Journal of Scientific Computing (2023) 96:28
https://doi.org/10.1007/s10915-023-02210-9

Front Transport Reduction for Complex Moving Fronts

Nonlinear Model Reduction for an Advection–Reaction–Diffusion Equation with a Kolmogorov–Petrovsky–Piskunov Reaction Term

Philipp Krah[1] · Steffen Büchholz[3] · Matthias Häringer[2] · Julius Reiss[3]

Received: 16 February 2022 / Revised: 28 February 2023 / Accepted: 4 April 2023 /
Published online: 3 June 2023

Abstract

This work addresses model order reduction for complex moving fronts, which are transported by advection or through a reaction–diffusion process. Such systems are especially challenging for model order reduction since the transport cannot be captured by linear reduction methods. Moreover, topological changes, such as splitting or merging of fronts pose difficulties for many nonlinear reduction methods and the small non-vanishing support of the underlying partial differential equations dynamics makes most nonlinear hyper-reduction methods infeasible. We propose a new decomposition method together with a hyper-reduction scheme that addresses these shortcomings. The decomposition uses a level-set function to parameterize the transport and a nonlinear activation function that captures the structure of the front. This approach is similar to autoencoder artificial neural networks, but additionally provides insights into the system, which can be used for efficient reduced order models. In addition to the presented decomposition method, we outline a tailored hyper-reduction method that is based on the reduced integration domain method. The capability of the approach is illustrated by various numerical examples in one and two spatial dimensions, including an advection–reaction–diffusion system with a Kolmogorov–Petrovsky–Piskunov reaction term and real life application to a two-dimensional Bunsen flame.

Keywords Fluid dynamics · Combustion · Complex moving fronts · Model order reduction · Advection–reaction–diffusion equation · Machine learning

✉ Philipp Krah
philipp.krah@tu-berlin.de

[1] Institute of Mathematics, Technische Universität Berlin, Straße des 17. Juni 136, 10623 Berlin, Germany

[2] Institute of Thermofluiddynamics, Technische Universität München, Boltzmannstr. 15, 85747 Garching, Germany

[3] Institute of Fluid Mechanics and Engineering Acoustics, Technische Universität Berlin, Müller-Breslau-Str. 15, 10623 Berlin, Germany

1 Introduction

This article addresses model order reduction for reactive flows. These flows often exhibit sharp fronts, like flames, which makes their simulation computational expensive. This suggests applying model reduction for reducing simulation costs. However, classical model order reduction methods fail [1] due to the sharp, moving fronts, that pose challenges for reducing and predicting new system states. This manuscript addresses these issues by presenting a new decomposition method together with efficient strategies to evaluate the dynamics of the reduced system. For our study we use advection–reaction–diffusion systems (ARD) with a nonlinear Kolmogorov–Petrovsky–Piskunov (KPP) reaction term, as the complex front dynamics with topology changes of such systems feature essential difficulties for model order reduction, while the analysis is simplified because the reacting quantity is scalar and bounded.

Model order reduction (MOR) has been studied for various ARD systems [2–6], including those related to self-ignition from coal [2], wildland fires [3], combustion systems [4, 5, 7, 8] and financial risk marketing [6]. In this study we focus on systems that exhibit locally one-dimensional traveling fronts. The compact support of the moving fronts is challenging for linear reduced basis methods, such as the proper orthogonal decomposition (POD). The POD approximates a set of snapshots $q(\boldsymbol{x}, t_i)$, with $i = 1, \dots, N_t$, by separation of variables

$$q(\boldsymbol{x}, t) \approx \sum_{k=1}^{r} \hat{a}_k(t)\hat{\psi}_k(\boldsymbol{x}), \tag{1}$$

with help of time amplitudes $\hat{a}_k(t)$ and spatial modes $\hat{\psi}_k(\boldsymbol{x})$, computed by a singular value decomposition (SVD). Based on (1), (Petrov–)Galerkin methods (see for a review [9, 10]) are then used to project the original dynamics of the system onto a lower dimensional subspace to obtain a reduced order model (ROM) of only a few variables that enable an efficient evaluation. In parametric MOR (pMOR) this is referred to as a two-stage offline–online procedure [9]. In the offline step, the reduced basis is built from a large number of trajectories that have been simulated for a predefined set of parameters using the full order model (FOM/offline). Thereafter, in the online step, any new parameter can be simulated efficiently using the ROM.

Unfortunately, moving fronts with sharp gradients significantly slow down the convergence of (1), thereby limiting the use of pMOR. This has been numerically investigated for reactive flows in [1]. The reducibility, i.e. the smallest approximation error that is obtained using an r-dimensional linear subspace (1), can be theoretically quantified by the Kolmogorov width [11] or equivalently by the Hankel singular values as shown in [12]. While an exponential decay of the Kolmogorov width is necessary for the success of classical MOR, it was shown in [13, 14] that for typical transport-dominated systems the Kolmogorov width decays only with $\sim 1/\sqrt{r}$, where r is the dimension of the subspace. The slow decay is often referred to as the Kolmogorov barrier [15].

Currently, three different groups of methods can be identified in the literature that attempt to break the Kolmogorov barrier:

1. *Transport compensating methods*. These methods improve the convergence of (1) by compensating the transport, for which many authors use coordinate transformations [16–24] to align the front onto a reference frame in which the moving front is stationary. This allows to efficiently decompose the temporal variation of the front shape into few basis functions. Similarly, ideas have been applied to parameter-dependent shock positions [25–27] or fluid–structure interaction with parametrized geometries [28, 29]. Often these methods make assumptions that limit their applicability to the specific problem at hand.

For example the shifted POD assumes multiple wave like structures, where the transport dependent movement is known n[16, 18] or at least a sufficiently smooth function in time [17], which is easy to parametrize and itself independent of $\boldsymbol{x}$. First attempts to automate the generation of the transforms have been introduced in [17, 30, 31]. Similarly [28, 29] construct mappings that are based on the knowledge of the parametrization of the deformation or movement of the structures that interacts with the flow. Other authors [23] for example make use of knowledge of the advection speed/characteristics in the system. Although these methods have already been applied to reactive flows, as in [3] for a wildland fire model or in [7, 18, 32] for detonation combustion, they were studied for 1D systems only. Essential for transport compensating methods is a mapping from a transformed domain on the simulation domain, which is often assumed to be a smooth bijection. This complicates their applicability to complex transports with topological changes as they often appear in reactive flows.

2. *Artificial neural networks.* A common dimension reduction method uses artificial neural networks. Although the specific implementations differ, most authors [33–36] 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 then 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 transport compensating methods, 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 an alignment of the transported quantities is difficult. 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. Here, an optimal approximation is not guaranteed and it is not clear which structures or features are identified by an AE. Existing feature interpretation methods are usually based on first-order linear approximations around a given input. However, they may fail if the inputs are perturbed [37–39].
3. *Adaptive basis methods.* Adaptive basis methods such as the dynamic low rank approximation [40], the adaptive basis and adaptive sampling discrete empirical interpolation method (AADEIM) [41], the reduced basis method with adaptive time partitioning [42] or the online adaptive basis refinement and compression method [43] break the Kolmogorov barrier since they exploit the time-local low-rank structure of transport-dominated fluid systems. This is achieved by adapting the dominant modes to the solution locally in time. Here, [42, 43] build on an offline–online procedure. However, to account for the time-local structure [42, 43], either assign a basis to each sub-intervals in time that is switched during the online stage [42] or locally refine a hierarchical basis according to an online error estimator [43]. In contrast, DLRA and AADEIM do not require an offline stage since they compute the reduced basis during the online stage. Dynamical low-rank approximation [40] decomposes the PDEs solution with help of a low-rank matrix factorization and derives evolution equations to update the factors of the solution in time. These evolution equations are determined by minimizing a residual, that arises from projecting the solution onto the manifold of low-rank matrices. The AADEIM method [41], updates the basis using an adaptive version (ADEIM [44]) of the discrete empirical interpolation method [45]. Since the underlying description of the adaptive basis methods does not make explicit assumptions on the transport, they can be readily applied to ARD systems, as shown for 1D premixed flames in [41, 46]. Nevertheless, as recently observed in [47] for hyperbolic shallow water moment equations, an offline–online procedure can lead to

substantially faster ROMs during the online stage compared to an adaptive basis method [40], as the offline–online procedure does not require expensive online updates.

In this work, we build on the idea of transport compensation together with the classical offline–online model order reduction procedure. However, in contrast to previous works [16–21], where the authors aim for a simple coordinate transformation and compensate with a couple of modes, we use a single mode to describe the front, but use a transformation that is only locally defined. To this end, we make use of an auxiliary field $\phi(\boldsymbol{x}, t)$, which parameterizes the transport efficiently, together with a shape function f to retain the front shape:

$$q(\boldsymbol{x}, t) \approx f(\phi(\boldsymbol{x}, t)) \quad \text{s. t.} \quad \phi(\boldsymbol{x}, t) = \sum_{k=1}^{r} a_k(t)\psi_k(\boldsymbol{x}). \tag{2}$$

The auxiliary field $\phi\colon \mathbb{R}^d \times [0, T] \to \mathbb{R}$ allows to embed the local one-dimensional front movement into a d-dimensional transport. Since the transport is only parameterized locally, changes in the topology of the front surface can be captured. A similar approach was introduced in [4], where ϕ was constructed with help of a signed distance function and the front function f was determined from a fit to the reacting front. While improving the approximation, this was found to be not optimal, since the obtained signed distance function does not have to be of low rank. Here, we follow a similar approach, but we formulate an optimization problem to compute ϕ. The resulting description (2) is called Front Transport Reduction (FTR) in the remainder of this manuscript. Due to the nonlinearly activated linear space created by the span of $\{\psi_k(\boldsymbol{x})\}_{k=1,\dots,r}$, this approach shows many parallels to artificial neural networks. It can be seen as the decoder part of a shallow autoencoder structure. While shallow autoencoders have been used in previous studies [36], we are explicitly incorporating the underlying physical assumptions and thereby obtain interpretable results of reduced variables.

The second part of this manuscript addresses dynamical ROM predictions of ARD systems using the FTR ansatz (2). Here, many different methods have been applied to ARD systems in combination with the POD (1), which can be categorized into intrusive (based on the initial set of equations) [1, 48, 49] or non-intrusive (based on data of the system state) [5, 50, 51] reduced order models. Intrusive models project the original equation system on the reduced manifold, which is nonlinear in our approach. For non-linear reduced mappings, so-called manifold Galerkin methods have been used in combination with neural networks in [52] and with dynamical transformed modes in [53]. Unfortunately, manifold Galerkin methods require special hyper-reduction schemes to gain speedups in the resulting ROM. Examples of these methods are the extended-ECSW scheme proposed by [54], the gappy-POD based GNAT procedure [52] first introduced for nonlinear manifolds in [36] or the shifted DEIM [3]. The idea of all of these methods is to evaluate the nonlinear dynamics of the underlying system for a small number of points to determine the evolution of the parameters in the reduced space. Unfortunately, the extended-ECSW scheme [54] and the GNAT procedure [36, 54], cannot be used for ARD systems with sharp advected fronts, since they preselect a fixed set of points, but the dynamics are localized only near the moving front. As already noted by [55–57] for linear projection-based model order reduction of nonlinear equations, the choice of sampling points is crucial for the accuracy and complexity of the resulting ROM. We discuss this problem and state a practical solution using a special hyper-reduction scheme, based on the reduced integration domain (RID) method [58]. Furthermore, we examine the ability of the FTR mapping to predict new system states with the help of a non-intrusive method introduced by [59].

Structure of the Article

The remainder of the article is structured into three parts. The first part, Sect. 2, is dedicated to the FTR decomposition, where we motivate the decomposition and introduce two algorithms to solve the corresponding high dimensional optimization problem. While the first algorithm is based on iterative thresholding of singular values (see Sect. 2.2), the other algorithm uses artificial neural networks (see Sect. 2.3). The algorithms are applied and compared for two synthetic examples in Sect. 2.4 and one example of a two-dimensional (2D) ARD system with topology change in Sect. 2.5. In the second part, Sect. 3, we use the low dimensional description of the FTR to predict new system states via non-intrusive MOR (Sect. 3.1) and intrusive MOR (Sect. 3.2). For the latter, we propose a special hyper-reduction method. The resulting ROM is tested for 1D and 2D ARD systems in Sect. 3.4. Finally, we summarize our results in the last part, Sect. 4.

Nomenclature

Matrices are denoted in capital letters with straight, bold font $\mathbf{A} \in \mathbb{R}^{M\times M}$ and vectors are denoted by $\boldsymbol{x} \in \mathbb{R}^M$. Whenever a scalar function $f : \mathbb{R} \to \mathbb{R}$ is applied on a vector valued quantity $\boldsymbol{x}$, we assume element-wise operation on the entries of $\boldsymbol{x} = (x_1, \dots, x_M)$, if not stated otherwise, and write $f(\boldsymbol{x})$ instead of $(f(x_1), \dots, f(x_M))$, similarly for matrices $f(\mathbf{A})$. The Hadamard product $\odot$ is defined by the element-wise multiplication of matrix entries. Furthermore, if $q(\boldsymbol{x}, t, \mu) \in \mathbb{R}$ is the solution of a scalar PDE, its (discrete space) ODE counterpart is denoted by a vector $\boldsymbol{q}(t, \mu) \in \mathbb{R}^M$ containing the spatial values of q in its components. Correspondingly, the snapshot matrix $\mathbf{Q}$ contains all time and parameter snapshots in its columns: $\mathbf{Q} = [\boldsymbol{q}(t_1, \mu_1), \boldsymbol{q}(t_2, \mu_1), \dots, \boldsymbol{q}(t_{N_t}, \mu_P)]$. 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 $\dot{q} = \frac{\partial q}{\partial t}$. The Laplacian of a function q is defined by: $\nabla^2 q = \sum_{i=1}^d \frac{\partial^2}{\partial x_i^2} q$, where we make use of the nabla operator: $\nabla = \left(\frac{\partial}{\partial x_1}, \dots, \frac{\partial}{\partial x_d}\right)$.

2 Dimension Reduction Methods for Complex Moving Reaction Fronts

In this section, we introduce the front transport reduction (FTR) and motivate why our special nonlinear reduction method is advantageous when decomposing reactive flows. We present two FTR decomposition approaches, that are based on iterative thresholding (Sect. 2.2) and a special autoencoder network (Sect. 2.3).

2.1 A Nonlinear Decomposition Approach for Moving Fronts

To motivate our decomposition approach, we consider advection–reaction–diffusion systems:

$$\frac{\partial q}{\partial t} = \boldsymbol{u} \cdot \nabla q + \kappa \nabla^2 q + R(q). \tag{3}$$

with a Kolmogorov–Petrovsky–Piskunov (KPP) [60] reaction term $R(q) = \mu q^{\alpha}(q-1)$ with $\alpha > 0, \mu > 0$. This systems describes how a quantity or reactant $q(\boldsymbol{x}, t)$ spreads in space $\boldsymbol{x} \in \Omega \subset \mathbb{R}^d$, $d > 0$ over time $t \in [0, T]$. This spread can be caused by the advection with

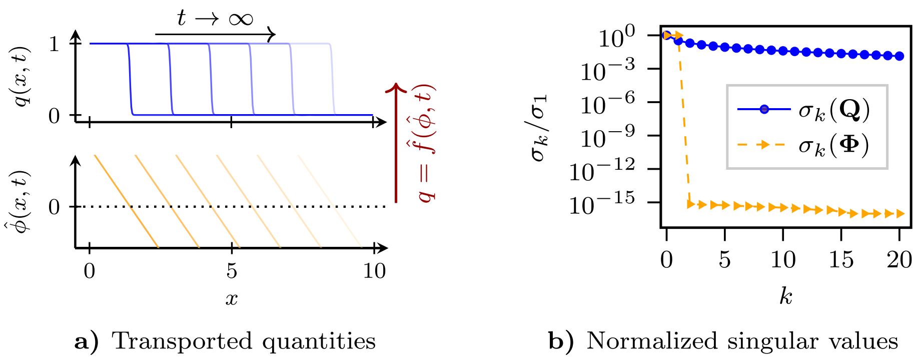


Fig. 1 Illustration of the basic idea of the front transport reduction method (Example 1). **a** The FTR replaces the sharp traveling front structure q (blue curves), by a level set function $\hat{\phi}$ (orange lines) and a nonlinear front function $\hat{f}$ (indicated by the red arrow). Both quantities share locally the same transport (shift). However, the level set field $\hat{\phi}(x,t)$ is of low rank and can be therefore parameterized with only a few POD basis functions (here: $\{x, 1\}$). **b** Comparison of the singular values of the snapshot matrices $\mathbf{\Phi}, \mathbf{Q} \in \mathbb{R}^{101,101}$. The snapshot matrices $\hat{\mathbf{\Phi}}_{ij} = \hat{\phi}(i\Delta x, j\Delta t)$, $\mathbf{Q}_{ij} = q(i\Delta x, j\Delta t)$ are computed from (6) using $\Delta x = 10\Delta t = 0.1$ and $\mu = 0.2$ (Color figure online)

velocity $\boldsymbol{u} \in \mathbb{R}^d$ or an interplay between diffusion $\nabla^2 q$ and reaction processes $R(q)$. KPP-systems exhibit traveling or pulsating fronts [61–64] that are locally one-dimensional. Locally one-dimensional means that the spatial variable $\boldsymbol{x} = (x_1, x_2, \dots, x_d)$ of the reactant q can be transformed to $\hat{\boldsymbol{x}} = (\hat{\phi}, \hat{x}_2, \dots, \hat{x}_d)$, where $\hat{x}_2, \dots, \hat{x}_d$ are on a hyperplane tangential to the front of the traveling wave. On this hyperplane, all gradients in the equation vanish relative to the $\hat{\phi}$-terms that are normal to the traveling wave. The resulting equation is therefore only dependent on one spatial variable $\hat{\phi}$ (see [65, p. 87], for details). Therefore, we can assume that the solution of (3) can be transformed into

$$q(\boldsymbol{x}, t) = \hat{f}(\hat{\phi}(\boldsymbol{x}, t), t). \tag{4}$$

where the front-profile of the traveling wave is described by $\hat{f}\colon \mathbb{R}^2 \to \mathbb{R}$ and $\hat{\phi}\colon \Omega \times \mathbb{R} \to \mathbb{R}$ determines the location of the front. Note, that this is not a special property for KPP-systems, but a general property shared by other reaction or combustion systems [65–67].

Example 1 (One dimensional KPP-system) We consider the one-dimensional reaction–diffusion system taken from [68]

$$\partial_t q = \partial_{xx} q + \frac{8}{\mu^2} q^2 (q - 1) \tag{5}$$

with $\mu > 0$. A special solution in form of a traveling wave is

$$q(x,t) = \frac{1}{2}\left(1 + \tanh\left(\frac{x - 2t/\mu}{\mu}\right)\right). \tag{6}$$

If we identify $\hat{\phi}(x,t) = \frac{x - 2t/\mu}{\mu}$ as a local distance to the front and the time constant front profile $\hat{f}(x,t) = \frac{1}{2}(1 + \tanh(x))$ we exactly recover the form (4). The relation between the analytical solution q, the front function $\hat{f}$ and the transport field $\hat{\phi}$ are shown in Fig. 1a.

As shown in Fig. 1 the transport field $\hat{\phi}$ is usually smooth and slowly varying compared to q, thus we assume it to be of low rank. For the Example 1 this is visualized by the comparison of the normalized singular values when decomposing the snapshot matrices $\hat{\mathbf{\Phi}}_{ij} = \hat{\phi}(x_i, t_j)$ and $\mathbf{Q}_{ij} = q(x_i, t_j)$ with help of the SVD.

Our decomposition approach exploits this observation. For given snapshot data $\mathbf{Q} \in \mathbb{R}^{M \times N_t}$ with $\mathbf{Q}_{ij} = q(\boldsymbol{x}_i, t_j)$ and a time constant front function f, the approach decomposes the data with help of the nonlinear mapping

$$q(\boldsymbol{x}, t) \approx \tilde{q}(\boldsymbol{x}, t) = f(\phi(\boldsymbol{x}, t)) \quad \text{s. t.} \quad \phi(\boldsymbol{x}, t) = \sum_{k=1}^{r} a_k(t)\psi_k(\boldsymbol{x}), \quad r \ll N_t \tag{7}$$

and a low rank field $\mathbf{\Phi}_{ij} = \phi(\boldsymbol{x}_i, t_j)$, that allows to embed the local one-dimensional front movement into a d-dimensional transport. Since the transport is only parameterized locally, changes in the topology of the front surface can be captured. The profile of the wave f can be analytically computed with help of perturbation theory after transforming (3) into the co-moving frame (see for example [69]) or by fitting the front profile [4]. Unlike (4) the ansatz (7) drops the time dependence of the front function since it can be attributed to ϕ. Note, that additionally to the truncation error $\mathcal{R} = \hat{\phi} - \phi$ the approach (7) introduces another error $\Delta f = \hat{f} - f$ (see for further details [4]). Thus even for vanishing truncation error the overall error is limited by the so-called approximation error Δf of the non-linear function, which is known from machine learning (see Section 5.2 in [70]). This error is caused by the choice of the unknown function class, that is describing the front.

The following problem formalizes finding a suitable $\hat{\phi}$ (encoding transport including change) of Fig. 1 for given snapshots of q and prescribed front shape f. The (non-unique) field ϕ would be in the simplest transport of Fig. 1 a linear function with its offset changing in time, which is a rank two structure. The decomposition goal is formulated as an optimization problem.

Problem 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 nonlinear smooth monotone increasing function $f : \mathbb{R} \to [0, 1]$, find a rank r matrix $\mathbf{\Phi} \in \mathbb{R}^{M \times N_t}$, such that the error $\|\mathbf{Q} - \tilde{\mathbf{Q}}\|_{\mathrm{F}}^2$ for $\tilde{\mathbf{Q}}_{ij} = f(\mathbf{\Phi}_{ij})$ is minimized.

Remark 1 In Problem 1 we assume that the data $\mathbf{Q}$ has been rescaled, which is a common practice in machine learning to help the convergence of the optimization algorithm (see Section 25.2 in [70]).

Two possible algorithms that solve the optimization problem 1 are provided in the following sections.

2.2 Front Transport Reduction via Iterative Thresholding of Singular Values

A simple iterative algorithm to determine the auxiliary field $\mathbf{\Phi} \in \mathbb{R}^{M \times N_t}$ of the front transport reduction Problem 1 is stated in Algorithm 1.

Our algorithm is constructed by combining a gradient descent step (line 4) to minimize $\|\mathbf{Q} - \tilde{\mathbf{Q}}\|_{\mathrm{F}}^2$, together with a rank-$r$ projection step of $\mathbf{\Phi}$ (line 5). In the gradient descent step, the FTR residual

$$\mathcal{L}_{\mathrm{FTR}}(\mathbf{\Phi}) = \frac{1}{2}\|\mathbf{Q} - \tilde{\mathbf{Q}}\|_{\mathrm{F}}^2 \quad \text{with} \quad \tilde{\mathbf{Q}} = f(\mathbf{\Phi}) \tag{8}$$

is minimized in 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 and bounded, it is sufficient to replace $D_{\mathbf{\Phi}}\mathcal{L}_{\mathrm{FTR}}$ by $\mathbf{R} = f(\mathbf{\Phi}) - \mathbf{Q}$ in line 4. Neglecting $f'(\mathbf{\Phi})$ in the gradient prevents a dying gradient for points where $f'(\Phi) \to 0$, i.e. $\left|\mathbf{\Phi}_{ij}\right| \gg 0$.

Note that replacing the simple gradient descent step by a quasi-Newton method or a line search would not affect the convergence rate, since it is followed by a projection step (line 5), which is likely to destroy the possible larger step of a more sophisticated method.

Algorithm 1 FTR as iterative thresholding

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

The computational costs of Algorithm 1 scale with the complexity of the singular value decomposition (SVD). For large systems it can be advantageous to use randomized- or wavelet techniques [71, 72] to compute the SVD. Different initialization of $\boldsymbol{\Phi}^0$ can be used, which may increase the convergence of the algorithm. For example, $\boldsymbol{\Phi}^0$ might be initialized with help of a signed distance function since it was found to be a good candidate for a low-rank decomposition [4]. Other possibilities include using a rank r approximation of $\boldsymbol{\Phi}^0 = f^{-1}(\mathbf{Q})$.

2.3 Front Transport Reduction via Neural Autoencoder Networks

Another way to solve the optimization problem 1 is with the help of neural autoencoder networks, which are commonly used in dimensionality reduction [73]. For a general introduction to neural autoencoder networks, we refer to [74]. Here, we briefly explain the concept and the specifications of our network.

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

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

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

The composition of the two parts

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

defines the autoencoder. The task of the optimization procedure is now to determine $g_{\mathrm{dec}}, g_{\mathrm{enc}}$, such that the reconstruction error over the training data $\mathbf{Q} = [\boldsymbol{q}_1, \dots, \boldsymbol{q}_{N_t}]$:

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

is minimized. After the network has been trained, the reduction is achieved as the dimension $r \ll M$ of the latent variables $\boldsymbol{a}_i = g_{\mathrm{enc}}(\boldsymbol{q}_i) \in \mathbb{R}^r$ is much smaller than the input dimension

M. Therefore, the decoder $\boldsymbol{q}_i \approx g_{\mathrm{dec}}(\boldsymbol{a}_i)$ represents a reduced map of the high dimensional data contained in the columns of $\mathbf{Q}$.

In the training procedure, the functions $g_{\mathrm{enc}}, g_{\mathrm{dec}}$ are determined by trainable parameters of the network, called weights and biases. The networks are constructed by a composition of layers, $g_{\mathrm{enc}} = L_1 \circ L_2 \circ \cdots \circ L_N$. 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 nonlinear function h_n. The choice of the input and output dimension i, o in each layer, the activation function and the number of layers is called architecture of the network.

As the FTR-autoencoder network (FTR-NN) should implement the structure motivated in Problem 1, we choose a special architecture. It consists of a single layer decoder, without bias

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

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 $\boldsymbol{a}_i = g_{\mathrm{enc}}(\boldsymbol{q}_i)$ correspond to the columns of the discrete transport field $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_{N_t}]$. 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.

The encoder network consists of four convolutional layers, each followed by an exponential linear unit (ELU) and a batch normalization layer [75]. After flattening the output, the convolutional layers are followed by two linear layers, where the first one is again followed by an ELU activation and a batch normalization layer. We apply a stride of two in all convolutional layers after the first, to downsample the spatial resolution of the input data. Further details of the architecture and training procedure can be found in Appendix A.

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 $\boldsymbol{\Psi} \in \mathbb{R}^{M \times r}$

$$\mathcal{L}_{\mathrm{smooth}} = \lambda_{\mathrm{smooth}} \sum_{n=1}^{r} \frac{\|\mathbf{D}\boldsymbol{\psi}_n\|_{\mathrm{F}}^2}{\|\boldsymbol{\psi}_n\|_{\mathrm{F}}^2}. \tag{9}$$

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 $\boldsymbol{\Psi}$. For the examples in this manuscript $\lambda_{\mathrm{smooth}} = 10^{-7}$, was found to be optimal. 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 $\boldsymbol{\Phi}_{ij} = \phi(\boldsymbol{x}_i, t_j)$, which is assumed to be smooth.

2.4 Synthetic Examples

In this subsection, we provide two synthetic examples. The first example illustrates the application of the FTR to linear advection and compares the two decomposition methods outlined above with previous results [4]. The second example addresses topology changing fronts. The reduced mappings that are used for the comparisons are summarized in Table 1. Note that the number of parameters that are determined during the offline stage is different from the number of time/parameter dependent variables $\boldsymbol{a}(t) \in \mathbb{R}^r$. We refer to the latter as degrees of freedom in the following.

Table 1 Summary of the different reduced mappings and their number of degrees of freedom (DOF)

Methods	Reduced mapping	# DOF (denoted by r)
POD	$\boldsymbol{q}(t) = \boldsymbol{\Psi}^{\text{POD}}\boldsymbol{a}(t)$, $\boldsymbol{\Psi}^{\text{POD}} \in \mathbb{R}^{M\times r}$	Truncation rank of SVD of $\mathbf{Q}$
FTR	$\boldsymbol{q}(t) = f(\boldsymbol{\phi}(t))$, $\boldsymbol{\phi}(t) = \boldsymbol{\Psi}^{\text{FTR}}\boldsymbol{a}(t)$, $\boldsymbol{\psi}^{FTR} \in \mathbb{R}^{M\times r}$	Truncation rank of $\boldsymbol{\Phi}$ used in Algorithm 1
FTR–NN	$\boldsymbol{q}(t) = f(\boldsymbol{\phi}(t))$, $\boldsymbol{\phi}(t) = \boldsymbol{\Psi}^{\text{FTR-NN}}\boldsymbol{a}(t)$, $\boldsymbol{\Psi}^{\text{FTR-NN}} \in \mathbb{R}^{M\times r}$	Dimension of the latent space $\boldsymbol{a} \in \mathbb{R}^r$
NN	$\boldsymbol{q}(t) = g_{\text{dec}}(\boldsymbol{a}(t))$	Dimension of the latent space $\boldsymbol{a} \in \mathbb{R}^r$

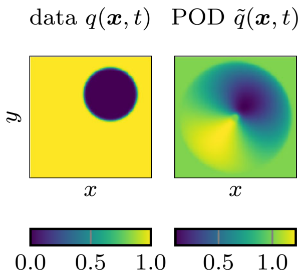


Fig. 2 Data at time $t = 0.11$ and its approximation with the Proper Orthogonal Decomposition (POD) using $r = 3$ modes

2.4.1 Linear Advection of a Disc

The first synthetic example is taken from [4]. It illustrates the idea of the FTR in the pure advection case, without any topological change. The example parameterizes a disc of radius $R = 0.15L$, which is moving in a circle:

$$q(\boldsymbol{x},t) = f(\phi(\boldsymbol{x},t)) \quad \text{and} \quad \phi(\boldsymbol{x},t) = \frac{1}{2R}(\|\boldsymbol{x} - \boldsymbol{x}_0(t)\|_2^2 - R^2) \tag{10}$$

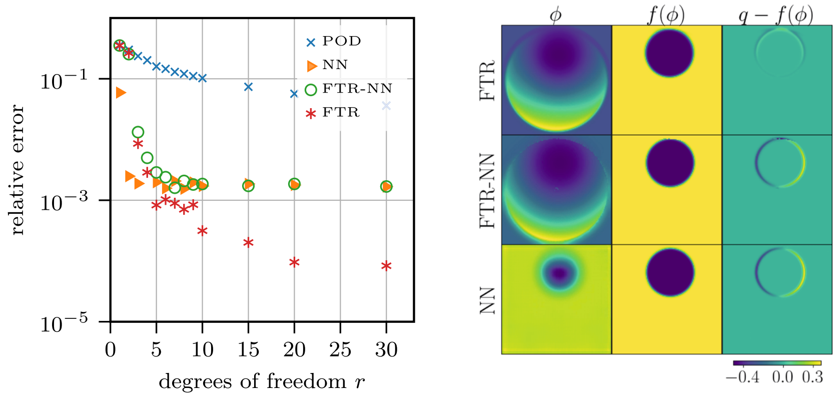
$$\text{with circle center } \boldsymbol{x}_0(t) = L\begin{pmatrix}0.5 + 1/4\cos(2\pi t)\\ 0.5 + 1/4\sin(2\pi t)\end{pmatrix}. \tag{11}$$

The snapshot data $q(\boldsymbol{x},t)$ is generated with the specific level set field $\phi(\boldsymbol{x},t)$ defined in (10), which is zero at the outer radius of the disc, 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. 2. As the authors of [4] have already pointed out, for this example $\phi(\boldsymbol{x},t) = \sum_{k=1}^{3} a_k(t)\psi_k(\boldsymbol{x})$ can be parameterized by only three functions and is therefore of low rank, even if the field $q(\boldsymbol{x},t)$ is not. The basis functions ψ_1, ψ_2, ψ_3 are shown in the top row of Fig. 4a. They can be interpreted as a quadratic basis function $\psi_1(x,y) = (x-0.5L)^2 + (y-0.5L)^2 - R^2 + L^2/4^2$ 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 $\psi_2(x,y) = (x-0.5L)$, $\psi_3(x,y) = (y-0.5L)$ for the shift in x/y-direction with $a_2(t) \sim \cos(2\pi t)$, $a_3(t) \sim \sin(2\pi t)$. Note, that the arrows in Fig. 4a indicate $\nabla\psi_2(x,y)$, $\nabla\psi_3(x,y)$ the direction of the shift.

To show that the singular value thresholding Algorithm 1 (FTR) and the neural network approach Sect. 2.3 (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 was split into a training and test set, where every

Table 2 Parameters of Algorithm 1 for the moving disc example

Name	Value
Number of snapshots	$N_t = 100$
Front function	$f(x) = \text{sigmoid}(x)$
Number of iterations	4000
Step size	$\tau = 0.3$



a) Quantitative error **b)** Qualitative error $r = 3$ degrees of freedom

Fig. 3 Comparison of POD, FTR, FTR-NN and the symmetrical autoencoder structure labeled with NN. **a** compares the relative errors in the Frobenius norm for different degrees of freedom. **b** 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

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 1 using the test samples. All parameters for Algorithm 1 are stated in Table 2. The results are visualized in Figs. 3 and 4. In Fig. 3 we compare the results of both FTR algorithms (FTR, FTR-NN) and a simple symmetrical autoencoder structure, labeled with NN (for details see Appendix A). The NN decoder attempts to implement the encoder in an inverse manner (see details Appendix A), which is a common practice in dimension reduction. The relative errors in the Frobenius norm are shown in Fig. 3a. 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 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 [76].

Furthermore, it is important to note that the FTR-thresholding algorithm outperforms both networks, when increasing the number of degrees of freedom, for this special example.

Remark 2 We will briefly make reference to other transport compensating methods such as the shifted POD (sPOD) [16] or transport reversal method [23]. In these, the movement of the disc is described by a two-dimensional shift transformation in x and y, denoted by

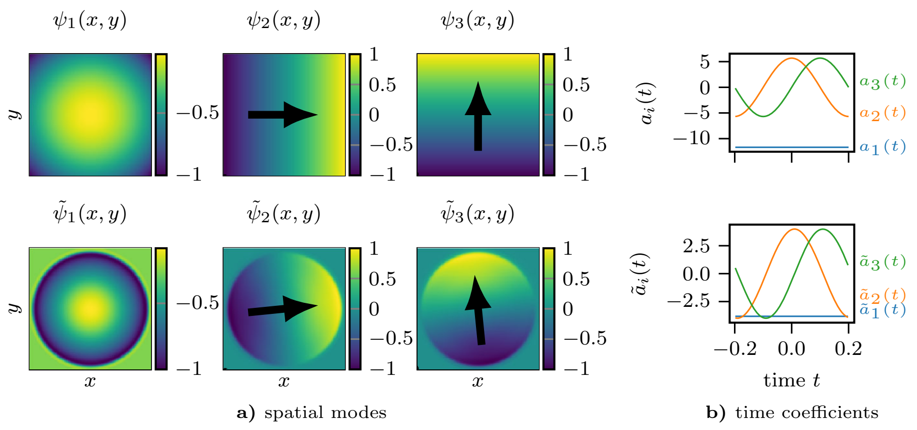


Fig. 4 Visualization of the FTR transport field $\phi(x,t)=\sum_{i=0}^{3} a_i(t)\psi_i(x,y)$ for the disc moving in a circle (see Eqs. (11), (10)). Displayed are the expected spatial modes ψ_i (Fig. 4a), their temporal amplitudes a_i (Fig. 4b) and FTR approximation $\tilde\psi_i, \tilde a_i, i = 1, 2, 3$. The modes have been rescaled by their maximal absolute value, for better comparison. The arrows in Fig. 4a indicate the direction of the shift. They are computed from the spatial mean of $\nabla\psi_2(x,y)$, $\nabla\psi_3(x,y)$. The corresponding amplitudes a_2, a_3 parameterize the circular movement in time

$\mathcal{T}^{\boldsymbol{\Delta}(t)}[q(x,y,t)] = q(x-\Delta_1(t), y-\Delta_2(t), t)$, and one mode $\psi_1^{\mathrm{sPOD}}(x,y) = q(x,y,0)$ for representing the structure of the disc in the co-moving frame. Thus, the state $q(x,y,t) = \mathcal{T}^{\boldsymbol{\Delta}(t)}[\psi_1^{\mathrm{sPOD}}(x,y)a_1^{\mathrm{sPOD}}(t)]$ is described by $r = 3$ DOF $(\Delta_1(t), \Delta_2(t), a_1^{\mathrm{sPOD}}(t))$, which is in accordance with the presented FTR approach for this specific example.

For qualitative comparison, Fig. 3b 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. 2 to the FTR results shows, that the typical stair casing 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 $\lambda_{\mathrm{smooth}} = 10^{-7}$ for regularizing the smoothness of ϕ at the output of the FTR-NN and NN decoder. As visualized in Appendix A Fig. 14, for larger smoothness parameter $\lambda_{\mathrm{smooth}} > 10^{-7}$ the transport field of the FTR-NN is smoothly continued at areas of no information (no transport), but the additional constraint (9) 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 compares the fields ψ_1, ψ_2, ψ_3, obtained by contemplation, 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 disc and two modes that parameterize 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. 4a) 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 can not in-paint ϕ in areas of no transport. This explains that the modes in Fig. 4 are zero outside the circle.

Comparing the CPU time of the different algorithms is difficult due to various factors that affect the convergence of the algorithms, especially for neural networks. These factors include initialization, learning rate, architecture, and others. Moreover, algorithms are typically run

Table 3 Average CPU/GPU time required to generate the reduced mapping of the moving disc with 3 degrees of freedom using either Algorithm 1 (FTR) or network training (FTR-NN, NN) was evaluated

Methods	GPU-time per iteration (s)	CPU-time per iteration (s)	# iterations
FTR	–	0.7	3401
FTR-NN	0.06	0.2	15,499
NN	0.16	1.3	10,499

Performance tests were conducted on `11th Gen Intel(R) Core(TM) i7-11850H` CPUs (8 processing units) and a `NVIDIA GeForce RTX 3070 Laptop GPU` with 8GB of RAM. The number of iterations was counted until the algorithms reached an error below 1%

on different hardware (CPU/GPU) which can further complicate comparisons. Despite these caveats, what we can say is that when aiming for a comparable level of accuracy, Algorithm 1 requires less time than neural networks when compared on a CPU. This is likely due to the fact that neural networks involve optimizing a larger number of parameters during the minimization process. To illustrate this, we provide the required CPU/GPU time needed by a modern laptop to reach an error below 1% using $r = 3$ degrees of freedom in the resulting reduced mapping in Table 3.

2.4.2 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

$$\varphi(\boldsymbol{x}, t) = \sum_{k=1}^{3} -A_k e^{-\sigma_k r_k} - t, \quad r_k = \|\boldsymbol{x} - \boldsymbol{x}_i\|_2, \tag{12}$$

which we try to approximate. The front f is chosen as above. The level-set field is sampled equidistantly using 256×265 grid points in $[0, 10]^2$, with $(A_1, A_2, A_3) = (1, 1.4, 1.2)$, $(\sigma_1, \sigma_2, \sigma_3) = (0.1, 0.3, 0.5)$ 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 (12). As above, we split the samples into a test and training 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. 5, together with the resulting snapshots of q as a color plot. The intersection of ϕ with the zero plane parameterizes 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 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 $t = 0.4$. This property makes it difficult for most dimension reduction methods, which can handle transports because these rely on one-to-one mappings between different time or parameter instances.

As presented in Fig. 6, 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 (12). The networks approximation errors behave as in the case of the moving disc. 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. Topology

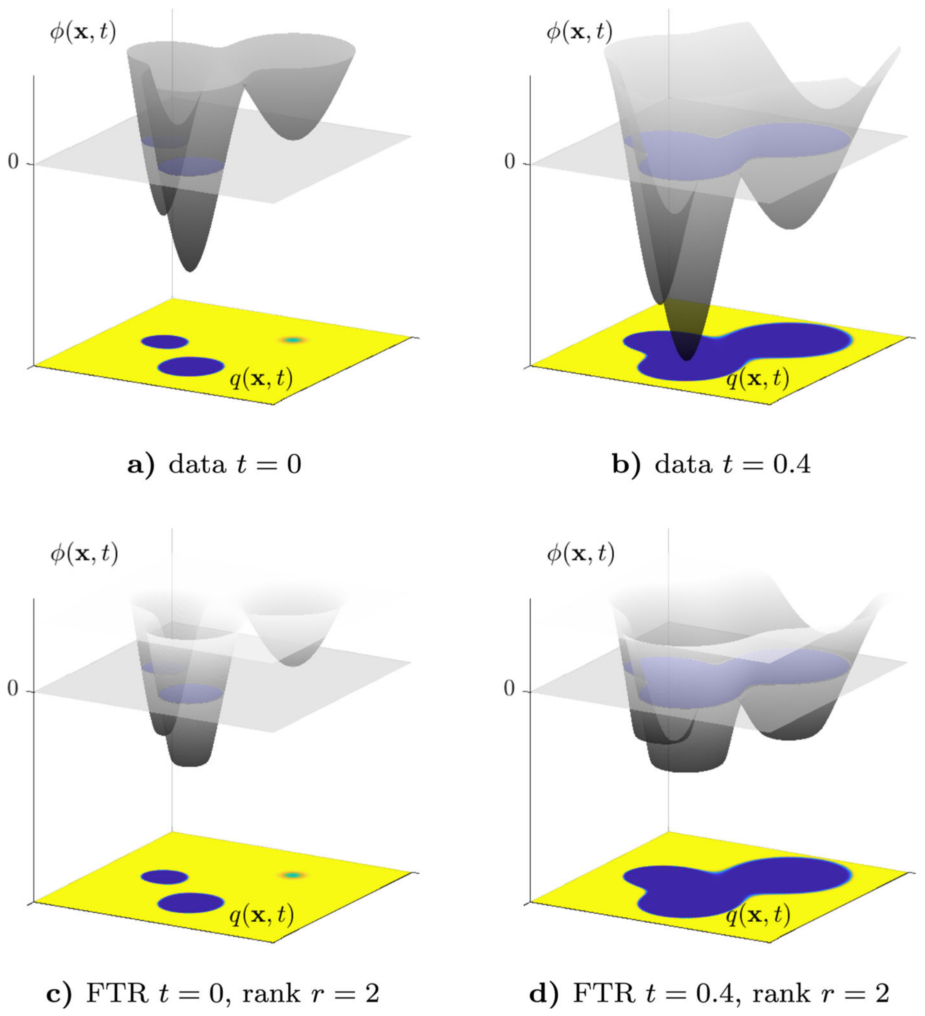


Fig. 5 Graph of the auxiliary field ϕ (12) in **a**, **b** and its FTR approximation in **c**, **d**. The resulting snapshots $q = f(\phi)$ are shown as a color plot in the xy plane. The intersection with the zero level is visualized (Color figure online)

changes of the zero level-set are nicely recovered as is illustrated in Fig. 5c and d, since the FTR approach can recover the initial auxiliary field ϕ in the regions of transport.

2.5 Application to Advection–Reaction–Diffusion Systems

To motivate the FTR approach for more complex examples, we introduce the advection–diffusion–reaction PDE with a KPP reaction term

$$\begin{cases} \partial_t q = -\boldsymbol{u} \cdot \nabla q + \kappa \nabla^2 q - \mu q^2 (q-1) \\ q(\boldsymbol{x}, 0) = q_0(\boldsymbol{x}) \end{cases}, \tag{13}$$

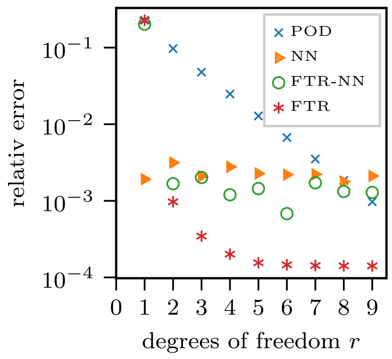


Fig. 6 Comparison of the relative errors for the advection example with topology change using the proper orthogonal decomposition (POD), the FTR iterative thresholding algorithm (FTR), the FTR autoencoder structure (FTR-NN) and a standard autoencoder (NN)

Table 4 Parameters of the 2D ARD simulation of Sect. 2.5

Name	Value
FOM—parameters	
Simulation time	$T = 3$
Grid resolution	$M = 512 \times 512$
Domain size	$L = 1$
Diffusion constant	$\kappa = 10^{-3}$
Reaction constant	$\mu = 10$
Advection of vortex-pair	$c = 10$
ROM—parameters	
Number of snapshots	$N_t = 200$
Front function	$f(x) = \text{sigmoid}(x)$
Number of iterations	5000
Step size Algorithm 1	$\tau = 1$

on a square, two-dimensional domain $\Omega = [0, L]^2$ with periodic boundary conditions and time interval $[0, T]$. The PDE is discretized in space using 6th-order central finite differences, and in time with an explicit Runge–Kutta method of 5th(4th) order [77]. In the following, we refer to the discretized system as the full order model (FOM). All simulation parameters are listed in Table 4.

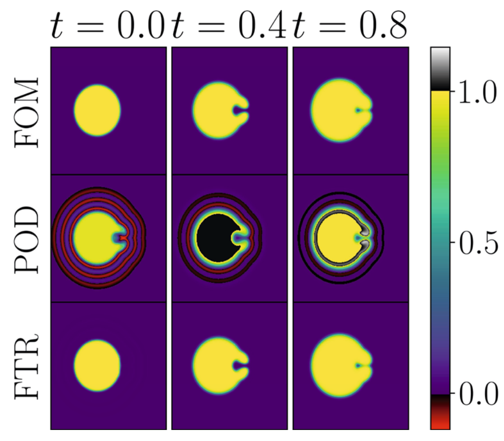
For our test case we choose a velocity field inspired by the vortex pair example in [78]. Therefore $\boldsymbol{u} = \nabla \times \omega$ is expressed in terms of the vorticity

$$\omega(\boldsymbol{x}, t) = \omega_0 e^{-t^2/\tau^2}(e^{-r_1^2(t)/r_0^2} + e^{-r_2^2(t)/r_0^2}), \quad r_i(t) = \|\boldsymbol{x} - \boldsymbol{x}_i(t)\|_2, \tag{14}$$

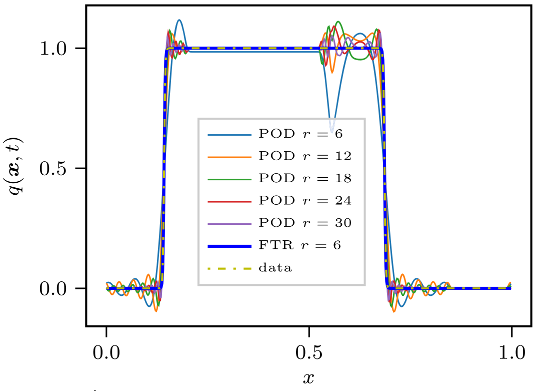
which parameterizes a moving vortex pair $\boldsymbol{x}_1 = L(0.6 - ct, 0.49)$, $\boldsymbol{x}_2 = L(0.6 - ct, 0.51)$, $r_0 = 5 \times 10^{-4}$ with an initial amplitude $\omega_0 = 10^3$ decaying slowly in time (decay constant $\tau = 3T$). The initial distribution of the reactant q is given by:

$$q_0(x, y) = \begin{cases} 1, & \text{for } \sqrt{(x - 0.4L)^2 + (y - 0.5L)^2} > 0.2L, \\ 0, & \text{else.} \end{cases} \tag{15}$$

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 inter- actions. During the simulation, the synthetic vortex pair (14) moves towards burning gas and mixes unburned



a) Colorplot of the reconstruction



b) Horizontal cut at $t = 0.4$ and $y = 0.5$

Fig. 7 Qualitative comparison of the reconstruction errors of the 2D ARD system (13). **a** Shows a colorplot 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. **b** Shows a profile of the solution and the different approximations along the horizontal line with at $y = 0.5$ and $t = 0.4$ (Color figure online)

($q = 1$) 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 front. We compare the FTR decomposition of the data with the POD in Fig. 7. The color plot in Fig. 7a shows the time evolution of the FOM for $t = 0.0, 0.4, 0.8$ in the top row. 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. The oscillations violate the initial range of values $0 \leq q \leq 1$, which is depicted as red and black areas in Fig. 7b. Even for a large number of modes ($r = 30$) the oscillations are still visible, as shown by the profile plot in Fig. 7b. Therefore, preservation of physical structure cannot be expected, which may lead to instabilities in the resulting ROM [56, 57, 79]. Here, the FTR approach gives much better results, restricting the approximation on the initial range of values due to the range of the sigmoid function $f(x) \in [0, 1]$.

3 Galerkin and Data-Driven Models for Moving Fronts

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 model generated by the FTR algorithm in Sect. 2.2 is nonlinear, which poses additional challenges for the online stage, to predict and interpolate new system states. This section is therefore dedicated to online prediction methods. In Sect. 3.1 we use a non-intrusive, i.e. equation free, approach of [59] and introduce an intrusive approach, the hyper-reduced Galerkin method in Sect. 3.2 for 1D and 2D ARD systems.

3.1 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 [34], SINDy [80, 81] or Fourier-Koopman forecasting [59], have become prominent. Although the methods make specific assumptions on 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 space is a nonlinear manifold, which makes any Galerkin-projection approach more complex and costly, as is shown in the next section.

Following the approach of [59], we can derive new system states and extrapolate in time with help of the Fourier-Koopman framework implemented in [82]. The Fourier-Koopman framework imposes the assumption that the reduced state $\boldsymbol{a}(t) \in \mathbb{R}^r$ is quasi-periodic in t and can be thus parameterized by:

$$\boldsymbol{a}(t) = \mathbf{A}\boldsymbol{\Omega}(t) \quad \text{with} \quad \boldsymbol{\Omega}(t) = \begin{pmatrix} \cos(\boldsymbol{\omega} t) \\ \sin(\boldsymbol{\omega} t) \end{pmatrix}. \tag{16}$$

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

$$\min_{\boldsymbol{\omega},\mathbf{A}} \sum_{n=0}^{N-1} \|\boldsymbol{a}(t_n) - \mathbf{A}\boldsymbol{\Omega}(t_n)\|_2^2, \tag{17}$$

in a smart way [59]. Since the dynamical system presented in Sect. 2.4.1 is quasi-periodic, we can apply the method to the FTR decomposition

$$\boldsymbol{q}(t) \approx \tilde{\boldsymbol{q}}(t) = f(\boldsymbol{\Psi}\boldsymbol{a}(t)) \tag{18}$$

using the basis functions $\boldsymbol{\Psi} = [\tilde{\boldsymbol{\psi}}_1, \tilde{\boldsymbol{\psi}}_2, \tilde{\boldsymbol{\psi}}_3]$, shown in Fig. 4 together with the amplitudes $\boldsymbol{a}(t) = (a_1(t), a_2(t), a_3(t))$ at the sampled time points $\{t_n = n\Delta t \mid n = 0, \dots, N-1\}$

Remark 3 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. 8. This can be done similarly for the POD.

From the sampled data we compute $\mathbf{A}, \boldsymbol{\omega}$. To test the resulting model $\tilde{\boldsymbol{q}}(t) = f(\boldsymbol{\Psi}\mathbf{A}\boldsymbol{\Omega}(t))$ we evaluate it at time instances $t_{n+1/2} = (n+1/2)\Delta t$ for $n = 0, \dots, 2N-1$, that have been not sampled during the offline phase. Similarly, we can derive an approximation with the POD. Both results are compared in Fig. 8. Furthermore, the online-prediction error is stated for $r = 2, 4, 6, 8, 10, 12, 15$ in Table 6.

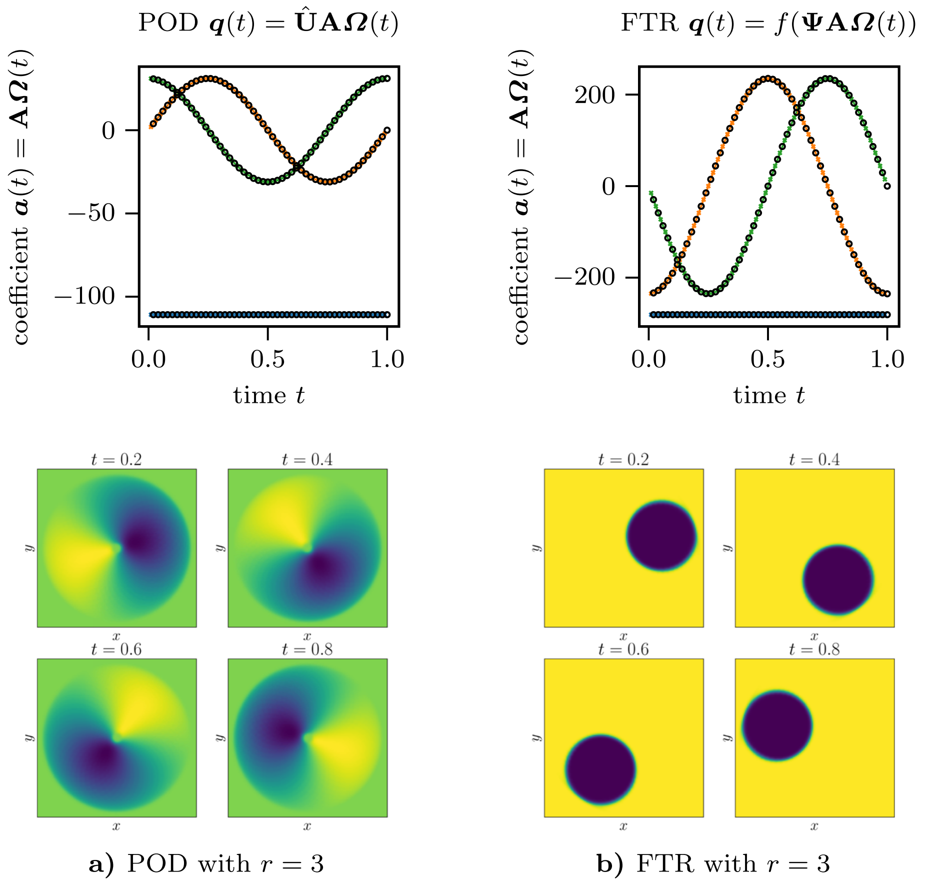


Fig. 8 Predictions using Fourier–Koopman forecasting with three POD modes (**a**) and three FTR modes (**b**). The black circles (∘) in the upper row indicate the predictions of the amplitudes $\boldsymbol{a}(t) = (a_1(t), a_2(t), a_3(t)) \widehat{=}$ (—, —, —) and the colored crosses mark the training samples. In the lower row, we show the corresponding snapshots at selected time instances $t = 0.2, 0.4, 0.6, 0.8$ (Color figure online)

Note that after solving (17) in the offline stage, the computational effort is reduced to the evaluation of $\tilde{\boldsymbol{q}}(t) = f(\boldsymbol{\Psi}\mathbf{A}\boldsymbol{\Omega}(t))$, which only takes milliseconds.

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 [83, 84]. The snapshots are generated with a customized, weakly compressible version of `rhoReactionFOAM` from the `OpenFOAM` software package (see [84, 85]). 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. 9. The data set.[1] 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 ($t_n = 2\Delta tn$) and test samples $t_{n+1/2} = (2n + 1)\Delta t$. While we use the training samples to generate the reduced model using Algorithm 1 with the parameters that are summarized in Table 2, the test samples are used to calculate the relative errors stated in Table 5. The flame pinch-off is not a special

[1] Data is available for download https://doi.org/10.5281/zenodo.7681268.

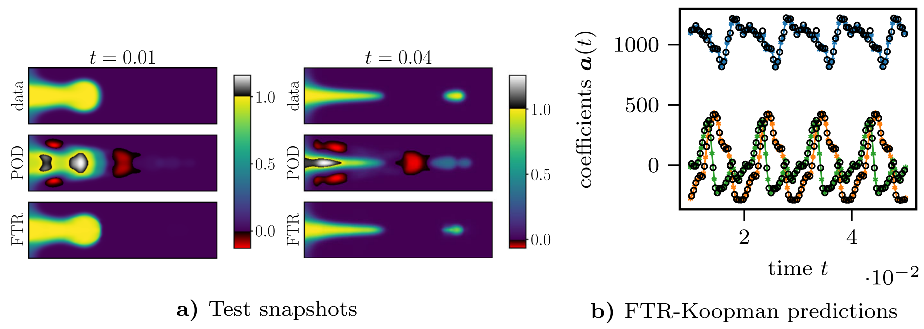


Fig. 9 Online predictions of the Bunsen flame example. **a** compares the test data in the top row with the FTR-Koopman and POD-Koopman results using $r = 8$ degrees of freedom for $t = 0.01$ and 0.04. The snapshots show how a burning fuel pocket is detached from the flame at $t = 0.04$ causing a change in the topology of the contour line of the front. **b** visualizes the Fourier–Koopman predictions (∘) for $\boldsymbol{a}(t) = (a_1(t), a_2(t), a_3(t)) \widehat{=}$ (—•—, —•—, —•—)

Table 5 Relative errors $\sum_{n=0}^{2N-1} \|\boldsymbol{q}(t_{n+1/2}) - \tilde{\boldsymbol{q}}(t_{n+1/2})\|_2^2 / \sum_{n=0}^{2N-1} \|\boldsymbol{q}(t_{n+1/2})\|_2^2$ using Fourier–Koopman (online) with POD and FTR on the Bunsen flame data

	FTR		POD	
Rank r	Offline	Online	Offline	Online
2	4.3e−01	4.3e−01	4.3e−01	4.3e−01
4	7.5e−02	1.2e−01	3.0e−01	3.1e−01
6	2.6e−02	8.9e−02	2.3e−01	2.3e−01
8	1.4e−02	7.0e−02	1.7e−01	1.8e−01
10	9.2e−03	7.2e−02	1.3e−01	1.6e−01
12	5.4e−03	7.5e−02	9.5e−02	1.5e−01
15	3.1e−03	6.9e−02	5.3e−02	1.4e−01

case in combustion systems, but it poses challenges to model order reduction methods, as described above. Figure 9 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.

3.2 Manifold Galerkin Methods

After discretizing the ARD system (3) in space, we obtain an ODE system of the form

$$\text{(FOM)} \quad \begin{cases} \dot{\boldsymbol{q}}(t, \mu) = \boldsymbol{F}(\boldsymbol{q}, t, \mu) \\ \boldsymbol{q}(0) = \boldsymbol{q}_0, \end{cases} \tag{19}$$

with the discretized non-linear right hand side (RHS) $\boldsymbol{F} : \mathbb{R}^M \times \mathbb{R} \times \mathbb{R}^+$. Here, the parameter $\mu > 0$ denotes the reaction constant. Using a reduced mapping

$$g : \mathbb{R}^r \to \mathbb{R}^M : \boldsymbol{a} \mapsto g(\boldsymbol{a}), \quad \text{with Jacobian} \quad \mathbf{J}_g(\boldsymbol{a}) = \left(\frac{\partial g_i}{\partial a_j}(\boldsymbol{a}) \right)_{\substack{i=1,\dots,M \\ j=1,\dots,r}} \tag{20}$$

Table 6 Relative error $\sum_{n=0}^{2N-1} \|\boldsymbol{q}(t_{n+1/2}) - \tilde{\boldsymbol{q}}(t_{n+1/2})\|_2^2 / \sum_{n=0}^{2N-1} \|\boldsymbol{q}(t_{n+1/2})\|_2^2$ using Fourier–Koopman (online) with POD and FTR on the moving disc data

	FTR		POD	
Rank r	Offline	Online	Offline	Online
2	2.7e−01	2.7e−01	3.0e−01	8.5e−01
4	2.9e−03	2.9e−03	2.0e−01	2.0e−01
6	1.0e−03	1.1e−03	1.5e−01	1.5e−01
8	7.1e−04	7.6e−04	1.2e−01	1.2e−01
10	3.1e−04	3.4e−04	1.0e−01	1.0e−01
12	2.5e−03	2.8e−03	8.8e−02	8.8e−02
15	2.0e−04	2.4e−04	7.4e−02	7.4e−02

as approximation $\boldsymbol{q} \approx \tilde{\boldsymbol{q}} = g(\boldsymbol{a})$ of the data and plugging it into (19) yields a reduced model:

$$\text{(ROM)} \quad \begin{cases} \dot{\boldsymbol{a}}(t,\mu) = \underset{\dot{\boldsymbol{a}}\in\mathbb{R}^r}{\arg\min} \|\mathbf{J}_g(\boldsymbol{a})\dot{\boldsymbol{a}}(t,\mu) - \boldsymbol{F}(g(\boldsymbol{a}),t,\mu)\|_2^2 & (21) \\ \boldsymbol{a}(0,\mu) = \underset{\boldsymbol{a}\in\mathbb{R}^r}{\arg\min} \|\boldsymbol{q}_0 - g(\boldsymbol{a})\|_2^2. & (22) \end{cases}$$

Minimizing the continuous time residual (21), yields the optimality condition:

$$0 = \frac{\mathrm{d}}{\mathrm{d}\dot{\boldsymbol{a}}} \|\mathbf{J}_g(\boldsymbol{a})\dot{\boldsymbol{a}} - \boldsymbol{F}(g(\boldsymbol{a}),t,\mu)\|_2^2 \tag{23}$$

$$= 2\mathbf{J}_g(\boldsymbol{a})^T \mathbf{J}_g(\boldsymbol{a})\dot{\boldsymbol{a}} - 2\mathbf{J}_g(\boldsymbol{a})^T \boldsymbol{F}(g(\boldsymbol{a}),t,\mu), \tag{24}$$

which is uniquely solved by

$$\dot{\boldsymbol{a}} = \mathbf{J}_g^+(\boldsymbol{a})\boldsymbol{F}(g(\boldsymbol{a}),t,\mu), \tag{25}$$

if the Jacobin has full column rank [35]. Here, $\mathbf{J}_g^+$ is the Moore-Penrose pseudo inverse of $\mathbf{J}_g$. In practice the inverse is never formed, but (24) is solved directly. Unfortunately, the calculation of (24) still depends on the dimension of the FOM, since the full RHS has to be evaluated. In order to circumvent this, we apply hyper-reduction explained in the following.

3.3 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 thin band along the front, which is shown in Fig. 10. This band is usually much smaller than the size of the domain or the traveling distance of the front. Hence, the FOM-RHS and its gradient posses only few spatial grid points per time step with non-vanishing support. Therefore, the hyper-reduction methods for nonlinear manifolds [36, 54] cannot be applied. For example, the extended-ECSW scheme proposed by [54], or the gappy-POD based GNAT procedure [52] first introduced for nonlinear manifolds in [36] cannot be used here, since they preselect a set of sample points, which is fixed for every time step and all $\mu \in \mathcal{P}$. This is supported by the observations made in [55, 57], where an oversampling of the hyper-reduced system is necessary in order to maintain stability of the resulting ROM. In contrast, the FTR-hyper-reduction approach outlined in Algorithm 2 can help to identify the locations of the front to reduce computational complexity, while sustaining an accurate solution and stability. Here, we propose an idea that is similar to the Reduced Integration Domain (RID) method [58] 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, using a threshold search we selected a number of M_p sampling points $\boldsymbol{s} = (s_1, \dots, s_{M_p})$ that are sensitive to the residual (24). This corresponds to replacing the Euclidean by a weighted norm in (24):

$$0 = \frac{\mathrm{d}}{\mathrm{d}\dot{\boldsymbol{a}}} \|\mathbf{J}_g(\boldsymbol{a})\dot{\boldsymbol{a}} - \boldsymbol{F}(g(\boldsymbol{a}), t, \mu)\|^2_{\mathbf{P}^2_{\boldsymbol{a}}} \tag{26}$$

$$= 2\mathbf{J}_g(\boldsymbol{a})^T\mathbf{P}_{\boldsymbol{a}}^T\mathbf{P}_{\boldsymbol{a}}\mathbf{J}_g(\boldsymbol{a})\dot{\boldsymbol{a}} - 2\mathbf{J}_g(\boldsymbol{a})^T\mathbf{P}_{\boldsymbol{a}}^T\mathbf{P}_{\boldsymbol{a}}\boldsymbol{F}(g(\boldsymbol{a}), t, \mu). \tag{27}$$

Each of the M_p selected sample points corresponds to an index $0 \leq s_i \leq M$, which is represented as the s_ith standard basis vector $\boldsymbol{e}_{s_i} \in \mathbb{R}^M$ 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 $\mathbf{P}_{\boldsymbol{a}}\mathbf{J}_g, \mathbf{P}_{\boldsymbol{a}}\boldsymbol{F}$ are only computed at M_p sample points. Note that the stencil size of our finite difference scheme requires computing $f(\boldsymbol{\phi})$ on additional supporting mesh points, contained in $\hat{\mathbf{P}}_{\boldsymbol{a}} \in \mathbb{R}^{\hat{M}_p \times M}$. In practice, $\hat{\mathbf{P}}_{\boldsymbol{a}} f(\boldsymbol{\phi}), \mathbf{P}_{\boldsymbol{a}}\mathbf{J}_g, \mathbf{P}_{\boldsymbol{a}}\boldsymbol{F}$ are not computed as matrix products, but as pointwise evaluations of $f(\boldsymbol{\phi}), \mathbf{J}_g, \boldsymbol{F}$ at the corresponding sample points. To be precise, in line 5 of Algorithm 2 we only iterate over the indices $\boldsymbol{s} \in \mathbb{R}^{M_p}$, when computing $\boldsymbol{b} \in \mathbb{R}^{M_p}$. Similarly, we compute the Jacobian in line 6 of Algorithm 2.

Algorithm 2 Hyper-reduced RHS

```
Require: dynamic inputs (∘), static inputs (+)
    ∘ current reduced state $\boldsymbol{a}^k \in \mathbb{R}^r$,
    ∘ time $t^k$ at time step $k$
    + FTR modes $\boldsymbol{\Psi} \in \mathbb{R}^{M \times r}$,
    + RHS function $\boldsymbol{F}(\boldsymbol{q}, t^k, \mu)$,
    + number of sampling points $M_p$
    + stencil support mesh indices $\{\hat{S}_i, i = 1, \dots, M\}$
      - $\hat{S}_i$ is a list of indices that are required for computing differential operators $\nabla$ and $\nabla^2$ at the sample point $\boldsymbol{x}_i$
1: function hyp_rhs($\boldsymbol{a}^k$, $\boldsymbol{F}(\boldsymbol{q}, t^k, \mu)$, $\hat{S}$, $\boldsymbol{\Psi}$)
2: compute current level set function: $\boldsymbol{\phi}^k = \boldsymbol{\Psi}\boldsymbol{a}^k \in \mathbb{R}^M$
3: find the indices $\boldsymbol{s} = (s_1, \dots, s_{M_p})$ of the first $M_p$ components of $\boldsymbol{\phi}^k = (\phi_1^k, \dots, \phi_M^k)$,
   with smallest absolute value:
   $\boldsymbol{s} = \texttt{argmin}(|\boldsymbol{\phi}^k|, M_p) \in \mathbb{R}^{M_p}$ with $\mathbf{P}_{\boldsymbol{a}} = [\boldsymbol{e}_{s_1}, \dots, \boldsymbol{e}_{s_{M_p}}]^T \in \mathbb{R}^{M_p \times M}$
4: select the stencil points (/indices) $\hat{\boldsymbol{s}}$ around the minimum points $\boldsymbol{s}$ from the pre-computed lists $\{\hat{S}_i, i = 1, \dots, M\}$
   $\hat{\boldsymbol{s}} = \texttt{unique}(\hat{S}_{s_1}, \dots, \hat{S}_{s_{M_p}}) \in \mathbb{R}^{\hat{M}_p}$, $\hat{\mathbf{P}}_{\boldsymbol{a}} = [\boldsymbol{e}_{\hat{s}_1}, \dots, \boldsymbol{e}_{\hat{s}_{\hat{M}_p}}]^T \in \mathbb{R}^{\hat{M}_p \times M}$
5: compute the RHS at the selected sample points:
   $\boldsymbol{b} = \mathbf{P}_a\boldsymbol{F}(f(\hat{\mathbf{P}}_a\boldsymbol{\phi}^k), t^k, \mu)$ with $b \in \mathbb{R}^{M_p}$
6: compute the Jacobian $\mathbf{A} = \mathbf{P}_a\mathbf{J}_g \in \mathbb{R}^{M_p \times r}$ of the mapping at the selected sample points
7: solve $\mathbf{A}\dot{\boldsymbol{a}} = \boldsymbol{b}$
8: return $\dot{\boldsymbol{a}}$
```

In contrast to RID, the selection matrix $\mathbf{P}_{\boldsymbol{a}} : \mathbb{R}^r \to \mathbb{R}^{M_p \times M}$ is dependent on the state $\boldsymbol{a}(t, \mu)$, which evolves over time (see Fig. 10). However, similar to what RID does for

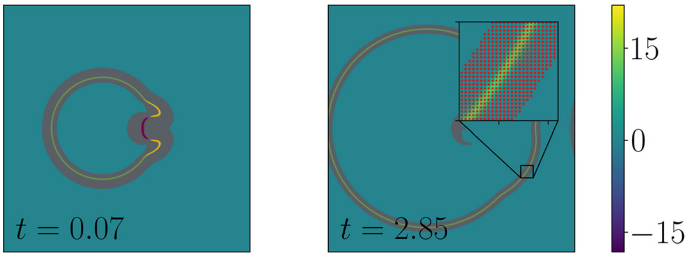


Fig. 10 Color plot of the right hand side $\boldsymbol{F}(\boldsymbol{q}, t, \mu)$ of the 2D advection–reaction–diffusion system (13) for two different time instances $t = 0.07$ (left) and $t = 2.85$ (right) and the corresponding sample points for a sample fraction of $M_p/M = 0.1$. The inset in the right color plot shows a close up of the location of the front (Color figure online)

external forces, we have to add nodes, i.e. sample points, at which the right hand side does not vanish. Thus, if all sample points are included for which the RHS does not vanish, we have an exact equality between the different norms:

$$\|\mathbf{J}_g(\boldsymbol{a})\dot{\boldsymbol{a}} - \boldsymbol{F}(g(\boldsymbol{a}), t, \mu)\|_2^2 = \|\mathbf{J}_g(\boldsymbol{a})\dot{\boldsymbol{a}} - \boldsymbol{F}(g(\boldsymbol{a}), t, \mu)\|_{\mathbf{P}_{\boldsymbol{a}}^2}^2. \tag{28}$$

Since for the FTR $\boldsymbol{F}$ is non-vanishing at the locations of the front, i.e. at the roots of the level-set function, we can perform a time-dependent adaptive thresholding, which defines $\mathbf{P}_{\boldsymbol{a}}$. Starting from the sorted entries $|\phi_{s_1}| \leq |\phi_{s_2}| \leq \cdots \leq |\phi_{s_{M_p}}| \leq |\phi_{s_{M_p+1}}| \cdots \leq |\phi_{s_M}|$ of the level set vector $\boldsymbol{\phi} = \boldsymbol{\Psi}\boldsymbol{a} \in \mathbb{R}^M$, the threshold search selects the M_p first indices $\boldsymbol{s} = (s_1, \ldots, s_{M_p}) \in \{1, \ldots, M\}^{M_p}$. These are the indices at which we evaluate $\hat{\mathbf{P}}_{\boldsymbol{a}} f(\boldsymbol{\phi})$, $\mathbf{P}_{\boldsymbol{a}}\mathbf{J}_g$, $\mathbf{P}_{\boldsymbol{a}}\boldsymbol{F}$. For two time instances, the sample points are visualized in Fig. 10 for the 2D ARD-system of Sect. 2.5. Note, that the threshold search is a heuristic in order to perform a cheap minimization of the residual (24) using the relation (28).

Further, it should be noted that in this work we are using explicit time integration schemes, as they are usually used inside finite difference solvers. Therefore, the aforementioned methods [36, 54] are not comparable in speedup, since they compare the ROM with implicit time integration schemes used in the FOM. 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 [3] for reaction–diffusion systems in one spatial dimension. Although the framework cannot cope with topological changes, since it relies on a smooth parameterization of the transport, the authors claim speedups of up to a factor of 130.

In the following section, we will show some numerical examples utilizing the here presented hyper-reduction approach.

3.4 Numerical Examples

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

$$\text{offline/online err} = \frac{\|\mathbf{Q}^{\text{train/test}} - \tilde{\mathbf{Q}}^{\text{train/test}}\|_{\text{F}}}{\|\mathbf{Q}^{\text{train/test}}\|_{\text{F}}}. \tag{29}$$

Table 7 Parameters of the 1D reaction–diffusion simulations and the decomposition procedure (Algorithm 1)

Property	Value
FOM—parameters	
Simulation time T	1
Domain Ω	$[-15, 15]$
Grid resolution M	4000
ROM—parameters	
FTR algorithm	Algorithm 1
Number of snapshots	202
FTR iterations	8000
FTR step width τ	4
front function $f(x)$	$0.5(1 + \tanh(x))$

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 $\mu \in \mathcal{P}$ in its columns. The superscript "train" ("test") belongs to the snapshots $\mu \in \mathcal{P}^{\text{train}}$ ($\mathcal{P}^{\text{test}}$) 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 1) or, in the case of the POD, the projection onto the first r left singular vectors of $\mathbf{Q}^{\text{train}}$ contained in $\mathbf{U} \in \mathbb{R}^{M\times r}$, i.e. $\tilde{\mathbf{Q}}^{\text{train}} = \mathbf{U}^T\mathbf{U}\mathbf{Q}^{\text{train}}$.

$\tilde{\mathbf{Q}}^{\text{test}}$ refers to the results evaluating the ROM (21) for the given time interval and parameters $\mu \in \mathcal{P}^{\text{test}}$ using the reduced mapping $g : \mathbb{R}^r \to \mathbb{R}^M$. Specifically, in the case of the POD, the dynamical ROM predictions use $g(\boldsymbol{a}) = \mathbf{U}\boldsymbol{a}$ as a reduced mapping, whereas $g(\boldsymbol{a}) = f(\boldsymbol{\Psi}\boldsymbol{a})$ for the FTR.

Furthermore, we define the projection error:

$$\text{proj. err} = \frac{\|\mathbf{Q}^{\text{test}} - \tilde{\mathbf{Q}}^{\text{test}}_*\|_\text{F}}{\|\mathbf{Q}^{\text{test}}\|_\text{F}} \tag{30}$$

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

3.4.1 Reaction–Diffusion System in 1D

First, we test our approach on an analytic test case taken and modified from [68]. The test case is based on a one-dimensional scalar nonlinear reaction–diffusion equation

$$\partial_t q = \partial_{xx} q + \frac{8}{\mu^2} q^2(q-1) \quad (t,x) \in [0,1] \times [-15, 15] \tag{31}$$

with corresponding analytical solution

$$q(x,t,\mu) = f\left(\frac{|x| - 2t/\mu - 2}{\mu}\right), \tag{32}$$

given that $f(x) = \frac{1}{2}(1+\tanh(x))$. We discretize (31) with central finite difference of 6th order and periodic boundary conditions. Furthermore, we use an explicit Runge-Kutta integration method of 5th(4th) order for adaptive time stepping of the FOM and ROM ODE-system [77]. The numerical parameters for the FTR-decomposition and discretization are stated in Table 7.

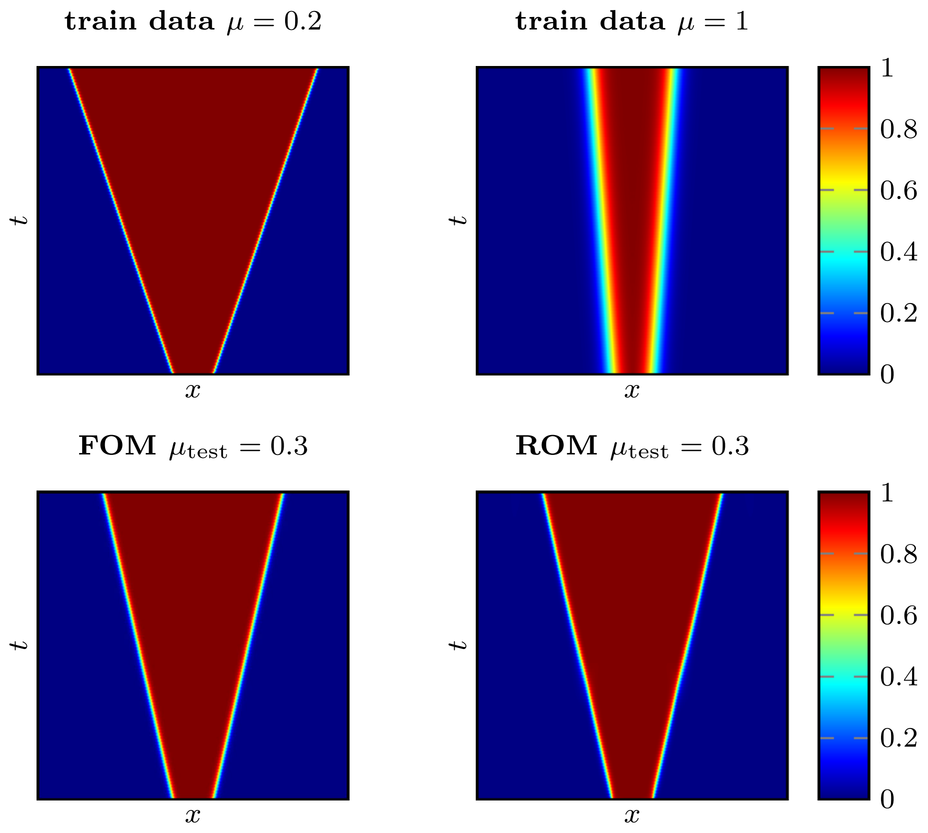


Fig. 11 Training and test data of the reaction diffusion system (31) with the FTR-ROM using $r = 3$ degrees of freedom

The training data consists of 202 samples, including 101 samples of each training parameter $\mu \in \mathcal{P}^{\text{train}} = \{0.2, 1\}$. The training data is visualized as color plot in Fig. 11 together with the ROM prediction of the FTR using $r = 3$ and $\mu = \mu_{\text{test}} = 0.3$ in (31). The FTR algorithm (Algorithm 1) is run for 8000 steps using $\tau = 4$ and different truncation ranks $1 < r < 10$. After we have computed the reduced mapping $\boldsymbol{q}(t, \mu) = f(\boldsymbol{\Psi}\boldsymbol{a}(t, \mu))$ from the training set, we can compute the starting values $\boldsymbol{a}(0, \mu)$, $\mu \in \mathcal{P}^{\text{test}}$ to test the ROM (21) by minimizing the initial condition of the ROM (22), using Gauss-Newton iterations [86]. As an initial 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 (32). The results are reported as online errors in Table 8 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 8 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-FTR (Algorithm 2) and compare it to

Table 8 Offline, online and projection errors for FTR and POD

	FTR			POD	
Rank	Offine 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

The errors are reported for the cumulated snapshot data of the training and test parameters used in Sect. 3.4.1

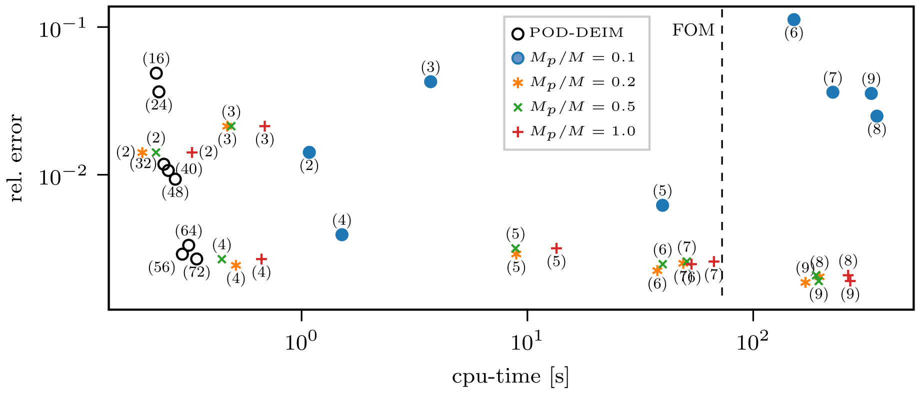


Fig. 12 Error versus CPU-time for the accumulated parameter range $\mu \in \mathcal{P}^{\text{test}}$. Different ranks r are indicated as (r) near the markers. The dashed line indicates the CPU time needed for solving the FOM. The sampled fraction M_p/M in the hyper-reduced FTR-ROM is given in terms of the size M of the FOM. The POD-DEIM approach of [45] is given for reference. In the POD-DEIM approach (r) denotes the number of DEIM points p, which is identical to the number of POD modes $r = p$ used

POD-DEIM (see Appendix B for details). Figure 12 compares CPU-time and error for $M_p = 0.1\,M, 0.2\,M, 0.5\,M$ and M number of grid points, where M is the dimension of the FOM. The figure indicates that even without hyper-reduction, speedups can be achieved compared to the FOM, due to larger step sizes in the reduced coordinates. Comparing the hyper-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. Figure 12 shows that for the cumulated parameter range, the POD-DEIM approach is superior to the presented hyper-reduced FTR-ROM, albeit we expect larger errors when comparing single trajectories for the smallest μ, i.e. steepest fronts.

3.4.2 Advection–Reaction–Diffusion System in 2D

Finally, we test the online performance of the hyper-FTR on the advection–reaction–diffusion example of (13), introduced in Sect. 2.5. We build the training/testing data $\mathbf{Q}^{\text{train}}$

from 101 equally spaced snapshots (visualized in Fig. 7) with $t \in [0, 3]$, $\mu \in \mathcal{P}^{\mathrm{train}} = \{10, 30, 50, 70, 100\}$ and respectively $\mu \in \mathcal{P}^{\mathrm{test}} = \{20, 40, 60, 80, 90\}$. The online, offline, and projection errors of the test case are shown in Fig. 13 together with the speedup generated by the hyper-reduction scheme. It is visible that the FTR outperforms the POD with respect to 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. 10) does not affect the online error, as can be seen from Fig. 13a. Unfortunately, for hyper-reduced systems with $M_p/M < 0.1$ the resulting ROM becomes unstable in the tested parameter range. Figure 13b shows, that for small r, the additional costs ($\mathcal{O}(rM)$) for the matrix multiplication $\boldsymbol{\phi}(t, \mu) = \boldsymbol{\Psi}\boldsymbol{a}(t, \mu)$ are negligible, compared to the evaluation of $\boldsymbol{F}$. However, as soon as r becomes large, the speedups of the hyper-reduction scheme are compensated by the computation of $\boldsymbol{\phi}$ inside Algorithm 2. The balance point at which the additional costs compensate the costs of the RHS is problem dependent, but computing the sample points from $\boldsymbol{\phi}$ is a bottleneck of this method, since it scales with the FOM dimension. 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.

4 Discussion and Conclusion

In this work, we have introduced the front transport reduction (FTR) method to decompose and simulate transports of complex moving fronts. The decomposition parameterizes moving fronts with the help of a transport-dependent auxiliary field $\boldsymbol{\phi}$ and a function f to approximate the front profile. Two different decomposition algorithms have been proposed based on singular value thresholding (Algorithm 1) and artificial neural networks (Sect. 2.3). These methods are purely data-driven since they only require a set of snapshots $\boldsymbol{q}(t, \mu) \in \mathbb{R}^M$ of the FOM as input. The resulting approximation $\boldsymbol{q}(t, \mu) \approx f(\boldsymbol{\phi}(t, \mu))$ is well suited for model order reduction of reacting fronts, since $\boldsymbol{\phi}(t, \mu) = \boldsymbol{\Psi}\boldsymbol{a}(t, \mu) \in \mathbb{R}^M$ 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 [61–64]). Making explicit use of the physical structure has advantages over other linear and nonlinear dimension reduction methods, for reasons we discuss in the following: It was shown, that for various ARD systems the FTR requires fewer modes to decompose the input snapshots compared with the proper orthogonal decomposition (POD), i.e. it has a better compression quality. Regarding artificial autoencoder networks, the FTR is similar in the sense that it uses a linear layer activated by a problem dependent nonlinear front function as a decoder. Here, other authors [36] use multiple nonlinear activated layers $\boldsymbol{q} \approx f(f \ldots f(\boldsymbol{a}))$, resulting in costly evaluations of the network itself. This can limit the overall performance of the ROM when evaluating the additional nonlinearities. Furthermore, the autoencoder networks are often difficult to tune and require training on GPUs. Similar to artificial autoencoder networks, the FTR can approximate topological changes in the evolution of the contour line of the front since it does not make explicit assumptions on the mapping. These topological changes cause problems for most of the transport compensating methods [16–24], because they use transformations that are smooth bijections on the simulation domain.

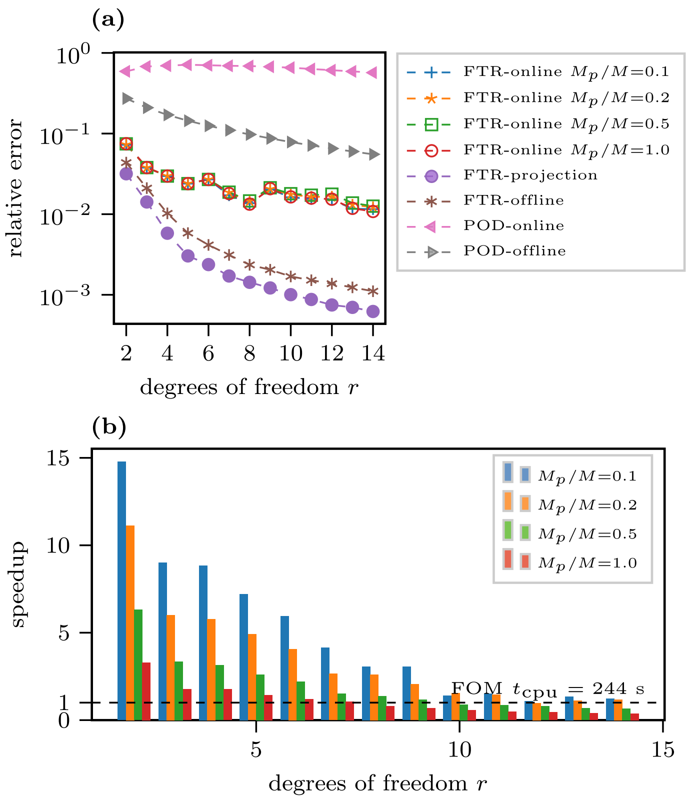


Fig. 13 Hyper-FTR results: **a** relative errors defined in Eqs. (29) and (30) for $\mathcal{P}^{\text{test/train}}$ using POD and FTR decomposition. **b** Speedup versus degrees of freedom for the cumulated parameter range $\mu \in \mathcal{P}^{\text{test}}$. The speedups are compared for different numbers of sampled grid points $M_p \leq M$. The full order model (FOM) using $M = 512^2$ grid points is marked with a dashed line

The ability of the FTR to predict new system states has been demonstrated for non-intrusive (Sect. 3.1) and intrusive (Sect. 3.2) ROMs. Since the FTR gives additional insights into the underlying structure (transport field $\boldsymbol{\phi}$), 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 $\boldsymbol{\phi}$. This can be seen as an adaptive version of the reduced integration domain method [58]. Other nonlinear hyper-reduction methods preselect a set of sample points, on which the dynamics are evaluated. Since for the studied systems, only sample points close to the front are relevant for the dynamics, 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.

To apply our findings to more complex advection–reaction–diffusion systems such as combustion systems in fluid mechanics with multiple reacting species, the decomposition has to be extended to allow arbitrary traveling front shapes. Here, the FTR method would benefit from a generalization or combination with the shifted POD [18], as this would allow decomposing of multiple traveling wave systems with topological changes. Furthermore, it would be interesting to see if our approach can be applied to multi-phase flows, as they inherit a similar front structure separating the fluids. Here, the similarity with level-set-based methods like the characteristic mapping method [87] should be exploited.

Acknowledgements Philipp Krah gratefully acknowledges the support of the Deutsche Forschungsgemeinschaft (DFG) as part of GRK2433 DAEDALUS. Furthermore, we thank Iris Wohnsiedler, Kai Schneider, Mathias Lemke and Volker Mehrmann for useful comments and discussions.

Author Contributions PK: concept, method, implementation, writing original draft. SB: implementation of neural networks, reviewing and editing. MH: computations of the Bunsen flame, reviewing and editing. JR: proposed initial idea and application to ARD-systems, supervision, reviewing and editing.

Funding Funding is provided by DFG funded GRK2433 DAEDALUS. Open Access funding enabled and organized by Projekt DEAL.

Data Availability Enquiries about data availability should be directed to the authors. The data for the Bunsen flame is available for download: https://doi.org/10.5281/zenodo.7681268.

Code Availability For some selected examples we provide MATLAB and Python code at: https://github.com/MOR-transport/FrontTransportReduction. The data for the Bunsen flame is available for download: https://doi.org/10.5281/zenodo.7681268.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

A 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 $\boldsymbol{q} \in \mathbb{R}^M$ 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 is not activated. A summary of the encoder architecture is listed in Table 9. 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 paper 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

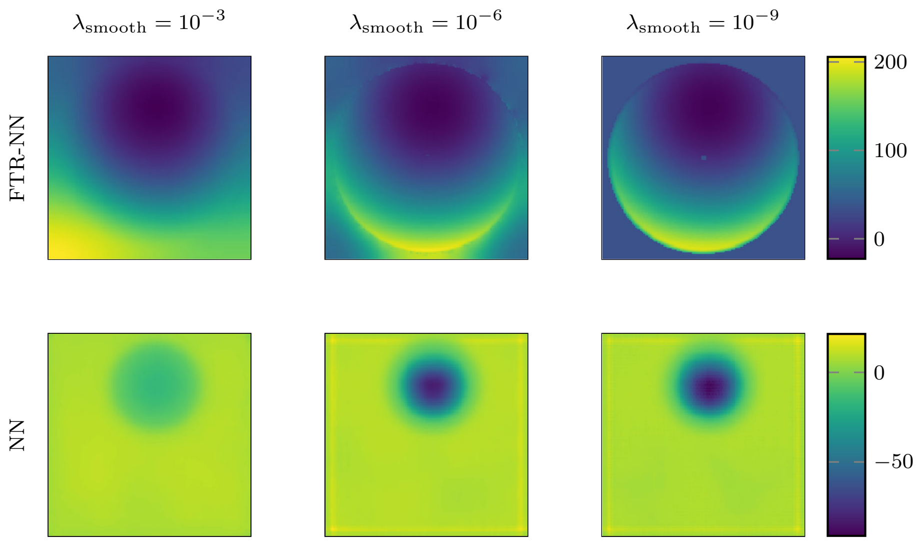


Fig. 14 Color plot of one snapshot of the FTR-NN and NN levelset field ϕ using three degrees of freedom and different smoothness strength λ_{smooth}. The smoothness parameter λ_{smooth} controls the strength of the smoothness constraint (9) (Color figure online)

weight matrix of the layer. Afterwards the resulting output $\boldsymbol{\phi} = \mathbf{W}\boldsymbol{a}$ is reshaped into the spatial domain. In analogy to the FTR ansatz $\boldsymbol{q} \approx \tilde{\boldsymbol{q}} = f(\boldsymbol{\phi})$, 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 10.

After splitting the data by taking every other time step into a training set 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.

B POD-DEIM for the 1D KPP-system

In this section we give additional details on the POD-DEIM approximation of the 1D KPP-system:

$$\begin{cases} 0 = \partial_t q - \partial_{xx} q + \frac{8}{\mu^2} q^2 (q-1) \\ q(x,t) = f(\frac{|x|-2-t/\mu}{\mu}) \end{cases} . \tag{33}$$

In the example 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 [77]. The numerical parameters for the FTR-decomposition and discretization are stated in Table 7.

Discretizing (33) yields the full order model

$$\text{(FOM)} \quad \begin{cases} \dot{\boldsymbol{q}}(t,\mu) = \boldsymbol{F}(\boldsymbol{q},t,\mu) \\ \boldsymbol{q}(0) = \boldsymbol{q}_0, \end{cases} \tag{34}$$

Table 9 Encoder network details

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		

$\tilde{M}$ 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 $N_{\text{out}} = (N_{\text{in}} - \text{kernel size})\,/\text{stride} + 1$

Table 10 NN decoder network details

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)				

with the FOM-RHS

$$\boldsymbol{F}(\boldsymbol{q}, t, \mu) = \mathbf{L}\boldsymbol{q} + \mu \boldsymbol{N}(\boldsymbol{q}), \tag{35}$$

that consists of a time independent linear operator $\mathbf{L} \in \mathbb{R}^{M \times M}$ and a nonlinear operator $\boldsymbol{N} \colon \mathbb{R}^M \to \mathbb{R}^M$. In the POD-Galerkin approach a linear mapping $g(\boldsymbol{a}) = \mathbf{U}\boldsymbol{a}$ for the trial and test spaces is used. The orthogonal matrix $\mathbf{U} \in \mathbb{R}^{M \times r}$ is computed with help of the POD. POD-Galerkin is applied in combination with the DEIM [45] to evaluate the non-linear terms in (33) efficiently. The resulting hyper-reduced system reads:

$$\dot{\boldsymbol{a}} = \mathbf{L}_r \boldsymbol{a} + \mu \tilde{\boldsymbol{N}}(\boldsymbol{a}) \quad \text{with} \quad \mathbf{L}_r = \mathbf{U}^T \mathbf{L} \mathbf{U} \in \mathbb{R}^{r \times r}, \tag{36}$$

$$\text{and} \quad \tilde{\boldsymbol{N}}(\boldsymbol{a}) = \mathbf{U}^T \boldsymbol{\Theta} (\mathbf{P}\boldsymbol{\Theta})^{-1} \mathbf{P} \boldsymbol{N}(\mathbf{U}\boldsymbol{a}). \tag{37}$$

In our studies the matrices $\boldsymbol{\Theta} \in \mathbb{R}^{M \times p}$ and $\mathbf{P} \in \mathbb{R}^{p \times M}$ are computed with the DEIM algorithm proposed in [45], setting $r = p$. Here, $\boldsymbol{\Theta}$ contains the p left singular vectors of the snapshots matrix of the RHS and $\mathbf{P}$ corresponds to interpolation point matrix which picks p points at which $\boldsymbol{N}$ is usually evaluated.

References

1. Huang, C., Duraisamy, K., Merkle, C.: Challenges in reduced order modeling of reacting flows. In: 2018 Joint Propulsion Conference, p. 4675 (2018)
2. Grepl, M.A.: Model order reduction of parametrized nonlinear reaction–diffusion systems. Comput. Chem. Eng. **43**, 33–44 (2012)
3. Black, F., Schulze, P., Unger, B.: Efficient wildland fire simulation via nonlinear model order reduction. Fluids **6**(8), 280 (2021)
4. Krah, P., Sroka, M., Reiss, J.: Model order reduction of combustion processes with complex front dynamics. In: Numerical Mathematics and Advanced Applications ENUMATH 2019, pp. 803–811, Aug (2020). https://doi.org/10.1007/978-3-030-55874-1_79
5. Swischuk, R., Kramer, B., Huang, C., Willcox, K.: Learning physics-based reduced-order models for a single-injector combustion process. AIAA J. **58**(6), 2658–2672 (2020). https://doi.org/10.2514/1.J058943
6. Binder, A., Jadhav, O., Mehrmann, V.: Model order reduction for the simulation of parametric interest rate models in financial risk analysis. J. Ind. Math. **11**(1), 1–34 (2021)
7. Schulze, P., Reiss, J., Mehrmann, V.: Model reduction for a pulsed detonation combuster via shifted proper orthogonal decomposition. In: Active Flow and Combustion Control 2018: Papers Contributed to the Conference "Active Flow and Combustion Control 2018", September 19–21, 2018, Berlin, Germany, pp. 271–286. Springer (2019)
8. Huang, C., Wentland, C.R., Duraisamy, K., Merkle, C.: Model reduction for multi-scale transport problems using structure-preserving least-squares projections with variable transformation. arXiv:2011.02072 (2020)
9. Benner, P., Gugercin, S., Willcox, K.: A survey of projection-based model reduction methods for parametric dynamical systems. SIAM Rev. **57**(4), 483–531 (2015). https://doi.org/10.1137/130932715
10. Kunisch, K., Volkwein, S.: Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. SIAM J. Numer. Anal. **40**(2), 492–515 (2002). https://doi.org/10.1137/S0036142900382612
11. Kolmogoroff, A.: Uber die beste annaherung von funktionen einer gegebenen funktionenklasse. Ann. Math. 107–110 (1936)
12. Unger, B., Gugercin, S.: Kolmogorov n-widths for linear dynamical systems. Adv. Comput. Math. **45**(5–6), 2273–2286 (2019)
13. Ohlberger, M., Rave, S.: Reduced basis methods: success, limitations and future challenges. In: Proceedings of the Conference Algoritmy, pp. 1–12 (2016)
14. Greif, C., Urban, K.: Decay of the Kolmogorov n-width for wave problems. Appl. Math. Lett. **96**, 216–222 (2019)
15. Peherstorfer, B.: Breaking the Kolmogorov barrier with nonlinear model reduction. Not. Am. Math. Soc. **69**(5), 725–733 (2022)
16. Reiss, J., Schulze, P., Sesterhenn, J., Mehrmann, V.: The shifted proper orthogonal decomposition: a mode decomposition for multiple transport phenomena. SIAM J. Sci. Comput. **40**(3), A1322–A1344 (2018)

17. Black, F., Schulze, P., Unger, B.: Modal decomposition of flow data via gradient-based transport optimization. In: R. King, D. Peitsch (eds) Active Flow and Combustion Control 2021, pp. 203–224. Springer International Publishing, Cham. ISBN 978-3-030-90727-3 (2022)
18. Reiss, J.: Optimization-based modal decomposition for systems with multiple transports. SIAM J. Sci. Comput. **43**(3), A2079–A2101 (2021). https://doi.org/10.1137/20M1322005
19. Fedele, F., Abessi, O., Roberts, P.J.: Symmetry reduction of turbulent pipe flows. J. Fluid Mech. **779**, 390–410 (2015)
20. Rowley, C.W., Kevrekidis, I.G., Marsden, J.E., Lust, K.: Reduction and reconstruction for self-similar dynamical systems. Nonlinearity **16**(4), 1257–1275 (2003). https://doi.org/10.1088/0951-7715/16/4/304
21. Tommaso, T., Lei, Z.: Space-time registration-based model reduction of parameterized one-dimensional hyperbolic PDEs. ESAIM: M2AN **55**(1), 99–130 (2021). https://doi.org/10.1051/m2an/2020073
22. Mojgani, R., Balajewicz, M.: Physics-aware registration based auto-encoder for convection dominated PDEs. arXiv:2006.15655 (2020)
23. Rim, D., Moe, S., LeVeque, R.J.: Transport reversal for model reduction of hyperbolic partial differential equations. SIAM/ASA J. Uncertain. Quantif. **6**(1), 118–150 (2018). https://doi.org/10.1137/17M1113679
24. Rim, D., Peherstorfer, B., Mandli, K.T.: Manifold approximations via transported subspaces: Model reduction for transport-dominated problems. arXiv:1912.13024 (2019)
25. Nair, N.J., Balajewicz, M.: Transported snapshot model order reduction approach for parametric, steady-state fluid flows containing parameter dependent shocks. Int. J. Numer. Methods Eng. **117**(12), 1234–1262 (2019)
26. Cagniart, N., Crisovan, R., Maday, Y., Abgrall, R.: Model order reduction for hyperbolic problems: a new framework. In: Working paper or preprint, August 2017. https://hal.science/hal-01583224
27. Ferrero, A., Taddei, T., Zhang, L.: Registration-based model reduction of parameterized two-dimensional conservation laws. J. Comput. Phys. **457**, 111068 (2022)
28. Nonino, M., Ballarin, F., Rozza, G., Maday, Y.: 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] (November, 2019)
29. Karatzas, E.N., Ballarin, F., Rozza, G.: Projection-based reduced order models for a cut finite element method in parametrized domains. Comput. Math. Appl. **79**(3), 833–851 (2020). https://doi.org/10.1016/j.camwa.2019.08.003
30. Papapicco, D., Demo, N., Girfoglio, M., Stabile, G., Rozza, G.: The neural network shifted-proper orthogonal decomposition: a machine learning approach for non-linear reduction of hyperbolic equations. Comput. Methods Appl. Mech. Eng. **392**, 114687 (2022)
31. Mendible, A., Brunton, S.L., Aravkin, A.Y., Lowrie, W., Kutz, J.N.: Dimensionality reduction and reduced-order modeling for traveling wave physics. Theoret. Comput. Fluid Dyn. **34**, 385–400 (2020)
32. Mendible, A., Koch, J., Lange, H., Brunton, S.L., Kutz, J.N.: Data-driven modeling of rotating detonation waves. Phys. Rev. Fluids **6**(5), 050507 (2021)
33. Fresca, S., Dede', L., Manzoni, A.: A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs. J. Sci. Comput. **87**(2), 61 (2021). https://doi.org/10.1007/s10915-021-01462-7. (**ISSN 1573-7691**)
34. Fresca, S., Manzoni, A.: POD-DL-ROM: Enhancing deep learning-based reduced order models for non-linear parametrized PDEs by proper orthogonal decomposition. Comput. Methods Appl. Mech. Eng. **388**, 114181 (2022)
35. Lee, K., Carlberg, K.T.: Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. J. Comput. Phys. **404**, 108973 (2020). https://doi.org/10.1016/j.jcp.2019.108973
36. Kim, Y., Choi, Y., Widemann, D., Zohdi, T.: A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. J. Comput. Phys. **451**, 110841 (2021)
37. Ghorbani, A., Abid, A., Zou, J.: Interpretation of neural networks is fragile. In: Proceedings of the AAAI Conference on Artificial Intelligence vol. **33**, pp. 3681–3688 (2019)
38. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv:1412.6572 (2014)
39. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks (2014)
40. Koch, O., Lubich, C.: Dynamical low-rank approximation. SIAM J. Matrix Anal. Appl. **29**(2), 434–454 (2007)
41. Peherstorfer, B.: Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling. SIAM J. Sci. Comput. **42**(5), A2803–A2836 (2020). https://doi.org/10.1137/19M1257275. (**Publisher: Society for Industrial and Applied Mathematics**)
42. Dihlmann, M., Drohmann, M., Haasdonk, B.: Model reduction of parametrized evolution problems using the reduced basis method with adaptive time-partitioning. Proc. ADMOS **2011**, 64 (2011)

43. Etter, P.A., Carlberg, K.T.: Online adaptive basis refinement and compression for reduced-order models via vector-space sieving. Comput. Methods Appl. Mech. Eng. **364**, 112931 (2020)
44. Peherstorfer, B., Willcox, K.: Online adaptive model reduction for nonlinear systems via low-rank updates. SIAM J. Sci. Comput. **37**(4), A2123–A2150 (2015). https://doi.org/10.1137/140989169
45. Chaturantabut, S., Sorensen, D.C.: Nonlinear model reduction via discrete empirical interpolation. SIAM J. Sci. Comput. **32**(5), 2737–2764 (2010)
46. Uy, W.I.T., Wentland, C.R., Huang, C., Peherstorfer, B.: Reduced models with nonlinear approximations of latent dynamics for model premixed flame problems. arXiv:2209.06957 (2022)
47. Koellermeier, J., Krah, P., Kusch, J.: Split conservative model order reduction for hyperbolic shallow water moment equations using dynamic low rank approximation and POD-Galerkin. Adv. Comput. Math. (2023). https://philipp137.github.io/assets/poster/Poster_KoellermeierKrahKusch.pdf. To be published soon
48. Buffoni, M., Willcox, K.: Projection-based model reduction for reacting flows. In: 40th Fluid Dynamics Conference and Exhibit, p. 5008 (2010)
49. Xu, J., Duraisamy, K.: Reduced-order modeling of model rocket combustors. In: 53rd AIAA/SAE/ASEE Joint Propulsion Conference, p. 4918 (2017)
50. Wang, Q., Hesthaven, J.S., Ray, D.: Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem. J. Comput. Phys. **384**, 289–307 (2019)
51. Corrochano, A., Freitas, R.S.M., Parente, A., Clainche, S.L.: A predictive physics-aware hybrid reduced order model for reacting flows. arXiv:2301.09860 (2023)
52. Carlberg, K., Farhat, C., Cortial, J., Amsallem, D.: The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. J. Comput. Phys. **242**, 623–647 (2013). https://doi.org/10.1016/j.jcp.2013.02.028
53. Black, F., Schulze, P., Unger, B.: Projection-based model reduction with dynamically transformed modes. ESAIM: M2AN **54**(6), 2011–2043 (2020). https://doi.org/10.1051/m2an/2020046
54. Jain, S., Tiso, P.: Hyper-reduction over nonlinear manifolds for large nonlinear mechanical systems. J. Comput. Nonlinear Dyn. **14**(8), 081008 (2019)
55. Peherstorfer, B., Drmac, Z., Gugercin, S.: Stability of discrete empirical interpolation and gappy proper orthogonal decomposition with randomized and deterministic sampling points. SIAM J. Sci. Comput. **42**(5), A2837–A2864 (2020)
56. Wentland, C.R., Huang, C., Duraisamy, K.: Investigation of sampling strategies for reduced-order models of rocket combustors. In: AIAA Scitech 2021 Forum, p. 1371 (2021)
57. Huang, C., Duraisamy, K., Merkle, C.L.: Investigations and improvement of robustness of reduced-order models of reacting flow. AIAA J. **57**(12), 5377–5389 (2019)
58. Ryckelynck, D.: A priori hyperreduction method: an adaptive approach. J. Comput. Phys. **202**(1), 346–366 (2005)
59. Lange, H., Brunton, S.L., Kutz, J.N.: From Fourier to Koopman: spectral methods for long-term time series prediction. J. Mach. Learn. Res. **22**(41), 1–38 (2021)
60. Kolmogorov, A., Petrovskii, I., Piscunov, N.: 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**(6), 1–25 (1937)
61. Hadeler, K.P., Rothe, F.: Travelling fronts in nonlinear diffusion equations. J. Math. Biol. **2**(3), 251–263 (1975)
62. Berestycki, H., Hamel, F., Nadirashvili, N.: Propagation speed for reaction–diffusion equations in general domains. C.R. Math. **339**(3), 163–168 (2004). https://doi.org/10.1016/j.crma.2004.05.020
63. Berestycki, H., Hamel, F., Roques, L.: Équations de réaction-diffusion et modèles d'invasions biologiques dans les milieux périodiques. C.R. Math. **339**(8), 549–554 (2004). https://doi.org/10.1016/j.crma.2004.07.025
64. Fisher, R.A.: The wave of advance of advantageous genes. Ann. Eugenic **7**(4), 355–369 (1937). https://doi.org/10.1111/j.1469-1809.1937.tb02153.x
65. Poinsot, T., Veynante, D.: Theoretical and numerical combustion. RT Edwards Inc, Morningside (2005)
66. Williams, F.A.: Combustion Theory. Benjamin Cummings, Menlo Park (1985)
67. Peters, N.: Turbulent combustion. Meas. Sci. Technol. **12**(11), 2022 (2001). https://doi.org/10.1088/0957-0233/12/11/708
68. Zhang, R., Zhu, J., Loula, A.F.D., Yu, X.: A new nonlinear Galerkin finite element method for the computation of reaction diffusion equations. J. Math. Anal. Appl. **434**(1), 136–148 (2016)
69. Shchepakina, E., Tropkina, E.: Order reduction for problems with traveling wave solutions to reaction–diffusion systems. J. Phys.: Conf. Ser. **1745**(1), 012109 (2021). https://doi.org/10.1088/1742-6596/1745/1/012109

70. Shalev-Shwartz, S., Ben-David, S.: Understanding machine learning: from theory to algorithms (2014). https://www.cs.huji.ac.il/%7Eshais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf#page=64
71. Halko, N., Martinsson, P.G., Tropp, J.A.: Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev. **53**(2), 217–288 (2011). https://doi.org/10.1137/090771806
72. Krah, P., Engels, T., Schneider, K., Reiss, J.: Wavelet adaptive proper orthogonal decomposition for large scale flow data. Adv. Comput. Math. (2021)
73. Van Der Maaten, L., Postma, E., Van den Herik, J.: Dimensionality reduction: a comparative. J. Mach. Learn. Res. **10**(66–71), 13 (2009)
74. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT Press, Cambridge (2016)
75. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: ICML, pp. 448–456. PMLR (2015)
76. Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., Sohl-Dickstein, J.: On the expressive power of deep neural networks. In: International Conference on Machine Learning, pp. 2847–2854. PMLR (2017)
77. Dormand, J.R., Prince, P.J.: A family of embedded Runge-Kutta formulae. J. Comput. Appl. Math. **6**(1), 19–26 (1980)
78. Reiss, J.: A family of energy stable, skew-symmetric finite difference schemes on collocated grids. J. Sci. Comput. **65**(2), 821–838 (2015)
79. Blonigan, P.J., Carlberg, K., Rizzi, F., Howard, M., Fike, J.A.: Model reduction for hypersonic aerodynamics via conservative LSPG projection and hyper-reduction. In: AIAA Scitech 2020 Forum, p. 0104 (2020)
80. Fukami, K., Murata, T., Zhang, K., Fukagata, K.: Sparse identification of nonlinear dynamics with low-dimensionalized flow representations. J. Fluid Mech. **926**, A10 (2021)
81. Quade, M., Abel, M., Kutz, J.N., Brunton, S.L.: Sparse identification of nonlinear dynamics for rapid model recovery. Chaos: An Interdis. J. Nonlinear Sci. **28**(6), 063116 (2018)
82. Lange, H.: Fourier to Koopman implementation. https://github.com/helange23/from_fourier_to_koopman (2019). Visited 6 December 2021
83. Kornilov, V.N., Rook, R., ten Thije Boonkkamp, J.H.M., De Goey, L.P.H.: Experimental and numerical investigation of the acoustic response of multi-slit Bunsen burners. Combust. Flame **156**(10), 1957–1970 (2009)
84. Jaensch, S., Merk, M., Gopalakrishnan, E.A., Bomberg, S., Emmert, T., Sujith, R.I., Polifke, W.: Hybrid CFD/low-order modeling of nonlinear thermoacoustic oscillations. Proc. Combust. Inst **36**(3), 3827–3834 (2017)
85. Weller, H.G., Tabor, G., Jasak, H., Fureby, C.: A tensorial approach to computational continuum mechanics using object-oriented techniques. Comput. Phys. **12**(6), 620–631 (1998)
86. Moré, J.J.: The Levenberg–Marquardt algorithm: implementation and theory. In: Numerical Analysis, pp. 105–116. Springer (1978)
87. Mercier, O., Yin, X.Y., Nave, J.C.: The characteristic mapping method for the linear advection of arbitrary sets. SIAM J. Sci. Comput. **42**(3), A1663–A1685 (2020)