)

REWS_U2_1161=u2_p6*(0.1142)+u2_p7*(0.2475)+u2_p8*(0.2766)+u2_p9*(0.2475)+
    u2_p10*(0.1142)

print('REWS: ',REWS_U2_1161)


# In[92]:


inlet_dmd3_6 = support.extract_inlet(grid2,  [1161,0,33360/137  ], 1 , 1,
    0, 270, 0)
inlet_dmd3_7 = support.extract_inlet(grid2,  [1161,0,271.7518248], 1 , 1,
    0, 270, 0)
inlet_dmd3_8 = support.extract_inlet(grid2,  [1161,0,300        ], 1 , 1,
    0, 270, 0)
```

```
inlet_dmd3_9 = support.extract_inlet(grid2,  [1161,0,44970/137   ], 1 , 1,
    0, 270, 0)
inlet_dmd3_10 = support.extract_inlet(grid2, [1161,0,48840/137   ], 1 , 1,
    0, 270, 0)

data_list_dmd3_6 , x_list , y_list , z_list   = plot.extract_inlet_data('norm
    ', Peak_dmd3, inlet_dmd3_6 , 0)
data_list_dmd3_7 , x_list , y_list , z_list   = plot.extract_inlet_data('norm
    ', Peak_dmd3, inlet_dmd3_7 , 0)
data_list_dmd3_8 , x_list , y_list , z_list   = plot.extract_inlet_data('norm
    ', Peak_dmd3, inlet_dmd3_8 , 0)
data_list_dmd3_9 , x_list , y_list , z_list   = plot.extract_inlet_data('norm
    ', Peak_dmd3, inlet_dmd3_9 , 0)
data_list_dmd3_10 , x_list , y_list , z_list  = plot.extract_inlet_data('norm
    ', Peak_dmd3, inlet_dmd3_10 , 0)

dmd3_p6  = np.sum(data_list_dmd3_6)/len(data_list_dmd3_6)
dmd3_p7  = np.sum(data_list_dmd3_7)/len(data_list_dmd3_7)
dmd3_p8  = np.sum(data_list_dmd3_8)/len(data_list_dmd3_8)
dmd3_p9  = np.sum(data_list_dmd3_9)/len(data_list_dmd3_9)
dmd3_p10 = np.sum(data_list_dmd3_10)/len(data_list_dmd3_10)


print('Points: ', dmd3_p6, dmd3_p7, dmd3_p8, dmd3_p9, dmd3_p10)

REWS_dmd3_1161=dmd3_p6*(0.1142)+dmd3_p7*(0.2475)+dmd3_p8*(0.2766)+dmd3_p9
    *(0.2475)+dmd3_p10*(0.1142)

print('REWS: ',REWS_dmd3_1161)


# In[93]:


inlet_dmd4_6 = support.extract_inlet(grid1,  [1161,0,33360/137   ], 1 , 1,
    0, 270, 0)
inlet_dmd4_7 = support.extract_inlet(grid1,  [1161,0,271.7518248], 1 , 1,
    0, 270, 0)
inlet_dmd4_8 = support.extract_inlet(grid1,  [1161,0,300         ], 1 , 1,
    0, 270, 0)
inlet_dmd4_9 = support.extract_inlet(grid1,  [1161,0,44970/137   ], 1 , 1,
    0, 270, 0)
inlet_dmd4_10 = support.extract_inlet(grid1, [1161,0,48840/137   ], 1 , 1,
    0, 270, 0)

data_list_dmd4_6 , x_list , y_list , z_list   = plot.extract_inlet_data('norm
    ', Peak_dmd4, inlet_dmd4_6 , 0)
data_list_dmd4_7 , x_list , y_list , z_list   = plot.extract_inlet_data('norm
    ', Peak_dmd4, inlet_dmd4_7 , 0)
data_list_dmd4_8 , x_list , y_list , z_list   = plot.extract_inlet_data('norm
    ', Peak_dmd4, inlet_dmd4_8 , 0)
data_list_dmd4_9 , x_list , y_list , z_list   = plot.extract_inlet_data('norm
    ', Peak_dmd4, inlet_dmd4_9 , 0)
```

```
data_list_dmd4_10 , x_list , y_list , z_list  = plot.extract_inlet_data('norm
    ', Peak_dmd4, inlet_dmd4_10 , 0)

dmd4_p6  = np.sum(data_list_dmd4_6)/len(data_list_dmd4_6)
dmd4_p7  = np.sum(data_list_dmd4_7)/len(data_list_dmd4_7)
dmd4_p8  = np.sum(data_list_dmd4_8)/len(data_list_dmd4_8)
dmd4_p9  = np.sum(data_list_dmd4_9)/len(data_list_dmd4_9)
dmd4_p10 = np.sum(data_list_dmd4_10)/len(data_list_dmd4_10)


print('Points: ', dmd4_p6, dmd4_p7, dmd4_p8, dmd4_p9, dmd4_p10)

REWS_dmd4_1161=dmd4_p6*(0.1142)+dmd4_p7*(0.2475)+dmd4_p8*(0.2766)+dmd4_p9
    *(0.2475)+dmd4_p10*(0.1142)

print('REWS: ',REWS_dmd4_1161)
```

Solberg, I. & Wassertheurer, H.

DMD as a CFD substitute for wind parks

NTNU
Kunnskap for en bedre verden

SINTEF