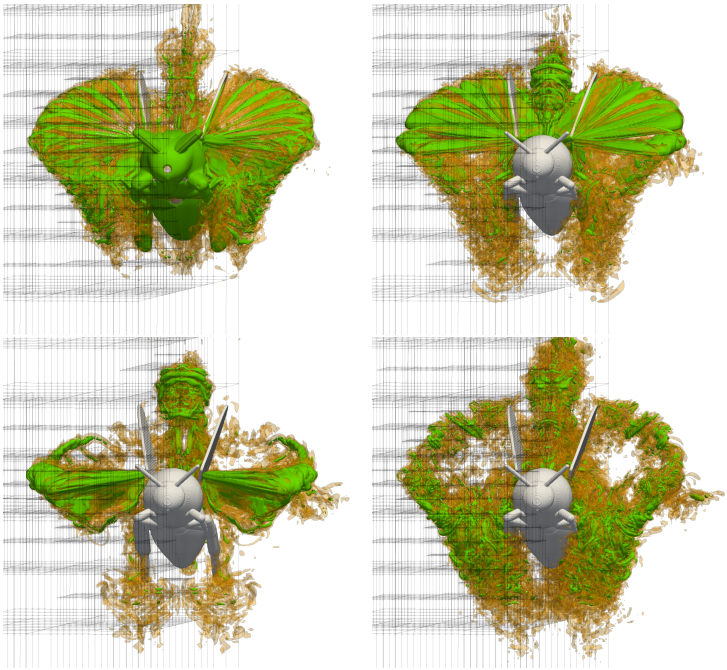
Philipp Krah

Non-linear reduced order modeling for transport dominated fluid systems



Berlin 2023

Non-linear reduced order modeling for transport dominated fluid systems

vorgelegt von
M. Sc.
Philipp Louis Krah
ORCID: 0000-0001-8982-4230

an der Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
-Dr. rer. nat.-

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	*Prof. Dr. Martin Henk*
Gutachter:	*Prof. Dr. Volker Mehrmann*
Gutachter:	*Prof. Dr. Mario Ohlberger*
Gutachter:	*Dr. Julius Reiss*

Tag der wissenschaftlichen Aussprache: *13. Dezember 2022*

Berlin 2023

Abstract

Reduced order modeling aims to approximate large and complex dynamical systems with smaller ones to reduce simulation costs in the design or control processes of these systems. Standard linear mode-based model order reduction (MOR) can fail for transport dominated fluid systems (TDFS), because the underlying transport is often inherently non-linear. However, given that, in case of TDFS, the transported quantity changes slowly with respect to the advection speed, only few degrees of freedom (DOF) are required if the system is parametrized in a reference frame that moves with the transported quantity. This thesis aims to improve MOR of TDFS by implementing well adapted non-linear coordinate transformations that take the transport of the systems into account. The first part of this thesis addresses non-linear adaptive wavelet-filtering of flow systems to adjust the computational resources to the co-moving reference frame, already when generating the data. To enable MOR with the utilized adaptive data structure, a wavelet-based adaptive version of the proper orthogonal decomposition (POD) is proposed that balances error contributions of wavelet compression and POD truncation. The second part addresses non-linear reduction methods that compensate the transport by a shift or with help of an auxiliary field parametrizing the transport. Compared to the POD, the new methods allow for efficient decomposition of TDFS with only few DOF, while providing better physical insight into the system compared to neural autoencoder networks. The presented methodology enables the decomposition of reactive systems with topologically changing front structure, such as splitting or merging reaction fronts, that pose difficulties for many non-linear reduction methods. The last part studies the ability of the non-linear reduction methods to predict new system states using intrusive and non-intrusive reduced order models. In the case of the latter, manifold Galerkin projections with a tailored hyper-reduction strategy are utilized, enabling rapid simulations of reactive flows. Given that reactive systems are considered challenging for classical MOR applications, this contribution is an essential building block for future applications.

Zusammenfassung

Um bei der Steuerung oder Optimierung von großen dynamischen Systemen Simulationskosten zu reduzieren, zielt die Modellreduktion darauf ab, diese durch kleine Systeme zu ersetzen. Herkömmliche, lineare moden-basierte Modellreduktionsmethoden sind jedoch nicht in der Lage transport-dominierte Fluidsysteme (TDFS) ausreichend zu reduzieren, da diese inherent nichtlinear sind. Dennoch lassen sich TDFS hinreichend gut reduzieren, wenn die transportierten Strukturen in einem mitbewegten Koordinatensystem approximiert werden. Das Ziel dieser Arbeit ist es daher, nichtlinearen Koordinatentransformationen zu implementieren, welche den Transport der Strömung explizit berücksichtigen. Im ersten Teil werden nichtlineare adaptive Waveletfilter auf Strömungssysteme angewendet, um die benötigten Rechenresourcen schon während der Datengeneration zu verringern. Die adaptive Datenstruktur erfordert eine gesonderte Behandlung während der weiteren Datenreduktion. Deshalb wurde eine wavelet adaptive Version der Hauptkomponentenanalyse (engl. POD) entwickelt, welche Abschneidefehler der Waveletkompression und Hauptkomponentenanalyse ausbalanciert. Der zweite Teil der Arbeit beschäftigt sich mit nichtlinearen Reduktionsmethoden, welche den Transport durch eine Verschiebungstransformation oder ein zusätzliches Hilfsfeld berücksichtigen. Im Gegensatz zu Methoden wie der POD erlaubt diese neue Methodik eine genauere Darstellung von TDFS mit weniger Freiheitsgraden. Darüber hinaus gibt die Methodik besseren Einblick in das analysierte System als herkömmliche nichtlineare Ansätze wie neuronale Autoencoder Netzwerke. Als besonderen Beitrag kann die niedrig dimensionale Zerlegung von reaktiven Strömungen mit topologischen Änderungen der Frontstruktur gesehen werden, da diese Zerlegungen als besonders anspruchsvoll gelten. Im letzten Teil der Arbeit werden die niedrig dimensionalen und nichtlinearen Beschreibungen der TDFS im Hinblick auf ihre Effizienz bei der Vorhersage neuer Systemzustände getestet. Dabei werden intrusive und nicht-intrusive Methoden verwendet. Für letztere wird eine spezielle Hyperreduktionsstrategie eingeführt, die speziell auf reaktive Strömungen mit komplexen Frontstrukturen zugeschnitten ist.

Acknowledgement

First of all, I would like to express my most sincere gratitude to Prof. Dr. Julius Reiss and Volker Mehrmann for their continuous support, for their patience, motivation and sharing of expertise. Throughout my PhD, I was often pessimistic about my research, but the clarity of ideas and optimism in visions they shared helped me to overcome many obstacles. I could not have undertaken this journey without Dr. Thomas Engels. He not only gave me valuable advice concerning wavelet-adaptation techniques, software development and scientific writing, but, above all, supported me as a mentor and friend. I would also like to express my deepest appreciation to Prof. Dr. Kai Schneider for inviting me to Marseille and giving me valuable advice during our collaboration. Along the same line, I would thank Dr.-Ing. Habil Mathias Lemke for valuable discussions and advice during my work at the institute of fluid mechanics.

My research would not have been possible without the IT support of Christian Westphal and Lars Oergel. Therefore, I would like to thank them for their kind support. Furthermore, I would like to acknowledge the German Research Foundation (DFG) for funding my research within the Research Training Group GRK 2433.

I also want to thank my former work colleagues and friends Dr. Laurent Bernier, Shubhaditya Burela, Elias Büchner, Gabriele Camerlengo, Sophie Knechtel, Dr. Leon Sallandt, Yannick Schubert, Paul Schwarz, Dr. Lewin Stein, Mario Sroka at the institute of fluid mechanics and my colleagues Ines Ahrens, Hannes Dänschel, Marine Froidevaux, Dorothea Hinsen, Riccardo Morandin, Philipp Schulze and Paul Schwerdtner at the institute of numerical analysis for valuable discussions during lunch, hiking, climbing or diving experiences.

Lastly, I would like to mention my family and friends. I know that during my PhD I was staring at a computer screen 90% of the time, which blocked most of my human interactions. Therefore, I want to thank you for giving me the best 10% of human interactions I possibly could have wished for. In particular, I want to thank my close family

"Mutsch", "Papsch", "Oma Gitsch", "Opscht" and "Elli" for their support throughout my studies and whole life. Last but not least, I would be remiss not to mention Iris Wohnsiedler, for being there for me in good and bad times.

Contents

Abbreviations

ACM Artificial Compressibility Method.

ADM Alternating Direction Method.

AE Autoencoder.

ARD Advection-Reaction-Diffusion.

CDF Cohen-Daubechies-Feauveau.

CFD Computational Fluid Dynamics.

CPU Central Processing Unit.

DOF Degrees of Freedom.

ECSW Energy-Conserving Sampling and Weighting.

FMM Fast Marching Method.

FOM Full Order Model.

FTR Front Transport Reduction.

IBVP Initial Boundary Value Problem.

ICN Incompressible Navier-Stokes.

MOR Model Order Reduction.

MR Multiresolution.

NN Neural Network.

ODE Ordinary Differential Equation.

PDE Partial Differential Equation.

POD Proper Orthogonal Decomposition.

RHS Right Hand Side.

RID Reduced Integration Domain.

ROM Reduced Order Model.

rPCA Robust Principal Component Analysis.

rSVD Randomized Singular Value Decomposition.

sPOD Shifted Proper Orthogonal Decomposition.

SVD Singular Value Decomposition.

TDFS Transport Dominated Fluid Systems.

TV Total Variation.

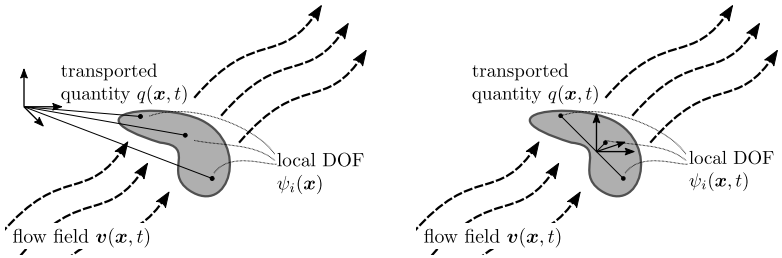
wPOD Wavelet Adaptive Proper Orthogonal Decomposition.

1 Introduction

Modern mathematical models of real-life fluid systems yield large dynamical systems that are expensive to simulate. In order to reduce the simulation costs, *model order reduction* (MOR) aims to approximate these systems with smaller ones, while sacrificing little accuracy. Unfortunately, standard mode-based model order reduction often fails for *transport dominated fluid systems* (TDFS), such as propagating flame kernels, fluid-structure interaction of moving objects, and acoustic or moving shock waves. The goal of this thesis is to improve and develop new techniques to address these shortcomings using non-linear reduction methods.

In the design and control of flow machines one is often interested in studying the evolution of fluid dynamic quantities $q(\boldsymbol{x}, t; \boldsymbol{\mu})$, such as pressure, velocity or density, over a range of different parameters $\boldsymbol{\mu}$ in space $\boldsymbol{x}$ and time t. Examples of these parameters can be viscosity of the fluid, reaction rate of burning fuel quantity, parameters describing the profile of an airfoil or the motion of a moving object immersed in a flow field. In *computational fluid dynamics* (CFD), these systems are discretized *partial differential equations* (PDEs) describing the conservation laws of fluid motion, specifically, conservation of mass, linear momentum, species and energy, which typically require billions of *degrees of freedom* (DOF). Simulating these systems is costly since computational resources scale with the number of unknowns (DOF). In the case of TDFS, particularly high dimensional systems are required since small scale quantities, described locally, are transported in time by an underlying velocity field $\boldsymbol{v}(\boldsymbol{x}, t)$, yielding a large number of DOF (see illustration in Fig. 1.1a). Therefore, model order reduction is inevitable

to reduce the number of unknowns in the system.



(a) time-independent basis **(b)** time-dependent basis

Figure 1.1: Quantity $q(\boldsymbol{x},t)$ of a TDFS, which is represented by a local time-independent basis $\psi_i(\boldsymbol{x})$ (a) and local time-dependent basis $\psi_i(\boldsymbol{x},t)$ (b). The quantity is transported by an underlying flow field $\boldsymbol{v}(\boldsymbol{x},t)$ indicated by the arrows.

Most commonly, parametric model order reduction is set up in a two-stage *offline-online* procedure (for a review see [4]). In the *offline* stage, the equations of motion are simulated for a small number of parameters. These simulations are costly because the discretized equations, referred to as *full order model* (FOM), are represented in a high dimensional space. Assuming that the gathered FOM-data represents the underlying dynamics, it can be used to generate a reduced mapping that represents the data on a low dimensional reduced system. In the second stage, the so-called *online stage*, the reduced description can be used to approximate new system states of the FOM over a range of different parameters. Since the online dynamics are evaluated only for the reduced set of variables, this can be done rapidly. The reduced set of equations is referred to as *reduced order model* (ROM).

For the full and reduced order model, the efficiency of the numerical simulation crucially depends on the chosen representation. In the simplest case, one chooses a representation in the offline and online stage, in which the degrees of freedom are represented in a predefined time-independent basis $\{\psi_i(\boldsymbol{x})\}_{i=1,\dots,\mathrm{DOF}}$ to describe the quantity of interest

$$q(\boldsymbol{x},t;\boldsymbol{\mu}) \approx \sum_{i=1}^{\mathrm{DOF}} \hat{q}_i(t;\boldsymbol{\mu})\psi_i(\boldsymbol{x})\,. \tag{1.1}$$

For example, in the offline stage, the FOM may be represented in a finite element basis $\psi_i \in V^{\mathrm{FEM}}$, using so called *shape functions* [112] or a spectral basis $\psi_n(\boldsymbol{x}) \sim e^{i\boldsymbol{x}\cdot\boldsymbol{k}_n}$ [20] if the domain is periodic. Similar approaches hold for the reduced model that is usually constructed with the *proper orthogonal decomposition* (POD), which was introduced by Sirovich in [127]. In contrast to the basis used for simulating the FOM, the POD-basis $\psi_i \in V^{\mathrm{POD}}$ is data-dependent, as it is usually computed by a *singular value decomposition* (SVD) of snapshots collected in the offline stage. However, since the reduced representation of the POD-ROM needs much fewer degrees of freedom $\sim \mathcal{O}(10)$ than the FOM $\sim \mathcal{O}(10^6)$, it enables efficient simulations. Since time and parameter dependencies are treated in a similar way, we omit the $\boldsymbol{\mu}$-dependency in the following.

Unfortunately, if transport of jumps and kinks dominate the dynamics, which is often the case in fluid flows, the ansatz in Eq. (1.1) to seperate space from time and parameters performs poorly, since many degrees of freedom are needed to describe the system. For reduced order models, this was numerically investigated for reactive flows in [74] and theoretically quantified with help of the Kolmogorov n-width in [108, 63]. The development of efficient representations for TDFS is the main goal of this thesis. In the subsequent sections, we will discuss the state of current research and its limitations, which motivate the contributions of this thesis.

1.1 State of the Art

As the quantity of interest is transported or shifted by the underlying fluid motion, it is necessary to account for the transport within the description of the flow quantity. The reference frame or coordinate system moving with the flow quantity, as illustrated in Fig. 1.1b, is called *co-moving* or *Lagrangian frame of reference*. For optimal efficiency, the basis elements need to be adapted to the co-moving reference frame over time. This affects both the simulation and representation of the FOM and ROM systems, which is addressed in the following subsections.

Adaptive Methods for Simulation of TDFS

Simulating transport dominated quantities efficiently requires a basis in a co-moving reference frame. Unfortunately, in the most general case, it is not possible to set up a co-moving basis a priori without physical or engineering insights into the system. Picture a complex reacting flame front that splits and merges within time or a deflagration to detonation transition that suddenly changes the propagation velocity from sub- to supersonic inside a combustion chamber. One common solution to this is to use highly resolved equidistant or adaptive meshes. Here, on the one hand, adaptation is advantageous for transport dominated systems, because it distributes computational efforts to places where relevant information is located. On the other hand, adaptivity comes with additional costs for data management and processing. Thus, for the same amount of data, adaptive methods are more complex and computationally demanding compared to non-adaptive equivalents. For example, [31] claim computational overheads of up to 31 percent per iteration and DOF, while the overall computational efforts concerning memory and CPU-time are reduced. From a point of view of the underlying structure of representation, one may divide the current adaptive methods loosely into two groups:

1) *Eulerian methods*, like most adaptive multiresolution methods (for a review see [36, 103]) or adaptive mesh refinement (for a review see [106]), are based on an Eulerian description of the flow. These methods usually rely on a high dimensional representation, similar to Eq. (1.1):

$$q(\boldsymbol{x}, t) \approx \sum_{i=1}^{\mathrm{DOF}} \mathcal{S}_\epsilon(\hat{q}_i(t))\psi_i(\boldsymbol{x})\,, \tag{1.2}$$

where a local error estimator $\mathcal{S}_\epsilon$ is controlled by a threshold ϵ that determines whether a basis element needs to be taken into account. Therefore, the number of possible DOF is large but the full state space is usually only partially exploited by the numerical scheme. In CFD, adaptive multiresolution or mesh refinement methods have become a popular tool. The idea is to refine the grid in areas where it is required by the precision of the solution and coarsen it whenever possible. This guarantees scalability and versatility with controlled errors. Eulerian methods have been successfully applied to various fluid dynamical

problems like combustion systems [30], compressible flows [116], incompressible flows with moving geometries [46], reactive and non-reactive Euler equations with moving boundaries [71], shock and detonation waves [32, 93].

2) In contrast, *Lagrangian methods* like the characteristic mapping method [99], Lagrangian particle-wavelet method [7] or moving mesh method (see review [75]) build on the idea of the Lagrangian description of the flow:

$$q(\boldsymbol{x},t) \approx \sum_{i=1}^{\mathrm{DOF}} \hat{q}_i(t)\psi_i(\boldsymbol{X}(\boldsymbol{x},t)), \tag{1.3}$$

where the underlying mesh $\boldsymbol{X}$ (represented by Eq. (1.1)) is transported by the flow. For example, the characteristic map $\boldsymbol{X}(\cdot,t)$ evolves the initial position $\boldsymbol{x}_0$ of a flow quantity along characteristic curves that are determined by the underlying velocity field $\boldsymbol{v}(\boldsymbol{x},t)$. The method assumes divergence-free velocity fields and is therefore only applicable to incompressible flows [143]. Another possibility is to evolve q by virtual particle positions that are carried by the flow and interpolated back on a multiresolution mesh [7] or the moving mesh method, where the Lagrangian frame is the solution of an optimization problem that includes error estimates and geometric considerations [75].

Comparing the different methods and representations regarding their efficiency is a complicated task. Multiple criteria, such as data-compression, conservation of physical properties, approximation error vs CPU-time, scalability and many others have to be considered and weighted against each other, depending on the problem at hand. However, since Lagrangian methods are often difficult to treat in a generic context, most fluid systems are studied using adaptive multiresolution and mesh refinement. For two-dimensional (2D) Euler equations, a quantitative comparison of both methods can be found in [31]. This comparison shows that adaptive multiresolution methods benefit from the sparse representation using wavelets, regarding data compression and overall CPU-time.

Non-linear Dimension Reduction Methods for TDFS

To obtain information about the fluid system, data are generated by simulating the FOM and then stored as snapshots of the flow at different time or parameter instances. The reduced order model is built from the snapshot data using special dimension reduction methods. These methods aim to approximate the flow field with only a few variables by exploiting the intrinsic structure or symmetries of the data. For one-dimensional (1D) transport, this intrinsic structure is usually one or multiple parameter or time-dependent shifts $x \mapsto x - \Delta_k(t)$, for example caused by expanding shock waves, acoustic waves or parameter-time-dependent moving structures in the flow.

To account for transport, the authors [115, 117, 12, 141] use shift transformations $\mathcal{T}^{-\Delta_k}\colon q(x + \Delta_k(t), t) \mapsto q(x, t)$ in order to align the data onto a co-moving reference frame. In this reference frame, the wave is stationary and can be described by few spatial basis functions determined with help of the POD. For example in [115], the *shifted proper orthogonal decomposition* (sPOD) was introduced, describing the flow by a superposition of co-moving fields $\{q^k(x,t)\}_{k=1,\dots,f}$:

$$q(x,t) \approx \sum_{k=1}^{f} \mathcal{T}^{\Delta_k} q^k(x,t) \tag{1.4}$$

$$\text{where} \quad q^k(x,t) = \sum_{i=1}^{r_k} \hat{q}_i^k(t)\psi_i^k(x)\,, \tag{1.5}$$

using a shift transformation. It builds on the idea that traveling waves or moving localized structures q^k can be perfectly described by their wave profile and a time-dependent transformation, usually a shift operation. Since the wave profile is only slowly changing over time, the resulting co-moving structures are of low-rank and therefore the total number of DOF $= \sum_{k=1}^{f} r_k$ is small. A similar representation was used in [117] building on the idea of symmetry reduction [120, 119]. In fact, [10] showed that the sPOD and the symmetry reduction framework [120, 9] are equivalent, if for a single reference frame approximation ($f = 1$) the continuous differentiable transformations form a group that commutes with the right hand side of the PDE. Under these assumptions, symmetry reduction and sPOD use the symmetry

group of translation to transform the data into the Lagrangian frame of reference. Therefore, Eq. (1.4) can be seen as a reduced version of Eq. (1.3). Similarly, other authors use more general mappings [104] to align steady-state parameter dependent shocks [104] or domains [107, 79].

Another common dimension reduction method uses *artificial neural networks*. Although the specific implementations differ, most authors [52, 89, 53, 81] rely on a so-called *autoencoder* (AE) structure. This structure is a composition of affine linear and non-linear functions that squeezes the input data through a small informational bottleneck, which is thereafter mapped back to its input dimension by a decoder. After the network has been trained to approximate its input data, the decoder can be used for model order reduction. However, in contrast to the methods outlined above, AE based dimension reduction does not make explicit use of the underlying transport. It is therefore more general in the sense that transport can also be handled where no one-to-one mapping exists to align the transported quantities. Examples are systems that feature topological changes like splitting and merging flame fronts or multi-phase flows. Unfortunately, the resulting AE descriptions lack structural insights and interpretability. For example, the POD representation Eq. (1.1) attributes the most relevant structures of the flow to spatial basis functions, called modes. These modes are L_2 optimal in the sense that they describe the data better than any other representation with the same number of basis functions. Furthermore, they enable rich interpretation in fluid dynamics in terms of coherent structures [72]. In contrast, it is not clear which structures or features are identified by an AE. Existing feature importance interpretation methods are usually based on first-order linear approximations around a given input. However, they may fail if the inputs are perturbed [58, 60, 132].

Reduced Order TDFS

While non-linear reduced order mappings enable the efficient reduction of TDFS, they pose additional challenges when attempting to predict new system states. New methods addressing this have been developed, which can be categorized as intrusive [89, 10] or non-intrusive [53, 52,

87, 123, 54, 111] reduced order models. *Intrusive* predictions are based on the initial system of equations, whereas *non-intrusive* models are equation-free. Therefore, non-intrusive models are purely data-driven and often based on additional model assumptions, such as smoothness in the reduced parameter space or conservation of physical properties.

Intrusive models project the original equation system on the tangent space of the manifold created by the reduced map, which is non-linear for transport dominated systems. These projections are called *manifold Galerkin*-projections. They were first implemented for neural networks in [89] and later for dynamical transformed modes in [10]. However, projection-based non-linear systems are not online efficient, since they require the evaluation of the dynamics of the full space. Therefore, special hyper-reduction schemes are required to gain speedups in the resulting ROM. Examples of these schemes are the extended Energy Conserving Sampling and Weighting (ESCW) scheme proposed by [78], the gappy-POD based Gauss–Newton with approximated tensors (GNAT) procedure [21] first introduced for non-linear manifolds in [81] or the shifted discrete empirical interpolation (DEIM) algorithm in [11]. The idea of all of these methods is to evaluate the non-linear dynamics of the underlying system for a small number of points to determine the evolution of the parameters in the reduced space.

In contrast, non-intrusive models do not need to evaluate the system from which they originate. However, their applicability is problem dependent. For example, the Fourier-Koopman framework [87] proposes an approximation of the reduced dynamics assuming that the model is quasi-periodic. Other methods, such as sparse identification of non-linear dynamics (SINDy) [54, 111], assume that the reduced dynamics can be approximated by a sparse dictionary of operators or a linear system in case of the dynamic mode decomposition (DMD) [123].

1.2 Contributions

This thesis contributes the following three developments:

1. Adaptive multiresolution methods for simulating and reducing transport dominated fluid systems,

2. Transport compensation for efficient reduced order models, and
3. Time-parameter predictions for non-linear reduced order models .

The individual contributions to these fields are discussed in the following:

1.) Compared to other methods, adaptive multiresolution schemes benefit from their sparse representation of the data using wavelets. Here, the nature of wavelets combined with multiresolution analysis [96] allows for the identification and compression of localized moving structures, while keeping control over introduced compression errors. The novelty of the presented results lies in that these properties are utilized in the creation of flow data and whilst reducing the dimension using POD. The basic adaptation method grounds on the idea of [37] and the recent works [151],[46]. Here, a block-based adaptive grid is refined or coarsened using biorthogonal wavelets. The non-linear approximation [35] of the flow data due to thresholding of the wavelet coefficients yields sparse and efficient representations to reduce memory and CPU-time requirements. Moreover, the implemented framework uses a locally uniform Cartesian grid structure for each block, which allows for the comparison and application of our methods to other 2D/3D equidistant equivalents. For example, a direct comparison of the *wavelet adaptive proper orthogonal decomposition* (wPOD) and the *randomized singular value decomposition* (rSVD) will be presented. While the randomized SVD requires several passes over the data when the singular values decay slowly [68], this can be circumvented by using a sparse wavelet representation. How to balance compression errors with other errors (such as the POD-truncation error, errors originating from the utilized PDE discretization scheme) constitutes a key challenge that will be addressed in this thesis. The developed software framework called `WABBIT` is publicly available at [129] and implemented in a multi-processing MPI environment to cope with large flow data.

2.) Secondly, this thesis addresses the slow decay of the Kolmogorov n-width, which manifests in a slow decay of the approximation errors, when reducing TDFS with the POD. To improve the convergence of the POD, two different methods are used:

- **Shifted POD**: The sPOD is used to improve the convergence of

the decomposition if transports along one-dimensional paths with multiple co-moving frames are expected. Multiple gradient-based optimization algorithms for the sPOD already exist in the literature [115, 12, 114]. However, these formulations of the sPOD have disadvantages, both in practice and theory, which are addressed in the thesis. Based on the work [114], two new sPOD algorithms are presented. The first algorithm minimizes

$$\mathcal{J}_2 = \frac{1}{2} \sum_{k=1}^{f} \sum_{j=r_k+1}^{m} (\sigma_j^k)^2, \tag{1.6}$$

the sum of the squares of the truncated singular values $\sigma_j^k \geq 0$ in each co-moving frame $k = 1, \ldots, f$. The developed algorithm delivers a solution to the optimization by redistributing the truncation error in each frame exactly. This avoids the need for a constraint-projection-step since the constraint Eq. (1.4) is always satisfied. Additionally, we regularize the total variation of the time amplitudes to obtain a unique decomposition at time instances where two traveling waves intersect. The additional constraint improves the convergence of the algorithm and is a key feature for non-intrusive model order reduction. Furthermore, since the computational complexity of the algorithm is reduced to the complexity of the SVD and the total variation operator, the application to 2D flow data is enabled. However, as already pointed out by [114], the optimization based on $\mathcal{J}_2$ is not optimal, since the rank r_k of the co-moving data frame in Eq. (1.5) has to be chosen beforehand. For complicated systems, this choice is often arbitrary. For example, if more co-moving frames or modes are used in the decomposition than necessary, these additional DOF are not automatically zero. This is why the second sPOD algorithm presented in this thesis is formulated as a convex optimization problem, based on minimizing the one-norm over the set of all singular values in the co-moving frames:

$$\mathcal{J}_1 = \sum_{k=1}^{f} \sum_{j=1}^{m} \sigma_j^k \,. \tag{1.7}$$

The $\mathcal{J}_1$ objective function avoids the selection of co-moving ranks, by promoting sparsity over the set of singular values, i.e. the

DOF. Therefore, associated frames or modes that are unnecessary in the decomposition are removed during the optimization. Unfortunately, it was reported in [114] that the $\mathcal{J}_1$ optimization based on a gradient descent method converges very slowly. To tackle this problem, we use the *Alternating Direction Method of Multipliers* (ADM, for a review see [15]) with proximal operators for minimizing one-norms. Moreover, the ADM based formulation allows to generalize the sPOD decomposition to incorporate additional constraints. We will demonstrate this by adding an extra term to the decomposition, which aims to capture noise. The resulting algorithm is shown to be more robust against interpolation noise, corrupted measurements or numerical artifacts and can be seen as a shifted version of the *robust principal component analysis* (rPCA).

- **Front transport reduction**: Unfortunately, the shifted POD can only be applied if the shifts due to the underlying flow are parametrizable, easily traceable and the resulting transformations are invertible. These properties limit the applicability, especially if topological changes in the flow pattern occur. Examples of such problems are multi-phase and reactive flows, where the contour line separating two flow regions may split or merge. This problem is tackled using a more direct approach, named *front transport reduction* (FTR). The basic ansatz assumes that the original flow field can be reconstructed by a front shape function f and a level set function ϕ:

$$q(\boldsymbol{x},t) \approx f(\phi(\boldsymbol{x},t)) \quad \text{s.t.} \quad \phi(\boldsymbol{x},t) = \sum_{i=1}^{\text{DOF}} \hat{\phi}_i(t)\psi_i(\boldsymbol{x}) \tag{1.8}$$

  The level set function is used to generate a local coordinate, which parametrizes the distance to the front. In this way, a local 1D representation is applied to describe complex 2D front dynamics with merging or splitting fronts, while seeking a low-rank description of the level set field ϕ. The ability of the new ansatz is demonstrated for 2D propagating flames. The ansatz can handle merging or splitting fronts because fronts are embedded as $D-1$ dimensional levels of a D dimensional level set function. Due to the non-linearly activated linear space created by the span of

$\{\psi_k(\boldsymbol{x})\}_{k=1,\dots,\mathrm{DOF}}$, this approach shows many parallels to neural networks. It can be viewed as the decoder part of a shallow autoencoder structure. While shallow autoencoders have been used in previous studies [81], we are explicitly incorporating the underlying physical assumptions and thereby obtain interpretable results of the reduced variables.

3.) Lastly, we address time-parameter predictions to test the quality of the non-linear reduced mappings generated in 2.). In order to do so, non-intrusive and intrusive reduced order models are implemented and compared. Non-intrusive reduced order models are generated from the parameter-time coefficients that are sampled during the offline stage. In particular, we make use of the Fourier-Koopman forecasting introduced by [87]. Intrusive reduced order models are generated and tested for an *advection-reaction-diffusion* (ARD) equation with moving fronts using [89, 110]. For the specific problem of moving fronts, we obtain speedups by introducing a novel hyper-reduction scheme for these systems, based on the *reduced integration domain* (RID) method [122].

1.3 Outline

The outline of the thesis is as follows:

Chapter 2 introduces the full order model as a discretized PDE system on a block-based adaptive grid. In this chapter, the specifics of the underlying wavelet adaptation scheme are explained and the efficiency regarding CPU-time is compared to non-adaptive schemes, for two specific TDFS.

Next, dimension reduction using linear and non-linear mappings is discussed. For dimension reduction on linear subspaces, we study the wavelet adaptive version of the snapshot POD in **Chapter 3**, which is based on the adaptation algorithm presented in Chapter 2. The wavelet adaptive POD (wPOD) allows for the decomposition of large-scale flows on adaptive and non-adaptive grids. We test its ability to reduce large-scale flows for systems with slow and fast decaying POD eigenvalues and compare it to the randomized SVD regarding memory efficiency and accuracy.

Chapter 4 deals with dimension reduction on non-linear manifolds. The chapter introduces the implemented non-linear dimension reduction methods to reduce TDFS. Among state of the art AE networks, two new sPOD-algorithms and a novel methodology to reduce moving fronts with topological changes, called front transport reduction (FTR), will be presented. The application regime of the methods will be discussed and illustrated by the 2D examples introduced in Chapter 2.

To test the FTR in the online stage, we examine non-intrusive and intrusive reduced order models in **Chapter 5** for complex moving reaction fronts. A purely data-driven approach using Fourier-Koopman forecasting combined with the FTR is tested on a Bunsen-flame. Furthermore, we use manifold Galerkin projections and a tailored hyper-reduction strategy to evaluate the FTR when predicting new system states of an advection-reaction-diffusion system with topological changes.

Finally, a conclusion and outlook is presented in **Chapter 6**. The main achievements and limitations of the presented findings will be discussed and areas for further research based on these insights are outlined.

1.4 Notation

In the following, the notational conventions that are used throughout this thesis are declared.

Matrices are denoted in capital letters with straight, bold font $\mathbf{A} \in \mathbb{R}^{K\times K}$ and vectors are denoted in bold font $\boldsymbol{a} \in \mathbb{R}^K$. The transpose of a matrix $\mathbf{A}$ or vector $\boldsymbol{a}$ is denoted by $\mathbf{A}^\top$ and $\boldsymbol{a}^\top$. For multi-indexing of vector components $\boldsymbol{a} = (a_1, \dots, a_K)$ we use the following notation $a_k := a[k_1, \dots, k_d]$ for $k_l \in [0, K_l - 1]$, where $k = \sum_{l=1}^{d} \left(\prod_{m=1}^{l-1} K_m\right) k_l$ is in lexicographic order and $K := \prod_{m=1}^{d} K_m$.

Furthermore, we denote $q(\boldsymbol{x}, t, \boldsymbol{\mu}) \in \mathbb{R}$ as the solution of a scalar, evolutionary PDE that is solved in $(\boldsymbol{x}, t) \in \mathbb{D} \times \mathbb{R}^+, \mathbb{D} \subset \mathbb{R}^D$ for a predefined parameter vector $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^k$. Its ordinary differential equation (ODE) counterpart $\underline{\boldsymbol{q}}\colon \mathbb{R}^+ \times \mathcal{P} \to \mathbb{R}^M$ is defined on a discrete space that is associated with a grid $\Omega = \{\boldsymbol{x}_k \in \mathbb{D}, k = 1, \dots, M\} \subset \mathbb{D}$ of M grid points. The ODE state vector contains the spatial values of the

solution in its components:

$$\underline{\boldsymbol{q}}(t,\boldsymbol{\mu}) = \begin{pmatrix} q(\boldsymbol{x}_1,t,\boldsymbol{\mu}) \\ \vdots \\ q(\boldsymbol{x}_M,t,\boldsymbol{\mu}) \end{pmatrix} \in \mathbb{R}^M . \tag{1.9}$$

If the PDEs solution is vector-valued $\boldsymbol{q} = (q_1,\ldots,q_K)^\top : \mathbb{D}\times\mathbb{R}^+\times\mathcal{P} \to \mathbb{R}^K$ its ODE counterpart $\underline{\boldsymbol{q}}(t,\boldsymbol{\mu}) \in \mathbb{R}^{KM}$ is given by

$$\underline{\boldsymbol{q}}(t,\boldsymbol{\mu}) = \begin{pmatrix} \underline{\boldsymbol{q}}_1(t,\boldsymbol{\mu}) \\ \vdots \\ \underline{\boldsymbol{q}}_K(t,\boldsymbol{\mu}) \end{pmatrix} \text{ with } \quad \underline{\boldsymbol{q}}_k(t,\boldsymbol{\mu}) = \begin{pmatrix} q_k(\boldsymbol{x}_1,t,\boldsymbol{\mu}) \\ \vdots \\ q_k(\boldsymbol{x}_M,t,\boldsymbol{\mu}) \end{pmatrix} . \tag{1.10}$$

We define the snapshot matrix $\mathbf{Q}$ as a matrix that contains samples of a trajectory with initial condition $\underline{\boldsymbol{q}}(t_0,\boldsymbol{\mu})$ as ODE-state vectors in its columns:

$$\mathbf{Q}(\mu) = [\underline{\boldsymbol{q}}(t_0,\boldsymbol{\mu}), \underline{\boldsymbol{q}}(t_1,\boldsymbol{\mu}), \ldots, \underline{\boldsymbol{q}}(t_N,\boldsymbol{\mu})] \in \mathbb{R}^{KM\times N_t} . \tag{1.11}$$

If multiple trajectories are sampled for different parameter vectors $\{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_N\} \subset \mathcal{P}$, they are concatenated and we write:

$$\mathbf{Q} = [\mathbf{Q}(\boldsymbol{\mu}_1), \mathbf{Q}(\boldsymbol{\mu}_2), \ldots, \mathbf{Q}(\boldsymbol{\mu}_N)] \in \mathbb{R}^{KM\times N_t N} . \tag{1.12}$$

Partial derivatives in space and time are denoted by $\partial_x = \frac{\partial}{\partial x}, \partial_t = \frac{\partial}{\partial t}, \partial_{xx} = \frac{\partial^2}{\partial x^2}$ and $\dot{q} = \frac{\partial q}{\partial t}$. Moreover, for a D-dimensional cartesian coordinate system $(x_1,\ldots,x_D)$, we define the nabla operator:

$$\boldsymbol{\nabla} = \left(\frac{\partial}{\partial x_1},\ldots,\frac{\partial}{\partial x_D}\right) \tag{1.13}$$

and the Laplace operator:

$$\boldsymbol{\nabla}^2 = \frac{\partial^2}{\partial x_1^2} + \cdots + \frac{\partial^2}{\partial x_D^2} . \tag{1.14}$$

The curl of a three-dimensional vector field $\boldsymbol{v}(x_1,x_2,x_3) = v_1\boldsymbol{e}_1 + v_2\boldsymbol{e}_2 + v_3\boldsymbol{e}_3$ is represented as:

$$\boldsymbol{\nabla}\times\boldsymbol{v} = \left(\frac{\partial v_3}{\partial x_2} - \frac{\partial v_2}{\partial x_3}\right)\boldsymbol{e}_1 + \left(\frac{\partial v_1}{\partial x_3} - \frac{\partial v_3}{\partial x_1}\right)\boldsymbol{e}_2 + \left(\frac{\partial v_2}{\partial x_1} - \frac{\partial v_1}{\partial x_2}\right)\boldsymbol{e}_3 , \tag{1.15}$$

where $\{\boldsymbol{e}_1,\boldsymbol{e}_2,\boldsymbol{e}_3\}$ are the unit vectors in $\mathbb{R}^3$.

2 Simulating Transport Dominated Fluid Systems

In this chapter, we present the adaptive multiresolution method used to simulate transport dominated fluid systems (TDFS). Section 2.1 introduces the concepts of the adaptation scheme, including block-based wavelet adaptation (Section 2.1.1), safety zone concept used for time-stepping (Section 2.1.2) and the volume penalization method for embedding moving geometries (Section 2.1.4). Thereafter, we introduce the specific case studies (Section 2.2) and compare the block-based multiresolution framework for 2D and 3D transport dominated flows with spectral and equidistant computations.

2.1 Adaptive Multiresolution Framework WABBIT

We use the open-source software framework *"wavelet adaptive block-based solver for interactions with turbulence"* `WABBIT` [129], which was in parts developed and improved in the context of this thesis. The following sections are based on the publications [148, 151] and [46].

`WABBIT` is designed to solve *initial boundary value problems* (IBVP) for large-scale non-linear evolution equations:

$$\begin{cases} \partial_t q(\boldsymbol{x},t) = \mathcal{N}(q(\boldsymbol{x},t),\boldsymbol{x},t) & \text{for} \quad (\boldsymbol{x},t) \in \mathbb{D}\times\mathbb{R}^+ \\ q(\boldsymbol{x},0) = q_0(\boldsymbol{x}) & \text{for} \quad \boldsymbol{x}\in\mathbb{D} \end{cases} \tag{2.1}$$

on rectangular periodic domains $\mathbb{D} = \prod_{\alpha=1}^{D}[0, L_\alpha]$ for $D = 2, 3$ spatial dimensions (2D/3D). Complex shaped boundary conditions are usually embedded in the domain using the volume penalization method,

as explained in Section 2.1.4. For efficient storing and handling of the large-scale flow field q and the PDE *right hand side* (RHS), i. e. the evaluated differential operator $\mathcal{N}$, we use block-based wavelet adaptation, which is explained in the following. For ease of notation, we consider only scalar evolutionary PDEs on two-dimensional domains in this section. Generalizations towards three spatial dimensions and vector-valued PDEs like the incompressible Navier-Stokes equations in 3D are mentioned in the text.

2.1.1 Block-Based Wavelet Adaptation

Our adaptation algorithm is block-based, meaning that the data are stored on a block-structured grid.

Block Structured Grid

As illustrated in Fig. 2.1 for the 2D case, the computational grid Ω is composed of blocks

$$\mathcal{B}_p^j = \{(x_p^j, y_p^j) + (k_1 \Delta x^j, k_2 \Delta y^j) \mid k_1 = 0, \dots, B_x - 1\,, \quad k_2 = 0, \dots, B_y - 1\} \tag{2.2}$$

of equal size $B_x \times B_y$. The subdivision of the grid is controlled by the tree level $j = J_{\min}, J_{\min}+1, \dots, J_{\max}$. With increasing tree level $j \to j+1$ the lattice spacing of the block is divided by two, i.e. $\Delta x^j = 2^{-j} L_x/(B_x - 1)$, $\Delta y^j = 2^{-j} L_y/(B_y - 1)$. Here, $L_x, L_y > 0$ is the size of the computational domain. Gradedness of the resulting grid is enforced, meaning that adjacent blocks differ only by one tree level. The level and location of a block are encoded in a unique tree code as indicated in fig. 2.1. The computational grid Ω is the union of all blocks,

$$\Omega = \bigcup_{j=J_{\min}}^{J_{\max}} \bigcup_{p \in \Lambda^j} \mathcal{B}_p^j, \tag{2.3}$$

where Λ^j is the set of all block IDs on a given tree level j. The block IDs $p \in \Lambda^j$ are called *tree code*. The synchronization between blocks is done by using an overlapping layer covering $g > 0$ points (light gray area in Fig. 2.1). These additional points are called *ghost nodes* and

they form the *ghost node layer*. The size of the ghost node layer is adjusted according to the spatial support of the chosen wavelet. In this thesis, we use $g = 6$ for the wavelets defined in Table 2.1. If neighboring blocks are not on the same tree level, we interpolate or downsample the corresponding nodes when synchronizing the ghost nodes. Redundant nodes on the boundary of adjacent blocks always belong to the block with a higher tree level, i.e. finer lattice spacing. The block-based definition of the grid allows combining the advantages of unstructured grids with the efficiency of structured grids. While the local block structure can be tuned for optimal CPU and memory performance, the global structure is flexible to adapt to the algorithm's requirements. Furthermore, block-based multiresolution methods require less administrative overhead compared to point-based like [118], at the cost of a reduced compression rate.

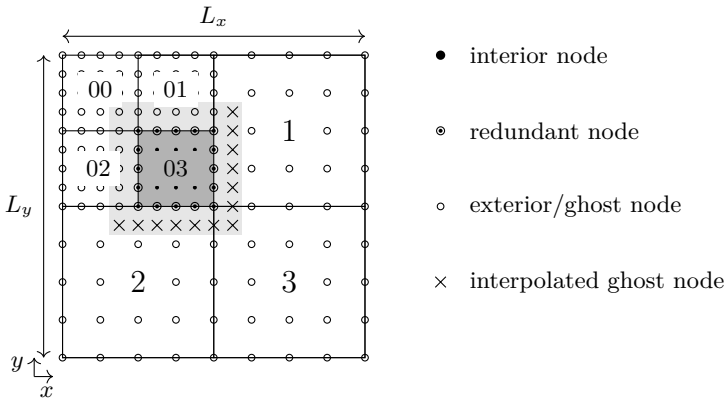


Figure 2.1: A 2D grid composed of seven blocks. A single block (dark grey) consists of an odd number of interior grid nodes $B_x = B_y = 5$. The block includes a ghost node layer (light gray) which is synchronized with neighboring blocks. The size of this layer depends on the support of the chosen wavelet. Ghost nodes are interpolated or downsampled if the levels of the neighboring blocks differ.

Refinement and Coarsening

For the wavelet adaptation scheme, here illustrated for the 2D case, we assume real-valued and continuous L^2-functions $q(x, y)$, such as the pressure or velocity component of a flow field. The function is sampled on a block-based multiresolution grid, as shown in Fig. 2.1, with maximum tree level $J_{\max}$, block size $B_x \times B_y$ and a ghost node layer of size g, needed for synchronization. The sampled values on a block $\mathcal{B}_p^j$ are denoted by:

$$x^j_{p,k_1} = x^j_p + k_1 \Delta x^j \qquad k_1 = -g, \dots, B_x + g - 1 \tag{2.4}$$

$$y^j_{p,k_2} = y^j_p + k_2 \Delta y^j \qquad k_2 = -g, \dots, B_y + g - 1 \tag{2.5}$$

$$\underline{q}^j[p, k_1, k_2] := q(x^j_{p,k_1}, y^j_{p,k_2}) \tag{2.6}$$

For a block *refinement* $j \to j + 1$ the lattice spacings are divided by two and dyadic points are added to the block by midpoint insertion, as shown in Fig. 2.2. The values at the refined blocks can be obtained by

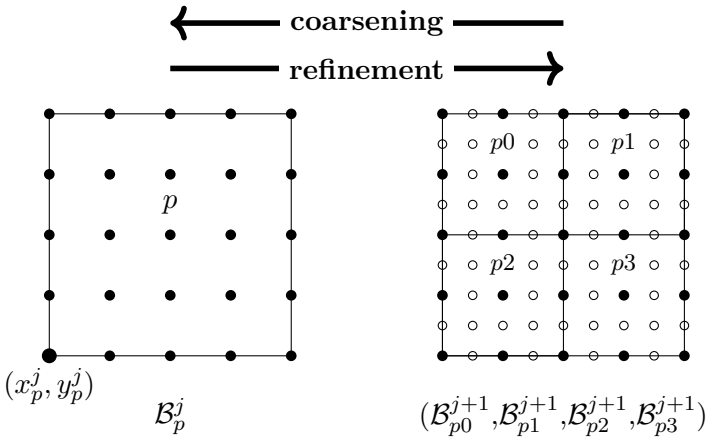


Figure 2.2: Dyadic grid refinement and coarsening of a single block. Refinement: First the block $\mathcal{B}_p^j$ is refined by midpoint insertion and then split into four new blocks $(\mathcal{B}^j_{p0}, \mathcal{B}^j_{p1}, \mathcal{B}^j_{p2}, \mathcal{B}^j_{p3})$. Coarsening: After a low pass filter is applied all midpoints (open symbols) are removed and merged into one block.

the refinement relation:

$$\underline{q}^{j+1}[p, l_1, l_2] = \sum_{k_1=-g}^{B_x+g-1} \sum_{k_2=-g}^{B_y+g-1} h_{l_1-2k_1} h_{l_2-2k_2}\, \underline{q}^{j}[p, k_1, k_2]\,, \tag{2.7}$$

where h_k denotes the weights of the one-dimensional interpolation scheme:

$$\varphi(x) = \sum_{k=-N}^{N} h_k \varphi(2x - k). \tag{2.8}$$

The procedure of recursive dyadic refinement is known as interpolatory subdivision and was first introduced by Deslauriers and Dubuc in [33, 34]. It was later shown by Donoho in [38] that the resulting interpolation or scaling functions φ can be used for constructing a multiresolution analysis. In the following, we will refer to Eq. (2.7) as the *prediction operation* $\mathcal{P}_j^{j+1} : \underline{\boldsymbol{q}}^j \mapsto \underline{\widehat{\boldsymbol{q}}}^{j+1}$ as it was introduced for point value multiresolution in [69]. Using Eq. (2.8) one can show that Eq. (2.7) is equivalent to the continuous refinement relation:

$$\widehat{q}^{(p)}(x, y) = \sum_{k_1=-g}^{B_x+g-1} \sum_{k_2=-g}^{B_y+g-1} \underline{q}^{j}[p, k_1, k_2] \varphi^{j}_{p,k_1,k_2}(x, y)\,, \tag{2.9}$$

$$\text{where} \quad \varphi^{j}_{p,k_1,k_2}(x, y) = \varphi\left(\frac{x - x^{j}_{p,k_1}}{\Delta x^j}\right) \varphi\left(\frac{y - y^{j}_{p,k_2}}{\Delta y^j}\right) \tag{2.10}$$

is the two-dimensional tensor product of one-dimensional *interpolating scaling functions* $\varphi(x)$. The refinement relation Eq. (2.9) for $j \in \mathbb{N}$ can be seen as the projection $\widehat{q}^{(p)} = \mathrm{proj}_{\mathcal{V}_j}\, q^{(p)}$ of $q \in L^2(\mathbb{D})$ onto a subspace $\mathcal{V}_j \subset L^2(\mathbb{D})$ that is spanned by the Reisz basis of scaling functions $\{\varphi^{j}_{p,k_1,k_2}\}$. The relation between the scaling functions Eq. (2.8) ensures that these spaces are nested $\mathcal{V}_j \subset \mathcal{V}_{j+1}$ and $\bigcup_{j>0} \mathcal{V}_{j+1}$ is dense in L^2. Thus, the sequence $\mathcal{V}_j$ forms a *multiresolution analysis*.

When a block is *coarsened*, the tree level is reduced by one: $j + 1 \to j$ and every second grid point is removed. The values at the coarser level are obtained by the coarsening relation:

$$\underline{q}^{j-1}[p, l_1, l_2] = \sum_{k_1=-g}^{B_x+g-1} \sum_{k_2=-g}^{B_y+g-1} \tilde{h}_{2l_1-k_1} \tilde{h}_{2l_2-k_2}\, \underline{q}^{j}[p, k_1, k_2] \tag{2.11}$$

In the notation of Harten [69] coarsening is called *decimation* and Eq. (2.11) is denoted by $\mathcal{D}_j^{j-1}: \underline{\boldsymbol{q}}^j \mapsto \underline{\boldsymbol{q}}^{j-1}$ in the following. After decimation, the block will be merged with its neighboring blocks, as shown in Fig. 2.2. Similar to the continuous refinement relation Eq. (2.9) there is a continuous counterpart for coarsening: the *dual scaling function* $\tilde{\varphi}$, which satisfies

$$\tilde{\varphi}(x) = \sum_{k=-N}^{N} \tilde{h}_k \tilde{\varphi}(2x-k). \tag{2.12}$$

In the same way as the scaling functions φ^j_{p,k_1,k_2}, the dual scaling functions $\tilde{\varphi}^j_{p,k_1,k_2}$ form a basis $\tilde{\mathcal{V}}_j$, that generates a dual multiresolution analysis. The basis functions $\varphi^j_{p,k_1,k_2} \in \mathcal{V}_j$, $\tilde{\varphi}^j_{p,k_1,k_2} \in \tilde{\mathcal{V}}_j$ are biorthogonal. The filter coefficients that define the primal and dual scaling functions h_k, $\tilde{h}_k$ are listed in Table 2.1.

continuous	k	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
$\tilde{\varphi}$	$\tilde{h}_k$	$-\frac{1}{256}$	0	$\frac{9}{128}$	$-\frac{1}{16}$	$-\frac{63}{256}$	$\frac{9}{16}$	$\frac{87}{64}$	$\frac{9}{16}$	$-\frac{63}{256}$	$-\frac{1}{16}$	$\frac{9}{128}$	0	$-\frac{1}{256}$
φ	h_k				$-\frac{1}{16}$	0	$\frac{9}{16}$	1	$\frac{9}{16}$	0	$-\frac{1}{16}$			

Table 2.1: Filter coefficients h_k and dual filter coefficients $\tilde{h}_k$ of the Cohen-Daubechies–Feauveau (CDF 4,4) wavelets applied in the prediction/restriction operation [46, Appendix B].

Computing Wavelet Coefficients and Adaptation Criterion

With the definition in Eqs. (2.7) and (2.11) we have introduced a biorthogonal multiresolution basis $\{\tilde{\varphi}^j_{p,k_1,k_2}, \varphi^j_{p,k_1,k_2}\}$, which can approximate any continuous function $q \in L^2(\mathbb{D})$ arbitrarily close [26, 96]. This concept first introduced by S. Mallat in [96] is the main property of a multiresolution analysis and can be used to relate and compare samples at different resolutions, i.e. different scales. As illustrated in Fig. 2.3, the difference between two consecutive approximations can be

represented by *wavelets* $\psi^j_{\mu,p,k_1,k_2} \in \mathcal{W}_{j-1} \subset \mathcal{V}_j$:

$$\Delta^{j-1}(x,y) = \mathrm{proj}_{\mathcal{V}_j}\, q^{(p)}(x,y) - \mathrm{proj}_{\mathcal{V}_{j-1}}\, q^{(p)}(x,y) \tag{2.13}$$

$$= \sum_{k_1,k_2} \sum_{\mu=1}^{3} d^{j-1}_{\mu}[p,k_1,k_2]\psi^{j-1}_{\mu,p,k_1,k_2}(x,y)\,. \tag{2.14}$$

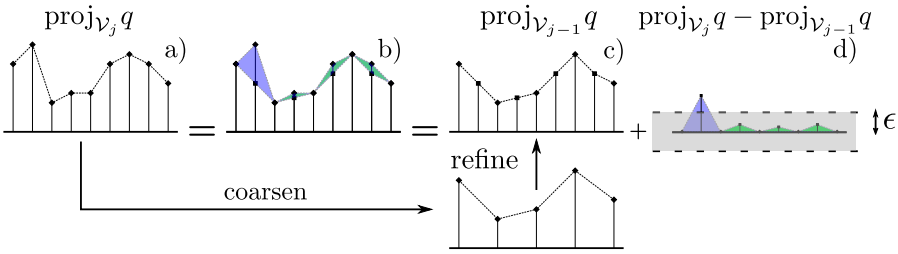


Figure 2.3: Illustration of linear interpolating wavelets. The linear approximation of the function q at level j (a,b) can be represented by linear interpolation at a coarser level (c) and additional details (d) that are the difference between two consecutive approximations. Depending on the absolute value of the details and a given threshold ϵ (grey area in d)), one may mark them as significant (blue) or insignificant (green).

The wavelet coefficients, known as *wavelet details*, are calculated by

$$d^{j-1}_{\mu}[p,k_1,k_2] = \begin{cases} \underline{\Delta}^{j-1}[p,2k_1+1,2k_2] & \text{for} \quad \mu=1 \\ \underline{\Delta}^{j-1}[p,2k_1,2k_2+1] & \text{for} \quad \mu=2 \\ \underline{\Delta}^{j-1}[p,2k_1+1,2k_2+1] & \text{for} \quad \mu=3\,, \end{cases} \tag{2.15}$$

$$\text{where} \quad \underline{\boldsymbol{\Delta}}^{j-1} = \underline{\boldsymbol{q}}^j - \mathcal{P}^j_{j-1}\mathcal{D}^{j-1}_j \underline{\boldsymbol{q}}^j\,. \tag{2.16}$$

Here, $\mathcal{P}^j_{j-1}, \mathcal{D}^{j-1}_j$ are computed using the prediction and coarsening relation Eqs. (2.7) and (2.11).

The wavelets span the basis of $\mathcal{W}_{j-1} \subset \mathcal{V}_j$, that is the complement of $\mathcal{V}_{j-1}$ in $\mathcal{V}_j$, i.e. $\mathcal{V}_j = \mathcal{V}_{j-1} \oplus \mathcal{W}_{j-1}$. Consequently, the details complement the scaling function coefficients at the odd lattice sites, as illustrated for two spatial dimensions in Fig. 2.4. Similarly to the scaling

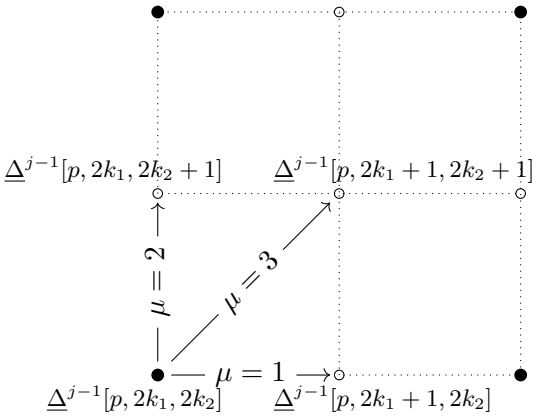


Figure 2.4: Illustration of the wavelet details $d_\mu^{j-1}[p, k_1, k_2]$ defined in Eq. (2.15) for $D = 2$ spatial dimensions. The details are represented by the difference $\underline{\Delta}^{j-1}$ at the odd lattice sites (marker ∘).

functions, the *dual wavelets* $\tilde{\psi}^j_{\mu,p,k_1,k_2}$ are biorthogonal to the wavelets and complement $\tilde{\mathcal{V}}_{j-1}$ in $\tilde{\mathcal{V}}_j$. The exact representation of ψ^j_{μ,p,k_1,k_2} and $\tilde{\psi}^j_{\mu,p,k_1,k_2}$ is not further needed to understand the adaptation criterion. However, for the interested reader, we will give a detailed description of the wavelet representation in the next subsection.

As shown in one space dimension by Unser in [135] for biorthogonal wavelets the difference Δ^j between two consecutive approximations is bounded for sufficiently smooth L^2 functions q. The bound depends on the local regularity of q and the order N (here $N = 4$) of the scaling function:

$$\left\|\Delta^j\right\|_2 = \left\|q - \mathrm{proj}_{\mathcal{V}_j}(q)\right\|_2 \leq C_{\varphi,\tilde{\varphi}}(\Delta x^j)^N \left\|\frac{\mathrm{d}^N}{\mathrm{d}x^N} q\right\|_2 , \tag{2.17}$$

where $\Delta x^j \sim 2^{-j}$ is the lattice size, $C_{\varphi,\tilde{\varphi}}$ is a constant independent of q. This result can be understood as an interpolation error of the data $\mathcal{D}_j^{j-1}\underline{\boldsymbol{q}}^j$. From Eq. (2.17) we can thus conclude: since the approximation error of the interpolation scheme depends on the smoothness of the sampled function and the lattice spacing on the block, we can

increase the local lattice spacing of the block for blocks where the function is smooth and keep the fine scales only for blocks where $q^{(p)}$ is not smooth. This is achieved by coarsening the block, as decreasing the tree level j increases the lattice spacing. For a desired approximation error we therefore define the *wavelet threshold* $\epsilon \geq 0$ together with the *coarsening indicator* $i_\epsilon(\mathcal{B}_p^j)$:

$$i_\epsilon(\mathcal{B}_p^j) := \begin{cases} 1 & \text{if } \max_{\substack{\mu=1,2,3 \\ k_1=0,\dots,B_x-1 \\ k_2=0,\dots,B_y-1}} \mid d_\mu^{j-1}[p,k_1,k_2] \mid < \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{2.18}$$

which marks the block for coarsening. The described adaptation strategy is illustrated in Fig. 2.3. In the green regions, the details are insignificant and thus can be removed. The blue regions represent significant details that are kept. However, for our block-based adaptation, we group the detail coefficients in blocks, i.e. all details on the block are kept if at least one detail is significant.

For vector-valued quantities $\boldsymbol{q} = (q_1, \dots, q_K)$, a block will be coarsened only if all components indicate coarsening. The pseudo-code in Algorithm 1 outlines the wavelet adaptation algorithm for vector-valued quantities. For the sake of completeness, we will put this algorithm in relation to the underlying wavelet representation.

Wavelet Representation in the Continuous Setting

For the completeness of this manuscript, we give a detailed description of the underlying wavelet representation in a concise way. For the interested reader we recommend Ref. [131] for a succinct introduction to biorthogonal wavelets and to Refs. [97, 38, 33, 34] for a more detailed description.

Assuming that we have sampled a continuous function $q \in L^2(\mathbb{D})$ inside a domain $\mathbb{D} \subset \mathbb{R}^2$ on an equidistant grid corresponding to refinement level $J_{\max}$, we can block-decompose it in terms of Eq. (2.6). By choosing $j = J_{\max}$ in Eq. (2.9) and summing over all blocks p, we can thus represent q using a basis of dilated and translated scaling functions

Algorithm 1 Wavelet adaptation

Require: $\boldsymbol{q}\colon \mathbb{D} \to \mathbb{R}^K$ where $\mathbb{D} \subset \mathbb{R}^D$ defined in Eq. (2.3) and $D \in \{2,3\}$, $K \in \mathbb{N}$

Require: threshold $\epsilon \geq 0$ and minimal and maximum tree level $J_{\min}, J_{\max}$

function ADAPT($\boldsymbol{q}, \epsilon, J_{\min}, J_{\max}$)
  set $N_{\text{blocks}} = 0$
  while $\left|\bigcup_j \Lambda^j\right| \neq N_{\text{blocks}}$ **do**
  ▷ *coarsening is stopped if the number of blocks has not changed*
    Synchronize all ghost layers of the blocks
    Count the number of active blocks N_{blocks}
    for all $p \in \bigcup_j \Lambda^j$ **do** ▷ *loop over all active treecodes*
      Compute the refinement indicator of the block:
      1) normalize every state vector component:
      $(\boldsymbol{q}^{(p)})_i \leftarrow (\boldsymbol{q}^{(p)})_i / \left\|(\boldsymbol{q}^{(p)})_i\right\|$, $i = 1, \ldots, K$
      2) compute $\widehat{\boldsymbol{q}}_p^j$ using Eqs. (2.7) and (2.11) in a restriction and prediction step
      3) compute the detail coefficients of all state vector components with Eq. (2.15)
      if $i_\epsilon(\mathcal{B}_p) = 1$ and $J_{\min} \leq j - 1$ **then**
        tag the block $\mathcal{B}_p$ for coarsening
      end if
    end for
    Remove coarsening tag from blocks,
    if coarsening would result in a non graded grid.
    Coarse all blocks marked for coarsening
  end while
  Balance the load between processors
  return $\boldsymbol{q}^\epsilon$ ▷ *wavelet filtered field*
end function

$\{\varphi^j_{p,k_1,k_2}\}$:

$$q(x,y) = \sum_{p\in\Lambda^{J_\max}} q^{(p)}(x,y) = \sum_{\lambda\in\overline{\Lambda}^{J_\max}} c^{J_\max}_\lambda \varphi^{J_\max}_\lambda(x,y)\,. \tag{2.19}$$

Here we have introduced a multi-index $\lambda = (p,k_1,k_2) \in \overline{\Lambda}^j := \Lambda^j \times \{-g,\dots,B_x+g-1\} \times \{-g,\dots,B_y+g-1\}$ for ease of notation. With this notation, we can rewrite Eq. (2.19) in a wavelet series

$$q(x,y) = \sum_{\lambda\in\overline{\Lambda}^{J_\min}} c^{J_\min}_\lambda \varphi^{J_\min}_\lambda(x,y) + \sum_{j=J_\min}^{J_\max-1} \sum_{\lambda\in\overline{\Lambda}^j} \sum_{\mu=1}^{3} d^j_{\mu\lambda}\psi^j_{\mu,\lambda}(x,y)\,, \tag{2.20}$$

where the interpolating scaling basis $\{\varphi^{J_\min}_\lambda\}_{\lambda\in\overline{\Lambda}^{J_\min}}$ approximates q at the coarsest scale $J_\min$ and the wavelet basis $\{\psi^j_{\mu,\lambda}\}_{\mu=1,2,3,\lambda\in\overline{\Lambda}^{j\geq J_\min}}$ contains all the additional information necessary to construct q.

In the biorthogonal setting we have biorthogonal scaling functions:

$$\varphi^j_\lambda(x,y) = \varphi\left(\frac{x-x^j_\lambda}{\Delta x^j_p}\right)\varphi\left(\frac{y-y^j_\lambda}{\Delta y^j_p}\right),\ \tilde{\varphi}^j_\lambda(x,y) = \tilde{\varphi}\left(\frac{x-x^j_\lambda}{\Delta x^j_p}\right)\tilde{\varphi}\left(\frac{y-y^j_\lambda}{\Delta y^j_p}\right)$$

with $\langle\varphi^j_{\lambda_1},\tilde{\varphi}^j_{\lambda_2}\rangle = \delta_{\lambda_1,\lambda_2}$

and the associated three biorthogonal wavelets (in the D -dimensional case we have 2^D-1 wavelets see [105, p.41]) in $D=2$ spatial dimensions:

$$\psi^j_{1,\lambda}(x,y) = \psi\left(\frac{x-x^j_\lambda}{\Delta x^j_p}\right)\varphi\left(\frac{y-y^j_\lambda}{\Delta y^j_p}\right),\ \tilde{\psi}^j_{1,\lambda}(x,y) = \tilde{\psi}\left(\frac{x-x^j_\lambda}{\Delta x^j_p}\right)\tilde{\varphi}\left(\frac{y-y^j_\lambda}{\Delta y^j_p}\right)$$

$$\psi^j_{2,\lambda}(x,y) = \varphi\left(\frac{x-x^j_\lambda}{\Delta x^j_p}\right)\psi\left(\frac{y-y^j_\lambda}{\Delta y^j_p}\right),\ \tilde{\psi}^j_{1,\lambda}(x,y) = \tilde{\varphi}\left(\frac{x-x^j_\lambda}{\Delta x^j_p}\right)\tilde{\psi}\left(\frac{y-y^j_\lambda}{\Delta y^j_p}\right)$$

$$\psi^j_{3,\lambda}(x,y) = \psi\left(\frac{x-x^j_\lambda}{\Delta x^j_p}\right)\psi\left(\frac{y-y^j_\lambda}{\Delta y^j_p}\right),\ \tilde{\psi}^j_{1,\lambda}(x,y) = \tilde{\psi}\left(\frac{x-x^j_\lambda}{\Delta x^j_p}\right)\tilde{\psi}\left(\frac{y-y^j_\lambda}{\Delta y^j_p}\right)$$

with $\langle\psi^{j_1}_{\mu_1,\lambda_1},\tilde{\psi}^{j_2}_{\mu_2,\lambda_2}\rangle = \delta_{\mu_1,\mu_2}\delta_{\lambda_1,\lambda_2}\delta_{j_1,j_2}$

for the horizontal ($\mu=1$), vertical ($\mu=2$) and diagonal ($\mu=3$) direction. The wavelet and its dual are defined by the same scaling relations as φ and $\tilde{\varphi}$:

$$\psi(x) = \sum_{k=-N}^{N} g_k\psi(2x-k) \quad\text{and}\quad \tilde{\psi}(x) = \sum_{k=-N}^{N} \tilde{g}_k\tilde{\psi}(2x-k). \tag{2.21}$$

The filter coefficients are $g_k = (-1)^{1-k}\tilde{h}_{1-k}$ and $\tilde{g}_k = (-1)^{1-k}h_{1-k}$ with $h_k, \tilde{h}_k$ listed in Table 2.1. The components of the scaling and wavelet coefficients (c^j_λ and $d^j_{\mu\lambda}$) are determined via component-wise projection of Eq. (2.20) onto the dual basis $\{\tilde{\varphi}^j_\lambda, \tilde{\psi}^j_{\mu\lambda}\}$:

$$c^j_\lambda = \langle q, \tilde{\varphi}^j_\lambda \rangle\,, \qquad d^j_{\mu\lambda} = \langle q, \tilde{\psi}^j_{\mu\lambda} \rangle\,, \qquad \text{where} \quad \lambda = (p, k_1, k_2) \tag{2.22}$$

assuming that $\langle \varphi^j_{\lambda_1}, \tilde{\psi}^j_{\lambda_2} \rangle = \langle \psi^j_{\lambda_1}, \tilde{\varphi}^j_{\lambda_2} \rangle = 0$ are orthogonal. Note that $\langle a, b \rangle = \int_{\mathbb{D}} a(\boldsymbol{x}) b(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$ denotes the L^2-inner product.

Instead of using *Deslauriers-Dubuc* (DD) wavelets, as in the framework of Harten [69], we use lifted Deslauriers-Dubuc wavelets, i.e. *biorthogonal Cohen–Daubechies—Feauveau wavelets* of fourth-order (CDF 4,4) with filter coefficients h_k and dual filter coefficients $\tilde{h}_k$ listed in Table 2.1. CDF 4,4 wavelets allow a better scale separation and can be easily implemented by replacing the loose down-sampling filter with a low pass filter before coarsening the grid.

In most wavelet adaptation schemes, one truncates Eq. (2.20) such that only detail coefficients are kept which carry significant information. According to [124] *"this can be expressed as a non-linear filter"*, which acts as a cut-off for wavelet coefficients with small magnitude. The cut-off is given by the threshold parameter $\epsilon > 0$. However, in contrast to these schemes, our block-based adaptation groups the detail coefficients in blocks, i.e. all details on the block are kept if at least one detail carries important information. This seems to be less efficient at first sight, because unnecessary information is kept, but grouping details in blocks is reasonable, since the block-based adaptation is computationally efficient for MPI-distributed architectures. Moreover, groups of significant details are often nearest neighbors, rather than a single significant detail in a block. Therefore, we define the set of blocks

$$I^j_\epsilon := \left\{ p \in \Lambda^j \;|\; \max_{\substack{\mu=1,2,3 \\ k_1=0,\dots,B_x-1 \\ k_2=0,\dots,B_y-1}} \left| d^j_\mu[p, k_1, k_2] \right| > \epsilon \right\} \tag{2.23}$$

with significant details in the predefined tree level range $J_{\min} \leq j \leq J_{\max}$. In the spirit of our previous notation we thus define: $\overline{I}^j_\epsilon :=$

$I_\epsilon^j \times \{0,\dots,B_x-1\} \times \{0,\dots,B_y-1\}$ for the set of all significant detail indices. The filtered block-based wavelet field in Eq. (2.20) can now be written as follows:

$$q^\epsilon = \sum_{\lambda\in\overline{\Lambda}^J_{\min}} c_\lambda^{J_{\min}}\varphi_\lambda^{J_{\min}} + \sum_{j=J_{\min}}^{J_{\max}-1}\sum_{\lambda\in\overline{I}_\epsilon^j}\sum_{\mu=0}^{3} d^j_{\mu,\lambda}\psi^j_{\mu,\lambda}\,. \tag{2.24}$$

In the following, we will denote all fields with an upper index ϵ, which have been filtered with Algorithm 1 and can be thus expressed as Eq. (2.24).

For an example we have computed the vorticity $\omega^\epsilon = \partial_x u_2^\epsilon - \partial_y u_1^\epsilon$ of a thresholded vector field $\boldsymbol{q}^\epsilon = (u_1^\epsilon, u_2^\epsilon, p^\epsilon)$ in Fig. 2.5 for various ϵ (more details can be found in Section 3.3.2). Here, $\epsilon > 0$ and $\epsilon = 0$ correspond to a filtered and unfiltered field, respectively. For increasing ϵ, fewer detail coefficients will be above the threshold and therefore the number of blocks decreases. Taking the difference between the

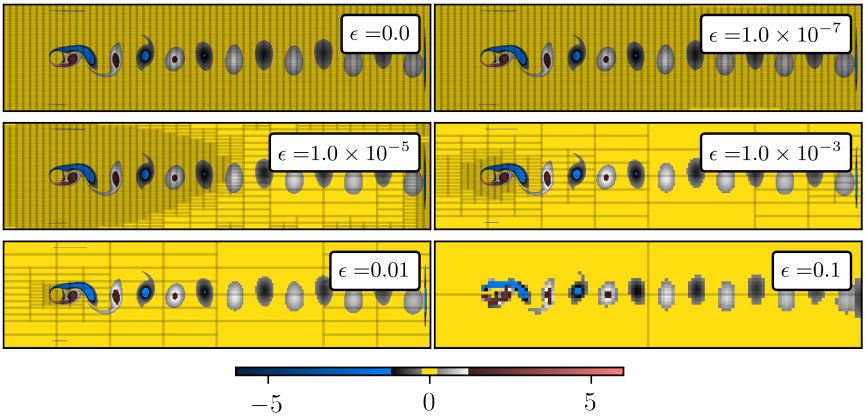


Figure 2.5: Block-based adaptation of a flow past a cylinder for different thresholds ϵ. Shown is the vorticity field $\omega^\epsilon = \partial_x u_2^\epsilon - \partial_y u_1^\epsilon$, which is computed after having applied the wavelet adaptation to the full state vector $\boldsymbol{q}^\epsilon = (u_1^\epsilon, u_2^\epsilon, p^\epsilon)$. Each block represents $B_x \times B_y = 65 \times 17$ points. More details can be found in Section 3.3.2.

thresholded Eq. (2.24) and the original field Eq. (2.20) only details

below the threshold are left. Hence the total error can be estimated and yields:

$$\|q - q^\epsilon\| \leq \sum_{j=J_{\min}}^{J_{\max}-1} \sum_{\lambda \in \overline{I}_\epsilon^{j\,c}} \sum_{\mu=0}^{3} \left|d^j_{\mu,\lambda}\right| \left\|\psi^j_{\mu,\lambda}\right\| . \tag{2.25}$$

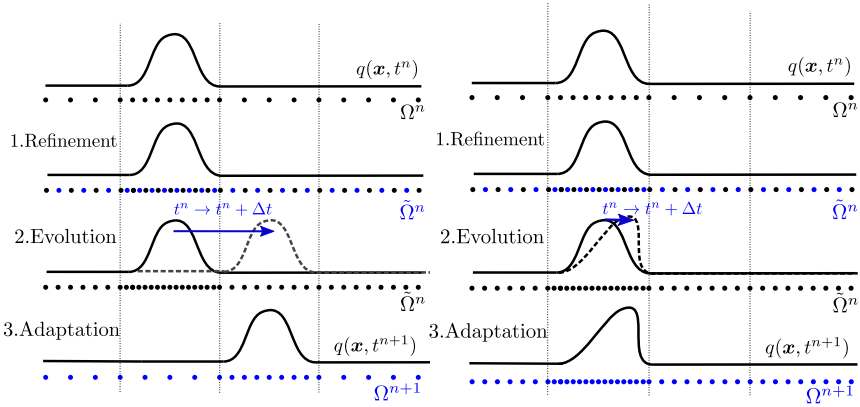
Because $\left\|\psi^j_{\mu,\lambda}\right\|_\infty = 1$ we finally get $\|q - q^\epsilon\|_\infty \leq C_\epsilon \epsilon$ for the total error in the L^∞-norm. Similarly one can normalize $\psi^j_{\mu,\lambda}$ in the L^2-norm, which corresponds to re-weighting the thresholding criterion $\left|d^j_{\mu,\lambda}\right| < \epsilon$ with a level (j) and dimension (D) dependent threshold: $\left|d^j_{\mu,\lambda}\right| < 2^{-D(j-J_{\max})/2}\epsilon$ [118].

The compression errors and compression rate of the vorticity field shown in Fig. 2.5 are further analyzed in Section 3.3.2.

2.1.2 Safety Zone and Time-stepping

The adaptation algorithm in Section 2.1.1 allows to represent large-scale data fields with fewer degrees of freedom, while retaining a good approximation. Unfortunately, the chosen representation is not co-moving, which would be advantageous for transported quantities. Therefore, additional detail coefficients are required to approximate the evolution of the transported quantity. A concept to ensure a sufficient approximation during time integration is called the *adjacent* or *safety zone*, that was introduced in [90]. The concept is illustrated for the block-based adaptive scheme in Fig. 2.6.

In the first step, the initial grid Ω^n is refined by midpoint insertion. The intermediate grid $\tilde{\Omega}^n$ Therefore, contains additional grid points associated with wavelet coefficients that can become significant during the time evolution. Vasilyev reports in [136, p.157] that the adjacent zone optimally includes only the nearest points of a significant detail coefficient, which is in accordance with the CFL condition (Courant–Friedrichs–Lewy [29]) a necessary condition for stability, which states that the time step size Δt is limited by the duration a wave requires to travel to adjacent grid points. Although only neighboring details are preferred for the safety zone, our block structure requires



(a) Linear transport **(b)** Non-linear transport

Figure 2.6: Illustration of the safety zone during time-stepping (disregarding the CFL-condition) in the linear case (a) and non-linear case (b).
The time-stepping procedure includes three steps:
1.) The initial grid Ω^n is refined $\Omega^n \to \tilde{\Omega}^n$ by midpoint insertion.
2.) The solution $\underline{\boldsymbol{q}}(t^n)$ is evolved on the fine grid $\tilde{\Omega}^n$: $\underline{\boldsymbol{q}}(t^n) \to \underline{\boldsymbol{q}}(t^{n+1})$.
3.) The grid is adapted ($\tilde{\Omega}^n \to \Omega^{n+1}$) to the significant details of $\underline{\boldsymbol{q}}(t^{n+1})$.

adding additional coefficients for the adjacent zone. In the second step, the numerical solution $\underline{\boldsymbol{q}}(t^n)$ is evolved on the refined grid $\tilde{\Omega}^n$ using explicit time integration. Usually, one time step is globally selected for all blocks. For this, the CFL condition is evaluated on each block and the smallest time step is chosen globally. In the third step, the grid is adapted according to the details of the new system state $\underline{\boldsymbol{q}}(t^{n+1})$ using Algorithm 2. Note that after the grid has been changed the workload has to be balanced among the CPUs. This issue is briefly discussed in Section 3.2.1.

Usually, `WABBIT` is limited to a maximal refinement level $J_{\max}$ during a simulation. In the case a block is still on $J_{\max}$ after the last step it might be beneficial to apply a filter if viscous terms in the operator do not stabilize the numerical scheme. To accomplish this filter we choose to coarsen all blocks on $J_{\max}$, which is termed *dealiasing* in the following. As the numerical studies in [46] indicate, dealiasing leads to slightly reduced errors if non-linearities are present in the differential operator $\mathcal{N}$.

2.1.3 Stability of the Finite Difference Discretization

Although block-based wavelet adaptation combined with finite-difference methods promises efficient simulations of TDFS, it is not clear if the outlined time-stepping (Section 2.1.2) leads to a stable numerical scheme. In fact, Svärd and Nordström state in [130, abstract]: *"High-order [accurate multi-block] finite difference methods are efficient, easy to program, scale well in multiple dimensions (...). The main drawback has been the complicated and sometimes mysterious stability treatment at boundaries and interfaces required for a stable scheme."* To tackle this issue, [130] propose summation-by-parts simultaneous-approximation-term (SBP-SAT) finite difference schemes, which are however not used here. Therefore, we investigate the stability of the wavelet adaptive block-based scheme in this section.

In the following we denote $\mathcal{H}_{\Delta t}^{\mathcal{CR}}$ as the explicit-time-evolution operator, which evolves the discretized solution $\underline{\boldsymbol{q}}^n \in \mathbb{R}^M$ at time t_n to $t_{n+1} = t_n + \Delta t$:

$$\underline{\boldsymbol{q}}^{n+1} = \mathcal{H}_{\Delta t}^{\mathcal{CR}}(\underline{\boldsymbol{q}}^n)\,. \tag{2.26}$$

For stability analysis of $\mathcal{H}_{\Delta t}^{\mathcal{CR}}$ we make use of the matrix-method [50, p.81], since it allows to analyze the total time evolution step (see Section 2.1.2), including the refinement $\mathcal{R}$, time evolution on the refined grid $\mathcal{H}_{\Delta t}$ and the adaptation/coarsening step $\mathcal{C}$. For the stability analysis, we approximate the right-hand side operator $\mathcal{N}$, with central finite differences of 4th-order accuracy. Furthermore, Φ denotes the time-stepping scheme. With the introduced notation the non-linear time-evolution operator can be written as:

$$\mathcal{H}_{\Delta t}^{\mathcal{CR}} := \mathcal{C} \underbrace{(\mathbb{1}_{2M} + \Delta t \Phi)}_{\mathcal{H}_{\Delta t}} \mathcal{R}\,, \tag{2.27}$$

with $\mathbb{1}_K \in \mathbb{R}^{K\times K}$ being the identity matrix. To simplify our analysis we assume that the evolved solution $\underline{\boldsymbol{q}}^{n+1}$ lies on the initial grid Ω^n, i.e. $\Omega^{n+1} = \Omega^n$. Therefore, $\mathcal{CR} = \mathbb{1}_M$ and the computed linearized operator

$$[\mathbf{H}_{\Delta t}^{\mathcal{CR}}]_{ji} = \left[\lim_{\tau\to 0} \frac{\mathcal{H}_{\Delta t}^{\mathcal{CR}}(\underline{\boldsymbol{q}}^n + \tau \underline{\boldsymbol{e}}_i) - \mathcal{H}_{\Delta t}^{\mathcal{CR}}(\underline{\boldsymbol{q}}^n)}{\tau}\right]_j \tag{2.28}$$

is of size $M \times M$, if the initial grid has M grid points. According to [50, p.81] for systems without non-normal transient growth, stability is achieved if all eigenvalues λ_i of $\mathcal{H}_{\Delta t}^{\mathcal{CR}}$ satisfy $|\lambda_i| \leq 1$, where eigenvalues with $|\lambda_i| = 1$ are simple. This can be intuitively understood: since $\mathbf{H}_{\Delta t}^{\mathcal{CR}}$ has distinct eigenvalues $\mathbf{H}_{\Delta t}^{\mathcal{CR}}$ is diagonalizable and thus every disturbance vector $\underline{\boldsymbol{\xi}}^0 = \sum_{i=1}^{M} \alpha_i \underline{\boldsymbol{v}}_i \in \mathbb{R}^M$ can be represented in the eigenbasis $\{\underline{\boldsymbol{v}}_1, \ldots, \underline{\boldsymbol{v}}_M\}$ of $\mathbf{H}_{\Delta t}^{\mathcal{CR}}$. Hence, after the nth iteration/time step the error $\underline{\boldsymbol{\xi}}^n$ is decreased:

$$\left\|\underline{\boldsymbol{\xi}}^n\right\| \leq \left\|(\mathbf{H}_{\Delta t}^{\mathcal{CR}})^n \underline{\boldsymbol{\xi}}^0\right\| = \left\|\sum_{i=1}^{M} \alpha_i (\lambda_i)^n \underline{\boldsymbol{v}}_i\right\| \leq \max_{1\leq i\leq M} |\lambda_i^n| \left\|\underline{\boldsymbol{\xi}}^0\right\| \tag{2.29}$$

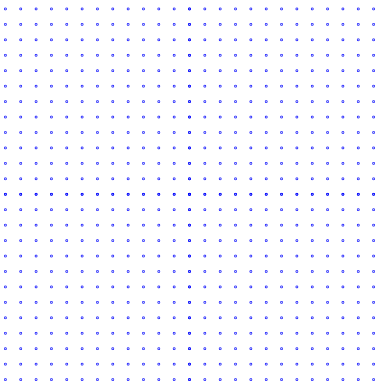
if $|\lambda_i| \leq 1$, for all $i = 1, \ldots, M$. We note that for the special case that $\mathcal{CR} = \mathbb{1}_M$ this criterion is trivially fulfilled if $\Delta t = 0$. For example, in 2D we make use of the RHS $\mathcal{N}(q) = -\boldsymbol{u}\cdot\boldsymbol{\nabla} q + \kappa \boldsymbol{\nabla}^2 q$, $\boldsymbol{u} = \frac{1}{\sqrt{2}}(1,1)$, $\kappa \geq 0$, (see Section 3.3). The three test meshes with block size $B_x \times B_y = 13 \times 13$ and $g = 6$ ghost nodes are shown in Fig. 2.7. The time-integration method Φ implements a classical Runge-Kutta of

fourth order with time step size $\Delta t = \min(hK_{1,\mathrm{CFL}}, h^2 K_{2,\mathrm{CFL}}/\kappa)$ for $\kappa > 0$ and $\Delta t = h$ for $\kappa = 0$, where h is the smallest lattice spacing of the grid. The time steps are chosen according to the CFL condition ($K_{1,\mathrm{CFL}} \leq 1$, $K_{2,\mathrm{CFL}} \leq (2D)^{-1}$), necessary for a stable scheme with equal lattice spacing h. Applying the stability criterion for the operator $\mathbf{H}_{\Delta t}^{\mathcal{CR}}$ that includes the coarsening and refinement step and the operator $\mathbf{H}_{\Delta t}$ without coarsening and refinement, leads to the results listed in Table 2.2. Some eigenvalues of $\mathbf{H}_{\Delta t}$ are shown in Fig. 2.8. The results

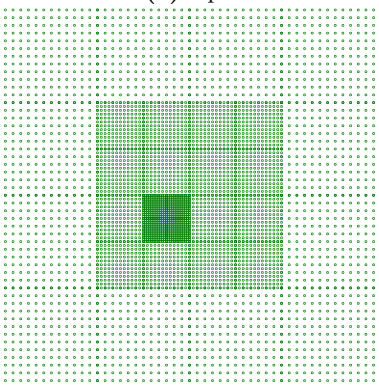
	$\mathbf{H}_{\Delta t}$			$\mathbf{H}_{\Delta t}^{\mathcal{CR}}$		
	equi	3-lvl	3-lvl-sym	equi	3-lvl	3-lvl-sym
$\kappa = 0$	stable	unstable	stable	stable	stable	stable
$\kappa = 1$	stable	stable	stable	stable	stable	stable

Table 2.2: Operator stability analysis for the advection-diffusion PDE $\mathcal{N}(q) = -\boldsymbol{u} \cdot \boldsymbol{\nabla} q + \kappa \boldsymbol{\nabla}^2 q$, $\boldsymbol{u} = \frac{1}{\sqrt{2}}(1,1)$, $\kappa \geq 0$, using the CFL condition $\Delta t = \min(hK_{1,\mathrm{CFL}}, h^2 K_{2,\mathrm{CFL}}\kappa)$ for $\kappa > 0$ and $\Delta t = h$ for $\kappa = 0$, where h is the smallest lattice spacing, $K_{1,\mathrm{CFL}} = 1$ and $K_{2,\mathrm{CFL}} = (2D)^{-1}$. The operator $\mathbf{H}_{\Delta t}$ (no refinement and coarsening) and $\mathbf{H}_{\Delta t}^{\mathcal{CR}}$ are analyzed on three different lattices "equi", "3-lvl", "3-lvl-sym" that are shown in Fig. 2.7.

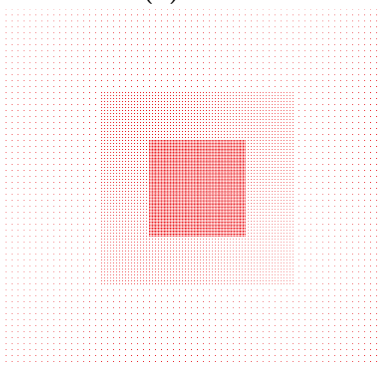
indicate that stability can vary for different mesh geometries if no additional damping is introduced. However, if the evolution is dampened by friction from a viscosity term $\kappa \boldsymbol{\nabla}^2 q$ or the numerical dissipation stemming from a CDF 4,4-filter in the coarsening step, the operator is stabilized. This observation also holds for higher-level differences not shown in this thesis. In contrast to the SBP-SAT formulation [130], the here presented method, does not guarantee global stability implied by structural properties like energy conservation, which is a weakness of the presented numerical scheme. However, we observe no instability for the specific 2D example, if energy is dissipated by the wavelet filter in the coarsening step $\mathcal{C}$ or due to viscosity $\kappa > 0$. This observation is supported by the estimated 4th-order of convergence in the numerical study of the advection-reaction-diffusion system in Section 3.3.



(a) equi



(b) 3-lvl



(c) 3-lvl-sym

Figure 2.7: Three different grids used for the stability analysis: a) equidistant mesh, b) asymmetric mesh with three different grid levels, c) symmetric mesh with three different grid levels.

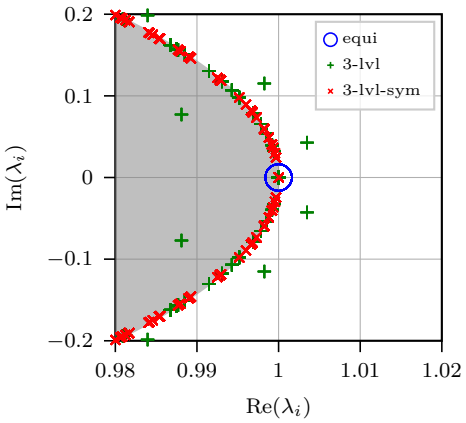


Figure 2.8: Eigenvalues of $\mathbf{H}_{\Delta t}$ in the complex plane. The eigenvalues correspond to the meshes "equi", "3-lvl", "3-lvl-sym" shown in Fig. 2.7. The grey shaded area marks the stable regime for which $|\lambda_i| \leq 1$. Not all eigenvalues are shown.

2.1.4 Volume Penalization for Moving Geometries

If a solid body is moving inside a flow, structures are created which are advected by the movement of the body. Examples are the leading edge vortex of a wingtip of a bumblebee (see Fig. 3.12), the tip vortex of a wind turbine, or the vortex shedding created by the undulatory swimming motion of fishes. To study such systems, we shortly discuss the implementation of non-slip boundary conditions for moving solids.

Often the computational grid is fit to the boundary of an obstacle $\mathbb{D}_\mathrm{s}$, using body-fit grids. Especially for complex moving geometries, like insects or deforming bodies, these methods maybe not be applicable or require costly remeshing in each time step. Instead, volume penalization [3, 18, 13, 24] embeds the solid obstacle by extending the computational domain $\mathbb{D}$ to the interior of the obstacle $\mathbb{D}_\mathrm{s} \subset \mathbb{D}$, where additional penalization terms are added to the set of equations to account for the boundary conditions on the solid-fluid interface $\partial\mathbb{D}_\mathrm{s}$. Hence, the IBVP Eq. (2.1) with initial condition $q(\boldsymbol{x}, 0) = q_0(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{D}$ can be reformu-

lated in the following way:

$$\begin{cases} \partial_t q = \mathcal{N}(q, \boldsymbol{x}, t) & \text{for} \quad (\boldsymbol{x}, t) \in \mathbb{D} \times \mathbb{R}^+ \\ q = q_{\text{ref}} & \text{for} \quad (\boldsymbol{x}, t) \in \partial\mathbb{D}_{\text{s}} \times \mathbb{R}^+ \end{cases} \tag{2.30}$$

$$\Big\downarrow \text{penalized}$$

$$\partial_t q = \mathcal{N}(q, \boldsymbol{x}, t) - \underbrace{\frac{\chi_{\mathbb{D}_{\text{s}}}}{C_\chi}(q - q_{\text{ref}})}_{\text{penalization term}} \quad \text{for} \quad (\boldsymbol{x}, t) \in \mathbb{D} \times \mathbb{R}^+ . \tag{2.31}$$

In the volume penalized equation system Eq. (2.31), the shape of the solid body is represented by a *mask function* $\chi_{\mathbb{D}_{\text{s}}} : \mathbb{D} \times \mathbb{R}^+ \to [0, 1]$, which is usually time-dependent if the structure is moving. As illustrated in Fig. 2.9, the mask function is $\chi_{\mathbb{D}_{\text{s}}} = 1$ inside and $\chi_{\mathbb{D}_{\text{s}}} = 0$ outside of $\mathbb{D}_{\text{s}}$. Hence, outside $\mathbb{D}_{\text{s}}$ one recovers the unpenalized equation

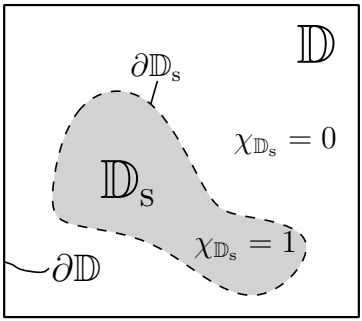


Figure 2.9: Illustration of the computational domain $\mathbb{D}$ with an obstacle $\mathbb{D}_{\text{s}}$ drawn in gray. The fluid-solid boundary $\partial\mathbb{D}_{\text{s}}$ is indicated with a dashed line. Periodic boundaries are imposed at $\partial\mathbb{D}$. The values of the mask function $\chi_{\mathbb{D}_{\text{s}}}$ inside and outside the domain are displayed.

system. Whereas inside $\mathbb{D}_{\text{s}}$, the additional term forces the state q to converge towards the reference value q_{ref}. The strength of the penalization term is regulated by the so-called *porosity parameter* $0 < C_\chi \ll 1$. Thus for decreasing porosity $C_\chi \to 0$ the difference $(q - q_{\text{ref}})$ converges to zero. Furthermore, it has been shown for incompressible flows that the penalized solution q_{C_χ} [3] converges towards the exact solution q

as:

$$\mathcal{E}_{\text{penal}} = \|q - q_{C_\chi}\|_2 = \mathcal{O}\left(C_\chi^{1/2}\right) . \tag{2.32}$$

Hence, a small C_χ seems to be desirable. However, C_χ limits the possible time step size to $\Delta t \leq C_\chi$ for stability reasons and as reported for incompressible flows in [43, 44] a C_χ that is too small results in a loss of regularity with increasing discretization errors. To balance modeling and discretization errors, [43, 44] propose to choose

$$C_\chi = (K_\chi^2/\nu)h^2 , \tag{2.33}$$

depending on the lattice spacing h, the viscosity of the fluid ν and a discretization dependent constant $K_\chi > 0$. The resulting penalisation error Eq. (2.32) thus becomes $\mathcal{E}_{\text{penal}} = \mathcal{O}(h)$.

Note that usually additional boundary conditions are imposed at the domain boundaries $\partial\mathbb{D}$. In this thesis we continue the domain periodically, although inflow and outflow conditions are imposed close to $\partial\mathbb{D}$ using volume penalization (see Section 2.2.3).

2.1.5 Error Estimation

For the here presented results all spatial differential operators in $\mathcal{N}$ are approximated by central finite differences of fourth-order. Hence, the expected discretization error of the numerical scheme is $\mathcal{E}_{\text{diff}}(h) = \mathcal{O}(h^4)$, assuming equal lattice spacing $h > 0$ in all directions and sufficient regularity of the solution. Furthermore, we use Runge-Kutta fourth-order time integration with truncation error $\mathcal{E}_T(\Delta t) = \mathcal{O}(\Delta t^4)$ assuming enough regularity in time. Disregarding penalization, the resulting approximation $q^{\epsilon,\Delta t,h}$ of the exact solution q yields:

$$\mathcal{E}_{\text{WABBIT}}(h, \Delta t, \epsilon) := \left\| q_n - q_n^{\epsilon,\Delta t,h} \right\|_2 \leq \mathcal{E}_T(\Delta t) + \mathcal{E}_{\text{diff}}(h) + \|q_n - q_n^\epsilon\|_2 \tag{2.34}$$

using the triangle inequality. Here, the lower index $q_n = q(\cdot, t_n)$ denotes the nth time step and the upper indices the corresponding truncation parameters affecting the solution. The last term in Eq. (2.34) is discussed in [27, 28]. It quantifies the error introduced by the cumulation of compression errors in each time step. Assuming exact integration in

time, no truncation errors of the differential operators in $\mathcal{N}$, same initial condition and the numerical scheme to be contractive (see Assumption 4.1 in [28]), the worst-case error introduced during time-stepping is the accumulation of the thresholding error Eq. (2.25) in each time step. Thus, the conservative error estimate yields

$$\|q_n - q_n^\epsilon\|_2 \leq n\tilde{C}_\epsilon \epsilon \tag{2.35}$$

after n time steps. We want to emphasize here, that Eq. (2.35) implies that the evolution operator is represented on a grid that is fine enough, such that no additional errors larger than $\mathcal{O}(\epsilon)$ are introduced when representing $\mathcal{N}$. This is achieved using the safety zone concept, introduced in Section 2.1.2. Nevertheless, according to [27] the safety zone concept gives excellent practical results but is only a heuristic that has not rigorously been proven. As [27, 46] suggests Eq. (2.35) should be balanced with the truncation error of the differential operators. Assuming a CFL time stepping $n = T/\Delta t = \mathcal{O}(h^{-1})$ with $h = \Delta x^{J_{\max}}$ being the lattice spacing on the finest scale, we conclude to set $\epsilon = \mathcal{O}(h^5)$ in order to balance the adaptation errors with $\mathcal{E}_{\text{diff}}$. However, as the numerical experiments in Section 2.2 show, this estimate is much too conservative.

2.2 Numerical Results

In the following, we provide numerical test cases of transport dominated fluid systems, which we address throughout this thesis. If not otherwise stated we use CDF 4,4 wavelets that are normalized in L^∞. All performance tests have been performed on `11th Gen Intel(R) Core(TM) i7-11850H` CPUs (8 processing units), if not stated otherwise. The listed CPU-times are the accumulation of the total CPU usage. Additionally to the CFD-simulation, the CPU-time includes input/output operations and MPI synchronization.

2.2.1 Advection Equation

The first test case implements the advection of a disk in a circle in two spatial dimensions. We introduce the two-dimensional advection

equation

$$\begin{cases} \partial_t q(\boldsymbol{x},t) = \boldsymbol{v}(t)\cdot\boldsymbol{\nabla} q(\boldsymbol{x},t) & \text{for} \quad (\boldsymbol{x},t)\in[0,L]^2\times[0,T] \\ q(\boldsymbol{x},0) = q_0(\boldsymbol{x}) & \text{for} \quad \boldsymbol{x}\in[0,L]^2 \end{cases} \tag{2.36}$$

with analytic solution

$$q(\boldsymbol{x},t) = q_0(\boldsymbol{x}-\boldsymbol{\Delta}(t)) \quad \text{with} \quad \boldsymbol{\Delta}(t) = \int_0^t \boldsymbol{v}(\tau)\mathrm{d}\tau\,. \tag{2.37}$$

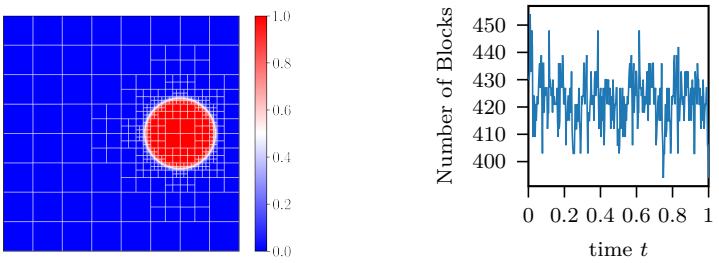
For the advected disk example specifically we have

$$q(\boldsymbol{x},t) = f(\|\boldsymbol{x}-\boldsymbol{\Delta}(t)\|_2 - R) \quad \text{with} \tag{2.38}$$

$$\boldsymbol{\Delta}(t) = L\begin{pmatrix} 0.5+1/4\cos(2\pi t)\\ 0.5+1/4\sin(2\pi t)\end{pmatrix},\ \boldsymbol{v}(t) = \frac{L\pi}{2}\begin{pmatrix} -\sin(2\pi t)\\ \cos(2\pi t)\end{pmatrix}, \tag{2.39}$$

using $f(x) = (1+\tanh(x/\lambda))/2$, with $\lambda = 0.005L$ and $R = 0.15L$, $T = 1$, $L = 1$. For the test case, we run simulations solving Eq. (2.36) with block size $B_x \times B_y = 17\times 17$ for various $J_{\max} = 2,3,\ldots,8$ and $\epsilon = 10^0, 10^{-1},\ldots,10^{-10}$. Since the operator is linear we do not use dealiasing here. Time steps are selected according to a CFL condition. The smallest time step $\Delta t^j = K_{\mathrm{CFL}}\Delta x^j/c, K_{\mathrm{CFL}} = 1$, $c = \|\boldsymbol{v}\|_2 = \frac{L\pi}{2}$ of all active levels j of the current mesh is chosen. First, we show a typical block distribution at the end of the simulation $t = 1$ in Fig. 2.10. Most blocks are located at the outer radius of the disc, where the derivative of q is large and many detail coefficients are required to represent the solution. Due to the special structures of the IBVP, the number of DOFs stays approximately constant as indicated by Fig. 2.10b.

Next, we investigate the impact of the adaptation algorithm on the utilized discretization scheme. Therefore, we plot the relative error of the numerical solution in Fig. 2.11. The errors are calculated after the simulation has ended at $t = T$. For all our obtained results we refine the grid onto the finest lattice spacing ($J_{\max} = 7$) and compare it to its initial condition. First of all, notice that with decreasing wavelet threshold ϵ the errors in Fig. 2.11a decrease linearly until they saturate to a minimal error that is dependent on the finest lattice spacing $J_{\max}$. This error is reached when the discretization errors become dominant over the thresholding errors. To show that the numerical accuracy is



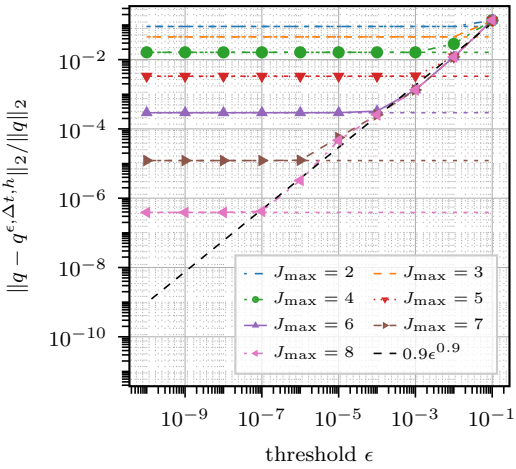
(a) Snapshot

(b) Number of blocks used during the simulation.

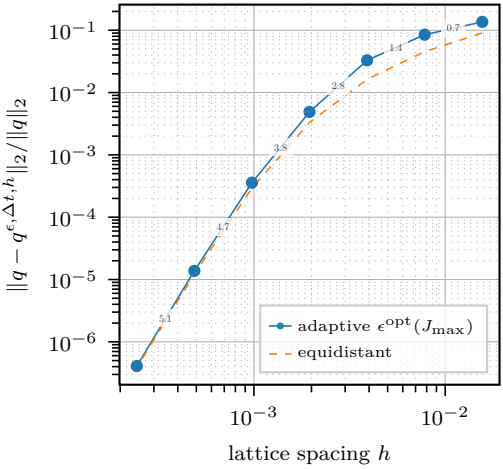
Figure 2.10: Block distribution of the solution during the simulation using $J_{\max} = 7$ and $\epsilon = 10^{-6}$. a) Block distribution at $t = 1$ and b) Number of blocks during the simulation.

kept, Engels et. al suggest in [46] to balance wavelet and discretization errors and compare them to the equidistant simulations, as shown in Fig. 2.11b. The balance point $\epsilon^{\text{opt}}(J_{\max})$ can be estimated for a given maximal refinement as the intersection of the vertical lower bound of the error with the sloped dashed line in Fig. 2.11a. The overlapping curves in Fig. 2.11b indicate that the precision of the differential scheme is preserved at the balance point. Note, that the experimental order of convergence (EOC) is dependent on the initial condition. In the regime of large lattice spacings $h \gtrapprox \lambda = 5 \times 10^{-3}$, the expected 4th-order convergence rate is not achieved, since the initial condition becomes non-smooth.

Although wavelet compression and discretization errors are balanced, a performance gain over the equidistant equivalent method might not always be achieved. For optimal performance, the costs for evaluating the operator $\mathcal{N}$ need to exceed the additional costs created by the wavelet adaptation (wavelet filtering, data management and synchronization). Depending on the application, this is not always feasible. However, increasing the block size ($B_x \times B_y$) of the grid will increase the costs for evaluation of $\mathcal{N}$ in favor of additional data manipulation stemming from the wavelet adaptation. This is especially true in three spatial dimensions since computational costs scale with the volume. Nevertheless, for the present example adaptivity has a benefit, as the



(a) Relative L^2-error vs. wavelet threshold ϵ



(b) Relative L^2-error vs. lattice spacing h

Figure 2.11: Relative error of the adaptive and equidistant simulation, calculated at $t = T$. a) Impact of the wavelet threshold ϵ on the overall error. b) Error of the equidistant computation and the adaptive computation with balanced wavelet threshold $\epsilon^{\text{opt}}(J_{\max})$. The threshold $\epsilon^{\text{opt}}(J_{\max})$ corresponds to the intersection of the vertical colored lines with the dashed black line in a).

CPU-times t_{CPU} in Table 2.3 show.

	adaptive			equidistant		
$J_{\max}$	DOF	rel. error	t_{CPU} (s)	DOF	rel. error	t_{CPU} (s)
2	4096	1.4e-01	7	4096	9.0e-02	1
3	12133	8.5e-02	14	16384	4.5e-02	5
4	39585	3.3e-02	45	65536	1.6e-02	32
5	114885	4.9e-03	190	262144	3.3e-03	232
6	265159	3.6e-04	827	1048576	2.9e-04	2076
7	1234660	1.4e-05	7694	4194304	1.2e-05	16930
8	5622661	4.1e-07	83266	16777216	3.9e-07	135584

Table 2.3: Adaptive $\epsilon^{\mathrm{opt}}(J_{\max})$ and equidistant results of the moving disk example. Listed are the maximal degrees of freedom (DOF) used during the simulation, relative errors at simulation time $t = T$ and the CPU-time t_{CPU} for the total simulation. The equidistant simulations have been computed with `WABBIT`. Therefore, the CPU-time of the equidistant simulation includes block synchronization, but not the adaptation of the computational grid.

2.2.2 Advection-Reaction-Diffusion with a KPP non-linearity

In this subsection we compute the numerical solution of the advection-reaction-diffusion (ARD) equation with a *Kolmogorov-Petrovsky-Piskunov* (KPP) non-linearity [83]

$$\begin{cases} \partial_t q & = -\boldsymbol{u}\cdot\boldsymbol{\nabla} q + \kappa\boldsymbol{\nabla}^2 q - \gamma q^2(q-1) \\ q(\boldsymbol{x},0) & = q_0(\boldsymbol{x}) \end{cases}, \tag{2.40}$$

in $(t,\boldsymbol{x}) \in [0,T]\times[0,L]^2$ with periodic boundary conditions. All simulation parameters are listed in Table 2.4.

For our test case we choose a velocity field inspired by the vortex pair example in [113]. Therefore, $\boldsymbol{u} = \boldsymbol{\nabla}\times\omega$ is expressed in terms of the vorticity

$$\omega(\boldsymbol{x},t) = \omega_0 e^{-t^2/\tau^2}(e^{-r_1^2(t)/r_0^2} + e^{-r_2^2(t)/r_0^2}), \tag{2.41}$$

which parametrizes a moving vortex pair with $r_i(t) = \|\boldsymbol{x}-\boldsymbol{x}_i(t)\|_2$, $\boldsymbol{x}_1 = L(0.6-ct, 0.49)$, $\boldsymbol{x}_2 = L(0.6-ct, 0.51)$, that decays slowly in

Name	Value
Simulation time	$T = 3$
Domain size	$L = 1$
Block size	$B_x = B_y = 17$
Diffusion constant	$\kappa = 10^{-4}$
Reaction constant	$\gamma = 10$
Advection of vortex pair	$c = 10$
Decay constant	$\tau = 3T$
Vortex amplitude	$\omega_0 = 10^3$
Vortex size	$r_0 = 5 \times 10^{-4}$

Table 2.4: Parameters of the 2D ARD simulation.

time. The initial distribution of the reactant q is given by:

$$q_0(\boldsymbol{x}) = f(\|\boldsymbol{x} - (0.4L, 0.5L)^\top\|_2 - 0.2L) \tag{2.42}$$

with $f(x) = (1+\tanh(x/\lambda))/2$ and $\lambda = 0.005L$, as before. The velocity field and initial distribution are tuned to mimic a flame kernel interacting with a vortex pair, which is a usual phenomenon in turbulence flame interactions. The time evolution of the simulation is visualized for $t = 0.0, 0.5, 1$ in Fig. 2.12. During the simulation, the prescribed

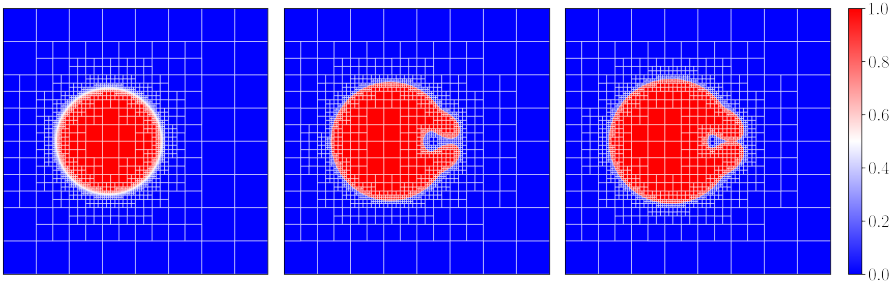


Figure 2.12: Visualization of the reactant q for $t = 0, 0.5, 1$ from left to right. The white lines indicate the block boundaries of the computational grid. The simulation corresponds to $J_{\max} = 7$ with $\epsilon = 10^{-6}$.

vortex pair Eq. (2.41) moves towards burning gas and mixes unburned

($q = 1$) with burned gas ($q = 0$), such that a small island of unburned gas detaches into the burned area, creating a topology change in the contour line of the moving front.

We compare our results to a 2D pseudo spectral (PS) simulation (for an introduction to PS methods we refer to [51]). The PS simulation with a resolution of 2048×2048 Fourier modes can be considered highly accurate since all spatial operators can be approximated up to machine precision. In fact, we make sure that the Fourier spectra $E(|\boldsymbol{k}|) = \frac{1}{2}\int_{|\boldsymbol{k}|-\Delta k}^{|\boldsymbol{k}|+\Delta k} |\mathcal{F}[q](\boldsymbol{k})|\,\mathrm{d}\,|\boldsymbol{k}|$, $\Delta k = 10^{-10}$ of the solution q decays to machine precision for all time instances. The spectra is shown in Fig. 2.13. Furthermore, we use an explicit Runge-Kutta method of order 5 with adaptive time stepping to control the errors based on the Dormand Prince method [40]. The relative error is set to 10^{-12}. Hence, the results of the PS simulation are taken as a reference.

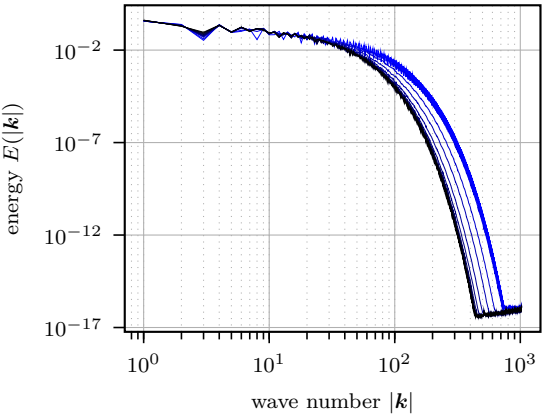


Figure 2.13: Radial Fourier spectra $E(|\boldsymbol{k}|) = \frac{1}{2}\int_{|\boldsymbol{k}|-\Delta k}^{|\boldsymbol{k}|+\Delta k} |\mathcal{F}[q](\boldsymbol{k})|\,\mathrm{d}\,|\boldsymbol{k}|$ of the Fourier transformed snapshots $\mathcal{F}[q](\boldsymbol{k})$. The color gradient of the lines indicates the time instance from dark blue $t = 0$ to bright blue at $t = 1$.

Similar to the analysis in Section 2.2.1 we perform adaptive and equidistant simulations for variable maximal refinement level $J_{\max} = 2, 3, 4, 5, 6$, where the adaptive simulations have been performed for different wavelet thresholds $\epsilon = 10^0, 10^{-1}, \ldots, 10^{-10}$. Time-stepping is done using the

CFL condition:

$$\Delta t = \min\left(h\frac{K_{1,\mathrm{CFL}}}{\|\boldsymbol{u}\|_2}, h^2\frac{K_{2,\mathrm{CFL}}}{\kappa}\right) \tag{2.43}$$

with $K_{1,\mathrm{CFL}} = K_{2,\mathrm{CFL}} = (2D)^{-1}$, $D = 2$ assuming that the reaction is mainly driven by the diffusion of the reactant or advection speed. The error of the solution is measured at the last snapshot $t = 1$ and is shown in Fig. 2.14. As before we define the optimal threshold ϵ^{opt} for a given $J_{\max}$ as the balance point between the compression and finite difference truncation error. Furthermore, we perform simulations

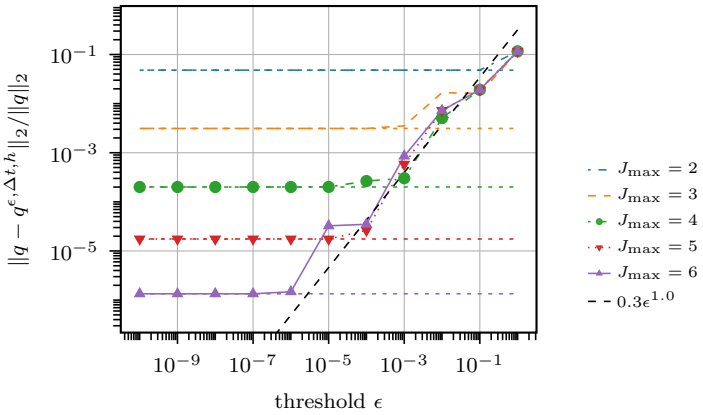


Figure 2.14: Numerical error of the solution at $t = 1$ vs. wavelet threshold ϵ without dealiasing. The dashed horizontal lines indicate the error of the equidistant computations.

using the dealiasing strategy outlined in Section 2.1.2. Note that for dealiasing we evaluate the right-hand side ($\mathcal{N}$) at $J_{\max} + 1$, but after the evolution, the blocks at $J_{\max} + 1$ are coarsened to $J_{\max}$. The final errors of the computations at ϵ^{opt} are listed in Table 2.5 together with the CPU-time and the maximal number of degrees of freedom. From the direct comparisons of equidistant and adaptive simulations, it is evident that the adaptive simulations become beneficial in the high-precision regime. In this regime, the adaptation scheme distributes most workload at the position of the front, where most information is captured (as seen in Fig. 2.12). It should be further mentioned that

	adaptive dealias			equidistant dealias		
$J_{\max}$	DOF	rel. error	t_{cpu} (s)	DOF	rel. error	t_{cpu} (s)
2	16384	3.6e-02	81	16384	3.9e-02	95
3	35378	1.3e-02	181	65536	3.9e-03	291
4	116014	2.9e-04	537	262144	2.1e-04	1093
5	316997	9.3e-06	2167	1048576	8.2e-06	6654
6	855818	1.1e-06	11465	4194304	7.4e-07	54501

	adaptive			equidistant		
$J_{\max}$	DOF	rel. error	t_{cpu} (s)	DOF	rel. error	t_{cpu} (s)
2	4096	4.8e-02	28	4096	4.8e-02	30
3	16384	1.7e-02	87	16384	3.1e-03	71
4	60862	3.0e-04	288	65536	2.0e-04	241
5	163318	2.2e-05	742	262144	1.8e-05	907
6	424480	1.3e-06	2954	1048576	1.3e-06	5641

Table 2.5: Adaptive $\epsilon^{\text{opt}}(J_{\max})$ and equidistant results of the ARD example Eq. (2.40). Listed are the maximal degrees of freedom (DOF) used during the simulation, relative errors at simulation time $t = T$ and the CPU-time t_{CPU} for the total simulation. The upper table indicates the results using dealiasing and the lower table the results without dealiasing.

for $J_{\max} < 4$ the equidistant DOF is equal to the maximal number of DOF during the adaptive simulations, which shows that no wavelet compression was achieved or at least not for all time steps in the evolution. Comparing the CPU-time t_{CPU} and DOF in Table 2.5 shows that simulations at $J_{\max}$ with dealiasing require almost the same computational effort as the ones without dealiasing at $J_{\max} + 1$, while gaining only little precision (see also Fig. 2.15). Dealiasing is therefore only recommended if the simulation needs stabilization.

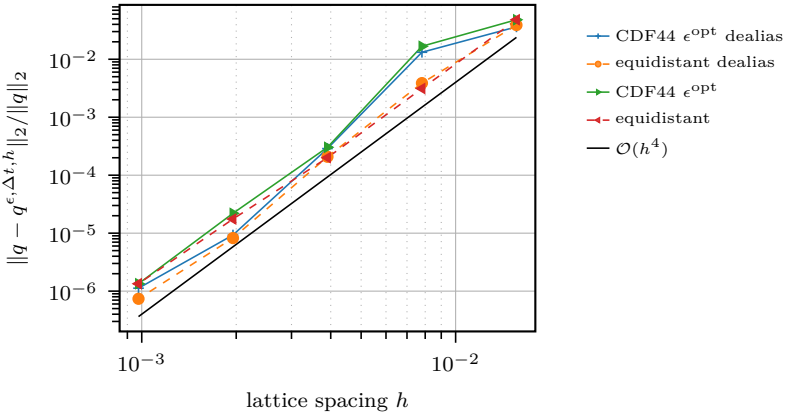


Figure 2.15: Spatial convergence of the numerical error using equidistant and adaptive simulations with and without dealiasing for the ARD example Eq. (2.40). The black solid line corresponds to the expected 4th-order scaling of the error when decreasing the lattice spacing.

2.2.3 Artificial Compressibility for Incompressible Flows

In this subsection, we introduce the vortex street, which is a prominent test case in fluid dynamics. In this test case, a uniform flow passes a cylinder and creates vortex shedding with a periodic pattern. To simulate this phenomenon, we closely follow the approach of [46] which combines volume penalization with wavelet adaptation for large-scale incompressible flows. In this study the *artificial compressibility method* (ACM)[25, 109, 64] is used, that approximates *incompressible Navier-*

Stokes (ICN) equations with constant density ρ

$$\partial_t \boldsymbol{u} + (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u} + \boldsymbol{\nabla} p - \nu\boldsymbol{\nabla}^2\boldsymbol{u} = 0\,, \tag{2.44}$$

$$\boldsymbol{\nabla}\cdot\boldsymbol{u} = 0\,, \tag{2.45}$$

$$\text{velocity}\quad \boldsymbol{u} = \hat{\boldsymbol{u}}/\rho\,,\quad \text{pressure}\quad p = \hat{p}/\rho, \tag{2.46}$$

$$\text{kinematic viscosity}\quad \nu = \mu/\rho \tag{2.47}$$

by introducing an artificial speed of sound $c_0 \gg \|\boldsymbol{u}\|_2$ that makes the fluid weakly compressible and thus avoids solving a Poisson equation in each time step. In the ACM model, the pressure p becomes a dynamic quantity and the incompressibility condition Eq. (2.45) is replaced by

$$\frac{1}{c_0^2}\partial_t p + \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0\,. \tag{2.48}$$

The resulting model Eqs. (2.44) and (2.48) converge towards the incompressible Navier-Stokes equations Eqs. (2.44) and (2.45) when $c_0 \to \infty$. The modelling error $\|\boldsymbol{q}_{\mathrm{ACM}} - \boldsymbol{q}_{\mathrm{ICN}}\|_2$ of the respective solutions $\boldsymbol{q} = (\boldsymbol{u}, p)$ converges with $\mathcal{O}\left(c_0^{-2}\right)$, which is numerically investigated in [46]. However, practically c_0 is chosen to be large enough such that $c_0 \gg \|\boldsymbol{u}\|_2$ is fulfilled and small enough to allow a sufficiently large time step size, when evaluating the CFL condition. Thus, for computational convenience, we choose $c_0 = 20$ in this study.

The values of the non-slip boundary conditions at the in- and outflow boundaries $\boldsymbol{q}_\infty = (\boldsymbol{u}_\infty, p_\infty)$, as well as the values of the boundary conditions at the cylinder $\boldsymbol{u}_s$ are implemented with volume penalization. Therefore, we add additional penalisation terms to the Eqs. (2.44) and (2.48):

$$\partial_t \boldsymbol{u} + (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u} + \boldsymbol{\nabla} p - \nu\boldsymbol{\nabla}^2\boldsymbol{u} + \frac{\chi}{C_\chi}(\boldsymbol{u} - \boldsymbol{u}_s) + \frac{\chi_{\mathrm{sp}}}{C_{\mathrm{sp}}}(\boldsymbol{u} - \boldsymbol{u}_\infty) = 0 \tag{2.49}$$

$$\partial_t p + c_0^2\boldsymbol{\nabla}\cdot\boldsymbol{u} + \frac{\chi_{\mathrm{sp}}}{C_{\mathrm{sp}}}(\boldsymbol{p} - \boldsymbol{p}_\infty) = 0\,, \tag{2.50}$$

where the mask functions χ, χ_{sp} parametrize the penalized volumes and $C_\chi, C_{\mathrm{sp}} > 0$ are the porosity parameters, as explained in Section 2.1.4. In our simulation domain (shown in Fig. 2.17) we distinguish between a so-called sponge region ($\mathbb{D}_{\mathrm{sp}}$) and a solid ($\mathbb{D}_{\mathrm{s}}$) region, at which we set up different boundary conditions. In the solid region, we set $\boldsymbol{u}_{\mathrm{s}} = \dot{\boldsymbol{\Delta}}(t)$

Name	Value
simulation time	$T = 1000$
domain size	$\mathbb{D} = [0, 64] \times [0, 16]$
viscosity	$\nu = 10^{-2}$
artificial speed of sound	$c_0 = 20$
thickness of sponge	$L_{\text{sp}} = 1$
porosity sponge	$C_{\text{sp}} = 2.5 \cdot 10^{-3}$
inflow speed	$u_\infty = 0.925$
cylinder radius	$R = 1$
wavelet threshold	$\epsilon = 10^{-4}$
Block data	$(J_{\max}, B_x, B_y, g) = (5, 65, 17, 6)$

Table 2.6: Parameters of the fixed cylinder simulation at Re = 185.

if $\boldsymbol{\Delta}(t) \in \mathbb{R}^2$ defines the time-dependent shift of the solid body. The penalization strength C_χ is calculated with Eq. (2.33) using $K_\chi = 0.365$ as suggested by [46]. Note, that according to [46] the order of the numerical scheme is reduced to $\mathcal{O}\left(h^2\right)$ for the penalized system. The sponge region implements non-reflecting boundary conditions. They are used to remove the wake caused by the cylinders and create a constant inflow with u_∞ in the x-direction. The sponge region is set up as suggested by [46]. For all simulations we make use of the CFL condition:

$$\Delta t = K_{\text{CFL}} \min \left(\frac{h}{\|\boldsymbol{u}\|_2 + \sqrt{\|\boldsymbol{u}\|_2^2 + c_0^2}}, C_\chi, C_{\text{sp}}, \frac{h^2}{2\nu D} \right) \tag{2.51}$$

with $K_{\text{CFL}} = 1$. A characteristic quantity for incompressible flows is the Reynolds number defined in terms of the cylinder diameter $2R$:

$$\text{Re} = \frac{2Ru_\infty}{\nu} \,. \tag{2.52}$$

Fixed Cylinder at Re = 185

In the first example, we compute the solution for a single fixed cylinder placed at $x_0, y_0 = (8, 8)$ in a rectangular domain $\mathbb{D} = [0, 64] \times [0, 16]$.

The vortex shedding (shown in Fig. 2.5) causes a periodic oscillation in the lift and drag forces of the cylinder, which are well studied in the literature. Lift and drag coefficients $C_{\mathrm{L}}, C_{\mathrm{D}}$ are defined as

$$(C_{\mathrm{D}}, C_{\mathrm{L}}) := \frac{2(F_1, F_2)}{\rho u_\infty^2 l} \quad \text{with} \quad \boldsymbol{F} = C_\chi^{-1} \int_{\mathbb{D}} \chi \rho (\boldsymbol{u} - \dot{\boldsymbol{\Delta}}(\boldsymbol{\mu}))\, \mathrm{d}^3\boldsymbol{x}\,. \tag{2.53}$$

Here $l = 2R, \rho = 1$ are the diameter of cylinder and density of the fluid. The formula for the aero-dynamic forces $\boldsymbol{F} = (F_1(\boldsymbol{x}, t, \mu), F_2(\boldsymbol{x}, t, \mu))$ is taken from [44]. The evolution of the lift and drag coefficients is shown for $0 \leq t \leq 500$ in Fig. 2.16a. From the lift and drag coefficients we calculate the mean drag

$$\overline{C}_{\mathrm{D}} = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} C_{\mathrm{D}}(t) \mathrm{d}t\ , \tag{2.54}$$

root mean square (RMS) of the lift

$$\hat{C}_{\mathrm{L}} = \sqrt{\frac{1}{t_2 - t_1} \int_{t_1}^{t_2} (C_{\mathrm{L}}(t))^2 \mathrm{d}t} \tag{2.55}$$

and Strouhal number

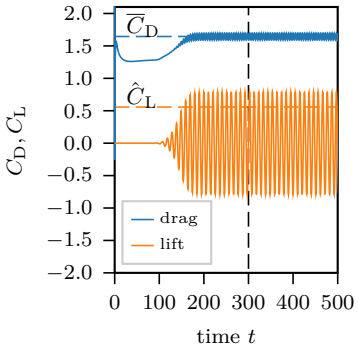
$$\mathrm{St} = \frac{l f_0}{u_\infty}\,, \tag{2.56}$$

as the dimensionless frequency of the vortex shedding. We calculate St from the time evolution of the lift. With help of its Fourier spectra, shown in Fig. 2.16b, we determine St from the frequency with the highest amplitude. For the statistics we are using all time samples $t_1 \leq t \leq t_2$ with $t_1 = 3T/10$, $t_2 = T$ in the oscillatory stage. The results are compared to literature values in Tables 2.7 and 2.8.

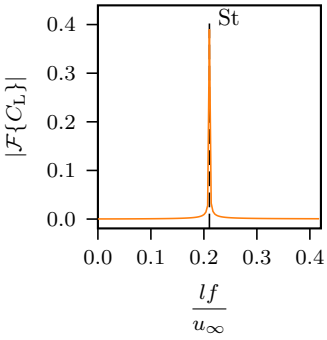
Path Optimization of a Moving Cylinder at $\mathrm{Re} = 200$

In the last part of this section, we introduce a typical fluid dynamic application of a TDFS, which will be revisited in Chapters 4 and 5. In the simulations, two cylinders of radius R are placed inside a rectangular domain $\mathbb{D} = [0, L]^2$, $L = 64R$ shown in Fig. 2.17.

The cylinder closest to the inflow is called the leader and the cylinder placed further downstream is called chaser in the following. This setup



(a) Lift and drag coefficients (C_D, C_L) for $\mathrm{Re} = 185$. The root mean square $\hat{C}_L$ and mean $\overline{C}_D$ are drawn as dashed lines.

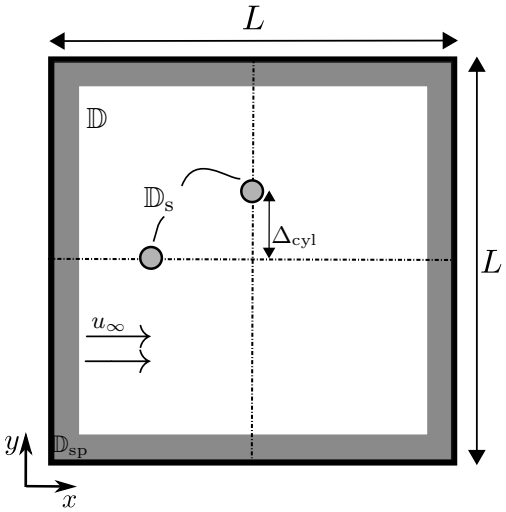


(b) Amplitude of the Fourier transformed temporal evolution of the lift coefficient $\mathcal{F}\{C_L\}$ for $t \geq 300$ vs. dimensionless frequency.

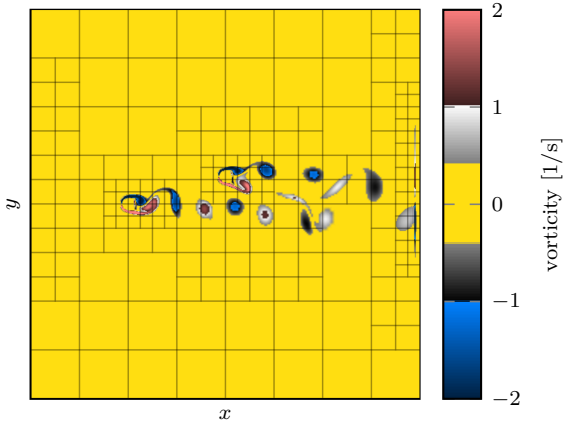
Figure 2.16: Statistical quantities $(\overline{C}_D, \hat{C}_L, \mathrm{St}$ defined in Eqs. (2.54) to (2.56)) of a fixed cylinder at $\mathrm{Re} = 185$ computed with $\epsilon = 10^{-4}$.

	$\overline{C_D}$	$\widehat{C_L}$	St
Lu et. al [94]	1.31	0.422	0.195
Liu et. al [92]	1.289	0.451	0.197
Guilmineau et. al [65]	1.287	0.443	0.195
Experimental results [65]	1.28		0.19
Khalili et. al [80]	1.282	0.431	0.191
literature average	1.29(1)	0.44(1)	0.194(3)
present study $\epsilon = 10^{-4}$	1.64	0.55	0.21
present study equi	1.63	0.54	0.21

Table 2.7: Comparison of the statistical values for the drag and lift defined in Eqs. (2.54) and (2.55) and Strouhal number defined in Eq. (2.56).



(a) Simulation domain



(b) Vorticity field

Figure 2.17: Simulation of the two moving cylinders:
a) Simulation domain with penalized volumes drawn in grey. The penalized volumes are the sponge region $\mathbb{D}_{\text{sp}}$ at the outer edges of the domain and the solid region $\mathbb{D}_{\text{s}}$ that implements the cylinders. In the setup we have a fixed cylinder located at $(x, y) = (L/4, L/2)$ and a moving cylinder at $(x_2, y_2) = (L/2, L/2 + \Delta_{\text{cyl}}(t; \mu))$.
b) Wavelet adapted vorticity field $\omega = \partial_x u_2 - \partial_y u_1$ of the velocity $\boldsymbol{u} = (u_1, u_2)$ during the simulation.

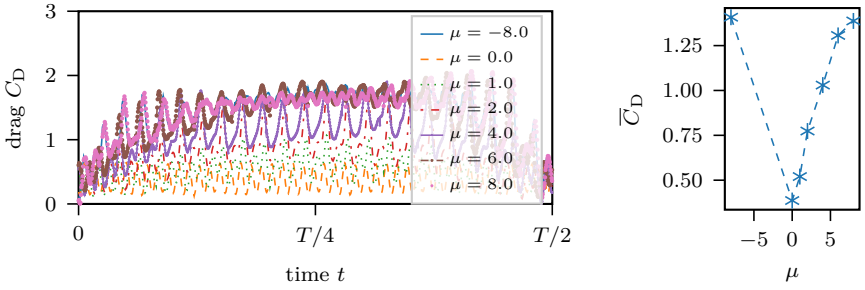
	adaptive		equidistant		
$J_{\max}$	DOF	t_{CPU} (s)	DOF	t_{CPU} (s)	speedup
6	974848	7.6e+06	4194304	2.0e+07	2.6

Table 2.8: CPU-time comparison using `Intel(R) Xeon(R) CPU E5-26090` CPUs.

is inspired by biolocomotion, where the leader is followed by a chaser in a free stream flow of uniform velocity u_∞. Specifically, in biolocomotion the interaction between animals in close proximity is studied to understand the swarm behavior of animals, like fish or birds [137, 70]. One may ask if there is a physics understanding, like energy minimization or a biological reason (breeding, defense, etc.) for this behavior. To investigate this from a physics standpoint, a leading cylinder is placed at a fixed position $(x_1, y_1) = (L/4, L/2)$ in a uniform flow at $\text{Re} = 200$ and the chaser further downstream $(x_2, y_2) = (L/2, L/2 + \Delta_{\text{cyl}}(t; \boldsymbol{\mu}))$ is shifted along a vertical path $\Delta_{\text{cyl}}(t; \boldsymbol{\mu})$ that is time-dependent. One can parametrize the path for instance in a sine series :

$$\Delta_{\text{cyl}}(t; \boldsymbol{\mu}) = \sum_{k=1}^{N_\mu} \mu_k \sin(2\pi f_k t), \quad \text{with} \quad \boldsymbol{\mu} = (\mu_1, \dots, \mu_{N_\mu}) \tag{2.57}$$

For simplicity, we aim to find an optimized path $\Delta_{\text{opt}}(t) = \Delta(t; \boldsymbol{\mu}_{\text{opt}})$ such that the mean drag $\overline{C}_{\text{D}}$ of the second cylinder is minimized. For this simple showcase study we choose $N_\mu = 1$, i.e. $\Delta_{\text{cyl}}(t; \mu) = \mu \sin(2\pi f_1 t)$ and $f_1 = 10 f_{\text{wake}} = 0.2 \times 10^{-2} s^{-1}$, where f_{wake} is calculated from the Strouhal number Eq. (2.56) of the leading cylinder. We sample the drag Eq. (2.53) in every time step over one period $T = 500 = 1/f_1$ in an interval $[3T, 4T]$ for seven different trajectories with amplitude $\mu = \{-8, 0, 1, 2, 4, 6, 8\}$. The evolution of the drag is shown in Fig. 2.18 together with the calculated mean $\overline{C}_{\text{D}}$ (Fig. 2.18b). Trivially, in this example, the drag is minimal if the chaser stays constant behind the leader ($\mu = 0$), because of the wind shadow zone with reduced vertical velocity. However, for increasing μ, the drag will increase until it saturates because the maximal drag is achieved in the free stream flow. Note that the mean drag is symmetric in μ because of the symmetry in the system.



(a) Time evolution of the drag **(b)** Mean drag

Figure 2.18: Drag evolution (a) and mean drag (b) for different maximal amplitudes μ of the vertical displacement $\Delta_{\text{cyl}}(t;\mu) = \mu \sin(2\pi f_1 t)$.

Since the simulations of multiple trajectories can become very costly when trying to optimize $\Delta_{\text{cyl}}(t;\boldsymbol{\mu})$, especially for a larger parameter space $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^{N_\mu}$ with $N_\mu > 1$, it is desirable to have a reduced system that is cheap to evaluate and can approximate the dynamics well to predict new system states. The generation of such a reduced order model is addressed in the following chapters.

2.3 Summary

In this chapter, we have introduced a block-based adaptive simulation software for transport-dominated flows. The method combines multiresolution (MR) analysis using CDF 4,4 wavelets with a fourth-order finite difference scheme. The MR allow us to represent transport dominated flow fields in a sparse way while controlling the introduced compression errors. The adaptive method we employ allows us to solve transport-dominated systems more efficiently compared to equidistant finite difference methods. This was numerically investigated for the pure advection of a disc, reacting fronts, and incompressible vortex shedding with moving objects. For the representative examples, if compression and finite difference truncation errors are balanced, the relative errors in space scale with $\mathcal{O}\left(h^4\right)$ of the smallest possible lattice spacing h. Therefore, the convergence of the utilized numerical finite difference

scheme is preserved. The comparison to a non-adaptive scheme on the finest level shows that, for the same precision, required resources both in memory and CPU-time are reduced.

However, the presented approach has some weaknesses, which should be addressed in future research. As investigated in Section 2.1.3, the stability of the resulting numerical scheme cannot be guaranteed from a theoretical standpoint. Based on this we recommend combining the method with summation-by-parts finite difference methods [130] that enable stability by conservation properties. Another opportunity for improvement is the precision of the solution in the presence of volume penalized boundary conditions. As shown by the numerical studies of the ACM equations in Section 2.2.3, drag and lift coefficients seem to disagree with common literature values and the overall convergence of the numerical scheme is reduced to $\mathcal{O}\left(h^2\right)$. The latter may be addressed in a similar fashion as proposed in [56] for the time-dependent Maxwell equation. Here the convergence is improved by constructing a penalty term that is a continuous extension of the electrical field.

Although the numerical scheme delivers savings in computing time, these savings are not sufficient for extensive parameter studies including millions of trajectory evaluations. Therefore, we extend the existing framework to include model order reduction techniques. Here the introduced block-based data structure poses challenges for classical model order reduction. This issue is addressed in Chapter 3.

In addition to classical model order reduction procedures, we further present non-linear model order reduction techniques tailored for transport dominated systems in Chapter 4 for the numerical examples presented in this section.

3 Dimension Reduction on Linear Subspaces

As seen in the previous section numerical simulations of fluid flows yield high dimensional data sets, which need to be pre-processed to a lower dimensional manifold, in which the essential information is captured and can be used for prediction and optimization (see Chapter 5). Most commonly in fluid dynamics the data is reduced to a low dimensional linear subspace using *proper orthogonal decomposition* (POD). In this chapter we therefore review and compare three techniques, which can be used to compute the POD for very large data: The singular value decomposition, its randomized version and the wavelet adaptive POD (wPOD). The methods will be the core technique of the non-linear dimension reduction methods explained in Chapter 4. The results presented here are taken from [148].

3.1 Proper Orthogonal Decomposition (POD)

The proper orthogonal decomposition is a powerful tool to find a lower-dimensional approximation of a flow field $\boldsymbol{q}$. For this chapter, we assume that the flow field is a continuous vector-valued L^2-function $\boldsymbol{q}(\boldsymbol{x}, t) \in \mathbb{R}^K$, $K > 0$ that has been sampled for different time instances $\{t_i\}_{i=1}^{N_t}$. The samples are called *snapshots* and are in the following indexed by $\boldsymbol{q}_i(\boldsymbol{x}) = \boldsymbol{q}(\boldsymbol{x}, t_i)$. In MOR, the snapshot data are collected from a simulation of an evolutionary or parametrized PDE, as for example the Von Kármán Vortex Street, that has been studied in previous Section 2.2.3. From the collected snapshot data, POD aims to find a

low dimensional description by approximating the snapshots in terms of a few orthogonal basis functions $\boldsymbol{\psi}_k(\boldsymbol{x}) \in \mathbb{R}^K$:

$$\boldsymbol{q}(\boldsymbol{x}, t_i) \approx \tilde{\boldsymbol{q}}_i(\boldsymbol{x}) := \sum_{k=1}^{r} a_k(t_i)\boldsymbol{\psi}_k(\boldsymbol{x}) \qquad r \ll N_t\,. \tag{3.1}$$

These basis functions are usually called *modes*. Note, that in contrast to the wavelet representation introduced in Section 2.1.1, which uses local basis functions, the POD-basis functions are global basis functions, reflecting the dynamics of the flow. They are thus problem specific because they have to be determined from data. Since this chapter makes no explicit use of the wavelet representation, it should be further noted that we use the same symbol for the POD-basis functions as for the wavelet basis. The POD-modes are defined as the solution of the optimization problem

$$\min_{\{\boldsymbol{\psi}_k\}} \sum_{i=1}^{N_t} \left\| \boldsymbol{q}_i - \sum_{k=1}^{r} \langle \boldsymbol{q}_i, \boldsymbol{\psi}_k \rangle \boldsymbol{\psi}_k \right\|_2^2, \quad \text{such that} \quad \langle \boldsymbol{\psi}_k, \boldsymbol{\psi}_l \rangle = \delta_{kl}\,, \tag{3.2}$$

with the L^2 inner product $\langle \cdot, \cdot \rangle$ and associated norm $\|\cdot\|_2 = \sqrt{\langle \cdot, \cdot \rangle}$. In fluid dynamics Eq. (3.2) is usually solved with the *method of snapshots or strobes* [127] since for data where the spatial resolution is much larger then the number of snapshots N_t, Eq. (3.2) is reduced to a small eigenvalue problem of size $N_t \times N_t \sim \mathcal{O}\,(100)$

$$\mathbf{C}\mathbf{v}_k = \lambda_k \mathbf{v}_k \quad \text{for} \quad k = 1, \ldots, r\,, \tag{3.3}$$

for the correlation matrix

$$\mathbf{C}_{ij} = \frac{1}{N_t V} \langle \boldsymbol{q}_i, \boldsymbol{q}_j \rangle\,, \tag{3.4}$$

together with the relation:

$$\boldsymbol{\psi}_k = \frac{1}{\sqrt{\lambda_k N_t}} \sum_{i=1}^{N_t} (\boldsymbol{v}_k)_i \boldsymbol{q}_i \quad k = 1, \ldots, r\,. \tag{3.5}$$

The method of snapshots is strongly connected to the *singular value decomposition* (SVD) [138], because left singular vectors correspond to

the solution of Eq. (3.2) and right singular vectors to $\boldsymbol{v}_k$, when assuming Euclidean space [138]. Respectively, the eigenvalues $\lambda_k = \sigma_k^2 \geq 0$ are squares of the singular values. Furthermore, it is known from the Eckart-Young-Mirsky theorem [42] that the resulting approximation error in the Frobenius norm, when truncating after the rth mode, is given by the sum $\sum_{k=r+1}^{N_t} \sigma_k^2$ of the remaining singular values.

However, caution must be taken when using this method instead of the SVD, because the condition number $\kappa(\mathbf{Q}) := \sigma_{\max}(\mathbf{Q})/\sigma_{\min}(\mathbf{Q})$ of the associated snapshot matrix $\mathbf{Q}$ is squared: $\kappa(\mathbf{Q}^\top\mathbf{Q}) = \kappa(\mathbf{Q})^2$. This can lead to inaccuracy of POD modes with small singular values (see the famous example of Lächli [88]). Nevertheless, one is often willing to accept this potential error in favor of a smaller problem size.

Another way of reducing the problem size, without squaring the condition number, is using a *randomized singular value decomposition* (rSVD) algorithm, which is outlined in [68]. Here, a matrix $\hat{\mathbf{Q}} \in \mathbb{R}^{M\times l}$, $l \ll N_t$ is formed with l orthogonal columns, that approximates the column space of $\mathbf{Q} \approx \hat{\mathbf{Q}}\mathbf{B}$. With its help, a singular value decomposition of the small $l \times N_t$ matrix $\mathbf{B} = \hat{\mathbf{Q}}^\top\mathbf{Q}$ is computed and the left singular vectors $\boldsymbol{\psi}_k^{\mathbf{B}}$ are projected back onto the full space via: $\boldsymbol{\psi}_k = \hat{\mathbf{Q}}\boldsymbol{\psi}_k^{\mathbf{B}}$. Usually, the $M \times l$ orthogonal matrix is formed by a QR decomposition taking l random samples of the column space of $\mathbf{Q}$. For a target number of r modes one usually oversamples $l = r + n$ by taking $n = 5$ or $n = 10$ additional random samples [68]. However, if the singular values decay slowly, $\hat{\mathbf{Q}}$ may not represent $\mathbf{Q}$ well enough and costly tricks, like *Power Iterations* using additional passes over the data, have to be applied [68]. Moreover, as pointed out in [68] for very large matrices $\mathbf{Q}$ the data cannot be loaded into fast memory and therefore the transfer from the slow memory typically dominates the arithmetic. In contrast, the wPOD algorithm presented in the next section seeks to avoid these problems by reducing the relevant information of each single snapshot to fit it into the fast memory.

3.2 POD on Wavelet Adaptive Grids

3.2.1 Technical Preliminaries

In the following, we discuss the numerical methods that are used in the wPOD algorithm and give detailed insight into its implementation, when handling multiple block-based adaptive grids. The basic wavelet adaptation technique used for our algorithm has been already discussed in Section 2.1.1. We hence limit the presentation here to changes specific to our algorithm. In the interest of readability, we will assume two-dimensional data ($D = 2$).

Block Structured Grid and Implementation

Multiresolution representations require a dedicated data structure. Here, spatial data is divided into a set of nested blocks, which are organized in a tree. We use a collection of trees, which we call a forest, to store multiple snapshots together with their designated tree simultaneously.

Forest Composed of Multiple Trees: For the administration of all blocks in the forest, we use a *multi-tree structure* illustrated in Fig. 3.1. Here each tree $\mathcal{T}_i$ holds a collection of blocks $\mathcal{B}_p$ and block values $\boldsymbol{q}^{(p)}$

$$\mathcal{T}_i = \{(\mathcal{B}_p, \boldsymbol{q}^{(p)}(\boldsymbol{x})) \mid \boldsymbol{x} \in \mathcal{B}_p, p \in \Lambda_i^j, j = J_{\min}, \dots, J_{\max}\}\,, \tag{3.6}$$

where a block is a leaf at the end of a branch, which can be uniquely identified by a tree code $p \in \Lambda_i^j$ and a tree ID i. The tree ID identifies the corresponding grid Ω_i and the tree code determines the topology, such as block position and lattice spacing. In the following, the collection of trees is called *forest*, given by

$$\mathcal{F} = \{\mathcal{T}_i \mid i = 1, \dots, N_{\text{tree}}\}\,. \tag{3.7}$$

In Section 3.2.2 we will use $\mathcal{F}$ to hold multiple spatial fields as time or parameter samples of the wPOD algorithm.

Light and Heavy Data Storage: To distinguish between administrative and physical information, we separate our data structures into light and heavy data.

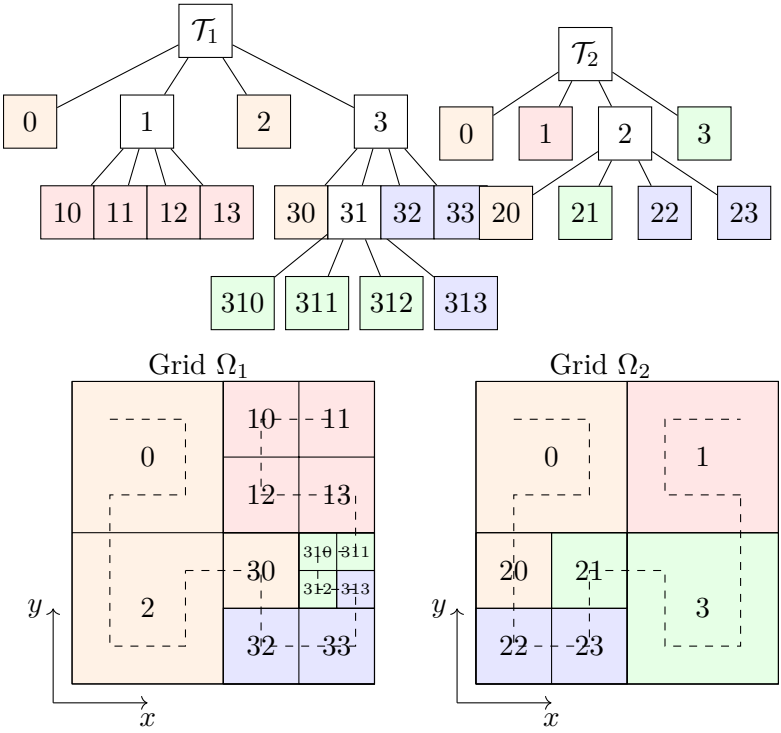


Figure 3.1: A forest of two trees and their associated grids Ω_i. The top part visualizes the tree structure for tree $i = 1, 2$. Colored leaves at the end of a branch correspond to blocks on the grid as shown in the bottom part. The block color encodes the processor that holds the block. The blocks are distributed among processors using space-filling curves (dashed lines).

The *light data* (`lgt_n`, `lgt_active`, `lgt_block`) is the minimal information necessary to organize the topology of our grids. From the light data, we can determine neighbor relations between the blocks and keep track of the processors holding the block. It is therefore synchronized among all processing units. Figure A.2 in Appendix A.3 illustrates an example of light data in the case of the tree structure shown in Fig. 3.1. All light data are stored in the `lgt_block` array. Each row holds the necessary information of one block, such as tree ID, grid refinement level, tree code and *refinement status/coarsening indicator* (see Appendix A.3). The row index is called light-ID (`lgt_id`). It is ordered lexicographically in the process-ID: $\texttt{lgt\_id} = (i_{\text{proc}}-1)N_{\text{blocks}}+j$, with $j = 1, \ldots, N_{\text{blocks}}$. In this way, we relate the position of the block with the process-ID i_{proc}. During the execution of the algorithm, blocks can be created or deleted. To avoid expensive memory allocation, we set a block inactive by marking the rows in `lgt_block` with -1. From `lgt_block` we compute active block lists (`lgt_active`) for each tree. In this way, we can manipulate blocks on different grids efficiently, by only looping over the active lists of a given tree specified by its `tree_id` (compare with Fig. A.2).

Besides the light administrative structure, we have to store large data fields with the physical information of the state vector quantity and all neighbor relations of the blocks. These data are called *heavy data* and they are equally distributed among the processors using the index of the space-filling curve (Hilbert/Z-curve) [145]. The index can be easily calculated from the tree code. In Fig. 3.1 the *Hilbert-curve* is visualized with the dashed line passing through the lattice and the processors-IDs are encoded with the color of the block.

Using *space filling curves* has two major advantages for our algorithm. Firstly, due to the locality properties of the space-filling curve, the communication between adjacent blocks which do not share the same processors is kept at a minimum. Secondly, the uniqueness of the curve ensures that trees with the same tree structure (i.e. same grid Ω_i) have identical processor distribution. This is advantageous when performing pointwise operations between trees.

The Weighted Inner Product and Pointwise Tree Operations

In the wPOD algorithm, we follow the same approach as in the method of snapshots explained in the previous section (Section 3.1). However, we use the sparsity of our wavelet block-based data representation to allow for a memory-efficient computation of the POD basis. In contrast to the representation in terms of the snapshot matrix, as used by the SVD, our data is represented in terms of a forest $\mathcal{F}$ or a collection of trees (see Section 3.2.1). Here each snapshot $\boldsymbol{q}_i(\boldsymbol{x}) \coloneqq \boldsymbol{q}(\boldsymbol{x}, t_i)$ is associated with a tree $\mathcal{T}_i$ on a hierarchical structured multiresolution grid Ω_i, $i = 1, \ldots, N_t$. The leaves of the tree correspond to blocks, where each block p stores coefficients $\{\underline{\boldsymbol{q}}^j[p, k_1, k_2]\}_{k_1,k_2}$ (see def. in Eq. (2.6)) of the underlying basis $\{\phi^j_{p,k_1,k_2}\}_{k_1,k_2}$ at tree level j. The interpolating basis allows to represent the data in a continuous form, when summing over all blocks $p \in \Lambda_i^j$ of each tree level j:

$$\boldsymbol{q}_i(\boldsymbol{x}) = \sum_{j=1}^{J_{\max}} \sum_{p \in \Lambda_i^j} \boldsymbol{q}_i^{(p)}(\boldsymbol{x}) \quad \text{for} \quad i = 1, \ldots, N_t \tag{3.8}$$

Here the components of $\boldsymbol{q}_i^{(p)}$ are interpolated on the block p using Eq. (2.9). We can now introduce the *snapshot set*:

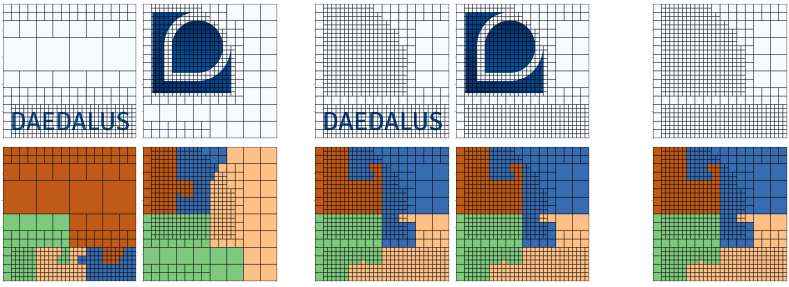
$$\mathcal{Q} = \{\boldsymbol{q}(\boldsymbol{x}, t_1), \ldots, \boldsymbol{q}(\boldsymbol{x}, t_{N_t}) \mid \boldsymbol{x} \in \mathbb{D}\}\,, \tag{3.9}$$

as the continuous counterpart of the snapshot matrix $\mathbf{Q}$. As mentioned earlier, our algorithm is capable of handling 2D and 3D data fields on a rectangular domain $\mathbb{D} \subset \mathbb{R}^D$, $D \in \{2, 3\}$. Note that with the new data representation in terms of functions in the L^2 Hilbert space, the inner product in the POD formulation has changed to

$$\langle \boldsymbol{q}_i, \boldsymbol{q}_j \rangle \coloneqq \int_{\mathbb{D}} \boldsymbol{q}_i^\top(\boldsymbol{x}) \boldsymbol{q}_j(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}\ . \tag{3.10}$$

However, for inner products or any pointwise operation (+,-) between snapshots $\boldsymbol{q}_i$,$\boldsymbol{q}_j$, represented on locally different grids Ω_i, Ω_j, it is required that both coefficient vectors $\underline{\boldsymbol{q}}_i, \underline{\boldsymbol{q}}_j$ are of the same length, i.e. expressed in the same basis. In contrast to the discussed FEM methods [134, 62], this can be achieved very efficiently because the hierarchical grid definition allows merging two grids by the union $\Omega_{ij} = \Omega_i \cup \Omega_j$ for

the two snapshots involved. Figure 3.2 visualizes the grid merging procedure. In Fig. 3.2a, the initial grids Ω_i and Ω_j are displayed together with their processor distribution. In this example, both trees have fundamentally different tree structures and processor distributions. In areas where Ω_i has large details, Ω_j does not and vice versa. The union of both grids aims to merge them, such that no detail of both trees gets lost. This implies that merging both trees only involves refinement operations, which are cheap when using wavelet up-sampling. As explained in Section 3.2.1 trees with identical tree structure have identical processor distribution because of our load-balancing strategy by space-filling curves. The `hvy-data`, i.e. grid quantities of Ω_i and Ω_j are therefore on the same processor (see Fig. 3.2b) after unification. The unification enables us to calculate the inner product in Eq. (3.10)



(a) $q_i(\Omega_i)$ and $q_j(\Omega_j)$ **(b)** $q_i(\Omega_{ij})$ and $q_j(\Omega_{ij})$ **(c)** $(q_i \times q_j)(\Omega_{ij})$

Figure 3.2: Visualization of pointwise operations on multiresolution grids Ω_i and Ω_j. On the top row the scalar fields q_i and q_j are shown, where blue colors represent 1 and white colors 0. Below each field the corresponding processor distribution grid is displayed. Each color of the processing grid represents one of the four processing units. Initial trees (Figure 3.2a), unified tree structures (Figure 3.2b) and the processed field (Fig. 3.2c).

as a weighted inner product

$$\langle \boldsymbol{q}_i, \boldsymbol{q}_j \rangle = \langle \underline{\boldsymbol{q}}_i, \underline{\boldsymbol{q}}_j \rangle_{\mathbf{M}_{\mathrm{mass}}} = \underline{\boldsymbol{q}}_i^{\top} \mathbf{M}_{\mathrm{mass}} \underline{\boldsymbol{q}}_j, \tag{3.11}$$

where $\underline{\boldsymbol{q}}_i$ represent the vectorized entries of tree i and $\mathbf{M}_{\mathrm{mass}}$ is a positive definite and symmetric matrix (explicitly given in Appendix A.1). In FEM literature this matrix is often called *mass matrix* and has been

used by [134, 62] in a similar approach. With Eq. (3.11) we can calculate the inner product and the associated norm of our block-based multiresolution fields. Replacing the L^2 inner products by weighted inner products therefore generalizes the POD minimization problem in Eq. (3.2) to multiresolution fields. With this intermediate result, we can go one step further to combine wavelet adaptation and POD truncation.

3.2.2 The wPOD Algorithm

The wPOD algorithm proceeds in the following steps:

1. **Read and Coarsen Data** $\boldsymbol{q}_i^\epsilon \leftarrow \texttt{adapt}(\boldsymbol{q}_i, \epsilon, J_{\min}, J_{\max})$
   In the first step of the algorithm, we read all block decomposed snapshots in $\mathcal{Q}$ and coarsen them for a given threshold ϵ using the wavelet adaptation scheme, if the input fields are not already adapted. The adapted fields are denoted by $\boldsymbol{q}^\epsilon$. This part of the algorithm is essential because it allows keeping only the most relevant information (see Section 2.1.1) of the input data using wavelet compression and therefore makes handling of large data feasible (see Section 3.3).

2. **Computation of Correlation Matrix** $(\mathbf{C})_{ij}^\epsilon = \frac{1}{N_t V}\langle \boldsymbol{q}_i^\epsilon, \boldsymbol{q}_j^\epsilon \rangle$
   The main computational effort of the algorithm is the construction of all elements of the correlation matrix $\mathbf{C}^\epsilon \in \mathbb{R}^{N_t \times N_t}$, for which the inner product of locally different resolved snapshots needs to be computed. For any pairwise operation $(+, -, \langle\cdot,\cdot\rangle)$, we refine to a union of both grids as shown in Fig. 3.2. After the operations (+,-) the resulting field is adapted again to the predefined threshold ϵ.

3. **Solving the Eigenvalue Problem** $\mathbf{C}^\epsilon \boldsymbol{v}_k^\epsilon = \lambda_k^\epsilon \boldsymbol{v}_k^\epsilon$
   After the correlation matrix $\mathbf{C}^\epsilon$ is constructed, we diagonalize it with Jacobi's method for real symmetric matrices `DSYEV` implemented in `LAPACK` [2]. As described in [2] sec. 8, the chosen method computes all the eigenvalues and eigenvectors close to machine precision. We therefore neglect errors made during the diagonalization. In contrast to the construction of $\mathbf{C}^\epsilon$, that scales with the average number of grid points M, the computational ef-

fort needed for diagonalization of $\mathbf{C}^\epsilon$ is relatively small, since it scales with $\mathcal{O}(N_t), N_t \ll M$ (see CPU-time comparison in Fig. A.3).

4. **Construction of POD Modes** $\boldsymbol{\psi}_k^\epsilon = \frac{1}{\sqrt{\lambda_j^\epsilon N_t}} \sum_{i=1}^{N_t} (\boldsymbol{v}_k^\epsilon)_i \boldsymbol{q}_i^\epsilon$
The elements of the correlation matrix are the inner products between two snapshots, i.e. the i-th row/column contains the coefficients of $\boldsymbol{q}_i$ represented by a linear combination of all snapshots in $\mathcal{Q}$. Diagonalizing $\mathbf{C}^\epsilon$ means finding a basis of coefficient vectors $\boldsymbol{v}_k^\epsilon \in \mathbb{R}^{N_t}$ which generate an optimal representation of $\mathcal{Q}$. The representation in terms of orthonormal modes $\{\boldsymbol{\psi}_k^\epsilon\}$ is computed according to Eq. (3.5). The summation in Eq. (3.5) proceeds in multiple steps. In the first step we copy $\boldsymbol{\psi}_k^\epsilon \leftarrow (\boldsymbol{v}_k^\epsilon)_1 \boldsymbol{q}_1^\epsilon$, after which we iteratively sum up $\boldsymbol{\psi}_k^\epsilon \leftarrow \boldsymbol{\psi}_k^\epsilon + (\boldsymbol{v}_k^\epsilon)_i \boldsymbol{q}_i^\epsilon$ for $i = 2, \ldots, N_t$ in the second step and divide by the normalization factor.

5. **Computation of POD Mode Coefficients** $a_{ki}^\epsilon = \frac{1}{V} \langle \boldsymbol{\psi}_k^\epsilon, \boldsymbol{q}_i^\epsilon \rangle$
The computation of the mode coefficients involves again a computation of the inner product between the orthonormal modes $\boldsymbol{\psi}_k^\epsilon$ and the snapshots $\boldsymbol{q}_i^\epsilon$. In most cases, this step needs fewer evaluations of the scalar product, since the number of modes r should be small, $r \ll N_t$.

In summary, our algorithm generates sparse modes $\boldsymbol{\psi}_k^\epsilon$, $k = 1, \ldots, r$, with amplitudes a_{ki}^ϵ to approximate any of the snapshots $\boldsymbol{q}_i \in \mathcal{Q}$ in terms of a linear subspace

$$\boldsymbol{q}_i(\boldsymbol{x}) \approx \tilde{\boldsymbol{q}}_i^\epsilon(\boldsymbol{x}) = \sum_{k=1}^{r} a_{ki}^\epsilon \boldsymbol{\psi}_k^\epsilon(\boldsymbol{x}) \quad \text{for} \quad i = 1, \ldots, N_t \,. \tag{3.12}$$

In this notation, the upper index ϵ denotes the quantities, which are indirectly affected by the wavelet threshold (e.g. $a_{ki}^\epsilon, C_{ij}^\epsilon$) or directly expressed as a truncated wavelet series (e.g. $\tilde{\boldsymbol{q}}_i^\epsilon(\boldsymbol{x}), \boldsymbol{\psi}_k^\epsilon(\boldsymbol{x})$). Furthermore, $\tilde{\boldsymbol{q}}$ denotes the truncation after the r-th mode.

3.2.3 Error Estimation

Here, we discuss the dependency of the approximation error on the wavelet threshold ϵ and the truncation rank r. Additionally, we explain how to choose both values to obtain a given accuracy.

To do so, we provide an error estimate of the approximation $\tilde{\boldsymbol{q}}_i^\epsilon$ in Eq. (3.12). The approximation projects our data $\mathcal{Q} = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{N_t}\}$ onto a linear subspace spanned by a set of modes $\{\boldsymbol{\psi}_k^\epsilon\}_{k=1,\ldots,r}$. The sparsity of the modes is determined by the wavelet threshold ϵ (see Section 2.1.1) and the dimension of the subspace r shall be much smaller than the number of snapshots: $r \ll N_t$. For given r, ϵ we define the relative error of our approximation in the L^2-norm,

$$\mathcal{E}_{\text{wPOD}}(r,\epsilon) := \frac{\sum_{i=1}^{N_t} \|\boldsymbol{q}_i(\boldsymbol{x}) - \tilde{\boldsymbol{q}}_i^\epsilon(\boldsymbol{x})\|_2^2}{\sum_{i=1}^{N_t} \|\boldsymbol{q}_i(\boldsymbol{x})\|_2^2} \tag{3.13}$$

which can be split into two contributions, i.e. compression and truncation errors. We thus have,

$$\sum_{i=1}^{N_t} \|\boldsymbol{q}_i(\boldsymbol{x}) - \tilde{\boldsymbol{q}}_i^\epsilon(\boldsymbol{x})\|_2^2 = \sum_{i=1}^{N_t} \|\boldsymbol{q}_i(\boldsymbol{x}) - \boldsymbol{q}_i^\epsilon(\boldsymbol{x}) + \boldsymbol{q}_i^\epsilon(\boldsymbol{x}) - \tilde{\boldsymbol{q}}_i^\epsilon(\boldsymbol{x})\|_2^2 \tag{3.14}$$

$$\leq \underbrace{\sum_{i=1}^{N_t} \|\boldsymbol{q}_i(\boldsymbol{x}) - \boldsymbol{q}_i^\epsilon(\boldsymbol{x})\|_2^2}_{\text{compression error}} + \underbrace{\sum_{i=1}^{N_t} \|\boldsymbol{q}_i^\epsilon(\boldsymbol{x}) - \tilde{\boldsymbol{q}}_i^\epsilon(\boldsymbol{x})\|_2^2}_{\text{POD truncation error}} . \tag{3.15}$$

Here, the relative error arising from thresholding details is (see Section 2.1.1):

$$\mathcal{E}_{\text{wavelet}}(\epsilon) := \frac{\|\boldsymbol{q}_i(\boldsymbol{x}) - \boldsymbol{q}_i^\epsilon(\boldsymbol{x})\|_2}{\|\boldsymbol{q}_i(\boldsymbol{x})\|_2} \leq \epsilon \tag{3.16}$$

and the relative error due to the truncation after the r-th POD mode in Eq. (3.12) is:

$$\mathcal{E}_{\text{POD}}(\epsilon,r) := \frac{\sum_{i=1}^{N_t} \|\boldsymbol{q}_i^\epsilon(\boldsymbol{x}) - \tilde{\boldsymbol{q}}_i^\epsilon(\boldsymbol{x})\|_2^2}{\sum_{i=1}^{N_t} \|\boldsymbol{q}_i^\epsilon(\boldsymbol{x})\|_2^2} = \frac{\sum_{k=r+1}^{N_t} \lambda_k^\epsilon}{\sum_{k=1}^{N_t} \lambda_k^\epsilon} \tag{3.17}$$

Using Eqs. (3.15) to (3.17) one obtains for the total relative error of the wPOD (see Appendix A.2):

$$\mathcal{E}_{\text{wPOD}}(\epsilon,r) \leq \mathcal{E}_{\text{POD}}(0,r) + \mathcal{M}_r\epsilon + \epsilon^2 \approx \mathcal{E}_{\text{POD}}(0,r) + \epsilon^2 , \tag{3.18}$$

$$\text{where} \quad \mathcal{M}_r = \frac{\sum_{k=r+1}^{N_t} l_k}{\sum_{k=1}^{N_t} \lambda_k} , \tag{3.19}$$

when assuming perturbed eigenvalues $\lambda_k^\epsilon = \lambda_k + l_k\epsilon$, with perturbation $l_k \in \mathbb{R}$. An error bound for the perturbation of the eigenvalues is given for a simplified correlation matrix in [22]. $\mathcal{M}_r$ is often small, in which case it can be neglected. This is further discussed in the numerical example section. Note that in the limit $\epsilon \to 0$ the wPOD error yields exactly the POD error. In the limit $r \to \infty$ the POD error vanishes and we are left with the wavelet compression error. These limits are visualized for our numerical studies in Figs. 3.6, 3.11 and 3.13.

In most applications, the wavelet threshold ϵ will be chosen according to the available memory of the hardware. If memory limitations are not an issue, it is advantageous to balance the wavelet and POD truncation error for better efficiency. Assuming that the approximation in Eq. (3.18) holds $\mathcal{E}_{\mathrm{POD}}(\epsilon, r) \approx \mathcal{E}_{\mathrm{POD}}(0, r)$ (i.e. $\mathcal{M}_r \leq \epsilon$), we can treat the two errors $\mathcal{E}_{\mathrm{wavelet}}, \mathcal{E}_{\mathrm{POD}}$ independently. For a predefined error $\mathcal{E}^*$, we first fix the compression error choosing $\epsilon^* \leq \sqrt{\mathcal{E}^*/2}$ and adjust the truncation error by $\texttt{err} = \mathcal{E}^* - (\epsilon^*)^2$. The truncation rank r^* is then chosen to compensate for the additional error introduced by the wavelet compression. In the case when $\mathcal{M}_r$ is expected to be larger than ϵ, the errors cannot be balanced without having an estimate of the POD eigenvalues. Eigenvalues λ_k^ϵ, which are smaller than the compression error are not reliable. Therefore we recommend the conservative setting, choosing $\epsilon^* < \mathcal{E}^*$, for a first estimate.

3.3 Numerical Results

In this section, we test the wPOD algorithm, outlined in Section 3.2.2, on 2D and 3D numerical data and assess its efficiency and precision. The algorithm is integrated into the open source software package `WABBIT` [129] and can be called as a post-processing routine.

Remark. *The post-processing routine can be called as follows:*

```
wabbit-post --POD --nmodes=<rank> --error=<err> --memory=<RAM> --adapt=
<eps> --components=<K> --list=<CompList1> ··· --list=<CompListK>
```

The number of modes **`rank`** *or the truncation error* **`err`** *can be specified. For the latter, the rank is automatically chosen from the error criterion given in Section 3.2.2. The algorithm requires specifying the*

number of components `K` *together with* `K` *lists of files that store the snapshots of each component in an HDF5 format. Furthermore, the memory and adaptation level should be chosen in accordance with the given resources.*

We provide two types of case studies: A synthetic test case (see Section 3.3.1) in 2D, which is used to benchmark our code. We also compare it to the randomized singular value decomposition (rSVD), outlined in Section 3.1, and case studies for 2D and 3D data obtained by numerical simulation of the incompressible Navier-Stokes equations in Section 3.3.2.

3.3.1 Synthetic Test Case

For the synthetic test case we define a combination of dyadic structures, inspired by [98]:

$$q(x, y, t) = \sum_{k=1}^{R} a_k(t)\Psi_k(x, y)\,,$$

of $R = 15^2$ orthogonal modes $\Psi_k : [0, 30]^2 \to \mathbb{R}$ and temporal amplitudes $a_k : [0, 2\pi] \to \mathbb{R}$. The modes are smooth, two-dimensional bumps

$$\Psi_{m+15n+1}(x, y) = b(\sqrt{(x - x_m)^2 + (y - y_n)^2}) \tag{3.20}$$

$$b(x) = \begin{cases} \exp\left(-\frac{1}{1-x^2}\right), & x \in (-1, 1) \\ 0, & \text{otherwise} \end{cases} \tag{3.21}$$

placed at $(x_m, y_n) = (1+2m, 1+2n)$, $n, m = 0, \ldots, 14$ in a checkerboard pattern. Note that the modes are orthogonal because of their non-overlapping support. Furthermore, we choose oscillating amplitudes:

$$a_k(t) = e^{-k/\Delta\lambda} \sin(\pi f_k t) \quad \text{for} \quad k = 1, \ldots, 15^2 \tag{3.22}$$

with randomly shuffled frequencies $f_k \in \{1, \ldots, 15^2\}$ and moderate decrease in magnitude: $\Delta\lambda = 3$. We choose $N_t = 2^7$ equally spaced snapshots on a $N_x \times N_y = 1024 \times 1024$ initial grid. Before starting the algorithm the initial grid has to be partitioned into blocks. This is done

using a python routine available in the `WABBIT` software package [45]. In our studies, we block-decompose the initial grid in three different configurations to compare the effect of different block sizes. The sizes of the blocks are $B_x = B_y = N_x/2^{J_{\max}} + 1 = 17, 33, 65$ with $J_{\max} = 6, 5, 4$, respectively.

Wavelet Compression

First, we examine the compression of the data with varying block sizes and thresholds ϵ with $10^{-15} \leq \epsilon \leq 10$. Figure 3.3 shows the adaptation of a single snapshot for $\epsilon = 1.0, 2.2 \times 10^{-2}, 1.0 \times 10^{-5}$. For larger ϵ the number of blocks decreases, leading to stronger compression of the data and increasing compression errors.

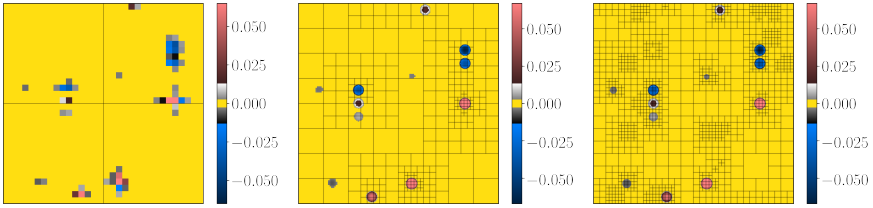


Figure 3.3: Block-based adaptation of $q^\epsilon(x, y, t)$ at $t = 42\Delta t$ for $\epsilon = 1.0, 2.2 \times 10^{-2}, 1.0 \times 10^{-5}$ (from left to right) and with $B_x = B_y = 17$.

This behavior is quantified for varying block size $B_x \times B_y$ and ϵ in Fig. 3.4. Here we plot the relative compression error $\mathcal{E}_{\text{wavelet}}$ and compression factor $C_{\text{f}} \leq 1$, i.e. the fraction between the number of blocks at a given threshold and the total number of blocks available needed for the full grid. As can be seen from Fig. 3.4, a higher maximal refinement level $J_{\max}$, i.e. smaller blocks, corresponds to a smaller overall compression factor, while the compression error $\mathcal{E}_{\text{wavelet}}$ is approximately the same. This observation is expected because smaller blocks enable better resolution of local structures, however increase the data handling effort. For all the compression curves in the numerical examples in Figs. 3.4 and 3.9 we see the classical saddle-shaped error curve: with rapid error decay for small $C_{\text{f}} \lesssim 0.05$ (i.e. large $\epsilon \gtrsim 10^{-2}$) until a plateau is reached with a saddle point from which it begins to decay again. Regardless of the final error of our algorithm, it is recommended

to set ϵ at the onset of the plateau, since after the plateau is reached only a little gain in precision is achieved. The saddle point behavior is an expected structure if an unknown signal with white noise is decomposed into a basis. For example in Fig. 3.11c, we see a similar behavior for the truncation error of the POD. The onset of the plateau may denote the transition from structures that are well presented in a given basis to those that represent noise. Studies of this transition point can be found for SVD and wavelets in the works of Donoho et al. in [57, 39].

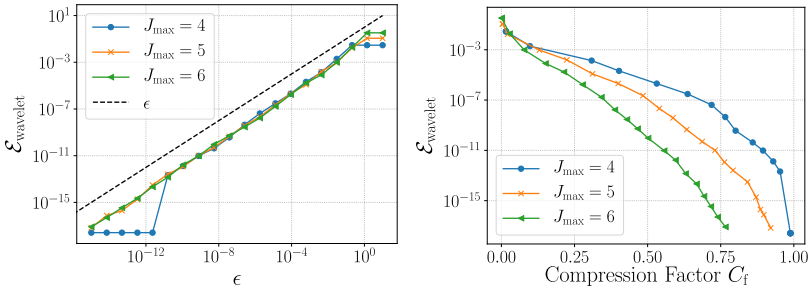


Figure 3.4: Compression error $\mathcal{E}_{\text{wavelet}}$ (left) defined in Eq. (3.16) and compression factor C_{f} (right) of the bump test case using different block sizes $B_x = B_y = N_x/2^{J_{\max}} + 1$, with maximal refinement levels $J_{\max} = 4, 5, 6$. The compression factor is the fraction between the sparse and dense number of grid points/blocks.

Furthermore, we emphasize that the compression error scales linearly in ϵ, independent from the chosen block size, which is an important property for the error control of our algorithm. The sudden drop of the error for $J_{\max} = 4$ is due to the fact, that after $\epsilon \lesssim 10^{-11}$ the blocks are refined to the maximal level.

POD Truncation

Next we study the impact of the wavelet compression on the computed POD modes and the overall approximation error. For $\epsilon = 10^{-5}$, Fig. 3.5 visualizes the first three modes and the corresponding amplitudes obtained with the wPOD algorithm. Note that the modes and amplitudes

of the POD problem Eq. (3.2) are unique up to an orthogonal transformation. Hence the initial input structures Ψ_k, a_k, defined in Eqs. (3.20) and (3.22), are not exactly recovered by the wPOD. However, we see that the magnitude of the amplitudes decreases and its frequency increases with increasing mode number. Moreover one observes that the computational grid is nicely adapted to the structure of the modes.

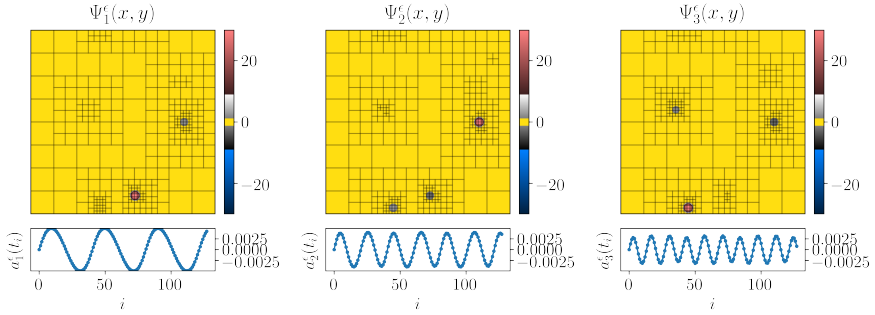


Figure 3.5: First three modes Ψ^ϵ_k and their amplitudes $a^\epsilon_k(t_i)$, $k = 1, 2, 3$, for $\epsilon = 10^{-5}$. Blocks are of size $B_x = B_y = 17$.

Additionally, we estimate the truncation error $\mathcal{E}_{\text{POD}}(\epsilon, r)$ and the total error $\mathcal{E}_{\text{wPOD}}(\epsilon, r)$ in the L^2-norm, as defined in Eq. (3.13). We compare the errors for $10^{-5} \leq \epsilon \leq 1.0, r \leq 30$ in Fig. 3.6. In these plots, the impact of the wavelet adaptation, corresponding to blue lines with $\epsilon > 0$, is visualized and compared to the classical snapshot POD procedure corresponding to $\epsilon = 0$, which is drawn in black. The exponential decay of the eigenvalue spectra, given by the magnitude of the input modes $|a_k| \sim \exp(k/\Delta\lambda)$, is nicely recovered in the $\epsilon = 0$ case up to values $r \lesssim 60$ below machine precision. For increasing ϵ we see that the eigenvalues become increasingly distorted, as expected. However, the eigenvalue distortion does not influence the overall approximation error if epsilon is chosen with care (see Section 3.2.3). In fact, the total approximation error converges approximately with ϵ^2 to the exact values as the difference in Fig. 3.6 (bottom) shows. The presented results are independent of the chosen block size $B_x \times B_y$, although only $B_x = B_y = 17$ is used in the shown figures.

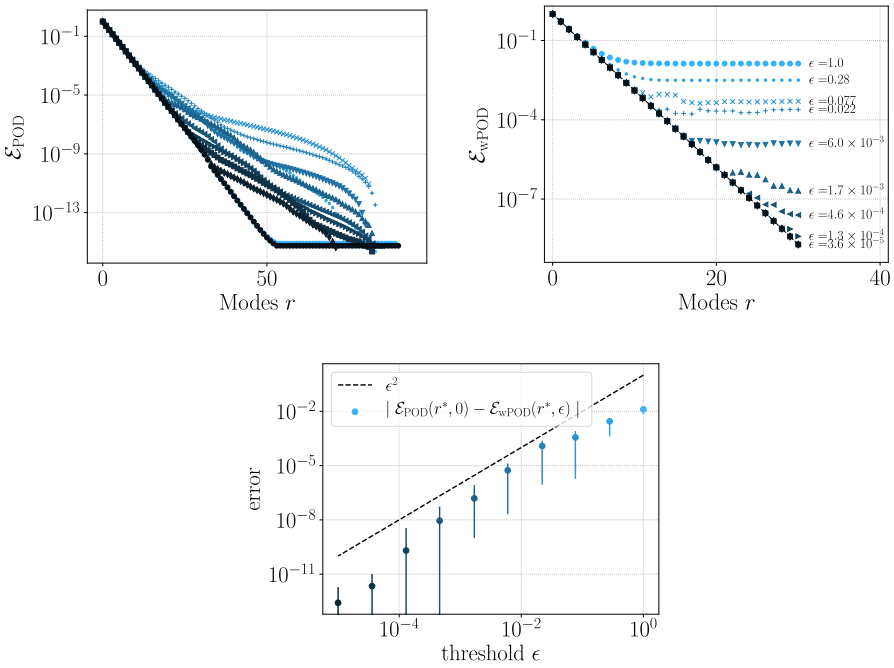


Figure 3.6: Relative errors $\mathcal{E}_{\text{POD}}$ (top, left) (see definition in Eq. (3.17)) and $\mathcal{E}_{\text{wPOD}}$ (top, right) (see definition in Eq. (3.13)) as a function of the truncation rank r and the wavelet threshold ϵ. Difference between classical POD and wPOD compared to the compression error defined in Eq. (3.13) (bottom). The error bars indicate the minimal and maximal value of $\Delta\mathcal{E}(r,\epsilon) = |\mathcal{E}_{\text{POD}}(r,0) - \mathcal{E}_{\text{wPOD}}(r,\epsilon)|$ for all ranks $1 \leq r \leq 30$ and the markers the mean $\overline{\Delta\mathcal{E}}(\epsilon) = 1/30 \sum_{r=1}^{30} \Delta\mathcal{E}(r,\epsilon)$. The colors vary from bright blue at $\epsilon = 1.0$ to black at $\epsilon = 0.0$. Blocks are of size $B_x = B_y = 17$.

Comparison to Randomized SVD

The overall purpose of our method is to be able to fit the snapshot data into the fast memory by tuning ϵ, in order to compute the POD without having to address the slow memory again after the data has been compressed. This enables us to cope with large data sets. Therefore, a comparison to other methods suited for large data like the randomized SVD suggests itself. We follow the algorithm outlined in Section 3.1 taken from [68]. For a fair comparison, we refrain from power iterations, which would need additional passes over the slow memory and we use $n = 5$ extra random samples as suggested in [68]. Hence for a target rank of $r^* = 30$ we take $l = 35$ random samples of our snapshot matrix $\mathbf{Q} \in \mathbb{R}^{M \times N_t}$, $M = 1024^2, N_t = 2^7$ to compute an orthogonal matrix $\hat{\mathbf{Q}} \in \mathbb{R}^{M \times l}$, which approximates the column rank of $\mathbf{Q}$. The computation of $\hat{\mathbf{Q}}$, however, is only feasible if the random samples fit in the fast memory. Therefore, the minimum amount of memory needed by the rSVD is given by $S_{\text{rSVD}} = Ml$ in units of the floating-point arithmetic. In contrast the wPODs memory requirements in units of the floating-point arithmetic: $S_{\text{wPOD}} = N_{\text{blocks}}^{\text{tot}} B_x B_y$ depend on the total number of blocks $N_{\text{blocks}}^{\text{tot}}$ in the snapshot set. Nevertheless S_{wPOD} is tunable with ϵ, but increasing ϵ also increases the compression error. To compare both methods we estimate $\mathcal{E}_{\text{wPOD}}(r, \epsilon)$ as before in a range from $10^{-5} \leq \epsilon \leq 1$ and r up to $r^* = 30$ together with the relative error

$$\mathcal{E}_{(\text{r})\text{SVD}}(r) = \left\| \mathbf{Q} - \tilde{\mathbf{\Psi}} \tilde{\mathbf{\Sigma}} \tilde{\mathbf{V}}^\top \right\|_{\text{F}} / \left\| \mathbf{Q} \right\|_{\text{F}} \tag{3.23}$$

of the truncated (r)SVD in the Frobenius norm. Here $\tilde{\mathbf{\Psi}} \in \mathbb{R}^{M \times r}, \tilde{\mathbf{V}} \in \mathbb{R}^{N_t \times r}$ are matrices, with columns composed of all spatial modes as orthonormal vectors $\boldsymbol{\psi}_k \in \mathbb{R}^M$ and the corresponding temporal coefficients $\boldsymbol{v}_k \in \mathbb{R}^{N_t}$ and $\tilde{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$ contain the POD eigenvalues λ_k as singular values $\sigma_k = \sqrt{\lambda_k}$ on the diagonals. A direct comparison of the total approximation errors $\mathcal{E}_{(\text{r})\text{SVD}}, \mathcal{E}_{\text{wPOD}}$ defined in Eqs. (3.13) and (3.23), respectively, is shown in the left of Fig. 3.7. In both figures the wPOD was set up with $\epsilon = 3.6 \times 10^{-5}$ and $J_{\max} = 5$, such that $S_{\text{wPOD}} \approx S_{\text{rSVD}}$.

One important aspect of this comparison has to be highlighted first. The performance of the rSVD and wPOD strongly depends on the data. The wPOD will always benefit from data with localized smooth

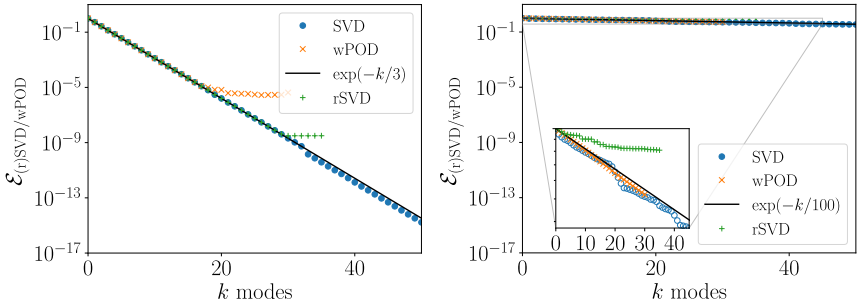


Figure 3.7: Comparison of the total approximation errors $\mathcal{E}_{(r)SVD}, \mathcal{E}_{wPOD}$ for $\Delta\lambda = 3$ (left) and $\Delta\lambda = 100$ (right). As a reference we plot the exact values $\exp(-k/3)$ and $\exp(-k/100)$ indicated by a black line. The rSVD is computed with 35 random samples of $\mathbf{Q}$ and the wPOD set up with $\epsilon = 6 \times 10^{-3}$ and $J_{max} = 5$, to ensure approximately the same memory consumption. The inset shows a zoom.

structures, but it will be less efficient than the rSVD for cases that are distorted by random noise. In our studies the oscillating bump structures are very localized, therefore the wPOD has an advantage over the rSVD. In fact, wPOD needs less memory than the rSVD to achieve the same overall error as shown in the studied parameter range, see Fig. 3.8.

A very interesting case, that cannot be efficiently dealt with by the rSVD, is the case of slowly decaying singular values. Here, random sampling can hardly capture the column rank of $\mathbf{Q}$, since all columns are nearly equally important. Note that also the SVD algorithm implemented in python's numpy package (using LAPACK, the implicit zero-shift QR algorithm after reducing $\mathbf{Q}$ to bi-diagonal form) exhibits numerical instabilities due to round-off errors. These errors occur, when the distance between the singular values gets close to their absolute value, which is especially visible in the case of slowly decaying singular values (see Fig. 3.7). This is not the case for the wavelet POD, as shown in Fig. 3.7. However, small deviations from the expected error decay $\sim \exp(-k/100)$ are visible. For this case we have chosen all parameters as before, except that the magnitude of the modes in Eq. (3.22)

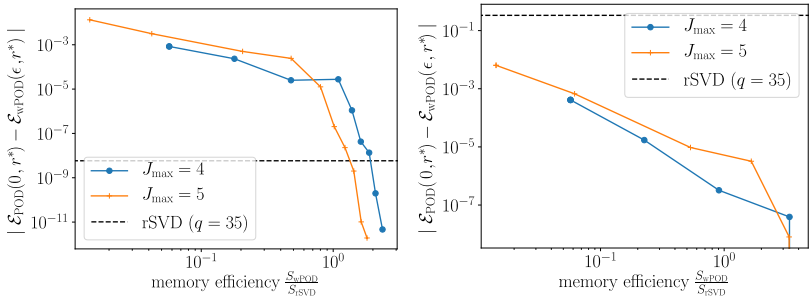


Figure 3.8: Memory consumption vs. deviation from the exact POD values. Comparison between rSVD and wPOD. On the left we show the test case with $\Delta\lambda = 3$ and on the right $\Delta\lambda = 100$. The horizontal dashed lines mark the difference between the rSVD and the SVD using $l = 35$ random samples.

decays much slower with $\Delta\lambda = 100$. This result demonstrates that our method can be used in the combination with large model order reduction problems, which suffer under slowly decaying eigenvalues or singular values. We would like to highlight that, especially in large transport dominated systems, as they often occur in fluid dynamics, MOR is negatively impacted by slowly decaying singular values as reported in [108]. Therefore, our method is very useful in the treatment of such problems. We will come back to this in Section 3.3.2, where we compute a POD of a bumblebee in forward flight.

3.3.2 Application to 2D and 3D Numerical Flow Data

In this section, we compute a sparse POD basis of 2D and 3D flow data, computed with `WABBIT`. In the first study, we use data from a numerical simulation with an equidistant grid of a 2D flow past a cylinder. In the second application, we apply the algorithm to highly resolved 3D data of a flapping flight simulation of a bumblebee [46].

2D Case - Von Kármán Vortex Street

In this first example, our dataset $\mathcal{Q}$ results from a 2D simulation of an incompressible flow past a cylinder using the artificial compressibil-

ity method computed with `WABBIT`. Details about the software can be found in Section 2.1. For this study the block structured grid of `WABBIT` is fixed at the resolution $J_{\max} = 6$, with $(B_x, B_y) = (65, 17)$ and domain size $\mathbb{D} := [0, 64] \times [0, 16]$. This is equivalent to an equidistant grid of 4096×1024 grid points. Each snapshot has three components $\boldsymbol{q} = (\boldsymbol{u}, p)$, where $\boldsymbol{u} = (u_1, u_2)$ denotes the velocity and p the pressure. For this case study, we chose the vortex street for a Reynolds number $\mathrm{Re} = 200$, because it is known to have fast decaying POD truncation errors. The Reynolds number $\mathrm{Re} = 2v_\infty R/\nu$ is based on the cylinder radius $R = 1$, the freestream velocity $u_\infty = 1$ and the kinematic viscosity $\nu = 0.01$. The simulation was run until a stable Kármán Vortex shedding was achieved. From this point onwards, we sample the solution in the time interval $500 \leq t \leq 612$ with $\Delta t = 0.5$, resulting in $N_t = 225$ snapshots. The relevant parameters for our algorithm are summarized in Table 3.1. For a concise overview we only show the scalar-valued vorticity $\omega = \partial_x u_2 - \partial_y u_1$ computed from the velocity components of the state vector in Figs. 2.5 and 3.10 and the curl of the two velocity components of the POD-basis in Fig. A.4.

Table 3.1: Parameters of Von Kármán Vortex Street test case.

Parameter	Value
Number Snapshots N_t	225
Resolution $N_x \times N_y$	4096×1024
Domain size $L_x \times L_y$	64 ×16
Reynolds Number Re	200

Wavelet Compression

As in the synthetic test case in Section 3.3.1, we first study the compression of the flow data. To this end, we sample $\mathcal{E}_{\mathrm{wavelet}}(\epsilon)$ of one representative snapshot $\boldsymbol{q}$ at $t = 550$ with different $\epsilon \in \{10^{-8}, 10^{-7}, \ldots, 1\}$. In Fig. 3.9 we plot the relative compression error $\mathcal{E}_{\mathrm{wavelet}}$ against the wavelet threshold ϵ (left) and against the compression factor C_{f} (right). In contrast to the scalar field in the synthetic example, the thresholded quantity is vector-valued. Therefore, we normalize each state vector component before thresholding. All norms in the plots are vec-

tor norms. Note, that for the threshold $\epsilon = 0.1$ the grid is on the coarsest resolution ($j = 1$) with 4 blocks only (corresponding blocks are shown in Fig. 2.5). This explains why the compression errors beyond $\epsilon = 0.1$ do not differ. Similarly $\epsilon = 1 \times 10^{-8}$ corresponds to the finest resolution at grid level $j = J_{\max}$ and the error in Fig. 3.9 drops to zero. However, as mentioned earlier, for maximal performance the wavelet threshold ϵ should be located at the onset of the plateau, which is reached at $\sim 1 \times 10^{-2}$ (i.e. $C_{\mathrm{f}} \sim 0.035$). Comparing this observation with the block distribution in Fig. 2.5, one observes that smaller values of ϵ produce denser grids, with only little gain in precision. It is remarkable that according to Fig. 3.9 less than 3.5% of the actual data is needed to represent the full data with an L^2-error less than 0.5‰. At this compression level, we have compressed the full data with 225 snapshots from $\sim$ 24GB to $\sim$ 0.84GB, which makes it easily manageable for most laptops.

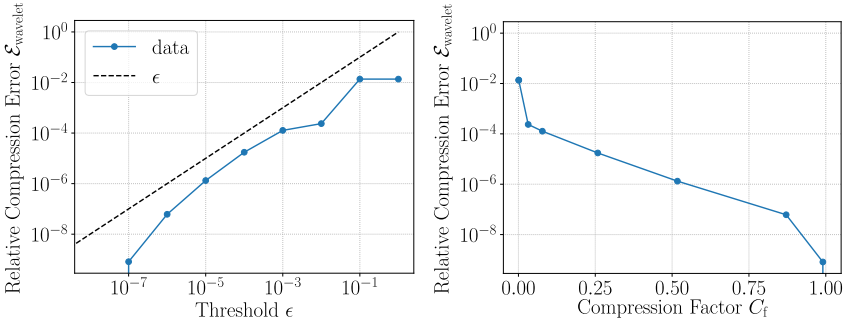


Figure 3.9: Compression error and compression factor C_{f} of the vortex street. Left: The compression error in the L^2-norm is bounded by ϵ, drawn with a dashed line (- - -). Right: Relative error in the L^2-norm vs. compression factor. For direct comparison the vertical axis limits of both figures are identical.

POD Truncation

We will now study the error behavior of the wPOD numerically for fixed threshold ϵ. To this end, we compute $\mathcal{E}_{\mathrm{POD}}(\epsilon, r), \mathcal{E}_{\mathrm{wPOD}}(\epsilon, r)$ for $\epsilon \in \{10^{-8}, 10^{-7}, \ldots, 1\}$. Three calculated modes and their temporal coefficients are visualized with the corresponding block structure in

Appendix A.3, Fig. A.4. Furthermore, we compare the reconstruction $\tilde{\boldsymbol{q}}^\epsilon$ in Fig. 3.10 for the dense case with $\epsilon = 0$ (right column) and one adaptive case $\epsilon = 10^{-2}$ (left column) using $r = 2, 6, 10$ modes. The comparison shows that with an increasing number of modes the typical vortex structure is recovered. No qualitative differences, except local changes in resolution, can be seen, when comparing the adaptive and non-adaptive results. For quantitative analysis, we have plotted the

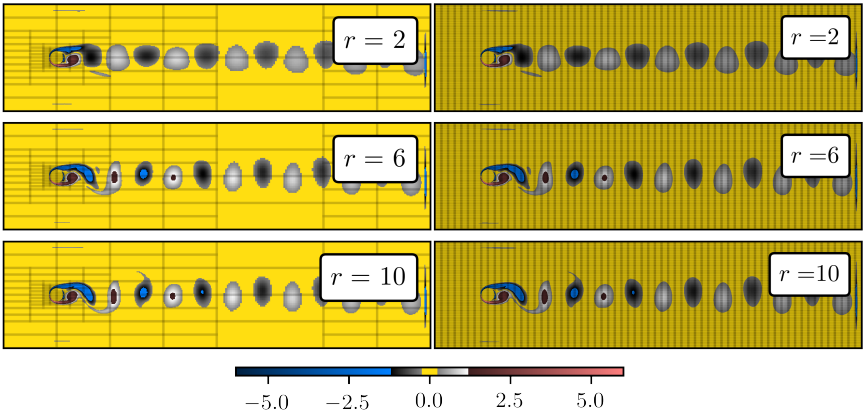
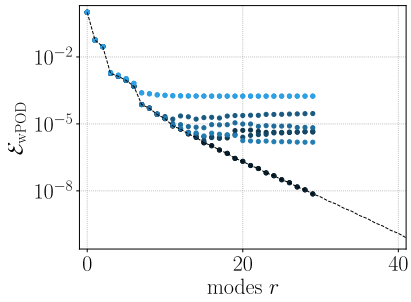


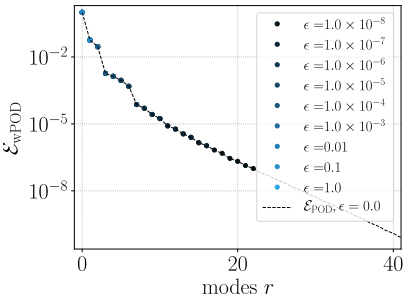
Figure 3.10: Direct comparison between sparse reconstruction $\tilde{\boldsymbol{q}}^\epsilon$ with $\epsilon = 10^{-2}$ (left) and dense reconstruction with $\epsilon = 0.0$ (right) using $r = 2, 6, 10$ modes (from top to bottom). In both columns we display the vorticity $\tilde{\omega}^\epsilon = \partial_x \tilde{u}_2^\epsilon - \partial_y \tilde{u}_1^\epsilon$ of the reconstructed velocity components $(\tilde{u}_1^\epsilon, \tilde{u}_2^\epsilon)$.

total L^2-error and the truncation error in Fig. 3.11. In both plots, the impact of the wavelet adaption (corresponding to $\epsilon > 0$, blue lines) is visualized and compared to the results of the classical POD procedure (corresponding to $\epsilon = 0$, black line). The numerical data show the behavior stated in Section 3.2.3: With the wavelet adaption of the snapshots, errors are introduced, which lead to the distortion of the POD eigenvalue problem and compression errors in the POD basis.

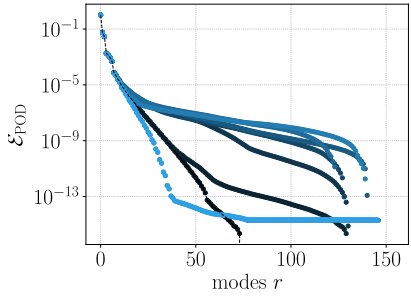
The perturbation l_k of the eigenvalues is especially visible in $\mathcal{E}_{\text{POD}}$ (Fig. 3.11c) for the low energy modes ($r > 30$), corresponding to small eigenvalues. In this regime the distorted eigenvalues $\lambda_r^\epsilon = \lambda_r + l_r \epsilon$



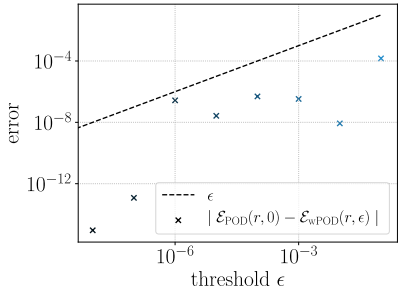
(a) Relative error with error control



(b) Relative error without error control



(c) Relative truncation error



(d) Influence of the wavelet threshold.

Figure 3.11: Relative errors $\mathcal{E}_{\text{wPOD}}$ for the vortex street without error control (a) and with conservative error control (b). (c): The relative errors of the POD ($\mathcal{E}_{\text{POD}}$) and the absolute difference between the POD results and the wPOD with conservative error control (d): $\Delta\mathcal{E}(r,\epsilon) = |\mathcal{E}_{\text{POD}}(r,0) - \mathcal{E}_{\text{wPOD}}(r,\epsilon)|$. Note, that the difference remains below ϵ in the conservative error setting. In all plots, we vary the colors from a bright blue at $\epsilon = 1$ to black at $\epsilon = 0$.

fluctuate around the exact value λ_r (black markers). For the smallest eigenvalues, the relative fluctuation can lead to a total failure of the algorithm, with even negative eigenvalues. This regime, however, can be ignored, since the total error $\mathcal{E}_{\mathrm{wPOD}}$, will be dominated by the compression effects for large r.

Figure 3.11a shows that the error behavior of $\mathcal{E}_{\mathrm{wPOD}}$ is dominated by the truncation error $\mathcal{E}_{\mathrm{POD}}$ for a small number of modes r. With increasing r the truncation error decreases, while the error introduced by the compression remains constant. This leads to a saturation of the total error, as soon as the truncation error is smaller than the error due to compression. This saturation effect is not a special case of the chosen wavelet compression scheme, as it appears in finite element schemes as well, see for instance [134, 61]. However, the wavelet basis has a major advantage over finite element schemes in this setting, since the grids are hierarchically structured, which is easier to handle and computationally efficient. In fact, no additional computations for *"(i) collision detection, (ii) mesh intersection (detect intersection interface) and (iii) integration of complex polyhedra"* [61, p.9] or special vertex bisection triangulation, as discussed in [134], are needed.

If we choose the conservative error setting $\epsilon^* < \mathcal{E}^*$, recommended in Section 3.2.3, with for example $\epsilon^* = 0.1\mathcal{E}^*$ and truncate all modes for which $\mathcal{E}_{\mathrm{POD}}(\epsilon^*, r) \leq \mathcal{E}^*$ we obtain the results shown in Fig. 3.11b. This is essentially the same information as shown in Fig. 3.11a, but all points are excluded which do not fulfill the conservative error criterion. With the help of the conservative error setting, we are able to control the errors introduced by the perturbation of the eigenvalues. Therefore, the difference $|\mathcal{E}_{\mathrm{wPOD}}(\epsilon, r) - \mathcal{E}_{\mathrm{POD}}(0, r)|$, shown in Fig. 3.11d, stays below ϵ.

3D Case - Insect Flight

The data come from a three-dimensional, highly resolved block-based adaptive simulation of a bumblebee in forward flight using `WABBIT` [129]. A summary of relevant parameters is given in Table 3.2. Additional details of the adaptive flight simulation can be found in [46].

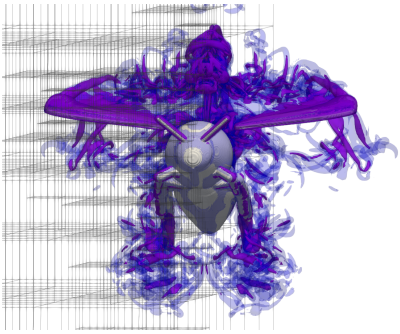
One representative snapshot is shown in Fig. 3.12a together with the

Table 3.2: Parameters of the bumblebee test case. Here ϵ^∞ is the wavelet threshold with CDF4,4 wavelets being normalized in L^∞.

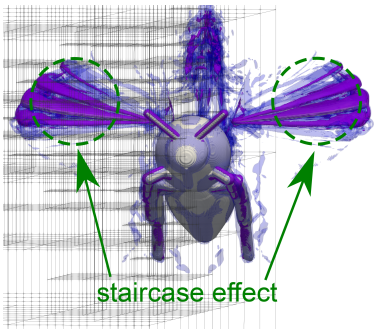
Parameter	Value
Number snapshots N_t	41
Maximal refinement $J_{\max}$	7
Block size $B_x \times B_y \times B_z$	$23 \times 23 \times 23$
Wavelet threshold ϵ^∞	0.01
Domain $\mathbb{D}$	$[0, 8]^3$
Reynolds Number Re	2000

reconstruction of our algorithm using either 5 (Fig. 3.12b) or 15 modes (Fig. 3.12c). Additionally we plot some selected modes in Fig. 3.14. The wPOD algorithm is applied to the vorticity vector $\boldsymbol{w} = \boldsymbol{\nabla} \times \boldsymbol{u}$, which is computed from the velocity $\boldsymbol{u}$. Two isosurfaces of the magnitude of the vorticity, 50 and 100, are shown in Figs. 3.12 and 3.14.

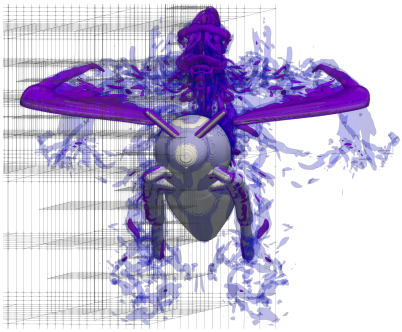
The moving wing geometry causes large gradients of the flow field at the interfaces of the object. These large gradients move with the flapping wing and cause major problems to the POD, like staircase effects (see Fig. 3.12b) of the reconstructed field with slowly decaying energy error (see Fig. 3.13). This drawback of the POD is known for transport dominated fields with large gradients and is theoretically studied in [108, 63] with help of the Kolmogorov n-width. A method to account for parametric moving discontinuities is discussed in Section 4.3. However, wavelet adaptation reduces the amount of computational resources needed in favor of additional accuracy, like the increased number of modes. In fact, for the data presented here ($J_{\max} = 7, B_x = B_y = B_z = 23, \epsilon^\infty = 0.01, N_{\text{blocks}} \leq 8000$) the factor in memory savings in comparison to the dense grid ($N_{\text{blocks}} = 2^{3J_{\max}} = 2097152$) is larger than 260. This factor can be further increased when increasing ϵ. A full POD would be prohibitive because of its tremendous memory demand of approximately 31 TB ($N_{\text{b}} = 2^{3J_{\max}} N_t$ Blocks with 0.4 MB each). It should be further noted that, all previous results in the literature including [134, 62, 61, 47, 22, 79] have been only applied to 1D or 2D cases.



(a) Reference snapshot



(b) Reconstruction with 5 modes



(c) Reconstruction with 15 modes

Figure 3.12: Comparison between a bumblebee snapshot at time $t_i, i = 13$ a) and its POD reconstruction $\tilde{\boldsymbol{q}}_i^\epsilon$ using 5 modes b) and 15 modes c) with $\epsilon = 0.01$. Shown are two isosurfaces of the magnitude of vorticity, i.e., $\|\boldsymbol{\nabla} \times \boldsymbol{u}\|_2 = 50$ and 100. The rigid body of the bumblebee is displayed in gray. The grid structure is indicated behind. Staircase effects at the wings are indicated in Fig. 3.12b. These artifacts appear when POD is applied to sharp structures or discontinuities which move.

The statements about the error made in Section 3.2.3 also hold for the adaptive data: Here the slowly decaying eigenvalues are rather large compared to their perturbation. Hence, $\mathcal{M}_r$ is negligible and the total error behavior is mainly dominated by $\mathcal{E}_{\text{POD}}(r,0)$ and $\mathcal{E}_{\text{wavelet}}(\epsilon)$ (see Fig. 3.13, left). For the case of $\epsilon = 1.0$, it can be nicely seen that the truncation error dominates the total error after it falls below the compression error plateau $\mathcal{E}_{\text{wavelet}}(\epsilon)^2$ at $r \geq 20$. In Fig. 3.13 (right) the difference between the total error $\mathcal{E}_{\text{wPOD}}$ and the POD truncation error $\mathcal{E}_{\text{POD}}$ is shown. Note that for adaptive input data of the bumblebee $\mathcal{E}_{\text{POD}}(0,r)$ can be only assessed approximately since wavelet details have been already discarded during the generation of the data. Hence for the adaptive case $\mathcal{E}_{\text{POD}}(0,r)$ means that no additional compression errors were introduced during the wPOD algorithm. The difference scales quadratically with the compression error $\mathcal{E}_{\text{wavelet}}(\epsilon)^2 \sim \epsilon^2$, drawn as dashed line in Fig. 3.13 (right). From these results, we thus

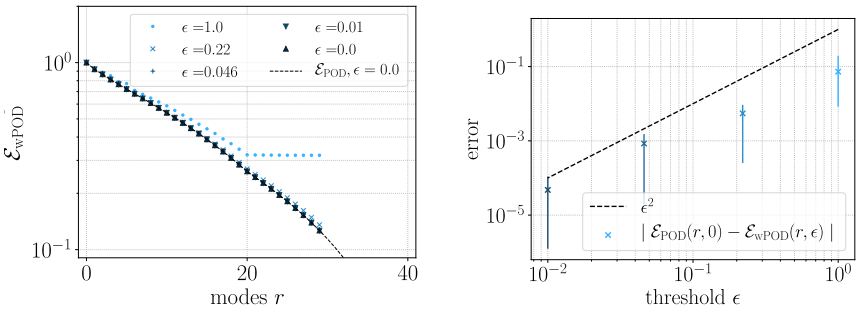


Figure 3.13: Total relative error, $\mathcal{E}_{\text{wPOD}}$, of the wPOD (left) and difference of the wPOD and POD error, $\Delta\mathcal{E}(r,\epsilon) = |\mathcal{E}_{\text{POD}}(r,0) - \mathcal{E}_{\text{wPOD}}(r,\epsilon)|$ (right). The error bars in the right figure indicate the minimal and maximal value of $\Delta\mathcal{E}$ for all ranks $1 \leq r \leq 30$ and the markers the mean $\overline{\Delta\mathcal{E}}(\epsilon) = 1/30 \sum_{r=1}^{30} \Delta\mathcal{E}(r,\epsilon)$.

can conclude that our algorithm can reproduce the POD eigenvalue spectra of the original data, even for larger thresholds $\epsilon > 0.01$ within a predefined precision given by the squared wavelet compression error. As shown in our synthetic test case, this would be a challenge for the randomized SVD, since the eigenvalues decay slowly.

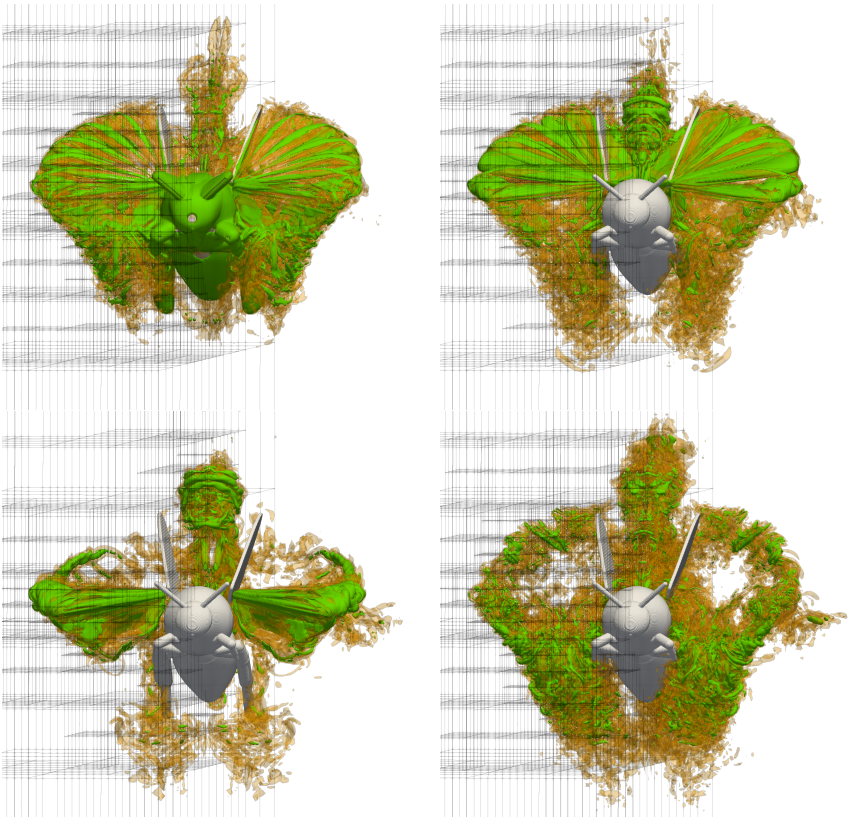


Figure 3.14: Bumblebee modes $\boldsymbol{\psi}_i^\epsilon$, with $i = 1, 9, 18, 27$ and $\epsilon = 1 \times 10^{-4}$ in row-major order. The modes are visualized as isosurfaces of the magnitude $\|\boldsymbol{\psi}_i^\epsilon\|_2 = 10, 20$ with colors green and orange, respectively. For reference, the rigid body of the insect is shown in gray. The adaptive grid is only indicated on the left of each figure.

3.4 Summary

In this chapter, we have discussed a traditional dimension reduction method, the proper orthogonal decomposition (POD), and its generalization for two or three-dimensional block-based adaptive grids. The wavelet-adaptive POD (wPOD) algorithm endows the method of snapshots, proposed by Sirovich [127], with a non-linear approximation to efficiently compute the POD. The method uses wavelet compression on block-based grids to spatially adapt vector fields and thus generate sparse POD modes.

The introduced compression errors of the POD basis are well controlled using non-linear approximation employing wavelet thresholding. While the compression error depends linearly on the chosen threshold ϵ, the total energy error of the wPOD procedure scales with ϵ^2 if the POD eigenvalues decay rapidly. The wavelet compression and POD truncation errors can thus be balanced, assuming that the perturbation of the eigenvalues remains sufficiently small. In comparison to the classical POD, the compression results in overall savings in memory and computational resources, while obtaining approximately the same error. As a consequence, data from highly resolved 3D direct numerical simulations computed on massively parallel platforms can be processed on much smaller systems.

However, these savings in terms of computational effort are accompanied by additional overhead for handling the tree-like data structure. Hence, the wPOD cannot yet compete with equivalent randomized techniques [144, 68] in terms of CPU-time if the memory is not a limiting factor. Nevertheless, we were able to show that our method can handle data efficiently when the singular values decay slowly. In this case, the proposed wPOD does indeed outperform the rSVD. Furthermore, our method expresses the correlation matrix in terms of the underlying wavelet basis, similar to what has been implemented for finite elements in [61]. However, using the scaling relations of wavelets allows us to express the inner products (Eq. (3.10)) effectively and thus avoid many problems associated with finite element schemes, cf. the listed caveats in [61, p.6].

The proposed method is not capable of handling transport dominated

flows efficiently, as pointed out for the bumblebee test case at the end of Section 3.3.2. Here the strong convection of the flow at the moving wings leads to slowly decaying truncation errors, which are especially visible as staircase effects in Fig. 3.12b. This is expected, since the transports do not challenge the calculation of the POD, but they pose difficulties for an approximation generated by the linear decomposition ansatz [108]. To capture the translation of the wings, a large number of modes are required, which results in an inefficient reduced-order model. To incorporate the transport, non-linear mappings are required, which are investigated in the following chapter.

4 Dimension Reduction using Non-Linear Mappings

In this chapter, three different non-linear dimension reduction methods are explored, that can be used to overcome the shortcomings of linear dimensional reduction for transport dominated fluid systems. In the first section (Section 4.1), *neural networks* (NN) are reviewed in the context of model order reduction of transport dominated flows. Due to the general approximation strength of NNs, they are used as a common tool in contexts where linear dimension reduction converges slowly. Second, a closely related approach, called *front transport reduction* (FTR) is introduced in Section 4.2, which has a similar structure as NNs, but yields better insight into the underlying transport. Lastly, a new formulation of the *shifted Proper Orthogonal Decomposition* (sPOD) is presented in Section 4.3, which can be used for systems where the transport is parametrizable along a one-dimensional path. This chapter closely follows [147, 150, 149].

Some numerical examples of the specific application regimes will be given in Sections 4.2.5 and 4.3.6 for the introduced methods.

Finally, Section 4.4 will summarize the methods and provide practical guidance for dimension reduction in the context of transport dominated flows.

4.1 General Purpose Neural Autoencoder Networks

In this section, we briefly explain the basic concept and give a short literature overview in the context of MOR for transport dominated systems with AE. Neural networks and especially autoencoder (AE) networks have become a common tool in dimension reduction [95]. A general introduction to autoencoder networks can be found in [59].

An autoencoder tries to reproduce the input data while squeezing it through an informational bottleneck. It consists of two parts, the

Encoder $g_{\text{enc}}\colon \mathbb{R}^M \to \mathbb{R}^r$, $\underline{\boldsymbol{q}} \mapsto \boldsymbol{a} = g_{\text{enc}}(\underline{\boldsymbol{q}})$, mapping the input data $\underline{\boldsymbol{q}}$ onto points $\boldsymbol{a}$ in a learned lower dimensional latent space and the

Decoder $g_{\text{dec}}\colon \mathbb{R}^r \to \mathbb{R}^M, \boldsymbol{a} \mapsto g_{\text{dec}}(\boldsymbol{a}) = \tilde{\underline{\boldsymbol{q}}}$, mapping the latent representation back to the input space.

The composition of both parts

$$\tilde{\underline{\boldsymbol{q}}} = g_{\text{dec}}(g_{\text{enc}}(\underline{\boldsymbol{q}}))$$

defines the autoencoder. The task of the optimization procedure is to determine $g_{\text{dec}}, g_{\text{enc}}$, such that the reconstruction error over some training data $\mathbf{Q} = [\underline{\boldsymbol{q}}_1, \dots, \underline{\boldsymbol{q}}_{N_t}] \in \mathbb{R}^{M \times N_t}$:

$$\mathcal{L}_{\text{AE}} = \sum_{i=1}^{N_t} \|\underline{\boldsymbol{q}}_i - \tilde{\underline{\boldsymbol{q}}}_i\|_{\text{F}}^2 = \sum_{i=1}^{N_t} \|\underline{\boldsymbol{q}}_i - g_{\text{dec}}(g_{\text{enc}}(\underline{\boldsymbol{q}}_i))\|_{\text{F}}^2$$

is minimized. After the network has been trained, the reduction is achieved as the dimension $r \ll M$ of the latent variables $\boldsymbol{a}_i = g_{\text{enc}}(\underline{\boldsymbol{q}}_i) \in \mathbb{R}^r$ is much smaller than the input dimension M. Therefore, the decoder $\underline{\boldsymbol{q}}_i \approx g_{\text{dec}}(\boldsymbol{a}_i)$ represents a reduced map of the high dimensional data contained in the columns of $\mathbf{Q}$.

In the training procedure, the functions $g_{\text{enc}}, g_{\text{dec}}$ are determined by trainable parameters of the network, called weights and biases. The networks are constructed by a composition of *layers* $g_{\text{enc}} = L_1 \circ L_2 \circ \cdots \circ L_N$. Usually, the layers of the network $L_n\colon \mathbb{R}^i \to \mathbb{R}^o$ are given by an affine linear mapping $\boldsymbol{x} \mapsto h_n(\mathbf{W}_n \boldsymbol{x} + \boldsymbol{b}_n)$, with weights $\mathbf{W}_n \in \mathbb{R}^{o,i}$ and biases $\boldsymbol{b}_n \in \mathbb{R}^o$ together with a predefined non-linear function h_n.

The choice of the input and output dimension $i, o \in \mathbb{N}$ in each layer, the activation function and the number of layers is called *architecture of the network*. The specific architecture for the general AE network that is used in this thesis can be found in Appendix B.1. All results with this architecture are abbreviated with NN in the following.

Autoencoder networks of symmetrical decoder and encoder size have been used successfully in the context of dimension reduction for TDFS in [89, 81, 53]. Although the implementation details differ, results of these studies show that the accuracy of the reduced representation decays rapidly if the latent space dimension (i.e DOF) r is larger or equal to the intrinsic dimension of the sampled solution manifold. This fact is also observed in our numerical results. Theoretical considerations towards the approximation accuracy of a network in the context of parametric MOR of PDEs can be found in [85].

4.2 Dimension Reduction for Complex Moving Fronts

In the following section, we motivate why special non-linear reduction methods are beneficial when decomposing advection-reaction-diffusion (ARD) systems and we introduce the front transport reduction as an iterative thresholding algorithm in Section 4.2.3 and 1-layer autoencoder networks in Section 4.2.4.

4.2.1 The Need for a Non-Linear Decomposition Approach

To motivate our decomposition approach, we consider ARD systems of the form:

$$\frac{\partial q}{\partial t} = \boldsymbol{u} \cdot \boldsymbol{\nabla} q + \kappa \boldsymbol{\nabla}^2 q + R(q)\,. \tag{4.1}$$

These systems describe how a quantity or reactant $q(\boldsymbol{x}, t)$ spreads in space $\boldsymbol{x} \in \mathbb{D} \subset \mathbb{R}^D$, $D > 0$ over time $t \in [0, T]$. This spread can be caused by the advection with velocity $\boldsymbol{u} \in \mathbb{R}^D$ or an interplay between diffusion Δq and reaction processes $R(q)$. For the sake of simplicity we focus on the reaction-diffusion described by a non-linear Kolmogorov–Petrovsky–Piskunov (KPP) reaction term $R(q) = \gamma q^\alpha (q - 1)$ with exponent $\alpha > 0$ and reaction rate $\gamma > 0$. These systems exhibit traveling

or pulsating fronts [66, 5, 6, 49], which are assumed to be locally one-dimensional near the front.

Remark. *Formally, this approach makes use of the assumption that the spatial variable* $\boldsymbol{x} = (x_1, x_2 \ldots, x_d)$ *of the reactant* q *can be transformed to* $\boldsymbol{x}' = (\phi', x_2', \ldots, x_d')$*, where* $x_2', \cdots, x_d'$ *are on a hyperplane tangential to the front of the traveling wave. On this hyperplane, all gradients in the equation vanish relative to the terms that are normal to the traveling wave. Therefore, the flow can be described by a one-dimensional equation in the variable* ϕ' *and we simply can rewrite* $q(\boldsymbol{x}', t) = q(\phi', t)$ *(see [142, p.5] for details).*

Therefore, the solution of Eq. (4.1) can be transformed into a co-moving frame

$$q(\boldsymbol{x}, t) = f(\phi(\boldsymbol{x}, t)) \,, \tag{4.2}$$

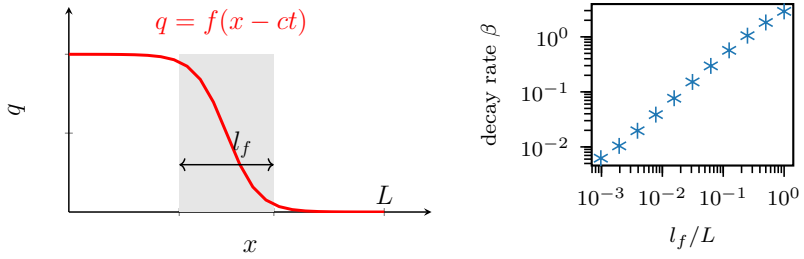
where the *front profile* of the traveling wave is described by f and $\phi(\boldsymbol{x}, t) = (\boldsymbol{x} - \boldsymbol{\Delta}(\boldsymbol{x}, t)) \cdot \boldsymbol{e_v}$, the location of the front with respect to the direction $\boldsymbol{e_v} = \boldsymbol{v} / \|\boldsymbol{v}\|$ of the wave speed $\boldsymbol{v}$. For a one-dimensional traveling wave, this is illustrated in Fig. 4.2a. The profile of the wave f can be approximated with help of perturbation theory after transforming Eq. (4.1) into the co-moving frame (see for example [126]) or by fitting the front profile, as done in [150]. However, the wave speed $\boldsymbol{v}$ is the most complex part in typical applications, since it is coupled to an outer transport/velocity field $\boldsymbol{u}$ in Eq. (4.1) and an additional constant propagation speed c^* of the reacting wave which depends on $R(q)$ (i.e., minimal propagation speed $c^* \geq 2\sqrt{\kappa R'(0)}$ for KPP nonlinearities [66, 5, 6]).

The dimensional analysis yields a definition of the propagating front's thickness in terms of the fraction of the diffusion and propagation speed:

$$l_f = \kappa / c^* \leq 0.5 \sqrt{\kappa / R'(0)} \,. \tag{4.3}$$

This characteristic length scale of the system is shown in Fig. 4.1. In a linear projection-based MOR approach, l_f plays an essential role, because its length is directly related to the success of the approximation. As already pointed out by [63, 108] for transport systems with vanishing front width ($l_f \to 0$), every front position is linearly independent of the others and therefore equally important when defining a projection

basis. Therefore, the typical exponential decay $\|q - \tilde{q}_n\| \sim e^{-\beta n}$ of the approximation error is reduced to $\sim n^{-\frac{1}{2}}$, when increasing the dimension of the ROM-basis n. Since the authors [63, 108] give no general results for $l_f > 0$, we quantify the decay of the approximation errors numerically in Fig. 4.1. It can be seen from Fig. 4.1b that the error decay rate per POD mode diminishes if the traveling distance L becomes large relative to the front width l_f, making a linear MOR approach impractical. In order to compensate the transport, many authors use one-to-one mappings [115, 12, 114, 48, 121], which however cannot be used in our case, since reacting fronts may split or merge.



(a) Front width l_f.

(b) Error decay rate per mode.

Figure 4.1: A wavefront $f(x) = \text{sigmoid}(xl_f)$ shown in Fig. 4.1a travels a distance L. The decay rate $\|q - \tilde{q}_n\| \sim e^{-\beta n}$ as a function of the relative front width l_f/L is shown in Fig. 4.1b when approximating q with n POD-modes.

Hence, for ARD systems we follow a more direct approach, which uses the underlying physical structure Eq. (4.2) of these systems. For given snapshot data $\mathbf{Q} \in \mathbb{R}^{M \times N_t}$ with $\mathbf{Q}_{ij} = q(\boldsymbol{x}_i, t_j) \in [0, 1]$ and front function $f : \mathbb{R} \to [0, 1]$, the approach decomposes the data with help of the non-linear mapping

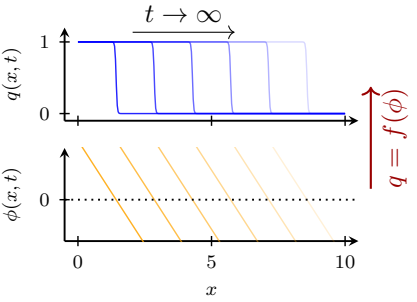
$$q(\boldsymbol{x}, t) \approx \tilde{q}(\boldsymbol{x}, t) = f(\phi_r(\boldsymbol{x}, t)) \quad \text{s.t.} \tag{4.4}$$

$$\phi_r(\boldsymbol{x}, t) = \sum_{k=1}^{r} a_k(t)\psi_k(\boldsymbol{x}) \;, r \ll N_t \tag{4.5}$$

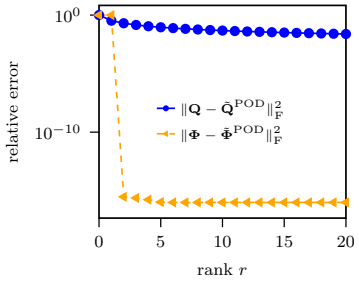
and a low-rank field $\phi_r(\boldsymbol{x}, t)$, that allows to embed the local one-dimensional front movement into a D-dimensional transport. Usually,

the low-rank field is computed with help of a truncated SVD of a snapshot matrix $\mathbf{\Phi} \in \mathbb{R}^{M \times N_t}$, with $\mathbf{\Phi}_{ij} = \phi(\boldsymbol{x}_i, t_j)$. The idea is visualized in Fig. 4.2. Since the transport is only parametrized locally, changes in the topology of the front surface can be captured. The decomposition goal is formulated as an optimization problem.

Problem 4.2.1. *Front Transport Reduction* For a given snapshot matrix $\mathbf{Q} \in \mathbb{R}^{M \times N_t}$ with $\mathbf{Q}_{ij} = q(\boldsymbol{x}_i, t_j) \in [0, 1]$ and non-linear smooth monotone increasing function $f \colon \mathbb{R} \to [0, 1]$, find a rank r matrix $\mathbf{\Phi} \in \mathbb{R}^{M \times N_t}$, such that the error $\left\| \mathbf{Q} - \tilde{\mathbf{Q}} \right\|_{\mathrm{F}}^2$ for $\tilde{\mathbf{Q}}_{ij} = f(\mathbf{\Phi}_{ij})$ is minimized.



(a) Transported quantities



(b) Relative POD approximation errors

Figure 4.2: Illustration of the basic idea of the front transport reduction method. Figure 4.2a: The FTR replaces the sharp traveling front structure q (blue curves), by a level set function ϕ (orange lines) and a non-linear mapping f (indicated by the red arrow). Both quantities share locally the same transport. However, the level set field $\phi(x, t) = x - \Delta(t)$ is of low dimension and can therefore be parametrized with only few POD basis functions (here: $\{x, 1\}$). Figure 4.2b: The generated snapshot data $\mathbf{\Phi}_{ij} = \phi(x_i, t_j)$ can be approximated efficiently with the POD, compared to $\mathbf{Q}_{ij} = q(x_i, t_j)$.

Three possible algorithms that address Problem 4.2.1 are provided in the following sections.

4.2.2 FTR via Signed Distance Functions - Introductory Example

A first heuristic algorithm that addresses Problem 4.2.1 was published in [150]. The introduced algorithm does not solve Problem 4.2.1 exactly in an optimization procedure, but constructs the level set field ϕ from a signed distance function

$$\phi(\boldsymbol{x}, t) = \begin{cases} \operatorname{dist}(\boldsymbol{x}, \mathcal{C}_p(t)) & \text{right of } \mathcal{C}_p(t) \\ -\operatorname{dist}(\boldsymbol{x}, \mathcal{C}_p(t)) & \text{left of } \mathcal{C}_p(t)\,, \end{cases} \tag{4.6}$$

$$\operatorname{dist}(\boldsymbol{x}, \mathcal{C}_p(t)) = \inf_{\boldsymbol{y} \in \mathcal{C}_p(t)} \|\boldsymbol{x} - \boldsymbol{y}\|_2\ . \tag{4.7}$$

In this approach we use `scikit-fmm` [55] to calculate the signed distance function, which is based on the fast marching method (FMM) [125]. The FMM is convenient for our purpose, since it allows to calculate the signed distance function $\phi(\boldsymbol{x}, t_j)$ very efficiently for a large number of snapshots $j = 1, \ldots, N_t$, based on the p-level set contour line $\mathcal{C}_p(t_j) = \{\boldsymbol{x} \in \mathbb{D} \mid q(\boldsymbol{x}, t_j) - p = 0\}, p \in \mathbb{R}$ of our data field q. At this point, a value of ϕ and q is available at every grid point from which the front shape function f is to be determined such that $q = f(\phi)$. This is complicated by the fact that such a relation is approximate and only discrete values are available. From the computed signed distance function we choose all grid points $\hat{\phi}_l = \phi(x_{i_l}, y_{j_l}, t_{i_l})$ with $\left|\hat{\phi}_l\right| \leq \Delta\phi$ and the corresponding samples $\hat{q}_l = q(x_{i_l}, y_{j_l}, t_{i_l})$. The sample vectors $(\hat{\phi}_l, \hat{q}_l)$ are then interpolated on a predefined support set $\phi_i^{\text{sup}} = i\Delta\phi/S$, $i = 1, \ldots, S$ with corresponding interpolation values $f_1, \ldots, f_S$, such that $f_i = f(\phi_i^{\text{sup}})$. In this way, the front profile f can be approximated by cubic splines, assuming that the front profile is sufficiently smooth. Alternatively, one can use more complex representations like fully connected neural networks to approximate the front profile. Note that in contrast to the decoder g_{dec} (Section 4.1), mapping a low input space $\mathbb{R}^r$ onto a high output space $\mathbb{R}^M$, the neural network of the front profile would only approximate a scalar quantity $f\colon \mathbb{R} \to \mathbb{R}$ and is therefore shallow and cheap to evaluate. In contrast to the AE network, dimension reduction is achieved by the low-rank description of ϕ, with help of the POD: $\phi_r(\boldsymbol{x}, t) = \sum_{k=1}^{r} a_k(t)\psi_k(\boldsymbol{x})$, $r \ll N_t$.

Although the front transport reduction signed distance (FTR-SD) approach generates the desired results for the 1D traveling front that is shown in Fig. 4.2, the choice of ϕ as a signed distance function is likely to be sub-optimal, since it is not guaranteed that ϕ can be represented well by only few basis functions. For instance in the moving disk example Eq. (2.38), the signed distance function $\phi(\boldsymbol{x}, t) = \|\boldsymbol{x} - \boldsymbol{\Delta}(t)\|_2 - R$ (shown in Fig. 4.3) together with $f(x) = \frac{1}{2}(1+\tanh(x/\lambda))$ parametrizes the solution of the advected disc. However, if we choose a paraboloid $\varphi(\boldsymbol{x}, t) = \frac{1}{2R}\|\boldsymbol{x} - \boldsymbol{\Delta}(t)\|_2^2 - R^2$ (shown in Fig. 4.3) instead, the transport can be parametrized with only three basis functions as shown in Fig. 4.3. Although the signed distance function might not be optimal for the example of the traveling disc, it is superior to the POD as can be seen from the comparison in Fig. 4.3.

The ability of the FTR-SD method is demonstrated for a more complex example in [150], where it was tested on a 2D propagating flame that interacts with a vortex pair. In this data set, 40 snapshots have been derived from 2D simulations of the reactive Navier-Stokes equations implemented in [128]. In this example, the data were restricted to the normalized mass fraction of hydrogen Y_{H_2}. The simulation was tuned such that a vortex pair moves towards burning H_2 and mixes unburned ($Y_{\mathrm{H}_2} = 1$) with burned gas ($Y_{\mathrm{H}_2} = 0$), so that a small bubble of unburned gas detaches into the burned area. The time evolution is visualized for some selected snapshots on the left side of Fig. 4.5a.

As can be seen from Fig. 4.5a, the Y_{H_2} snapshots contain a very interesting structure, in which the front changes along its contour line and even the topology of the line changes splitting from one curve at $t/\Delta t = 24$ into two curves $t/\Delta t = 29$ and then back to a single curve at $t/\Delta t = 34$. After computing the signed distance function using $C_p(t)$ with $p = 0.14$ for all 40 snapshots, we compute f using the interpolation procedure described above. One representative snapshot and its signed distance function is shown in Fig. 4.4. The approximation quality when we truncate ϕ using r degrees of freedom is compared to the POD in Fig. 4.5. For this specific data, we can decrease the relative errors up to a factor of three compared to the POD, until the overall approximation limit Δf is reached (see Fig. 4.5b). More importantly, by construction the FTR preserves the physical range of the normal-

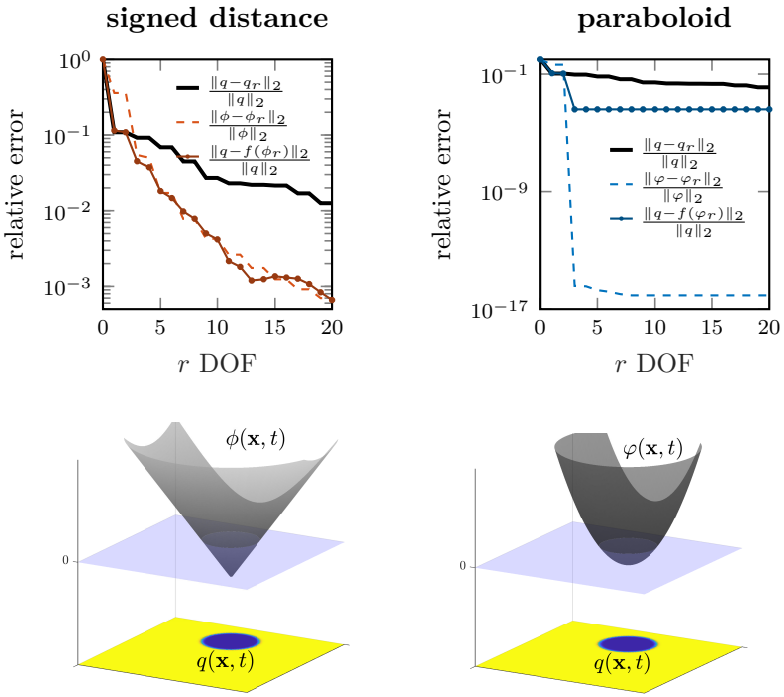


Figure 4.3: Visualization of the FTR decomposition in the case of the moving disk example introduced in Section 2.2.1 using a signed distance function (left) or a paraboloid (right). In the upper row, we compare the relative errors to the results of the POD when reconstructing the solution using r degrees of freedom (DOF). In the bottom row, one snapshot is depicted using 10 DOF for the representation of ϕ and 3 DOF for φ.

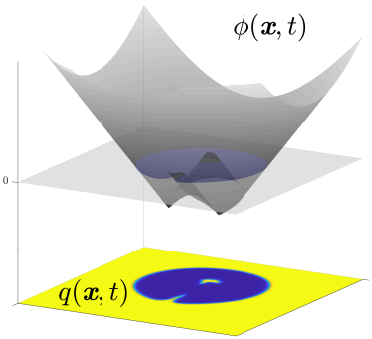


Figure 4.4: Signed distance function $\phi(x, t)$ and resulting approximation $q(\boldsymbol{x}, t)$ of the propagating flame example. The zero-level contour line parametrizes the location of the reacting front.

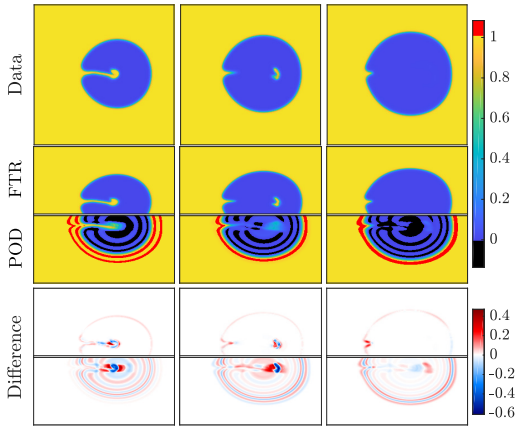
ized reactant $0 \leq Y_{\mathrm{H}_2} \leq 1$ and the front structure of the solution. In contrast, the POD overshoots and undershoots (black and red region in Fig. 4.5a) the physical range and shows staircase behavior.

From the ansatz $q \approx f(\phi_r)$ we can deduce two different errors contributing to the total error:

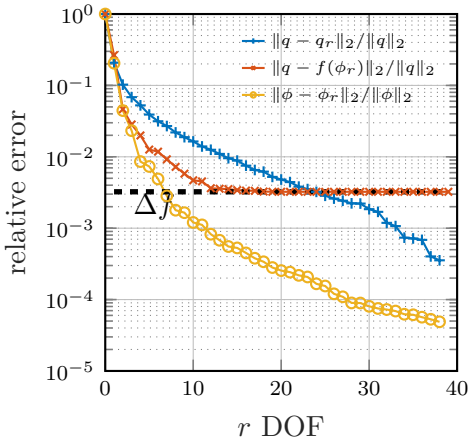
$$\|q - f(\phi_r)\| \leq \underbrace{\|q - f(\phi)\|}_{\Delta f} + \|f'(\phi)\mathcal{R}\| + \mathcal{O}\left(\left\|\mathcal{R}^2\right\|\right) . \tag{4.8}$$

The truncation error of the SVD is $\mathcal{R} = \phi - \phi_r$ and the approximation error of the data without truncation is Δf. Consequently, for vanishing approximation error, the truncation error $\|f'(\phi)\mathcal{R}\| \leq \|f'(\phi)\|\,\sigma_{n+1}$ bounds the total error in the spectral norm. Therefore, the total error scales with the decaying singular values of ϕ. This behavior can be seen in Fig. 4.3 and Fig. 4.5b. While for the signed distance function in the translated disk example $\Delta f = 0$ the relative truncation error aligns with the total error, the error of the paraboloid φ in the right plot is dominated by the approximation error $\Delta f \approx 10^{-3}$, similarly for Fig. 4.5b.

The above examples illustrate the basic concept of the FTR-optimization Problem 4.2.1, where a local 1D description of the front movement is embedded into a higher dimensional front structure. In contrast to areas close to the front, where $f'(\phi) \neq 0$ and the field ϕ has to mimic a



(a) Snapshots $t/\Delta t = 21, 26, 31$



(b) relative errors

Figure 4.5: Comparison of the POD and the FTR for a 2D propagating flame:
a) Direct comparison of the snapshots $t/\Delta t = 21, 26, 31$ with q plotted in the first row, the approximations $f(\phi_r)$, q_r in the second row with $r = 10$ modes and the difference between the data and its approximation in the last row. Note that the images in the lower rows contain only the fractions of the full snapshot that are relevant for our comparison.
b) Relative reconstruction error in the Euclidean norm.

signed distance function, ϕ can be chosen to minimize the truncation error far away from the zero level where $f'(\phi) \approx 0$. Moreover, the optimization procedure relaxes the assumption of constant front width as ϕ is chosen by minimizing Δf close to the zero level.

4.2.3 FTR via Iterative Thresholding

A simple iterative algorithm to determine the auxiliary field $\mathbf{\Phi} \in \mathbb{R}^{M \times N_t}$ of the front transport reduction Problem 4.2.1 is presented in Algorithm 2.

Algorithm 2 FTR as iterative thresholding

Require: $\mathbf{Q} \in \mathbb{R}^{M \times N_t}$ data $\mathbf{Q}_{ij} = q(\boldsymbol{x}_i, t_j)$, τ step size, r rank
1: initialize $\mathbf{\Phi}^k = 0$
2: **while** not converged **do**
3: residual $\mathbf{\Delta f} = f(\mathbf{\Phi}^k) - \mathbf{Q}$
4: $\mathbf{\Phi}^{k+1/2} = \mathbf{\Phi}^k - \tau \mathbf{\Delta f}$
5: decompose and truncate
6: $\mathbf{\Phi}^{k+1} = \mathrm{svd}(\mathbf{\Phi}^{k+1/2}, r)$
7: $k \leftarrow k + 1$
8: **end while**
9: **return** $\mathbf{\Phi}^k$

Our algorithm is constructed by combining a gradient descent step (line 4) to minimize $\left\|\mathbf{Q} - \tilde{\mathbf{Q}}\right\|_{\mathrm{F}}^2$, together with a rank-$r$ projection step of $\mathbf{\Phi}$ (line 6). In the gradient descent step, the FTR residual

$$\mathcal{L}_{\mathrm{FTR}}(\mathbf{\Phi}) = \frac{1}{2}\left\|\mathbf{Q} - \tilde{\mathbf{Q}}\right\|_{\mathrm{F}}^2 \quad \text{with} \quad \tilde{\mathbf{Q}} = f(\mathbf{\Phi}) \tag{4.9}$$

is minimized in the direction of the gradient $D_{\mathbf{\Phi}}\mathcal{L}_{\mathrm{FTR}}(\mathbf{\Phi}) = f'(\mathbf{\Phi}) \odot (f(\mathbf{\Phi}) - \mathbf{Q})$. Here, $f'(\mathbf{\Phi}), f(\mathbf{\Phi})$ are element-wise operations of f, f' on $\mathbf{\Phi}$. Since f is monotonically increasing, it is sufficient to replace $D_{\mathbf{\Phi}}\mathcal{L}_{\mathrm{FTR}}$ by $\mathbf{\Delta f} = f(\mathbf{\Phi}) - \mathbf{Q}$ in line 4. Neglecting $f'(\mathbf{\Phi})$ in the gradient prevents a vanishing gradient for points where $f'(\mathbf{\Phi}) \to 0$, i.e. $|\mathbf{\Phi}_{ij}| \gg 0$. Note, replacing the simple gradient descent step with a quasi-Newton method or a line search does not affect the convergence rate, since it is followed by a projection step (line 6), which is likely to destroy the possible larger step of a more sophisticated method.

The computational costs of Algorithm 2 scale with the complexity of the SVD. Thus, for large systems it can be advantageous to approximate the SVD with a randomized SVD or the wavelet adaptive POD outlined in Chapter 3.

4.2.4 FTR via Neural Autoencoder Networks

Another way to solve the optimization Problem 4.2.1 is with the help of autoencoder networks. The special architecture used for the FTR-autoencoder neural network (FTR-NN) is illustrated in Fig. 4.6. As the FTR-NN should implement the structure motivated in Problem 4.2.1, we choose an architecture, which consists of a single-layer decoder, without bias

$$\underline{\tilde{\boldsymbol{q}}} = g_{\mathrm{dec}}(\boldsymbol{a}) = f(\boldsymbol{\Psi}\boldsymbol{a}), \quad \boldsymbol{\Psi} \in \mathbb{R}^{M\times r},$$

which is activated by the physics-dependent front function f. Here, the images of the linear part $\boldsymbol{\phi}_i = \boldsymbol{\Psi}\boldsymbol{a}_i$, with respect to $\boldsymbol{a}_i = g_{\mathrm{enc}}(\underline{\boldsymbol{q}}_i)$ correspond to the columns of the discrete transport field $\boldsymbol{\Phi} = [\underline{\boldsymbol{\phi}}_1, \ldots, \underline{\boldsymbol{\phi}}_{N_t}]$. Since the image of the linear part is represented by $\boldsymbol{\Psi} \in \mathbb{R}^{M\times r}$, $r \ll M$ the resulting matrix is at most of rank r.

As shown in Fig. 4.6, the encoder network consists of four convolutional layers, each followed by an exponential linear unit (ELU) and a batch normalization layer [76]. We apply a stride of two in all convolutional layers after the first convolutional layer, to downsample the spatial resolution of the input data. The output of the convolutional layers is reshaped to a large vector, that is the input of a fully connected network. The fully connected network is followed by two linear layers, where the first one is activated by an ELU activation and a batch normalization layer. Further details of the architecture and training procedure can be found in Appendix B.1.

For the training of the FTR-NN, an additional smoothness constraint is added to the optimization goal $\mathcal{L}_{\mathrm{FTR}}$, which penalizes the non-smoothness of the columns $\boldsymbol{\psi}_n$ of $\boldsymbol{\Psi} \in \mathbb{R}^{M\times r}$

$$\mathcal{L}_{\mathrm{smooth}} = \lambda_{\mathrm{smooth}} \sum_{n=1}^{r} \frac{\|\mathbf{D}^{+}\boldsymbol{\psi}_n\|_{\mathrm{F}}^2}{\|\boldsymbol{\psi}_n\|_{\mathrm{F}}^2}. \tag{4.10}$$

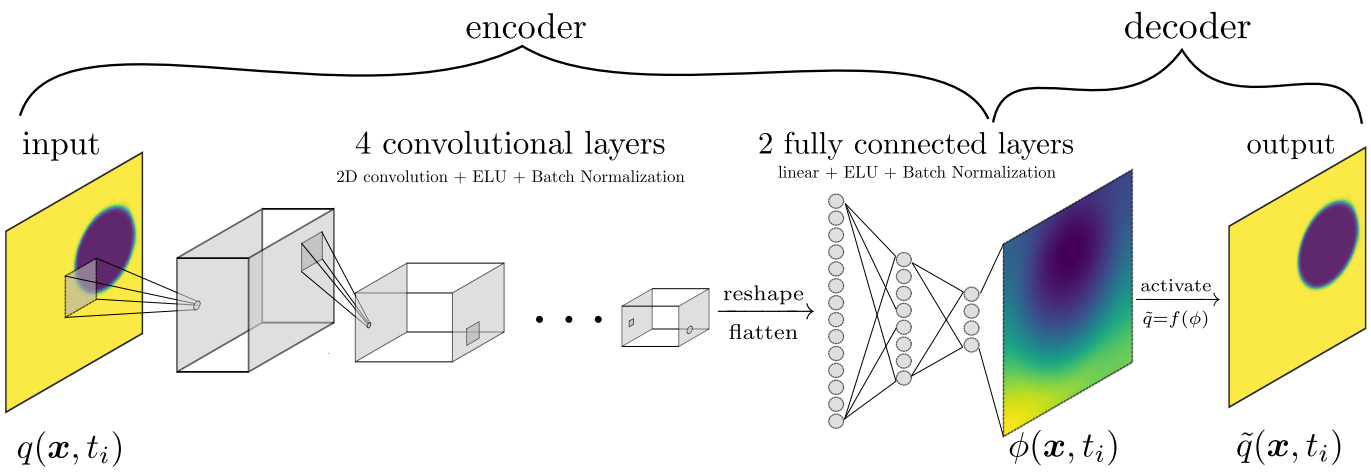


Figure 4.6: Network architecture of the FTR-autoencoder neural network (FTR-NN), that takes one state of the ODE system as input and produces an approximation as output. The encoder decreases the input dimension with help of four convolutional layers (represented by boxes) and two fully connected layers (circles) to only r DOF (here $r = 4$). The size of the box represents the channels in each convolutional layer. After the encoder, the input dimension is increased to the output dimension by the decoder. The decoder is a single linear layer, that is activated by a front function f.

Here, $\mathbf{D}^+ \in \mathbb{R}^{M\times M}$ denotes the coefficient matrix of a forward finite difference, which is implemented as a convolution operation over the columns of $\mathbf{\Psi}$. For the examples in this manuscript, $\lambda_{\text{smooth}} = 10^{-7}$ was found to yield the best results. The additional smoothness constraint allows for faster convergence of the network in the validation phase. The constraint is reasonable, since the columns represent the transport field $\mathbf{\Phi}_{ij} = \phi(\boldsymbol{x}_i, t_j)$, which is assumed to be smooth.

4.2.5 Numerical Results

In the following section, we will compare the different FTR methods to neural autoencoder networks and the POD.

Linear Advection of a Disk

The first synthetic example revisits the advected disk introduced in Section 2.2.1. It illustrates the idea of the FTR in the pure advection case without any topological change. The example parametrizes a disk of radius $R = 0.15L$, which is moving in a circle:autoencoder

$$q(\boldsymbol{x},t) = f(\phi(\boldsymbol{x},t)) \quad \text{and} \quad \phi(\boldsymbol{x},t) = \frac{1}{2R}(\|\boldsymbol{x}-\boldsymbol{x}_0(t)\|_2^2 - R^2) \tag{4.11}$$

$$\text{where } \boldsymbol{x}_0(t) = L\begin{pmatrix}0.5+1/4\cos(2\pi t)\\ 0.5+1/4\sin(2\pi t)\end{pmatrix}. \tag{4.12}$$

The snapshot data $q(\boldsymbol{x},t)$ is generated from a level set field $\phi(\boldsymbol{x},t)$, which is zero at the outer radius of the disk, i.e. location of the front. The front is generated with help of the function $f(x) = (\tanh(x/\lambda) + 1)/2$, $\lambda = 0.1$. One representative snapshot of the data is shown together with its approximation using the POD in Fig. 4.7. As it was already pointed out in Section 4.2.2, for this example $\phi(\boldsymbol{x},t) = \sum_{k=1}^{3}\psi_k(\boldsymbol{x})a_k(t)$ can be parametrized by only three functions and is therefore of low-rank, even if the field $q(\boldsymbol{x},t)$ itself is not. The basis functions ψ_1, ψ_2, ψ_3 are shown in the top row of Fig. 4.8a. They can be interpreted as a quadratic basis function $\psi_1(x,y) = (x-0.5L)^2 + (y-0.5L)^2 + R^2 + L^2/4$ that represents the initial shape of the contour line with constant time amplitude $a_1(t) = a_1 \in \mathbb{R}$, and the linear transport functions $\psi_2(x,y) = x, \psi_3(x,y) = y$ for the shift in x/y-direction with

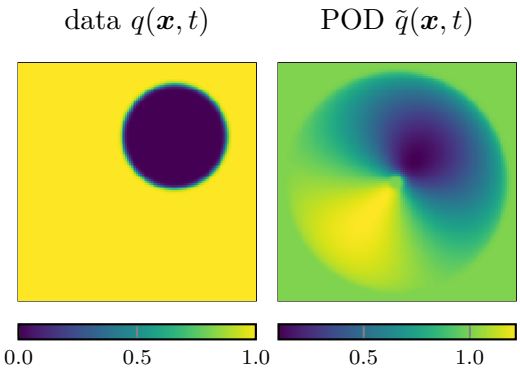
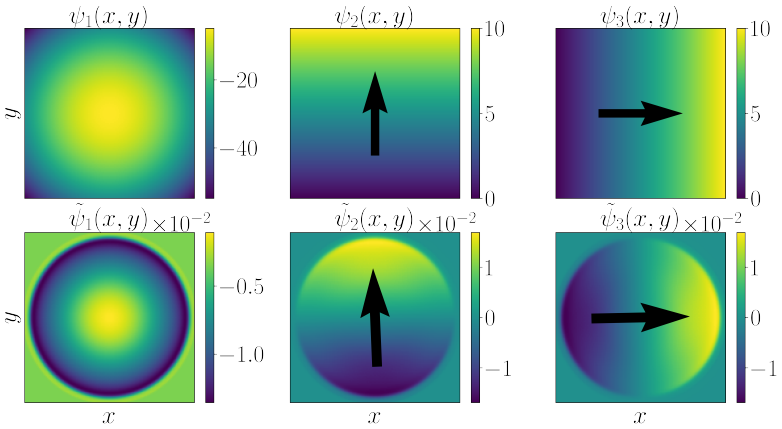


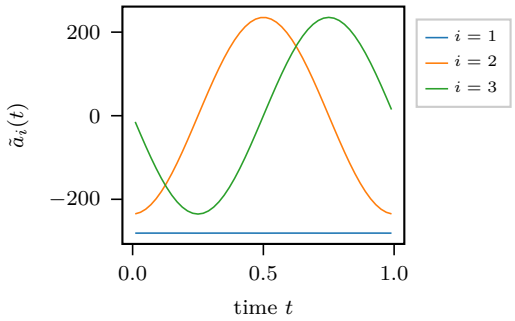
Figure 4.7: Data at time $t = 0.11$ and its approximation with the Proper Orthogonal Decomposition (POD) using $r = 3$ modes.

$a_2(t) \sim \cos(2\pi t), a_3(t) \sim \sin(2\pi t)$. Note that the arrows in Fig. 4.8a indicate $\boldsymbol{\nabla}\psi_2(x, y)$, $\boldsymbol{\nabla}\psi_3(x, y)$ the direction of the shift.

To illustrate that the singular value thresholding Algorithm 2 (FTR) and the neural network approach Section 4.2.4 (FTR-NN) can find a similar basis set, we generate 200 equally spaced snapshots from q in the time interval $0 < t \leq 1$, discretized with 129×129 grid points in the rectangular domain $[0, L]^2$. The data were split into a training and test set, where every second sample is a test sample. After training the neural network on the training samples, it is compared to the results of the POD and the thresholding Algorithm 2 using the test samples. The results are visualized in Figs. 4.8 and 4.9. In Fig. 4.9 we compare the results of both FTR-algorithms (FTR, FTR-NN) and a simple symmetrical autoencoder structure, labeled with NN (for details see Appendix B.1). The relative errors in the Frobenius norm are shown in Fig. 4.9a. The quantitative errors of FTR-NN and the FTR show a significant drop using $r = 3$ degrees of freedom (FTR basis functions/latent space dimension), which is in accordance with the proposed level set field. In contrast, the POD is showing a much slower convergence of the relative error. Comparing the two networks NN and FTR-NN regarding the quantitative errors shows that the additional depth of the NN-decoder compared to the one-layer decoder

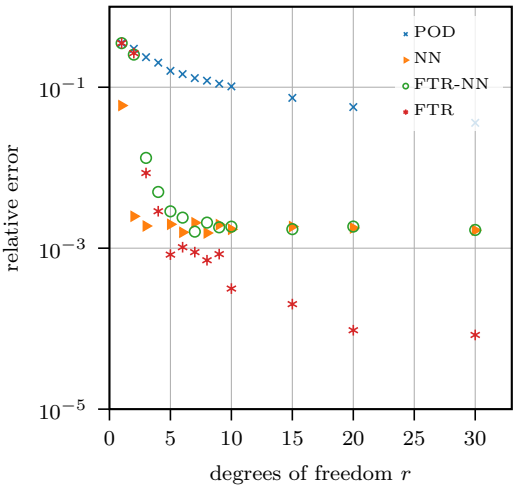


(a) spatial modes

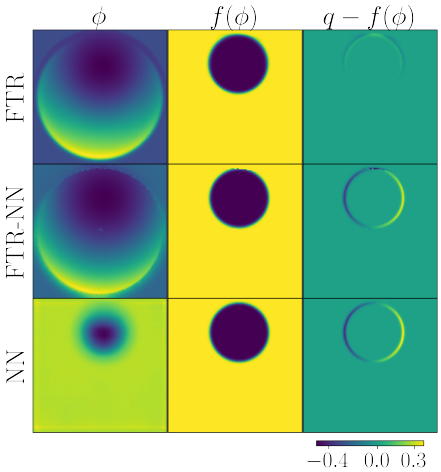


(b) time coefficients

Figure 4.8: Visualization of the FTR transport field $\phi(x,t) \approx \sum_{i=0}^{3} a_i(t)\psi_i(x,y)$ for the disk moving in a circle (see Eqs. (4.11) and (4.12)). Displayed are the expected spatial modes ψ_i (a), their temporal amplitudes a_i (b) and FTR approximation $\tilde{\psi}_i, \tilde{a}_i, i = 1,2,3$. The arrows in Fig. 4.8a indicate the direction of the shift. They are computed from the spatial mean of $\boldsymbol{\nabla}\psi_2(x,y)$, $\boldsymbol{\nabla}\psi_3(x,y)$. The corresponding amplitudes a_2, a_3 parametrize the circular movement in time.



(a) Quantitative error



(b) Qualitative error $r = 3$ degrees of freedom

Figure 4.9: Comparison of POD, FTR, FTR-NN and the symmetrical autoencoder structure labeled with NN. Fig. 4.9a compares the relative errors in the Frobenius norm for different degrees of freedom. Fig. 4.9b visualizes the level set field ϕ together with the approximation of the data $\tilde{q} = f(\phi)$ and the deviation from the exact data $q - \tilde{q}$ for one selected snapshot.

in FTR-NN does not influence the minimal relative error. This leads us to conclude that additional depth is not needed for a better representation. However, note that the NN needs fewer degrees of freedom to converge to its minimal relative error, which is due to the higher expressivity of a deeper network. Furthermore, it is important to note that the FTR-thresholding algorithm outperforms both networks for this special example, when increasing the number of degrees of freedom. For qualitative comparison, Fig. 4.9b shows the approximation of one snapshot before and after activation (first and second column), together with the difference in the last column. Comparing the POD in Fig. 4.7 to the FTR results shows that the typical staircase behavior (which becomes a blurring of the sharp structures for many snapshots as used here) of the POD can be overcome with the FTR ansatz that recaptures the sharp front. We observe that both qualitative and quantitative errors of the FTR-NN and iterative thresholding approach yield similar results. In this study, we use $\lambda_{\text{smooth}} = 10^{-7}$ for regularizing the smoothness of ϕ at the output of the FTR-NN and NN decoder. As visualized in Appendix B.1 Fig. B.1, for larger smoothness parameter $\lambda_{\text{smooth}} > 10^{-7}$ the transport field of the FTR-NN is smoothly continued at areas of no information (no transport), but the additional constraint Eq. (4.10) can cause a larger overall approximation error. However, the level set fields of the iterative thresholding approach and NN are almost identical inside areas where fronts have been transported. This is due to the special choice of the encoder.

Figure 4.8 compares the fields ψ_1, ψ_2, ψ_3 to the first three modes of ϕ. Similar to the proposed functions, the auxiliary field can be split into a mode ($\tilde{\psi}_1$) responsible for the shape of the disk and two modes that parametrize the transport ($\tilde{\psi}_2, \tilde{\psi}_3$). As expected for this special case, $\tilde{a}_1$ is constant and $\tilde{a}_2, \tilde{a}_3 \sim \cos(2\pi t + \theta)$, with $\theta \in \mathbb{R}$ depends on the alignment (indicated as arrows in Fig. 4.8a) of the two shifting functions $\tilde{\psi}_2, \tilde{\psi}_3$. The modes $\tilde{\psi}_2, \tilde{\psi}_3$ only have meaningful values along the trajectories of the front because the algorithm cannot reconstruct ϕ in areas of no transport. This explains that the modes in Fig. 4.8 are zero outside the circle.

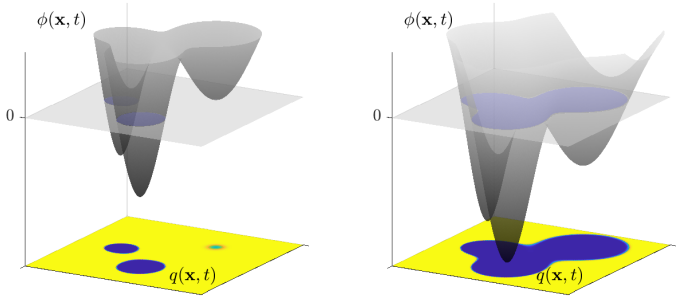
Advection with Topology Change

In this example, we show that our approach is capable of handling transport with topological changes. Therefore, we introduce the synthetic snapshot data $q(\boldsymbol{x}, t) = f(\phi(\boldsymbol{x}, t))$ build from the level set field

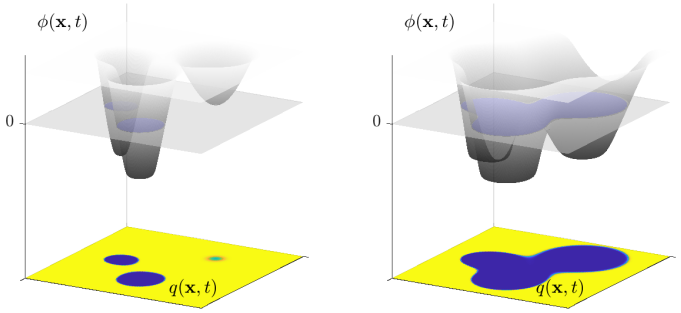
$$\varphi(\boldsymbol{x}, t) = \sum_{k=1}^{3} -A_k e^{-s_k p_k} - t, \quad p_k = \|\boldsymbol{x} - \boldsymbol{x}_k\|_2 \,, \tag{4.13}$$

which we try to approximate. The front is given by $f(x) = (\tanh(x/\lambda)+1)/2$, $\lambda = 0.1$. The level set field is sampled equidistantly using 256×265 grid points in $[0, 10]^2$, with $(A_1, A_2, A_3) = (1, 1.4, 1.2)$, $(s_1, s_2, s_3) = (0.1, 0.3, 0.5)$ and $\boldsymbol{x}_1 = (7.5, 3.5), \boldsymbol{x}_2 = (2.5, 5.0)$, $\boldsymbol{x}_3 = (5.0, 7.6)$. Furthermore, 101 equally spaced snapshots with $0 \leq t \leq 0.5$ are constructed from Eq. (4.13). As above, we split the samples into training and test set, where every second sample is used for testing the autoencoders. After training the networks, they are compared to the reconstruction errors of the POD and FTR using the test samples. The level set fields for $t = 0$ and $t = 0.4$ are visualized as a surface plot in Fig. 4.10, together with the resulting snapshots of q as a color plot. The intersection of ϕ with the zero plane parametrizes the surface of the front. For increasing t, the level set function is shifted vertically and produces an expanding surface of the front, which is merged from three independent contours into one single front contour. The merging of the fronts allows no smooth bijective mapping between the contour lines of the front at time $t = 0$ to $t = 0.4$. In these circumstances, other non-linear reduction methods like the sPOD [115], the low-rank registration based manifolds [101, 100], the registration-based data compression method [133] or transformed snapshot interpolation [140] that rely on one-to-one mappings between different time or parameter instances are not applicable.

As presented in Fig. 4.11, the FTR approximates the dynamics within two dyadic pairs with an error smaller than 0.2%, which is expected from the two-term dyadic structure in Eq. (4.13). The network approximation errors behave similarly to the ones of the moving disk. The FTR-NN gives similar results as the FTR, but with a larger minimal relative error. Due to the additional depth, the NN only needs one degree of freedom to converge towards its minimal error. Topol-



(a) data $t = 0$ **(b)** data $t = 0.4$



(c) FTR $t = 0$, rank $r = 2$ **(d)** FTR $t = 0.4$, rank $r = 2$

Figure 4.10: Graph of the auxiliary field ϕ (Eq. (4.13) in a)-b) and its FTR approximation in c)-d). The resulting snapshots $q = f(\phi)$ are shown as color plot in the xy plane. The intersection with the zero level is visualized.

ogy changes of the zero level set are nicely recovered as illustrated in Figs. 4.10c and 4.10d, since the FTR approach can recover the initial auxiliary field ϕ in the regions of transport.

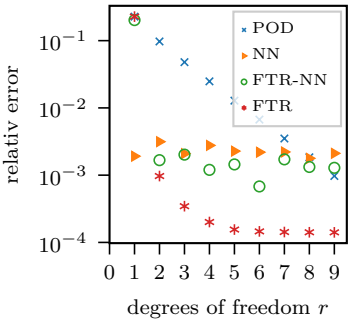


Figure 4.11: Comparison of the relative errors for the advection example with topology change using the Proper Orthogonal Decomposition (POD), the FTR iterative thresholding algorithm (FTR), the FTR autoencoder structure (FTR-NN) and a standard autoencoder (NN).

Advection-reaction-diffusion Systems

To motivate the FTR approach for more complex examples, we apply it to the advection-diffusion-reaction PDE with a Kolmogorov-Petrovsky-Piskunov [83] reaction term introduced in Section 2.2.2. In the following, we refer to the discretized system as the full order model (FOM). All parameters relevant for the reduced order model are listed in Table 4.1.

The results of the three different algorithms are compared with the POD in Fig. 4.12. Similar to Section 4.2.5, the FTR iterative thresholding algorithm outperforms the neural networks and the POD. Furthermore, the time evolution of the FOM is visualized in the top row of Fig. 4.13. In the second and third row, the FTR and POD are compared using $r = 6$ degrees of freedom. The POD approximation shows the typical staircase behavior as oscillations occur before and after the contour line of the front. These oscillations violate the initial range of values $0 \leq q \leq 1$, depicted as red and black areas in Fig. 4.13.

Name	Value
ROM - parameters	
Number of snapshots	$N_t = 200$
Front function	$f(x) = \text{sigmoid}(x)$
Number of FTR iterations	5000
Number of FTR step size	$\tau = 1$
Smoothness parameter	$\lambda_{\text{smooth}} = 10^{-7}$

Table 4.1: Parameters of the 2D ARD simulation of Section 4.2.5.

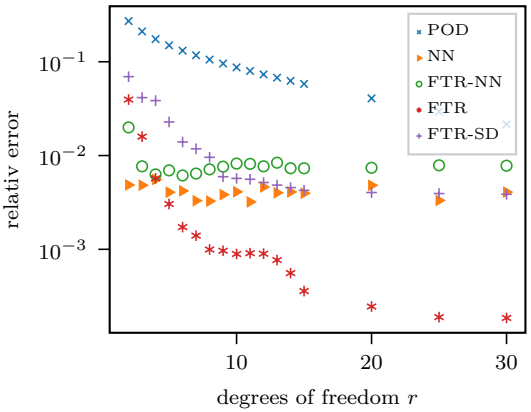


Figure 4.12: Comparison of the relative errors for the 2D ARD system using the Proper Orthogonal Decomposition (POD), the FTR iterative thresholding algorithm (FTR), the FTR signed distance method (FTR-SD), the FTR autoencoder structure (FTR-NN) and a standard autoencoder (NN).

Therefore, preservation of physical structure cannot be expected.

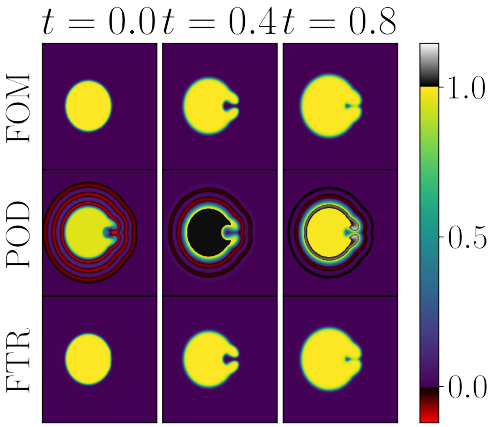


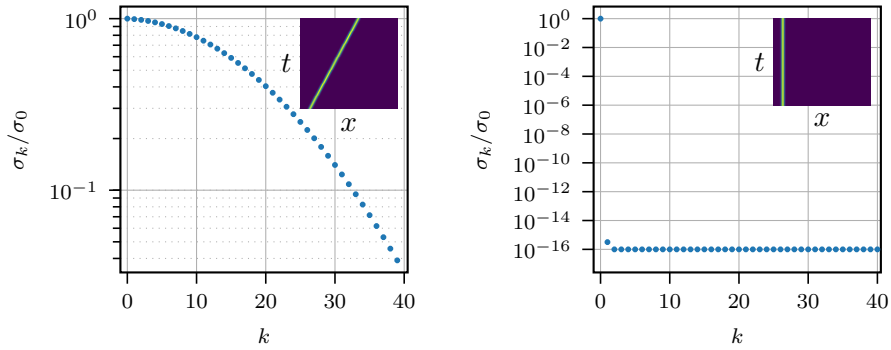
Figure 4.13: Qualitative comparison of the reconstruction errors of the 2D ARD system Eq. (2.40) at three different time instances $t = 0.0, 0.4, 0.8$ (respectively left, middle, right column). The plot shows the FOM data (top row) and its reconstructions using the POD (middle row) and FTR (bottom row). For the FTR and POD, $r = 6$ degrees of freedom are used. The color bar is chosen such that values outside the initial range of values $0 \leq q \leq 1$ are highlighted in black or red.

Here, the FTR approach gives much better results, restricting the approximation to the initial range of values due to the range of the sigmoid function $f(x) \in [0, 1]$.

4.3 Dimension Reduction along Parametrizable Paths

Another common idea for dimension reduction of transport dominated fluid systems (TDFS) is to align the moving localized structure via spatial transformations onto a fixed reference frame at which the moving structure is changing slowly in time and can be decomposed efficiently with the POD (see illustration Fig. 4.14).

In this section, we build on this idea using the *shifted proper orthogonal decomposition* (sPOD) [115], which allows decomposing TDFS with multiple transports. Based on [114], a new formulation is developed that allows iteratively solving the associated optimization problem for large-scale data sets and addresses additional issues like robustness against corrupted data and smoothness of the reduced space. Therefore, Section 4.3.1 introduces the utilized transformations, which are used in the subsequent sections. Sections 4.3.2 and 4.3.5 derive two new sPOD optimization algorithms. Since we here assume that the discontinuities move along parametrizable paths, we restrict the following section to one spatial dimensional ($D = 1$) and highlight generalizations needed for higher spatial dimensions.



(a) x-t-diagram and SVD truncation error of the data

(b) x-t-diagram and SVD truncation error of the shifted data

Figure 4.14: Data of a traveling wave structure and its approximation errors (a) compared to a traveling wave shifted to a reference frame (b).

4.3.1 Shifting into Co-Moving Frames

The shifted POD uses coordinate transformations that shift a data field $q(x,t)$ in a reference frame, in which the moving structures or waves are stationary. The transformations are invertible mappings G, that are illustrated in Fig. 4.15. We call $(x,t) \in \mathbb{D} \times [0,T]$ the *reference system* defined on the simulation domain $\mathbb{D}$ and the transformed system $(\tilde{x},t) = G^{-1}(x,t) \in \tilde{\mathbb{D}} \times [0,T]$ the *co-moving* system defined on the

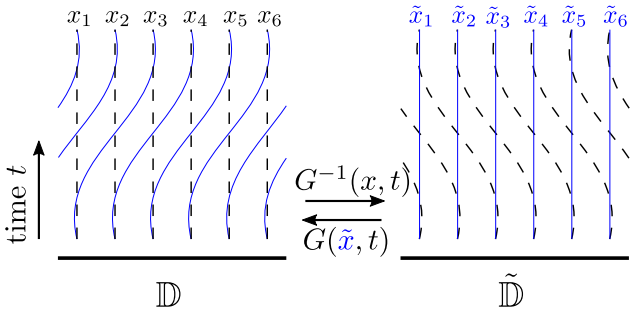


Figure 4.15: Illustration of the time-dependent grid transformations G and G^{-1} with periodic boundaries. The left domain, denoted by $\mathbb{D}$, represents the reference system with dashed coordinate lines. The right domain, denoted by $\tilde{\mathbb{D}}$, represents the co-moving reference system with solid blue lines. If a wave travels along the blue lines in the reference frame, it is stationary in the co-moving frame.

transformed domain $\tilde{\mathbb{D}}$. In the case of periodic boundary conditions the domain remains unchanged, i.e. $\tilde{\mathbb{D}} = \mathbb{D}$.

In this work, we focus on translations $G\colon \tilde{\mathbb{D}} \times [0,T] \to \mathbb{D} \times [0,T]$:

$$G(x,t) = (x + \Delta(t), t) \quad \text{and} \quad G^{-1}(x,t) = (x - \Delta(t), t) \tag{4.14}$$

where $\Delta(t) \in \mathbb{R}$ is a time-dependent shift. However, more general bijective mappings are possible, for example, combinations of rotations and translations as done in [146].

If a field is transformed into a different system, we write $\mathcal{T}^G\colon q(x,t) \mapsto q(G(x,t))$, where $\mathcal{T}^G$ is called *transport operator*. Since our snapshot data $\mathbf{Q} = [q(x_i,t_j)]_{ij}$ is assumed to be the solution of a discretized PDE on a rectangular domain $\mathbb{D}$ with equally spaced grid points, we introduce the discrete version of the transport operator $T^G : \mathbb{R}^{M\times N_t} \to \mathbb{R}^{M\times N_t}$, $\mathbf{Q} \mapsto T^G\mathbf{Q}$, which is given by $T^G\mathbf{Q} = [q(G(x_i,t_j))]_{ij} + \mathbf{E}^G$. As the transformed coordinates $G(x_i,t_j)$ do not necessarily lie on a sampled grid point, we have to interpolate $q(G(x_i,t_j))$ from the sampled points $\mathbf{Q} \in \mathbb{R}^{M\times N_t}$. In this work, we use Lagrange polynomials for the approximation of T^G (see Appendix C.1). The error with respect to the exact transformation $\mathbf{Q}^G = [q(G(x_i,t_j))]_{ij}$ is therefore given by

(see Appendix C.1):

$$\mathbf{E}^G := \mathbf{Q}^G - T^G\mathbf{Q} \tag{4.15}$$

$$\left\|\mathbf{E}^G\right\|_\infty \leq \max_{(\xi,t)\in\mathbb{D}\times[0,T]} \frac{\left|\partial_\xi^{(n+1)} q(\xi,t)\right|}{(n+1)!} \max_{\Delta p\in[0,1]} |w(\Delta p)| \tag{4.16}$$

with

$$\max_{\Delta p\in[0,1]} |w(\Delta p)| = \begin{cases} \frac{1}{4}h^2 & \text{for } n=1 \\ \frac{9}{16}h^4 & \text{for } n=3 \\ \frac{225}{64}h^6 & \text{for } n=5\,. \end{cases} \tag{4.17}$$

The error Eq. (4.15) and its error bound Eq. (4.16) are shown in Fig. 4.16 for a simple traveling wave $q(x,t) = \exp(-((x-x_0)-ct)^2/\sigma^2)$ with $(x,t) \in [0,1]\times[0,0.5]$ and $\sigma = 0.015$, $c = 1$, $x_0 = 0.1$. The data are transformed using the transformation $G(x,t) = x + ct$. The initial data $\mathbf{Q}$ and the reference field $\mathbf{Q}^G$ are shown in Fig. 4.17. Note that the

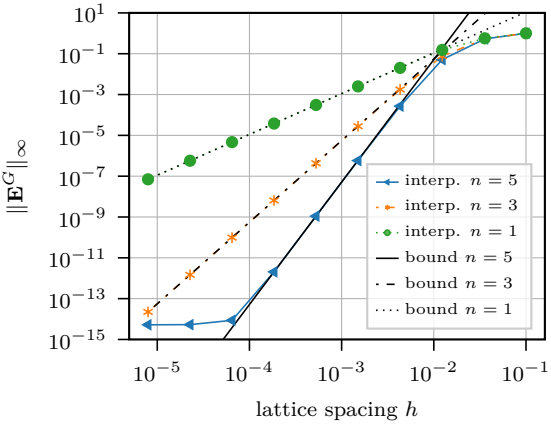
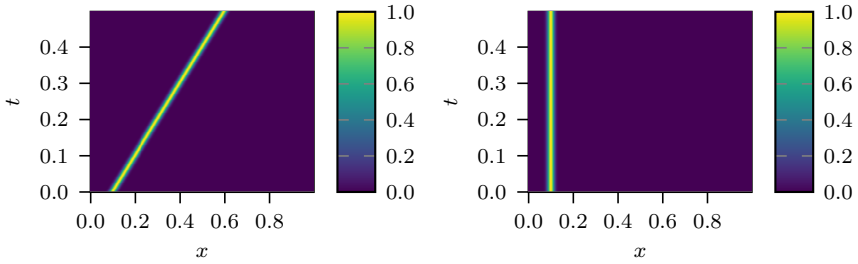


Figure 4.16: Convergence study of the discrete transformation operators T^G using Lagrange polynomials of nth order. The lattice spacing is given by $h = 1/(M-1)$, where M is the spatial resolution.

error bound is reached only in the asymptotic regime. In this regime, the interpolation order of $n = 5$ shows the fastest convergence of $\mathcal{O}(h^6)$ for the lattice spacing $h \to 0$. Furthermore, it should be noted that for

shifts that are multiples of the lattice spacing, $\Delta p = 0$ and therefore the error Eq. (4.16) is zero, since $w(\Delta p) = 0$.



(a) Reference field $q(x,t)$ **(b)** Transformed field $q(G(x,t))$

Figure 4.17: Traveling wave $q(x,t) = \exp(-(x-x_0)-ct)^2/\sigma^2)$ evaluated in its reference frame (a) and co-moving frame (b).

In the pure advection case, the bound Eq. (4.16) can be calculated from the initial condition $q(x,0) = q_0(x)$, since the derivative with respect to the spatial variable is unchanged. This fact should be acknowledged when simulating the FOM, since the interpolation error limits the overall precision of the ROM as will be shown in Section 4.3.2. In the general case, the initial condition can still be a good approximation for the overall interpolation error.

Note that to perform rotations or translations on non-periodic grids, we redefine our rectangular domain $\mathbb{D}$ to contain all coordinates in the co-moving system $G(x,t)$. Therefore the snapshot data has to be padded with additional data, which will be explained in Section 4.3.4.

4.3.2 Optimization Goals of the sPOD

Let $\{\mathcal{T}^{G_k}\}_{k=1,\dots,f}$ be a set of predefined operators with transformations G_k as defined in Section 4.3.1 and $\{T^{G_k}\}_{k=1,\dots,f}$ the discrete counterparts of $\mathcal{T}^{G_k}$. Each transformation G_k maps the coordinate system from a co-moving frame to a reference frame. For simplicity we will write $\mathcal{T}^k \coloneqq \mathcal{T}^{G_k}$, $\mathcal{T}^{-k} \coloneqq \mathcal{T}^{G_k^{-1}}$ and $T^k \coloneqq T^{G_k}$, $T^{-k} \coloneqq T^{G_k^{-1}}$. For a given data field $\mathbf{Q} = [q(x_i,t_j)]_{ij} \in \mathbb{R}^{M\times N_t}$ we seek for approximations

of the form:

$$q(x,t) \approx \tilde{q}(x,t) := \sum_{k=1}^{f} \mathcal{T}^k q^k(x,t)\,, \qquad \text{(continuous)} \qquad (4.18)$$

$$\mathbf{Q} \approx \tilde{\mathbf{Q}} := \sum_{k=1}^{f} T^k \mathbf{Q}^k\,, \qquad \text{(discrete)} \qquad (4.19)$$

where the *co-moving fields* $\{\mathbf{Q}^k\}_{k=1,\dots,f}$ (i.e. $\{q^k\}_{k=1,\dots,f}$) need to be determined. One example of such decomposition is shown for multiple traveling waves in Fig. 4.18. Each co-moving field q^k is approximated

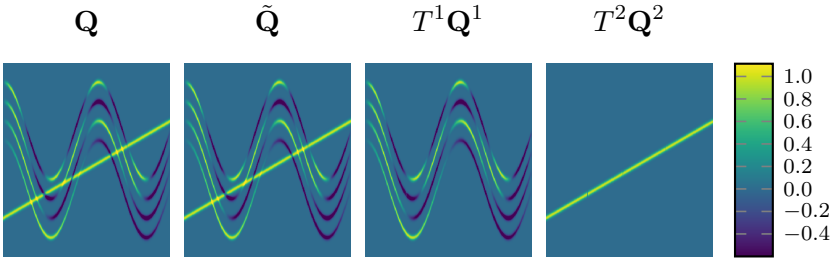


Figure 4.18: Decomposition Eq. (4.19) for multiple traveling waves using the sPOD-$\mathcal{J}_2$ formulation. In this example we use two frames: $\tilde{\mathbf{Q}} = T^1\mathbf{Q}^1 + T^2\mathbf{Q}^2$ with corresponding shifts $\Delta_1(t) = -0.25\sin(7\pi t)$ and $\Delta_2(t) = -t$, $t \in [0, 0.5]$. Shown is the input snapshot matrix $\mathbf{Q} = [q(x_i, t_j)]_{i,j}$ and the resulting approximation $\tilde{\mathbf{Q}}$ with the shifted snapshot matrices $T^1(\mathbf{Q}^1), T^2(\mathbf{Q}^1)$. The co-moving ranks are $(r_1, r_2) = (4, 1)$. The input data are parametrized as follows: $q(x,t) = \exp(-(x - (\Delta_2(t) + 0.2))^2/\delta^2) + \sum_{r=1}^{4} \sin(4\pi r t)\exp(-(x - (\Delta_1(t) + 0.25 + 0.1r))^2/\delta^2)$ for $x \in [0,1]$, $\delta = 0.0125$.

with r_k POD modes $\{\psi_i^k(x)\}_{i=1,\dots,r_k}$:

$$q^k(x,t) = \sum_{i=1}^{r_k} a_i^k(t)\psi_i^k(x) \quad \text{and} \quad a_i^k(t) = \langle q^k(x,t), \psi_i^k(x)\rangle\,. \qquad (4.20)$$

In the discrete setting, the POD modes are computed by an SVD of the co-moving data field $\mathbf{Q}^k \in \mathbb{R}^{M\times N_t}$:

$$\mathbf{Q}^k = \mathbf{\Psi}^k \mathbf{\Sigma}^k (\mathbf{V}^k)^\top \quad k = 1, \dots, f\,. \qquad (4.21)$$

Here, $\mathbf{\Sigma}^k = \text{diag}\,(\sigma_1^k, \ldots, \sigma_m^k)$, $m = \min(M, N_t)$ is a diagonal matrix containing the singular values $\sigma_1^k \geq \sigma_2^k \geq \ldots \sigma_m^k$ and $\mathbf{\Psi}^k \in \mathbb{R}^{M \times m}$, $\mathbf{V}^k \in \mathbb{R}^{N_t \times m}$ are orthogonal matrices containing the left and right singular vectors. The POD modes are contained in the first r_k columns of $\mathbf{\Psi}^k = [\psi_j^k(x_i)]_{ij} \in \mathbb{R}^{M \times m}$. For maximal efficiency we aim for a small number of modes $r = \sum_{k=1}^{f} r_k \ll N_t$. Note that in contrast to the POD (see Section 3.1) the linear subspace created by Eqs. (4.18) and (4.19) changes with time, if any mapping G^k is time-dependent. Therefore, Eq. (4.18) is a non-linear mapping. In the following, we will focus on the discrete notation and only go back to the continuous representation if needed.

The individual optimization goals that lead to a low-rank decomposition of Eq. (4.19) can vary depending on the application. In this thesis, we focus on the optimization goals that have been first presented in [114]:

$$\mathcal{J}_2[\{\mathbf{Q}^k\}_k] = \sum_{k=1}^{f} \frac{1}{2} \left\| \mathbf{Q}^k - \left\lfloor \mathbf{Q}^k \right\rfloor_{r_k} \right\|_F^2 = \frac{1}{2} \sum_{k=1}^{f} \sum_{j=r_k+1}^{m} (\sigma_j^k)^2 \tag{4.22}$$

$$\mathcal{J}_1[\{\mathbf{Q}^k\}_k] = \sum_{k=1}^{f} \left\| \mathbf{Q}^k \right\|_* = \sum_{k=1}^{f} \sum_{j=1}^{m} \sigma_j^k \tag{4.23}$$

subject to the constraint Eq. (4.19). In Eq. (4.22) $\left\lfloor \mathbf{Q}^k \right\rfloor_{r_k}$ denotes the rank r_k approximation of the matrix $\mathbf{Q}^k$.

The optimization goal $\mathcal{J}_2$ aims to find a set of co-moving fields $\{\mathbf{Q}^k\}_k$ that minimize the difference with their rank r_k approximation, i.e. it minimizes the truncated singular values in each frame. Therefore, the *co-moving ranks* r_k have to be chosen a priori. For complicated systems, this choice is often unclear but critical for the quality of the decomposition. For this reason, $\mathcal{J}_1$ is preferred, since it is convex [114] and the truncation rank is a result of the one-norm minimization over the set of all singular values $\{\sigma_j^k\}_{j,k}$. $\mathcal{J}_1$ serves as a heuristic measure of the singular value decay. Although optimizing $\mathcal{J}_1$ leads to fast decaying singular values of the co-moving fields $\mathbf{Q}^k$, they must not be strictly low-rank. Unfortunately, $\mathcal{J}_1$ is difficult to optimize in a gradient-based procedure as pointed out by [114]. Hence, a formu-

lation based on the alternating direction method is introduced, which alleviates this difficulty by introducing proximal operators to efficiently minimize one-norms in Section 4.3.5. Furthermore, Section 4.3.3 derives an alternative formulation to the gradient-based procedure of the $\mathcal{J}_2$ optimization goal, that was presented in [114]. The new formulation is more efficient as it only requires recursive applications of truncated singular value decompositions.

4.3.3 Optimization of $\mathcal{J}_2$ via Iterative Thresholding

Given a matrix $\mathbf{Q} \in \mathbb{R}^{M \times N_t}$ and a set of transport operators T^k and ranks r_k, $k = 1, \ldots, f$. The sequence given by the iterations of Algorithm 3 converges to a solution of the $\mathcal{J}_2$-sPOD approximation problem

$$\min_{\{\mathbf{Q}^k\}_k^f} \sum_{k=1}^{f} \frac{1}{2} \left\| \mathbf{Q}^k - \left\lfloor \mathbf{Q}^k \right\rfloor_{r_k} \right\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \mathbf{Q} = \sum_{k=1}^{f} T^k \mathbf{Q}^k \,. \tag{4.24}$$

First, note that according to the Eckart-Young Theorem [42] the minimum of the individual terms $\left\| \mathbf{Q}^k - \left\lfloor \mathbf{Q}^k \right\rfloor_{r_k} \right\|_{\mathrm{F}}^2$ of $\mathcal{J}_2$ is given by the truncated SVD: $\mathrm{svd}(\mathbf{Q}^k, r_k) = \mathbf{\Psi}_{r_k}^k \mathbf{\Sigma}_{r_k}^k (\mathbf{V}_{r_k}^k)^\top$, where $\mathbf{\Psi}_{r_k}^k \in \mathbb{R}^{M \times r_k}$, $\mathbf{V}_{r_k}^k \in \mathbb{R}^{N_t \times r_k}$ contain the r_k singular vectors corresponding to the r_k largest singular values contained in $\mathbf{\Sigma}_{r_k}^k \in \mathbb{R}^{r_k \times r_k}$. After the SVD truncation of $\mathbf{Q}^k$ in line 7 of Algorithm 3, the constraint $\mathbf{Q} = \sum_{k=1}^{f} T^k \mathbf{Q}^k$ is usually not satisfied. In order to incorporate the constraint, we use the redistribution identity

$$\hat{\mathbf{Q}}^k = \mathbf{Q}^k + \alpha_k T^{-k} \mathbf{R} \quad \text{with} \quad 1 = \sum_{k=1}^{f} \alpha_k \,, \tag{4.25}$$

where $\mathbf{R} = \mathbf{Q} - \tilde{\mathbf{Q}}$, with $\tilde{\mathbf{Q}} = \sum_{k=1}^{N} T^k(\mathbf{Q}^k)$. Note that after redistribution, the $\hat{\mathbf{Q}}^k$ fulfill the constraint for any choice $\{(\alpha_1, \ldots, \alpha_f) \in \mathbb{R}_+^f \mid$

$1 = \sum_{k=1}^{f} \alpha_k\}$, since:

$$\sum_{k=1}^{f} T^k \hat{\mathbf{Q}}^k = \sum_{k=1}^{f} T^k(\mathbf{Q}^k + \alpha_k T^{-k}\mathbf{R}) \tag{4.26}$$

$$= \tilde{\mathbf{Q}} + \sum_{k=1}^{f} \alpha_k \underbrace{T^k T^{-k}}_{\approx \mathbb{1}} (\mathbf{Q} - \sum_{p=1}^{f} T^p \mathbf{Q}^p)) \tag{4.27}$$

$$= \tilde{\mathbf{Q}} + \left(\mathbf{Q} - \tilde{\mathbf{Q}}\right) \sum_{k=1}^{f} \alpha_k = \mathbf{Q}\,. \tag{4.28}$$

Remark. *As stated in Eq.* (4.15)*, the shift transformations are not exact because of interpolation errors and thus* $T^k T^{-k} \approx \mathbb{1}$ *in Eq.* (4.27) *is only approximate. However, an iterative redistribution using Eq.* (4.25) *will still converge to machine precision, since the introduced interpolation errors will be always split among the contributing frames.*

Usually, the residual $\mathbf{R}$ is redistributed equally by setting $a_k = 1/f$ for all k, as done in Algorithm 3. Therefore, by iteratively distributing the residual $\mathbf{R} = \mathbf{Q} - \tilde{\mathbf{Q}}$ over the individual frames $\mathbf{Q}^k$ using Eq. (4.25) we always fulfill the constraint, while minimizing the individual terms in $\mathcal{J}_2$ using the truncated SVD.

Algorithm 3 shifted POD $\mathcal{J}_2$ optimization

Require: $\mathbf{Q} \in \mathbb{R}^{M \times N_t}$, $\{G^k\}_{k=1,\ldots,f}$, $\{r^k\}_{k=1,\ldots,f}$

```
init Q^k = 0, Q̃ = 0
while not converged do
    residual R = Q − Q̃
    for frame k = 1, ..., f do
        compute shifted residuum R^k = T^{−k}(R)
        add residuum to frame Q^k ← Q^k + R^k/f
        decompose and truncate Q^k ← svd(Q^k, r_k)
    end for
    update approximation Q̃ = Σ_{k=1}^{f} T^k(Q^k)
end while
return {Q^k}
```

Stopping Criteria

The decomposition into co-moving frames yields two different kinds of errors, which limit the convergence $\|\mathbf{R}_i\|_\mathrm{F} \to 0$ of the residual (line 3 in Algorithm 3) for an increasing number of iterations $i \to \infty$. The first error is caused by the truncation of the co-moving frames, in case the exact co-moving ranks are larger than the truncation rank of the algorithm. The second error is due to the interpolation error of the transport operators (see Section 4.3.1). Algorithm 3 should terminate if either of the two errors dominates. This is why we introduce the following two stopping criteria:

Criterion 1 Algorithm 3 is stopped at iteration i if the sequence of residuals $(\mathbf{R}_0, \mathbf{R}_1, \mathbf{R}_2, \dots)$ fulfills

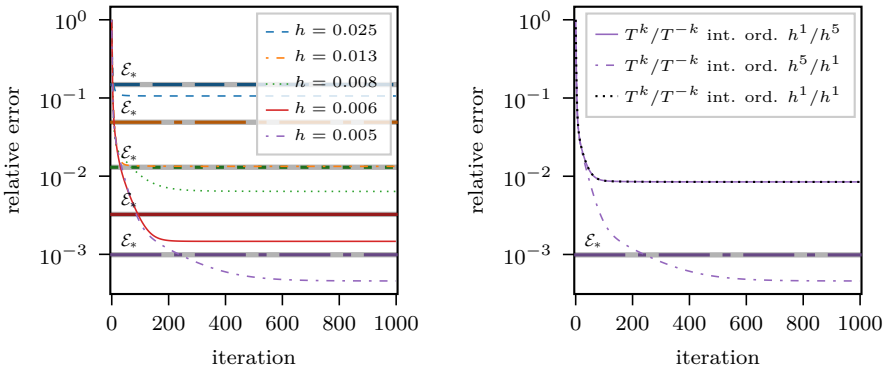
$$\frac{\|\mathbf{R}_i\|_\mathrm{F} - \|\mathbf{R}_{i-\Delta i}\|_\mathrm{F}}{\|\mathbf{R}_i\|_\mathrm{F}} \leq \texttt{Rtol} \tag{4.29}$$

for some $\Delta i \in \mathbb{N}$ and $i \geq \Delta i$. The criterion Eq. (4.29) is satisfied if the residuals norm relative to its current value has not changed more than $\texttt{Rtol} > 0$ during Δi iterations.

Criterion 2 As stated in Section 4.3.1, the overall error of the decomposition is limited by the interpolation error of the shift operators. Therefore, it may serve as a stopping criterion for the presented algorithms. The interpolation error can be calculated from the error bound Eq. (4.16) if the maximum of the $(n+1)$th order derivatives is known. Alternatively, we propose to use:

$$\frac{\|\mathbf{R}_i\|_\mathrm{F}}{\|\mathbf{Q}\|_\mathrm{F}} \leq \mathcal{E}_* \quad \text{with} \quad \mathcal{E}_* = \max_{k=1,\dots,f} \frac{\left\|\mathbf{Q} - T^k T^{-k} \mathbf{Q}\right\|_\mathrm{F}}{2\,\|\mathbf{Q}\|_\mathrm{F}} \tag{4.30}$$

as a stopping criterion. The stopping criterion is visualized for the convergence of Algorithm 3 in Fig. 4.19 for different lattice spacings h (Fig. 4.19a) and interpolation order of the operators (Fig. 4.19b). A nice property of the presented algorithm (Algorithm 3) is that for the backward transformation T^{-k} a lower interpolation order as the forward transformation T^k can be used since the redistribution compensates the additional error. This is numerically investigated in Fig. 4.19b



(a) Convergence for different lattice spacings

(b) Convergence order of transformations

Figure 4.19: Convergence of the shifted POD $\mathcal{J}_2$ optimization (Algorithm 3) for the example shown in Fig. 4.18. For all optimizations we use the exact truncation ranks $(r_1, r_2) = (4, 1)$ in the co-moving frames. In a) the convergence of the algorithm is studied for different lattice spacings h. The stopping criterion $\mathcal{E}_*$ is marked as a straight line of the same line style with a grey background. Figure b) shows the convergence of the algorithm for $h = 0.005$ using lower accuracy interpolation for the backward or/and forward transform.

for the example shown in Fig. 4.18. In Fig. 4.19b we compare the convergence of the algorithm using different combinations of 5th and 1st order interpolation for T^k and T^{-k}.

Note that criterion 1 is already sufficient for terminating Algorithm 3. However, criterion 2 is necessary to avoid that the interpolation errors are captured as noise in additional modes of the co-moving fields.

Total Variation Algorithm 3 generates low-rank fields by projecting $\mathbf{Q}^k \in \mathbb{R}^{M\times N_t}$ onto the manifold of $\operatorname{rank}(\mathbf{Q}^k) = r_k$ matrices in each frame $k = 1,\dots,f$. However, low rankness of the resulting fields $q^k(x,t) = \sum_{i=1}^{r_k} a_i^k(t)\psi(x)$ does not imply that the time amplitudes $a_i^k(t)$ of the decomposition are a smooth function in time. But smoothness in the parameter space is necessary for an interpolatory data-driven model. Consequently, we include a total variation (TV) minimization in Algorithm 5 to smoothen the time amplitudes $\mathbf{V}^k = [a_j^k(t_i)]_{i,j}$ in each frame k. The TV minimization is based on a primal-dual algorithm (see Algorithm 4 taken from [23, p.55]) for the TV-L1 discrete denoising model:

$$\min_{\tilde{\mathbf{V}}\in\mathbb{R}^{N_t\times r}} \left\|\mathbf{D}^+\tilde{\mathbf{V}}\right\|_{\mathrm{F}} + \lambda_{\mathrm{TV}} \left\|\tilde{\mathbf{V}} - \mathbf{V}\right\|_{\mathrm{F}}, \tag{4.31}$$

where

$$(\mathbf{D}^+\mathbf{V})_{i,j} = \begin{cases} V_{i+1,j} - V_{i,j} & \text{for} \quad i < N_t \\ 0 & \text{for} \quad i = N_t \end{cases} \tag{4.32}$$

is the forward derivative on the columns of $\mathbf{V} = [V_{i,j}] \in \mathbb{R}^{N_t\times r}$. The primal-dual algorithm (Algorithm 4) makes use of the operators:

$$\operatorname{proj}(x) = \frac{x}{\max\left(|x|, 1\right)}, \tag{4.33}$$

$$\mathcal{S}_\tau^f(x) = \begin{cases} x - \tau & x > f + \tau \\ x + \tau & x < f - \tau \\ f & |x - f| \le \tau\,. \end{cases} \tag{4.34}$$

The operators are evaluated element-wise on the input matrices in Algorithm 4. For the convenience of the reader, we show the soft thresholding operator $\mathcal{S}_\tau^f$ in Fig. 4.20.

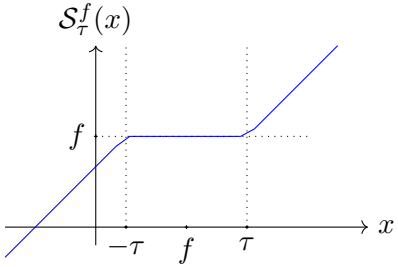


Figure 4.20: Soft thresholding operator $\mathcal{S}_\tau^f$ defined in Eq. (4.34).

Algorithm 4 total variation minimization $\mathrm{TV}(\mathbf{V}, \lambda_{\mathrm{TV}}, N_{\mathrm{TV}})$ [23]

Require: $\mathbf{V} \in \mathbb{R}^{N_t \times r}$, N_{TV}, λ_{TV}

init $\mathbf{P} = \mathbf{D}^+\mathbf{V}$
init normalized $\widehat{\mathbf{V}} = \mathrm{proj}(\mathbf{V})$
init $\tau = 0.02, l_2 = 8, \sigma = (l_2\tau)^{-1}$
for $k = 1, \ldots, N_{\mathrm{TV}}$ **do**
    project element-wise $\mathbf{P} \leftarrow \mathrm{proj}(\mathbf{P} + \sigma\mathbf{D}^+\mathbf{V})$
    shrink element-wise $\tilde{\mathbf{V}} \leftarrow \mathcal{S}^{\widehat{\mathbf{V}}}_{\lambda_{\mathrm{TV}}\tau}(\mathbf{V} - \tau(\mathbf{D}^+)^\top\mathbf{P})$
    update $\mathbf{V} \leftarrow 2\tilde{\mathbf{V}} - \mathbf{V}$
end for
return V

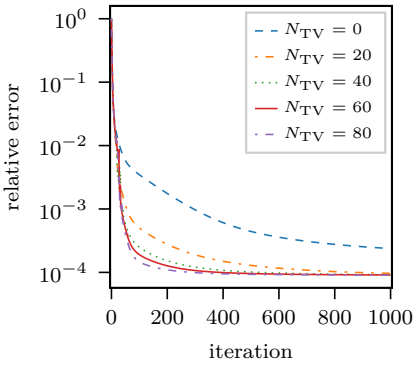
The $\mathcal{J}_2$ shifted POD algorithm with total variation minimization listed in Algorithm 5, was tested on the two frames example shown in Fig. 4.18. The results are shown in Fig. 4.21. Figure 4.21a indicates that the total variation sub-steps in Algorithm 5 improve the convergence of the overall algorithm, while Fig. 4.21b shows that the resulting time amplitudes are smooth. The TV-minimization in line 8 is comparatively cheap to the computation of the SVD (line 7), since it is only applied to the columns of a small matrix $\mathbf{V}_k \in \mathbb{R}^{N_t \times r_k}$.

Algorithm 5 optimization shifted POD $\mathcal{J}_2$ with total variation

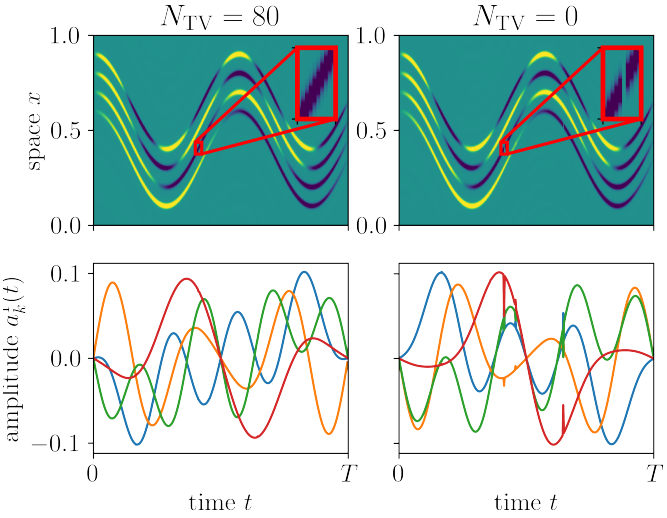
Require: $\mathbf{Q} \in \mathbb{R}^{M \times N_t}$, $\{G^k\}_{k=1,\dots,f}$, $\{r^k\}_{k=1,\dots,f}$, N_{TV}
1: init $\mathbf{Q}^k = 0, \tilde{\mathbf{Q}} = 0$, $\lambda_{\mathrm{TV}} = 1$
2: **while** not converged **do**
3: residual $\mathbf{R} = \mathbf{Q} - \tilde{\mathbf{Q}}$
4: **for** frame $k = 1, \dots, f$ **do**
5: compute shifted residuum $\mathbf{R}^k = T^{-k}(\mathbf{R})$
6: add residuum to frame $\mathbf{Q}^k \leftarrow \mathbf{Q}^k + \mathbf{R}^k / f$
7: decompose and truncate $\mathbf{\Psi}^k_{r_k} \mathbf{\Sigma}^k_{r_k} (\mathbf{V}^k_{r_k})^\top = \mathrm{svd}(\mathbf{Q}^k, r_k)$
8: minimize total time variation $\mathbf{V}^k_{r_k} \leftarrow \mathrm{TV}(\mathbf{V}^k_{r_k}, \lambda_{\mathrm{TV}}, N_{\mathrm{TV}})$
9: compute frame field $\mathbf{Q}^k \leftarrow \mathbf{\Psi}^k_{r_k} \mathbf{\Sigma}^k_{r_k} (\mathbf{V}^k_{r_k})^\top$
10: **end for**
11: update approximation $\tilde{\mathbf{Q}} = \sum_{k=1}^{f} T^k(\mathbf{Q}^k)$
12: **end while**
13: **return** $\{\mathbf{Q}^k\}_{k=1,\dots,f}$

4.3.4 $\mathcal{J}_2$ Optimization on Non-Periodic Domains

For completeness, we include the treatment of non-periodic boundaries, although they are not used in this thesis. In the presence of non-periodic boundary conditions, transformations $G^{-1}(\cdot, t) : \mathbb{D} \to \tilde{\mathbb{D}}$ of the the form Eq. (4.14) map points inside the domain $\mathbb{D}$ onto grid points $\tilde{\mathbb{D}}$, which may not be included in $\mathbb{D}$. As suggested in [114], we therefore extend the original domain $\mathbb{D}$ by $\mathbb{D}^{\mathrm{ext}} = \mathbb{D} \cap \tilde{\mathbb{D}}$ and relax the



(a) Convergence for N_{TV} total variation sub-steps



(b) Smoothness of $\mathcal{T}^1 q^1(x,t)$, $q^1(x,t) = \sum_{i=1}^{4} a_i^1(t)\psi_i^1(x)$ within time.

Figure 4.21: Convergence of the shifted POD $\mathcal{J}_2$ optimization (Algorithm 5) for the example shown in Fig. 4.18. For all optimizations we use the exact truncation ranks $(r_1, r_2) = (4, 1)$ in the co-moving frames. In a) the convergence of the algorithm is studied for different numbers of iterations N_{TV} in the total variation optimization. Figure b) compares the first frame $T^1\mathbf{Q}^1$ and its time amplitudes $a_k^1(t), k = 1, \ldots, 4$ with total variation optimization and without. The insets show a zoom into the wave structure.

constraint Eq. (4.18) to

$$w(x,t)\left(q(x,t)-\sum_{k=1}^{f}\mathcal{T}^k q^k(x,t)\right)=0\,,\quad w(x,t)=\begin{cases}1 & x\in\mathbb{D}\\ 0 & x\in\mathbb{D}^{\text{ext}}\end{cases}. \tag{4.35}$$

The relaxed constraint Eq. (4.35) is implemented in Algorithm 6 by replacing the residual $\mathbf{R}\in\mathbb{R}^{\widehat{M}\times N_t}$ by $\mathbf{R}=\mathbf{W}\odot(\widehat{\mathbf{Q}}-\tilde{\mathbf{Q}})$. Here $\mathbf{W}=[w(x_i,t_j)]\in\mathbb{R}^{\widehat{M}\times N_t}$ defines the weights and $\widehat{\mathbf{Q}}\in\mathbb{R}^{\widehat{M}\times N_t}$ with entries

$$\widehat{\mathbf{Q}}_{ij}=\begin{cases}q(x_i,t_j) & x_i\in\mathbb{D}\\ 0 & x_i\in\mathbb{D}^{\text{ext}}\end{cases}\qquad i=1,\dots,\widehat{M},j=1,\dots,N_t \tag{4.36}$$

the zero-padded data field with $\widehat{M}$ being the total number of grid points in $\mathbb{D}\cup\mathbb{D}^{\text{ext}}$. The generalized procedure is listed in Algorithm 6.

Algorithm 6 Shifted POD $\mathcal{J}_2$ algorithm for non-periodic domains

Require: $\mathbf{Q}\in\mathbb{R}^{M\times N_t}$, $\{G^k\}_{k=1,\dots,f}$, $\{r_k\}_{k=1,\dots,f}$

init $\mathbf{Q}^k,\tilde{\mathbf{Q}},\widehat{\mathbf{Q}},\mathbf{W}\in\mathbb{R}^{\widehat{M}\times N_t}$ with $\mathbf{Q}^k=\tilde{\mathbf{Q}}=\widehat{\mathbf{Q}}=\mathbf{W}=0$
set $\widehat{\mathbf{Q}}_{\widehat{m}n}=\mathbf{Q}_{mn}$ for all $\widehat{m},n$ with $\boldsymbol{x}_{\widehat{m}}=\boldsymbol{x}_m\in\mathbb{D}$
set $\mathbf{W}_{\widehat{m}n}=1$ for all $\widehat{m},n$ with $\boldsymbol{x}_{\widehat{m}}\in\mathbb{D}$

while not converged **do**
  residual $\mathbf{R}=\mathbf{W}\odot(\widehat{\mathbf{Q}}-\tilde{\mathbf{Q}})$
  for frame $k=1,\dots,f$ **do**
    compute shifted residuum $\mathbf{R}^k=T^{-k}(\mathbf{R})$
    add residuum to frame $\mathbf{Q}^k\leftarrow\mathbf{Q}^k+\mathbf{R}^k/f$
    decompose and truncate $\mathbf{Q}^k\leftarrow\text{svd}(\mathbf{Q}^k,r_k)$
  end for
  update approximation $\tilde{\mathbf{Q}}=\sum_{k=1}^{f}T^k(\mathbf{Q}^k)$
end while
return $\{\mathbf{Q}^k\}_{k=1,\dots,f}$

4.3.5 Optimization of $\mathcal{J}_1$ and the Shifted Robust Principal Component Analysis

Instead of minimizing $\mathcal{J}_2$, one can minimize the Schatten 1-norm $\left\|\mathbf{Q}^k\right\|_* = \sum_i \sigma_i^k$ for each frame k. The optimization problem can be stated as follows: For given transformations $\{T^k\}_{k=1,\dots,f}$ and $\mathbf{Q} \in \mathbb{R}^{M\times N_t}$, $M > N_t$ with $\mathbf{Q} = [q(x_i,t_j)]_{i,j}$ find $\{\mathbf{Q}^k \in \mathbb{R}^{M\times N_t}\}_{k=1,\dots,f}$

$$\min_{\mathbf{Q}^k} \sum_{k=1}^{f} \left\|\mathbf{Q}^k\right\|_* \quad \text{s.t.} \quad \mathbf{Q} = \sum_{k=1}^{f} T^k(\mathbf{Q}^k)\,. \tag{4.37}$$

This $\mathcal{J}_1$ optimization problem has two major advantages over optimizing $\mathcal{J}_2$: 1.) It avoids the choice of the individual ranks r_k in each frame k that is required by the $\mathcal{J}_2$ optimization. The choice of r_k is often arbitrary, although it crucially determines the accuracy of the overall decomposition. 2.) As already pointed out in Section 4.3.2, in contrast to $\mathcal{J}_1$, $\mathcal{J}_2$ is not known to be convex. Therefore, it is not guaranteed that the $\mathcal{J}_2$ optimization results in a global minimum. However, the slow convergence of $\mathcal{J}_1$, which was observed in [114], made it unfeasible for realistic applications.

Therefore, in this work we use an alternative formulation to the one presented in [114], based on an augmented Lagrangian with multiplier $\mathbf{Y} \in \mathbb{R}^{M\times N_t}$ [8, 91]:

$$\begin{aligned}\mathcal{L}(\{\mathbf{Q}^k\}_k, \mathbf{Y}) = \sum_{k=1}^{f} \left\|\mathbf{Q}^k\right\|_* + \frac{\eta}{2}\left\|\mathbf{Q} - \sum_{k=1}^{f} T^k(\mathbf{Q}^k)\right\|^2 \\ + \left\langle \mathbf{Y}, \mathbf{Q} - \sum_{k=1}^{f} T^k(\mathbf{Q}^k)\right\rangle .\end{aligned} \tag{4.38}$$

With the new formulation, we can employ the *alternating direction method* (ADM) [8], which allows a rapid minimization of $\mathcal{J}_1$ with help of the singular value thresholding operator [17, 67]:

$$\operatorname{svt}(\mathbf{A},\tau) = \mathbf{U}\mathcal{S}_\tau^0(\mathbf{\Sigma})\mathbf{V}^* \tag{4.39}$$

$$= \arg\min_{\tilde{\mathbf{A}}} \tau\left\|\tilde{\mathbf{A}}\right\|_* + \frac{1}{2}\left\|\tilde{\mathbf{A}} - \mathbf{A}\right\|_{\mathrm{F}}^2 , \tag{4.40}$$

which technically boils down to a singular value decomposition of the matrix $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$ with a soft thresholding operator $\mathcal{S}_\tau^0$ defined in Eq. (4.34). The thresholding operator $\mathcal{S}_\tau^f$ is displayed in Fig. 4.20.

In ADM we minimize $\mathcal{L}(\{\mathbf{Q}^k\}_k, \mathbf{Y})$ iteratively in every direction $\mathbf{Q}^p \in \mathbb{R}^{M\times N_t}$, $p = 1, \ldots, f$ after which we update the multiplier matrix $\mathbf{Y} \leftarrow \mathbf{Y} + \eta(\mathbf{Q} - \sum_k T^k(\mathbf{Q}^k))$. To minimize $\mathcal{L}$ w.r.t. $\mathbf{Q}^p$ we first reformulate Eq. (4.38):

$$\arg\min_{\mathbf{Q}^p} \mathcal{L} = \arg\min_{\mathbf{Q}^p} \frac{1}{\eta} \|\mathbf{Q}^p\|_* + \left\| \mathbf{Q} - \sum_{k=1}^{f} T^k(\mathbf{Q}^k) + \frac{1}{\eta}\mathbf{Y} \right\|_{\mathrm{F}}^2 . \quad (4.41)$$

Note that since T^{-p} leaves the Frobenius norm unaltered we can rewrite

$$\left\| \mathbf{Q} - \sum_{k=1}^{f} T^k(\mathbf{Q}^k) + \frac{1}{\eta}\mathbf{Y} \right\|_{\mathrm{F}} = \left\| T^{-p}(\mathbf{Q} - \sum_{k=1}^{f} T^k(\mathbf{Q}^k) + \frac{1}{\eta}\mathbf{Y}) \right\|_{\mathrm{F}} \quad (4.42)$$

$$= \left\| \mathbf{Q}^p - T^{-p}(\mathbf{Q} - \sum_{\substack{k=1 \\ k\neq p}}^{f} T^k(\mathbf{Q}^k) + \frac{1}{\eta}\mathbf{Y}) \right\|_{\mathrm{F}} . \quad (4.43)$$

Remark. *T^{-p} leaves the Frobenius norm unaltered up to interpolation errors, but the interpolation errors only contribute as an offset in the overall minimum.*

Using Eq. (4.43) for the second term in Eq. (4.41) and comparing it to Eq. (4.40), we see that $\arg\min_{\mathbf{Q}^p} \mathcal{L}$ is given by $\mathrm{svt}(\tilde{\mathbf{Q}}^p, \eta^{-1})$ with matrix $\tilde{\mathbf{Q}}^p = \mathcal{T}^{-p}(\mathbf{Q} - \sum_{k=1,k\neq p}^{f} \mathcal{T}^k(\mathbf{Q}^k) + \frac{1}{\eta}\mathbf{Y})$. With this result, we can formulate the ADM in Algorithm 7. Note that according to [91] the algorithm outlined in Algorithm 7 is not exact, since we only apply a single step in the direction $\mathbf{Q}^p$ before updating the multiplier.

Compared to Algorithm 3 the $\mathcal{J}_1$ optimization (Algorithm 7) replaces the co-moving ranks r_k by a new parameter η, that chooses r_k by truncating singular values that are smaller then η^{-1}. We call the parameter η stiffness, analogous to the stiffness of a spring, which is motivated by the quadratic term in the minimization Eq. (4.41). When

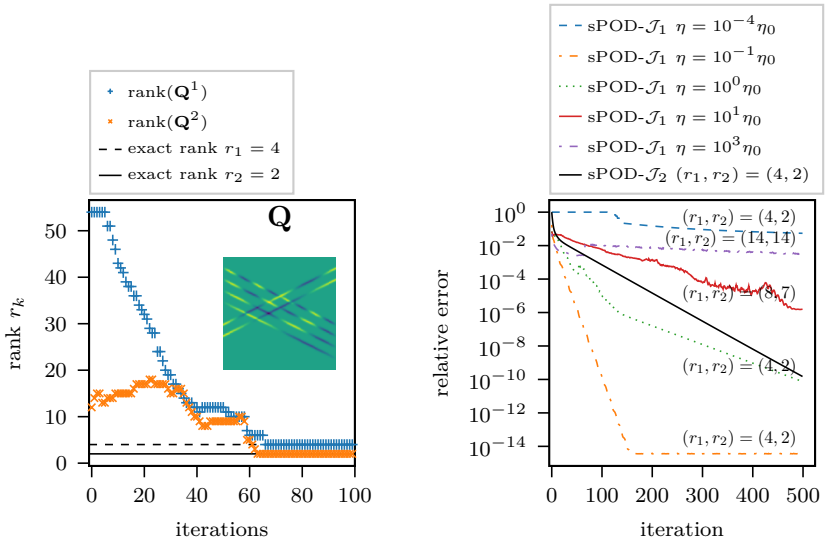
Algorithm 7 ADM for shifted POD $\mathcal{J}_1$ minimization

Require: $\mathbf{Q} \in \mathbb{R}^{m\times n}$, $\{T^k\}_{k=1,\dots,f}$, η
1: init $\mathbf{Q}^k = 0\ \forall\, k, \tilde{\mathbf{Q}} = 0, \mathbf{Y} = 0$
2: **while** not converged **do**
3: **for** frame $p = 1, \dots, f$ **do**
4: compute $\mathbf{Q}^p = T^{-p}(\mathbf{Q} - \sum_{k=1,k\neq p}^{f} T^k(\mathbf{Q}^k) + \frac{1}{\eta}\mathbf{Y})$
5: apply singular value thresholding $\mathbf{Q}^p \leftarrow \mathrm{svt}(\mathbf{Q}^p, \eta^{-1})$
6: **end for**
7: update multiplier $\mathbf{Y} \leftarrow \mathbf{Y} + \eta(\mathbf{Q} - \sum_k T^k(\mathbf{Q}^k))$
8: **end while**
9: **return** $\{\mathbf{Q}^k\}$

re-scaling Eq. (4.41) by η and $\hat{\mathbf{Y}} = \frac{\mathbf{Y}}{\eta}$, one can interpret the second term $\mathcal{V}(\mathbf{Q}_k) = \eta \left\| \mathbf{Q} - \sum_{k=1}^{f} T^k(\mathbf{Q}^k) + \hat{\mathbf{Y}} \right\|_{\mathrm{F}}^2$ as a spring potential, that is preloaded by $\hat{\mathbf{Y}}$ to match the constraint.

Hence, the parameter η regulates the stiffness of the optimization. If the stiffness is too large ($\eta \to \infty$) the problem becomes ill-conditioned and no minimum is found. In fact, it is proven in [91, Theorem 3] for a similar problem that increasing η rapidly during the iterations may result in a loss of convergence. On the other hand, if the stiffness is too small ($\eta \to 0$) the constraint will not be matched. Specifically for the $\mathcal{J}_1$ problem Eq. (4.37), this can be observed in Fig. 4.22. For small η, all singular values are truncated and the constraint is not matched or converges very slowly (compare the $\eta = 10^{-4}\eta_0$ curve in Fig. 4.22b). In contrast, if η is too large the co-moving ranks will be too large ($\eta \gtrapprox 10\eta_0$). Thus, depending on the choice of η, different convergence results are expected. But usually the suggestion $\eta = \eta_0 = MN_t/(4\,\|\mathbf{Q}\|_1)$ of [19] for a similar problem gives a good starting value for choosing η. As proposed by [19], we keep η fixed during the iterations in Algorithm 7. Based on our studies we suggest choosing $10^{-4} \leq \eta/\eta_0 \leq 1$. Z. Lin et al. propose in [91] to slowly increase the stiffness η_k in every iteration step k to remove the dependence on η. However, as already pointed out in [91], the choice of $\{\eta_k\}$ should be made judicious as to minimize the number of total SVDs, since η^{-1} controls the step size of the method.

We note that Algorithm 7 will not converge to the exact co-moving ranks if the data are distorted by any noise. This noise is usually caused by the interpolation errors of the transformation operators (see Section 4.3.1) or artifacts in the data. Therefore, the example in Fig. 4.22a was tuned such that the shift transformations do not cause any interpolation errors (i.e. the shift is a multiple of the lattice spacing).



(a) Co-moving Ranks for $\eta = \eta_0$ **(b)** Rel. error $\left\|\mathbf{Q} - \tilde{\mathbf{Q}}\right\|_{\mathrm{F}} / \|\mathbf{Q}\|_{\mathrm{F}}$

Figure 4.22: Convergence results for the $\mathcal{J}_1$- and $\mathcal{J}_2$-minimization Algorithms 3 and 7. a) Convergence of the co-moving ranks to the exact numerical ranks $(r_1, r_2) = (4, 2)$. The input snapshot matrix $\mathbf{Q} = [q(x_i, t_j)]_{ij} \in \mathbb{R}^{400,200}$ with $q(x,t) = \sum_{k=1}^{r_1} \sin(kt\pi) \exp(-(x - 0.1k + \Delta_1(t))^2/\delta^2) + \sum_{k=1}^{r_2} \cos(kt\pi) \exp(-(x + 0.2 + 0.1k + \Delta_2(t))^2/\delta^2), \delta = 0.0125, \Delta_{1/2}(t) = \pm t, (x,t) \in [0.5, -0.5] \times [0, 0.5]$ is shown as an inset. b) Comparison of the relative error convergence for different parameters η sampled around $\eta_0 = MN_t/(4\|\mathbf{Q}\|_1)$. The co-moving ranks at iteration 500 are assigned to the corresponding curves.

To predict the right co-moving ranks in the presence of noise, we make the algorithm robust against noise. In the spirit of [91] we therefore

decompose: $\mathbf{Q} = \tilde{\mathbf{Q}} + \mathbf{E}$ in a low-rank part $\tilde{\mathbf{Q}} = \sum_k T^k(\mathbf{Q}^k)$ and sparse component $\mathbf{E}$ that captures the noise:

$$\min_{\mathbf{Q}^k, \mathbf{E}} \sum_{k=1}^{f} \left\| \mathbf{Q}^k \right\|_* + \varsigma \left\| \mathbf{E} \right\|_1 \quad \text{s.t.} \quad \mathbf{Q} = \sum_{k=1}^{f} T^k(\mathbf{Q}^k) + \mathbf{E}\,. \tag{4.44}$$

Here, $\varsigma \geq 0$ and $\|\mathbf{E}\|_1 = \sum_{ij} |E_{ij}|$ is not the usual matrix 1-norm, but the vector 1-norm of a long vector $\mathbf{E} \in \mathbb{R}^{M \times N_t}$.

The corresponding augmented Lagrange function of (4.44) is given by:

$$\begin{aligned} \mathcal{L}_{\mathbf{E}}(\{\mathbf{Q}^k\}_k, \mathbf{E}, \mathbf{Y}) = \frac{\eta}{2} \left\| \mathbf{Q} - \sum_{k=1}^{f} T^k(\mathbf{Q}^k) - \mathbf{E} \right\|_{\mathrm{F}}^2 \\ + \langle \mathbf{Y}, \mathbf{Q} - \sum_{k=1}^{f} T^k(\mathbf{Q}^k) - \mathbf{E} \rangle + \varsigma \left\| \mathbf{E} \right\|_1 + \sum_{k=1}^{f} \left\| \mathbf{Q}^k \right\|_* . \end{aligned} \tag{4.45}$$

Analogous to the derivation in [91], we use the soft thresholding operator to minimize the 1-norm:

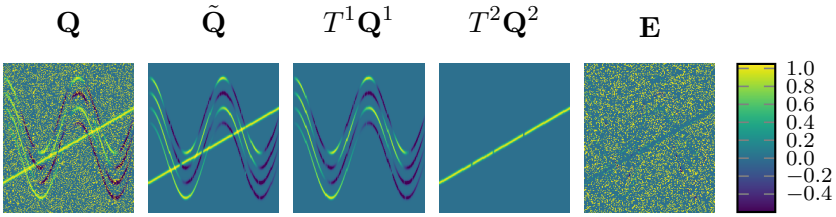
$$\mathcal{S}_\tau^0(\mathbf{A}) = \arg\min_{\tilde{\mathbf{A}}} \tau \left\| \tilde{\mathbf{A}} \right\|_1 + \left\| \tilde{\mathbf{A}} - \mathbf{A} \right\|_{\mathrm{F}}^2 . \tag{4.46}$$

In the robust version of the $\mathcal{J}_1$ algorithm (Algorithm 8) we therefore add another line to minimize with respect to $\mathbf{E}$:

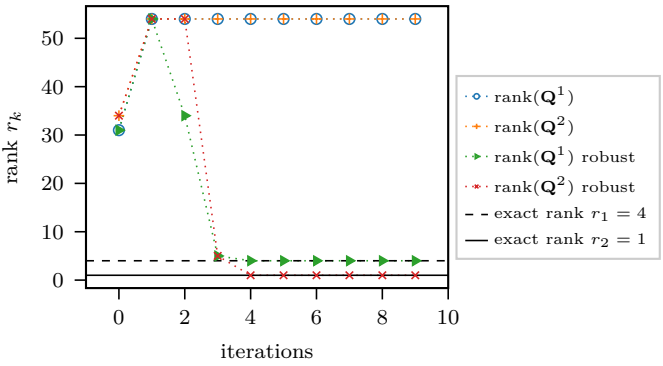
$$\arg\min_{\mathbf{E}} \mathcal{L}_{\mathbf{E}} = \arg\min_{\mathbf{E}} \frac{\varsigma}{\eta} \left\| \mathbf{E} \right\|_1 + \left\| \mathbf{E} - (\sum_{k=1}^{f} T^k(\mathbf{Q}^k) + \frac{1}{\eta}\mathbf{Y} - \mathbf{Q}) \right\|_{\mathrm{F}}^2 . \tag{4.47}$$

The numerical results in Fig. 4.23 indicate that Algorithm 8 is more robust against interpolation noise of the shift operators, corrupted measurements or numerical artifacts. It can be interpreted as a shifted version of the *robust Principle Component Analysis* (shifted rPCA). Its performance scales with the complexity of the singular value decomposition, which can be further accelerated by randomized- or wavelet techniques (see Chapter 3).

As suggested in [19], we choose $\varsigma = 1/\sqrt{\min(M, N_t)}$ for the noise penalty and $\eta = MN_t/(4\left\|\mathbf{Q}\right\|_1)$ for the stiffness parameter in Algorithm 8. For the initial example Fig. 4.18 we show in Fig. 4.23 that



(a) Decomposition with 12.5% noise.



(b) Co-moving Ranks for $\eta = 0.1\eta_0$

Figure 4.23: Convergence results for the $\mathcal{J}_1$- and shifted robust $\mathcal{J}_1$ Algorithms 7 and 8 for the same example shown in Fig. 4.18 but with noise. The noise is computed by randomly setting 12.5% of the input entries of $\mathbf{Q}$ to 1. a) Shown is the input data $\mathbf{Q}$ and its decompostion into a low-rank part $\tilde{\mathbf{Q}} = T^1\mathbf{Q}^1 + T^2\mathbf{Q}^2$ and the noise matrix $\mathbf{E}$. b) Convergence of the co-moving ranks to the exact numerical ranks $(r_1, r_2) = (4, 1)$. The robust algorithm (Algorithm 8) computes the correct numerical ranks of the co-moving frames within 4 iterations. Algorithm 7 without the robustness term overestimates the numerical rank.

Algorithm 8 ADM for shifted rPCA

Require: $\mathbf{Q} \in \mathbb{R}^{M \times N_t}$, $\{T^k\}_{k=1,\dots,f}$, η, ς
1: init $\mathbf{Q}^k = 0 \; \forall \, k, \tilde{\mathbf{Q}} = \mathbf{E} = \mathbf{Y} = 0$
2: **while** not converged **do**
3: **for** frame $p = 1, \dots, f$ **do**
4: compute $\mathbf{Q}^p = T^{-p}(\mathbf{Q} - \sum_{k=1, k\neq p}^{f} T^k(\mathbf{Q}^k) + \frac{1}{\eta}\mathbf{Y})$
5: apply singular value thresholding $\mathbf{Q}^p \leftarrow \mathrm{svt}(\mathbf{Q}^p, \eta^{-1})$
6: **end for**
7: update noise matrix $\mathbf{E} = \mathcal{S}^0_{\varsigma\eta^{-1}}(\mathbf{Q} - \sum_k T^k(\mathbf{Q}^k) + \frac{1}{\eta}\mathbf{Y})$
8: update multiplier $\mathbf{Y} \leftarrow \mathbf{Y} + \eta(\mathbf{Q} - \sum_k T^k(\mathbf{Q}^k) - \mathbf{E})$
9: **end while**
10: **return** $\{\mathbf{Q}^k\}_{k=1,\dots,f}$

even for 12.5% of corrupted or noisy data, the exact ranks can be recovered.

4.3.6 Numerical Results: Two Cylinder Wake Flow

In this section, we revisit the two cylinders example introduced in Section 2.2.3 to study the ability of the sPOD methods for a realistic TDFS. The snapshot set is built from the trajectory corresponding to the path $\Delta_{\mathrm{cyl}}(t, \mu)$ $\mu = 8$ of the second cylinder, sampled with $\Delta t = 1$ in time, resulting in 500 snapshots on a 512×512 grid. The presented analysis is based on two test cases. The first case (decoupled system) is a synthetic test case, in which both cylinders have been simulated separately and added synthetically, to analyze the performance of the sPOD algorithms. In the second test case (coupled system), we analyze the flow with the sPOD when both cylinders are immersed in a flow field at the same time. The test cases will allow us to isolate and understand the effects of the interaction between the two cylinders.

Decoupled System

First, we study the decoupled system in which we simulate the leading cylinder ($\boldsymbol{q}^1 = (u_1^1, u_2^1, p^1)$ labeled with superscript 1) and chaser ($\boldsymbol{q}^{\mathrm{ch}} = (u_1^{\mathrm{ch}}, u_2^{\mathrm{ch}}, p^{\mathrm{ch}})$) separately.

To reduce the data with help of the shifted POD, we introduce the shift transformations. For the leading cylinder, $\mathcal{T}^1(\boldsymbol{q}^1) = \boldsymbol{q}^1$ no shift transformation is needed since the cylinder is stationary. For the second cylinder, we introduce two types of shift transformations:

$$\mathcal{T}^2_{\mathrm{n}}(\boldsymbol{q})(x,y,t) = \boldsymbol{q}(x, y + \Delta_{\mathrm{cyl}}, t) \quad \text{(naive transf.)} \tag{4.48}$$

$$\mathcal{T}^2_{\mathrm{wc}}(\boldsymbol{q})(x,y,t) = \begin{cases} \boldsymbol{q}(x, y + \Delta_{\mathrm{cyl}}(t), t) & \text{for } x < x_2 + R \\ \boldsymbol{q}(x, y + \Delta_{\mathrm{cyl}}(\frac{x-x_2}{u_\infty} - t), t) & \text{for } x \geq x_2 + R \end{cases} \tag{4.49}$$

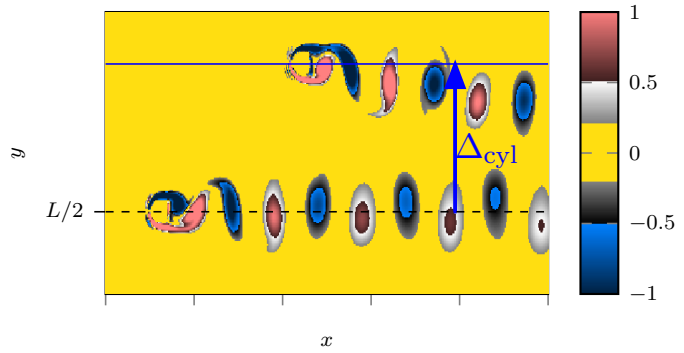
(wake corrected transf.).

The two different shifts are visualized for one snapshot in Fig. 4.24. The corresponding interpolation errors of the transformations are stated in Table 4.2.

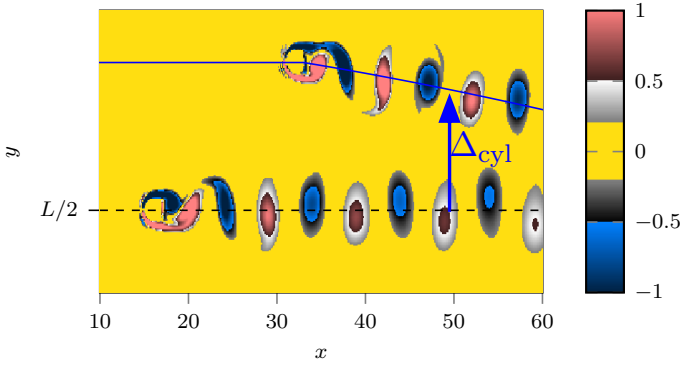
	u_1	u_2	p
interp. err. naive	8.8e-05	1.0e-03	4.7e-03
interp. err. wake corr.	1.3e-04	1.3e-03	5.2e-03
stiffness parameter η/η_0	5.0e-03	4.0e-04	1.0e-03

Table 4.2: Relative interpolation error $\mathcal{E}_*$ defined in Eq. (4.30) and stiffness parameter relative to $\eta_0 = MN_t/(4\,\|\mathbf{Q}\|_1)$ of Algorithm 7 listed for the different flow quantities of the decoupled system.

Note that for the decoupled system we do not need to separate the two frames for each cylinder because they have been simulated separately. Therefore, Algorithms 3 and 5 are not needed. However, compensating the movement of the chaser by applying the time-dependent transformations $\boldsymbol{q}^2 := \mathcal{T}^{-2}_{\mathrm{n}}(\boldsymbol{q}^{\mathrm{ch}})$ and $\boldsymbol{q}^2 := \mathcal{T}^{-2}_{\mathrm{wc}}(\boldsymbol{q}^{\mathrm{ch}})$ on the second cylinder enables a better singular value decay as shown in Fig. 4.25 for the pressure field. Similar observations hold for both velocity components (see Fig. C.2). For all components, we observe that the singular values with the wake-corrected shift decay faster than the ones with a naive shift at the cylinder position. Unfortunately, even for the wake-corrected shift the singular values that correspond to the second cylinder exhibit significantly slower decay. This can be explained by the additional



(a) Naive shift $\mathcal{T}_{\mathrm{n}}^2$



(b) Wake corrected shift $\mathcal{T}_{\mathrm{wc}}^2$

Figure 4.24: Shift transformations for the two-cylinder test case here visualized for the combined vorticity field $\omega^1 + \omega^{\mathrm{ch}}$. The black dashed line indicates the symmetry axis at $L/2$. The blue solid line is the x, t dependent shift.

fluctuations in the wake pattern due to the sinusoidal movement of the second cylinder. The first three modes of the chaser cylinder in its

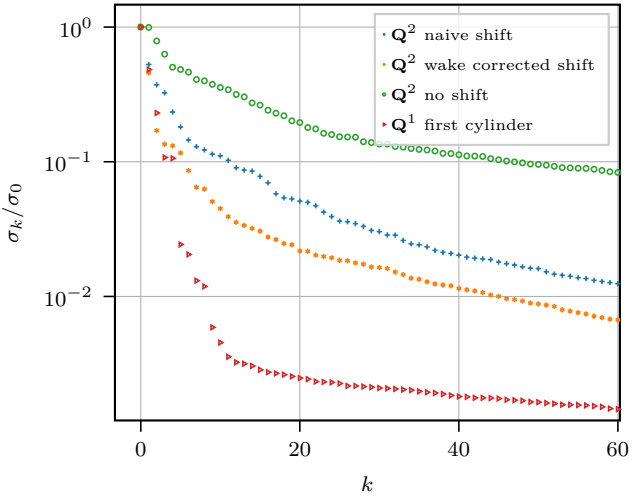


Figure 4.25: Comparison of the singular values of the pressure snapshot matrix for the decoupled system using the wake corrected shift (Eq. (4.49)), naive shift (Eq. (4.48)) and no shift for the second cylinder (chaser). Compare also to Fig. 4.24.

co-moving reference frame are shown in the top row of Fig. 4.31.

To test our algorithm, we add up the separately simulated cylinders $\boldsymbol{q}^1 = (u_1^1, u_2^1, p^1)$ and $\boldsymbol{q}^{\mathrm{ch}} = (u_1^{\mathrm{ch}}, u_2^{\mathrm{ch}}, p^{\mathrm{ch}})$ in the following way:

$$\boldsymbol{q}^{\mathrm{dc}} = \begin{pmatrix} \frac{u_1^1 + u_2^{\mathrm{ch}}}{2} \\ u_2^1 + u_2^{\mathrm{ch}} \\ p_2^1 + p^{\mathrm{ch}} \end{pmatrix} \tag{4.50}$$

and try to decompose the decoupled system, such that

$$\boldsymbol{q}^{\mathrm{dc}} \approx \tilde{\boldsymbol{q}}^{\mathrm{dc}} = \mathcal{T}^1(\boldsymbol{q}^{\mathrm{dc},1}) + \mathcal{T}^2(\boldsymbol{q}^{\mathrm{dc},2}) \tag{4.51}$$

with the help of two algorithms: sPOD-$\mathcal{J}_1$ algorithm (Algorithm 7) and sPOD-$\mathcal{J}_2$ algorithm (Algorithm 5) with 20 total variation steps. Here, we use the shift transformations $\mathcal{T}^1$ (no shift) and $\mathcal{T}^2 = \mathcal{T}^2_{\mathrm{wc}}$

as stated in Eq. (4.49). The algorithms are applied to the individual components of the state vector.

To compare the different methods we use the following strategy:
First, we run the sPOD-$\mathcal{J}_1$ algorithm (Algorithm 7) until the relative error gets below the interpolation error $\mathcal{E}_*$ defined in Eq. (4.30) or if the value of the residual does not change over four consecutive iterations by more then $\texttt{Rtol} = 10^{-3}$ of its current value (stopping criteria defined in Eq. (4.29)). The relative interpolation errors and tuning parameters of Algorithm 7 are stated for the individual components in Table 4.2. For computational convenience, we choose the stiffness of all quantities on the lower limit of the interval $10^{-4} \leq \eta/\eta_0 \leq 1$ suggested in Section 4.3.5. The sPOD-$\mathcal{J}_1$ algorithm calculates the two co-moving reference fields $\{\boldsymbol{q}^{\mathrm{dc},k}\}_{k=1,2}$ with their corresponding truncation rank $\{r_k^*\}_{k=1,2}$. Second, we truncate the $\{\boldsymbol{q}^{\mathrm{dc},k}\}_{k=1,2}$ for all possible rank combinations $(r_1, r_2) \in \{1, \ldots, r_1^*\} \times \{1, \ldots, r_2^*\}$ and select the pairs (r_1, r_2) for which the ROM with $r = r_1 + r_2$ degrees of freedom (DOF) has the smallest truncation error. In the third step, we run the sPOD-$\mathcal{J}_2$ algorithm (Algorithm 5) with 20 total variation steps on the exact same pairs (r_1, r_2) determined from the previous step. To compare sPOD-$\mathcal{J}_2$ with sPOD-$\mathcal{J}_1$, we use the same stopping criterion as for the sPOD-$\mathcal{J}_1$ algorithm.

The relative approximation error of the decomposition is stated for all three state vector components in the left column of Fig. 4.26. Note that since $\mathcal{T}^1$ is the identity, any pair (r_1, r_2) with $r_2 = 0$ falls back to the POD. Therefore, the smallest approximation errors of the sPOD for any r can never be higher than the ones of the POD using r number of modes.

As shown in Fig. 4.26, the approximation errors are similar for both algorithms. Here it should be pointed out that in contrast to the sPOD-$\mathcal{J}_1$ algorithm, the sPOD-$\mathcal{J}_2$ algorithm requires a separate run of the algorithm for every data point shown in Fig. 4.26. This is because Algorithm 5 optimizes $\boldsymbol{q}^{\mathrm{dc},1}, \boldsymbol{q}^{\mathrm{dc},2}$ for a fixed rank. However, this does not imply that the optimized co-moving fields have a fast singular value decay. This fact is investigated in Figs. 4.27 and 4.28. Here, Figs. 4.27 and 4.28 compare the exact singular values of $\boldsymbol{q}^1, \boldsymbol{q}^2$ (marker $+$) with the ones $\boldsymbol{q}^{\mathrm{dc},1}, \boldsymbol{q}^{\mathrm{dc},2}$ (marker $\circ$) determined with Algorithms 5 and 7.

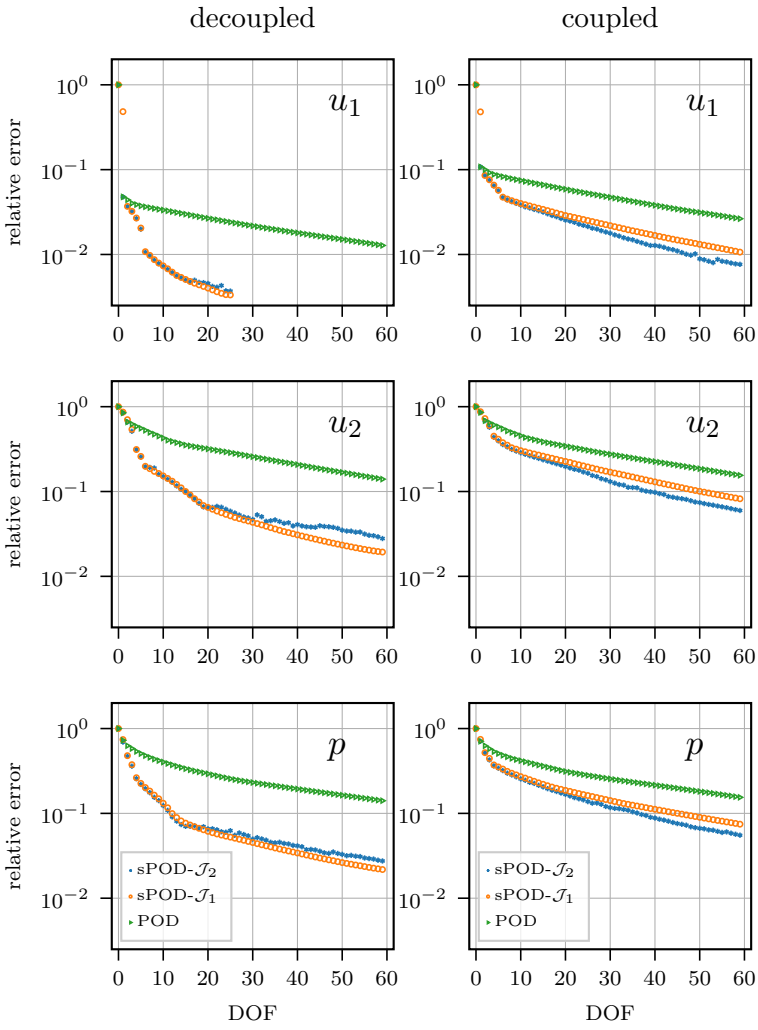


Figure 4.26: Relative reconstruction error in the Frobenius norm vs. the number of degrees of freedom (DOF) for the decoupled and coupled system. The DOF are determined from the truncation rank of the POD and as the sum of the co-moving ranks DOF $= r_1 + r_2$ in the case of the sPOD-$\mathcal{J}_2$ Algorithm 5 and sPOD-$\mathcal{J}_1$ Algorithm 7.

Although both algorithms cannot exactly recover the singular value spectra in each frame, the sPOD-$\mathcal{J}_1$ gives much better results than the sPOD-$\mathcal{J}_2$ algorithm.

Coupled System

In the coupled system, the flow pattern is a combination of the free-stream flow (labeled by $\boldsymbol{q}^{\text{fs}}$) that can be determined from the decoupled system and interactions (labeled by $\boldsymbol{q}^{\text{i}}$) when the chaser hits the wake of the leading cylinder:

$$\boldsymbol{q} = \boldsymbol{q}^{\text{fs}} + \boldsymbol{q}^{\text{i}} \tag{4.52}$$

Therefore it is reasonable that the singular value decay in the co-moving systems and the overall approximation error will decay slower than expected from the decoupled system. This is confirmed by the comparison of the approximation error for the coupled and decoupled system in Fig. 4.26 and the singular values shown in Figs. 4.27 and 4.28. For the coupled system we apply the same strategy to compare both algorithms as outlined for the decoupled system. All relevant parameters are listed in Table 4.3.

	u_1	u_2	p
interp. err. wake corr.	3.1e-04	1.6e-03	5.1e-03
stiffness parameter η/η_0	5.0e-03	4.0e-04	1.0e-03

Table 4.3: Relative interpolation error $\mathcal{E}_*$ defined in Eq. (4.30) and stiffness parameter relative to $\eta_0 = MN_t/(4\,\|\mathbf{Q}\|_1)$ of Algorithm 7 listed for the different flow quantities of the coupled system.

Furthermore, we visualize the separation of the two cylinders using sPOD-$\mathcal{J}_1$ and sPOD-$\mathcal{J}_2$ in Fig. 4.28 for the vorticity field. The sPOD-$\mathcal{J}_2$ is not able to separate the two cylinders if for the chosen co-moving ranks the algorithm decides to combine structures from both cylinders. In contrast, the sPOD-$\mathcal{J}_1$ requires a rapid decay of the singular values in each frame, which is only achieved when the cylinders are separated in the frames.

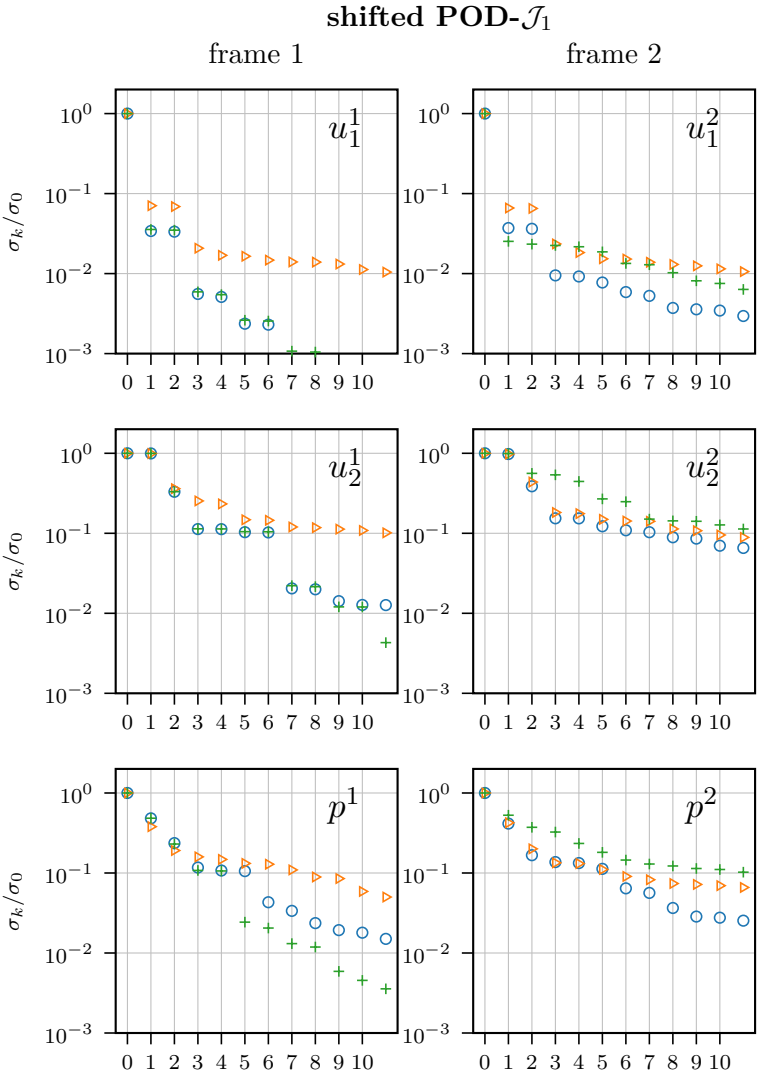


Figure 4.27: Singular values of u_1^f, u_2^f, p^f (top to bottom) in the wake-corrected co-moving reference frames $f = 1, 2$ (left to right). Shown are the singular values of the separate system (+) in the shifted frame, the decoupled system where the individual components have been added (○) and the coupled system (◁). The singular values corresponding to ○ and ◁ have been computed with the sPOD-$\mathcal{J}_1$ Algorithm 7. Compare to sPOD-$\mathcal{J}_2$ in Fig. 4.28.

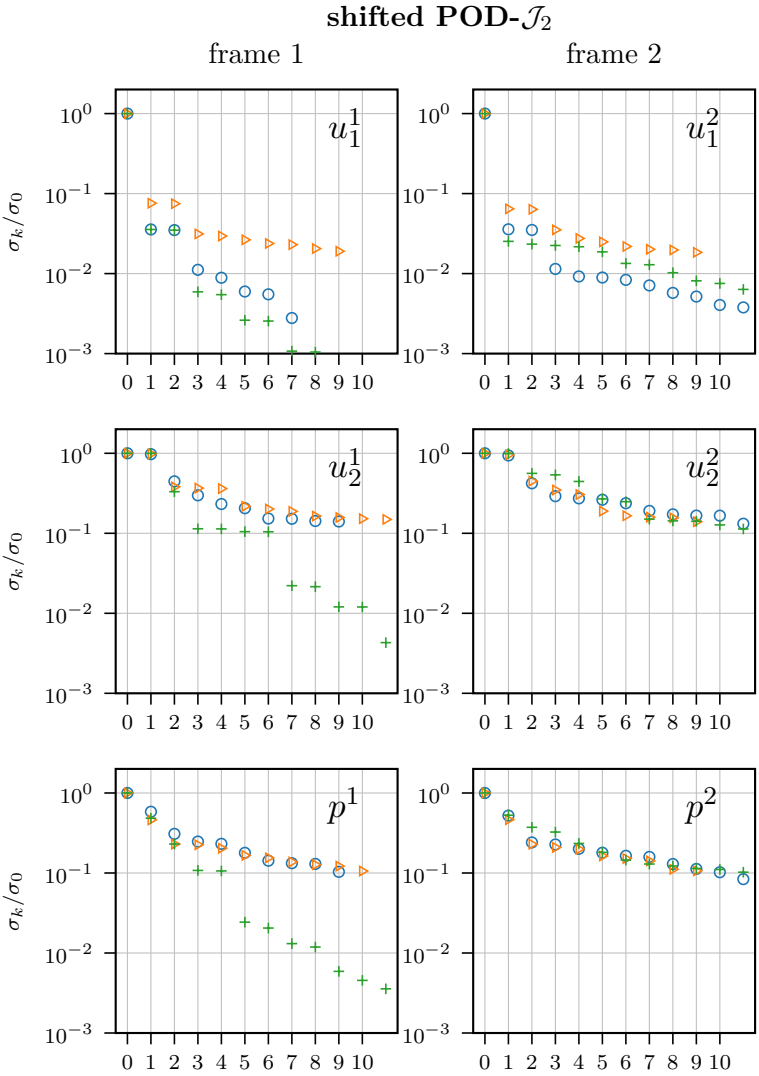


Figure 4.28: Singular values of u_1^f, u_2^f, p^f (top to bottom) in the wake-corrected co-moving reference frames $f = 1, 2$ (left to right). Shown are the singular values of the separate system (+) in the shifted frame, the decoupled system where the individual components have been added (○) and the coupled system (◁). The singular values corresponding to ○ and ◁ have been computed with the sPOD-$\mathcal{J}_2$ Algorithm 5. Compare to sPOD-$\mathcal{J}_1$ in Fig. 4.27.

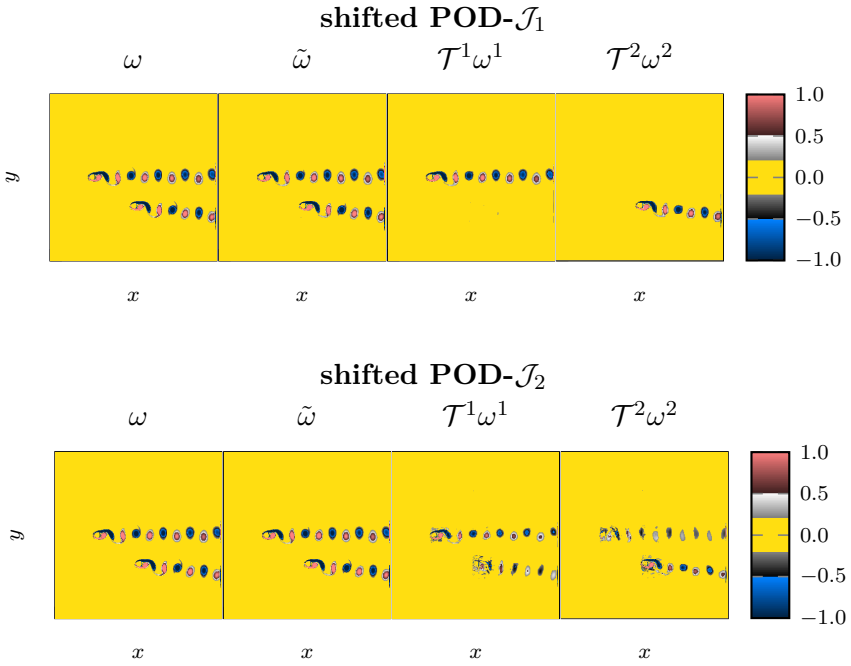


Figure 4.29: Separation of the two moving cylinders. Shown is the vorticity field $\omega = \partial_x u_2 - \partial_y u_1$ calculated from the coupled cylinder pair and the reconstructed vorticity field $\tilde{\omega} = \omega^1 + \omega^2$, with $\omega^i = \partial_x T^i u_2^i - \partial_y T^i u_1^i$, $i = 1, 2$. Here u_1^i, u_2^i have been calculated with Algorithm 7 (shifted POD-$\mathcal{J}_1$, first row) and Algorithm 5 (shifted POD, second row). The co-moving ranks that are estimated by the shifted POD-$\mathcal{J}_1$ and used as an input in the shifted POD-$\mathcal{J}_2$ algorithm are $(r_1, r_2) = (50, 53)$ for (u_1^1, u_1^2) and $(r_1, r_2) = (95, 96)$ for (u_2^1, u_2^2).

Since Algorithm 7 allows us to separate both physical structures nicely into the two co-moving frames:

$$\boldsymbol{q} = \mathcal{T}^1(\boldsymbol{q}^1) + \mathcal{T}^2(\boldsymbol{q}^2), \tag{4.53}$$

we can go one step further in studying the interacting and free-stream-flow of the second cylinder (chaser) in its co-moving frame:

$$\boldsymbol{q}^2 = \boldsymbol{q}^{\mathrm{i},2} + \boldsymbol{q}^{\mathrm{fs},2}\,. \tag{4.54}$$

To identify the structures that correspond to the free-stream flow, we can project $\boldsymbol{q}^2$ onto the POD modes of the chasing cylinder in the decoupled system, labeled by $\boldsymbol{\Psi}^{\mathrm{dc}} = [\boldsymbol{\psi}_1^{\mathrm{dc},2}(x,y), \ldots, \boldsymbol{\psi}_{N_t}^{\mathrm{dc},2}(x,y)]$ in the following:

$$\boldsymbol{q}^{\mathrm{fs},2} := \sum_{k=1}^{N_t} \langle \boldsymbol{\psi}_k^{\mathrm{dc},2}, \boldsymbol{q}^2 \rangle \boldsymbol{\psi}_k^{\mathrm{dc},2}\,. \tag{4.55}$$

After projecting $\boldsymbol{q}^2$ onto its free-stream part, we can compute the structures responsible for the interaction:

$$\boldsymbol{q}^{\mathrm{i},2} = \boldsymbol{q}^2 - \boldsymbol{q}^{\mathrm{fs},2}\,. \tag{4.56}$$

The first three modes that correspond to

$$\boldsymbol{q}^{\mathrm{i},2}(x,y,t) \approx \boldsymbol{q}_l^{\mathrm{i},2}(x,y,t) := \sum_{n=1}^{l} \alpha_n^{\mathrm{i},2}(t)\ \boldsymbol{\psi}_n^{\mathrm{i},2}(x,y) \tag{4.57}$$

$$\boldsymbol{q}^{\mathrm{fs},2}(x,y,t) \approx \boldsymbol{q}_k^{\mathrm{fs},2}(x,y,t) := \sum_{n=1}^{k} \alpha_n^{\mathrm{fs},2}(t) \boldsymbol{\psi}_n^{\mathrm{fs},2}(x,y) \tag{4.58}$$

are visualized as vorticity in Fig. 4.31 together with the first three modes of the decoupled system. Furthermore, we visualize the energy contribution to the overall co-moving field $\boldsymbol{q}^2$

$$\mathcal{E}^{\mathrm{tot}}(k,l) := \frac{\left\|\boldsymbol{q}_l^{\mathrm{i},2} + \boldsymbol{q}_k^{\mathrm{fs},2}\right\|^2}{\|\boldsymbol{q}^2\|^2} \leq \underbrace{\frac{\left\|\boldsymbol{q}_l^{\mathrm{i},2}\right\|^2}{\|\boldsymbol{q}^2\|^2}}_{:=\mathcal{E}^{\mathrm{i}}(l)} + \underbrace{\frac{\left\|\boldsymbol{q}_k^{\mathrm{fs},2}\right\|^2}{\|\boldsymbol{q}^2\|^2}}_{:=\mathcal{E}^{\mathrm{fs}}(k)} \tag{4.59}$$

in Fig. 4.30. Figure 4.30 indicates that in the first three free-stream and interaction modes more than 99 percent of the energy is contained. However, the contribution of the interaction is very small as the orange portion in Fig. 4.30 indicates.

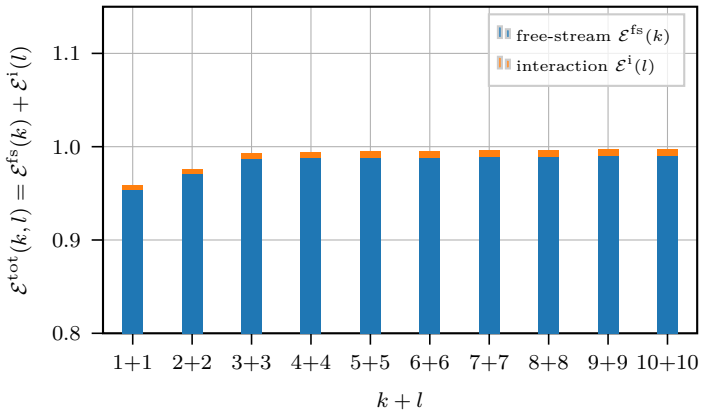


Figure 4.30: Energy contribution of the interacting and free-stream modes defined in Eq. (4.59).

4.4 Summary

Non-Linear reduction methods can help to improve the approximation of transport-dominated fluid systems (TDFS), when trying to reduce their dimensionality. However, the presented methods, namely:

- neural autoencoder networks (AE)
- front transport reduction (FTR)
- shifted proper orthogonal decomposition (sPOD)

have limitations and specific application regimes. Based on these regimes, we propose the strategy outlined in Fig. 4.32. Starting from the snapshot matrix $\mathbf{Q} = [\underline{\boldsymbol{q}}(t_1), \ldots, \underline{\boldsymbol{q}}(t_{N_t})]$ that collects samples of the FOM ODE-state $\underline{\boldsymbol{q}} \in \mathbb{R}^M$, the aim is to find a reduced mapping that can reproduce the samples on a lower dimensional manifold. In most cases, a linear subspace created by the proper orthogonal decomposition (POD) is sufficiently accurate for building a reduced system. As suggested in the flow chart Fig. 4.32, the overall quality of this approximation can be estimated by the decay of the singular values, i.e. the square root of the POD eigenvalues of $\mathbf{Q}$ (see Eq. (3.17)). However, for TDFS the singular values decay slowly and a large linear subspace might be required. Under these circumstances the proposed non-linear methods

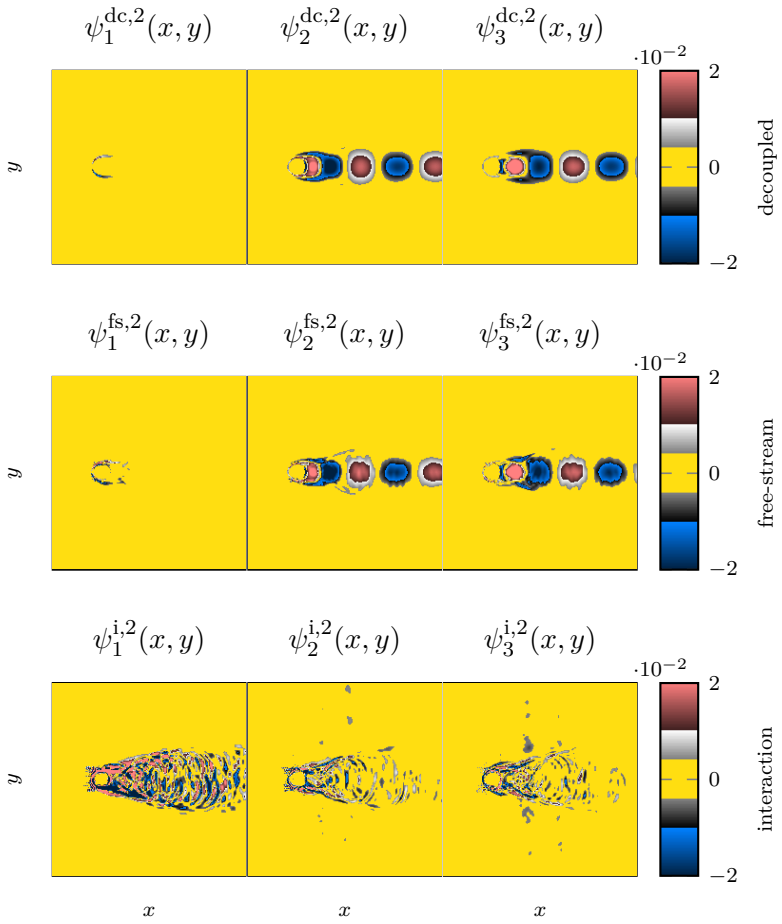


Figure 4.31: First three decoupled (labeled with dc), free-stream (labeled with fs), and interaction (labeled with i) modes of the chaser cylinder in its co-moving reference frame.

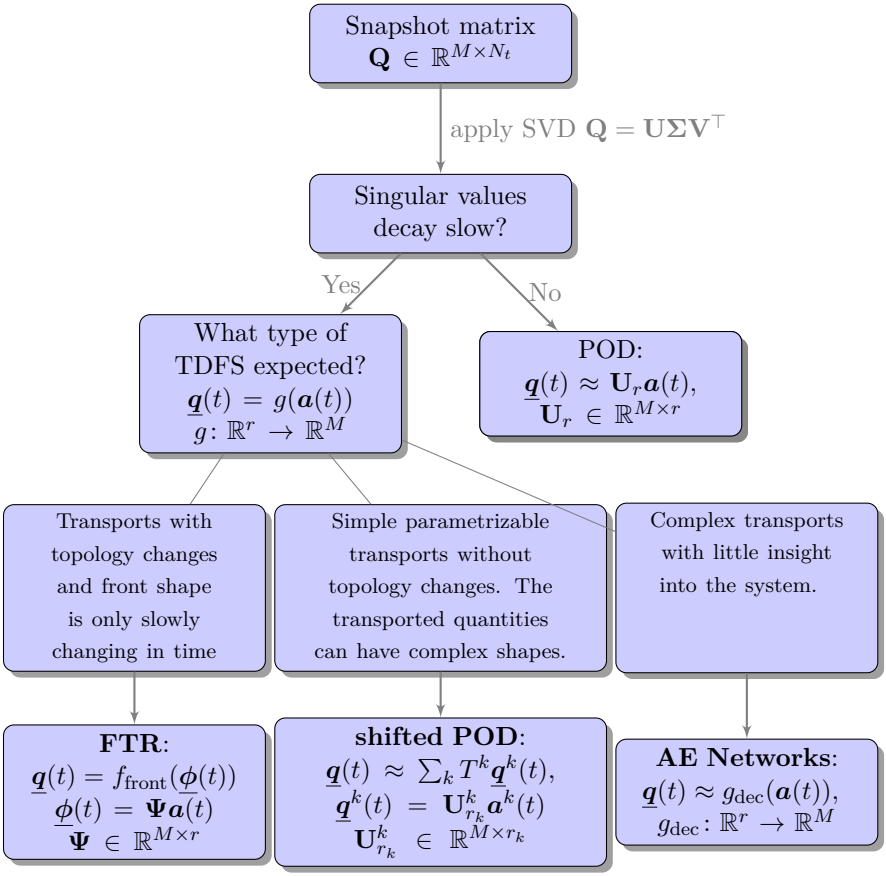


Figure 4.32: Flow chart of the basic strategy for TDFS dimension reduction.

might be beneficial.

If the system is complex and allows no, or little, physical insight, *autoencoder* (AE) networks can be used, as their expressivity enables us to identify low dynamical structures without additional guidance or assumptions on the complexity of the flow system. Unfortunately, AE allows only limited interpretability and is often difficult to set up for the specific case at hand. Therefore, this thesis proposes two non-linear reduction methods that provide better insight into the system. However, the additional insight comes at the cost of general applicability:

If the transport of the flow quantity is simply given by a shift or simple parametrizable transformation, the *shifted proper orthogonal decomposition* (sPOD) is advantageous, since it does not require any assumptions regarding the shape of the moving quantity. The sPOD was already introduced in [115] as a heuristic algorithm, but has been reformulated in [114]. It shifts the data field in a so-called co-moving reference frame, in which the moving structure can be decomposed efficiently with the POD. The contribution of this thesis is based on the optimization goals given in [114]. The presented algorithms enable the application of the sPOD to high dimensional data, since they avoid gradient-based optimization, as done in [114]. For example, the shifted robust principal component analysis (srPCA) (Algorithm 8) or the total variation diminishing sPOD algorithm (Algorithm 5) use non-smooth high dimensional optimization techniques. As the sPOD allows us to split physical systems that are not interacting, it can be used to decouple and study flows around moving objects, as presented in Section 4.3.6.

The *front transport reduction* (FTR) is a method to decompose complex moving fronts. The decomposition parametrizes moving fronts with the help of a transport-dependent auxiliary field $\underline{\boldsymbol{\phi}} \in \mathbb{R}^M$ and a front function f to approximate the front profile. Three different decomposition algorithms have been proposed, which construct $\underline{\boldsymbol{\phi}}$ using signed distance function Section 4.2.2, singular value thresholding (Algorithm 2) or artificial neural networks (Section 4.2.4). The resulting approximations $\underline{\boldsymbol{q}}(t,\mu) \approx f(\underline{\boldsymbol{\phi}}(t,\mu))$ are well suited for model order reduction of reacting fronts, since $\underline{\boldsymbol{\phi}}(t,\mu) = \boldsymbol{\Psi}\boldsymbol{a}(t,\mu) \in \mathbb{R}^M$ can be represented by a few $r \ll M$ spatial modes collected in $\boldsymbol{\Psi} \in \mathbb{R}^{M\times r}$. We emphasize that the utilized front-structure is inherent for advection-reaction-diffusion (ARD) systems (see for example [66, 5, 6, 49]). Regarding AE networks, the FTR is similar in the sense that it uses a linear layer activated by a problem-dependent non-linear front function as a decoder. Other studies [81, 89] use multiple non-linear activated layers, resulting in costly evaluations of the networks themselves. This can limit the overall performance of the ROM when evaluating the additional non-linearities. Similar to AE networks, the FTR can approximate topological changes in the evolution of the contour line of the front, since it does not make explicit assumptions about the mapping. Here methods like the shifted

POD, previously applied to similar problems in [11], cannot be used, since they assume one-to-one mappings to align the front. In order to parametrize complex transports with topological changes, a combination of the sPOD and FTR would be desirable, as this would enable the decomposition of multiple traveling wave systems with topological changes. Furthermore, it would be interesting to see whether the FTR approach can be applied to multi-phase flows, as they inherit a similar front structure separating the fluids.

5 Dynamic ROM - Predictions of Transport Dominated Systems

In the previous sections, we have addressed the so-called *offline stage* of a model order reduction procedure, in which data is collected and its dimension is reduced. The reduced models generated by the neural autoencoder network, sPOD or FTR algorithm are non-linear, which poses additional challenges for the *online stage*, to predict and interpolate new system states. This chapter is therefore dedicated to online prediction methods. In Section 5.1 we use a non-intrusive, i.e. equation-free, approach of [87] and in Section 5.2 we introduce an intrusive approach, the hyper-reduced Galerkin method for 1D and 2D ARD systems. Since non-linear Galerkin methods have been studied for the shifted POD in [10, 11] and for neural networks in [89], we restrict our studies to the FTR in this chapter. The chapter closely follows the work presented in [147].

5.1 Data-Driven Methods

With the rise of data-driven methods in model order reduction, non-intrusive prediction methods of the reduced system, e.g. POD-DL-ROM [52], SINDy [54, 111] or Fourier-Koopman forecasting [87], have become prominent. Although the methods make specific assumptions about the system at hand, they can be useful, since they allow rapid evaluation of the reduced variables with good accuracy. This is especially beneficial if the reduced map parametrizes a non-linear manifold, which makes any Galerkin-projection approach more complex and

costly, as is shown in the next section.

Following the approach of [87], we can derive new system states and extrapolate in time with help of the Fourier-Koopman framework implemented in [86]. Under the assumption that the reduced state $\boldsymbol{a}(t) \in \mathbb{R}^r$ is quasi-periodic in t, it can be parametrized by:

$$\boldsymbol{a}(t) = \mathbf{A}\boldsymbol{\Omega}(t) \quad \text{with} \quad \boldsymbol{\Omega}(t) = \begin{pmatrix} \cos(\boldsymbol{\omega} t) \\ \sin(\boldsymbol{\omega} t) \end{pmatrix} . \tag{5.1}$$

Here, $\mathbf{A} \in \mathbb{R}^{r \times p}$ and $\boldsymbol{\omega} \in \mathbb{R}^{p/2}$ are determined by solving the optimization problem:

$$\min_{\boldsymbol{\omega}, \mathbf{A}} \sum_{n=0}^{N-1} \| \boldsymbol{a}(t_n) - \mathbf{A}\boldsymbol{\Omega}(t_n) \|_2^2 , \tag{5.2}$$

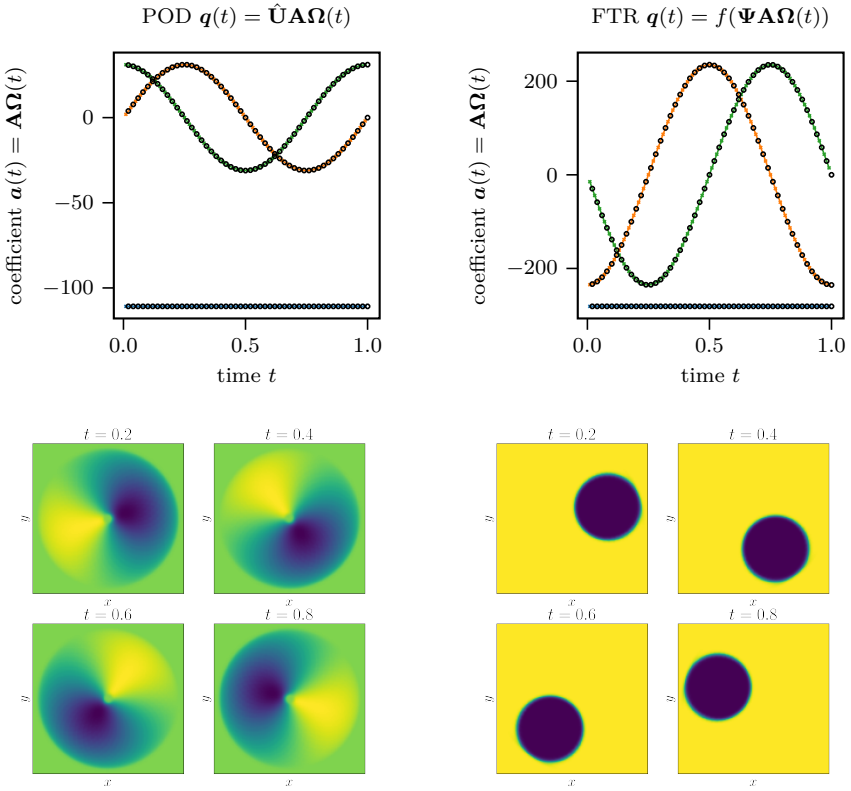
in an efficient way [87]. Since the dynamical system presented in Section 4.2.5 is quasi-periodic, we can apply the method to the FTR decomposition

$$\underline{\boldsymbol{q}}(t) \approx \underline{\tilde{\boldsymbol{q}}}(t) = f(\boldsymbol{\Psi}\boldsymbol{a}(t)) \tag{5.3}$$

using the basis functions $\boldsymbol{\Psi} = [\tilde{\boldsymbol{\psi}}_1, \tilde{\boldsymbol{\psi}}_2, \tilde{\boldsymbol{\psi}}_3]$, shown in Fig. 4.8 together with the amplitudes $\boldsymbol{a}(t) = (a_1(t), a_2(t), a_3(t))$ at the sampled time points $\{t_n = n\Delta t \mid n = 0, \ldots, N-1\}$. From the sampled data we compute $\mathbf{A}, \boldsymbol{\omega}$. The resulting model $\tilde{\boldsymbol{q}}(t) = f(\boldsymbol{\Psi}\mathbf{A}\boldsymbol{\Omega}(t))$ is evaluated at $t_{n+1/2} = (n+1/2)\Delta t$ for $n = 0, \ldots, 2N-1$. Similarly, we can derive an approximation with the POD. Both results are compared in Fig. 5.1. Furthermore, the online-prediction error is stated for $r = 2, 4, \ldots, 12, 15$ in Table 5.1.

Remark. *The systems dynamics can be further reduced by rewriting* $f(\boldsymbol{\Psi}\boldsymbol{a}(t)) = f(\tilde{\boldsymbol{\Psi}}\tilde{\boldsymbol{a}}(t) + \boldsymbol{b}), \boldsymbol{b} \in \mathbb{R}^M, \tilde{\boldsymbol{a}} \in \mathbb{R}^{r-1}, \tilde{\boldsymbol{\Psi}} \in \mathbb{R}^{M \times (r-1)}$. *The offset vector* $\boldsymbol{b}$ *then contains the time-independent part of the decomposition shown as a constant line in Fig. 5.1. This can be done similarly for the POD.*

Note that after solving Eq. (5.2) in the offline stage, the computational effort is reduced to the evaluation of $\underline{\tilde{\boldsymbol{q}}}(t) = f(\boldsymbol{\Psi}\mathbf{A}\boldsymbol{\Omega}(t))$, which only takes milliseconds. A similar approach has been used in [146] that time-forecasts system states using the shifted POD-$\mathcal{J}_2$ algorithm Algorithm 3 in combination with deep forward networks. Instead of



(a) POD with $r = 3$ **(b)** FTR with $r = 3$

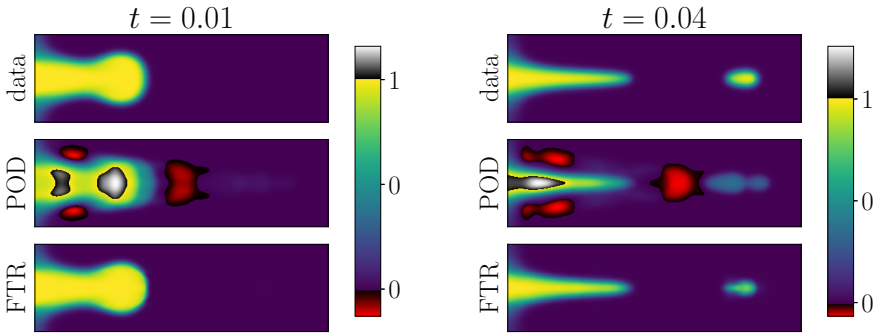
Figure 5.1: Predictions using Fourier-Koopman forecasting with three POD modes (a) and three FTR modes (b). The black circles $(\cdot)$ in the upper row indicate the predictions of the amplitudes $\boldsymbol{a}(t) = (a_1(t), a_2(t), a_3(t)) \widehat{=}$ (—,—,—) and the colored crosses mark the training samples. In the lower row, we show the corresponding snapshots at selected time instances $t = 0.2, 0.4, 0.6, 0.8$.

the system Eq. (5.2) the authors use feedforward neural networks to circumvent the restriction to quasi-periodic systems.

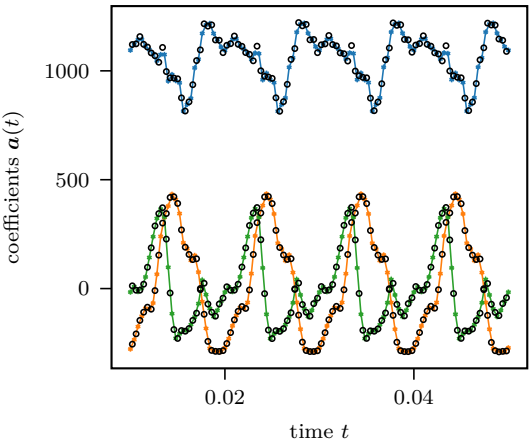
For a realistic test case, we apply the FTR-Fourier-Koopman procedure to the methane mass fraction Y_{CH_4} of one flame of a multi-slit Bunsen burner simulation analyzed and studied in [84, 77]. The snapshots are generated with a customized, weakly compressible version of `rhoReactionFOAM` from the `OpenFOAM` software package (see [139, 77]). In the simulation, a flame is periodically excited by an incoming velocity pulse. The acceleration of the fuel detaches a burning pocket shown in Fig. 5.2. The data set consists of 200 snapshots, with $M = 128 \times 430$ grid points, sampled in a time interval $t \in [0.01, 0.05]$ in which the Bunsen flame is quasi-periodic. Again, we split the data into training ($t_n = 2\Delta t n$) and test samples $t_{n+1/2} = (2n+1)\Delta t$. While we use the training samples to generate the reduced model, the test samples are used to calculate the relative errors stated in Table 5.1. The flame pinch-off is not a special case in combustion systems, but it poses challenges to model order reduction methods, as described above. Figure 5.2 shows that for the FTR the structure of the solution is well captured and the physical bound $0 \leq Y_{\mathrm{CH}_4} \leq 1$ is preserved.

	Moving Disc		Bunsen Flame	
rank r	FTR	POD	FTR	POD
2	2.7e-01	3.0e-01	4.2e-01	3.1e-01
4	7.4e-03	2.0e-01	1.4e-01	3.1e-01
6	2.2e-03	1.5e-01	1.1e-01	2.3e-01
8	1.6e-03	1.2e-01	7.6e-02	1.8e-01
10	2.2e-03	1.0e-01	8.1e-02	1.6e-01
12	2.0e-03	8.8e-02	7.1e-02	1.5e-01
15	1.2e-03	7.4e-02	6.9e-02	1.4e-01

Table 5.1: Relative error $\sum_{n=0}^{2N-1} \|\underline{\boldsymbol{q}}(t_{n+1/2}) - \tilde{\underline{\boldsymbol{q}}}(t_{n+1/2})\|_2^2 / \sum_{n=0}^{2N-1} \|\underline{\boldsymbol{q}}(t_{n+1/2})\|_2^2$ for the FTR-Fourier-Koopman predictions using the moving disk and Bunsen flame data.



(a) Test snapshots



(b) FTR-Koopman predictions

Figure 5.2: Online predictions of the Bunsen flame example. Fig. a) compares the test data in the top row with the FTR-Koopman and POD-Koopman results using $r = 8$ degrees of freedom for $t = 0.01$ and 0.04. The snapshots show how a burning fuel pocket is detached from the flame at $t = 0.04$ causing a change in the topology of the contour line of the front. Fig. b) visualizes the Fourier-Koopman predictions $(\cdot)$ for $\boldsymbol{a}(t) = (a_1(t), a_2(t), a_3(t)) \widehat{=} (-, -, -)$.

5.2 Manifold Galerkin Methods

After discretizing the ARD system Eq. (4.1) in space, we obtain an ODE system of the form

$$\text{(FOM)} \quad \begin{cases} \dot{\underline{\boldsymbol{q}}}(t,\boldsymbol{\mu}) = \boldsymbol{N}(\underline{\boldsymbol{q}},t,\boldsymbol{\mu}) \\ \underline{\boldsymbol{q}}(0,\boldsymbol{\mu}) = \underline{\boldsymbol{q}}_0(\boldsymbol{\mu})\,. \end{cases} \tag{5.4}$$

Here, the parameters $\boldsymbol{\mu} \in \mathcal{P}$ contain diffusion or reaction constants κ, γ. After discretizing the re-scaled system it yields the FOM-RHS

$$\boldsymbol{N}(\underline{\boldsymbol{q}},t,\boldsymbol{\mu}) = \mathbf{L}(t)\underline{\boldsymbol{q}} + \boldsymbol{F}(\underline{\boldsymbol{q}},\boldsymbol{\mu}) \tag{5.5}$$

with a linear operator $\mathbf{L}\colon [0,T] \to \mathbb{R}^{M\times M}$ and a non-linear operator $\boldsymbol{F}\colon \mathbb{R}^M \times \mathcal{P} \to \mathbb{R}^M$. Using a reduced mapping

$$g\colon \mathbb{R}^r \to \mathbb{R}^M : \boldsymbol{a} \mapsto g(\boldsymbol{a}), \quad \text{with Jacobian} \quad \mathbf{J}_g(\boldsymbol{a}) = \left[\frac{\partial g_i}{\partial a_j}(\boldsymbol{a})\right]_{\substack{i=1,\dots,M \\ j=1,\dots,r}} \tag{5.6}$$

as approximation $\underline{\boldsymbol{q}} \approx \tilde{\underline{\boldsymbol{q}}} = g(\boldsymbol{a})$ of the data and plugging it into Eq. (5.4) yields a reduced model:

$$\text{(ROM)} \quad \begin{cases} \dot{\boldsymbol{a}}(t,\boldsymbol{\mu}) = \operatorname*{arg\,min}_{\dot{\boldsymbol{a}}\in\mathbb{R}^r} \|\mathbf{J}_g(\boldsymbol{a})\dot{\boldsymbol{a}}(t,\boldsymbol{\mu}) - \boldsymbol{N}(g(\boldsymbol{a}),t,\boldsymbol{\mu})\|_2^2 & (5.7) \\ \boldsymbol{a}(0,\boldsymbol{\mu}) = \operatorname*{arg\,min}_{\boldsymbol{a}\in\mathbb{R}^r} \|\underline{\boldsymbol{q}}_0(\boldsymbol{\mu}) - g(\boldsymbol{a})\|_2^2\,. & (5.8) \end{cases}$$

Minimizing the continuous time residual Eq. (5.7), yields the optimality condition:

$$\begin{aligned} 0 &= \frac{\mathrm{d}}{\mathrm{d}\dot{\boldsymbol{a}}} \|\mathbf{J}_g(\boldsymbol{a})\dot{\boldsymbol{a}} - \boldsymbol{N}(g(\boldsymbol{a}),t,\boldsymbol{\mu})\|_2^2 && (5.9) \\ &= 2\mathbf{J}_g(\boldsymbol{a})^\top \mathbf{J}_g(\boldsymbol{a})\dot{\boldsymbol{a}} - 2\mathbf{J}_g(\boldsymbol{a})^\top \boldsymbol{N}(g(\boldsymbol{a}),t,\boldsymbol{\mu})\,, && (5.10) \end{aligned}$$

which is uniquely solved by

$$\dot{\boldsymbol{a}} = \mathbf{J}_g^+(\boldsymbol{a})\boldsymbol{N}(g(\boldsymbol{a}),t,\boldsymbol{\mu})\,, \tag{5.11}$$

if the Jacobin has full column rank [89]. Here, $\mathbf{J}_g^+$ is the Moore-Penrose pseudo inverse of $\mathbf{J}_g$. Note, that for the common POD-Galerkin approach orthogonal mappings $g(\boldsymbol{a}) = \mathbf{U}_r\boldsymbol{a}$, with $\mathbf{U}_r \in \mathbb{R}^{M\times r}$ and $\mathbf{U}_r^\top \mathbf{U}_r =$

I are used. Therefore, Eq. (5.11) and Eq. (5.10) are identical. When neglecting $\boldsymbol{F}(\underline{\boldsymbol{q}}, \boldsymbol{\mu})$ in Eq. (5.5) for the time being, we obtain a small r-dimensional system:

$$\dot{\boldsymbol{a}} = \mathbf{L}_r(t)\boldsymbol{a} \quad \text{with} \quad \mathbf{L}_r(t) = \mathbf{U}_r^\top \mathbf{L}(t)\mathbf{U}_r \in \mathbb{R}^{r\times r}\,, \tag{5.12}$$

which can be solved efficiently, when $\mathbf{L}_r(t)$ is precomputed. For example in the case of pure advection:

$$\dot{q}(\boldsymbol{x}, t) = \boldsymbol{u} \cdot \boldsymbol{\nabla} q = \sum_{k=1}^{d} u_k(t)\partial_{x_k} q(\boldsymbol{x}, t)\,, \tag{5.13}$$

the spatial derivative has the form $\mathbf{L}(t) = \sum_{k=1}^{d} u_k(t)\mathbf{L}^{(k)}$ and therefore

$$\mathbf{L}_r(t) = \sum_{k=1}^{d} u_k(t)\mathbf{U}_r^\top \mathbf{L}^{(k)}\mathbf{U}_r \in \mathbb{R}^{r\times r} \tag{5.14}$$

can be precomputed and is much smaller than the operator $\mathbf{L}(t) \in \mathbb{R}^{M\times M}$, $r \ll M$ of the FOM. Although POD-Galerkin enables solving Eq. (5.12) efficiently, this approach cannot be used for advection dominated systems, because of its slow decaying approximation errors. Here, non-linear methods like artificial neural networks can accelerate the convergence of the overall online and offline error. However, any non-linear reduction method will imply that even linear systems like Eq. (5.13) become non-linear, causing additional effort for evaluating non-linearities. At least in the special case of an advection system, this can be avoided with the FTR approach. Due to its special structure $\tilde{q}(\boldsymbol{x}, t) = f(\phi(\boldsymbol{x}, t))$, we can rewrite the advection equation $\partial_t q - \boldsymbol{u} \cdot \boldsymbol{\nabla} q = 0$ into the form

$$f'(\phi)(\partial_t \phi - \boldsymbol{u} \cdot \boldsymbol{\nabla}\phi) = 0\,. \tag{5.15}$$

The prefactor $f'(\phi)$ can be dropped, when assuming that ϕ features the same transport then q and thus the non-linear manifold Galerkin system (5.11) can be simplified to a linear Galerkin system for $\underline{\boldsymbol{\phi}}(t) = \boldsymbol{\Psi}\boldsymbol{a}(t)$:

$$\dot{\boldsymbol{a}} = \boldsymbol{\Psi}\mathbf{L}(t)\boldsymbol{\Psi}\boldsymbol{a} \approx \mathbf{L}_r(t)\boldsymbol{a}\,. \tag{5.16}$$

Since the operator $\mathbf{L}_r$ can be precomputed in the same fashion as for the POD-Galerkin approach, the resulting ROM complexity is reduced

and the online/offline error is compensated due to the additional nonlinearity f to retain $\underline{\boldsymbol{q}} = f(\boldsymbol{\Psi a})$. However, these findings need to be interpreted with caution. One might expect that a pure transport of q implies a pure transport of ϕ. However, if f' becomes (approximately) zero, ϕ might locally change its value without changing q, so that q is transported everywhere, while ϕ is not. If, however, this cancellation is justified, it can speed up the calculation considerably. The advection of fronts according to a given transport field $\boldsymbol{u}(t)$ within milliseconds is impressively shown for a 1D advection example in Fig. 5.3. In this example, only two trajectories of constant advection speed ($u(t) = \pm 2$) are used for building the reduced system. Thereafter, almost any parametrization of $u(t)$ can be computed with the ROM.

The relative error and speedup for the test trajectory are shown in the lower part of Fig. 5.4 and are plotted together with the POD-Galerkin results. We see that the errors are reduced compared to the results of the POD. Further details about the simulation are reported in Appendix B.2. The results apply similarly to higher spatial dimension $D > 1$, for example in the case of the moving disk (Section 4.2.5). Note, that the path simulated in the online phase is limited to areas where the transport field is initialized. These are the areas where any front has traveled during the offline phase as shown in Fig. 4.8. This restriction is, however, shared with classical linear methods. The dynamics that are not covered in the ansatz space created from the initial set of snapshots are usually not covered by the ROM.

The success of the heuristic approach Eq. (5.16) is somewhat obvious since the level set function ϕ parametrizes the transport (see Section 4.2.5), which implies it to be a good basis for the advection operator. The idea to use transport capturing level set functions to accelerate simulations for advection laws is not new. For example it is intensively used by the characteristic mapping method (CMM) [99], which evolves the initial condition $q_0(\boldsymbol{x})$ of a PDE along characteristic curves $\boldsymbol{X}(\boldsymbol{x}, t)$, such that $q(\boldsymbol{x}, t) = q_0(\boldsymbol{X}(\boldsymbol{x}, t))$. Nevertheless, in [99] the authors use an invertible mapping $\boldsymbol{X}(\boldsymbol{x}, t)$, which hinders the applicability for systems with topological changes. Intentionally, this is not done here, since we aim for systems, where topological changes are possible. However, it would be interesting to see if snapshots of the characteristic map can

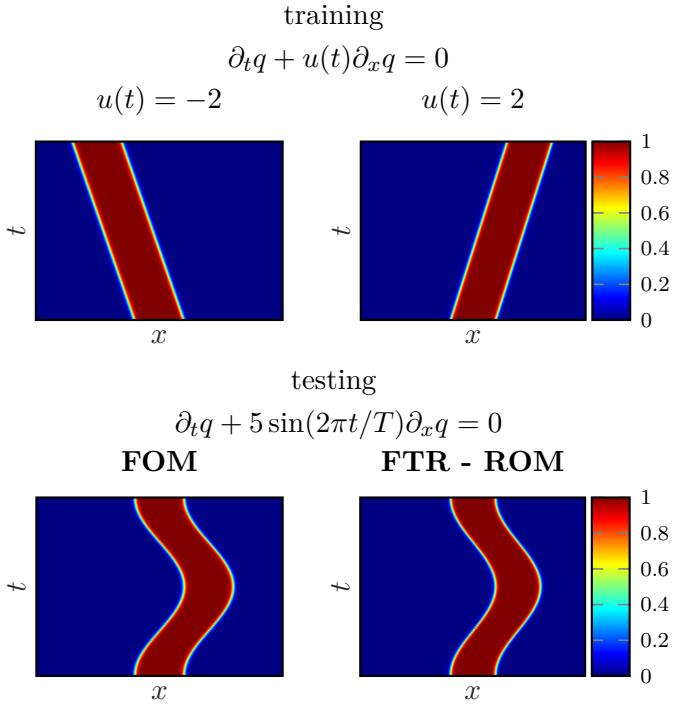
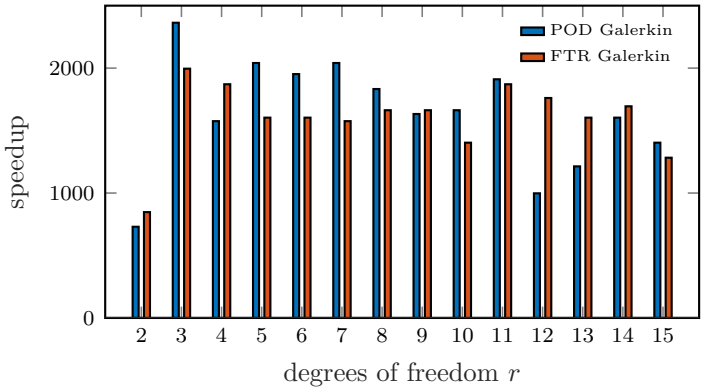
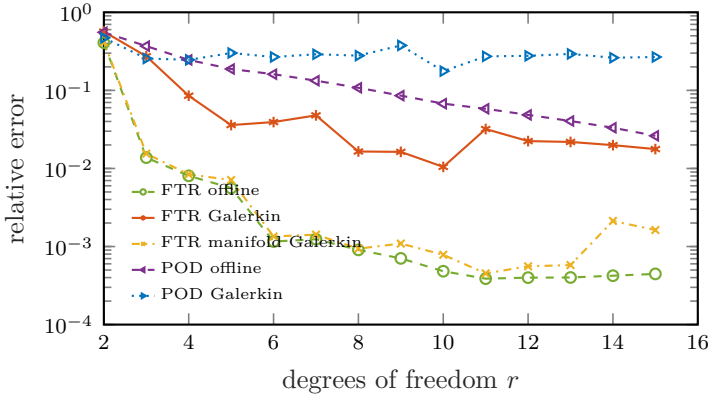


Figure 5.3: The 1D advection test case for moving fronts. In the upper row, the two training samples computed with the full order model (FOM) are shown. They are used to build the snapshot matrix $\mathbf{Q} \in \mathbb{R}^{1000\times 202}$ for the FTR decomposition. In the lower row, the trajectory of the FOM and the FTR-ROM Eq. (5.16) ($r = 4$) are compared.



(a) Speedup comparison



(b) Relative error in the Frobenius norm

Figure 5.4: Relative error (a) and speedup (b) for $r = 2, \ldots, 15$ using the FTR- and POD-Galerkin approach. The error and speedup are measured using the testing data shown in Fig. 5.3. The FTR/POD offline errors mark the reconstruction error of the training data, which is collected in the offline phase (snapshots are shown in the upper row of Fig. 5.3). All FTR Galerkin results are computed using Eq. (5.16) (snapshots are shown in the lower right of Fig. 5.3). Accordingly, the POD Galerkin results are computed using Eq. (5.12). The FTR manifold Galerkin results are computed using Eq. (5.11) directly.

be similarly utilized for MOR as the snapshots of the auxiliary field $\mathbf{\Phi}$ inside the FTR.

Nevertheless, it should be noted that the procedure proposed for the advection equation cannot be generalized to advection-reaction-diffusion equations and therefore special hyper-reduction methods are needed for an efficient ROM.

5.2.1 Hyper-Reduction for Moving Fronts

Apart from the slow decaying POD approximation errors, advection-reaction-diffusion systems pose another difficulty for model order reduction. The dynamics of advection-reaction-diffusion systems take place at a characteristic length scale l_f defined in Eq. (4.3). This characteristic scale is usually much smaller than the size of the domain or the traveling distance of the front. Hence, the FOM-RHS Eq. (5.5) and its gradient posses only a few spatial grid points with non-vanishing support per time step. Therefore, the hyper-reduction methods for non-linear manifolds [81, 78] cannot be applied. For example, the extended-ECSW scheme proposed by [78], or the gappy-POD based GNAT procedure [21] first introduced for non-linear manifolds in [81] cannot be used here, since they preselect a set of sample points, which is fixed for every time step and all $\boldsymbol{\mu} \in \mathcal{P}$.

In contrast, the FTR-hyper-reduction approach can help to identify the locations of the front to reduce computational complexity, while sustaining an accurate solution. Here, we propose an idea that is similar to the reduced integration domain (RID) method [122] for finite elements. By imposing a threshold criterion on each finite element, RID is choosing a reduced number of elements to describe a balance condition, i.e. to minimize the residual between internal and external forces. Similar to RID, we choose a selected number of M_p sampled/selected points to minimize the error of the projected right hand side (i.e. external/internal force):

$$\begin{aligned} 0 = \frac{\mathrm{d}}{\mathrm{d}\dot{\boldsymbol{a}}} \| \mathbf{J}_g(\boldsymbol{a})\dot{\boldsymbol{a}} - \boldsymbol{N}(g(\boldsymbol{a}), t, \boldsymbol{\mu}) \|^2_{\mathbf{P}_{\boldsymbol{a}}^2} = 2\mathbf{J}_g(\boldsymbol{a})^\top \mathbf{P}_{\boldsymbol{a}}^\top \mathbf{P}_{\boldsymbol{a}} \mathbf{J}_g(\boldsymbol{a})\dot{\boldsymbol{a}} \\ -2\mathbf{J}_g(\boldsymbol{a})^\top \mathbf{P}_{\boldsymbol{a}}^\top \mathbf{P}_{\boldsymbol{a}} \boldsymbol{N}(g(\boldsymbol{a}), t, \boldsymbol{\mu}) \,. \end{aligned} \tag{5.17}$$

Each of the M_p selected sample points corresponds to an index $0 \leq$

$i \leq M$, which is represented as the ith standard basis vector $\underline{\boldsymbol{e}}_i \in \mathbb{R}^M$ inside the rows of the selection matrix $\mathbf{P}_{\boldsymbol{a}} \in \mathbb{R}^{M_p \times M}$. Thus, the hyper-reduced Jacobian and right hand side $\mathbf{P}_{\boldsymbol{a}}\mathbf{J}_g, \mathbf{P}_{\boldsymbol{a}}\boldsymbol{N}$ are only computed at M_p sample points. Note that the stencil size of our finite difference scheme requires to compute $f(\underline{\boldsymbol{\phi}})$ on additional supporting mesh points, contained in $\hat{\mathbf{P}}_{\boldsymbol{a}} \in \mathbb{R}^{\hat{M}_p \times M}$. In practice, $\hat{\mathbf{P}}_{\boldsymbol{a}} f(\underline{\boldsymbol{\phi}}), \mathbf{P}_{\boldsymbol{a}}\mathbf{J}_g, \mathbf{P}_{\boldsymbol{a}}\boldsymbol{N}$ are not computed as matrix products, but as pointwise evaluations of $f(\underline{\boldsymbol{\phi}}), \mathbf{J}_g, \boldsymbol{N}$ at the corresponding sample points.

In contrast to RID, the selection matrix $\mathbf{P}_{\boldsymbol{a}} \colon \mathbb{R}^r \to \mathbb{R}^{M_p \times M}$ is dependent on the state $\boldsymbol{a}(t, \boldsymbol{\mu})$, which evolves over time (see Fig. 5.5). However, similar to what RID does for external forces, we have to add nodes, i.e. sample points, at which the RHS does not vanish. Since for the FTR $\boldsymbol{N}$ is non-vanishing at the locations of the front, i.e. at the roots of the level set function, we can perform time-dependent adaptive thresholding which defines $\mathbf{P}_{\boldsymbol{a}}$. The threshold search selects the M_p smallest values of the level set function $\underline{\boldsymbol{\phi}} = \boldsymbol{\Psi}\boldsymbol{a} \in \mathbb{R}^M$ at which we evaluate $\hat{\mathbf{P}}_{\boldsymbol{a}} f(\underline{\boldsymbol{\phi}}), \mathbf{P}_{\boldsymbol{a}}\mathbf{J}_g, \mathbf{P}_{\boldsymbol{a}}\boldsymbol{N}$. For two time instances, the sample points are visualized in Fig. 5.5 for the 2D ARD-system of Section 4.2.5. To reduce the costs of the threshold search, one might recompute the sample points only after a fraction of the characteristic time scale $t_f = l_f / c^*$, where l_f is defined in Eq. (4.3). Note, that the threshold search is a heuristic to perform a cheap minimization of the residual Eq. (5.17).

Further, it should be noted that in this work we are using explicit time integration schemes. Therefore, the aforementioned methods [81, 78] are not comparable in speedup, since they compare the ROM with implicit time integration schemes used in the offline stage. Nevertheless, applying implicit integration schemes during the online phase may benefit the stability of the resulting ROM. A promising and efficient method for explicit time integration schemes was proposed by [11] for reaction-diffusion systems in one spatial dimension. Although the framework cannot cope with topological changes since it relies on a smooth parametrization of the transport, the authors claim speedups of up to a factor of 130.

In the following, we will present some numerical examples utilizing the

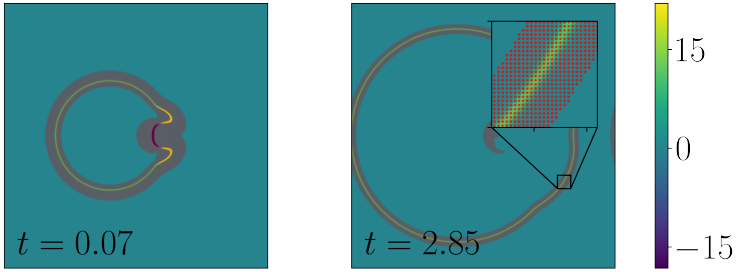


Figure 5.5: Color plot of the RHS $\boldsymbol{N}(\boldsymbol{q}, t, \mu)$ of the 2D advection-reaction-diffusion system Eq. (2.40) for two different time instances $t = 0.07$ (left) and $t = 2.85$ (right) and the corresponding sample points for a sample fraction of $M_p/M = 0.1$. The inset in the right color plot shows a close up of the location of the front.

hyper-reduction approach.

5.2.2 Numerical Examples

In this section, we numerically investigate the applicability of our framework. Therefore, we define the offline and online errors:

$$\text{offline/online err} = \frac{\left\| \mathbf{Q}^{\text{train/test}} - \tilde{\mathbf{Q}}^{\text{train/test}} \right\|_{\text{F}}}{\left\| \mathbf{Q}^{\text{train/test}} \right\|_{\text{F}}} . \tag{5.18}$$

Here, $\mathbf{Q} \in \mathbb{R}^{M \times (N_t N_{\mathcal{P}})}$ is the snapshot matrix containing all snapshots for the N_t time and $N_{\mathcal{P}}$ parameter instances $\boldsymbol{\mu} \in \mathcal{P}$ in its columns. The superscript "train" ("test") belongs to the snapshots $\boldsymbol{\mu} \in \mathcal{P}^{\text{train}}$ ($\mathcal{P}^{\text{test}}$) computed during the offline (online) phase.

The approximation $\tilde{\mathbf{Q}}^{\text{train}}$ is therefore either the reconstruction of the training data using the FTR-ansatz (Algorithm 2) or, in case of the POD, the projection onto the first r left singular vectors of $\mathbf{Q}^{\text{train}}$ contained in $\mathbf{U}_r \in \mathbb{R}^{M \times r}$, i.e. $\tilde{\mathbf{Q}}^{\text{train}} = \mathbf{U}_r^{\top} \mathbf{U}_r \mathbf{Q}^{\text{train}}$.

$\tilde{\mathbf{Q}}^{\text{test}}$ refers to the results evaluating the ROM Eq. (5.7) for the given time interval and parameters $\boldsymbol{\mu} \in \mathcal{P}^{\text{test}}$ using the reduced mapping

$g : \mathbb{R}^r \to \mathbb{R}^M$. Specifically, in the case of the POD, the dynamical ROM predictions use $g(\boldsymbol{a}) = \mathbf{U}_r \boldsymbol{a}$ as a reduced mapping, whereas $g(\boldsymbol{a}) = f(\boldsymbol{\Psi a})$ for the FTR.

Furthermore, we define the projection error (called manifold projection error in [16]):

$$\text{proj. err} = \frac{\left\| \mathbf{Q}^{\text{test}} - \tilde{\mathbf{Q}}_*^{\text{test}} \right\|_{\text{F}}}{\| \mathbf{Q}^{\text{test}} \|_{\text{F}}} \tag{5.19}$$

where $\tilde{\mathbf{Q}}_*^{\text{test}}$ is the best fit of $\mathbf{Q}^{\text{test}}$ with help of our the mapping g.

Reaction-Diffusion System in 1D

First, we test our approach on an 1D reaction-diffusion system that was taken and modified from [82]. The test case is based on a one-dimensional scalar non-linear reaction-diffusion equation

$$\partial_t q = \partial_{xx} q + \frac{8}{\mu^2} q^2 (q-1) \qquad (t, x) \in [0, 1] \times [-15, 15] \tag{5.20}$$

with corresponding analytical solution

$$q(x, t, \mu) = f\left(\frac{|x| - 2t/\mu - 2}{\mu} \right), \tag{5.21}$$

given that $f(x) = \text{sigmoid}(2x)$. We follow an offline-online procedure. First, we solve the FOM and set up a reduced map. In the second step, we simulate the projected and hyper-reduced system and compare the predictions to the FOM. First compute the numerical solution by discretizing Eq. (5.20) with $M = 4000$ grid points and solving it for $\mu \in \mathcal{P}^{\text{train}} = \{0.2, 1\}$ (further details can be found in Appendix B.2). The training data consists of 202 samples, including 101 samples of each training parameter. The training data is visualized as color plot in Fig. 5.6 together with the ROM prediction of the FTR using $r = 3$ and $\mu = \mu_{\text{test}} = 0.3$ in Eq. (5.20). The FTR algorithm (Algorithm 2) is run for 8000 steps using $\tau = 4$ and different truncation ranks $1 < r < 10$. After we have computed the reduced mapping $\underline{\boldsymbol{q}}(t, \mu) = f(\boldsymbol{\Psi a}(t, \mu))$ from the training set, we can compute the starting values $\boldsymbol{a}(0, \mu), \mu \in \mathcal{P}^{\text{test}}$ to test the ROM Eq. (5.7) by minimizing the initial condition of the ROM Eq. (5.8), using Gauss-Newton iterations [102]. As an initial

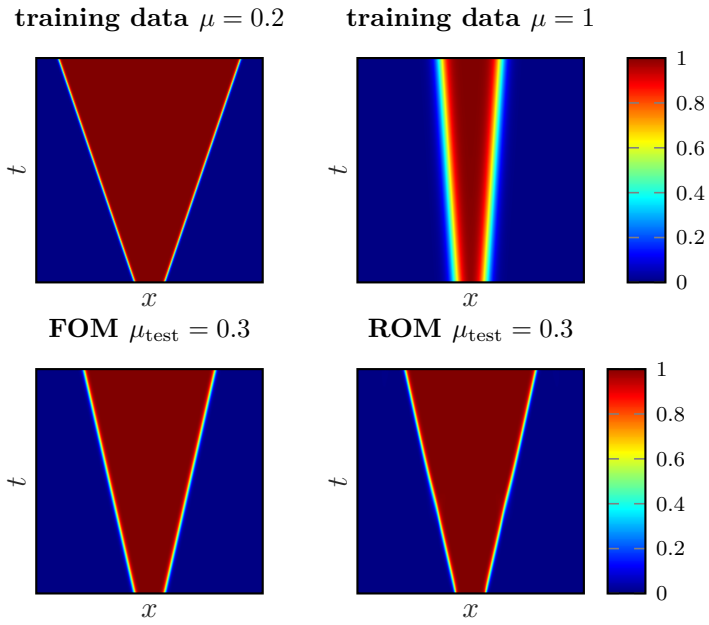


Figure 5.6: Training and test data of the reaction-diffusion system Eq. (5.20) with the ROM using $r = 3$ degrees of freedom.

guess for the minimization, we use the set of initial points $\{(\mu, \boldsymbol{a}(0,\mu)) \mid \mu \in \mathcal{P}^{\text{train}}\}$ and interpolate them for any given test parameter $\mu \in \mathcal{P}^{\text{test}}$. Thereafter, the ROM-solution for all test parameters $\mu \in \mathcal{P}^{\text{test}}$ is compared to the analytical solution Eq. (5.21). The results are reported as online errors in Table 5.2 together with the offline and projection errors. The online and projection errors are stated for the cumulated snapshots including the time interval $[0, 1]$ and all parameters $\mu \in \mathcal{P}^{\text{test}} = \{0.3, 0.4, \ldots, 0.9\}$. Table 5.2 also compares the results with the POD-Galerkin approach. The starting values for the POD-Galerkin-ROM are simply given by the orthogonal projection of $\boldsymbol{q}(0,\mu)$ onto the POD modes. It is remarkable to see that the FTR outperforms the POD by two orders of magnitude.

Next, we are interested in whether the gain in precision can be translated to speedups. Therefore, we study the performance of the hyper-reduced FTR-ROM explained in Section 5.2.1. Fig. 5.7 compares CPU-time and error for $M_p = 0.1M, 0.2M, 0.5M$ and M number of grid points, i.e. the dimension of the FOM. The figure indicates that even without hyper-reduction, speedups can be achieved compared to the FOM, due to larger time steps in the reduced coordinates. Comparing the hyper-reduced FTR with a sample fraction of $M_p/M = 0.2$ to 1, we see another speedup in CPU-time. For a reduction below $0.1M$ grid points, the solution is unstable and can lead to additional time steps, making the overall simulation slower.

	FTR			POD	
DOF	offline error	online error	proj. error	online error	proj. error
2	8.2e-03	1.4e-02	3.0e-03	3.6e-01	2.7e-01
3	2.6e-03	2.1e-02	6.6e-03	2.8e-01	2.0e-01
4	6.2e-04	2.7e-03	5.3e-04	2.4e-01	1.4e-01
5	5.3e-04	3.2e-03	7.2e-04	2.3e-01	1.1e-01
6	5.4e-04	2.5e-03	3.7e-04	2.5e-01	9.0e-02
7	5.0e-04	2.6e-03	2.7e-04	3.4e-01	7.3e-02
8	4.4e-04	2.1e-03	1.6e-04	2.7e-01	6.0e-02
9	2.0e-04	1.9e-03	2.2e-04	2.0e-01	5.0e-02

Table 5.2: Offline, online and projection errors for FTR and POD. The errors are reported for the cumulated snapshot data of the training and test parameters used in Section 5.2.2.

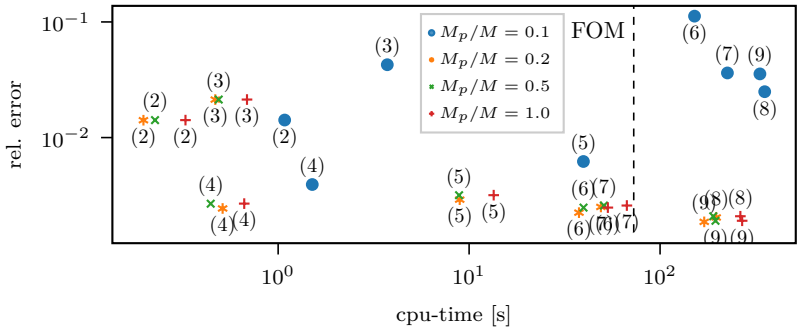


Figure 5.7: Error vs. CPU-time for the accumulated parameter range $\mu \in \mathcal{P}^{\text{test}}$. Different ranks r are indicated as (r) above or below the markers. The dashed line indicates the CPU-time needed for solving the FOM. The sampled fraction M_p/M in the hyper-reduced ROM is given in terms of the size M of the FOM.

Advection-Reaction-Diffusion System in 2D

Finally, we test the online performance of the hyper-reduced FTR-ROM on the advection-reaction-diffusion example of Eq. (2.40), introduced in Section 4.2.5. We build the training/test data $\mathbf{Q}^{\text{train}}$ from 101 equally spaced snapshots (visualized in Fig. 4.13) with $t \in [0, 3]$, $\gamma \in \mathcal{P}^{\text{train}} = \{10, 30, 50, 70, 100\}$ and respectively $\gamma \in \mathcal{P}^{\text{test}} = \{20, 40, 60, 80, 90\}$. The online, offline, and projection errors of the test case are shown in Fig. 5.8 together with the speedup generated by the hyper-reduction scheme. It is visible that the FTR outperforms the POD concerning offline and online errors. Furthermore, the utilized hyper-reduction strategy results in speedups with moderate online errors. Note that reducing the integration domain to about 10% of its original size (see sample points in Fig. 5.5) does not affect the online error, as can be seen from Fig. 5.8 (a). Fig. 5.8 (b) shows, that for small r, the additional costs ($\mathcal{O}(rM)$) for the matrix multiplication $\underline{\boldsymbol{\phi}}(t, \mu) = \boldsymbol{\Psi}\boldsymbol{a}(t, \mu)$ are negligible, compared to the evaluation of $\boldsymbol{N}$. However, as soon as r becomes large, the speedups of the hyper-reduction scheme are compensated by the computation of $\underline{\boldsymbol{\phi}}$ inside the threshold search. The balance point at which the additional costs com-

pensate the costs of the RHS is problem-dependent, but computing the sample points from $\underline{\phi}$ is a bottleneck of this method. Nevertheless, when aiming for more complex examples like combustion systems or 3D ARD systems, the outlined hyper-reduction approach will benefit from a more computationally complex RHS, which will shift the balance point towards a higher number of modes.

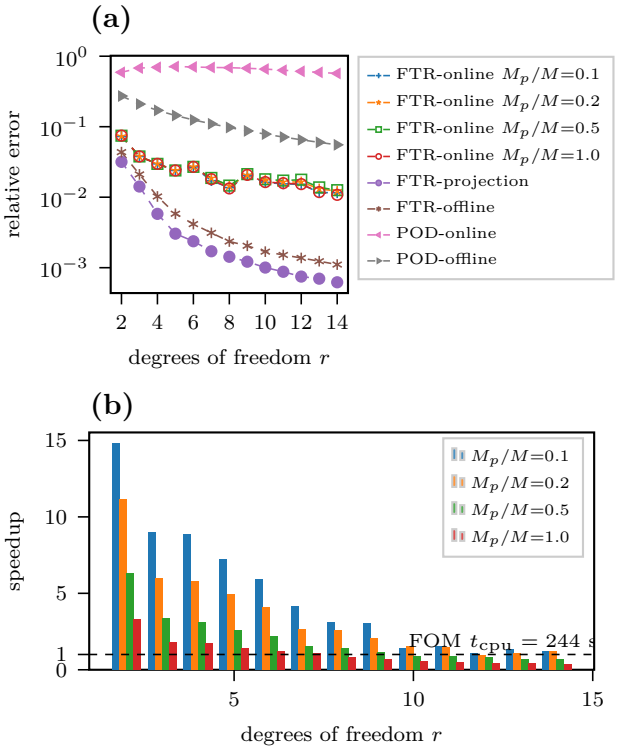


Figure 5.8: Hyper-reduced FTR results: (a) relative errors defined in Eqs. (5.18) and (5.19) for $\mathcal{P}^{\text{test/train}}$ using POD and FTR decomposition. (b) Speedup vs. degrees of freedom for the cumulated parameter range $\gamma \in \mathcal{P}^{\text{test}}$. The speedups are compared for different numbers of sampled grid points $M_p \leq M$. The CPU-time of the full order model (FOM) using $M = 512^2$ grid points is marked with a dashed line.

5.3 Summary

The ability of the FTR to predict new system states has been demonstrated for non-intrusive (Section 5.1) and intrusive (Section 5.2) ROMs. Since the FTR gives additional insights into the underlying structure (transport field $\underline{\phi}$), it allows us to use this information when predicting new system states. As an example, we heuristically reduced the integration domain during the online evaluation of the Galerkin projected ODE system, using the knowledge of $\underline{\phi}$. This can be seen as an adaptive version of the reduced integration domain method [122]. Other non-linear hyper-reduction methods preselect a set of sample points, on which the dynamics are evaluated. Since only sample points close to the front are relevant for the dynamics of the studied systems, such hyper-reduction methods may fail. Although the outlined hyper-reduction procedure yields speedups in CPU-time, it needs a substantially larger number of sample points M_p than required by the dimensions of the ROM $r \ll M_p \ll M$. Therefore, the construction of more efficient hyper-reduction schemes is left open for future research.

6 Conclusion and Outlook

This thesis contributes to methodical developments in the field of numerical analysis of fluid flows with an emphasis on model order reduction (MOR) for transport dominated fluid systems (TDFS).

TDFS are systems for which the transported quantity changes slowly with respect to the advection speed and therefore only require a few degrees of freedom (DOF) if the system is parametrized in a reference frame that moves with the transported quantity. Therefore, this thesis exploits adaptive and spatial coordinate transformations to accelerate simulations in the offline and online stage. These transformations can be implemented in any classical MOR approach.

To summarize, the major contributions of this thesis are

1. combining adaptive multiresolution methods to simulate and reduce TDFS;
2. transport compensation in dimensionality reduction; and
3. time-parametric non-linear reduced order models (ROM).

1. In various numerical studies we observe that spatial adaptivity can alleviate costs during simulation and reduction of TDFS while balancing the additional errors introduced by the adaptation scheme. Specifically, the employed block-based wavelet adaptation scheme allows for efficient storage and simulation of transport dominated flows. Acceleration up to a factor of three and memory savings up to a factor of 260 can be achieved for highly resolved 2D and 3D TDFS. The devised methodology provides a powerful tool to simulate and reduce

high-resolution data of TDFS.

Nevertheless, future work should address some of the limitations discussed in the following. The stability of the discretization scheme is not guaranteed on a block-based grid. One way to address this shortcoming is the combination of summation-by-parts simultaneous-approximation-term (SBP-SAT) discretization methods (for review [130]) with conservative wavelet methods (similar to [41, 1]), which guarantee stability by conservation properties. Furthermore, to improve the contribution of the compression errors ($\mathcal{O}(\epsilon)$) in the wavelet adaptive POD (wPOD) algorithm, the implemented biorthogonal wavelets should be substituted by orthogonal ones to ensure that the additional error introduced by the compression scales with $\mathcal{O}(\epsilon^2)$.

2. The main contribution of this thesis is the reduction of TDFS by compensating the transport with help of non-linear methods. The non-linear methods (autoencoder networks (AE), the shifted POD (sPOD) and front transport reduction (FTR)) show an improvement in reduction capability compared to dimensionality reduction on a linear subspace. Hence, significantly fewer degrees of freedom are required when approximating the input data, in contrast to the linear subspace created by standard ROM techniques, in particular the POD. Although FTR and sPOD have specific application regimes, they usually outperform the reduction capability of the general purpose AE. The developed FTR method can reduce TDFS with topological changing front structure, such as splitting or merging reaction fronts. As reactive systems were considered to be challenging [73] for classical MOR applications, the FTR can become an essential building block in future applications. Therefore, this contribution can be seen as the most important part of this thesis. Furthermore, we have improved the existing sPOD algorithms that have been presented in [114]. The newly developed algorithms allow for efficient, non-smooth optimization of high dimensional flow data. The new decomposition procedure enables the use of multiple constraints, which is an essential feature for further development.

Given that the FTR and sPOD are based on multiple evaluations of the SVD, they are much more computationally expensive than the POD. Similar costs are required for training a neural AE network. Therefore,

future work should combine sPOD and FTR with the wPOD to enable the applicability of these methods for highly resolved fluid systems. Furthermore, the generalization of FTR and sPOD for more complex fluid systems would be required. Finally, this would allow for the decoupling of physical systems in their co-moving reference frames so as to study their interactions, as presented for the two cylinder wake flow in Section 4.3.6.

3. Lastly, the newly developed FTR reduction method has been studied in the context of time parametric predictions for reactive flow systems. As the method is non-linear, manifold Galerkin projections have been used, which enable projecting the system dynamics onto the non-linear manifold created by the FTR mapping. Although the online prediction error can be significantly improved, speedups are comparably small. This is because the method requires a special hyper-reduction strategy that needs to select grid points close to the moving front at which the right-hand side is evaluated in every time step. These additional costs for the online selection of grid points limit the overall speedup. As an alternative, we propose the use of equation-free data-driven models. Given that such models need not evaluate the original dynamics, the computational costs are mainly limited by the evaluation of the non-linear reduced mapping.

Future work on the presented FTR-Galerkin approach should address the hyper-reduction strategy, as this is key to improving the speedup and stability of the resulting ROM. For example, a hyper-reduction strategy similar to the one presented in this work could be used in combination with the energy-conserving sampling and weighting (ECSW) method [78]. In the context of a multiresolution architecture, the joint application of the wavelet thresholding criterion and the selection of sample points could provide a promising framework.

A Appendix: Wavelet POD

A.1 L^2 Inner Products Expressed in the Wavelet Basis

The L^2 inner product is computed as a weighted sum of two fields $\boldsymbol{q}$ and $\boldsymbol{v}$ with K vector-components. To obtain this, we first refine both fields onto a unified grid with identical treecodes Λ^j, as explained in Section 3.2.1. Then we are able to compute Eq. (3.11) as a weighted sum over all blocks:

$$\langle \boldsymbol{q}, \boldsymbol{v} \rangle = \sum_{j=1}^{J_{\max}} \sum_{p \in \Lambda^j} \langle \boldsymbol{q}^{(p)}, \boldsymbol{v}^{(p)} \rangle \tag{A.1}$$

$$= \sum_{j=1}^{J_{\max}} \sum_{p \in \Lambda^j} \langle \underline{\boldsymbol{q}}^{(p)}, \underline{\boldsymbol{v}}^{(p)} \rangle_{\mathbf{I}_K \otimes \mathbf{W} \otimes \mathbf{W}} \Delta x_p \Delta y_p \tag{A.2}$$

$$\text{with weights: } (\mathbf{W})_{lm} = \langle \varphi_l^j, \varphi_m^j \rangle \,, \tag{A.3}$$

Note that this quadrature rule is exact for $\epsilon = 0$. We denote by $\mathbf{I}_K \otimes \mathbf{W} \otimes \mathbf{W}$ the Kronecker product between the weight matrix $\mathbf{W}$ and the identity matrix $\mathbf{I}_K \in \mathbb{R}^{K,K}$. The weight matrix is pre-computed by Eq. (A.3) and its non vanishing values $(\mathbf{W})_{ik} = w_{i-k}$ are shown in Table A.1. The listed matrix elements are the discrete values of the autocorrelation function between two compactly supported scaling functions φ, see Fig. A.1. Therefore $\mathbf{W}$ is sparse, symmetric and circulant. Since $\mathbf{W}$ is also strictly diagonal dominant and all diagonal entries are positive, $\mathbf{W}$ and the Kronecker product of such matrices is also positive definite.

$\lvert k\rvert$ (Order)	0	1	2	3	4	5
w_k $(N=2)$	2/3	1/6				
w_k $(N=4)$	0.8001	0.1370	-0.0402	0.0028	-7.6×10^{-5}	-1.5×10^{-7}

Table A.1: Values of the autocorrelation function $w_k = \int \varphi(x-k)\varphi(x)\mathrm{d}x$ of Deslauriers Dubuc interpolating functions of order $N = 2$ and $N = 4$ in the interior of the block. The values for DD4 are rounded.

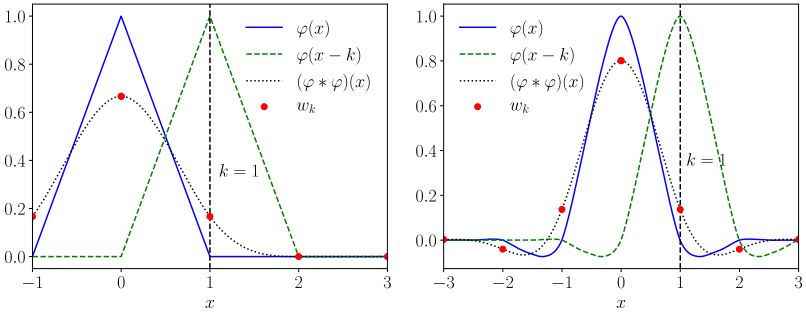


Figure A.1: Autocorrelation functions of Deslauriers interpolating scaling functions φ of order two (left) and order four (right).

A.2 Derivation of the Error Estimation given in eq. (23)

Using Eq. (3.15) the total error in Eq. (3.13) becomes

$$\mathcal{E}_{\text{wPOD}} \leq \frac{\sum_{i=1}^{N_t} \|\boldsymbol{q}_i - \boldsymbol{q}_i^\epsilon\|_2^2}{\sum_{i=1}^{N_t} \|\boldsymbol{q}_i\|_2^2} + \frac{\sum_{i=1}^{N_t} \|\boldsymbol{q}_i^\epsilon - \tilde{\boldsymbol{q}}_i^\epsilon\|_2^2}{\sum_{i=1}^{N_t} \|\boldsymbol{q}_i\|_2^2}. \tag{A.4}$$

Furthermore, we can simplify the first term with Eq. (3.16) inserting $\|\boldsymbol{q}_i - \boldsymbol{q}_i^\epsilon\|_2 \leq \epsilon \|\boldsymbol{q}_i\|_2$ into the nominator:

$$\frac{\sum_{i=1}^{N_t} \|\boldsymbol{q}_i - \boldsymbol{q}_i^\epsilon\|_2^2}{\sum_{i=1}^{N_t} \|\boldsymbol{q}_i\|_2^2} = \frac{\sum_{i=1}^{N_t} (\epsilon \|\boldsymbol{q}_i\|_2)^2}{\sum_{i=1}^{N_t} \|\boldsymbol{q}_i\|_2^2} = \epsilon^2 .$$

The second term in Eq. (A.4) can be expressed with the help of the eigenvalues of the correlation matrix.
We use the identities: $\sum_{i=1}^{N_t} \|\boldsymbol{q}_i^\epsilon - \tilde{\boldsymbol{q}}_i^\epsilon\|_2^2 = \sum_{k=r+1}^{N_t} \lambda_k^\epsilon$ for perturbed eigenvalues $\lambda_k^\epsilon = \lambda_k + l_k \epsilon$ and $\sum_{i=1}^{N_t} \|\boldsymbol{q}_i\|_2^2 = \sum_{k=1}^{N_t} \lambda_k$, yielding

$$\begin{aligned} \frac{\sum_{i=1}^{N_t} \|\boldsymbol{q}_i^\epsilon - \tilde{\boldsymbol{q}}_i^\epsilon\|_2^2}{\sum_{i=1}^{N_t} \|\boldsymbol{q}_i\|_2^2} &= \frac{\sum_{k=r+1}^{N_t} (\lambda_k + \epsilon l_k)}{\sum_{k=1}^{N_t} \lambda_k} \\ &= \mathcal{E}_{\text{POD}}(r, 0) + \mathcal{M}_r \epsilon . \end{aligned} \tag{A.5}$$

Here, $\mathcal{M}_r = \sum_{k=r+1}^{N_t} l_k / \sum_{k=1}^{N_t} \lambda_k$ is the *perturbation coefficient* of the total error. Note that the perturbations l_k are caused by the non vanishing mixed terms $\langle \varphi_\lambda^j, \psi_{\mu,\lambda}^j \rangle$ (see also [22]), when computing the correlation matrix from thresholded snapshots u_i^ϵ. For orthogonal wavelets, the first order perturbations would vanish. For slowly decaying eigenvalues λ_k, the perturbation coefficient $\mathcal{M}_r$ is typically very small, since the sum of perturbations l_k is small compared to the total energy. In this case it is reasonable to neglect the second term in Eq. (A.5):

$$\mathcal{E}_{\text{wPOD}} \lesssim \mathcal{E}_{\text{POD}}(r, 0) + \epsilon^2 . \tag{A.6}$$

However, in general $\mathcal{M}_r$ does not vanish and we only have linear convergence in ϵ:

$$\mathcal{E}_{\text{wPOD}} \leq \mathcal{E}_{\text{POD}}(r, 0) + \mathcal{M}_r \epsilon + \mathcal{O}(\epsilon^2) . \tag{A.7}$$

Note that $\mathcal{M}_r$ does not depend on ϵ, as all epsilon dependence has been removed. Hence, it is asymptotically a first order scheme in ϵ only. For a certain range, we can observe the second order, if $\mathcal{M}_r$ is sufficiently small. Eventually, for sufficiently small ϵ, the first order term will dominate.

A.3 Technical Details and Supplementary Material

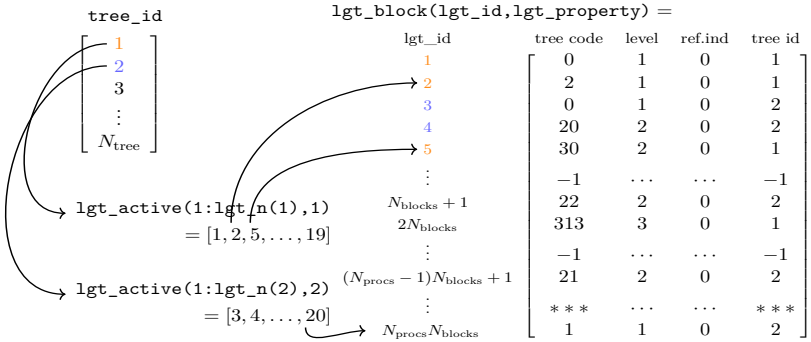


Figure A.2: Example of the light data structure in `WABBIT`, to be compared with Fig. 3.1. For each tree, `lgt_active` stores a light-ID list of all active blocks. With the blocks light-ID (`lgt_id`) all parameters in the forest (tree code, tree-ID, tree level, refinement status) can be accessed, from the corresponding row in `lgt_block`. Note that the order of the light-ID depends on the process rank.

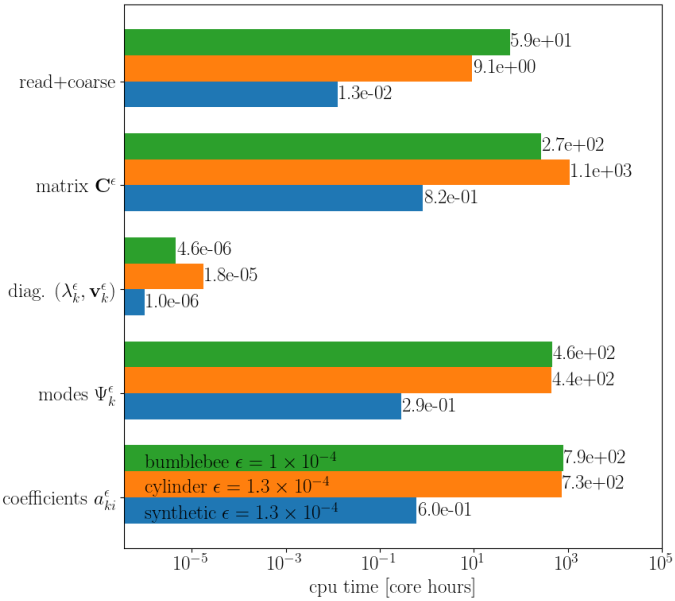


Figure A.3: CPU-time required for the individual steps of the wPOD algorithm described in Section 3.2.2. The CPU-time is shown for three different cases: 1.) the synthetic test case from Section 3.3.1 computed with $J_{\max} = 5$ and $\epsilon = 1.3 \times 10^{-4}$ on a Intel Core i5-7200U cpu (blue), 2.) the flow past a cylinder data Section 3.3.2 computed with $\epsilon = 1.3 \times 10^{-4}$ on Intel Xeon E5645 cpus (orange) and 3.) the bumblebee data of Section 3.3.2 with $\epsilon = 1.3 \times 10^{-4}$ using Intel Xeon Gold 6142 cpus (green). The comparison of the individual cases should be conducted with care, since they have been computed on different hardware and the data-structure (number of snaphots, blocks, block size, spatial dimension etc.) is different. Therefore, the computational costs of MPI communication overhead, block administration may vary. However, the overall proportions between the individual steps of the algorithm are comparable among the test cases.

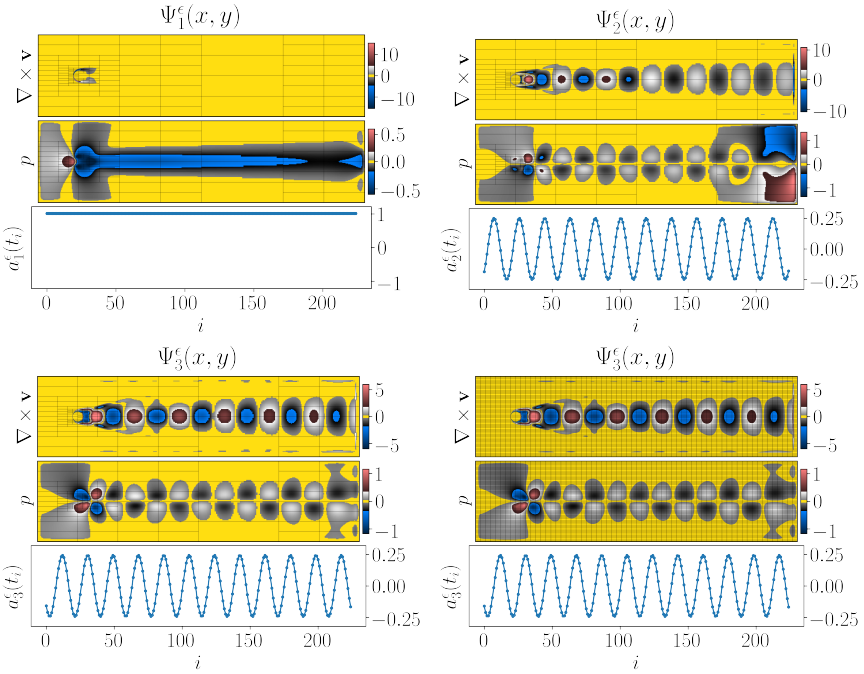


Figure A.4: First, three sparse modes $\boldsymbol{\psi}_k^\epsilon$ ($k = 1$ upper left, $k = 2$ upper right, $k = 3$ bottom left) with their corresponding amplitudes $a_k^\epsilon(t_i)$ computed with $\epsilon = 1.0 \times 10^{-2}$ and one dense ($\epsilon = 0$) mode $\boldsymbol{\psi}_3^\epsilon$ for comparison (bottom, right). Each figure shows the modes "vorticity" (labeled by $\boldsymbol{\nabla} \times \boldsymbol{v}$), computed from the two velocity components of the modes, and the pressure component (labeled by p). Note that the first mode represents the base flow, which is non oscillating, whereas the other modes always have an oscillating structure, with a frequency increasing with the mode number. When comparing the dense mode $\boldsymbol{\psi}_3^\epsilon$ of the non-adaptive case in the lower right of Fig. A.4 with the adapted modes on the lower left, no qualitative differences can be observed, except the local changes in the resolution.

B Appendix: Front Transport Reduction

B.1 Details on the Autoencoders Network Architecture and Training Hyperparameters

In this section, we provide detailed information on the architecture and training hyperparameters for the autoencoder networks. For both autoencoder variants (NN and FTR-NN), the encoder architecture $g_{\text{enc}} : \mathbb{R}^M \to \mathbb{R}^r$ is the same. Its task is to encode the spatial field $\underline{\boldsymbol{q}} \in \mathbb{R}^M$ into a latent space $\boldsymbol{a} \in \mathbb{R}^r$. It consists of four convolutional layers, each followed by a ELU activation and a batch normalization layer. After flattening the output, two fully connected layers follow, with another ELU activation and batch normalization layer in between. The output of the second fully connected layer represents the latent space with r degrees of freedom and it is not activated. A summary of the encoder architecture is listed in Table B.1. The decoder, $g_{\text{dec}} : \mathbb{R}^r \to \mathbb{R}^M$ maps the latent representation back to the spatial domain.

There are two different decoders used in this thesis labeled NN and FTR-NN autoencoder. The NN decoder mirrors the encoder architecture, using transposed convolutional layers instead of convolutional layers. The FTR-NN decoder consists of only a single fully connected layer with no bias with M (number of grid points) output channels. It applies a simple Matrix multiplication $\mathbf{W}\boldsymbol{a}$, where $\boldsymbol{a}$ is the vector with the latent representations and $\mathbf{W}$ is the learnable weight matrix of the layer. Next, the resulting output $\underline{\boldsymbol{\phi}} = \mathbf{W}\boldsymbol{a}$ is reshaped into the spatial domain. In analogy to the FTR ansatz $\underline{\boldsymbol{q}} \approx \tilde{\underline{\boldsymbol{q}}} = f(\underline{\boldsymbol{\phi}})$, both networks are activated with the physics dependent front function f in

the output layer. The layer details for both decoder networks are listed in Table B.2.

After splitting the data into a training and a test set, each network was trained using the ADAM optimizer with a learning rate of 0.0025 for up to $2 \cdot 10^4$ iterations, using all training samples as input batch. Every 500 iterations, the performance is tested on the test set. The network parameters that yield the best test results are saved.

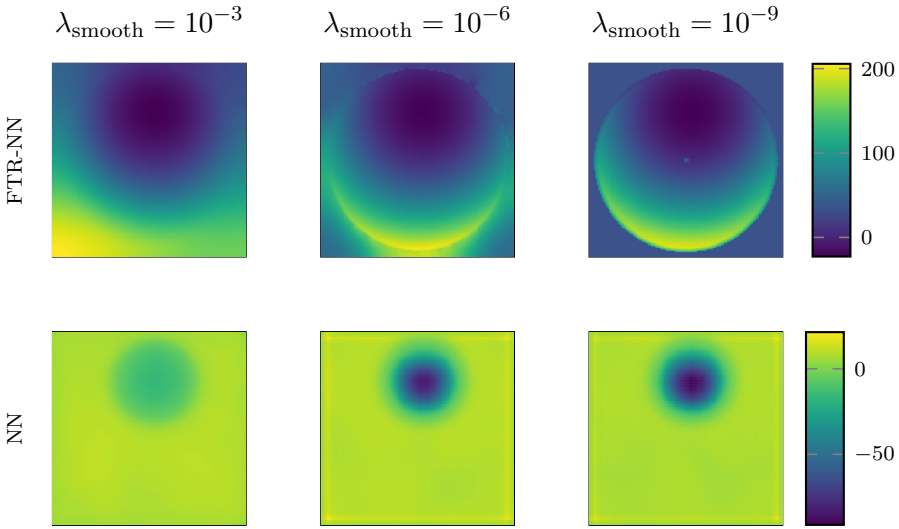


Figure B.1: Color plot of one snapshot of the FTR-NN and NN level set field ϕ using three degrees of freedom and different smoothness strength λ_{smooth}. The smoothness parameter λ_{smooth} controls the strength of the smoothness constraint Eq. (4.10).

Layer	Details			
	Input channels	Output channels	Kernel Size	Stride
Input of q (M grid points)	1			
2D Convolution	1	8	5	1
ELU + 2D BatchNorm				
2D Convolution	8	16	5	2
ELU + 2D BatchNorm				
2D Convolution	16	32	5	2
ELU + 2D BatchNorm				
2D Convolution	32	16	5	2
ELU + 2D BatchNorm				
Flatten Spatialy				
Fully Connected	$16 \cdot \tilde{M}$	512		
ELU + 1D BatchNorm				
Fully Connected	512	r		
Output of latent representation a		r		

Table B.1: Encoder network details. $\tilde{M}$ describes the number of remaining spatial grid points after all convolutional layers are applied. Each convolutional layer reduces the spatial resolution in each spatial direction by $N_{\text{out}} = (N_{\text{in}} - \text{kernel size})\,/\text{stride}+1$.

Layer	Details			
	Input channels	Output channels	Kernel Size	Stride
Input of latent representation a	r			
Fully Connected	r	512		
ELU + 1D BatchNorm				
Fully Connected	512	$16 \cdot \tilde{M}$		
ELU				
Unflatten Spatialy		16		
2D BatchNorm				
2D Transposed Convolution	16	32	5	2
ELU + 2D BatchNorm				
2D Transposed Convolution	32	16	5	2
ELU + 2D BatchNorm				
2D Transposed Convolution	16	8	5	2
ELU + 2D BatchNorm				
2D Transposed Convolution	8	1	5	1
Output of ϕ (M grid points)				

Table B.2: NN decoder network details

B.2 Simulation Details of the 1D Advection and Reaction-Diffusion PDE

In this section, we give additional details on the two PDE-examples with analytical solution. Namely, the PDE-example for

$$\text{advection} \qquad \begin{cases} 0 & = \partial_t q - u(t)\partial_x q \\ q(x,t) & = f(|x - u(t)| - 2) \end{cases}, \tag{B.1}$$

shown in Fig. 5.3 and

$$\text{reaction-diffusion} \qquad \begin{cases} 0 & = \partial_t q - \partial_{xx} q + \frac{8}{\mu^2} q^2(q-1) \\ q(x,t) & = f(\frac{|x| - 2 - t/\mu}{\mu}) \end{cases}. \tag{B.2}$$

In both examples, we use central finite difference of 6th order with periodic boundary conditions and an explicit Runge-Kutta integration method of 5th(4th) order for adaptive time stepping of the FOM and ROM ODE-system [40]. The numerical parameters for the FTR-decomposition and discretization are stated in Table B.3.

property	advection	reaction-diffusion
FOM- parameters		
Simulation time T	2.5	1
Domain $\mathbb{D}$	$[-20, 20]$	$[-15, 15]$
Grid resolution M	1000	4000
ROM- parameters		
Number of snapshots	202	202
FTR iterations	3000	8000
FTR step width τ	1	4
front function $f(x)$	$0.5(1 - \tanh(2.5x))$	$0.5(1 - \tanh(x))$

Table B.3: Parameters of the 1D advection and reaction-diffusion simulations and the decomposition procedure (Algorithm 2)

C Appendix: Shifted POD

C.1 Shift Transformations

The discrete transformation $(T^G\mathbf{Q})_{ij} = q(G(x_i, t_j))$ is realized with help of the snapshot matrix:

$$\mathbf{Q} = \left[\underline{\boldsymbol{q}}(t_1), \ldots, \underline{\boldsymbol{q}}(t_{N_t})\right] \in \mathbb{R}^{M \times N_t}, \quad \text{with}$$
$$\underline{\boldsymbol{q}}(t) = [q(x_1, t), \ldots, q(x_M, t)]^\top \in \mathbb{R}^M .$$

We assume an equidistant, periodic grid with lattice spacing h and use the shorthand notation $q_i = q((i-1)h, t)$. For optimal performance of the sPOD algorithm $T^G\mathbf{Q} = [T^{G(\cdot,t_1)}\underline{\boldsymbol{q}}(t_1), \ldots, T^{G(\cdot,t_{N_t})}\underline{\boldsymbol{q}}(t_{N_t})]$ is implemented as N_t sparse matrix multiplications, on the columns of $\mathbf{Q}$. The N_t matrices are set up prior to the algorithm.

In the co-moving frame, the transformed field at the position x_i corresponds to $\tilde{x} = G(x_i, t)$ in the reference frame. As $\tilde{x}$ can lie between sampled grid points (see Fig. C.1), we have to interpolate them. There-

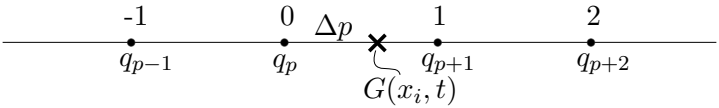


Figure C.1: Interpolation of the transformed grid values $G(x_i, t)$ with polynomial order $n = 3$.

fore, we make use of the Lagrangian basis polynomials:

$$\mathcal{L}_j = \prod_{\substack{k=(1-n)/2,\\ k\neq j}}^{(n+1)/2} \frac{\Delta p - k}{j-k} \qquad \Delta p = \frac{\tilde{x}}{h} - p \tag{C.1}$$

with $p = \left\lfloor \frac{\tilde{x}}{h} \right\rfloor$. The interpolated value at the new position is therefore:

$$I_n q(G(x_i,t)) = \sum_{j=(1-n)/2}^{(n+1)/2} q_{p+j}\mathcal{L}_j. \tag{C.2}$$

The most simple case of a shift transformation $G(x_i,t) = (x_i + \Delta_x(t), t)$ results in the index $p = \left\lfloor \frac{x_i+\Delta_x}{h} \right\rfloor = (i-1) + \left\lfloor \frac{\Delta_x}{h} \right\rfloor$. The corresponding transformation matrix $T^{\Delta_x} \in \mathbf{R}^{M\times M}$ for $n = 3$ is therefore:

$$T^{\Delta_x} = \begin{array}{c} \begin{array}{ccccccccccc} 1 & & & & & \left\lfloor \frac{\Delta_x}{h} \right\rfloor & & & & & M \end{array} \\ \begin{pmatrix} 0 & \cdots & \cdots & 0 & \mathcal{L}_{-1} & \mathcal{L}_0 & \mathcal{L}_1 & \mathcal{L}_2 & 0 & \cdots & 0 \\ \vdots & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & & \ddots & & & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & & & \ddots & \ddots & \ddots & \ddots & \mathcal{L}_2 \\ \mathcal{L}_2 & \ddots & \ddots & & \ddots & & & \ddots & \ddots & \ddots & \mathcal{L}_1 \\ \mathcal{L}_1 & \ddots & \ddots & \ddots & & \ddots & & & \ddots & \ddots & \mathcal{L}_0 \\ \mathcal{L}_0 & \ddots & \ddots & \ddots & \ddots & & \ddots & & & \ddots & \mathcal{L}_{-1} \\ \mathcal{L}_{-1} & \ddots & \ddots & \ddots & \ddots & \ddots & & \ddots & & & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & \ddots & \vdots \\ 0 & \cdots & 0 & \mathcal{L}_{-1} & \mathcal{L}_0 & \mathcal{L}_1 & \mathcal{L}_2 & 0 & 0 & \cdots & 0 \end{pmatrix} \end{array} \begin{array}{c} 1 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ M - \left\lfloor \frac{\Delta_x}{h} \right\rfloor \end{array}$$

Note that the lower diagonals are the result of the periodic continuation of the lattice. In two spatial dimensions $G(x,y,t) = (x + \Delta_x(t), y +$

$\Delta_y(t), t)$, the shift matrix is a Kronecker product of T^{Δ_x} and T^{Δ_y}:

$$T^G(\underline{\boldsymbol{q}}) = T^{\Delta_x} \otimes T^{\Delta_y} \cdot \begin{bmatrix} q^k_{1,1} \\ \vdots \\ q^k_{M_x,1} \\ q^k_{1,2} \\ \vdots \\ q_{M_x,M_y} \end{bmatrix} + \underline{\boldsymbol{\zeta}}. \tag{C.3}$$

Here $q^k_{ij} = q^k((i-1)h, (j-1)h, t)$, $i = 1, \ldots, M_x, j = 1, \ldots, M_y$ is a 2D field in a rectangular domain, with equal lattice spacing h and $\boldsymbol{\zeta}$ is the error term. In one spatial dimension, the ith component of the interpolation error vector $\boldsymbol{\zeta}$ is given by [14]

$$\zeta_i = q(\tilde{x}, t) - [T^{\Delta_x}\underline{\boldsymbol{q}}]_i = \frac{\partial^{(n+1)}_\xi q(\xi, t)}{(n+1)!} \underbrace{\prod_{j=(1-n)/2}^{(n+1)/2} (\Delta p - j)h}_{w(\Delta p) :=} \tag{C.4}$$

for some $\xi \in h[p - \frac{n-1}{2}, p + \frac{n+1}{2}]$ at the shifted position $\tilde{x} = G(x_i, t)$. Here, we use the shorthand notation $\partial^{(k)}_\xi q = \frac{\partial^k q}{\partial \xi^k}$ for the partial derivative of order k. Hence, for the maximums norm of the error vector at time instance t, we obtain

$$\left\|\underline{\boldsymbol{\zeta}}\right\|_\infty \leq \max_{\xi \in \mathbb{D}} \frac{\left|\partial^{(n+1)}_\xi q(\xi, t)\right|}{(n+1)!} \max_{\Delta p \in [0,1]} |w(\Delta p)| \tag{C.5}$$

and correspondingly for the whole shifted snapshot set $\mathbf{Q}^G = [q(G(x_i, t_j))]_{ij}$:

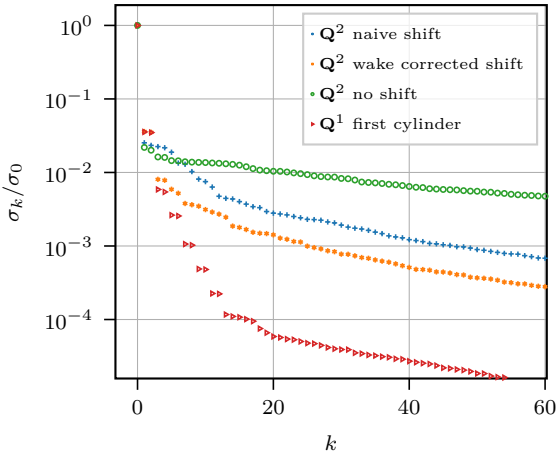
$$\mathbf{E} := \mathbf{Q}^G - T^G\mathbf{Q} \tag{C.6}$$

$$\|\mathbf{E}\|_\infty \leq \max_{(\xi,t) \in \mathbb{D} \times [0,T]} \frac{\left|\partial^{(n+1)}_\xi q(\xi, t)\right|}{(n+1)!} \max_{\Delta p \in [0,1]} |w(\Delta p)| \tag{C.7}$$

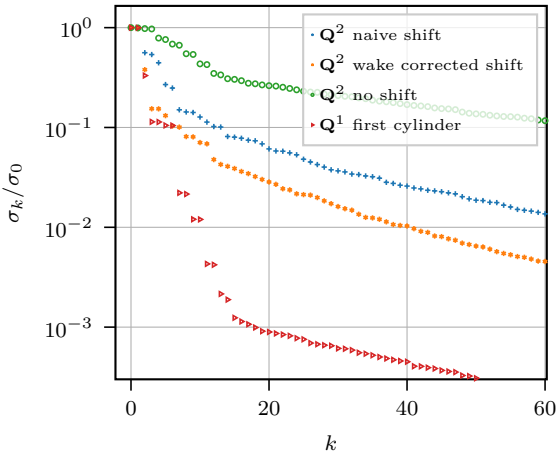
with

$$\max_{\Delta p \in [0,1]} |w(\Delta p)| = \begin{cases} \frac{1}{4}h^2 & \text{for } n = 1 \\ \frac{9}{16}h^4 & \text{for } n = 3 \\ \frac{225}{64}h^6 & \text{for } n = 5\,. \end{cases} \tag{C.8}$$

C.2 Shifted POD: Two Cylinder Wake Flow



(a) Velocity component u_1



(b) Velocity component u_2

Figure C.2: Singular values of the u_1 and u_2 snapshot matrix of the decoupled two cylinder system.

Bibliography

[1] M. Aechtner, N. K.-R. Kevlahan and T. Dubos, *A conservative adaptive wavelet method for the shallow-water equations on the sphere*, *Quarterly Journal of the Royal Meteorological Society* **141** (2015) 1712–1726.

[2] E. Anderson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz et al., *LAPACK: A portable linear algebra library for high-performance computers*, in *Proceedings of the 1990 ACM/IEEE Conference on Supercomputing*, Supercomputing '90, (Los Alamitos, CA, USA), pp. 2–11, IEEE Computer Society Press, 1990.

[3] P. Angot, C.-H. Bruneau and P. Fabrie, *A penalization method to take into account obstacles in incompressible viscous flows*, *Numerische Mathematik* **81** (1999) 497–520.

[4] P. Benner, S. Gugercin and K. Willcox, *A survey of projection-based model reduction methods for parametric dynamical systems*, *SIAM Review* **57** (2015) 483–531.

[5] H. Berestycki, F. Hamel and N. Nadirashvili, *Propagation speed for reaction–diffusion equations in general domains*, *Comptes Rendus Mathematique* **339** (2004) 163–168.

[6] H. Berestycki, F. Hamel and L. Roques, *Équations de réaction–diffusion et modèles d'invasions biologiques dans les milieux périodiques*, *Comptes Rendus Mathematique* **339** (2004) 549–554.

[7] M. Bergdorf and P. Koumoutsakos, *A lagrangian particle-wavelet method*, *Multiscale Modeling & Simulation* **5** (2006) 980–995.

[8] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[9] W.-J. Beyn and V. Thümmler, *Freezing solutions of equivariant evolution equations*, *SIAM Journal on Applied Dynamical Systems* **3** (2004) 85–116.

[10] F. Black, P. Schulze and B. Unger, *Projection-based model reduction with dynamically transformed modes*, *ESAIM: M2AN* **54** (2020) 2011–2043.

[11] F. Black, P. Schulze and B. Unger, *Efficient wildland fire simulation via nonlinear model order reduction*, *Fluids* **6** (2021) .

[12] F. Black, P. Schulze and B. Unger, *Modal decomposition of flow data via gradient-based transport optimization*, in *Active Flow and Combustion Control 2021* (R. King and D. Peitsch, eds.), (Cham), pp. 203–224, Springer International Publishing, 2022.

[13] O. Boiron, G. Chiavassa and R. Donat, *A high-resolution penalization method for large Mach number flows in the presence of obstacles*, *Computers & Fluids* **38** (2009) 703 – 714.

[14] M. Bollhöfer and V. Mehrmann, *Numerische Mathematik: Eine projektorientierte Einführung für Ingenieure, Mathematiker und Naturwissenschaftler*. Springer-Verlag, 2013.

[15] S. Boyd, N. Parikh and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[16] N. Cagniart, R. Crisovan, Y. Maday and R. Abgrall, *Model order reduction for hyperbolic problems: a new framework*, .

[17] J.-F. Cai, E. J. Candès and Z. Shen, *A singular value thresholding algorithm for matrix completion*, *SIAM Journal on optimization* **20** (2010) 1956–1982.

[18] J.-P. Caltagirone, *Sur l'intéraction fluide-milieu poreux; application au calcul des efforts exercés sur un obstacle par un fluide visqueux*, *Comptes rendus de l'Académie des sciences. Série II, Mécanique, physique, chimie, astronomie* **318** (1994) 571–577.

[19] E. J. Candès, X. Li, Y. Ma and J. Wright, *Robust principal component analysis?*, *Journal of the ACM (JACM)* **58** (2011) 1–37.

[20] C. Canuto, M. Y. Hussaini, A. Quarteroni, A. Thomas Jr et al., *Spectral methods in fluid dynamics*. Springer Science & Business Media, 2012.

[21] K. Carlberg, C. Farhat, J. Cortial and D. Amsallem, *The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows*, *Journal of Computational Physics* **242** (2013) 623–647.

[22] J. E. Castrillon-Candas and K. Amaratunga, *Fast estimation of continuous karhunen-loeve eigenfunctions using wavelets*, *IEEE Transactions on Signal Processing* **50** (2002) 78–86.

[23] A. Chambolle, V. Caselles, D. Cremers, M. Novaga and T. Pock, *An introduction to total variation for image analysis*, *Theoretical foundations and numerical methods for sparse recovery* **9** (2010) 227.

[24] G. Chiavassa and R. Donat, *A penalization technique for the efficient computation of compressible fluid flow with obstacles*, in *Hyperbolic Problems: Theory, Numerics, Applications* (S. Benzoni-Gavage and D. Serre, eds.), (Berlin, Heidelberg), pp. 89–100, Springer Berlin Heidelberg, 2008.

[25] A. J. Chorin, *A numerical method for solving incompressible viscous flow problems*, *Journal of Computational Physics* **2** (Aug., 1967) 12–26.

[26] A. Cohen, I. Daubechies and J. C. Feauveau, *Biorthogonal bases of compactly supported wavelets*, *Comm. Pure and Appl. Math.* **45** (1992) 485–560.

[27] A. Cohen, N. Dyn, S. M. Kaber and M. Postel, *Multiresolution schemes on triangles for scalar conservation laws*, *Journal of Computational Physics* **161** (June, 2000) 264–286.

[28] A. Cohen, S. Kaber, S. Müller and M. Postel, *Fully adaptive multiresolution finite volume schemes for conservation laws*, *Mathematics of Computation* **72** (2003) 183–225.

[29] R. Courant, K. Friedrichs and H. Lewy, *Über die partiellen Differenzengleichungen der mathematischen Physik*, *Mathematische Annalen* **100** (Dec., 1928) 32–74.

[30] R. Deiterding, *A parallel adaptive method for simulating shock-induced combustion with detailed chemical kinetics in complex domains*, *Computers & Structures* **87** (June, 2009) 769–783.

[31] R. Deiterding, M. O. Domingues, S. M. Gomes, O. Roussel and K. Schneider, *Adaptive multiresolution or adaptive mesh refinement? A case study for 2D Euler equations*, *ESAIM: Proceedings* **29** (2009) 28–42.

[32] R. Deiterding, R. Radovitzky, S. P. Mauch, L. Noels, J. C. Cummings and D. I. Meiron, *A virtual test facility for the efficient simulation of solid material response under strong shock and detonation wave loading*, *Engineering with Computers* **22** (Dec., 2006) 325–347.

[33] G. Deslauriers and S. Dubuc, *Interpolation dyadique*. École polytechnique de Montréal, 1987.

[34] G. Deslauriers and S. Dubuc, *Symmetric iterative interpolation processes*, in *Constructive Approximation: Special Issue: Fractal Approximation* (E. B. DeVore, Ronald A.and Saff, ed.), pp. 49–68. Springer US, Boston, MA, 1989.

[35] R. A. DeVore, *Nonlinear approximation*, *Acta Numerica* **7** (1998) 51–150.

[36] M. O. Domingues, S. M. Gomes, O. Roussel and K. Schneider, *Adaptive multiresolution methods*, *ESAIM: Proceedings* **34** (Dec., 2011) 1–96.

[37] M. Domingues, S. Gomes and L. Díaz, *Adaptive wavelet representation and differentiation on block-structured grids*, *Applied Numerical Mathematics* **47** (2003) 421 – 437.

[38] D. L. Donoho, *Interpolating wavelet transforms*, *Preprint, Department of Statistics, Stanford University* **2** (1992) 1–54.

[39] D. L. Donoho and J. M. Johnstone, *Ideal spatial adaptation by wavelet shrinkage*, *biometrika* **81** (1994) 425–455.

[40] J. Dormand and P. Prince, *A family of embedded Runge-Kutta formulae*, *Journal of Computational and Applied Mathematics* **6** (1980) 19–26.

[41] T. Dubos and N. K.-R. Kevlahan, *A conservative adaptive wavelet method for the shallow-water equations on staggered grids*, *Quarterly Journal of the Royal Meteorological Society* **139** (2013) 1997–2020.

[42] C. Eckart and G. Young, *The approximation of one matrix by another of lower rank*, *Psychometrika* **1** (1936) 211–218.

[43] T. Engels, D. Kolomenskiy, K. Schneider and J. Sesterhenn, *FluSI: A novel parallel simulation tool for flapping insect flight using a Fourier method with volume penalization*, *SIAM Journal on Scientific Computing* **38** (2016) S3–S24.

[44] T. Engels, D. Kolomenskiy, K. Schneider and J. Sesterhenn, *Numerical simulation of fluid–structure interaction with the volume penalization method*, *Journal of Computational Physics* **281** (Jan., 2015) 96–115.

[45] T. Engels and P. Krah, "*Python tools for the wavelet adaptive block-based solver for interactions in turbulence.*" `https://github.com/adaptive-cfd/python-tools`, Visited 25. Oct 2022.

[46] T. Engels, K. Schneider, J. Reiss and M. Farge, *A 3D wavelet-based incompressible Navier-Stokes solver for fully adaptive computations in time-varying geometries*, *Communications in Computational Physics* (2021) .

[47] F. Fang, C. Pain, I. Navon, M. Piggott, G. Gorman, P. Allison et al., *Reduced-order modeling of an adaptive mesh ocean model*, *International journal for numerical methods in fluids* **59** (2009) 827–851.

[48] F. Fedele, O. Abessi and P. Roberts, *Symmetry reduction of turbulent pipe flows*, *Journal of Fluid Mechanics* **779** (2015) 390–410.

[49] R. Fisher, *The wave of advance of advantageous genes*, *Annals of Eugenics* **7** (1937) 355–369.

[50] C. A. Fletchet, *Computational techniques for fluid dynamics; Vol 1.* Springer, Springer, NY (United States), 1991.

[51] B. Fornberg, *A practical guide to pseudospectral methods*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 1996.

[52] S. Fresca and A. Manzoni, *POD-DL-ROM: Enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition*, *Computer Methods in Applied Mechanics and Engineering* **388** (2022) 114181.

[53] S. Fresca, L. Dede' and A. Manzoni, *A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs*, *Journal of Scientific Computing* **87** (Apr., 2021) 61.

[54] K. Fukami, T. Murata, K. Zhang and K. Fukagata, *Sparse identification of nonlinear dynamics with low-dimensionalized flow representations*, *Journal of Fluid Mechanics* **926** (2021) A10.

[55] J. Furthney, "`scikit-fmm` - *a python extension module which implements the fast marching method*." `https://github.com/scikit-fmm/scikit-fmm`, Visited 27. Apr 2022.

[56] R. Galagusz, D. Shirokoff and J.-C. Nave, *A Fourier penalty method for solving the time-dependent Maxwell's equations in*

domains with curved boundaries, *Journal of Computational Physics* **306** (Feb., 2016) 167–198.

[57] M. Gavish and D. L. Donoho, *The optimal hard threshold for singular values is* $4/\sqrt{3}$, *IEEE Transactions on Information Theory* **60** (2014) 5040–5053.

[58] A. Ghorbani, A. Abid and J. Zou, *Interpretation of neural networks is fragile*, in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 3681–3688, 2019.

[59] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016.

[60] I. J. Goodfellow, J. Shlens and C. Szegedy, *Explaining and harnessing adversarial examples*, *arXiv preprint arXiv:1412.6572* (2014) .

[61] C. Gräßle and M. Hinze, *POD reduced-order modeling for evolution equations utilizing arbitrary finite element discretizations*, *Advances in Computational Mathematics* **44** (2018) 1941–1978.

[62] C. Gräßle, M. Hinze, J. Lang and S. Ullmann, *POD model order reduction with space-adapted snapshots for incompressible flows*, *Advances in Computational Mathematics* **45** (2019) 2401–2428.

[63] C. Greif and K. Urban, *Decay of the Kolmogorov n-width for wave problems*, *Applied Mathematics Letters* **96** (2019) 216–222.

[64] J.-L. Guermond and P. Minev, *High-order time stepping for the incompressible Navier–Stokes equations*, *SIAM Journal on Scientific Computing* **37** (Jan., 2015) A2656–A2681.

[65] E. Guilmineau and P. Queutey, *A numerical simulation of vortex shedding from an oscillating circular cylinder*, *Journal of Fluids and Structures* **16** (2002) 773 – 794.

[66] K. Hadeler and F. Rothe, *Travelling fronts in nonlinear diffusion equations*, *Journal of Mathematical Biology* **2** (1975) 251–263.

[67] E. T. Hale, W. Yin and Y. Zhang, *Fixed-point continuation for l_1-minimization: methodology and convergence*, *SIAM Journal on Optimization* **19** (2008) 1107–1130.

[68] N. Halko, P. G. Martinsson and J. A. Tropp, *Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions*, *SIAM Review* **53** (2011) 217–288.

[69] A. Harten, *Discrete multi-resolution analysis and generalized wavelets*, *Applied Numerical Mathematics* **12** (1993) 153 – 192.

[70] C. K. Hemelrijk and H. Hildenbrandt, *Schools of fish and flocks of birds: their shape and internal structure by self-organization*, *Interface Focus* **2** (2012) 726–737.

[71] W. D. Henshaw and D. W. Schwendeman, *Moving overlapping grids with adaptive mesh refinement for high-speed reactive and non-reactive flow*, *Journal of Computational Physics* **216** (Aug., 2006) 744–779.

[72] P. Holmes, J. L. Lumley, G. Berkooz and C. W. Rowley, *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press, 2012.

[73] C. Huang, K. Duraisamy and C. Merkle, *Challenges in reduced order modeling of reacting flows*, in *2018 Joint Propulsion Conference*, p. 4675, 2018.

[74] C. Huang, K. Duraisamy and C. Merkle, *Challenges in reduced order modeling of reacting flows*, in *2018 Joint Propulsion Conference*, p. 4675, 2018.

[75] W. Huang and R. D. Russell, *Adaptive Moving Mesh Methods*, vol. 174 of *Applied Mathematical Sciences*. Springer New York, New York, NY, 2011, 10.1007/978-1-4419-7916-2.

[76] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[77] S. Jaensch, M. Merk, E. Gopalakrishnan, S. Bomberg, T. Emmert, R. Sujith et al., *Hybrid CFD/low-order modeling of nonlinear thermoacoustic oscillations*, *Proceedings of the Combustion Institute* **36** (2017) 3827–3834.

[78] S. Jain and P. Tiso, *Hyper-reduction over nonlinear manifolds for large nonlinear mechanical systems*, *Journal of Computational and Nonlinear Dynamics* **14** (05, 2019) .

[79] E. N. Karatzas, F. Ballarin and G. Rozza, *Projection-based reduced order models for a cut finite element method in parametrized domains*, *Computers & Mathematics with Applications* **79** (2020) 833–851.

[80] M. E. Khalili, M. Larsson and B. Müller, *Immersed boundary method for viscous compressible flows around moving bodies*, *Computers & Fluids* **170** (2018) 77 – 92.

[81] Y. Kim, Y. Choi, D. Widemann and T. Zohdi, *A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder*, *Journal of Computational Physics* (2021) 110841.

[82] O. M. Knio, H. N. Najm and P. S. Wyckoff, *A semi-implicit numerical scheme for reacting flow: Ii. stiff, operator-split formulation*, *Journal of Computational Physics* **154** (1999) 428–467.

[83] A. Kolmogorov, I. Petrovskii and N. Piscunov, *A study of the equation of diffusion with increase in the quantity of matter, and its application to a biological problem*, *Byul. Moskovskogo Gos. Univ.* **1** (1937) 1–25.

[84] V. Kornilov, R. Rook, J. ten Thije Boonkkamp and L. D. Goey, *Experimental and numerical investigation of the acoustic response of multi-slit bunsen burners*, *Combustion and Flame* **156** (2009) 1957–1970.

[85] G. Kutyniok, P. Petersen, M. Raslan and R. Schneider, *A theoretical analysis of deep neural networks and parametric PDEs*, *Constructive Approximation* **55** (Feb., 2022) 73–125.

[86] H. Lange, "*Fourier to Koopman implementation.*" `https://github.com/helange23/from_fourier_to_koopman`, Visited 6. Dec 2021.

[87] H. Lange, S. Brunton and J. Kutz, *From Fourier to Koopman: Spectral methods for long-term time series prediction.*, *J. Mach. Learn. Res.* **22** (2021) 1–38.

[88] P. Läuchli, *Jordan-Elimination und Ausgleichung nach kleinsten Quadraten*, *Numerische Mathematik* **3** (1961) 226–240.

[89] K. Lee and K. Carlberg, *Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders*, *Journal of Computational Physics* **404** (2020) 108973.

[90] J. Liandrat and P. Tchamitchian, *Resolution of the 1d regularized burgers equation using a spatial wavelet approximation*, NASA Contractor Rep. 187480, ICASE Rep. 90-83, NASA Langley Research Center, Hampton, 1990.

[91] Z. Lin, M. Chen and Y. Ma, *The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices*, *arXiv preprint arXiv:1009.5055* (2010) .

[92] C. Liu and C. Hu, *An efficient immersed boundary treatment for complex moving object*, *Journal of Computational Physics* **274** (2014) 654 – 680.

[93] E. Loth, S. Sivier and J. Baum, *Adaptive unstructured finite element method for two-dimensional detonation simulations*, *Shock Waves* **8** (Feb., 1998) 47–53.

[94] X.-Y. Lu and C. Dalton, *Calculation of the timing of vortex formation from an oscillating cylinder*, *Journal of Fluids and Structures* **10** (1996) 527 – 541.

[95] L. V. D. Maaten, E. Postma and J. V. den Herik, *Dimensionality reduction: a comparative*, *Journal of Machine Learning Research* **10** (2009) 13.

[96] S. G. Mallat, *Multiresolution approximations and wavelet*

orthonormal bases of $L^2(\mathbb{R})$, *Transactions of the American mathematical society* **315** (1989) 69–87.

[97] S. Mallat, *A wavelet tour of signal processing*. Academic Press, Boston, third edition ed., 2009.

[98] M. Mendez, M. Balabane and J.-M. Buchlin, *Multi-scale proper orthogonal decomposition of complex fluid flows*, *Journal of Fluid Mechanics* **870** (2019) 988–1036.

[99] O. Mercier, X. Yin and J. Nave, *The characteristic mapping method for the linear advection of arbitrary sets*, *SIAM Journal on Scientific Computing* **42** (2020) A1663–A1685.

[100] R. Mojgani and M. Balajewicz, *Physics-aware registration based auto-encoder for convection dominated pdes*, *arXiv preprint arXiv:2006.15655* (2020) .

[101] R. Mojgani and M. Balajewicz, *Low-rank registration based manifolds for convection-dominated pdes*, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 399–407, 2021.

[102] J. Moré, *The Levenberg-Marquardt algorithm: implementation and theory*, in *Numerical analysis*, pp. 105–116. Springer, 1978.

[103] S. Müller, *Adaptive multiscale schemes for conservation laws*, vol. 27. Springer Science & Business Media, 2002.

[104] N. J. Nair and M. Balajewicz, *Transported snapshot model order reduction approach for parametric, steady-state fluid flows containing parameter dependent shocks*, *International Journal for Numerical Methods in Engineering* **117** (2019) 1234–1262.

[105] R. Nguyen van yen, *Wavelet-based study of dissipation in plasma and fluid flows*. PhD thesis, l'École Normale Supérieure de Paris, December, 2010.

[106] R. H. Nochetto, K. G. Siebert and A. Veeser, *Theory of adaptive finite element methods: an introduction*, in *Multiscale, nonlinear and adaptive approximation*, pp. 409–542. Springer, 2009.

[107] M. Nonino, F. Ballarin, G. Rozza and Y. Maday, *Overcoming slowly decaying Kolmogorov n-width by transport maps: application to model order reduction of fluid dynamics and fluid–structure interaction problems*, *arXiv:1911.06598 [cs, math]* (Nov., 2019) .

[108] M. Ohlberger and S. Rave, *Reduced basis methods: Success, limitations and future challenges*, *Proceedings of the Conference Algoritmy* (2016) 1–12.

[109] T. Ohwada and P. Asinari, *Artificial compressibility method revisited: Asymptotic numerical method for incompressible Navier–Stokes equations*, *Journal of Computational Physics* **229** (Mar., 2010) 1698–1723.

[110] E. J. Parish, C. R. Wentland and K. Duraisamy, *The adjoint Petrov–Galerkin method for non-linear model reduction*, *Computer Methods in Applied Mechanics and Engineering* **365** (2020) 112991.

[111] M. Quade, M. Abel, J. Kutz and S. Brunton, *Sparse identification of nonlinear dynamics for rapid model recovery*, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **28** (2018) 063116.

[112] J. N. Reddy and D. K. Gartling, *The finite element method in heat transfer and fluid dynamics*. CRC press, 2010.

[113] J. Reiss, *A family of energy stable, skew-symmetric finite difference schemes on collocated grids*, *Journal of Scientific Computing* **65** (2015) 821–838.

[114] J. Reiss, *Optimization-based modal decomposition for systems with multiple transports*, *SIAM Journal on Scientific Computing* **43** (2021) A2079–A2101.

[115] J. Reiss, P. Schulze, J. Sesterhenn and V. Mehrmann, *The shifted proper orthogonal decomposition: A mode decomposition for multiple transport phenomena*, *SIAM Journal on Scientific Computing* **40** (2018) A1322–A1344.

[116] J.-F. Remacle, J. E. Flaherty and M. S. Shephard, *An adaptive*

discontinuous Galerkin technique with an orthogonal basis applied to compressible flow problems, *SIAM Review* **45** (Jan., 2003) 53–72.

[117] D. Rim, S. Moe and R. J. LeVeque, *Transport reversal for model reduction of hyperbolic partial differential equations*, *SIAM/ASA Journal on Uncertainty Quantification* **6** (Jan., 2018) 118–150.

[118] O. Roussel and K. Schneider, *Coherent vortex simulation of weakly compressible turbulent mixing layers using adaptive multiresolution methods*, *Journal of Computational Physics* **229** (2010) 2267–2286.

[119] C. W. Rowley, I. G. Kevrekidis, J. E. Marsden and K. Lust, *Reduction and reconstruction for self-similar dynamical systems*, *Nonlinearity* **16** (May, 2003) 1257–1275.

[120] C. W. Rowley and J. E. Marsden, *Reconstruction equations and the Karhunen–Loève expansion for systems with symmetry*, *Physica D: Nonlinear Phenomena* **142** (Aug., 2000) 1–19.

[121] C. Rowley, I. Kevrekidis, J. Marsden and K. Lust, *Reduction and reconstruction for self-similar dynamical systems*, *Nonlinearity* **16** (may, 2003) 1257–1275.

[122] D. Ryckelynck, *A priori hyperreduction method: an adaptive approach*, *Journal of Computational Physics* **202** (Jan., 2005) 346–366.

[123] P. J. Schmid, *Dynamic mode decomposition of numerical and experimental data*, *Journal of Fluid Mechanics* **656** (2010) 5–28.

[124] K. Schneider and O. V. Vasilyev, *Wavelet methods in computational fluid dynamics*, *Annual review of fluid mechanics* **42** (2010) 473–503.

[125] J. A. Sethian, *A fast marching level set method for monotonically advancing fronts*, *Proceedings of the National Academy of Sciences* **93** (1996) 1591–1595.

[126] E. Shchepakina and E. Tropkina, *Order reduction for problems*

with traveling wave solutions to reaction–diffusion systems, *Journal of Physics: Conference Series* **1745** (feb, 2021) 012109.

[127] L. Sirovich, *Turbulence and the dynamics of coherent structures. part i-iii*, *Quarterly of Applied Mathematics* **45** (1987) 561–571.

[128] M. Sroka, "*Wavelet adaptive block-based solver for interactions in turbulence - RNS (reactive navier stokes physics)*." `https://github.com/mario-sroka/WABBIT`, Visited 03. May 2022.

[129] M. Sroka, T. Engels, S. Mutzel, P. Krah and J. Reiss, "`WABBIT`-*Wavelet adaptive block-based solver for interactions in turbulence*." `https://github.com/adaptive-cfd/WABBIT`, Visited 9. Oct 2022.

[130] M. Svärd and J. Nordström, *Review of summation-by-parts schemes for initial–boundary-value problems*, *Journal of Computational Physics* **268** (2014) 17–38.

[131] W. Sweldens and P. Schröder, *Building your own wavelets at home*, *Wavelets in Computer Graphics* (1997) .

[132] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow et al., *Intriguing properties of neural networks*, 2014.

[133] T. Taddei, *A registration method for model order reduction: Data compression and geometry reduction*, *SIAM Journal on Scientific Computing* **42** (Jan., 2020) A997–A1027.

[134] S. Ullmann, M. Rotkvic and J. Lang, *POD-Galerkin reduced-order modeling with adaptive finite element snapshots*, *Journal of Computational Physics* **325** (2016) 244–258.

[135] M. Unser, *Approximation power of biorthogonal wavelet expansions*, *IEEE Transactions on Signal Processing* **44** (1996) 519–527.

[136] O. V. Vasilyev, *Solving multi-dimensional evolution problems with localized structures using second generation wavelets*,

International Journal of Computational Fluid Dynamics **17** (Mar., 2003) 151–168.

[137] S. Verma, G. Novati and P. Koumoutsakos, *Efficient collective swimming by harnessing vortices through deep reinforcement learning*, *Proceedings of the National Academy of Sciences* **115** (2018) 5849–5854.

[138] S. Volkwein, *Optimal control of a phase-field model using proper orthogonal decomposition*, *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik: Applied Mathematics and Mechanics* **81** (2001) 83–97.

[139] H. G. Weller, G. Tabor, H. Jasak and C. Fureby, *A tensorial approach to computational continuum mechanics using object-oriented techniques*, *Computers in physics* **12** (1998) 620–631.

[140] G. Welper, *Transformed snapshot interpolation with high resolution transforms*, *SIAM Journal on Scientific Computing* **42** (2020) A2037–A2061.

[141] G. Welper, *Interpolation of functions with parameter dependent jumps by transformed snapshots*, *SIAM Journal on Scientific Computing* **39** (2017) A1225–A1250.

[142] F. Williams, *Combustion theory*. Benjamin Cummings, Menlo Park, CA, 1985.

[143] X.-Y. Yin, K. Schneider and J.-C. Nave, *A characteristic mapping method for the three-dimensional incompressible euler equations*, *arXiv preprint arXiv:2107.03504* (2021) .

[144] D. Yu and S. Chakravorty, *A randomized proper orthogonal decomposition technique*, in *2015 American Control Conference (ACC)*, pp. 1137–1142, July, 2015.

[145] G. Zumbusch, *Parallel multilevel methods: adaptive mesh refinement and loadbalancing*. Springer Science & Business Media, 2012.

Related Publications and Preprints

[146] A. Kovárnová, P. Krah, J. Reiss and M. Isoz, *Shifted proper orthogonal decomposition and artificial neural networks for time-continuous reduced order models of transport-dominated systems*, in *Topical Problems of Fluid Mechanics 2022*, 2022. DOI.

[147] P. Krah, S. Büchholz, M. Häringer and J. Reiss, *Front transport reduction for complex moving fronts*, *arXiv preprint arXiv:2202.08208* (2022) .

[148] P. Krah, T. Engels, K. Schneider and J. Reiss, *Wavelet adaptive proper orthogonal decomposition for large-scale flow data*, *Advances in Computational Mathematics* **48** (2022) 1–40.

[149] P. Krah, M. Goldack, T. Engels, K. Schneider and J. Reiss, *Data-driven reduced order modeling for flows with moving geometries using shifted POD*, *hal-archives hal-03396325* (Feb., 2022) .

[150] P. Krah, M. Sroka and J. Reiss, *Model order reduction of combustion processes with complex front dynamics*, in *Numerical Mathematics and Advanced Applications ENUMATH 2019* (F. J. Vermolen and C. Vuik, eds.), (Cham), pp. 803–811, Springer International Publishing, 2021.

[151] M. Sroka, T. Engels, P. Krah, S. Mutzel, K. Schneider and J. Reiss, *An open and parallel multiresolution framework using block-based adaptive grids*, in *Active Flow and Combustion Control 2018*, pp. 305–319. Springer, 2019.

Author Contribution Statement

In the following I declare the individual contributions to the publications and preprints, which contribute to this thesis.

P. Krah, T. Engels, K. Schneider and J. Reiss, *Wavelet adaptive proper orthogonal decomposition for large-scale flow data*, Advances in Computational Mathematics 48(2022) 1–40.

Contribution to this thesis:
Main parts of the publication have been used in a slightly modified form in Section 2.1.1 and Chapter 3.

Author contributions:

Philipp Krah	concept, implementation and validation of the wPOD algorithm, writing the original draft of the paper
Thomas Engels	implementation of the wavelet adaptation scheme, review of manuscript, bumblebee simulation
Kai Schneider	initial idea and concept of algorithm, theoretical details of the wavelet representation, normalization and error balancing, reviewing and editing
Julius Reiss	proposed initial idea to combine POD with wavelet adaptation, supervision, reviewing and editing

P. Krah, M. Sroka and J. Reiss, *Model Order Reduction of Combustion Processes with Complex Front Dynamics*, in *Numerical Mathematics and Advanced Applications ENUMATH 2019* (F. J. Vermolen and C. Vuik, eds.), (Cham), pp. 803–811, Springer International Publishing, 2021.

Contribution to this thesis:
Main parts of the publication have been used in a slightly modified form in Section 4.2.2.

Author contributions:

Philipp Krah	implementation and validation of the FTR signed distance algorithm, writing the original draft of the paper
Mario Sroka	simulation of the reactive Navier Stokes equations and review of the paper
Julius Reiss	proposed the initial idea and implemented the prototype, supervision, reviewing and editing

P. Krah, S. Büchholz, M. Häringer and J. Reiss, *Front transportreduction for complex moving fronts*, *arXiv preprint arXiv:2202.08208* (2022) (under review)

Contribution to this thesis:
Parts of the publication have been used in a slightly modified form in Sections 4.2 and 4.4 and Chapter 5.

Author contributions:

Philipp Krah	concept, method, implementation, writing original draft
Steffen Büchholz	implementation of neural networks, reviewing and editing
Mathias Häringer	computations of the Bunsen flame, reviewing and editing
Julius Reiss	proposed initial idea and application to ARD-systems, supervision, reviewing and editing

P. Krah, M. Goldack, T. Engels, K. Schneider and J. Reiss, *Data-driven reduced order modeling for flows with moving geometries using shifted POD*, *hal-archives hal-03396325* (Feb.,2022)

Contribution to this thesis:
Parts of the publication have been used in a slightly modified version in Section 4.3.

Author contributions:

Philipp Krah	initial idea, concept, method, implementation, writing original draft
Miriam Goldack	implementation, reviewing and editing
Thomas Engels	CFD-computations, reviewing and editing
Kai Schneider	supervision, reviewing and editing
Julius Reiss	supervision, reviewing and editing

M. Sroka, T. Engels, **P. Krah**, S. Mutzel, K. Schneider and J. Reiss, *An open and parallel multiresolution framework usingblock-based adaptive grids*, in *Active Flow and CombustionControl 2018*, pp. 305–319. Springer, 2019.

Author contributions:

Mario Sroka	writing the original draft, computations, implementation
Thomas Engels	prototype, writing the original draft, computations, implementation
Philipp Krah	implementation, reviewing and editing
Sophie Mutzel	implementation, reviewing and editing
Kai Schneider	proposed the initial idea, supervision, reviewing and editing
Julius Reiss	proposed the initial idea, supervision, reviewing and editing

A. Kovárnová, **P. Krah**, J. Reiss and M. Isoz, *Shifted proper orthogonal decomposition and artificial neural networks for time-continuous reduced order models of transport-dominated systems*, in *Topical Problems of Fluid Mechanics 2022*, 2022.

Author contributions:

Anna Kovárnová	implementation, validation, simulation, writing the original draft
Philipp Krah	implementation of the shifted POD, reviewing and editing
Julius Reiss	supervision, reviewing and editing
Martin Isoz	proposed initial idea, supervision, writing the original draft

Reduced order modeling aims to approximate large and complex dynamical systems with smaller ones to reduce simulation costs in the design or control processes of these systems. Standard linear mode-based model order reduction (MOR) can fail for transport dominated fluid systems (TDFS), because the underlying transport is often inherently non-linear. However, given that, in case of TDFS, the transported quantity changes slowly with respect to the advection speed, only few degrees of freedom (DOF) are required if the system is parametrized in a reference frame that moves with the transported quantity. This thesis aims to improve MOR of TDFS by implementing well adapted non-linear coordinate transformations that take the transport of the systems into account. The first part of this thesis addresses non-linear adaptive wavelet-filtering of flow systems to adjust the computational resources to the co-moving reference frame, already when generating the data. To enable MOR with the utilized adaptive data structure, a wavelet-based adaptive version of the proper orthogonal decomposition (POD) is proposed that balances error contributions of wavelet compression and POD truncation. The second part addresses non-linear reduction methods that compensate the transport by a shift or with help of an auxiliary field parametrizing the transport. Compared to the POD, the new methods allow for efficient decomposition of TDFS with only few DOF, while providing better physical insight into the system compared to neural autoencoder networks. The presented methodology enables the decomposition of reactive systems with topologically changing front structure, such as splitting or merging reaction fronts, that pose difficulties for many non-linear reduction methods. The last part studies the ability of the non-linear reduction methods to predict new system states using intrusive and non-intrusive reduced order models. In the case of the latter, manifold Galerkin projections with a tailored hyper-reduction strategy are utilized, enabling rapid simulations of reactive flows. Given that reactive systems are considered challenging for classical MOR applications, this contribution is an essential building block for future applications.