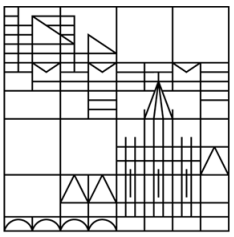
Data-Driven Modeling and Parameter Estimation for Reaction-Diffusion Systems

Master Thesis

submitted by
Lennart Lohrmann
at the

Universität Konstanz



Department of Mathematics and Statistics

Supervisor and 1st Reviewer: Prof. Dr. Stefan Volkwein
2nd Reviewer: Jun.-Prof. Dr. Behzad Azmi

July 22, 2025

Declaration

I hereby declare that I have written this thesis on the topic

Data-Driven Modeling and Parameter
Estimation for Reaction-Diffusion Systems

independently and have not used any sources other than those indicated. I have marked the passages that are taken from other works, either verbatim or in sense, in each individual case by indicating the source, including any secondary literature used, as borrowed.

If I have used text-generating AI tools as an aid, I acknowledge that I am solely responsible for the accuracy of the content in AI-generated text passages, as well as for properly attributing formulations and ideas of others in accordance with the principles of good scientific practice.

The thesis has not yet been submitted to any other examination authority and has not yet been published.

Konstanz, July 22, 2025

_____

Lennart Lohrmann

Abstract

One aspect of this thesis is the exploration of data-driven model reduction techniques for efficiently analyzing the dynamic behavior captured in temporal datasets generated by reaction-diffusion partial differential equation systems. The first such technique is the Dynamic Mode Decomposition, an equation-free method originally introduced by Peter Schmid and Jörn Sesterhenn in 2008 ([23]). Building on this, a randomized variant of Dynamic Mode Decomposition is employed to improve computational efficiency. However, both versions struggle in providing accurate reconstructions for datasets that exhibit periodic behavior or spatio-temporal Turing instability. To address this issue, we propose a piecewise approach that partitions the datasets and applies Dynamic Mode Decomposition locally to each subset. The second technique is Proper Orthogonal Decomposition, which is a well-established method for model order reduction. To further reduce computational costs, Proper Orthogonal Decomposition is combined with the Discrete Empirical Interpolation Method. Despite this, the reconstruction accuracy remains insufficient in some cases. Therefore, we introduce a correction-based strategy to enhance the quality of the reduced model. Moreover, by leveraging the specific structure of datasets that exhibit Turing instability, we improve the computational effectiveness even further by extending the previous approaches in an adaptive manner. Another key aspect of this thesis is parameter identification. The proposed strategy relies on computing gradients of the cost functional using a sensitivity approach. These gradients are then used within the projected Barzilai-Borwein optimization method to identify optimal parameter values. Finally, we investigate a specific reaction-diffusion system in which nonlinearities in the reaction kinetics arise from a Hill function, commonly used to model cooperative effects in biochemical reactions.

Contents

Acronyms

BB	Barzilai-Borwein
DMD	Dynamic Mode Decomposition
DEIM	Discrete Empirical Interpolation Method
DOFS	Degrees of freedom
FEM	Finite element method
FE	Finite element
FDM	Finite difference method
FHN	FitzHugh-Nagumo
FOM	Full order model
IBVP	Initial-boundary value problem
IMEX Euler	Implicit-explicit Euler
MOL	Method of lines
MOR	Model order reduction
ODE	Ordinary differential equation
PBB	Projected Barzilai-Borwein
PDE	Partial differential equation
pDMD	Piecewise Dynamic Mode Decomposition
POD	Proper Orthogonal Decomposition
PODc	Corrected POD
POD-DEIMc	Corrected POD-DEIM
RD-PDE	Reaction-diffusion partial differential equation
rDMD	Randomized Dynamic Mode Decomposition
rQB	Randomized QB decomposition
ROM	Reduced order model
SD	Steepest descent
SVD	Singular Value Decomposition

1 Introduction

The analysis of dynamical systems plays a central role in many scientific and engineering disciplines. In particular, reaction-diffusion systems provide a mathematical framework for modeling spatio-temporal phenomena in areas such as biology, chemistry and physics. These systems are often governed by semilinear parabolic partial differential equations, whose high-dimensional discrete solution spaces and long simulation times pose significant computational challenges.
To address this, model reduction techniques have become increasingly important. The aim is to approximate the original, high-dimensional system with a significantly lower-dimensional surrogate model that captures the dominant dynamics and greatly reduces computational cost. In recent years, data-driven approaches have emerged as powerful tools, as they rely solely on snapshot data – either from measurements or simulations – without requiring detailed knowledge of the governing equations.

This thesis investigates two prominent model reduction techniques: Dynamic Mode Decomposition and Proper Orthogonal Decomposition. Dynamic Mode Decomposition is a data-driven method that does not require explicit knowledge of the underlying equations. It approximates the system's dynamics using a best-fit linear operator derived directly from the data. To improve efficiency for large datasets, we also consider a randomized variant of Dynamic Mode Decomposition, which first projects the data onto a lower-dimensional random subspace before applying the standard algorithm.
However, both Dynamic Mode Decomposition variants often struggle to accurately reconstruct systems exhibiting periodic behavior or spatio-temporal Turing instabilities. To overcome this limitation, we propose a piecewise approach, in which the dataset is partitioned into smaller segments and Dynamic Mode Decomposition is applied locally to each segment.
The second method explored is Proper Orthogonal Decomposition, a technique closely related to the Singular Value Decomposition. To efficiently handle nonlinearities, it is combined with the Discrete Empirical Interpolation Method, which approximates nonlinear terms using a small number of interpolation points.
Despite these improvements, the reduced-order models still exhibit insufficient accuracy in certain scenarios. For this reason, correction-based strategies are introduced to enhance reconstruction quality. Furthermore, by leveraging the specific structure of data exhibiting Turing patterns, we propose an adaptive extension to further improve computational efficiency.

Another central component of this thesis is parameter identification. To this end, we formulate an optimization problem that aims to minimize a suitable cost functional subject to a reaction-diffusion equation as a constraint. The optimization is performed with respect to both the state variable and the diffusion constant. The diffusion constant appears in the cost functional as a penalty term and is further constrained by a box-constraint. We employ a sensitivity-based approach to compute gradients of the cost functional, which are then used within the projected Barzilai-Borwein optimization algorithm.

The numerical implementation of the methods developed in this thesis is carried out using both MATLAB and PYTHON. MATLAB is primarily employed for finite difference schemes as well as computations based on Dynamic Mode Decomposition and Proper

Orthogonal Decomposition. The use of finite differences and MATLAB ensures consistency with related studies referenced in this thesis, which adopt the same numerical approach. In contrast, finite element simulations and the projected Barzilai-Borwein optimization are implemented in PYTHON, using the open-source library FENICS.

Finally, as a concrete application, we investigate a specific reaction-diffusion system whose nonlinear reaction kinetics are described by a Hill function. We perform a detailed analysis of the system's behavior across various parameter settings and evaluate the effectiveness of the proposed model order reduction techniques by applying both Dynamic Mode Decomposition and Proper Orthogonal Decomposition.

2 Preliminaries

In this section, we introduce fundamental concepts from functional analysis that are used throughout the remainder of the thesis. We follow the references [22] and [24].

Let $\Omega \subset \mathbb{R}^{d_\Omega}$ be a non-empty, bounded domain with Lipschitz-continuous boundary $\Gamma := \partial\Omega$, where $d_\Omega \in \mathbb{N}$ denotes the spatial dimension.

Definition 2.1 (L^p-spaces)**.** For $p \in [1, \infty)$, the *Lebesgue space* $L^p(\Omega)$ is defined as the set of all (equivalence classes of) measurable functions $f\colon \Omega \to \mathbb{R}$ such that

$$\int_\Omega |f(x)|^p \,\mathrm{d}x < \infty.$$

The space $L^p(\Omega)$ endowed with the norm

$$\|f\|_{L^p(\Omega)} := \left(\int_\Omega |f(x)|^p \,\mathrm{d}x\right)^{1/p}$$

is a Banach space. In particular, the space $L^2(\Omega)$ with the associated inner product

$$\langle f, g\rangle_{L^2(\Omega)} := \int_\Omega f(x)g(x) \,\mathrm{d}x \tag{2.1}$$

is a Hilbert space. An equivalence class of a function is understood as the set of all functions that are equal almost everywhere.

Let $v\colon \Omega \to \mathbb{R}$ be a real-valued function on Ω and $\overline{\Omega}$ the closure of Ω.

Definition 2.2. The set

$$\operatorname{supp}(v) := \overline{\{x \in \Omega \,|\, v(x) \neq 0\}}$$

is called the *support of* v and $C_0^k(\Omega)$, $0 \leq k \leq \infty$, denotes the space of k-times continuously differentiable functions v with compact support in Ω.

The case $k = \infty$ is of special interest. On the one hand, functions in $C_0^\infty(\Omega)$ vanish on the boundary Γ, which is advantageous for integration by parts. On the other hand, functions in $C_0^\infty(\Omega)$ are infinitely differentiable. Both properties are important for the definition of Sobolev spaces.
Similarly, the space $C^k(\Omega)$ is defined as the set of all functions that are k-times continuously differentiable in Ω. We define $C^0(\Omega) := C(\Omega)$.
Let

$$\alpha := \begin{bmatrix} \alpha_1 & \cdots & \alpha_{d_\Omega} \end{bmatrix}^\top \in \mathbb{N}_0^{d_\Omega}$$

be a multi-index with order

$$|\alpha| := \alpha_1 + \ldots + \alpha_{d_\Omega}.$$

Then $D^\alpha f(x)$ is a short notation for

$$D^\alpha f(x) := \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \cdots \partial x_{d_\Omega}^{\alpha_{d_\Omega}}}(x).$$

Definition 2.3. The linear space of all continuous real-valued functions $f\colon \bar{\Omega} \to \mathbb{R}$ is denoted by $C(\bar{\Omega})$ and the linear space $C^k(\bar{\Omega})$, $k \in \mathbb{N}$, consists of all $C^k(\Omega)$ functions that can be extended together with their partial derivatives up to order k continuously to $\bar{\Omega}$. $C(\bar{\Omega})$ and $C^k(\bar{\Omega})$ endowed with the norms

$$\begin{aligned} \|f\|_{C(\bar{\Omega})} &:= \max_{x\in\bar{\Omega}} |f(x)|, \\ \|f\|_{C^k(\bar{\Omega})} &:= \sum_{|\alpha|\le k} \|D^\alpha f\|_{C(\bar{\Omega})} \end{aligned}$$

are Banach spaces.

By setting

$$\begin{aligned} L^1_{\mathrm{loc}}(\Omega) := \{ f\colon \Omega \to \mathbb{R} \,|\, & f \text{ is Lebesgue measurable on } \Omega, \\ & \|f\|_{L^1(K)} < \infty \text{ for all compact } K \subset \Omega \} \end{aligned}$$

we can introduce the concept of *weak derivatives*.

Definition 2.4 (Weak derivative)**.** Let $f \in L^1_{\mathrm{loc}}(\Omega)$ and $\alpha \in \mathbb{N}_0^{d_\Omega}$. A function $g \in L^1_{\mathrm{loc}}(\Omega)$ is called weak derivative of f, written as $g = D^\alpha f$, if

$$\int_\Omega f(x) D^\alpha \varphi(x)\,\mathrm{d}x = (-1)^{|\alpha|} \int_\Omega g(x)\varphi(x)\,\mathrm{d}x \quad \text{for all } \varphi \in C_0^\infty(\Omega).$$

This leads to the definition of *Sobolev spaces*.

Definition 2.5 ($W^{k,p}$-spaces)**.** For $k \in \mathbb{N}_0$ and $1 \le p \le \infty$ we define the Sobolev space $W^{k,p}(\Omega)$ as the linear space of all $L^p(\Omega)$ functions f such that all weak derivatives $D^\alpha f$ with $|\alpha| \le k$ exist and are elements of $L^p(\Omega)$. $W^{k,p}(\Omega)$ endowed with the norm

$$\|f\|_{W^{k,p}(\Omega)} := \left(\sum_{|\alpha|\le k} \|D^\alpha f\|^p_{L^p(\Omega)} \right)^{1/p}$$

is a Banach space. For $p = 2$, we write $H^k(\Omega) := W^{k,2}(\Omega)$.

In particular, we have

$$H^1(\Omega) = \{ f \in L^2(\Omega) \,|\, D_i f \in L^2(\Omega) \text{ for } i = 1, \dots, d \}$$

endowed with the norm

$$\begin{aligned} \|f\|_{H^1(\Omega)} &= \left(\|f\|^2_{L^2(\Omega)} + \sum_{i=1}^{d_\Omega} \|D_i f\|^2_{L^2(\Omega)} \right)^{1/2} \\ &= \left(\int_\Omega (|f(x)|^2 + \sum_{i=1}^{d_\Omega} |D_i f(x)|^2)\,\mathrm{d}x \right)^{1/2} \\ &= \left(\int_\Omega (|f(x)|^2 + \|\nabla f(x)\|_2^2)\,\mathrm{d}x \right)^{1/2}. \end{aligned}$$

The space $H^1(\Omega)$ is a Banach space. Introducing the inner product

$$\langle f, g\rangle_{H^1(\Omega)} := \int_\Omega f(x)g(x)\,\mathrm{d}x + \int_\Omega \nabla f(x)\cdot\nabla g(x)\,\mathrm{d}x \tag{2.2}$$

we obtain the Hilbert space $(H^1(\Omega), \langle\cdot,\cdot\rangle_{H^1(\Omega)})$. Notice that

$$\nabla f(x)\cdot\nabla g(x) := \sum_{i=1}^{d_\Omega} \frac{\partial f}{\partial x_i}(x)\frac{\partial g}{\partial x_i}(x) \quad \text{for } f, g \in V.$$

Bochner spaces generalize Lebesgue spaces (cf. 2.1) by allowing the function values to be elements of a Banach space. Let $(X, \|\cdot\|_X)$ be a real Banach space.

Definition 2.6 (Bochner spaces)**.** The Bochner space $L^p(0,T;X)$, $1 \le p < \infty$, is the linear space of (equivalence classes of) measurable functions $f\colon [0,T] \to X$ such that

$$\int_0^T \|f(t)\|_X^p\,\mathrm{d}t < \infty.$$

The space $L^p(0,T;X)$ endowed with the norm

$$\|f\|_{L^p(0,T;X)} := \left(\int_0^T \|f(t)\|_X^p\,\mathrm{d}t\right)^{1/p}.$$

is a Banach space. If X is a Hilbert space, the space $L^2(0,T;X)$ is a Hilbert space too and the standard inner product is given by

$$\langle f, g\rangle_{L^2(0,T;X)} := \int_0^T \langle f(t), g(t)\rangle_X\,\mathrm{d}t \quad \text{for } f, g \in L^2(0,T;X).$$

For functions $x \mapsto f(t,x)$ f.a.a. $t \in [0,T]$, $f \in L^2(0,T;L^2(\Omega))$, we use the brief notation $f(t) = f(t,\cdot)$. 'F.a.a.' stands for 'for almost all'.
In order to introduce the concept of weak solutions, we first define *weak time derivatives*.

Definition 2.7 (Weak time derivative)**.** Let $f \in L^2(0,T;X)$. A function $g \in L^2(0,T;X)$ is called the weak derivative of f, written as $g = f_t$, if

$$\int_0^T f(t)\varphi'(t)\,\mathrm{d}t = -\int_0^T g(t)\varphi(t)\,\mathrm{d}t \quad \text{for all } \varphi \in C_0^\infty(0,T).$$

We consider the following setting.

Definition 2.8. Let H, V be separable Hilbert spaces with the continuous and dense embedding $V \hookrightarrow H$. We identify H with its dual space H'. Then we have the continuous and dense embeddings

$$V \hookrightarrow H = H' \hookrightarrow V'.$$

The triple (V, H, V') is called a *Gelfand triple*. The space $W(0,T;V,V')$ is defined as the space of all $f \in L^2(0,T;V)$ with weak time derivative $f_t \in L^2(0,T;V')$ and it is equipped with the norm

$$\|f\|_{W(0,T;V,V')} := \sqrt{\|f\|^2_{L^2(0,T;V)} + \|f_t\|^2_{L^2(0,T;V')}}. \tag{2.3}$$

Theorem 2.1. Let $V \hookrightarrow H \hookrightarrow V'$ be a Gelfand triple. Then the space

$$W(0,T;V,V')$$

is a Hilbert space with the inner product

$$\langle \cdot, \cdot \rangle_{W(0,T;V,V')} := \langle \cdot, \cdot \rangle_{L^2(0,T;V)} + \langle \cdot, \cdot \rangle_{L^2(0,T;V')}.$$

Furthermore, the embedding

$$W(0,T;V,V') \hookrightarrow C([0,T];H)$$

holds and is continuous. Moreover, the integration by parts formula

$$\int_0^T \langle f_t(t), g(t) \rangle_{V',V} \,\mathrm{d}t = (f(T), g(T))_H - (f(0), g(0))_H - \int_0^T \langle g_t(t), f(t) \rangle_{V',V} \,\mathrm{d}t$$

holds for all $f, g \in W(0,T;V,V')$, where the notation

$$\langle F, v \rangle_{V',V} = F(v) \quad \text{for } F \in V', v \in V,$$

is used for the *dual pairing*.

Since $W(0,T;V,V')$ is continuously embedded into $C([0,T];H)$, $f(0)$ and $f(T)$ are meaningful in H for $f \in W(0,T;V,V')$. In addition, the formula

$$\langle f_t(t), g \rangle_{V',V} = \frac{\mathrm{d}}{\mathrm{d}t} \langle f(t), g \rangle_H$$

holds for $(f,g) \in W(0,T;V,V') \times V$ and almost all $t \in [0,T]$. Analogously to $W(0,T;V,V')$, we define the space $W(0,T;V,H)$ as

$$W(0,T;V,H) := \{ f \in L^2(0,T;V) \mid f_t \in L^2(0,T;H) \}.$$

3 Dynamic Mode Decomposition

This section is based on the work of [8], [10] and [15].

Before introducing the (exact) *Dynamic Mode Decomposition (DMD)*, we briefly review the *Singular Value Decomposition (SVD)* – a fundamental technique in numerical linear algebra and the basis for DMD. The SVD is used to compute low-rank approximations of matrices and to determine the pseudoinverse of non-square matrices, enabling the computation of least-squares solutions for matrix equations. It is a matrix decomposition into three matrices, a rotation matrix, followed by a scaling matrix, followed by another rotation matrix and generalizes the eigendecomposition of a square matrix to any $n \times (m+1)$ matrix. Building on this matrix decomposition, we introduce the DMD method. DMD extracts low-rank structures from high-dimensional temporal datasets. More precisely, it reconstructs the best linear dynamical system underlying the given dataset. Additionally, we introduce a randomized DMD version, called *randomized DMD (rDMD)*, which is computationally more efficient. Finally, we compare the exact DMD method with rDMD using a straightforward example.

3.1 Singular Value Decomposition

Let $X \in \mathbb{R}^{n\times(m+1)}$ be a matrix.

Theorem 3.1 (Singular Value Decomposition)**.** There exist orthogonal matrices

$$\begin{aligned} U &\in \mathbb{R}^{n\times n}, \\ V &\in \mathbb{R}^{(m+1)\times(m+1)} \end{aligned}$$

and uniquely determined real numbers

$$\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_p = 0, \quad r \leq \min\{n, m+1\} =: p,$$

such that

$$U^\top X V = \left[\begin{array}{ccc|ccc} \sigma_1 & & & & \vdots & \\ & \ddots & & \ldots & 0 & \ldots \\ & & \sigma_r & & \vdots & \\ \hline & \vdots & & & \vdots & \\ \ldots & 0 & \ldots & \ldots & 0 & \ldots \\ & \vdots & & & \vdots & \end{array}\right] =: \Sigma \in \mathbb{R}^{n\times(m+1)}. \tag{3.1}$$

In particular, we have $X = U\Sigma V^\top$.

Proof. See [8, Satz 4.27]. □

The values σ_i are called *singular values*. They correspond to the eigenvalues in the eigendecomposition of a diagonalizable square matrix.
The integer r denotes the rank of X, written as $\mathrm{rk}(X)$, as shown in the next lemma. Since the matrices involved in the SVD are often high-dimensional, it is advantageous to reduce their size by truncation. To this end, a so-called *truncation value* – also

denoted by r – is chosen, and only the first r columns, the first r singular values and the first r rows of the corresponding matrices in the SVD are considered. This results in the *reduced matrices* $U \in \mathbb{R}^{n\times r}$, $\Sigma \in \mathbb{R}^{r\times r}$ and $V \in \mathbb{R}^{(m+1)\times r}$ (see Figure 1).

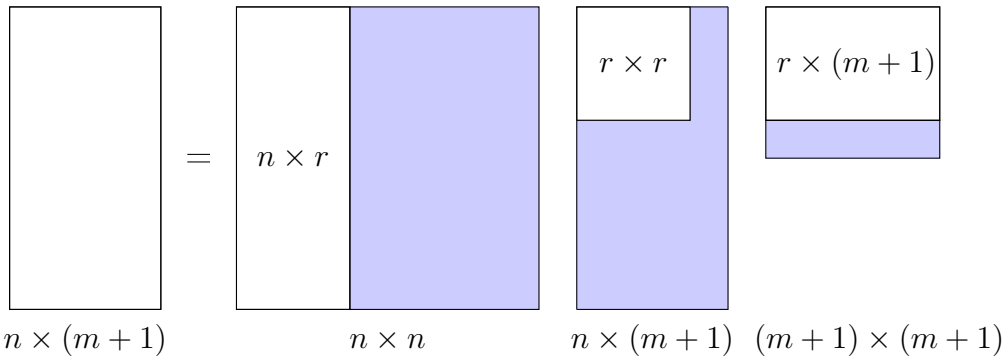


Figure 1: Full SVD versus reduced SVD.

If $r = \mathrm{rk}(X)$, the decompostion is called the *reduced SVD* and yields an exact representation of X. Otherwise, if $r < \mathrm{rk}(X)$, the reduced SVD is often referred to as the *truncated SVD* and provides only an approximate representation of X. It results in increased computational efficiency and reduces the memory requirements for further calculations. Due to the non-ascending order of the singular values, the loss of information resulting from SVD truncation is often small.
The columns of U and V are referred to as the *left* and *right singular vectors*, respectively. We denote the i-th left singular vector by $u_i \in \mathbb{R}^n$ and the i-th right singular vector by $v_i \in \mathbb{R}^{m+1}$. The following lemma summarizes some important properties of the SVD.

Lemma 3.1. Let $X = U\Sigma V^\top$ as in Theorem 3.1. Then:

(i) $Xv_i = \sigma_i u_i \quad \text{for } i = 1, \dots, p,$

(ii) $X^\top u_i = \sigma_i v_i \quad \text{for } i = 1, \dots, p,$

(iii) $\mathrm{rk}(X) = r,$

(iv) $\|X\|_2 = \sigma_1.$

Proof. See [8, Lemma 4.29]. □

It follows directly from the representation $X = U\Sigma V^\top$ that the matrices $XX^\top$ and $\Sigma\Sigma^\top$ are similar:

$$XX^\top = (U\Sigma V^\top)(V\Sigma^\top U^\top) = U\Sigma\Sigma^\top U^\top. \tag{3.2}$$

Hence, $XX^\top$ and $\Sigma\Sigma^\top$ have the same eigenvalues, namely $\sigma_1^2 \geq \cdots \geq \sigma_r^2 > 0$ and 0 with multiplicity $n - r$. Multiplying both sides of (3.2) by U from the right yields:

$$XX^\top U = U\Sigma\Sigma^\top.$$

Interpreting this equation column-wise, we obtain:

$$XX^\top u_i = \sigma_i^2 u_i \quad \text{for } i = 1, \dots, r,$$
$$XX^\top u_i = 0 \cdot u_i \quad \text{for } i = r+1, \dots, n.$$

Thus, the first r left singular vectors u_i are eigenvectors of $XX^\top$ corresponding to the nonzero eigenvalues σ_i^2, while the remaining $n-r$ left singular vectors u_i correspond to the eigenvalue 0. Similarly, the first r right singular vectors v_i are eigenvectors of $X^\top X$ associated with the nonzero eigenvalues σ_i^2, whereas the remaining $m+1-r$ right singular vectors v_i correspond to the eigenvalue 0.

A generalization of the inverse of a square matrix to arbitrary $n \times (m+1)$ matrices is the *pseudoinverse*. It can be determined via SVD.

Definition 3.1 (Pseudo inverse)**.** Let $X = U\Sigma V^\top$ as in Theorem 3.1. Define X^+ by

$$X^+ := V\Sigma^+ U^\top \in \mathbb{R}^{(m+1)\times n},$$

where

$$\Sigma^+ := \left[\begin{array}{ccc|ccc} \sigma_1^{-1} & & & & \vdots & \\ & \ddots & & \dots & 0 & \dots \\ & & \sigma_r^{-1} & & \vdots & \\ \hline & \vdots & & & \vdots & \\ \dots & 0 & \dots & \dots & 0 & \dots \\ & \vdots & & & \vdots & \end{array}\right] \in \mathbb{R}^{(m+1)\times n}.$$

X^+ is called the pseudoinverse of X.

In MATLAB, the pseudoinverse of a matrix X can be computed using the built-in command `pinv(X)`.

The *Frobenius norm* is a matrix norm derived from the Euclidean norm and is commonly used to measure the error in linear regression problems, particularly in least squares regression. It is defined as

$$\|X\|_F := \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m+1}|X_{ij}|^2} \tag{3.3}$$

for $X \in \mathbb{R}^{n\times(m+1)}$. In the following lemma, we show that the Frobenius norm of X can be expressed using the *trace* of $X^\top X$, denoted by $\mathrm{tr}(X^\top X)$. Additionally, we prove its invariance under orthogonal transformations and provide an estimate relating it to the spectral norm.

Lemma 3.2. Let $X = U\Sigma V^\top$ as in Theorem 3.1. Then:

(i) $\|X\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m+1}|X_{ij}|^2} = \sqrt{\mathrm{tr}(X^\top X)}$,

(ii) $\|U^\top X\|_F = \|XV\|_F = \|X\|_F$,

(iii) $\|X\|_F = \sqrt{\sum_{i=1}^{r}\sigma_i^2}$ and $\sigma_1 \le \|X\|_F \le \sqrt{r}\sigma_1$.

Proof.

(i) By definition

$$\|X\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m+1} X_{ij}X_{ij} = \sum_{j=1}^{m+1}\sum_{i=1}^{n} X_{ji}^\top X_{ij} = \sum_{j=1}^{m+1}(X^\top X)_{jj} = \operatorname{tr}(X^\top X).$$

(ii) We have

$$\|U^\top X\|_F^2 = \operatorname{tr}(X^\top UU^\top X) = \operatorname{tr}(X^\top X) = \|X\|_F^2$$

and

$$\|XV\|_F^2 = \operatorname{tr}((XV)^\top XV) = \operatorname{tr}(XV(XV)^\top) = \operatorname{tr}(XX^\top) = \operatorname{tr}(X^\top X) = \|X\|_F^2.$$

(iii) Finally, we infer from (ii) that

$$\|X\|_F^2 = \|U\Sigma V^\top\|_F^2 = \|\Sigma\|_F^2 = \operatorname{tr}(\Sigma^\top\Sigma) = \sum_{i=1}^{r}\sigma_i^2.$$

The last claim follows directly from the fact that σ_1 is the largest singular value.

□

3.2 DMD Architecture

The initial setting for DMD is a (possibly nonlinear) dynamical system of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}x = f(t, x; \mu), \quad x(0) = x_0, \tag{3.4}$$

where $f\colon \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ represents the dynamics, $x(t) \in \mathbb{R}^n$ is the state of the dynamical system at time t and $\mu \in \mathbb{R}^p$ contains parameters of the system. Typically, $x(t)$ is a high-dimensional vector ($n \gg 1$) that arises from the total number of pixels in a given frame of a video stream for instance.
In general, it is not possible to construct a solution to the well-posed initial value problem (3.4). Therefore, numerical methods are used to approximate the solution. This is where DMD comes into play. The DMD algorithm even works if the dynamics $f(t, x; \mu)$ are unknown. Instead, snapshots $x^0, \ldots, x^m$ that represent measurements of $x(t)$ are sampled in regular time intervals Δt, such that $x^k \approx x(k\Delta t)$. This means that measurements of the system alone are sufficient to approximate the dynamics of the system and to predict future states.

The key assumption of DMD is that the dynamics of the system can be approximated with sufficient accuracy by a time-invariant linear map, that is,

$$\frac{\mathrm{d}}{\mathrm{d}t}x = \mathcal{A}x, \quad x(0) = x_0, \tag{3.5}$$

with an unknown matrix $\mathcal{A} \in \mathbb{R}^{n\times n}$. The linear dynamical system (3.5) has the well-known unique solution

$$x(t) := \exp(\mathcal{A}t)x_0 \quad \text{for } t \in \mathbb{R}. \tag{3.6}$$

We suppose that $\mathcal{A} \in \mathbb{R}^{n\times n}$ is symmetric, i.e., $\mathcal{A}$ can be diagonalized by an orthogonal matrix. More precisely, there exists an orthonormal basis $\hat{\phi}_1, \ldots, \hat{\phi}_n \in \mathbb{R}^n$ consisting of eigenvectors of $\mathcal{A}$ and corresponding eigenvalues $\omega_1, \ldots, \omega_n \in \mathbb{R}$ such that the spectral decomposition

$$\mathcal{A} = \hat{\Phi}\hat{\Omega}\hat{\Phi}^\top$$

holds, where

$$\begin{aligned}\hat{\Phi} &:= \begin{bmatrix} \hat{\phi}_1 & \cdots & \hat{\phi}_n \end{bmatrix} \in \mathbb{R}^{n\times n},\\ \hat{\Omega} &:= \operatorname{diag}(\omega_1, \ldots, \omega_n) \in \mathbb{R}^{n\times n}.\end{aligned}$$

The matrix exponential $\exp(\mathcal{A}t)$ can thus be expressed as

$$\exp(\mathcal{A}t) = \hat{\Phi}\exp(\hat{\Omega}t)\hat{\Phi}^\top. \tag{3.7}$$

Moreover, the initial condition x_0 can be represented as a linear combination of the basis vectors

$$x_0 = \sum_{k=1}^{n} \hat{b}_k \hat{\phi}_k, \tag{3.8}$$

where the coefficients $\hat{b}_1, \ldots, \hat{b}_n \in \mathbb{R}$ are chosen appropriately. By using (3.7) and (3.8), we can rewrite the solution formula (3.6) as

$$\begin{aligned}x(t) &= \exp(\mathcal{A}t)x_0\\ &= \hat{\Phi}\exp(\hat{\Omega}t)\hat{\Phi}^\top \sum_{k=1}^{n} \hat{b}_k\hat{\phi}_k\\ &= \sum_{k=1}^{n} \hat{b}_k \begin{bmatrix} \hat{\phi}_1 & \cdots & \hat{\phi}_n \end{bmatrix} \exp(\hat{\Omega}t) \begin{bmatrix} \hat{\phi}_1 & \cdots & \hat{\phi}_n \end{bmatrix}^\top \hat{\phi}_k\\ &= \sum_{k=1}^{n} \hat{b}_k \begin{bmatrix} \hat{\phi}_1 & \cdots & \hat{\phi}_n \end{bmatrix} \operatorname{diag}(\exp(\omega_1 t), \ldots, \exp(\omega_n t))\tilde{e}_k\\ &= \sum_{k=1}^{n} \hat{b}_k \begin{bmatrix} \hat{\phi}_1 & \cdots & \hat{\phi}_n \end{bmatrix} \exp(\omega_k t)\tilde{e}_k\\ &= \sum_{k=1}^{n} \hat{\phi}_k \exp(\omega_k t)\hat{b}_k,\end{aligned}$$

where $\tilde{e}_k$ denotes the k-th canonical basis vector in $\mathbb{R}^n$. In terms of matrices, we can express the solution (3.6) as

$$x(t) = \hat{\Phi}\exp(\hat{\Omega}t)b,$$

where the coefficient vector $b \in \mathbb{R}^n$ is defined as

$$b := \begin{bmatrix} \hat{b}_1 & \cdots & \hat{b}_n \end{bmatrix}^\top \in \mathbb{R}^n.$$

By setting

$$A := \exp(\mathcal{A}\Delta t) \in \mathbb{R}^{n\times n}.$$

and using

$$\begin{aligned} x^{k+1} &\approx x((k+1)\Delta t) \\ &= \exp(\mathcal{A}(k+1)\Delta t)x_0 \\ &= \exp(\mathcal{A}k\Delta t)\exp(\mathcal{A}\Delta t)x_0, \\ x^k &\approx x(k\Delta t) \\ &= \exp(\mathcal{A}k\Delta t)x_0 \end{aligned}$$

the time-discretization with time step size Δt of the continuous system (3.5) is given as

$$x^{k+1} = Ax^k.$$

Since A, assuming that A is symmetric, is orthogonally diagonalizable, there exists a spectral decomposition

$$A = \Phi\Lambda\Phi^\top,$$

where

$$\begin{aligned} \Phi &:= \begin{bmatrix} \phi_1 & \cdots & \phi_n \end{bmatrix} \in \mathbb{R}^{n\times n}, \\ \Lambda &:= \operatorname{diag}(\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^{n\times n} \end{aligned}$$

are the matrices containing the orthonormal eigenvectors and eigenvalues of A, respectively. We obtain the solution formula

$$x^{k+1} = Ax^k = A^{k+1}x^0 = (\Phi\Lambda\Phi^\top)^{k+1}x^0 = \Phi\Lambda^{k+1}\Phi^\top x^0. \tag{3.9}$$

We approximate the initial condition by

$$x^0 \approx x(0) = x_0 = \sum_{k=1}^{n} b_k\phi_k = \Phi b,$$

where

$$b := \begin{bmatrix} b_1 & \cdots & b_n \end{bmatrix}^\top \in \mathbb{R}^n$$

is the coordinate vector of x_0 in the orthonormal basis $\phi_1, \ldots, \phi_n$. Using the orthogonality of Φ, we can rewrite (3.9) as

$$x^{k+1} = \Phi\Lambda^{k+1}b.$$

Finally, the goal of the DMD method is to construct A such that

$$\|x^{k+1} - Ax^k\|_2$$

is minimized across all $k = 0, \ldots, m-1$. This minimization problem can be rewritten with

$$X := \begin{bmatrix} x^0 & \cdots & x^{m-1} \end{bmatrix} \in \mathbb{R}^{n\times m}, \tag{3.10a}$$

$$Y := \begin{bmatrix} x^1 & \cdots & x^m \end{bmatrix} \in \mathbb{R}^{n\times m}, \tag{3.10b}$$

as

$$\min_{A\in\mathbb{R}^{n\times n}} \|Y - AX\|_F^2. \tag{3.11}$$

The solution to (3.11) is given by the matrix

$$YX^+ \in \mathbb{R}^{n\times n}. \tag{3.12}$$

Remark 3.1.

(i) We prove that (3.12) is the solution to (3.11).

Proof. Using matrix vectorization and the Kronecker product $\otimes$ (see [26]), we obtain

$$\begin{aligned}
\bar{A} = \operatorname*{argmin}_{A\in\mathbb{R}^{n\times n}} \|Y - AX\|_F^2 &\iff \bar{A} = \operatorname*{argmin}_{A\in\mathbb{R}^{n\times n}} \|\operatorname{vec}(Y) - \operatorname{vec}(AX)\|_2^2 \\
&\iff \bar{A} = \operatorname*{argmin}_{A\in\mathbb{R}^{n\times n}} \|\operatorname{vec}(Y) - (X^\top \otimes I_n)\operatorname{vec}(A)\|_2^2 \\
&\iff \operatorname{vec}(\bar{A}) = \operatorname*{argmin}_{a\in\mathbb{R}^{n^2}} \|\operatorname{vec}(Y) - (X^\top \otimes I_n)a\|_2^2,
\end{aligned}$$

The solution to this minimization problem is given by $(X^\top \otimes I_n)^+ \operatorname{vec}(Y)$ according to [8, Satz 2.28]. Since $(X^\top \otimes I_n)^+ = ((X^\top)^+ \otimes I_n^+)$ (cf. [17]) and using some basic properties of the pseudoinverse, we have

$$\begin{aligned}
\operatorname{vec}(\bar{A}) &= (X^\top \otimes I_n)^+ \operatorname{vec}(Y) \\
&= ((X^\top)^+ \otimes I_n^+) \operatorname{vec}(Y) \\
&= ((X^+)^\top \otimes I_n) \operatorname{vec}(Y) \\
&= \operatorname{vec}(YX^+).
\end{aligned}$$

Thus, $\bar{A} := YX^+$ is the solution to (3.11). □

(ii) Typically, the snapshots x^k are high-dimensional such that the matrix X is computationally expensive to calculate by the above product. Also the decomposition of X is very costly. Instead, it is convenient to compute a low rank approximation of X as demonstrated in Section 3.3.

3.3 DMD Algorithm

The following provides a step-by-step description of the DMD algorithm, followed by a pseudocode summary.

1. Assemble the matrices X and Y as in (3.10a) and (3.10b).
2. Compute the truncated SVD of X:

$$X \approx U\Sigma V^\top,$$

where $U \in \mathbb{R}^{n\times r}$, $\Sigma \in \mathbb{R}^{r\times r}$, $V \in \mathbb{R}^{m\times r}$ and $r \leq \operatorname{rk}(X)$.

3. Define the matrix A by

$$A := YV\Sigma^{-1}U^\top \approx YX^+ \in \mathbb{R}^{n\times n}. \tag{3.13}$$

It is worth noting that Σ^{-1} is well-defined, since Σ is a diagonal matrix with only nonzero entries on the diagonal, as r is chosen to be smaller than $\mathrm{rk}(X)$.

As noted in Remark 3.1, computing A directly can be inefficient when n is large. Instead, one considers the projected matrix $\tilde{A} \in \mathbb{R}^{r\times r}$, obtained by projecting A onto the subspace spanned by the leading r left singular vectors of X:

$$\tilde{A} := U^\top AU = U^\top YV\Sigma^{-1}U^\top U = U^\top YV\Sigma^{-1}. \tag{3.14}$$

Note that $U^\top U = I_r$, where I_r denotes the $r \times r$ identity matrix, since the columns of U are pairwise orthonormal. This approach is also often referred to as the *projected DMD algorithm*.

4. Determine the spectral decomposition of $\tilde{A}$, assuming $\tilde{A}$ is symmetric:

$$\tilde{A}W = W\Lambda. \tag{3.15}$$

Here, the matrix W contains the eigenvectors of $\tilde{A}$, given explicitly by

$$W := \begin{bmatrix} w_1 & \cdots & w_r \end{bmatrix} \in \mathbb{R}^{r\times r},$$

while the diagonal matrix

$$\Lambda := \mathrm{diag}(\lambda_1, \ldots, \lambda_r) \in \mathbb{R}^{r\times r}$$

contains the corresponding eigenvalues.

If $\tilde{A}$ is not symmetric, the computed matrices W and Λ are not necessarily real. We therefore proceed with complex-valued computations. At the end of the computation, only the real part of the result is retained.

5. Compute the eigenvectors of A by

$$\begin{aligned} \Phi &:= \begin{bmatrix} \phi_1 & \cdots & \phi_r \end{bmatrix} \\ &:= YV\Sigma^{-1}W \in \mathbb{C}^{n\times r}. \end{aligned} \tag{3.16}$$

It is proven in [25, Theorem 1] that each pair (λ_k, ϕ_k), for $k = 1, \ldots, r$, is indeed an *eigenpair* of A, provided the SVD of X is computed exactly, i.e., if $r \geq \mathrm{rk}(X)$. Moreover, all nonzero eigenvalues of A are captured. Using (3.13), (3.16), (3.14) and (3.15), we observe that

$$\begin{aligned} A\Phi &= YV\Sigma^{-1}U^\top\Phi \\ &= YV\Sigma^{-1}U^\top YV\Sigma^{-1}W \\ &= YV\Sigma^{-1}\tilde{A}W \\ &= YV\Sigma^{-1}W\Lambda \\ &= \Phi\Lambda. \end{aligned}$$

6. Determine the coefficient vector b by

$$b \approx \Phi^{+} x_0 \in \mathbb{C}^r$$

and reconstruct the snapshots via

$$\tilde{x}^k := \Phi \Lambda^k b \in \mathbb{C}^n \quad \text{for } k = 0, \ldots, m. \tag{3.17}$$

If the initial condition satisfies $x_0 = 0$, then the coefficient vector b and, consequently, all reconstructed snapshots $\tilde{x}^k$ vanish. In this case, it may be helpful to set $\tilde{x}^0 := 0$ and to initialize the algorithm using data matrices $X, Y \in \mathbb{R}^{n \times (m-1)}$ shifted by one time step. Accordingly, the vector b can be approximated by $b \approx \Phi^{+} x^1$ and the reconstructed snapshots $\tilde{x}^k$, $k = 1, \ldots, m$, can be computed as in (3.17).

The pseudocode algorithm for the DMD method is given in Algorithm 1.

Algorithm 1 (Exact) DMD

1: **INPUT** $X_{\text{in}} \in \mathbb{R}^{n \times (m+1)}$, $r \in \mathbb{N}$
2: Assemble $X, Y \in \mathbb{R}^{n \times m}$ as in (3.10a) and (3.10b), respectively
3: Compute the truncated SVD of X, $X \approx U \Sigma V^{\top}$
4: Define $\tilde{A} = U^{\top} Y V \Sigma^{-1} \in \mathbb{R}^{r \times r}$
5: Determine the spectral decomposition of $\tilde{A}$, $\tilde{A} W = W \Lambda$
6: Compute the eigenvalues of A, $\Phi = Y V \Sigma^{-1} W \in \mathbb{C}^{n \times r}$
7: Determine the coefficient vector $b \approx \Phi^{+} x_0 \in \mathbb{C}^r$
8: Reconstruct the snapshots via $\tilde{x}^k = \Phi \Lambda^k b \in \mathbb{C}^n$, $k = 0, \ldots, m$
9: **RETURN** $\text{Re}(\Phi) \in \mathbb{R}^{n \times r}$, $\text{Re}(\Lambda) \in \mathbb{R}^{r \times r}$, $\text{Re}(\begin{bmatrix} \tilde{x}^0 & \cdots & \tilde{x}^m \end{bmatrix}) \in \mathbb{R}^{n \times (m+1)}$

3.4 Randomized Dimensionality Reduction Approach

As the amount of data generated by experiments and simulations has increased, deterministic algorithms for extracting dominant coherent structures have become more computationally expensive. Fortunately, many datasets exhibit low-rank structures, which indicates that they contain a significant amount of redundant information.
Therefore, randomized numerical techniques have emerged as an efficient approach for matrix decompositions. They generate a low-dimensional version of the original data matrix, called a *sketch*. This sketch retains the critical features of the data while significantly reducing the size, enabling an efficient computation of an approximate low-rank factorization of the data matrix.

3.4.1 Randomized QB Decomposition

Given a data matrix $X \in \mathbb{R}^{n \times (m+1)}$ and a target rank $r \ll min\{n, m+1\}$, the goal is to compute a near-optimal orthonormal basis $Q \in \mathbb{R}^{n \times r}$ in the sense that

$$X \approx Q Q^{\top} X.$$

In order to achieve this, we first construct a random test matrix $\Omega \in \mathbb{R}^{(m+1) \times r}$ drawn from the normal Gaussian distribution and set

$$Y := X \Omega \in \mathbb{R}^{n \times r}.$$

The matrix Y can be decomposed into an orthogonal matrix $\tilde{Q} \in \mathbb{R}^{n \times n}$ and an upper triangular matrix $\tilde{R} \in \mathbb{R}^{n \times r}$ by the *QR decomposition* (cf. [8, Satz 3.47]):

$$Y = \tilde{Q}\tilde{R}.$$

Since we have $n > r$, the matrix $\tilde{R}$ is of the form

$$\tilde{R} = \begin{bmatrix} R \\ 0_{(n-r)\times r} \end{bmatrix} \text{ with } R = \begin{bmatrix} * & * & \cdots & * \\ & * & \cdots & * \\ & & \ddots & \vdots \\ & & & * \end{bmatrix} \in \mathbb{R}^{r \times r}.$$

Here, $0_{(n-r)\times r}$ denotes the zero matrix of size $(n-r) \times r$. Hence, we obtain the desired basis Q by the *reduced QR decomposition of* Y:

$$Y = QR,$$

where Q is the matrix consisting of the first r columns of $\tilde{Q}$.

The objective is to derive a smaller matrix B from the high-dimensional matrix X. Therefore, we use Q to project X onto the low-dimensional space

$$B := Q^\top X \in \mathbb{R}^{r \times (m+1)}.$$

As a result, the matrix X can be approximated via

$$X \approx QB.$$

Oversampling is a technique employed to improve the quality of the sampled matrix Y by slightly increasing the number of columns of Ω. Thus, we introduce an oversampling parameter p and set $l = r + p$. The test matrix Ω is hence a matrix of size $(m+1) \times l$.

Another strategy used to improve the accuracy of Y is the so-called *power iteration scheme*. Here, the matrix Y is calculated by

$$Y := ((XX^\top)^q X)\Omega,$$

where q is the number of iterations. Taking into account the SVD of X, $X = U\Sigma V^\top$, the product $(XX^\top)^q X$ can be simplified to

$$(XX^\top)^q X = U\Sigma^{2q+1}V^\top.$$

Compared to the standard approach, the power iteration scheme accelerates the growth and decay of the singular values of X, respectively. More precisely, the dominant singular values are amplified, while the smaller ones are suppressed. Since the small singular values gain the noise in the data, and the dominant ones contain the essential information, this technique increases the quality of Y. Even a small number of q is sufficient for this purpose. However, this method requires more computational effort than the standard approach, because the matrix X must be multiplied by its transpose several times.

Before we present the pseudocode for the randomized QB decomposition with oversampling and power iteration scheme, we make two points.

Remark 3.2.

(i) A straightforward execution of the power iteration method, as previously described, can suffer from numerical instability because of round-off errors. To counteract this, the sampling matrix Y is orthogonalized between each computational step, which helps to improve stability.

(ii) Recommended default values for the oversampling parameter are $p = 10$ and for the power iteration count, $q = 2$ (cf. [10, Remark 3.2]).

The randomized QB decomposition is outlined in Algorithm 2.

Algorithm 2 Randomized QB decomposition (rQB)

INPUT $X \in \mathbb{R}^{n\times(m+1)}$, $r, p, q \in \mathbb{N}$
Compute $l = r + p$
Generate a random Gaussian matrix $\Omega \in \mathbb{R}^{(m+1)\times l}$
Compute the sampling matrix $Y = X\Omega \in \mathbb{R}^{n\times l}$
for $j = 1, \ldots, q$ **do**
    Compute the reduced QR decomposition of Y, $Y = QR$
    Compute the reduced QR decomposition of $X^\top Q$, $X^\top Q = Z\tilde{R}$
    Compute $Y = XZ$
end for
Compute the reduced QR decomposition of Y, $Y = QR$
Compute $B = Q^\top X \in \mathbb{R}^{l\times(m+1)}$
RETURN $Q \in \mathbb{R}^{n\times l}$, $B \in \mathbb{R}^{l\times(m+1)}$

3.4.2 Randomized DMD

To enhance the efficiency of the DMD algorithm, we combine it with the randomized QB decomposition. First, the randomized QB decomposition is performed on the data matrix $X_{\text{in}} \in \mathbb{R}^{n\times(m+1)}$. This yields the approximation

$$X_{\text{in}} \approx QB,$$

where $Q \in \mathbb{R}^{n\times l}$ and

$$B := \begin{bmatrix} b_0 & \cdots & b_m \end{bmatrix} \in \mathbb{R}^{l\times(m+1)}$$

are the resulting matrices from Algorithm 2. It is worth noting that the size of B is significantly smaller than that of X_{in}. We assume that the approximations $X \approx QQ^\top X$ and $Y \approx QQ^\top Y$ hold for the matrices

$$X := \begin{bmatrix} x^0 & \cdots & x^{m-1} \end{bmatrix},$$
$$Y := \begin{bmatrix} x^1 & \cdots & x^m \end{bmatrix},$$

respectively. Hence, we can approximate the best-fit matrix A by

$$A := (QQ^\top Y)(QQ^\top X)^+ = (QB_2)(QB_1)^+ = QB_2B_1^+Q^+ \approx YX^+$$

where $B_1 := Q^\top X$ and $B_2 := Q^\top Y$. Introducing the low-dimensional matrix

$$A_B := B_2B_1^+ \in \mathbb{R}^{l\times l},$$

we obtain

$$A = QA_BQ^+. \tag{3.18}$$

Next, using the rank r truncated SVD of B_1, $B_1 \approx U\Sigma V^\top$, we define the projected matrix $\tilde{A}_B$ via

$$\tilde{A}_B := U^\top B_2 V\Sigma^{-1} = U^\top B_2 V\Sigma^{-1}U^\top U \approx U^\top A_B U \in \mathbb{R}^{r\times r}.$$

The spectral decomposition of $\tilde{A}_B$ is then computed as

$$\tilde{A}_B W = W\Lambda, \tag{3.19}$$

assuming that $\tilde{A}_B$ is symmetric. We observe that the columns of $\hat{W} := B_2V\Sigma^{-1}W$ approximate the eigenvectors of A_B:

$$\begin{aligned} A_B\hat{W} &= B_2B_1^+\hat{W} \\ &\approx B_2V\Sigma^{-1}U^\top B_2V\Sigma^{-1}W \\ &= B_2V\Sigma^{-1}\tilde{A}_BW \\ &= B_2V\Sigma^{-1}W\Lambda \\ &= \hat{W}\Lambda \end{aligned}$$

Multiplying (3.18) by $Q\hat{W}$ and using (3.19), we obtain

$$\begin{aligned} AQ\hat{W} &= QA_BQ^+Q\hat{W} \\ &= QA_B\hat{W} \\ &\approx Q\hat{W}\Lambda. \end{aligned}$$

Hence, the approximated eigenvectors of A are given by $Q\hat{W}$ and the corresponding approximated eigenvalues are given by Λ.

Algorithm 3 summarizes the randomized DMD algorithm.

Algorithm 3 Randomized DMD (rDMD)

1: **INPUT** $X_{\text{in}} \in \mathbb{R}^{n\times(m+1)}$, $r, p, q \in \mathbb{N}$
2: Perform the randomized QB decomposition on X_{in} to obtain $Q \in \mathbb{R}^{n\times l}$ and $B \in \mathbb{R}^{l\times(m+1)}$ according to Algorithm 2
3: Assemble $B_1 = \begin{bmatrix} b_0 & \cdots & b_{m-1} \end{bmatrix} \in \mathbb{R}^{l\times m}$ and $B_2 = \begin{bmatrix} b_1 & \cdots & b_m \end{bmatrix} \in \mathbb{R}^{l\times m}$
4: Compute the rank r truncated SVD of B_1, $B_1 \approx U\Sigma V^\top$
5: Define $\tilde{A}_B = U^\top B_2 V\Sigma^{-1} \in \mathbb{R}^{r\times r}$
6: Determine the spectral decomposition of $\tilde{A}_B$, $\tilde{A}_BW = W\Lambda$
7: Approximate the eigenvalues of A, $\Phi \approx QB_2V\Sigma^{-1}W \in \mathbb{C}^{n\times r}$
8: Compute the coefficient vector $b \approx \Phi^+x_0 \in \mathbb{C}^r$
9: Reconstruct the snapshots via $\tilde{x}^k = \Phi\Lambda^k b \in \mathbb{C}^n$, $k = 0, \ldots, m$
10: **RETURN** $\mathrm{Re}(\Phi) \in \mathbb{R}^{n\times r}$, $\mathrm{Re}(\Lambda) \in \mathbb{R}^{r\times r}$, $\mathrm{Re}(\begin{bmatrix} \tilde{x}^0 & \cdots & \tilde{x}^m \end{bmatrix}) \in \mathbb{R}^{n\times(m+1)}$

3.5 Example: Spatio-Temporal Cosine Pattern

As an example, we perform DMD and rDMD on a dataset generated from the spatio-temporal cosine function

$$f(t, x) := \cos(t + x).$$

There are $m + 1 := 1000$ snapshots taken equidistant, where the first snapshot is taken at $t_0 := 0$ and the last one at $T := 2\pi$. Each snapshot consists of $n_x := 2000$ equidistant points in the interval $[0, 2\pi]$. Accordingly, the snapshot matrix X_{in} is of size 2000×1000.

First, we take a look at the first 50 singular values of the data matrix (cf. Figure 2).

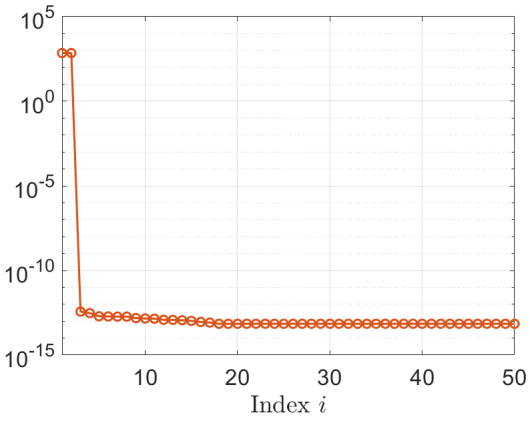


Figure 2: Decay of the first 50 singular values of the input matrix X_{in}.

Only the first two singular values are significantly larger than the others, which are close to zero. This suggests that a rank truncation value of $r := 2$ is appropriate.

Next, we reconstruct the data matrix using the exact DMD algorithm and the randomized DMD algorithm, where we set the oversampling parameter to $p := 10$ and the number of power iterations to $q := 2$. The results are visualized in Figure 3.

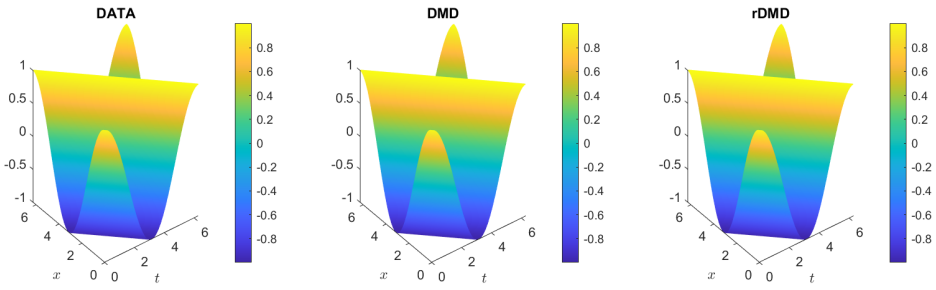


Figure 3: Left: original data matrix X_{in}. Middle: reconstructed data matrix using the DMD algorithm. Right: reconstructed data matrix using the rDMD algorithm.

We observe that the DMD algorithm and the randomized DMD algorithm are able to approximate the exact data very well. A measure for the quality of the reconstructed

matrix X_{rec} is the *relative Frobenius error* defined as

$$\mathcal{E}(X_{\text{rec}}, r) = \frac{\|X_{\text{in}} - X_{\text{rec}}\|_F}{\|X_{\text{in}}\|_F}, \tag{3.20}$$

which clearly depends on the rank truncation value r. Table 1 shows that both algorithms achieve a very low relative Frobenius error. The randomized DMD (rDMD) algorithm is slightly faster than the exact DMD while maintaining a comparable level of accuracy.

Table 1: Relative Frobenius error and running time of the DMD and rDMD algorithms. The data is generated with a running time of 0.012 seconds.

	DMD	rDMD
$\mathcal{E}(X_{\text{rec}}, r)$	$2.195 \cdot 10^{-13}$	$3.143 \cdot 10^{-13}$
Running time $[s]$	0.394	0.067

4 Piecewise DMD

The foundation of this section relies on the work of [1], [2] and [7].

Although we have seen in Section 3.5 that the DMD/rDMD algorithm is able to reconstruct the data very well, this is not the case for all types of data. There are at least two classes of datasets, where the DMD/rDMD algorithm fails to provide a good approximation. The first involves data that exhibit periodic behavior, while the second includes data that describe spatio-temporal Turing instabilities.
We take a closer look at these two classes and introduce an enhanced DMD method that addresses these shortcomings, the so-called *piecewise DMD (pDMD) method*. By utilizing some error indicators, we demonstrate through several examples how this method can effectively detect and correct the issues that arise with the DMD/rDMD method from Section 3.

4.1 Reaction-Diffusion Systems

In order to illustrate the limitations of the DMD/rDMD method, it is first necessary to create suitable datasets. We generate them by solving *reaction-diffusion PDE (RD-PDE) systems* in one or two space dimensions for different parameter choices.
Let $\Omega \subset \mathbb{R}^{d_\Omega}$, $d_\Omega \in \{1, 2\}$, be a non-empty and bounded domain with Lipschitz-continuous boundary $\Gamma := \partial\Omega$. For $T > 0$, we define

$$\begin{aligned} \Omega_T &:= (0, T] \times \Omega, \\ \Gamma_T &:= [0, T] \times \Gamma. \end{aligned}$$

The RD-PDE systems are of the form

$$\begin{aligned} u_t - \mu_u \Delta u + d_u(u, v) &= 0, \quad \text{in } \Omega_T, \\ v_t - \mu_v \Delta v + d_v(u, v) &= 0, \quad \text{in } \Omega_T, \\ \nabla u \cdot n &= g_u, \quad \text{on } \Gamma_T, \\ \nabla v \cdot n &= g_v, \quad \text{on } \Gamma_T, \\ u(0) &= u_0, \quad \text{in } \Omega, \\ v(0) &= v_0, \quad \text{in } \Omega. \end{aligned} \tag{4.1}$$

System (4.1) is also referred to as a semilinear *initial-boundary value problem (IBVP)*. Here, u and v are real-valued functions, $u, v \colon \Omega_T \to \mathbb{R}$. The diffusion constants are denoted by $\mu_u, \mu_v \in \mathbb{R}_{>0}$ and the term Δu is defined by the Laplace operator

$$\Delta u(t, x) := \sum_{j=1}^{d_\Omega} \frac{\partial^2 u}{\partial x_j^2}(t, x) \quad \text{for } (t, x) \in \Omega_T, \tag{4.2}$$

analogously Δv. In the one-dimensional case, ∇u and ∇v denote the derivatives of u and v, respectively, while in the two-dimensional case, they represent the spatial gradients of u and v. Moreover, $d_u, d_v \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ are given nonlinear functions and they account for physical, chemical or biological processes (e.g., see [11] and [18] for applications in biomedicine). We suppose Neumann boundary conditions, where n denotes the outward unit normal vector and $g_u, g_v \colon \Gamma_T \to \mathbb{R}$ are given boundary functions. The

function value of u on the boundary is thus not fixed. Finally, $u_0, v_0 \colon \Omega \to \mathbb{R}$ are given initial functions.

For the discretization of (4.1), we employ the *finite difference method (FDM)*. In a first step, we apply the *method of lines (MOL)* to discretize in space only, which yields a system of ODEs where time remains continuous. In a second step, we discretize this ODE system in time, again using finite differences – specifically, the *IMEX Euler method*, in which the diffusion terms are treated implicitly, while the (nonlinear) reaction terms are handled explicitly.

4.2 Limitations of DMD

In this section, we demonstrate how the DMD algorithms introduced in Section 3 fail when applied to the two classes of datasets discussed above.
For the remainder of Section 4, we employ the rDMD algorithm with oversampling parameter $p := 0$ and number of power iterations $q := 2$ to illustrate the limitations of DMD and rDMD. While we could have also used the exact DMD algorithm for this purpose, we opted for rDMD as it is generally faster and yields comparable results.
Furthermore, when referring to the rDMD method from now on, we often use the term *global*. This terminology becomes clearer once we introduce the piecewise DMD approach, which, in a certain sense, localizes the global rDMD method.

4.2.1 Limitations on Periodic Datasets

FitzHugh-Nagumo System: Limit Cycle
First, we consider the one-dimensional *FitzHugh-Nagumo (FHN) system*, which models the activation and deactivation dynamics of a spiking neuron. The domain is given by $\Omega := (0, 1)$ and the final time is $T := 6$. Moreover, we set

$$\begin{aligned} \mu_u &:= 1.5 \cdot 10^{-2}, \\ \mu_v &:= 0, \end{aligned}$$

i.e., the function v does not diffuse. The nonlinear functions d_u and d_v are defined as

$$\begin{aligned} d_u(u, v) &:= -\frac{u(u - 0.1)(1 - u)}{\mu_u} + \frac{v}{\mu_u} - \frac{c}{\mu_u}, \\ d_v(u, v) &:= -bu + \gamma v - c, \end{aligned}$$

where

$$\begin{aligned} b &:= 0.5, \\ \gamma &:= 2, \\ c &:= 0.05. \end{aligned}$$

To get rid of the fractions in the nonlinearity d_u, we multiply the first equation of (4.1) by μ_u, relabel d_u as

$$d_u(u, v) := -u(u - 0.1)(1 - u) + v - c$$

and obtain

$$\mu_u u_t - \mu_u^2 \Delta u + d_u(u, v) = 0.$$

The boundary function for u is given by

$$g_u(t,x) := \begin{cases} -(5 \cdot 10^4 t^3 e^{-15t}), & \text{if } x = 0, \\ 0, & \text{if } x = 1 \end{cases} \tag{4.3}$$

for $t \in [0,T]$. It is not necessary to specify boundary conditions for v, as the vanishing diffusion constant ensures that during discretization this has no effect. Furthermore, we define the initial conditions by

$$\begin{aligned} u_0 &:= 0 \quad \text{in } \Omega, \\ v_0 &:= 0 \quad \text{in } \Omega, \end{aligned}$$

corresponding to vanishing and spatially constant initial data.

The spatial discretization is performed with $n_x := 1024$ equidistant interior points in the interval Ω. Hence, the spatial step size is

$$\Delta x := \frac{1}{n_x + 1}.$$

The number of discrete problems to be solved is $m := 6 \cdot 10^3$, yielding a time step size of $\Delta t := {}^{T}/_{m} = 10^{-3}$. Thus, the time discretization is also performed on an equidistant grid. Accordingly, the snapshot matrix X_{in} is of size 2048×6001.

The application of the MOL leads to the following *full order model (FOM)* of ODEs, where time remains continuous:

$$\begin{aligned} \mu_u \dot{\mathbf{u}} - \mu_u^2(\mathbf{A}\mathbf{u} + \mathbf{r}) + \mathbf{d}_{\mathbf{u}}(\mathbf{u},\mathbf{v}) &= 0_{n_x} \quad \text{for } t \in (0,T], \\ \dot{\mathbf{v}} + \mathbf{d}_{\mathbf{v}}(\mathbf{u},\mathbf{v}) &= 0_{n_x} \quad \text{for } t \in (0,T], \\ \mathbf{u}(0) &= \mathbf{u}_0, \\ \mathbf{v}(0) &= \mathbf{v}_0. \end{aligned} \tag{4.4}$$

Here, the vectors

$$\begin{aligned} \mathbf{u} &:= \mathbf{u}(t) \\ &:= \begin{bmatrix} \mathbf{u}_1(t) & \cdots & \mathbf{u}_{n_x}(t) \end{bmatrix}^\top, \\ \mathbf{v} &:= \mathbf{v}(t) \\ &:= \begin{bmatrix} \mathbf{v}_1(t) & \cdots & \mathbf{v}_{n_x}(t) \end{bmatrix}^\top \end{aligned}$$

represent the spatially discretized solutions u and v, respectively, where

$$\begin{aligned} \mathbf{u}_i(t) &\approx u(t, x_i), \\ \mathbf{v}_i(t) &\approx v(t, x_i) \end{aligned}$$

are the approximations of the continuous solutions at the spatial grid points x_i. The vector 0_{n_x} denotes the zero vector of size n_x. Similarly, the initial conditions are given by

$$\begin{aligned} \mathbf{u}_0 &:= \mathbf{u}(0) \\ &:= \begin{bmatrix} u_0(x_1) & \cdots & u_0(x_{n_x}) \end{bmatrix}^\top, \end{aligned}$$

$$\begin{aligned}\mathbf{v}_0 &:= \mathbf{v}(0)\\ &:= \begin{bmatrix} v_0(x_1) & \cdots & v_0(x_{n_x}) \end{bmatrix}^\top.\end{aligned}$$

The matrix $\mathbf{A} \in \mathbb{R}^{n_x \times n_x}$ and the vector $\mathbf{r} \in \mathbb{R}^{n_x}$ are derived from the spatial discretization of the Laplace operator and the boundary conditions, where the second-order central difference quotient

$$\Delta u(t, x_i) \approx \frac{1}{\Delta x^2} \left(u(t, x_{i-1}) - 2u(t, x_i) + u(t, x_{i+1}) \right)$$

is used for $i = 1, \ldots, n_x$. It remains to specify how to handle the boundary terms $u(t, x_0)$ and $u(t, x_{n_x+1})$. This is where the Neumann boundary conditions are applied. We have

$$\nabla u \cdot n = g_u \iff \begin{cases} \frac{\partial u}{\partial x}(t, x_0) = -g_u(t, x_0), \\ \frac{\partial u}{\partial x}(t, x_{n_x+1}) = g_u(t, x_{n_x+1}), \end{cases}$$

since we are considering the outward unit normal vector. By applying a second-order forward and backward difference quotient for the spatial derivative at the left and right boundary, respectively, we obtain

$$\begin{aligned} -g_u(t, x_0) &= \frac{\partial u}{\partial x}(t, x_0) \\ &\approx \frac{u(t, x_2) - 4u(t, x_1) + 3u(t, x_0)}{2\Delta x}, \\ g_u(t, x_{n_x+1}) &= \frac{\partial u}{\partial x}(t, x_{n_x+1}) \\ &\approx \frac{u(t, x_{n_x-1}) - 4u(t, x_{n_x}) + 3u(t, x_{n_x+1})}{2\Delta x}. \end{aligned}$$

Taking into account (4.3), we deduce

$$\begin{aligned} u(t, x_0) &\approx -\frac{1}{3}u(t, x_2) + \frac{4}{3}u(t, x_1) - \frac{2}{3}\Delta x g_u(t, 0) \\ &= -\frac{1}{3}u(t, x_2) + \frac{4}{3}u(t, x_1) + \frac{2}{3}\Delta x (5 \cdot 10^4 t^3 e^{-15t}), \\ u(t, x_{n_x+1}) &\approx -\frac{1}{3}u(t, x_{n_x-1}) + \frac{4}{3}u(t, x_{n_x}) + \frac{2}{3}\Delta x g_u(t, 1) \\ &= -\frac{1}{3}u(t, x_{n_x-1}) + \frac{4}{3}u(t, x_{n_x}). \end{aligned}$$

Thus, the matrix $\mathbf{A}$ is of the form

$$\mathbf{A} := \frac{1}{\Delta x^2} \begin{bmatrix} -\frac{2}{3} & \frac{2}{3} & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & \frac{2}{3} & -\frac{2}{3} \end{bmatrix} \in \mathbb{R}^{n_x \times n_x},$$

while the time-dependent vector $\mathbf{r}$ is given by

$$\begin{aligned}\mathbf{r} &:= \mathbf{r}(t)\\ &:= \frac{1}{\Delta x^2}\begin{bmatrix} \frac{2}{3}\Delta x(50000t^3e^{-15t}) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{n_x}.\end{aligned}$$

Finally, the two functions

$$\begin{aligned}\mathbf{d_u}&\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_x},\\ \mathbf{d_v}&\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}\end{aligned}$$

are defined analogously to d_u and d_v in a component-wise manner.

To discretize the ODE system (4.4) in time, we employ the IMEX Euler method in vector form. We begin by introducing an equidistant time grid

$$t_k := k\Delta t \quad \text{for } k = 0, \dots, m$$

and approximate the time derivatives using the first-order backward difference quotients

$$\begin{aligned}\dot{\mathbf{u}}(t_k) &\approx \frac{\mathbf{u}^k - \mathbf{u}^{k-1}}{\Delta t},\\ \dot{\mathbf{v}}(t_k) &\approx \frac{\mathbf{v}^k - \mathbf{v}^{k-1}}{\Delta t}\end{aligned}$$

for $k = 1, \dots, m$, where $\mathbf{u}^k \approx \mathbf{u}(t_k)$ and $\mathbf{v}^k \approx \mathbf{v}(t_k)$. If we further treat the diffusion terms implicitly and the reaction terms explicitly, we obtain the system

$$\begin{aligned}\mu_u\frac{\mathbf{u}^k - \mathbf{u}^{k-1}}{\Delta t} - \mu_u^2(\mathbf{A}\mathbf{u}^k + \mathbf{r}^k) + \mathbf{d_u}(\mathbf{u}^{k-1}, \mathbf{v}^{k-1}) &= 0_{n_x}, \quad \text{for } k = 1, \dots, m,\\ \frac{\mathbf{v}^k - \mathbf{v}^{k-1}}{\Delta t} + \mathbf{d_v}(\mathbf{u}^{k-1}, \mathbf{v}^{k-1}) &= 0_{n_x}, \quad \text{for } k = 1, \dots, m,\\ \mathbf{u}^0 &= \mathbf{u}_0,\\ \mathbf{v}^0 &= \mathbf{v}_0.\end{aligned}$$

Rearranging leads to the fully discretized system

$$\begin{aligned}\left(\mu_u I_{n_x} - \Delta t\mu_u^2\mathbf{A}\right)\mathbf{u}^k &= \mu_u\mathbf{u}^{k-1} - \Delta t\mathbf{d_u}(\mathbf{u}^{k-1}, \mathbf{v}^{k-1}) + \Delta t\mu_u^2\mathbf{r}^k, \quad \text{for } k = 1, \dots, m,\\ \mathbf{v}^k &= \mathbf{v}^{k-1} - \Delta t\mathbf{d_v}(\mathbf{u}^{k-1}, \mathbf{v}^{k-1}), \quad \text{for } k = 1, \dots, m,\\ \mathbf{u}^0 &= \mathbf{u}_0,\\ \mathbf{v}^0 &= \mathbf{v}_0.\end{aligned}$$

In block form, this time-stepping scheme can be written as the linear system

$$\underbrace{\begin{bmatrix} \mu_u I_{n_x} - \Delta t\mu_u^2\mathbf{A} & 0_{n_x} \\ 0_{n_x} & I_{n_x} \end{bmatrix}}_{=:\mathcal{A}}\underbrace{\begin{bmatrix} \mathbf{u}^k \\ \mathbf{v}^k \end{bmatrix}}_{=:\mathbf{x}^k} = \underbrace{\begin{bmatrix} \mu_u\mathbf{u}^{k-1} - \Delta t\mathbf{d_u}(\mathbf{u}^{k-1}, \mathbf{v}^{k-1}) + \Delta t\mu_u^2\mathbf{r}^k \\ \mathbf{v}^{k-1} - \Delta t\mathbf{d_v}(\mathbf{u}^{k-1}, \mathbf{v}^{k-1}) \end{bmatrix}}_{=:\mathbf{b}^k}$$

for $k = 1, \ldots, m$, with initial conditions

$$\begin{bmatrix} \mathbf{u}^0 \\ \mathbf{v}^0 \end{bmatrix} = \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{v}_0 \end{bmatrix}.$$

To solve the fully discretized system efficiently, we make use of MATLAB's `lu` function, which computes the *LU decomposition* of the system matrix $\mathcal{A}$. This factorization yields a lower triangular matrix $\mathbf{L}$ and an upper triangular matrix $\mathbf{U}$ such that $\mathcal{A} = \mathbf{LU}$. The resulting triangular systems can then be solved efficiently via forward and backward substitution. In the first step, we solve $\mathbf{L}\mathbf{y}^k = \mathbf{b}^k$ for $\mathbf{y}^k$ using forward substitution, which exploits the lower triangular structure of $\mathbf{L}$ by progressing row by row from top to bottom. In the second step, we solve $\mathbf{U}\mathbf{x}^k = \mathbf{y}^k$ for $\mathbf{x}^k$ via backward substitution, proceeding from bottom to top due to the upper triangular form of $\mathbf{U}$. Since the structure of the system matrix $\mathcal{A}$ remains constant throughout all time steps, the LU decomposition needs to be computed only once and can be reused for every $k = 1, \ldots, m$. This not only reduces computational effort significantly but also takes advantage of the matrix's sparsity and block-diagonal structure.

The solution of this system provides the data for the snapshot matrix X_{in}, where the k-th column is given by the concatenated state vector $[\mathbf{u}^k, \mathbf{v}^k]^\top$. Calculating the rank of X_{in} with MATLAB's `rank` function yields a value of 51, which indicates a low rank structure of the data matrix. It is worth noting that `rank` computes the number of singular values that are larger than a certain threshold. By default, MATLAB's threshold depends on the size and the norm of the matrix.

The singular values of X_{in} are depicted in Figure 4. These singular values are computed using MATLAB's `svd(`$X_{\texttt{in}}$`, 'econ')` function, which produces an economy-sized decomposition of the matrix X_{in}. This means that the decomposition removes unnecessary rows or columns of zeros from the diagonal matrix of singular values, as well as the corresponding left and right singular vectors, depending on the size of X_{in}, thereby improving computational efficiency and reducing storage requirements without affecting the accuracy of the decomposition. Since X_{in} has more columns than rows, this is particularly efficient.

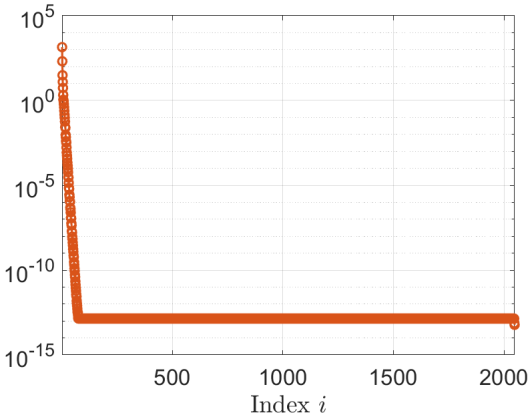


Figure 4: Singular values of the data matrix X_{in}.

Next, we compute the relative Frobenius error (3.20) of the rDMD reconstructions X_{rec} for different rank truncation values $r = 1, \dots, 51$. The results are shown in the left panel of Figure 5. The maximum relative Frobenius error is reached for $r = 43$, with a value of $\mathcal{E}(X_{\text{rec}}, 43) \approx 1.366$, whereas the minimum relative Frobenius error occurs at $r = 46$, reaching a value of $\mathcal{E}(X_{\text{rec}}, 46) \approx 0.941$. Overall, we observe that the relative Frobenius error is large for all rank truncation values, indicating that the rDMD algorithm fails to provide a good approximation for this dataset.

Remark 4.1.

(i) Note that the DMD algorithm tends to achieve higher accuracy for coupled systems when the datasets of all variables are concatenated into a single matrix (cf. [2], [27]). The two approaches are referred to as *coupled and uncoupled DMD*, respectively. In this thesis we only consider the coupled DMD approach.

(ii) Instead of using the full dataset

$$X_{\text{in}} \in \mathbb{R}^{2048 \times 6001},$$

obtained from the solution of the FHN system, we consider only the last $6 \cdot 10^3$ columns for both the rDMD and the (later introduced) pDMD analysis. After the reconstruction, the initial condition is reattached to the reconstructed solution (cf. Section 3.3).

As an additional measure to assess the reconstruction quality of the rDMD method, we compare the temporal evolution of the snapshot matrix X_{in} and the reconstructed matrix X_{rec} for a fixed rank r. Therefore, we introduce the *spatial mean*, defined as

$$\langle u(t) \rangle := \frac{1}{|\Omega|} \int_{\Omega} u(t, x)\, dx \tag{4.5a}$$

for the variable u and $t \in [0, T]$ in the one-dimensional case. Analogously, in the two-dimensional case, it is given by

$$\langle u(t) \rangle := \frac{1}{|\Omega|} \int_{\Omega} u(t, x_1, x_2)\, dx_1 dx_2. \tag{4.5b}$$

Here, $|\Omega|$ denotes the measure of the spatial domain Ω, which corresponds to the length of the interval $(0, 1)$ in the one-dimensional case and to the area of the rectangle

$$(0, L_x) \times (0, L_y)$$

in the two-dimensional case (later), where L_x and L_y are given positive constants. For the reconstructed matrix X_{rec}, the spatial mean of the variable u and v at time t_k can concretely determined as

$$\langle u(t_k) \rangle := \frac{1}{n_x} \sum_{i=1}^{n_x} (X_{\text{rec}})_{i,k},$$

$$\langle v(t_k) \rangle := \frac{1}{n_x} \sum_{i=n_x+1}^{2n_x} (X_{\text{rec}})_{i,k},$$

respectively. In MATLAB, the spatial means over time of the reconstructed matrix X_{rec} can be computed using the command `mean(`$X_{\texttt{rec}}(1:n_x,:)$`, 1)` and `mean(`$X_{\texttt{rec}}(n_x+1:2*n_x,:)$`, 1)`, respectively.
The results are displayed in the middle and right panel of Figure 5. We observe that the reconstruction of both u and v fails entirely to capture the periodic behavior and the amplitude of the oscillations present in the snapshot matrix.

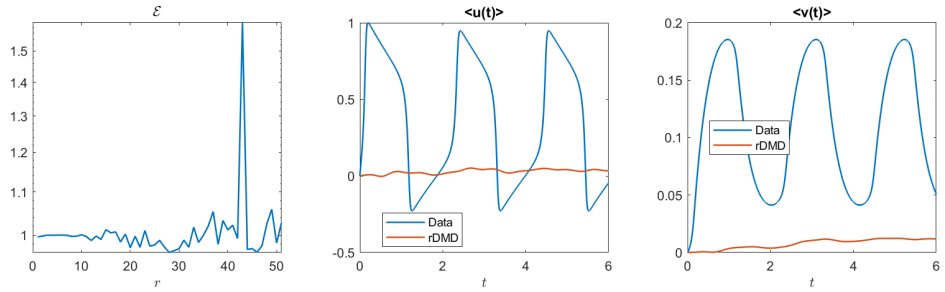


Figure 5: Left: relative Frobenius error of the rDMD reconstruction for $r = 1, \ldots, 51$. Middle/Right: spatial mean over time for u (middle) and v (right), where the rDMD reconstruction is performed with $r = 46$.

The snapshot matrix X_{in} and the reconstructed matrix X_{rec} for $r = 46$ are shown in Figure 6. The figure clearly illustrates that the rDMD method fails to accurately capture the system's dynamics.

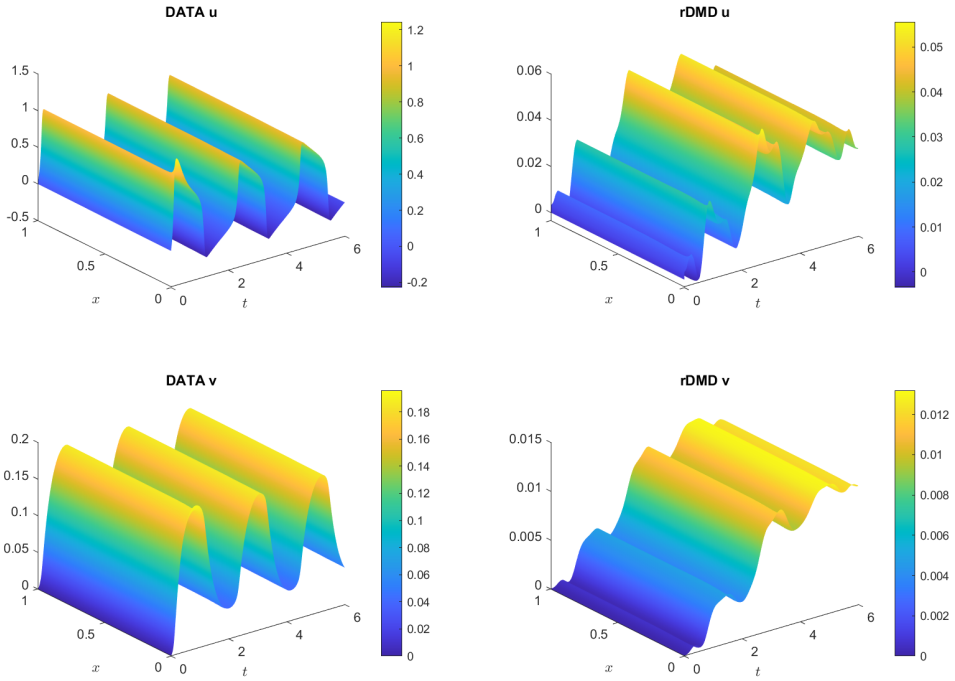


Figure 6: Left: variable u (top) and v (bottom). Right: rDMD reconstruction of u (top) and v (bottom) performed with a rank truncation parameter of $r = 46$.

Finally, the phase planes $(\langle u \rangle, \langle v \rangle)$ are visualized in Figure 7. The limit cycle corresponding to X_{in} is shown in the left panel, while the phase plane of the rDMD

reconstruction appears in the right panel. Again, we observe that the rDMD reconstruction fails to reproduce the characteristic periodic behavior of the original data.

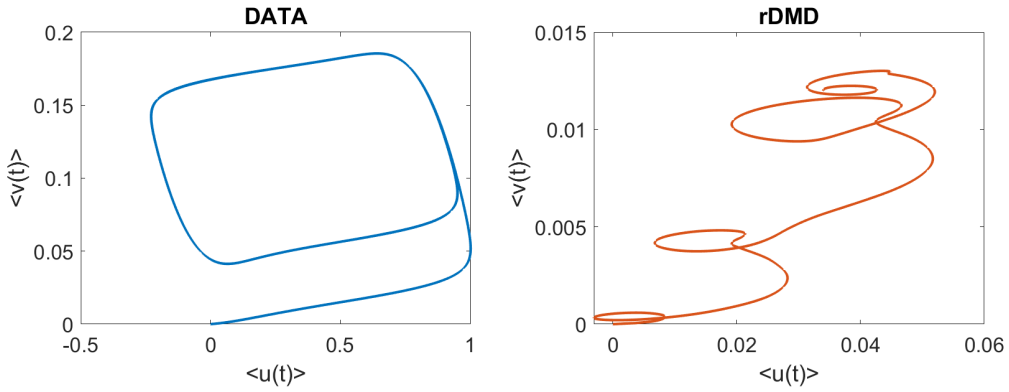


Figure 7: Phase planes $(\langle u \rangle, \langle v \rangle)$. The rDMD reconstruction is performed with a rank truncation parameter of $r = 46$.

λ-ω System: Spiral Waves

Next, we consider the λ-ω *system*. The two-dimensional spatial domain is defined as

$$\begin{aligned}\Omega &:= (0, L_{x_1}) \times (0, L_{x_2}) \\ &:= (0, 130) \times (0, 130),\end{aligned}$$

and the final time is set to $T := 50$. Moreover, we set the diffusion constants as $\mu_u := \mu_v := 4$. The nonlinear functions d_u and d_v are defined by

$$\begin{aligned}d_u(u, v) &:= \rho(\omega(u, v)v - \lambda(u, v)u), \\ d_v(u, v) &:= \rho(-\omega(u, v)u - \lambda(u, v)v),\end{aligned}$$

where the parameters are given by

$$\begin{aligned}\rho &:= 10, \\ \beta &:= 1, \\ \lambda(u, v) &:= 1 - (u^2 + v^2), \\ \omega(u, v) &:= -\beta(u^2 + v^2).\end{aligned}$$

The boundary functions are specified as

$$\begin{aligned}g_u &:= 0 \quad \text{on } \Gamma_T, \\ g_v &:= 0 \quad \text{on } \Gamma_T,\end{aligned}$$

corresponding to homogeneous Neumann boundary conditions. Furthermore, we define the initial functions by

$$\begin{aligned}u_0(x_1, x_2) &:= \frac{1}{10}\left(x_1 - \frac{L_{x_1}}{2}\right) \quad \text{in } \Omega, \\ v_0(x_1, x_2) &:= \frac{1}{10}\left(-\frac{x_2}{2} + \frac{L_{x_2}}{4}\right) \quad \text{in } \Omega.\end{aligned}$$

The spatial discretization is performed using $n_{x_1} := 99$ equidistant interior points in x_1-direction and $n_{x_2} := 99$ equidistant interior points in x_2-direction. Thus, the total number of interior grid points is

$$n := n_{x_1} n_{x_2} = 9801.$$

The corresponding spatial step sizes are given by

$$\Delta x_1 := \frac{L_{x_1}}{n_{x_1} + 1},$$
$$\Delta x_2 := \frac{L_{x_1}}{n_{x_2} + 1},$$

respectively. The temporal step size is $\Delta t := 10^{-3}$, which means that the number of discrete problems to be solved is $m := {}^{T}/_{\Delta t} = 5 \cdot 10^4$. To reduce computational costs, we store only every κ-th snapshot with $\kappa := 4$. For ease of notation, the retained snapshots are relabeled consecutively from 0 to $m_\kappa := {}^{m}/_{\kappa}$. Accordingly, the snapshot matrix X_{in} is of size

$$2n \times (m_\kappa + 1) = 19602 \times 12501.$$

Note that the spatial grid points are numbered in a natural ordering. This indexing is illustrated in Figure 8 for the exemplary rectangular domain $(0, 1)^2$ with $n_{x_1} = n_{x_2} = 2$.

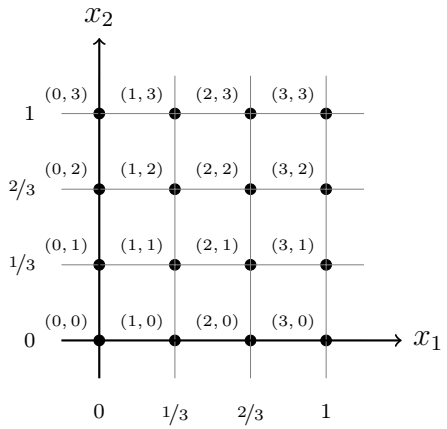


Figure 8: Natural indexing of $(0, 1)^2$ with $n_{x_1} = n_{x_2} = 2$.

The FOM resulting from the spatial semi-discretization using the MOL is given by

$$\begin{aligned} \dot{\mathbf{u}} - \mu_u \mathbf{A}\mathbf{u} + \mathbf{d}_{\mathbf{u}}(\mathbf{u}, \mathbf{v}) &= 0_n \quad \text{for } t \in (0, T], \\ \dot{\mathbf{v}} - \mu_v \mathbf{A}\mathbf{v} + \mathbf{d}_{\mathbf{v}}(\mathbf{u}, \mathbf{v}) &= 0_n \quad \text{for } t \in (0, T], \\ \mathbf{u}(0) &= \mathbf{u}_0, \\ \mathbf{v}(0) &= \mathbf{v}_0. \end{aligned} \tag{4.6}$$

Here, the vectors

$$\mathbf{u} := \mathbf{u}(t)$$

$$
\begin{aligned}
&:= \begin{bmatrix} \mathbf{u}_{11}(t) & \cdots & \mathbf{u}_{n_{x_1}1}(t) & \cdots & \mathbf{u}_{1n_{x_2}}(t) & \cdots & \mathbf{u}_{n_{x_1}n_{x_2}}(t) \end{bmatrix}^\top, \\
\mathbf{v} &:= \mathbf{v}(t) \\
&:= \begin{bmatrix} \mathbf{v}_{11}(t) & \cdots & \mathbf{v}_{n_{x_1}1}(t) & \cdots & \mathbf{v}_{1n_{x_2}}(t) & \cdots & \mathbf{v}_{n_{x_1}n_{x_2}}(t) \end{bmatrix}^\top
\end{aligned}
$$

represent the spatial discretizations of u and v, respectively, where

$$
\begin{aligned}
\mathbf{u}_{ij}(t) &\approx u(t, x_i, x_j), \\
\mathbf{v}_{ij}(t) &\approx v(t, x_i, x_j).
\end{aligned}
$$

The initial conditions are given by

$$
\begin{aligned}
\mathbf{u}_0 &:= \mathbf{u}(0) \\
&:= \begin{bmatrix} u_0(x_1, x_1) & \cdots & u_0(x_{n_{x_1}}, x_1) & \cdots & u_0(x_1, x_{n_{x_2}}) & \cdots & u_0(x_{n_{x_1}}, x_{n_{x_2}}) \end{bmatrix}^\top, \\
\mathbf{v}_0 &:= \mathbf{v}(0) \\
&:= \begin{bmatrix} v_0(x_1, x_1) & \cdots & v_0(x_{n_{x_1}}, x_1) & \cdots & v_0(x_1, x_{n_{x_2}}) & \cdots & v_0(x_{n_{x_1}}, x_{n_{x_2}}) \end{bmatrix}^\top.
\end{aligned}
$$

Again, the matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is derived from the discretization of the Laplace operator combined with the boundary conditions. We use the second-order central difference approximation

$$
\begin{aligned}
\Delta u(t, x_i, x_j) \approx{}& \frac{1}{(\Delta x_1)^2} \left(u(t, x_{i-1}, x_j) - 2u(t, x_i, x_j) + u(t, x_{i+1}, x_j) \right) \\
&+ \frac{1}{(\Delta x_2)^2} \left(u(t, x_i, x_{j-1}) - 2u(t, x_i, x_j) + u(t, x_i, x_{j+1}) \right)
\end{aligned}
$$

for $i = 1, \ldots, n_{x_1}$ and $j = 1, \ldots, n_{x_2}$. What remains to be specified is how to handle the function values at the boundary points. This is where the Neumann boundary conditions come into play. They can be written as

$$
\nabla u \cdot n = 0 \iff \begin{cases}
-\dfrac{\partial u}{\partial x_1}(t, x_0, x_j) = 0 & \text{for } j = 1, \ldots, n_{x_2}, \\
\dfrac{\partial u}{\partial x_1}(t, x_{n_{x_1}+1}, x_j) = 0 & \text{for } j = 1, \ldots, n_{x_2}, \\
-\dfrac{\partial u}{\partial x_2}(t, x_i, x_0) = 0 & \text{for } i = 1, \ldots, n_{x_1}, \\
\dfrac{\partial u}{\partial x_2}(t, x_i, x_{n_{x_2}+1}) = 0 & \text{for } i = 1, \ldots, n_{x_1},
\end{cases}
$$

since we are considering the outward unit normal vector. By applying a second-order forward difference quotient for the spatial derivative at the grid points (x_i, x_0) and (x_0, x_j), and a second-order backward difference quotient at the grid points $(x_{n_{x_1}+1}, x_j)$ and $(x_i, x_{n_{x_2}+1})$, we obtain the approximations

$$
\begin{aligned}
0 = \frac{\partial u}{\partial x_1}(t, x_0, x_j) &\approx \frac{u(t, x_2, x_j) - 4u(t, x_1, x_j) + 3u(t, x_0, x_j)}{2\Delta x_1}, \\
0 = \frac{\partial u}{\partial x_1}(t, x_{n_{x_1}+1}, x_j) &\approx \frac{u(t, x_{n_{x_1}}, x_j) - 4u(t, x_{n_{x_1}-1}, x_j) + 3u(t, x_{n_{x_1}+1}, x_j)}{2\Delta x_1}, \\
0 = \frac{\partial u}{\partial x_2}(t, x_i, x_0) &\approx \frac{u(t, x_i, x_2) - 4u(t, x_i, x_1) + 3u(t, x_i, x_0)}{2\Delta x_2}, \\
0 = \frac{\partial u}{\partial x_2}(t, x_i, x_{n_{x_2}+1}) &\approx \frac{u(t, x_i, x_{n_{x_2}}) - 4u(t, x_i, x_{n_{x_2}-1}) + 3u(t, x_i, x_{n_{x_2}+1})}{2\Delta x_2}.
\end{aligned}
$$

Rearranging leads to

$$\begin{aligned}
u(t, x_0, x_j) &\approx \frac{4}{3}u(t, x_1, x_j) - \frac{1}{3}u(t, x_2, x_j),\\
u(t, x_{n_{x_1}+1}, x_j) &\approx \frac{4}{3}u(t, x_{n_{x_1}}, x_j) - \frac{1}{3}u(t, x_{n_{x_1}-1}, x_j),\\
u(t, x_i, x_0) &\approx \frac{4}{3}u(t, x_i, x_1) - \frac{1}{3}u(t, x_i, x_2),\\
u(t, x_i, x_{n_{x_2}+1}) &\approx \frac{4}{3}u(t, x_i, x_{n_{x_2}}) - \frac{1}{3}u(t, x_i, x_{n_{x_2}-1}).
\end{aligned}$$

By using the two matrices $\mathbf{T}_{x_1}$ and $\mathbf{T}_{x_2}$, defined by

$$\mathbf{T}_{x_1} := \frac{1}{\Delta x_1^2}\begin{bmatrix}
-\frac{2}{3} & \frac{2}{3} & & & \\
1 & -2 & 1 & & \\
& \ddots & \ddots & \ddots & \\
& & 1 & -2 & 1 \\
& & & \frac{2}{3} & -\frac{2}{3}
\end{bmatrix} \in \mathbb{R}^{n_{x_1}\times n_{x_1}},$$

$$\mathbf{T}_{x_2} := \frac{1}{\Delta x_2^2}\begin{bmatrix}
-\frac{2}{3} & \frac{2}{3} & & & \\
1 & -2 & 1 & & \\
& \ddots & \ddots & \ddots & \\
& & 1 & -2 & 1 \\
& & & \frac{2}{3} & -\frac{2}{3}
\end{bmatrix} \in \mathbb{R}^{n_{x_2}\times n_{x_2}},$$

we can express the discrete Laplacian operator on a rectangular domain in Kronecker form as

$$\mathbf{A} = I_{n_{x_2}} \otimes \mathbf{T}_{x_1} + \mathbf{T}_{x_2} \otimes I_{n_{x_1}} \in \mathbb{R}^{n\times n}. \tag{4.7}$$

Finally, the two functions

$$\begin{aligned}
\mathbf{d_u}&: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n,\\
\mathbf{d_v}&: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n
\end{aligned}$$

are defined analogously to d_u and d_v in a component-wise manner.

To discretize the ODE system (4.4) in time, we first use the IMEX Euler method in vector form. Analogously to the FHN system, we obtain the fully discretized linear system

$$\begin{bmatrix} I_n - \Delta t\mu_u\mathbf{A} & 0_n \\ 0_n & I_n - \Delta t\mu_v\mathbf{A} \end{bmatrix}\begin{bmatrix} \mathbf{u}^k \\ \mathbf{v}^k \end{bmatrix} = \begin{bmatrix} \mathbf{u}^{k-1} - \Delta t\mathbf{d_u}(\mathbf{u}^{k-1}, \mathbf{v}^{k-1}) \\ \mathbf{v}^{k-1} - \Delta t\mathbf{d_v}(\mathbf{u}^{k-1}, \mathbf{v}^{k-1}) \end{bmatrix}$$

for $k = 1, \ldots, m$, with initial conditions

$$\begin{bmatrix} \mathbf{u}^0 \\ \mathbf{v}^0 \end{bmatrix} = \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{v}_0 \end{bmatrix}.$$

Again, we can solve this system efficiently using the LU decomposition of the system matrix.

To accurately capture the pattern structure of the solution and to ensure stability due to the presence of explicit components, a fine discretization is required in both space and time. Thus, the IMEX Euler method in vector form can be too costly for large dimensions (e.g., $n \geq 2500$) and large number m of problems to be solved. A significantly more efficient alternative for time discretization is to employ the IMEX Euler method in matrix form. To apply this, the vectors $\mathbf{u}$ and $\mathbf{v}$ are reshaped into matrices $\mathbf{U}$ and $\mathbf{V}$:

$$\mathbf{U} := \begin{bmatrix} \mathbf{u}_{11} & \cdots & \mathbf{u}_{1n_{x_2}} \\ \vdots & \ddots & \vdots \\ \mathbf{u}_{n_{x_1}1} & \cdots & \mathbf{u}_{n_{x_1}n_{x_2}} \end{bmatrix} \in \mathbb{R}^{n_{x_1} \times n_{x_2}},$$
$$\mathbf{V} := \begin{bmatrix} \mathbf{v}_{11} & \cdots & \mathbf{v}_{1n_{x_2}} \\ \vdots & \ddots & \vdots \\ \mathbf{v}_{n_{x_1}1} & \cdots & \mathbf{v}_{n_{x_1}n_{x_2}} \end{bmatrix} \in \mathbb{R}^{n_{x_1} \times n_{x_2}}.$$

Thus, the vectors $\mathbf{u}$ and $\mathbf{v}$ represent the vectorizations of $\mathbf{U}$ and $\mathbf{V}$, respectively, i.e.,

$$\mathbf{u} = \operatorname{vec}(\mathbf{U}),$$
$$\mathbf{v} = \operatorname{vec}(\mathbf{V}).$$

Together with (4.7), we obtain the identity

$$\begin{aligned} \mathbf{A}\mathbf{u} &= \left(I_{n_{x_2}} \otimes \mathbf{T}_{x_1}\right) \operatorname{vec}(\mathbf{U}) + \left(\mathbf{T}_{x_2} \otimes I_{n_{x_1}}\right) \operatorname{vec}(\mathbf{U}) \\ &= \operatorname{vec}(\mathbf{T}_{x_1}\mathbf{U}I_{n_{x_2}}^\top) + \operatorname{vec}(I_{n_{x_1}}\mathbf{U}\mathbf{T}_{x_2}^\top) \\ &= \operatorname{vec}(\mathbf{T}_{x_1}\mathbf{U}) + \operatorname{vec}(\mathbf{U}\mathbf{T}_{x_2}^\top) \\ &= \operatorname{vec}(\mathbf{T}_{x_1}\mathbf{U} + \mathbf{U}\mathbf{T}_{x_2}^\top) \end{aligned}$$

and analogously

$$\mathbf{A}\mathbf{v} = \operatorname{vec}(\mathbf{T}_{x_1}\mathbf{V} + \mathbf{V}\mathbf{T}_{x_2}^\top).$$

This leads to the fully discrete system:

$$\begin{aligned} &\frac{\mathbf{U}^k - \mathbf{U}^{k-1}}{\Delta t} - \mu_u(\mathbf{T}_{x_1}\mathbf{U}^k + \mathbf{U}^k\mathbf{T}_{x_2}^\top) + \mathbf{d}_\mathbf{u}(\mathbf{U}^{k-1}, \mathbf{V}^{k-1}) = 0_{n_{x_1} \times n_{x_2}} \quad \text{for } k = 1, \ldots, m, \\ &\frac{\mathbf{V}^k - \mathbf{V}^{k-1}}{\Delta t} - \mu_v(\mathbf{T}_{x_1}\mathbf{V}^k + \mathbf{V}^k\mathbf{T}_{x_2}^\top) + \mathbf{d}_\mathbf{v}(\mathbf{U}^{k-1}, \mathbf{V}^{k-1}) = 0_{n_{x_1} \times n_{x_2}} \quad \text{for } k = 1, \ldots, m, \end{aligned}$$

with the initial conditions

$$\mathbf{U}^0 = \mathbf{U}_0,$$
$$\mathbf{V}^0 = \mathbf{V}_0.$$

Here, $\mathbf{U}_0$ and $\mathbf{V}_0$ are matrices representing the initial conditions, and $\mathbf{d}_\mathbf{u}$ and $\mathbf{d}_\mathbf{v}$ are again the component-wise representations of d_u and d_v. Rewriting the above system yields the so-called *Sylvester equations*:

$$\begin{aligned} \left(I_{n_{x_1}} - \Delta t\mu_u\mathbf{T}_{x_1}\right)\mathbf{U}^k + \mathbf{U}^k\left(-\Delta t\mu_u\mathbf{T}_{x_2}^\top\right) &= \mathbf{U}^{k-1} - \Delta t\mathbf{d}_\mathbf{u}(\mathbf{U}^{k-1}, \mathbf{V}^{k-1}), \\ \left(I_{n_{x_1}} - \Delta t\mu_v\mathbf{T}_{x_1}\right)\mathbf{V}^k + \mathbf{V}^k\left(-\Delta t\mu_v\mathbf{T}_{x_2}^\top\right) &= \mathbf{V}^{k-1} - \Delta t\mathbf{d}_\mathbf{v}(\mathbf{U}^{k-1}, \mathbf{V}^{k-1}). \end{aligned} \tag{4.8}$$

Thus, at each time step k, two Sylvester equations must be solved with coefficient matrices

$$\begin{aligned}\left(I_{n_{x_1}} - \Delta t \mu_u \mathbf{T}_{x_1}\right), \quad \left(-\Delta t \mu_u \mathbf{T}_{x_2}^\top\right),\\ \left(I_{n_{x_1}} - \Delta t \mu_v \mathbf{T}_{x_1}\right), \quad \left(-\Delta t \mu_v \mathbf{T}_{x_2}^\top\right),\end{aligned}$$

and right-hand sides

$$\begin{aligned}\mathbf{U}^{k-1} - \Delta t \mathbf{d_u}(\mathbf{U}^{k-1}, \mathbf{V}^{k-1}),\\ \mathbf{V}^{k-1} - \Delta t \mathbf{d_v}(\mathbf{U}^{k-1}, \mathbf{V}^{k-1}).\end{aligned}$$

Compared to the IMEX Euler method in vector form, where the system matrices are of size $n \times n$, the Sylvester equation approach operates directly on matrices of significantly smaller sizes. Although the $n \times n$ matrices are typically sparse and structured – enabling the use of direct and iterative solvers – the Sylvester formulation benefits from working with significantly smaller matrices. This not only reduces memory requirements but also permits the use of highly efficient small size factorization methods. In particular, since the matrices $\mathbf{T}_{x_1}$ and $\mathbf{T}_{x_2}^\top$ remain consistent across all time steps, their factorizations are reusable, further improving the overall computational efficiency of the scheme.

One of this small size factorization methods is based on an a-priori eigendecomposition of $\mathbf{T}_{x_1}^\top$ and $\mathbf{T}_{x_2}$. In the following we assume that $\mathbf{T}_{x_1}$ and $\mathbf{T}_{x_2}^\top$ are diagonalizable, i.e., their eigendecompositions

$$\begin{aligned}\mathbf{T}_{x_1} &= \mathbf{Q}_{x_1} \mathbf{\Lambda}_{x_1} \mathbf{Q}_{x_1}^{-1},\\ \mathbf{T}_{x_2}^\top &= \mathbf{Q}_{x_2} \mathbf{\Lambda}_{x_2} \mathbf{Q}_{x_2}^{-1}\end{aligned}$$

can be determined. The matrices $\mathbf{Q}_{x_1}$ and $\mathbf{Q}_{x_2}$ are invertible and

$$\begin{aligned}\mathbf{\Lambda}_{x_1} &:= \operatorname{diag}(\lambda_{x_1,1}, \ldots, \lambda_{x_1,n_{x_1}}),\\ \mathbf{\Lambda}_{x_2} &:= \operatorname{diag}(\lambda_{x_2,1}, \ldots, \lambda_{x_2,n_{x_2}})\end{aligned}$$

are diagonal matrices with the eigenvalues $\lambda_{x_1,i}$ and $\lambda_{x_2,j}$ on the diagonal. Although $\mathbf{T}_{x_1}^\top$ and $\mathbf{T}_{x_2}$ deviate from symmetry due to boundary modifications, their overall structure remains sufficiently close to symmetric matrices, so that numerical stability of the decompositions can be expected.
To solve the Sylvester equations (4.8) using the eigendecompositions of $\mathbf{T}_{x_1}$ and $\mathbf{T}_{x_2}^\top$, we first introduce a matrix L. The elements of L are defined as

$$\mathrm{L}_{ij} := \frac{1}{(1 - \Delta t \mu_u \lambda_{x_1,i}) + (-\Delta t \mu_u \lambda_{x_2,j})}$$

for $i = 1, \ldots, n_{x_1}$ and $j = 1, \ldots, n_{x_2}$. Hence, at each time step k, we compute

$$\begin{aligned}\widehat{\mathbf{U}}^k &:= \mathbf{Q}_{x_1}^{-1} \left(\mathbf{U}^{k-1} - \Delta t \mathbf{d_u}(\mathbf{U}^{k-1}, \mathbf{V}^{k-1})\right) \mathbf{Q}_{x_2},\\ \widehat{\mathbf{V}}^k &:= \mathbf{Q}_{x_1}^{-1} \left(\mathbf{V}^{k-1} - \Delta t \mathbf{d_v}(\mathbf{U}^{k-1}, \mathbf{V}^{k-1})\right) \mathbf{Q}_{x_2}\end{aligned}$$

and obtain the iterates as

$$\mathbf{U}^k = \mathbf{Q}_{x_1} \left(\mathrm{L} \odot \widehat{\mathbf{U}}^k\right) \mathbf{Q}_{x_2}^{-1},$$

$$\mathbf{V}^k = \mathbf{Q}_{x_1} \left(\mathrm{L} \odot \widehat{\mathbf{V}}^k \right) \mathbf{Q}_{x_2}^{-1}.$$

In these expressions, $\odot$ denotes the element-wise (Hadamard) product between two matrices.

In Table 2, the running times of the IMEX Euler method in vector form and matrix form are compared for solving the λ-ω system.

Table 2: Comparison of the running times for the IMEX Euler method in vector form and matrix form when solving the λ-ω system.

	IMEX Euler vector form	IMEX Euler matrix form
Running time $[s]$	459.3	50.3

Table 2 clearly demonstrates the computational advantage of the matrix-based IMEX Euler scheme. While the classical vector form approach requires a running time of 459.3 seconds, the matrix formulation completes in only 50.3 seconds. This significant speed-up is mainly due to the use of small-size Sylvester solvers and the reuse of eigendecompositions across all time steps.

Next, we evaluate the quality of the rDMD method with respect to the reference data X_{in}, obtained via the IMEX Euler scheme in matrix form. Starting from simple step profiles in u and v at $t = 0$, the solution develops into an Archimedean spiral wave. This spiral forms a stationary center located at $(x_{1c}, x_{2c}) := (65, 65)$ and begins to expand outward. The spatio-temporal oscillations of the spiral continue until a characteristic time $\bar{t}$, at which point the spiral pattern has fully developed and occupies the entire domain Ω (see Figure 9).

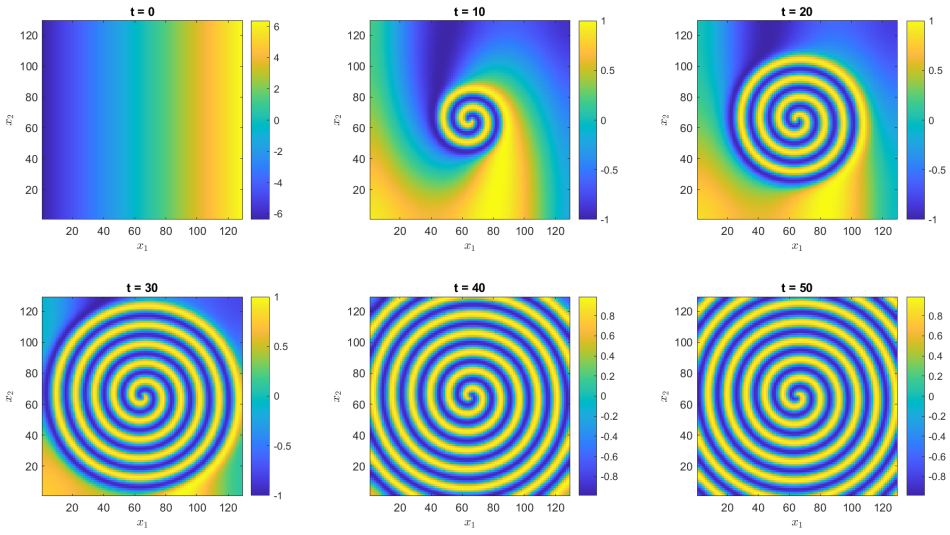


Figure 9: Variable u at the time points $t = 0, 10, 20, 30, 40, 50$.

For $t \geq \bar{t}$, the system enters a new time regime where it continues to oscillate. The spatial means then evolve along a closed trajectory in the phase plane, indicating a limit cycle (cf. Figure 12, left).

In Figure 10 (left), we show the behavior of the relative Frobenius error $\mathcal{E}(X_{\text{rec}}, r)$ for $r = 1, \ldots, 314$, where 314 corresponds to the rank of X_{in}. Initially, the error decreases and attains a minimum value of $2.644 \cdot 10^{-3}$ at $r = 136$. However, for $r \gtrapprox 150$, the error increases drastically due to the ill-conditioning of the matrix $\tilde{A}_B$ in the rDMD algorithm. This highlights a key limitation of DMD: although one might expect improved accuracy with increasing r, this is not the case here due to numerical instability.
Figure 10 (middle and right) further illustrates how the rDMD reconstruction quality declines for increasing ranks by comparing the spatial mean over time for u, where the rDMD reconstruction is performed with $r = 136$ (middle) and $r = 300$ (right). For $r = 300$, the relative Frobenius error reaches $\mathcal{E}(X_{\text{rec}}, 300) = 7.838$. In both cases, the two distinct time regimes $[0, \bar{t}]$ and $(\bar{t}, T]$ with $\bar{t} \approx 25$ observed in the reference data are not accurately reproduced. Moreover, noticeable discrepancies appear in the wave amplitudes, especially for $r = 300$.

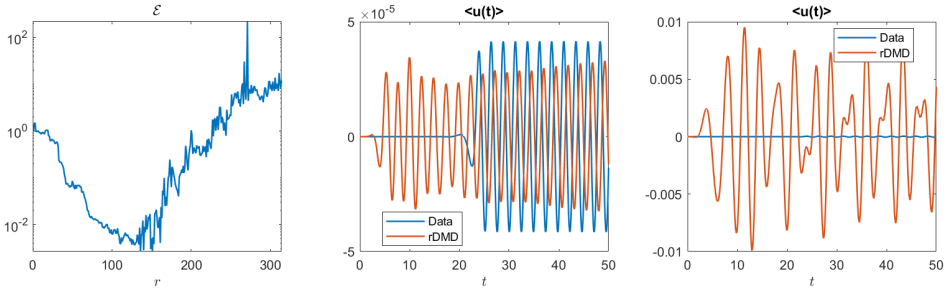


Figure 10: Left: relative Frobenius error of the rDMD reconstruction for $r = 1, \ldots, 314$. Middle/Right: spatial mean over time for u, where the rDMD reconstruction is performed with $r = 136$ (middle) and $r = 300$ (right).

The variable u at the final time T is shown in Figure 11 (left). Reconstructions obtained via rDMD for $r = 136$ and $r = 300$ are presented in the middle and right panels, respectively.

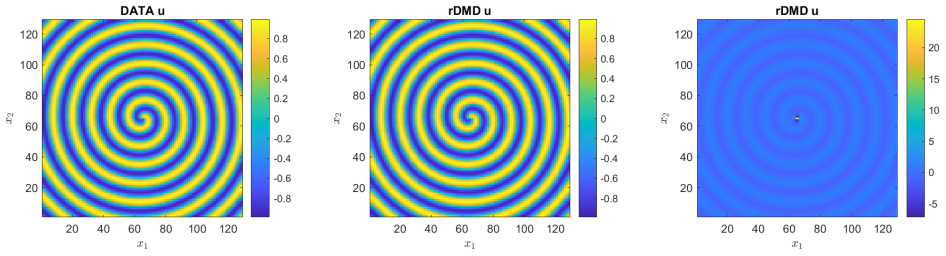


Figure 11: Variable u at the final time T. Left: reference data. Middle: rDMD reconstruction with $r = 136$. Right: rDMD reconstruction with $r = 300$.

The reconstruction at $r = 136$ aligns remarkably well with the reference data, although its temporal evolution differs significantly (cf. Figure 10, middle). In contrast, the case with $r = 300$ does not accurately capture the spiral wave structure.

Finally, we analyze the phase planes $(\langle u \rangle, \langle v \rangle)$ for both the reference data and the rDMD reconstructions. As described earlier, the reference trajectory forms a clear limit cycle, a feature that is not captured by either of the rDMD reconstructions (see

Figure 12). These findings further emphasize the limitations of rDMD in capturing the dynamic behavior present in the reference data.

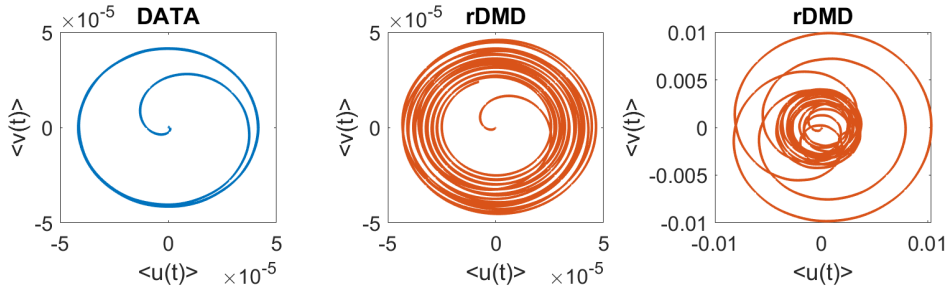


Figure 12: Phase planes ($\langle u \rangle, \langle v \rangle$). Left: reference data. Middle: rDMD reconstruction with $r = 136$. Right: rDMD reconstruction with $r = 300$.

4.2.2 Limitations on a Dataset with Spatio-Temporal Turing Instability

DIB Morphochemical System

As a final example, we study the *DIB morphochemical system* – a model for metal growth in electrodeposition. It captures phenomena occurring during the recharge processes of batteries with metal electrodes and exhibits the so-called *Turing instability*. This instability arises when a spatially homogeneous steady state becomes unstable due to diffusion-driven mechanisms, leading to the spontaneous emergence of stationary spatial patterns like labyrinths, stripes or spots, dependending on the parameters of the kinetics.

The simulation begins with a small (random) spatially distributed pertubation around the homogeneous stationary point $P_e := (u_e, v_e)$ – the equilibrium of the system in the absence of diffusion. As the system evolves, two different time regimes emerge: first comes the *reactivity zone* $\mathcal{I}_1$, where diffusion destabilizes the initial equilibrium. This is followed by the *stability zone* $\mathcal{I}_2$, in which the system converges toward a stationary, spatially structured inhomogeneous state – a *Turing pattern*. In other words, Turing patterns are stationary solutions of RD-PDE systems.

We consider the DIB morphochemical system on the square domain

$$\Omega := (0, L_{x_1}) \times (0, L_{x_2}),$$

where $L_{x_1} := 20$ and $L_{x_2} := 20$. The final time is set to $T := 40$ and the diffusion constants are chosen as $\mu_u := 1$ and $\mu_v := 20$. The reaction kinetics are governed by the nonlinear terms

$$d_u(u, v) := -\rho\Big(A_1(1 - v)u - A_2u^3 - B(v - \alpha)\Big), \tag{4.9a}$$

$$d_v(u, v) := -\rho\Big(C(1 + k_2u)(1 - v)\,(1 - \gamma(1 - v)) - Dv(1 + k_3u)(1 + \gamma v)\Big) \tag{4.9b}$$

with parameters

A_1	A_2	B	C	α	ρ	k_2	k_3	γ
10	1	66	3	0.5	$^{25}/_{4}$	2.5	1.5	0.2

as well as

$$D := \frac{C(1-\alpha)(1-\gamma+\gamma\alpha)}{\alpha(1+\gamma\alpha)}. \tag{4.9c}$$

Here, u models the morphology of the electrodeposit, while v describes its chemical composition. Further insights into the electrochemical relevance of the parameters can be found, for instance, in [16] and the references therein.
We suppose homogeneous Neumann boundary conditions for both components:

$$\begin{aligned} g_u &:= 0 \quad \text{on } \Gamma, \\ g_v &:= 0 \quad \text{on } \Gamma. \end{aligned}$$

Finally, the initial functions are given by

$$\begin{aligned} u_0(x_1, x_2) &:= u_e + 10^{-5}\texttt{rand}(x_1, x_2), \\ v_0(x_1, x_2) &:= v_e + 10^{-5}\texttt{rand}(x_1, x_2), \end{aligned}$$

where $(u_e, v_e) := (0, \alpha)$ is the homogeneous steady state and $\texttt{rand}(x_1, x_2)$ indicates MATLAB's `rand` function, which generates a random number uniformly distributed in the interval $[0, 1]$.

The spatial discretization is performed with $n_{x_1} := 10^2$ equidistant interior points in x_1-direction and $n_{x_2} := 10^2$ equidistant interior points in x_2-direction. This results in a total of

$$n := n_{x_1} n_{x_2} = 10^4$$

spatial *degrees of freedom (DOFS)*. The corresponding spatial step sizes are defined as

$$\begin{aligned} \Delta x_1 &:= \frac{L_{x_1}}{n_{x_1}+1}, \\ \Delta x_2 &:= \frac{L_{x_2}}{n_{x_2}+1}. \end{aligned}$$

For the time integration, we employ a time step size of $\Delta t := 10^{-3}$, which yields $m := {}^{T}/_{\Delta t} = 4 \cdot 10^4$ discrete problems to be solved. Only every κ-th snapshot is stored with $\kappa := 4$. The retained snapshots are relabeled consecutively from 0 to $m_\kappa := {}^{m}/_{\kappa}$. Consequently, the snapshot matrix X_{in} is of size

$$2n \times (m_\kappa + 1) = 2 \cdot 10^4 \times 10001.$$

The FOM for the DIB morphochemical system is again of the form (4.6). Employing the IMEX Euler method in matrix form yields a snapshot matrix X_{in} of rank 303. Hence, we apply the rDMD method for all target ranks $r = 1, \ldots, 303$ and compute the corresponding relative Frobenius errors $\mathcal{E}(X_{\text{rec}}, r)$. The corresponding error curve is depicted in Figure 13 (left). For larger values of r, the relative Frobenius error $\mathcal{E}(X_{\text{rec}}, r)$ drastically increases and blows up for $r \geq 43$. So in contrast to what one might expect, the error does not decrease with increasing r. This behavior is again attributed to the ill-conditioning of the matrix $\tilde{A}_B$ in the rDMD algorithm. The minimal error $\mathcal{E}(X_{\text{rec}}, r) = 1.022 \cdot 10^{-1}$ is attained at $r = 19$.

In Figure 13 (right), the spatial mean over time for u of the rDMD reconstruction with $r = 19$ oscillates around that of the reference data. Moreover, the reference data reveals two distinct time regimes, namely $\mathcal{I}_1 := [0, \bar{t}]$ and $\mathcal{I}_2 := (\bar{t}, T]$, with a transition at approximately $\bar{t} \approx 4$. While this temporal structure is clearly visible in the reference solution, the rDMD reconstruction fails to accurately reproduce it. In particular, the reconstruction destabilizes the equilibrium earlier.

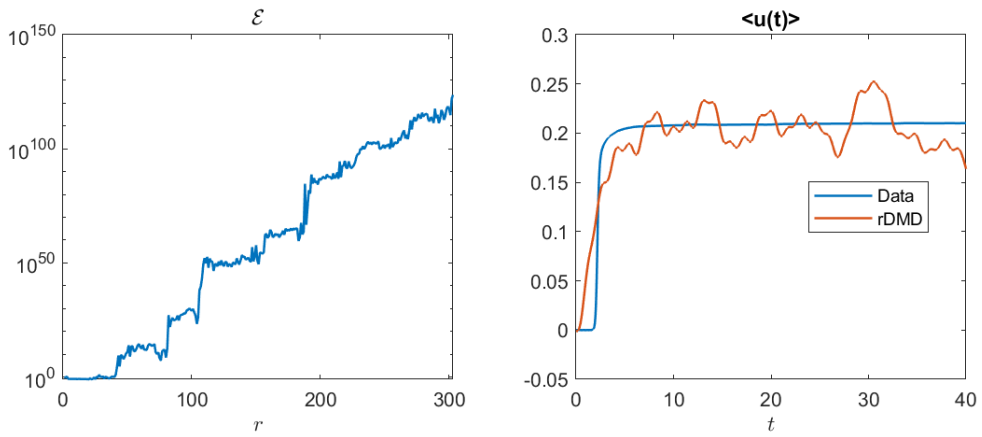


Figure 13: Left: relative Frobenius error $\mathcal{E}(X_{\text{rec}}, r)$ for $r = 1, \ldots, 303$. Right: spatial mean $\langle u \rangle$ for the reference data and the rDMD reconstruction with $r = 19$.

Figure 14 (left) shows the variables u and v at the final time T. The middle panels display the corresponding rDMD reconstructions obtained with $r = 19$. We observe that the characteristic Turing pattern is well reproduced. However, the reconstruction does not accurately capture the amplitudes. This discrepancy becomes more evident in the right panels, where the absolute errors between the reference data and the rDMD reconstructions are illustrated.

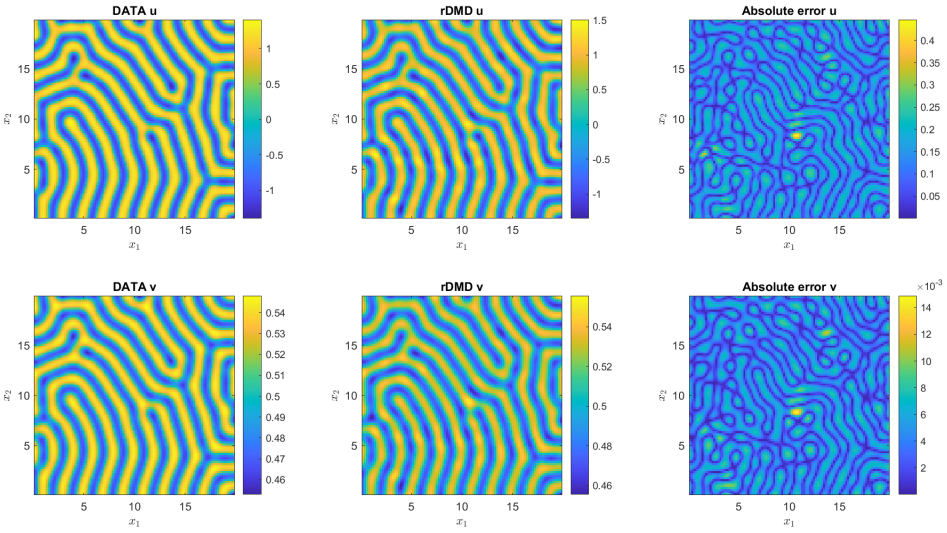


Figure 14: Variables u (top) and v (bottom) at the final time T. Left: reference data. Middle: rDMD reconstruction with $r = 19$. Right: absolute error between the reference data and the rDMD reconstruction.

To summarize, while the rDMD method successfully captures the Turing pattern structure, it once again fails in accurately approximating the reference data. In particular, the relative Frobenius error consistently exceeds 10%.

4.3 Piecewise DMD Algorithm

Section 4.2 highlights that rDMD fails to reconstruct the dynamics of datasets exhibiting peridoic behavior or Turing pattern formation. Moreover, increasing the rank does not improve the reconstruction quality in terms of the relative Frobenius error. To address these limitations, we propose a novel technique, the pDMD method, which leverages a localized reconstruction strategy inspired by classical divide-and-conquer techniques in numerical analysis.

The central idea of pDMD is to replace the global linear approximation over the full time horizon $[0, T]$ with several local linear models. More precisely, the time domain is partitioned into $N \geq 1$ segments and a separate low-rank rDMD reconstruction is performed on each subinterval. This allows the method to more effectively capture regime shifts and transient phenomena that are typically missed by a single global fit.

Let $X_{\text{in}} \in \mathbb{R}^{n\times(m+1)}$ denote the reference snapshot matrix. We begin by selecting an initial number of segments N_{in}, which defines the initial partitioning of X_{in}. This segment count is then incremented in steps of size N_{inc} until either a target reconstruction accuracy is achieved or a maximum number of segments $N_{\max}$ is reached. To ensure that each segment contains at least 10 snapshots, we define

$$N_{\max} := \lfloor \frac{m+1}{10} \rfloor.$$

Thus, for each value of N, we obtain a decomposition of the form

$$X_{\text{in}} := \begin{bmatrix} X_1 & \cdots & X_N \end{bmatrix},$$

where each segment X_i is given by

$$X_i := \begin{bmatrix} (X_{\text{in}})_{:,\lceil (i-1)\nu+1 \rceil} & \cdots & (X_{\text{in}})_{:,\lceil i\nu \rceil} \end{bmatrix},$$

with $\nu := \frac{m+1}{N}$ denoting the (possibly non-integer) segment width. Here, for example, $(X_{\text{in}})_{:,\lceil (i-1)\nu+1 \rceil}$ refers to the $\lceil (i-1)\nu + 1 \rceil$-th column of X_{in}. The rDMD method (algorithm 3) with target rank $r_i := \text{rk}(X_i)$, oversampling parameter $p := 0$ and two power iterations ($q := 2$) is then applied to each X_i. To ensure accurate reconstruction, the target rank r_i is iteratively decreased until the relative error condition

$$\text{err}(i) := \max_{(i-1)\nu+1 \leq k \leq i\nu} \frac{\|(X_i)_{:,k} - (\tilde{X}_i)_{:,k}\|_\infty}{\|(X_i)_{:,k}\|_\infty} \leq \varepsilon \tag{4.10}$$

is satisfied for a given tolerance $\varepsilon > 0$. If this condition is not satisfied for all $r_i > 0$ ($i = 1, \ldots, N$) the current segmentation is considered unsuccessful and we increment N before restarting the procedure. When all segments are successfully reconstructed, the local approximations $\tilde{X}_i$ are concatenated to yield the full reconstruction X^N_{rec}. For analysis purposes, we also store the rank vector $\mathbf{r} := \begin{bmatrix} r_1 & \cdots & r_N \end{bmatrix}$ and the maximum target rank

$$\tilde{r}(N) := \|r\|_\infty := \max_{1 \leq i \leq N} r_i$$

for each successful partition in N segments. Afterwards, if the N-th partition is successful, the global relative Frobenius error $\mathcal{E}_p(X_{\text{rec}}^N, \mathbf{r})$ is determined analogously to (3.20) as

$$\mathcal{E}_p(X_{\text{rec}}^N, \mathbf{r}) := \frac{\|X_{\text{in}} - X_{\text{rec}}^N\|_F}{\|X_{\text{in}}\|_F}.$$

This error clearly depends on the number of segments N and the rank vector $\mathbf{r}$. For $N = 1$, it reduces to the standard relative Frobenius error $\mathcal{E}(X_{\text{rec}}, r_1)$. As an additional error indicator, we compute the relative Frobenius error over time

$$\epsilon_k(X_{\text{rec}}^N, \mathbf{r}) := \frac{\|(X_{\text{in}})_{:,k} - (X_{\text{rec}}^N)_{:,k}\|_2}{\|(X_{\text{in}})_{:,k}\|_2} \quad \text{for } k = 1, \ldots, m+1. \tag{4.11}$$

Next, we evaluate whether the error $\mathcal{E}_p(X_{\text{rec}}^N, \mathbf{r})$ is below a given threshold $\tau > 0$. If this condition is satisfied, the algorithm terminates. Otherwise, for convenience, we assign $\mathcal{E}_p(X_{\text{rec}}^N, \mathbf{r})$ the placeholder value -100 and increment the partition number N by N_{inc}. This process is repeated until either the error $\mathcal{E}_p(X_{\text{rec}}^N, \mathbf{r})$ falls below the threshold τ, or the maximum number of partitions $N_{\max}$ is reached.

Remark 4.2. The choice of r_i in each partition is crucial. By default, the full rank $\text{rk}(X_i)$ is used initially and then decremented until the error criterion is satisfied. Alternatively, a fixed upper bound, such as $r_i = \min\{\nu, 200\}$, can be employed to reduce computational cost (see [2]). However, this alternative is not pursued further in this work.

The pDMD algorithm is summarized in Algorithm 4.

Algorithm 4 Piecewise Dynamic Mode Decomposition (pDMD)

INPUT $X_{\text{in}} \in \mathbb{R}^{n\times(m+1)}$, $\varepsilon, \tau > 0$, $N_{\text{in}}, N_{\text{inc}} \in \mathbb{N}$
Set $N_{\max} = \lfloor \frac{m+1}{10} \rfloor$
for $N = N_{\text{in}} : N_{\text{inc}} : N_{\max}$ **do**
  Set $\nu = \frac{m+1}{N}$
  for $i = 1 : 1 : N$ **do**
    Assemble $X_i = (X_{\text{in}})_{:,\lceil (i-1)\nu+1\rceil:\lceil i\nu\rceil}$
    Compute $r_i = \text{rk}(X_i)$
    Set $err(i) = 1$ and $flag_{\text{succ}} = \text{true}$
    while $err(i) > \varepsilon$ **do**
      $[\tilde{X}_i, \sim, \sim] = \text{rDMD}(X_i, r_i, 0, 2)$
      Compute $err(i) = \max\limits_{(i-1)\nu+1\le k\le i\nu} {\|(X_i)_{:,k} - (\tilde{X}_i)_{:,k}\|_\infty}/{\|(X_i)_{:,k}\|_\infty}$
      Update $r_i = r_i - 1$
      if $r_i \le 0$ **then**
        $flag_{\text{succ}} = \text{false}$
        Break
      end if
    end while
    if $flag_{\text{succ}} = \text{false}$ **then**
      Break
    end if
    Assemble $(X^N_{\text{rec}})_{:,\lceil (i-1)\nu+1\rceil:\lceil i\nu\rceil} = \tilde{X}_i$
  end for
  if $flag_{\text{succ}}$ **then**
    Set $\mathbf{r} = \begin{bmatrix} r_1 & \cdots & r_N \end{bmatrix}$
    Determine $\tilde{r}(N) = \max\limits_{1\le i\le N} r_i$
    Compute $\mathcal{E}_p(X^N_{\text{rec}}, \mathbf{r}) = {\|X_{\text{in}} - X^N_{\text{rec}}\|_F}/{\|X_{\text{in}}\|_F}$
    Compute $\epsilon_k(X^N_{\text{rec}}, \mathbf{r}) = {\|(X_{\text{in}})_{:,k} - (X^N_{\text{rec}})_{:,k}\|_2}/{\|(X_{\text{in}})_{:,k}\|_2}$ for $k = 1, \ldots, m+1$
    if $\mathcal{E}_p(X^N_{\text{rec}}, \mathbf{r}) < \tau$ **then**
      Break
    end if
  else
    Set $\mathcal{E}_p(X^N_{\text{rec}}, \mathbf{r}) = -100$
  end if
end for
RETURN X^N_{rec}, $\mathcal{E}_p(X^N_{\text{rec}}, \mathbf{r})$, $\epsilon_k(X^N_{\text{rec}}, \mathbf{r})$, $\tilde{r}(N)$, $\mathbf{r}$

4.4 Numerical Results for pDMD

4.4.1 Results on Periodic Datasets

In this section, we apply the proposed pDMD method to the datasets introduced in Section 4.2.1.

FitzHugh-Nagumo System: Limit Cycle
As a first example we consider the dataset X_{in}, generated by numerically solving the FHN system. The pDMD algorithm is initialized with parameters $N_{\text{in}} := 1$ and

$N_{\text{inc}} := 1$. The threshold values are set to $\varepsilon := 10^{-1}$ and $\tau := 10^{-6}$.

In Figure 15 (left), we show the relative Frobenius error $\mathcal{E}_p(X^N_{\text{rec}}, \mathbf{r})$ for those values of N that satisfy the error condition (4.10) for all $i = 1, \ldots, N$. The first successful value of N is $N = 14$, which corresponds to an error of $\mathcal{E}_p(X^{14}_{\text{rec}}, \mathbf{r}) = 1.816 \cdot 10^{-2}$. Then, the error initially remains relatively high and fluctuates around 10^{-2}, but begins to decrease significantly once N exceeds 60. Finally, the termination criterion of the pDMD algorithm is met for $N = 94$, where the relative Frobenius error is $\mathcal{E}_p(X^{94}_{\text{rec}}, \mathbf{r}) = 9.197 \cdot 10^{-7}$. Although the error curve has a few spikes, the overall trend is clearly decreasing. This indicates that increasing the number of partitions enables a more accurate approximation of the system dynamics. We note that for $N = 14$, the subdatasets X_i consist of 429 snapshots, while for $N = 94$, they consist of 64 snapshots. The right panel of Figure 15 illustrates the relative Frobenius error over time $\epsilon_k(X^N_{\text{rec}}, \mathbf{r})$ for the two successful values of N that yield the maximum and the minimum error: $N = 14$ and $N = 94$, respectively. For $N = 14$, the error exhibits significant fluctuations and peaks. In contrast, for $N = 94$, the error is considerably lower and the spikes are markedly reduced. This demonstrates the clear benefit of refined temporal partitioning in improving reconstruction quality.

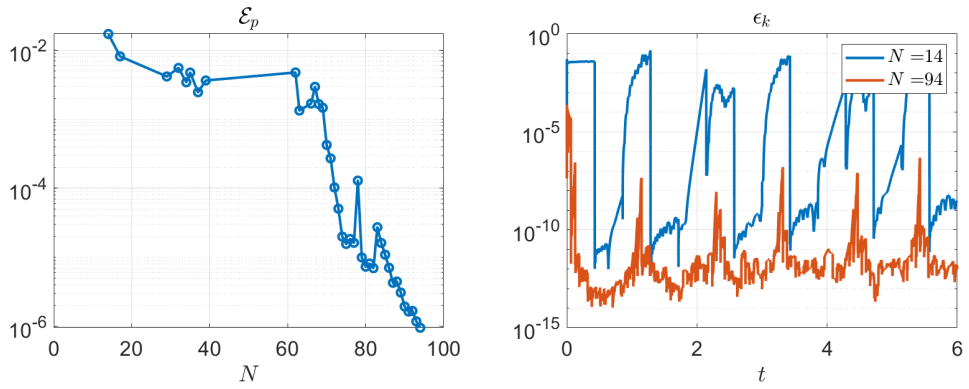


Figure 15: Left: relative Frobenius error $\mathcal{E}_p(X^N_{\text{rec}}, \mathbf{r})$ for those values of N that satisfy the error condition (4.10) for all $i = 1, \ldots, N$ until the algorithm terminates. Right: relative Frobenius error over time $\epsilon_k(X^N_{\text{rec}}, \mathbf{r})$ for $N = 14$ and $N = 94$.

Figure 16 (left, middle) shows a comparison of the spatial mean over time for u and v based on the pDMD reconstruction with $N = 94$. The right panel illustrates the corresponding phase plane $(\langle u \rangle, \langle v \rangle)$. All three plots demonstrate that pDMD is capable of accurately reproducing the periodic behavior of the FitzHugh-Nagumo system.

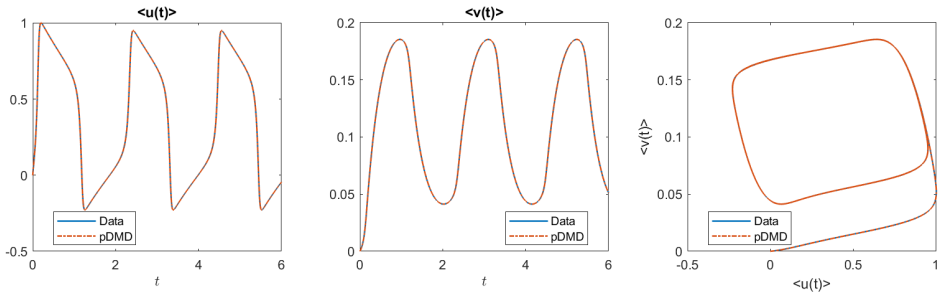


Figure 16: Comparison between the reference data and the pDMD reconstruction with $N = 94$. Left: spatial mean over time for u. Middle: spatial mean over time for v. Right: phase plane $(\langle u \rangle, \langle v \rangle)$.

Next, Figure 17 illustrates the behavior of the ranks for various successful values of N. The left panel shows the target ranks r_i of each submatrix X_i in the N-partition, for $N = 14$ and $N = 94$. We observe that both rank curves exhibit six peaks, corresponding to the maxima and minima of the spatial mean curve over time. Furthermore, we see that the ranks for $N = 94$ are, as expected, significantly lower than those for $N = 14$.
In the right panel, the maximum target rank $\tilde{r}(N)$ is plotted for each successful value of N. We note that each $\tilde{r}(N)$ is smaller than the rank of the original snapshot matrix X_{in}, which is $\text{rk}(X_{\text{in}}) = 51$. Moreover, it is evident that from approximately $N = 60$ onward, the maximum target rank $\tilde{r}(N)$ stabilizes at a relatively low level between 15 and 18. This underlines the computational advantage of pDMD, as it enables a significant reduction in problem dimensionality.

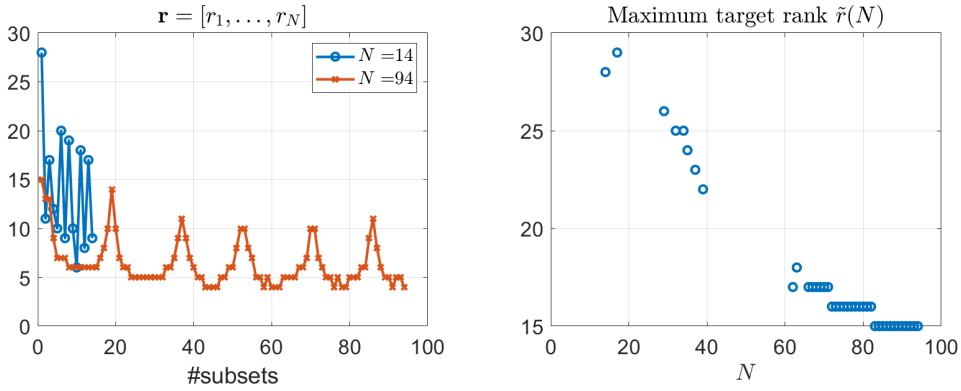


Figure 17: Left: target ranks r_i of each submatrix X_i in the N-partition for $N = 14$ and $N = 94$. Right: maximum target rank $\tilde{r}(N)$ for each successful value of N.

λ-ω System: Spiral Waves

As a second example, we apply the pDMD method to the λ-ω system dataset X_{in}. Here, the pDMD algorithm is initialized with $N_{\text{in}} := 1$, $N_{\text{inc}} := 1$ and threshold values $\varepsilon := 10^{-2}$, $\tau := 10^{-6}$.

As before, we begin by analyzing the relative Frobenius errors. This time, the first successful value of N is $N = 3$, yielding an error of $\mathcal{E}_p(X^3_{\text{rec}}, \mathbf{r}) = 6.327 \cdot 10^{-5}$. We observe that the error initially decreases and then exhibits a slightly oscillatory behavior

for $12 \leq N \leq 26$, such that $1.6 \cdot 10^{-5} \leq \mathcal{E}_p(X^N_{\text{rec}}, \mathbf{r}) \leq 1.1 \cdot 10^{-6}$ holds. For $N = 27$, the error falls below the threshold τ with a value of $\mathcal{E}_p(X^{27}_{\text{rec}}, \mathbf{r}) = 7.718 \cdot 10^{-7}$. It is worth noting that the relative Frobenius error is for each N significantly lower than the error obtained by the global rDMD method (cf. Figure 10, left). Again, we remark that for $N = 3$, the subdatasets X_i consist of 4167 snapshots, while for $N = 27$, the subdatasets contain 463 snapshots.
For $N = 3$ and $N = 27$, the right panel of Figure 18 shows the relative Frobenius error over time $\epsilon_k(X^N_{\text{rec}}, \mathbf{r})$. We can clearly distinguish two temporal regimes, $[0, \bar{t}]$ and $(\bar{t}, T]$, in the error behavior, particularly for $N = 27$, which we had already identified in Section 4.2.1. In general, we also observe that the relative Frobenius error over time is considerably larger for $N = 3$ than for $N = 27$.

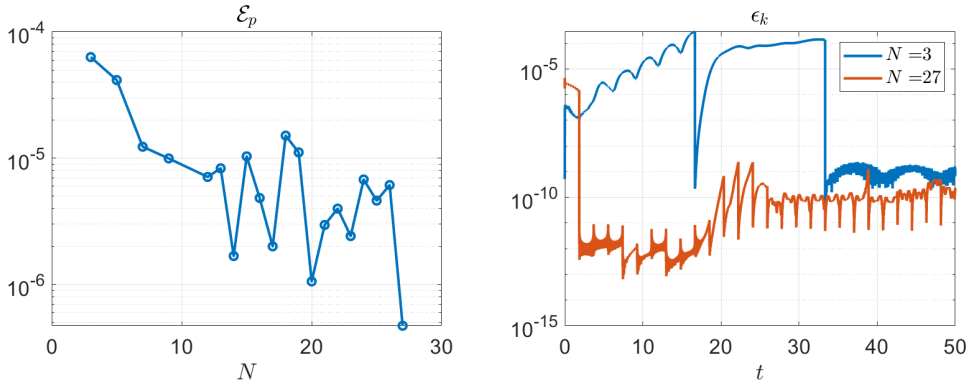


Figure 18: Left: relative Frobenius error $\mathcal{E}_p(X^N_{\text{rec}}, \mathbf{r})$ for those values of N that satisfy the error condition (4.10) for all $i = 1, \ldots, N$ until the algorithm terminates. Right: relative Frobenius error over time $\epsilon_k(X^N_{\text{rec}}, \mathbf{r})$ for $N = 3$ and $N = 27$.

As for the FHN system, we observe in Figure 19 that the spatial means over time and the phase plane $(\langle u \rangle, \langle v \rangle)$ of the reference data and the pDMD reconstruction with $N = 27$ match perfectly.

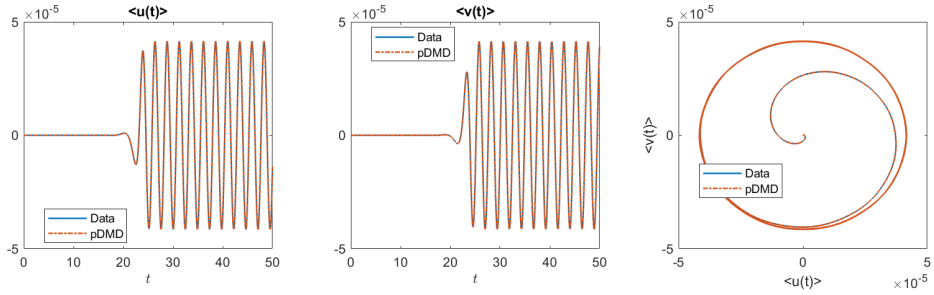


Figure 19: Comparison between the reference data and the pDMD reconstruction with $N = 27$. Left: spatial mean over time for u. Middle: spatial mean over time for v. Right: phase plane $(\langle u \rangle, \langle v \rangle)$.

Finally, Figure 20 illustrates the behavior of the ranks for various successful values of N. In the left panel, the ranks for $N = 27$ are, as expected, significantly lower than those for $N = 3$.
The maximum target ranks $\tilde{r}(N)$ are shown in the right panel and exhibit a decreasing trend as N increases. We again observe that $\tilde{r}(N)$ is consistently lower than the rank

of the original snapshot matrix X_{in} ($\text{rk}(X_{\text{in}}) = 314$). In particular, the maximum observed target rank $\tilde{r}(N)$ is 140, occurring at $N = 3$.

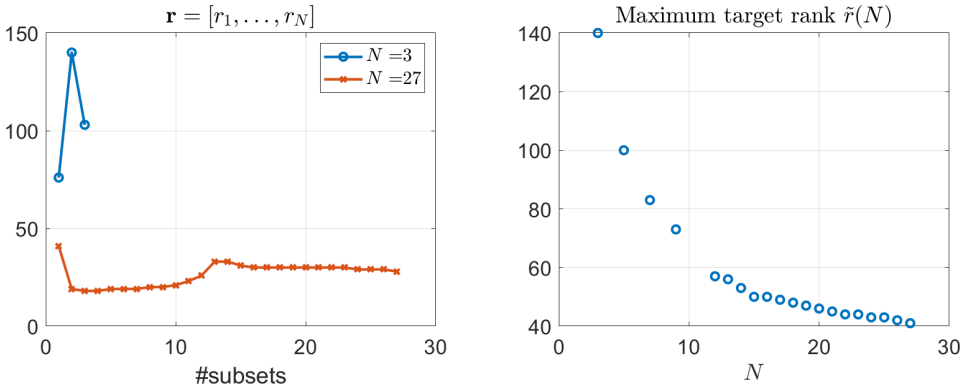


Figure 20: Left: target ranks r_i of each submatrix X_i in the N-partition for $N = 3$ and $N = 27$. Right: maximum target rank $\tilde{r}(N)$ for each successful value of N.

4.4.2 Results on a Dataset with Spatio-Temporal Turing Instability

DIB Morphochemical System

As a third and final example, we apply the pDMD method to the DIB morphochemical system dataset X_{in}. We initialize the pDMD algorithm with $N_{\text{in}} := 1$, $N_{\text{inc}} := 1$ and use the thresholds $\varepsilon := 10^{-3}$, $\tau := 10^{-6}$.

The first successful value of N is $N = 29$, which yields an error of $\mathcal{E}_p(X^8_{\text{rec}}, \mathbf{r}) = 3.023 \cdot 10^{-6}$. Afterwards, the error sinks for $N = 31$, but then rises again to a maximum of $3.530 \cdot 10^{-6}$ for $N = 47$. Finally, the error falls below the threshold τ for $N = 48$ ($\mathcal{E}_p(X^{48}_{\text{rec}}, \mathbf{r}) = 1.383 \cdot 10^{-7}$). Thus, convergence is achieved after only four successful partitioning steps. It is worth noting that for all successful values of N, the relative Frobenius error is consistently small (cf. Figure 21, left) on the order of magnitude

$$1.383 \cdot 10^{-7} \leq \mathcal{E}_p(X^N_{\text{rec}}, \mathbf{r}) \leq 3.530 \cdot 10^{-6}.$$

In the right panel of Figure 21, we report the relative Frobenius error over time $\epsilon_k(X^N_{\text{rec}}, \mathbf{r})$ for the two values of N at which the maximum and minimum errors occur, namely $N = 47$ and $N = 48$. We observe that both error curves exhibit a similar behavior: the maximum error occurs at the beginning – corresponding to the reactivity zone $\mathcal{I}_1$ – and then decreases almost uniformly over time.

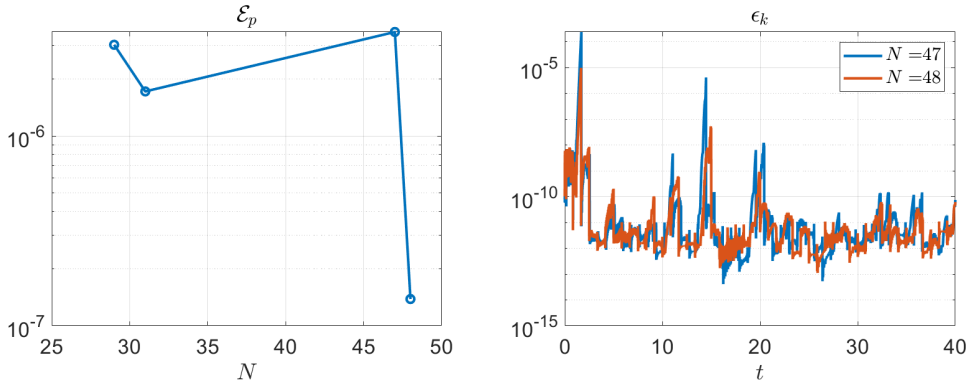


Figure 21: Left: relative Frobenius error $\mathcal{E}_p(X_{\text{rec}}^N, \mathbf{r})$ for those values of N that satisfy the error condition (4.10) for all $i = 1, \ldots, N$ until the algorithm terminates. Right: relative Frobenius error over time $\epsilon_k(X_{\text{rec}}^N, \mathbf{r})$ for $N = 47$ and $N = 48$.

In Figure 22, we once again observe excellent agreement between the spatial means over time of the reference data and those of the pDMD reconstruction with $N = 48$, as well as in the corresponding phase plane.

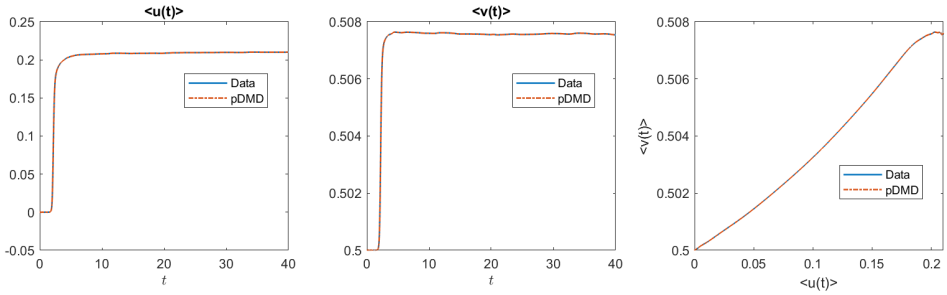


Figure 22: Comparison between the reference data and the pDMD reconstruction with $N = 48$. Left: spatial mean over time for u. Middle: spatial mean over time for v. Right: phase plane $(\langle u \rangle, \langle v \rangle)$.

Finally, we revisit the rank behavior. It is evident that the subdatasets corresponding to the reactivity zone $\mathcal{I}_1$ require significantly higher ranks, whereas those associated with the stability zone $\mathcal{I}_2$ can be approximated with substantially lower ranks.
The maximum target ranks $\tilde{r}(N)$ are shown in the right panel of Figure 23 and, as in previous examples, exhibit a decreasing trend with increasing N. Moreover, $\tilde{r}(N)$ consistently remains below the rank of the original snapshot matrix X_{in} ($\text{rk}(X_{\text{in}}) = 303$). The highest observed value is $\tilde{r}(N) = 46$, which occurs at $N = 27$.

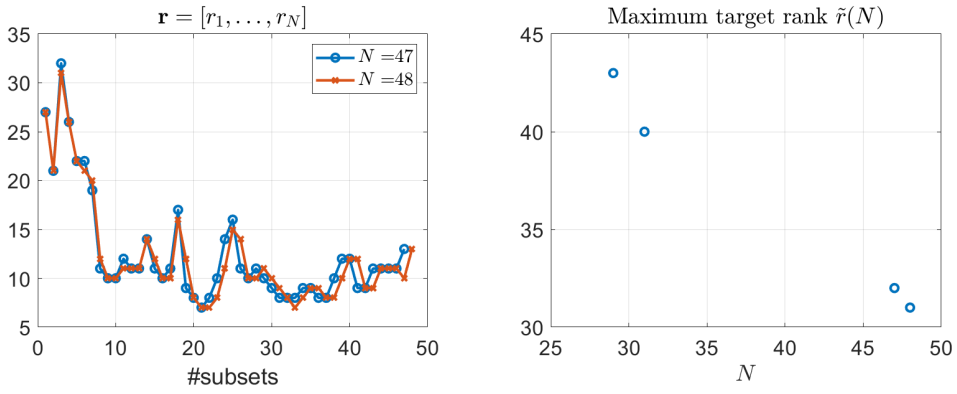


Figure 23: Left: target ranks r_i of each submatrix X_i in the N-partition for $N = 47$ and $N = 48$. Right: maximum target rank $\tilde{r}(N)$ for each successful value of N.

We conclude that pDMD accurately reconstructs the dynamics of the FHN system, the λ-ω system and the DIB morphochemical system. While the global rDMD method struggles to reconstruct the temporal datasets from these systems – often due to ill-conditioning for large target ranks – the pDMD method achieves accurate reconstructions by operating with significantly lower target ranks.
We observe, in particular, that as the number of successful segments N increases, the relative Frobenius error $\mathcal{E}_p(X^N_{\text{rec}}, \mathbf{r})$ decreases and remains considerably lower than that of the global rDMD approach. The spatial means over time, the phase plane and the full spatio-temporal behavior – including transitions between different temporal regimes – are all captured very accurately by the pDMD method.
These findings highlight a key limitation of the global rDMD approach: its reliance on a single linear approximation over the entire time horizon prevents it from capturing the rich variety of dynamical regimes that may emerge, such as oscillatory behavior and Turning pattern formation. In contrast, pDMD constructs local linear models within suitably chosen data segments, thereby enabling a more accurate representation of spatio-temporal dynamics.

5 Adaptive POD-DEIMc

The main references for this section are [1] and [7].

After exploring DMD and its variants, we introduce additional *model order reduction (MOR) techniques*, which – like DMD – are also strongly related to the SVD and aim to reduce the computational costs of standard numerical methods. We begin by considering the (discrete) *Proper Orthogonal Decomposition (POD)*. Building upon this foundational method, we then introduce the *Discrete Empirical Interpolation Method (DEIM)* to achieve the faster, albeit less accurate, *POD-DEIM* method. To address the accuracy limitations, we propose stabilized variants: the *corrected POD (PODc)* and the *corrected POD-DEIM (POD-DEIMc)*. Both of these corrected methods systematically incorporate carefully derived correction terms to enhance their accuracy. Finally, we present an adaptive approach for these methods to further enhance efficiency and accuracy.
In contrast to Section 4, where only coupled approaches are used, we consider uncoupled approaches that do not stack the datasets for each variable into a single snapshot matrix.

As an illustrative example for the following techniques, we again consider a two-equation RD-PDE system as introduced in Section 4. Let

$$\Omega := (0, L_{x_1}) \times (0, L_{x_2}) \subset \mathbb{R}^2$$

denote a non-empty and bounded spatial domain with Lipschitz-continuous boundary $\Gamma := \partial\Omega$. We suppose the parameter choices $L_{x_1} := 20$ and $L_{x_2} := 20$. For a fixed final time $T := 100$, we define

$$\begin{aligned}\Omega_T &:= (0, T] \times \Omega,\\ \Gamma_T &:= [0, T] \times \Gamma.\end{aligned}$$

In contrast to the previous section, we restrict our attention to a system with homogeneous Neumann boundary conditions. All other aspects of the system remain unchanged, so it is again of the form (4.1). Furthermore, we suppose that there exists at least one spatially homogeneous equilibrium (cf. Section 4.2.2)

$$\begin{aligned}P_e &:= (u_e, v_e)\\ &:= (0, \alpha) \in \mathbb{R}^2,\end{aligned}$$

i.e., we have

$$\begin{aligned}d_u(u_e, v_e) &= 0,\\ d_v(u_e, v_e) &= 0.\end{aligned}$$

The reaction terms are given by (4.9). Consequently, we are once again working with the DIB morphochemical system. The diffusion constants, system parameters and initial conditions are kept identical to those used in Section 4.2.2.

We spatially discretize the domain using $n_{x_1} := 10^2$ and $n_{x_2} := 10^2$ equidistant interior points in the x_1- and x_2-direction, respectively. This results in a total of

$$n := n_{x_1} n_{x_2} = 10^4$$

interior grid points. The spatial step sizes are given by

$$\Delta x_1 := \frac{L_{x_1}}{n_{x_1}+1},$$
$$\Delta x_2 := \frac{L_{x_2}}{n_{x_2}+1}.$$

As in (4.6), we derive the FOM for the DIB morphochemical system as

$$\begin{aligned} \dot{\mathbf{u}} - \mu_u \mathbf{A}\mathbf{u} + \mathbf{d}_{\mathbf{u}}(\mathbf{u},\mathbf{v}) &= 0_n \quad \text{for } t \in (0,T], \\ \dot{\mathbf{v}} - \mu_v \mathbf{A}\mathbf{v} + \mathbf{d}_{\mathbf{v}}(\mathbf{u},\mathbf{v}) &= 0_n \quad \text{for } t \in (0,T], \\ \mathbf{u}(0) &= \mathbf{u}_0, \\ \mathbf{v}(0) &= \mathbf{v}_0. \end{aligned} \tag{5.1}$$

For the temporal discretization, we first make use of the IMEX Euler method in vector form introduced in Section 4.2.1. Given the equidistant time grid

$$t_k := k\Delta t \quad \text{for } k = 0,\dots,m,$$

with step size $\Delta t := 10^{-3}$ and $m := {}^{T}/_{\Delta t} = 10^5$, the fully discretized system reads:

$$\begin{aligned} (I_n - \Delta t \mu_u \mathbf{A})\,\mathbf{u}^k &= \mathbf{u}^{k-1} - \Delta t \mathbf{d}_{\mathbf{u}}(\mathbf{u}^{k-1},\mathbf{v}^{k-1}) \quad \text{for } k = 1,\dots,m, \\ (I_n - \Delta t \mu_u \mathbf{A})\,\mathbf{v}^k &= \mathbf{v}^{k-1} - \Delta t \mathbf{d}_{\mathbf{v}}(\mathbf{u}^{k-1},\mathbf{v}^{k-1}) \quad \text{for } k = 1,\dots,m, \\ \mathbf{u}^0 &= \mathbf{u}_0, \\ \mathbf{v}^0 &= \mathbf{v}_0. \end{aligned}$$

Next, since both the spatial dimension n and the number m of problems to be solved are large, we introduce the IMEX Euler method in matrix form (cf. Section 4.2.1), which is given by

$$\begin{aligned} \left(I_{n_{x_1}} - \Delta t \mu_u \mathbf{T}_{x_1}\right)\mathbf{U}^k + \mathbf{U}^k\left(-\Delta t \mu_u \mathbf{T}_{x_2}^\top\right) &= \mathbf{U}^{k-1} - \Delta t \mathbf{d}_{\mathbf{u}}(\mathbf{U}^{k-1},\mathbf{V}^{k-1}), \\ \left(I_{n_{x_1}} - \Delta t \mu_v \mathbf{T}_{x_1}\right)\mathbf{V}^k + \mathbf{V}^k\left(-\Delta t \mu_v \mathbf{T}_{x_2}^\top\right) &= \mathbf{V}^{k-1} - \Delta t \mathbf{d}_{\mathbf{v}}(\mathbf{U}^{k-1},\mathbf{V}^{k-1}) \end{aligned} \tag{5.2a}$$

for $k = 1,\dots,m$, together with the initial conditions

$$\mathbf{U}^0 = \mathbf{U}_0, \tag{5.2b}$$
$$\mathbf{V}^0 = \mathbf{V}_0. \tag{5.2c}$$

Moreover, to reduce storage requirements, we store only every κ-th snapshot, with $\kappa := 4$. For ease of notation, we relabel the stored snapshots consecutively as $\mathbf{u}^0, \mathbf{u}^1, \dots, \mathbf{u}^{m_\kappa}$, where $\mathbf{u}^k \approx u(t_{\kappa k})$, $t_{\kappa k} := \kappa k \Delta t$ and $m_\kappa := {}^{m}/_{\kappa}$.

Table 3 highlights the computational advantage of the IMEX Euler method in matrix form over its vector-based counterpart in terms of running time for solving the full-order DIB morphochemical system.

Table 3: Comparison of the running times for the IMEX Euler method in vector form versus matrix form for solving the DIB morphochemical system.

	IMEX Euler vector form	IMEX Euler matrix form
Running time $[s]$	983.01	230.5

The numerical solutions for u and v at the final time T are shown in Figure 24 (top), where a characteristic Turing pattern emerges. In the lower part of the figure, two indicators are presented. On the left, we display the spatial mean over time for u and v, as defined in (4.5b). On the right, we plot the increments $\mathrm{inc}(u)_k$ and $\mathrm{inc}(v)_k$, defined by

$$\begin{aligned} \mathrm{inc}(u)_k &:= \|\mathbf{u}^{k+1} - \mathbf{u}^k\|_2 \quad \text{for } k = 0, \dots, m_\kappa - 1, \\ \mathrm{inc}(v)_k &:= \|\mathbf{v}^{k+1} - \mathbf{v}^k\|_2 \quad \text{for } k = 0, \dots, m_\kappa - 1. \end{aligned}$$

We observe that the behavior of u and v is very similar across all four plots. Therefore, in the following discussion, we focus solely on u as a reference. Moreover, the two indicators clearly reveal the existence of two distinct temporal regimes, $\mathcal{I}_1 := [0, \bar{t}]$ and $\mathcal{I}_2 := (\bar{t}, T]$, where

$$\bar{t} := 2.124 = \kappa \Delta t \, \mathrm{argmax}_{k \in \{0, \dots, m_\kappa - 1\}} \, \mathrm{inc}(u)_k. \tag{5.3}$$

In the first regime, referred to as the reactivity zone $\mathcal{I}_1$, the spatial mean increases sharply over time and the increments are relatively large. In contrast, in the second regime, the stability zone $\mathcal{I}_2$, both indicators gradually stabilize toward constant values. This behavior is consistent with the fact that a Turing pattern is an asymptotic solution of (4.1), implying that the spatial mean converges to a constant value and the temporal increments decay to zero.

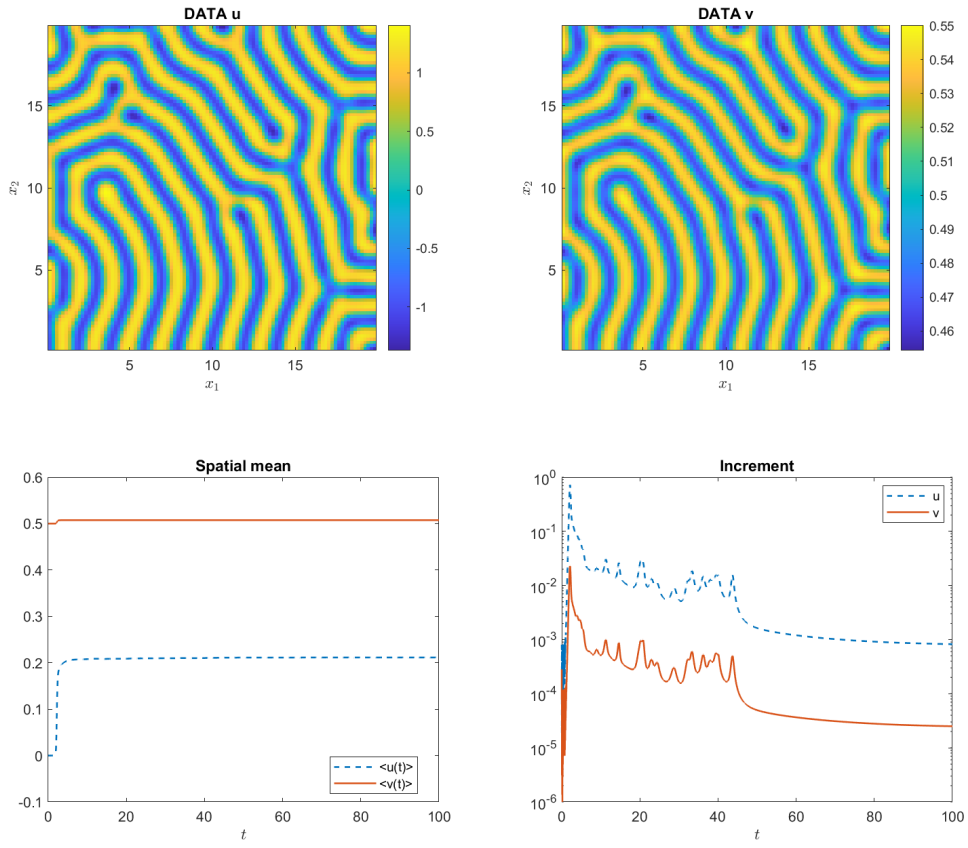


Figure 24: Top: variable u (left) and v (right) at the final time T. Bottom left: spatial mean over time for u and v. Bottom right: increment over time for u and v.

5.1 Model Order Reduction

To reduce the computational cost associated with solving the FOM, we employ the (discrete) POD method as a model order reduction technique. The goal of the POD method is to approximate the high-dimensional discretized system by projecting it onto a significantly lower-dimensional subspace of dimension $1 \leq r \ll n$. This is achieved by identifying a set of orthonormal basis functions that optimally capture the essential information contained in the snapshot data.
To clarify what is meant by 'information' in this context, consider the snapshots $\mathbf{u}^0, \ldots, \mathbf{u}^{m_\kappa} \in \mathbb{R}^n$. The discrete POD problem then consists in finding an orthonormal set $\psi_1, \ldots, \psi_r \in \mathbb{R}^n$ that minimizes the mean square error between the elements $\mathbf{u}^k$ and their r-th partial Fourier sum on average:

$$\min_{\psi_1,\ldots,\psi_r \in \mathbb{R}^n} \sum_{k=0}^{m_\kappa} \left\| \mathbf{u}^k - \sum_{i=1}^{r} \langle \mathbf{u}^k, \psi_i \rangle_2 \psi_i \right\|_2^2 \quad \text{s.t. } \langle \psi_i, \psi_j \rangle_2 = \delta_{ij}, 1 \leq i, j \leq r, \tag{5.4}$$

where 's.t.' stand for 'subject to'. An optimal solution to this problem is referred to as a *POD basis of rank* r. It is well-known (see, e.g., [12]) that the solution to the optimization problem (5.4) is given by the first r left singular vectors of the snapshot

matrix

$$S_{\mathbf{u}} := \begin{bmatrix} \mathbf{u}^0 & \cdots & \mathbf{u}^{m_\kappa} \end{bmatrix} \in \mathbb{R}^{n\times(m_\kappa+1)}.$$

The choice of the reduced dimension r is a crucial aspect in the practical application of POD, as there is no general a priori rule for selecting it. A commonly used criterion is the *energy ratio*, which measures the proportion of the system's total energy captured by the first r POD modes. It is defined as the ratio of the modeled energy to the total energy of the system:

$$\mathcal{E}(r) := \frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_{i=1}^{\min\{n,m_\kappa+1\}} \sigma_i^2} \in [0,1].$$

Here, $\sigma_1, \ldots, \sigma_{\min\{n,m_\kappa+1\}}$ are the singular values of the snapshot matrix $S_{\mathbf{u}}$. This ratio can also be determnined as

$$\mathcal{E}(r) = \frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_{k=0}^{m_\kappa} \|\mathbf{u}^k\|_2^2}.$$

Thus, the computation of the singular values $\sigma_{r+1}, \ldots, \sigma_{\min\{n,m_\kappa+1\}}$ can be omitted, as they are not required for evaluating the energy ratio. This is much more efficient since the computation of each singular value is not necessary.

Nonetheless, we determine the singular values of the snapshot matrix $S_{\mathbf{u}}$ using `svd(`$S_{\mathbf{u}}$`, 'econ')` and plot the largest 1000 of them (see Figure 25). We observe a very slow decay of the singular values, which is justified by the complex structure of the Turing pattern. We suggest that the rank r can be chosen as $r := 175$, where the corresponding singular values is $\sigma_{175} = 4.273 \cdot 10^{-4}$. In fact, the energy ratio $\mathcal{E}(r)$ is very close to 1 with respect to the snapshot matrix $S_{\mathbf{u}}$ such that the previous choice of r is reasonable:

$$\mathcal{E}(r) \approx 0.99.$$

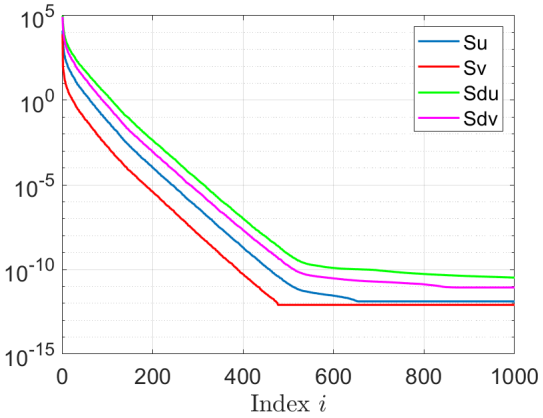


Figure 25: Largest 1000 singular values of the snapshot matrices $S_{\mathbf{u}}$, $S_{\mathbf{v}}$, $S_{\mathbf{d_u}}$ and $S_{\mathbf{d_v}}$.

Remark 5.1. There are several approaches to compute the left singular vectors and singular values of the snapshot matrix $S_{\mathbf{u}}$.
The most accurate and numerically stable method is to directly compute the SVD of $S_{\mathbf{u}}$. However, this approach can be computationally expensive, especially for large matrices.
A more efficient alternative is the so-called *method of snapshots*, which computes the eigenpairs (λ_i, ϕ_i) of the matrix $S_{\mathbf{u}}^\top S_{\mathbf{u}}$. The corresponding left singular vectors are then obtained as

$$\psi_i = \frac{1}{\sqrt{\lambda_i}} S_{\mathbf{u}} \phi_i$$

and the singular values are given by $\sigma_i = \sqrt{\lambda_i}$ for all i with $\lambda_i > 0$. The left singular vectors and singular values corresponding to $\lambda_i = 0$ are mostly not of interest, as they do not contribute to the system's energy. This method is particularly suitable when $S_{\mathbf{u}}$ has significantly more rows than columns. Nevertheless, it is generally less accurate and less stable than computing the full SVD.
A third approach is to compute the eigenpairs of the matrix $S_{\mathbf{u}} S_{\mathbf{u}}^\top$. This is advantageous when $S_{\mathbf{u}}$ has more columns than rows. In this case, the singular values are again given by $\sigma_i = \sqrt{\lambda_i}$, and the corresponding eigenvectors are the left singular vectors. Similar to the method of snapshots, this approach is computationally efficient, but it may offer reduced numerical robustness in certain scenarios.

Let $\psi_{\mathbf{u}}^1, \ldots, \psi_{\mathbf{u}}^r$ denote the rank-r POD basis associated with the snapshot matrix $S_{\mathbf{u}}$, and similarly, let $\psi_{\mathbf{v}}^1, \ldots, \psi_{\mathbf{v}}^r$ denote the rank-r POD basis corresponding to the snapshot matrix

$$S_{\mathbf{v}} := \begin{bmatrix} \mathbf{v}^0 & \cdots & \mathbf{v}^{m_\kappa} \end{bmatrix} \in \mathbb{R}^{n \times (m_\kappa + 1)}.$$

We collect the POD basis vectors in the matrices

$$\begin{aligned} \Psi_{\mathbf{u}} &:= \begin{bmatrix} \psi_{\mathbf{u}}^1 & \cdots & \psi_{\mathbf{u}}^r \end{bmatrix} \in \mathbb{R}^{n \times r}, \\ \Psi_{\mathbf{v}} &:= \begin{bmatrix} \psi_{\mathbf{v}}^1 & \cdots & \psi_{\mathbf{v}}^r \end{bmatrix} \in \mathbb{R}^{n \times r}, \end{aligned}$$

where each column corresponds to a POD basis vector for the variables $\mathbf{u}$ and $\mathbf{v}$, respectively. These matrices define the reduced-order POD approximations of the state variables as

$$\begin{aligned} \mathbf{u}(t) &\approx \Psi_{\mathbf{u}} \tilde{\mathbf{u}}_r(t) \in \mathbb{R}^n \quad \text{for } t \in [0, T], \\ \mathbf{v}(t) &\approx \Psi_{\mathbf{v}} \tilde{\mathbf{v}}_r(t) \in \mathbb{R}^n \quad \text{for } t \in [0, T], \end{aligned} \tag{5.5}$$

where the unknown, reduced state trajectories

$$\begin{aligned} \tilde{\mathbf{u}}_r &: [0, T] \to \mathbb{R}^r, \\ \tilde{\mathbf{v}}_r &: [0, T] \to \mathbb{R}^r \end{aligned}$$

represent the time-dependent coefficients corresponding to the reduced subspaces. By substituting the reduced-order approximations (5.5) into the FOM (5.1) and projecting

onto the reduced subspaces via multiplication with the transposed POD basis matrices, we obtain the so-called POD *reduced order model (ROM)*

$$\begin{aligned} \dot{\tilde{\mathbf{u}}}_r - \mu_u \mathbf{A}_r \tilde{\mathbf{u}}_r + \mathbf{d}_{\mathbf{u}}^r(\tilde{\mathbf{u}}_r, \tilde{\mathbf{v}}_r) &= 0_r \qquad \text{for } t \in (0,T], \\ \dot{\tilde{\mathbf{v}}}_r - \mu_v \mathbf{B}_r \tilde{\mathbf{v}}_r + \mathbf{d}_{\mathbf{v}}^r(\tilde{\mathbf{u}}_r, \tilde{\mathbf{v}}_r) &= 0_r \qquad \text{for } t \in (0,T], \\ \tilde{\mathbf{u}}_r(0) &= \Psi_{\mathbf{u}}^\top \mathbf{u}_0, \\ \tilde{\mathbf{v}}_r(0) &= \Psi_{\mathbf{v}}^\top \mathbf{v}_0, \end{aligned} \tag{5.6}$$

where we use the orthogonalities $\Psi_{\mathbf{u}}^\top \Psi_{\mathbf{u}} = I_r$ and $\Psi_{\mathbf{v}}^\top \Psi_{\mathbf{u}} = I_r$. Here, $\mathbf{A}_r$ and $\mathbf{B}_r$ denote the reduced diffusion matrices, defined by

$$\begin{aligned} \mathbf{A}_r &:= \Psi_{\mathbf{u}}^\top \mathbf{A} \Psi_{\mathbf{u}} \in \mathbb{R}^{r \times r}, \\ \mathbf{B}_r &:= \Psi_{\mathbf{v}}^\top \mathbf{A} \Psi_{\mathbf{v}} \in \mathbb{R}^{r \times r}. \end{aligned}$$

The nonlinear kinetic terms are defined as

$$\begin{aligned} \mathbf{d}_{\mathbf{u}}^r(\tilde{\mathbf{u}}_r, \tilde{\mathbf{v}}_r) &:= \Psi_{\mathbf{u}}^\top \mathbf{d}_{\mathbf{u}}(\Psi_{\mathbf{u}} \tilde{\mathbf{u}}_r, \Psi_{\mathbf{v}} \tilde{\mathbf{v}}_r) \in \mathbb{R}^r, \\ \mathbf{d}_{\mathbf{u}}^r(\tilde{\mathbf{u}}_r, \tilde{\mathbf{v}}_r) &:= \Psi_{\mathbf{v}}^\top \mathbf{d}_{\mathbf{v}}(\Psi_{\mathbf{u}} \tilde{\mathbf{u}}_r, \Psi_{\mathbf{v}} \tilde{\mathbf{v}}_r) \in \mathbb{R}^r. \end{aligned} \tag{5.7}$$

Remark 5.2.

(i) It is important to note that the matrices $\mathbf{A}_r$ and $\mathbf{B}_r$ are not necessarily in Kronecker form due to the projection process. Moreover, they are small, dense matrices of size $r \times r$, in contrast to the large, sparse matrices $\mathbf{A}$ and $\mathbf{B}$ of size $n \times n$. Consequently, we apply the IMEX Euler method in vector form to the reduced system (5.6), since a Sylvester-type formulation is no longer feasible.

(ii) To efficiently solve (5.6), several components can be precomputed during the *offline stage*, i.e., before time integration begins. This includes the computation of the snapshot matrices $S_{\mathbf{u}}$ and $S_{\mathbf{v}}$, the POD basis matrices $\Psi_{\mathbf{u}}$ and $\Psi_{\mathbf{v}}$ and the projected diffusion matrices $\mathbf{A}_r$ and $\mathbf{B}_r$. By doing so, we avoid redundant computations during the *online stage*, thereby significantly accelerating the time integration process.

Finally, the fully discretized POD-ROM reads:

$$\begin{aligned} (I_r - \Delta t \mu_u \mathbf{A}_r)\, \tilde{\mathbf{u}}_r^k &= \tilde{\mathbf{u}}_r^{k-1} - \Delta t \mathbf{d}_{\mathbf{u}}^r(\tilde{\mathbf{u}}_r^{k-1}, \tilde{\mathbf{v}}_r^{k-1}), \quad \text{for } k = 1, \dots, m, \\ (I_r - \Delta t \mu_v \mathbf{B}_r)\, \tilde{\mathbf{v}}_r^k &= \tilde{\mathbf{v}}_r^{k-1} - \Delta t \mathbf{d}_{\mathbf{v}}^r(\tilde{\mathbf{u}}_r^{k-1}, \tilde{\mathbf{v}}_r^{k-1}), \quad \text{for } k = 1, \dots, m, \\ \tilde{\mathbf{u}}_r^0 &= \Psi_{\mathbf{u}}^\top \mathbf{u}_0, \\ \tilde{\mathbf{v}}_r^0 &= \Psi_{\mathbf{v}}^\top \mathbf{v}_0. \end{aligned} \tag{5.8}$$

Unfortunately, system (5.8) still depends on the full dimension since the kinetics are evaluated at vectors of dimension n, as can be seen in (5.7). Therefore, we introduce the DEIM method, which interpolates the nonlinear functions using only l, $1 \leq l \ll n$, interpolation points. Thus, we need a *DEIM basis of rank l*. Let

$$\begin{aligned} S_{\mathbf{d_u}} &:= \left[\begin{array}{ccc} \mathbf{d}_{\mathbf{u}}(\mathbf{u}^0, \mathbf{v}^0) & \cdots & \mathbf{d}_{\mathbf{u}}(\mathbf{u}^{m_\kappa}, \mathbf{v}^{m_\kappa}) \end{array} \right] \in \mathbb{R}^{n \times (m_\kappa + 1)}, \\ S_{\mathbf{d_v}} &:= \left[\begin{array}{ccc} \mathbf{d}_{\mathbf{v}}(\mathbf{u}^0, \mathbf{v}^0) & \cdots & \mathbf{d}_{\mathbf{v}}(\mathbf{u}^{m_\kappa}, \mathbf{v}^{m_\kappa}) \end{array} \right] \in \mathbb{R}^{n \times (m_\kappa + 1)} \end{aligned}$$

be the snapshot matrices of the kinetic terms. The DEIM basis is then obtained by computing the POD basis of rank l of the snapshot matrices $S_{\mathbf{d_u}}$ and $S_{\mathbf{d_v}}$. As rank-l value, we choose the the maximum between the ranks of the snapshot matrices $S_{\mathbf{d_u}}$ and $S_{\mathbf{d_v}}$:

$$l := \max\{\mathrm{rk}(S_{\mathbf{d_u}}), \mathrm{rk}(S_{\mathbf{d_v}})\} = \max\{362, 343\} = 362.$$

We denote the DEIM basis vectors as $\phi^1_{\mathbf{d_u}}, \dots, \phi^l_{\mathbf{d_u}}$ and $\phi^1_{\mathbf{d_v}}, \dots, \phi^l_{\mathbf{d_v}}$ and collect them in the matrices

$$\Phi_{\mathbf{d_u}} := \begin{bmatrix} \phi^1_{\mathbf{d_u}} & \cdots & \phi^l_{\mathbf{d_u}} \end{bmatrix} \in \mathbb{R}^{n\times l},$$
$$\Phi_{\mathbf{d_v}} := \begin{bmatrix} \phi^1_{\mathbf{d_v}} & \cdots & \phi^l_{\mathbf{d_v}} \end{bmatrix} \in \mathbb{R}^{n\times l}.$$

Additionally, we determine the singular values of the snapshot matrices $S_{\mathbf{d_u}}$ and $S_{\mathbf{d_v}}$ (see Figure 25). The decay of the singular values is even a bit slower than that of the snapshot matrices $S_{\mathbf{u}}$ and $S_{\mathbf{v}}$.

Next, we apply a pivoting strategy by performing QR decompositions with column pivoting on the transposed DEIM basis matrices $\Phi^\top_{\mathbf{d_u}}$ and $\Phi^\top_{\mathbf{d_v}}$:

$$\Phi^\top_{\mathbf{d_u}} P_{\mathbf{d_u}} = Q_{\mathbf{d_u}} R_{\mathbf{d_u}},$$
$$\Phi^\top_{\mathbf{d_v}} P_{\mathbf{d_v}} = Q_{\mathbf{d_v}} R_{\mathbf{d_v}}.$$

From the resulting $n \times n$ permutation matrices $P_{\mathbf{d_u}}, P_{\mathbf{d_v}}$, we retain only the first l columns, resulting in relabeled matrices of size $n \times l$:

$$P_{\mathbf{d_u}}, P_{\mathbf{d_v}} \in \mathbb{R}^{n\times l}.$$

We call the columns of $P_{\mathbf{d_u}}$ and $P_{\mathbf{d_v}}$ *DEIM points*. In the offline stage, we compute the matrices

$$P^\top_{\mathbf{d_u}}\Psi_{\mathbf{u}}, P^\top_{\mathbf{d_v}}\Psi_{\mathbf{u}}, P^\top_{\mathbf{d_u}}\Psi_{\mathbf{v}}, P^\top_{\mathbf{d_v}}\Psi_{\mathbf{v}} \in \mathbb{R}^{l\times r} \tag{5.9}$$

and the inverses

$$(P^\top_{\mathbf{d_u}}\Phi_{\mathbf{d_u}})^{-1}, (P^\top_{\mathbf{d_v}}\Phi_{\mathbf{d_v}})^{-1} \in \mathbb{R}^{l\times l}, \tag{5.10}$$

where in practice, the inverses are computed using the pseudoinverse. Using these inverses from (5.10), we then determine, still in the offline stage,

$$\Phi^D_{\mathbf{d_u}} := \Phi_{\mathbf{d_u}}(P^\top_{\mathbf{d_u}}\Phi_{\mathbf{d_u}})^{-1} \in \mathbb{R}^{n\times l},$$
$$\Phi^D_{\mathbf{d_v}} := \Phi_{\mathbf{d_v}}(P^\top_{\mathbf{d_v}}\Phi_{\mathbf{d_v}})^{-1} \in \mathbb{R}^{n\times l}.$$

Hence, the DEIM approximations of the kinetics are given by

$$\begin{aligned} \mathbf{d_u}(\Psi_{\mathbf{u}}\tilde{\mathbf{u}}_r, \Psi_{\mathbf{v}}\tilde{\mathbf{v}}_r) &\approx \Phi^D_{\mathbf{d_u}}\mathbf{d_u}(P^\top_{\mathbf{d_u}}\Psi_{\mathbf{u}}\tilde{\mathbf{u}}_r, P^\top_{\mathbf{d_u}}\Psi_{\mathbf{v}}\tilde{\mathbf{v}}_r), \\ \mathbf{d_v}(\Psi_{\mathbf{u}}\tilde{\mathbf{u}}_r, \Psi_{\mathbf{v}}\tilde{\mathbf{v}}_r) &\approx \Phi^D_{\mathbf{d_v}}\mathbf{d_v}(P^\top_{\mathbf{d_v}}\Psi_{\mathbf{u}}\tilde{\mathbf{u}}_r, P^\top_{\mathbf{d_v}}\Psi_{\mathbf{v}}\tilde{\mathbf{v}}_r). \end{aligned} \tag{5.11}$$

Note that in (5.11), the kinetics are evaluated component-wise at the vectors of dimension l instead of n. This leads to a second ROM, known as the POD-DEIM-ROM:

$$\begin{aligned} \dot{\tilde{\mathbf{u}}}_r - \mu_u \mathbf{A}_r\tilde{\mathbf{u}}_r + \Psi^\top_{\mathbf{u}}\Phi^D_{\mathbf{d_u}}\mathbf{d_u}(P^\top_{\mathbf{d_u}}\Psi_{\mathbf{u}}\tilde{\mathbf{u}}_r, P^\top_{\mathbf{d_u}}\Psi_{\mathbf{v}}\tilde{\mathbf{v}}_r) &= 0_r \qquad \text{for } t \in (0,T], \\ \dot{\tilde{\mathbf{v}}}_r - \mu_v \mathbf{B}_r\tilde{\mathbf{v}}_r + \Psi^\top_{\mathbf{v}}\Phi^D_{\mathbf{d_v}}\mathbf{d_v}(P^\top_{\mathbf{d_v}}\Psi_{\mathbf{u}}\tilde{\mathbf{u}}_r, P^\top_{\mathbf{d_v}}\Psi_{\mathbf{v}}\tilde{\mathbf{v}}_r) &= 0_r \qquad \text{for } t \in (0,T], \\ \tilde{\mathbf{u}}_r(0) &= \Psi^\top_{\mathbf{u}}\mathbf{u}_0, \\ \tilde{\mathbf{v}}_r(0) &= \Psi^\top_{\mathbf{v}}\mathbf{v}_0. \end{aligned} \tag{5.12}$$

It is worth remarking that $\Psi_{\mathbf{u}}^\top \Phi_{\mathbf{d_u}}^D$ and $\Psi_{\mathbf{v}}^\top \Phi_{\mathbf{d_v}}^D$ can also be determined in the offline stage. Discretizing (5.12) also in time yields

$$\begin{aligned}
(I_r - \Delta t \mu_u \mathbf{A}_r)\,\tilde{\mathbf{u}}_r^k &= \tilde{\mathbf{u}}_r^{k-1} - \Delta t \Psi_{\mathbf{u}}^\top \Phi_{\mathbf{d_u}}^D \mathbf{d}_{\mathbf{u}}(P_{\mathbf{d_u}}^\top \Psi_{\mathbf{u}} \tilde{\mathbf{u}}_r^{k-1}, P_{\mathbf{d_u}}^\top \Psi_{\mathbf{v}} \tilde{\mathbf{v}}_r^{k-1}), \quad \text{for } k = 1, \dots, m,\\
(I_r - \Delta t \mu_v \mathbf{B}_r)\,\tilde{\mathbf{v}}_r^k &= \tilde{\mathbf{v}}_r^{k-1} - \Delta t \Psi_{\mathbf{v}}^\top \Phi_{\mathbf{d_v}}^D \mathbf{d}_{\mathbf{v}}(P_{\mathbf{d_v}}^\top \Psi_{\mathbf{u}} \tilde{\mathbf{u}}_r^{k-1}, P_{\mathbf{d_v}}^\top \Psi_{\mathbf{v}} \tilde{\mathbf{v}}_r^{k-1}), \quad \text{for } k = 1, \dots, m,\\
\tilde{\mathbf{u}}_r^0 &= \Psi_{\mathbf{u}}^\top \mathbf{u}_0,\\
\tilde{\mathbf{v}}_r^0 &= \Psi_{\mathbf{v}}^\top \mathbf{v}_0.
\end{aligned}$$

For both methods, POD and POD-DEIM, we store only every κ-th snapshot. Again, we relabel the stored snapshots consecutively as $\tilde{\mathbf{u}}_r^0, \dots, \tilde{\mathbf{u}}_r^{m_\kappa}$ and denote the resulting snapshot matrices for the variable $\mathbf{u}$ as U_{POD} and $U_{\text{POD-DEIM}}$, respectively.
The left panel of Figure 26 visualizes the relative Frobenius error (4.11) at the final time for the POD and the POD-DEIM method for $r = 1, \dots, 200$:

$$\epsilon_{m_\kappa}(U_{\text{POD}}, r),$$
$$\epsilon_{m_\kappa}(U_{\text{POD-DEIM}}, r).$$

Note that, for the error calculations and the visualization of the reduced-order solutions, the reduced-order solutions must be projected back onto the full space by multiplying them from the left with the corresponding POD basis matrices. We first observe that the POD errors are generally smaller, but still remain relatively high overall. Contrary to the expected monotonic decay of the error with increasing r, the error behavior for both methods is highly irregular and features several jumps of varying orders of magnitude. For POD-DEIM, the situation is even more problematic – for certain values of r, the reduced-order solutions yield only `NaN` values, indicating numerical instabilities. Moreover, a consistent error decay cannot be observed.

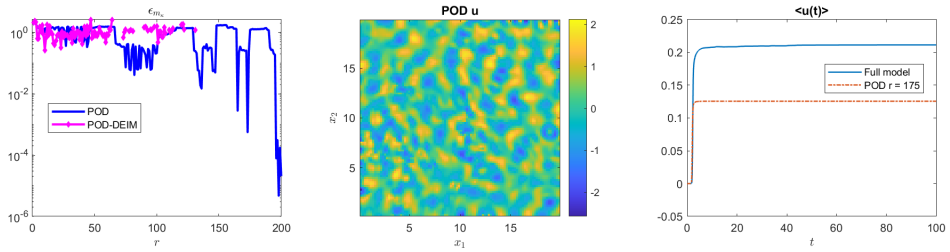


Figure 26: Left: relative Frobenius error (4.11) of u at the final time T, where u is reconstructed using POD and POD-DEIM for $r = 1, \dots, 200$. Middle: u at the final time T, where u is reconstructed using POD with $r = 175$. Right: spatial mean over time of u, where the POD reconstruction is performed with $r = 175$.

The middle and right panels further emphasize the insufficient accuracy of the POD method. In the middle panel, the reduced-order solution obtained by POD with $r = 175$ at final time T projected back onto the full space is depicted. The corresponding relative Frobenius error at the final time is $\epsilon_{m_\kappa}(U_{\text{POD}}, 175) \approx 1.2397$. The right panel compares the spatial mean of this POD solution for $r = 175$ projected back onto the full space with the spatial mean of the reference solution. In both cases, it becomes evident that POD fails to approximate the reference solution (cf. Figure 24). Instead,

the reduced-order solution converges to a different spatial pattern. To address this issue and to ensure a more consistent and monotonic error behavior, we introduce a stabilized version of POD called corrected POD, extend it to the corrected POD-DEIM method and finally improve the efficiency by an adaptive approach.

5.2 Stabilization and Adaptivity

The idea of the PODc and the POD-DEIMc method is to add a correction term to the ROM (5.6) that provides additional sufficient accurate infomration. Therefore, consider a target rank R such that $r < R \ll n$. We choose $R := 200$ to be somewhat below the maximum rank of the snapshot matrices $S_{\mathbf{u}}$ and $S_{\mathbf{v}}$:

$$\max\{\mathrm{rk}(S_{\mathbf{u}}), \mathrm{rk}(S_{\mathbf{v}})\} = \max\{341, 289\}.$$

Let $\tilde{\mathbf{u}}_r$, $\tilde{\mathbf{v}}_r$, $\tilde{\mathbf{u}}_R$ and $\tilde{\mathbf{v}}_R$ denote the time-continuous reduced-order solutions to the ROM (5.6) corresponding to reduced dimensions r and R, respectively. Furthermore, let $\Psi_{\mathbf{u}}^r$, $\Psi_{\mathbf{v}}^r$, $\Psi_{\mathbf{u}}^R$ and $\Psi_{\mathbf{v}}^R$ denote the associated POD basis matrices. Inserting the four POD reduced-order approximations

$$\begin{aligned} \mathbf{u} &\approx \Psi_{\mathbf{u}}^r \tilde{\mathbf{u}}_r, \\ \mathbf{v} &\approx \Psi_{\mathbf{v}}^r \tilde{\mathbf{v}}_r, \\ \mathbf{u} &\approx \Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R, \\ \mathbf{v} &\approx \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R \end{aligned} \tag{5.13}$$

into the FOM (5.1) and projecting onto the reduced subspaces spanned by $\Psi_{\mathbf{u}}^r$ and $\Psi_{\mathbf{v}}^r$, respectively, yields two reduced systems of dimension r. For the variable $\mathbf{u}$, this results in:

$$\begin{aligned} \dot{\tilde{\mathbf{u}}}_r - \mu_u \mathbf{A}_r \tilde{\mathbf{u}}_r + \mathbf{d}_{\mathbf{u}}^r(\tilde{\mathbf{u}}_r, \tilde{\mathbf{v}}_r) - \mathcal{R}(r) &= 0_r, \\ (\Psi_{\mathbf{u}}^r)^\top (\Psi_{\mathbf{u}}^R \dot{\tilde{\mathbf{u}}}_R) - \mu_u (\Psi_{\mathbf{u}}^r)^\top \mathbf{A}(\Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R) + (\Psi_{\mathbf{u}}^r)^\top \mathbf{d}_{\mathbf{u}}(\Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R, \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R) - \mathcal{R}(R) &= 0_r \end{aligned} \tag{5.14}$$

for $t \in (0, T]$, where $\mathcal{R}(\cdot)$ denotes the residual that arises from the POD approximation. These residuals satisfy

$$\begin{aligned} \lim_{r \to \infty} \mathcal{R}(r) &= 0, \\ \lim_{R \to \infty} \mathcal{R}(R) &= 0. \end{aligned}$$

Next, we aim to approximate the term $(\Psi_{\mathbf{u}}^r)^\top (\Psi_{\mathbf{u}}^R \dot{\tilde{\mathbf{u}}}_R)$. For this purpose, we decompose the matrix $\Psi_{\mathbf{u}}^R$ into its first r columns and the remaining $(R - r)$ columns:

$$\Psi_{\mathbf{u}}^R = \begin{bmatrix} \Psi_{\mathbf{u}}^r & (\Psi_{\mathbf{u}}^R)_{:,r+1:R} \end{bmatrix}. \tag{5.15}$$

Due to the orthonormality of the POD basis vectors, we have

$$\begin{aligned} (\Psi_{\mathbf{u}}^r)^\top \Psi_{\mathbf{u}}^r &= I_r, \\ (\Psi_{\mathbf{u}}^r)^\top (\Psi_{\mathbf{u}}^R)_{:,r+1:R} &= 0_{r \times (R-r)}. \end{aligned} \tag{5.16}$$

Using the approximations in (5.13), the orthonormality $(\Psi_{\mathbf{u}}^R)^\top \Psi_{\mathbf{u}}^R = I_R$, the decomposition (5.15) and again the approximation (5.13), we obtain

$$\begin{aligned}(\tilde{\mathbf{u}}_R)_{1:r} &\approx \left((\Psi_{\mathbf{u}}^R)^\top \mathbf{u}\right)_{1:r} \\ &= (\Psi_{\mathbf{u}}^r)^\top \mathbf{u} \\ &\approx \tilde{\mathbf{u}}_r.\end{aligned}$$

Hence, the reduced coefficient vector $\tilde{\mathbf{u}}_R \in \mathbb{R}^R$ can be approximately written as

$$\tilde{\mathbf{u}}_R \approx \begin{bmatrix} \tilde{\mathbf{u}}_r \\ (\tilde{\mathbf{u}}_R)_{r+1:R} \end{bmatrix}.$$

This implies

$$\Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R \approx \Psi_{\mathbf{u}}^r \tilde{\mathbf{u}}_r + (\Psi_{\mathbf{u}}^R)_{:,r+1:R} (\tilde{\mathbf{u}}_R)_{r+1:R}.$$

Taking the time derivative, projecting onto the subspace spanned by $\Psi_{\mathbf{u}}^r$ and using (5.16) yields

$$\begin{aligned}(\Psi_{\mathbf{u}}^r)^\top (\Psi_{\mathbf{u}}^R \dot{\tilde{\mathbf{u}}}_R) &\approx (\Psi_{\mathbf{u}}^r)^\top \left(\Psi_{\mathbf{u}}^r \dot{\tilde{\mathbf{u}}}_r + (\Psi_{\mathbf{u}}^R)_{:,r+1:R} \dot{\tilde{\mathbf{u}}}_{R-r}\right) \\ &= I_r \dot{\tilde{\mathbf{u}}}_r + 0_{r\times(R-r)} \dot{\tilde{\mathbf{u}}}_{R-r} \\ &= \dot{\tilde{\mathbf{u}}}_r.\end{aligned} \tag{5.17}$$

From this point onward, we make use of (5.17) as an identity.

By using (5.17) and subtracting the second equtation of (5.14) from the first one, we obtain

$$\begin{aligned}&-\mu_u (\Psi_{\mathbf{u}}^r)^\top \mathbf{A}\left(\Psi_{\mathbf{u}}^r \tilde{\mathbf{u}}_r - \Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R\right) + (\Psi_{\mathbf{u}}^r)^\top \left(\mathbf{d}_{\mathbf{u}}(\Psi_{\mathbf{u}}^r \tilde{\mathbf{u}}_r, \Psi_{\mathbf{v}}^r \tilde{\mathbf{v}}_r) - \mathbf{d}_{\mathbf{u}}(\Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R, \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R)\right) \\ =&\mathcal{R}(r) - \mathcal{R}(R) \\ =:&\tilde{\mathcal{R}}_{\mathbf{u}}.\end{aligned}$$

Similarly, we can derive the equation for the variable $\mathbf{v}$:

$$\begin{aligned}&-\mu_v (\Psi_{\mathbf{v}}^r)^\top \mathbf{A}\left(\Psi_{\mathbf{v}}^r \tilde{\mathbf{v}}_r - \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R\right) + (\Psi_{\mathbf{v}}^r)^\top \left(\mathbf{d}_{\mathbf{v}}(\Psi_{\mathbf{u}}^r \tilde{\mathbf{u}}_r, \Psi_{\mathbf{v}}^r \tilde{\mathbf{v}}_r) - \mathbf{d}_{\mathbf{v}}(\Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R, \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R)\right) \\ =:&\tilde{\mathcal{R}}_{\mathbf{v}}.\end{aligned}$$

We add this two correction terms $\tilde{\mathcal{R}}_{\mathbf{u}}$ and $\tilde{\mathcal{R}}_{\mathbf{v}}$ to the right-hand side of the ROM (5.6) and obtain the PODc-ROM

$$\begin{aligned}\dot{\tilde{\mathbf{u}}}_r - \mu_u \mathbf{A}_r \tilde{\mathbf{u}}_r + \mathbf{d}_{\mathbf{u}}^r(\tilde{\mathbf{u}}_r, \tilde{\mathbf{v}}_r) &= \tilde{\mathcal{R}}_{\mathbf{u}}, && \text{for } t \in (0,T], \\ \dot{\tilde{\mathbf{v}}}_r - \mu_v \mathbf{B}_r \tilde{\mathbf{v}}_r + \mathbf{d}_{\mathbf{v}}^r(\tilde{\mathbf{u}}_r, \tilde{\mathbf{v}}_r) &= \tilde{\mathcal{R}}_{\mathbf{v}}, && \text{for } t \in (0,T], \\ \tilde{\mathbf{u}}_r(0) &= (\Psi_{\mathbf{u}}^r)^\top \mathbf{u}_0, \\ \tilde{\mathbf{v}}_r(0) &= (\Psi_{\mathbf{v}}^r)^\top \mathbf{v}_0.\end{aligned}$$

This PODc-ROM can be simplified to

$$\begin{aligned}\dot{\tilde{\mathbf{u}}}_r - \mu_u (\Psi_{\mathbf{u}}^r)^\top \mathbf{A} \Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R + (\Psi_{\mathbf{u}}^r)^\top \mathbf{d}_{\mathbf{u}}(\Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R, \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R) &= 0_r, && \text{for } t \in (0,T], \\ \dot{\tilde{\mathbf{v}}}_r - \mu_v (\Psi_{\mathbf{v}}^r)^\top \mathbf{A} \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R + (\Psi_{\mathbf{v}}^r)^\top \mathbf{d}_{\mathbf{v}}(\Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R, \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R) &= 0_r, && \text{for } t \in (0,T], \\ \tilde{\mathbf{u}}_r(0) &= (\Psi_{\mathbf{u}}^r)^\top \mathbf{u}_0, \\ \tilde{\mathbf{v}}_r(0) &= (\Psi_{\mathbf{v}}^r)^\top \mathbf{v}_0,\end{aligned} \tag{5.18}$$

by cancelling out redundant terms. It is worth remarking that (5.18) represents an r-dimensional system that leverages information from an R-dimensional system. The fully discretized form of (5.18) reads:

$$\begin{aligned}
\tilde{\mathbf{u}}_r^k &= \tilde{\mathbf{u}}_r^{k-1} + \Delta t \mu_u (\Psi_{\mathbf{u}}^r)^\top \mathbf{A} \Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R^k \\
&\quad - \Delta t (\Psi_{\mathbf{u}}^r)^\top \mathbf{d}_{\mathbf{u}}(\Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R^{k-1}, \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R^{k-1}), \quad \text{for } k = 1, \dots, m, \\
\tilde{\mathbf{v}}_r^k &= \tilde{\mathbf{v}}_r^{k-1} + \Delta t \mu_v (\Psi_{\mathbf{v}}^r)^\top \mathbf{A} \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R^k \\
&\quad - \Delta t (\Psi_{\mathbf{v}}^r)^\top \mathbf{d}_{\mathbf{v}}(\Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R^{k-1}, \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R^{k-1}), \quad \text{for } k = 1, \dots, m, \\
\tilde{\mathbf{u}}_r^0 &= (\Psi_{\mathbf{u}}^r)^\top \mathbf{u}_0, \\
\tilde{\mathbf{v}}_r^0 &= (\Psi_{\mathbf{v}}^r)^\top \mathbf{v}_0.
\end{aligned}$$

Remark 5.3.

(i) To derive the fully discretized version of the PODc-ROM (5.18), we employ the IMEX Euler method, although the snapshots $\tilde{\mathbf{u}}_R^k$ and $\tilde{\mathbf{v}}_R^k$ are already known for $k = 0, \dots, m$. Moreover, since we are solving for $\tilde{\mathbf{u}}_r^k$ and $\tilde{\mathbf{v}}_r^k$, no linear system needs to be solved.

(ii) The terms

$$\begin{aligned}
&\Delta t \mu_u (\Psi_{\mathbf{u}}^r)^\top \mathbf{A} \Psi_{\mathbf{u}}^R, \\
&\Delta t \mu_v (\Psi_{\mathbf{v}}^r)^\top \mathbf{A} \Psi_{\mathbf{v}}^R
\end{aligned}$$

can be determined in the offline stage.

(iii) The larger the choice of R, the more information is incorporated into the reduced-order model (5.18) and the more effective the resulting stabilization becomes. In principle, one could choose $R = n$, but this is generally not convenient due to memory and computational costs.

In the next step, we extend the introduced stabilization approach to the POD-DEIM method. The corresponding POD-DEIMc-ROM is given by

$$\begin{aligned}
\dot{\tilde{\mathbf{u}}}_r - \mu_u (\Psi_{\mathbf{u}}^r)^\top \mathbf{A} \Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R + (\Psi_{\mathbf{u}}^r)^\top \Phi_{\mathbf{d_u}}^D \mathbf{d}_{\mathbf{u}}(P_{\mathbf{d_u}}^\top \Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R, P_{\mathbf{d_u}}^\top \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R) = 0_r, \\
\dot{\tilde{\mathbf{v}}}_r - \mu_v (\Psi_{\mathbf{v}}^r)^\top \mathbf{A} \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R + (\Psi_{\mathbf{v}}^r)^\top \Phi_{\mathbf{d_v}}^D \mathbf{d}_{\mathbf{v}}(P_{\mathbf{d_v}}^\top \Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R, P_{\mathbf{d_v}}^\top \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R) = 0_r
\end{aligned}$$

for $t \in (0, T]$, with the initial conditions

$$\begin{aligned}
\tilde{\mathbf{u}}_r(0) &= (\Psi_{\mathbf{u}}^r)^\top \mathbf{u}_0, \\
\tilde{\mathbf{v}}_r(0) &= (\Psi_{\mathbf{v}}^r)^\top \mathbf{v}_0,
\end{aligned}$$

where $\Phi_{\mathbf{d_u}}^D$, $\Phi_{\mathbf{d_v}}^D$, $P_{\mathbf{d_u}}$ and $P_{\mathbf{d_v}}$ are exactly the same matrices as in (5.9). Finally, the fully discretized version of the POD-DEIMc-ROM reads

$$\begin{aligned}
\tilde{\mathbf{u}}_r^k &= \tilde{\mathbf{u}}_r^{k-1} + \Delta t \mu_u (\Psi_{\mathbf{u}}^r)^\top \mathbf{A} \Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R^k \\
&\quad - \Delta t (\Psi_{\mathbf{u}}^r)^\top \Phi_{\mathbf{d_u}}^D \mathbf{d}_{\mathbf{u}}(P_{\mathbf{d_u}}^\top \Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R^{k-1}, P_{\mathbf{d_u}}^\top \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R^{k-1}) \quad \text{for } k = 1, \dots, m, \\
\tilde{\mathbf{v}}_r^k &= \tilde{\mathbf{v}}_r^{k-1} + \Delta t \mu_v (\Psi_{\mathbf{v}}^r)^\top \mathbf{A} \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R^k \\
&\quad - \Delta t (\Psi_{\mathbf{v}}^r)^\top \Phi_{\mathbf{d_v}}^D \mathbf{d}_{\mathbf{v}}(P_{\mathbf{d_v}}^\top \Psi_{\mathbf{u}}^R \tilde{\mathbf{u}}_R^{k-1}, P_{\mathbf{d_v}}^\top \Psi_{\mathbf{v}}^R \tilde{\mathbf{v}}_R^{k-1}) \quad \text{for } k = 1, \dots, m, \\
\tilde{\mathbf{u}}_r^0 &= (\Psi_{\mathbf{u}}^r)^\top \mathbf{u}_0, \\
\tilde{\mathbf{v}}_r^0 &= (\Psi_{\mathbf{v}}^r)^\top \mathbf{v}_0.
\end{aligned} \tag{5.19}$$

In Figure 27 and 28, we visualize in the left panel the final-time Turing pattern obtained using the PODc and the POD-DEIMc method, respectively. The right panels show the spatial mean over time of the reduced-order solution projected back onto the full space as well as the spatial mean over time of the full-order solution. For both methods, the final-time Turing pattern seems to match the full-order solution well (cf. Figure 24). The spatial mean over time of the reduced-order solution projected back onto the full space is also in good agreement with the full-order solution for the PODc method. However, for the POD-DEIMc method, the spatial mean over time in the stability zone is slightly below that of the reference solution.

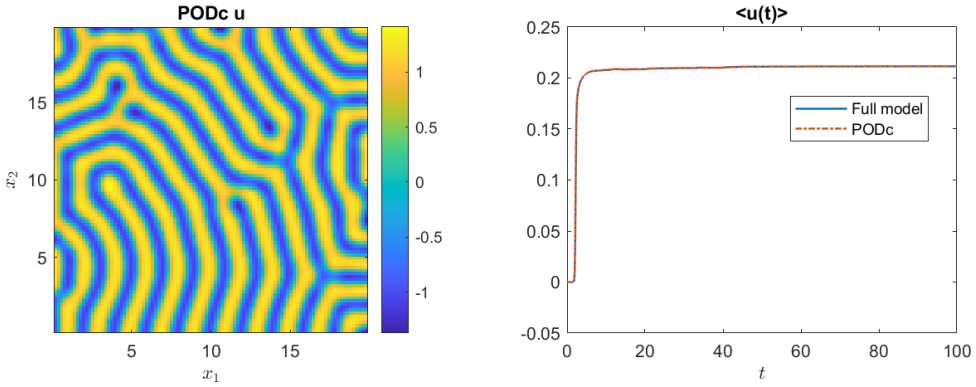


Figure 27: Left: reduced-order solution obtained by PODc with $r = 175$ at final time T projected back onto the full space. Right: comparison of the spatial mean over time of the PODc solution for $r = 175$ projected back onto the full space with the spatial mean over time of the reference solution.

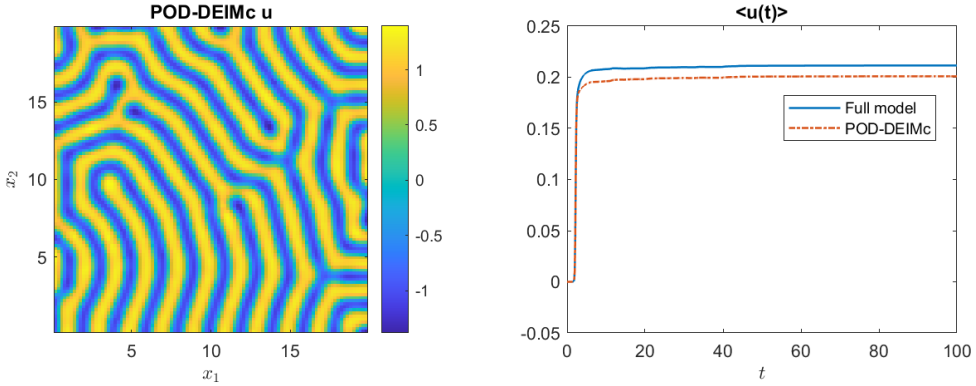


Figure 28: Left: reduced-order solution obtained by POD-DEIMc with $r = 175$, $R = 200$ and $l = 362$ at final time T projected back onto the full space. Right: comparison of the spatial mean over time of the POD-DEIMc solution for $r = 175$, $R = 200$ and $l = 362$ projected back onto the full space with the spatial mean over time of the reference solution.

Moreover, in accordance with Figure 26 (left), the relative Frobenius error (4.11) at the final time for the POD, PODc, POD-DEIM and POD-DEIMc method for $r = 1, \ldots, 200$ is shown in Figure 29. We observe a monotonic error decay for the PODc method: the error decreases rapidly at the beginning, but starting from $r \gtrapprox 100$,

the decay stagnates on the order of magnitude around 10^{-5}. Similarly, the POD-DEIMc method exhibits improved behavior: the error decreases monotonically at the beginning, reaching a minimum of $1.931 \cdot 10^{-2}$ at $r = 10$. After that, the error increases slightly again. However, there is no longer any evidence of completely irregular error behavior. So there has clearly been a stabilization. In a final step, we present an adaptive approach with which we can further reduce the error of both methods and improve the computational efficiency.

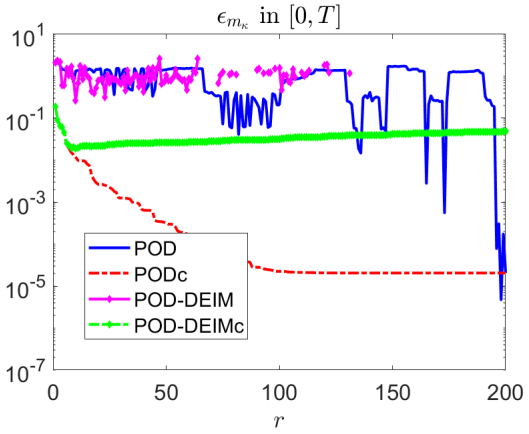


Figure 29: Relative Frobenius error (4.11) of u at the final time T, where u is reconstructed using the POD, PODc, POD-DEIM and POD-DEIMc method for $r = 1, \ldots, 200$.

The adaptive approach is based on the idea of splitting the time interval $[0, T]$ into two distinct temporal regimes using $\bar{t}$ (cf. (5.3)): the reactivity zone $\mathcal{I}_1$ and the stability zone $\mathcal{I}_2$. The advantage of treating the time regimes separately lies in the expectation that the number of basis vectors R required for the correction to capture the dynamics within each individual regime is significantly smaller than the number of basis vectors needed to accurately represent the dynamics over the entire time interval. In other words we can adapt the number of POD basis vectors R in line with the qualitative behavior of the dynamics. Hence, we get an speed-up for both, the offline and the online stage. In particular, the computation of the POD solutions $\tilde{\mathbf{u}}_R$ and $\tilde{\mathbf{v}}_R$ becomes more efficient for both time intervals, as well as for the SVD computations, due to the reduced size of the snapshot matrices.

Let us therefore split the snapshot matrices $S_{\mathbf{u}}$ and $S_{\mathbf{v}}$ into two parts. The first part contains the snapshots of the reactivity zone, taken at every second time step:

$$\begin{aligned} S_{\mathbf{u}}^{(1)} &:= \begin{bmatrix} \mathbf{u}^0 & \mathbf{u}^2 & \cdots & \mathbf{u}^{m_{\bar{t}}} \end{bmatrix}, \\ S_{\mathbf{v}}^{(1)} &:= \begin{bmatrix} \mathbf{v}^0 & \mathbf{v}^2 & \cdots & \mathbf{v}^{m_{\bar{t}}} \end{bmatrix}, \end{aligned} \tag{5.20}$$

where $m_{\bar{t}} := \bar{t}/\Delta t$ is time step index corresponding to $\bar{t}$. We choose every second time step in the reactivity zone because it is a short time interval and this choice provides a sufficient number of snapshots to accurately capture the dynamics. The second part contains the snapshots of the stability zone, taken every fourth time step:

$$\begin{aligned} S_{\mathbf{u}}^{(2)} &:= \begin{bmatrix} \mathbf{u}^{m_{\bar{t}}} & \mathbf{u}^{m_{\bar{t}}+4} & \cdots & \mathbf{u}^{m} \end{bmatrix}, \\ S_{\mathbf{v}}^{(2)} &:= \begin{bmatrix} \mathbf{v}^{m_{\bar{t}}} & \mathbf{v}^{m_{\bar{t}}+4} & \cdots & \mathbf{v}^{m} \end{bmatrix}. \end{aligned} \tag{5.21}$$

It is worth noting that the starting index of the matrices $S_{\mathbf{u}}^{(2)}$ and $S_{\mathbf{v}}^{(2)}$ coincides with the final index of $S_{\mathbf{u}}^{(1)}$ and $S_{\mathbf{v}}^{(1)}$ (a justification is provided later). In the same way, we split snapshot matrices of the kinetics $S_{\mathbf{d_u}}$ and $S_{\mathbf{d_v}}$ into two parts:

$$\begin{aligned} S_{\mathbf{d_u}}^{(1)} &:= \left[\begin{array}{cccc} \mathbf{d_u}(\mathbf{u}^0, \mathbf{v}^0) & \mathbf{d_u}(\mathbf{u}^2, \mathbf{v}^2) & \cdots & \mathbf{d_u}(\mathbf{u}^{m_{\bar{t}}}, \mathbf{v}^{m_{\bar{t}}}) \end{array} \right], \\ S_{\mathbf{d_v}}^{(1)} &:= \left[\begin{array}{cccc} \mathbf{d_v}(\mathbf{u}^0, \mathbf{v}^0) & \mathbf{d_v}(\mathbf{u}^2, \mathbf{v}^2) & \cdots & \mathbf{d_v}(\mathbf{u}^{m_{\bar{t}}}, \mathbf{v}^{m_{\bar{t}}}) \end{array} \right] \end{aligned} \tag{5.22}$$

and

$$\begin{aligned} S_{\mathbf{d_u}}^{(2)} &:= \left[\begin{array}{cccc} \mathbf{d_u}(\mathbf{u}^{m_{\bar{t}}}, \mathbf{v}^{m_{\bar{t}}}) & \mathbf{d_u}(\mathbf{u}^{m_{\bar{t}}+4}, \mathbf{v}^{m_{\bar{t}}+4}) & \cdots & \mathbf{d_u}(\mathbf{u}^{m}, \mathbf{v}^{m}) \end{array} \right], \\ S_{\mathbf{d_v}}^{(2)} &:= \left[\begin{array}{cccc} \mathbf{d_v}(\mathbf{u}^{m_{\bar{t}}}, \mathbf{v}^{m_{\bar{t}}}) & \mathbf{d_v}(\mathbf{u}^{m_{\bar{t}}+4}, \mathbf{v}^{m_{\bar{t}}+4}) & \cdots & \mathbf{d_v}(\mathbf{u}^{m}, \mathbf{v}^{m}) \end{array} \right]. \end{aligned} \tag{5.23}$$

We define the POD target rank values

$$\begin{aligned} R_1 &:= 41, \\ R_2 &:= 150 \end{aligned}$$

for the first and second time interval, respectively, slightly below the maximum rank of the snapshot matrices $S_{\mathbf{u}}^{(i)}$ and $S_{\mathbf{v}}^{(i)}$:

$$\begin{aligned} \max\{\mathrm{rk}(S_{\mathbf{u}}^{(1)}), \mathrm{rk}(S_{\mathbf{v}}^{(1)})\} &= \max\{56, 44\}, \\ \max\{\mathrm{rk}(S_{\mathbf{u}}^{(2)}), \mathrm{rk}(S_{\mathbf{v}}^{(2)})\} &= \max\{305, 260\}. \end{aligned}$$

It is worth emphasizing that $R_1, R_2 \ll 200$ holds. The corresponding rank-R_i POD bases are then denoted as $\Psi_{\mathbf{u}}^{(i),R_i}, \Psi_{\mathbf{v}}^{(i),R_i}$ for $i = 1, 2$. Using these bases, we compute the reduced-order solutions

$$\begin{aligned} &\tilde{\mathbf{u}}_{R_1}^{(1),0}, \ldots, \tilde{\mathbf{u}}_{R_1}^{(1),m_{\bar{t}}}, \\ &\tilde{\mathbf{v}}_{R_1}^{(1),0}, \ldots, \tilde{\mathbf{v}}_{R_1}^{(1),m_{\bar{t}}} \end{aligned}$$

by solving (5.8) with $r = R_1$ for $k = 0, \ldots, m_{\bar{t}}$. Similarly, for the second time interval, the solutions

$$\begin{aligned} &\tilde{\mathbf{u}}_{R_2}^{(2),m_{\bar{t}}}, \ldots, \tilde{\mathbf{u}}_{R_2}^{(2),m}, \\ &\tilde{\mathbf{v}}_{R_2}^{(2),m_{\bar{t}}}, \ldots, \tilde{\mathbf{v}}_{R_2}^{(2),m} \end{aligned}$$

are obtained by solving (5.8) with $r = R_2$ for $k = m_{\bar{t}}, \ldots, m$. To ensure continuity between the two intervals, the initial conditions for the second time interval are defined by projecting the final reduced-order solutions from the first interval onto the corresponding second-interval bases:

$$\begin{aligned} \tilde{\mathbf{u}}_{R_2}^{(2),m_{\bar{t}}} &= (\Psi_{\mathbf{u}}^{(2),R_2})^\top \Psi_{\mathbf{u}}^{(1),R_1} \tilde{\mathbf{u}}_{R_1}^{(1),m_{\bar{t}}}, \\ \tilde{\mathbf{v}}_{R_2}^{(2),m_{\bar{t}}} &= (\Psi_{\mathbf{v}}^{(2),R_2})^\top \Psi_{\mathbf{v}}^{(1),R_1} \tilde{\mathbf{v}}_{R_1}^{(1),m_{\bar{t}}}. \end{aligned}$$

This two-stage projection is essential: the solutions at $m_{\bar{t}}$ must first be lifted to the full-order space and then projected onto the reduced basis of the second interval. Next, we define the DEIM target ranks as

$$l_1 := 60,$$

$$l_2 := 324,$$

based on the ranks

$$\max\{\mathrm{rk}(S^{(1)}_{\mathbf{d_u}}), \mathrm{rk}(S^{(1)}_{\mathbf{d_v}})\} = \max\{62, 59\},$$
$$\max\{\mathrm{rk}(S^{(2)}_{\mathbf{d_u}}), \mathrm{rk}(S^{(2)}_{\mathbf{d_v}})\} = \max\{323, 308\}.$$

The associated rank-l_i DEIM bases and interpolation points are denoted by $\Phi^{(i)}_{\mathbf{d_u}}$, $\Phi^{(i)}_{\mathbf{d_v}}$, $P^{(i)}_{\mathbf{d_u}}$ and $P^{(i)}_{\mathbf{d_v}}$ for $i = 1, 2$. With these components, we then compute the reduced-order solutions

$$\tilde{\mathbf{u}}^{(1),0}_{r_1}, \ldots, \tilde{\mathbf{u}}^{(1),m_{\bar{t}}}_{r_1},$$
$$\tilde{\mathbf{v}}^{(1),0}_{r_1}, \ldots, \tilde{\mathbf{v}}^{(1),m_{\bar{t}}}_{r_1}$$

by solving (5.19) with $r = r_1$ and $R = R_1$ for $k = 0, \ldots, m_{\bar{t}}$. For the second time interval, the corresponding solutions

$$\tilde{\mathbf{u}}^{(2),m_{\bar{t}}}_{r_2}, \ldots, \tilde{\mathbf{u}}^{(2),m}_{r_2},$$
$$\tilde{\mathbf{v}}^{(2),m_{\bar{t}}}_{r_2}, \ldots, \tilde{\mathbf{v}}^{(2),m}_{r_2}$$

are obtained by solving the same system (5.19) with $r = r_2$ and $R = R_2$ for $k = m_{\bar{t}}, \ldots, m$. The initial conditions for this stage are again computed via projection:

$$\tilde{\mathbf{u}}^{(2),m_{\bar{t}}}_{r_2} = \left(\Psi^{(2),r_2}_{\mathbf{u}}\right)^\top \Psi^{(1),r_1}_{\mathbf{u}} \tilde{\mathbf{u}}^{(1),m_{\bar{t}}}_{r_1},$$
$$\tilde{\mathbf{v}}^{(2),m_{\bar{t}}}_{r_2} = \left(\Psi^{(2),r_2}_{\mathbf{v}}\right)^\top \Psi^{(1),r_1}_{\mathbf{v}} \tilde{\mathbf{v}}^{(1),m_{\bar{t}}}_{r_1}.$$

The complete adaptive POD-DEIMc algorithm is summarized in Algorithm 5.

Algorithm 5 Adaptive POD-DEIMc

INPUT κ, m, Δt, $\mathbf{u}_0$, $\mathbf{v}_0$, $\mathbf{A}$, $\mathbf{d_u}$, $\mathbf{d_v}$, μ_u, μ_v
Compute the snapshots $\mathbf{u}^0, \dots, \mathbf{u}^m$ and $\mathbf{v}^0, \dots, \mathbf{v}^m$
Store every κ-th snapshot and determine $\operatorname{argmax}_{k \in \{0,\dots,m_\kappa - 1\}} \operatorname{inc}(u)_k$
Compute $\bar{t}$ according to (5.3)
for $i = 1, 2$ **do**
  if $i = 1$ **then**
    Define the matrices $S^{(1)}_{\mathbf{u}}$ and $S^{(1)}_{\mathbf{v}}$ according to (5.20)
    Define the matrices $S^{(1)}_{\mathbf{d_u}}$ and $S^{(1)}_{\mathbf{d_v}}$ according to (5.22)
  else
    Define the matrices $S^{(2)}_{\mathbf{u}}$ and $S^{(2)}_{\mathbf{v}}$ according to (5.21)
    Define the matrices $S^{(2)}_{\mathbf{d_u}}$ and $S^{(2)}_{\mathbf{d_v}}$ according to (5.23)
  end if
  Fix the POD target rank $R_i \approx \max\{\operatorname{rk}(S^{(i)}_{\mathbf{u}}), \operatorname{rk}(S^{(i)}_{\mathbf{v}})\}$
  Compute the rank-R_i POD bases $\Psi^{(i),R_i}_{\mathbf{u}}$ and $\Psi^{(i),R_i}_{\mathbf{v}}$ from $S^{(i)}_{\mathbf{u}}$ and $S^{(i)}_{\mathbf{v}}$
  if $i = 1$ **then**
    Let $\tilde{\mathbf{u}}^{(1),0}_{R_1} = (\Psi^{(1),R_1}_{\mathbf{u}})^\top \mathbf{u}_0$, $\tilde{\mathbf{v}}^{(1),0}_{R_1} = (\Psi^{(1),R_1}_{\mathbf{v}})^\top \mathbf{v}_0$
    Compute $\tilde{\mathbf{u}}^{(1),k}_{R_1}$, $\tilde{\mathbf{v}}^{(1),k}_{R_1}$ for $k = 1, \dots, m_{\bar{t}}$ by solving (5.8) for $r = R_1$
  else
    Let $\tilde{\mathbf{u}}^{(2),m_{\bar{t}}}_{R_2} = (\Psi^{(2),R_2}_{\mathbf{u}})^\top \Psi^{(1),R_1}_{\mathbf{u}} \tilde{\mathbf{u}}^{(1),m_{\bar{t}}}_{R_1}$, $\tilde{\mathbf{v}}^{(2),m_{\bar{t}}}_{R_2} = (\Psi^{(2),R_2}_{\mathbf{v}})^\top \Psi^{(1),R_1}_{\mathbf{v}} \tilde{\mathbf{v}}^{(1),m_{\bar{t}}}_{R_1}$
    Compute $\tilde{\mathbf{u}}^{(2),k}_{R_2}$, $\tilde{\mathbf{v}}^{(2),k}_{R_2}$ for $k = m_{\bar{t}} + 1, \dots, m$ by solving (5.8) for $r = R_2$
  end if
  Fix the DEIM target rank $l_i \approx \max\{\operatorname{rk}(S^{(i)}_{\mathbf{d_u}}), \operatorname{rk}(S^{(i)}_{\mathbf{d_v}})\}$
  Compute the rank-l_i DEIM bases $\Phi^{(i)}_{\mathbf{d_u}}$, $\Phi^{(i)}_{\mathbf{d_v}}$ from $S^{(i)}_{\mathbf{d_u}}$ and $S^{(i)}_{\mathbf{d_v}}$
  Compute the DEIM points $P^{(i)}_{\mathbf{d_u}}$, $P^{(i)}_{\mathbf{d_v}}$ from $\Phi^{(i)}_{\mathbf{d_u}}$ and $\Phi^{(i)}_{\mathbf{d_v}}$
  Fix $r_i \leq R_i$
  if $i = 1$ **then**
    Let $\tilde{\mathbf{u}}^{(1),0}_{r_1} = (\Psi^{(1),r_1}_{\mathbf{u}})^\top \mathbf{u}_0$, $\tilde{\mathbf{v}}^{(1),0}_{r_1} = (\Psi^{(1),r_1}_{\mathbf{v}})^\top \mathbf{v}_0$
    Compute $\tilde{\mathbf{u}}^{(1),k}_{r_1}$, $\tilde{\mathbf{v}}^{(1),k}_{r_1}$ for $k = 1, \dots, m_{\bar{t}}$ by solving (5.19) for $r = r_1$ and $R = R_1$
  else
    Let $\tilde{\mathbf{u}}^{(2),m_{\bar{t}}}_{r_2} = (\Psi^{(2),r_2}_{\mathbf{u}})^\top \Psi^{(1),r_1}_{\mathbf{u}} \tilde{\mathbf{u}}^{(1),m_{\bar{t}}}_{r_1}$, $\tilde{\mathbf{v}}^{(2),m_{\bar{t}}}_{r_2} = (\Psi^{(2),r_2}_{\mathbf{v}})^\top \Psi^{(1),r_1}_{\mathbf{v}} \tilde{\mathbf{v}}^{(1),m_{\bar{t}}}_{r_1}$
    Compute $\tilde{\mathbf{u}}^{(2),k}_{r_2}$, $\tilde{\mathbf{v}}^{(2),k}_{r_2}$ for $k = m_{\bar{t}} + 1, \dots, m$ by solving (5.19) for $r = r_2$ and $R = R_2$
  end if
end for
RETURN $\tilde{\mathbf{u}}^{(1),0}_{r_1}, \dots, \tilde{\mathbf{u}}^{(1),m_{\bar{t}}}_{r_1}, \tilde{\mathbf{v}}^{(1),0}_{r_1}, \dots, \tilde{\mathbf{v}}^{(1),m_{\bar{t}}}_{r_1}, \tilde{\mathbf{u}}^{(2),m_{\bar{t}}}_{r_2}, \dots, \tilde{\mathbf{u}}^{(2),m}_{r_2}$ and $\tilde{\mathbf{v}}^{(2),m_{\bar{t}}}_{r_2}, \dots, \tilde{\mathbf{v}}^{(2),m}_{r_2}$

Remark 5.4.

(i) The adaptive POD, adaptive POD-DEIM and adaptive PODc method is derived from Algorithm 5 by omitting the corresponding DEIM or correction steps.

(ii) In our implementation of the adaptive methods, we consistently use $\Psi^{(1),10}_{\mathbf{u}} \tilde{\mathbf{u}}^{(1),m_{\bar{t}}}_{10}$ to compute the initial values, instead of $\Psi^{(1),R_1}_{\mathbf{u}} \tilde{\mathbf{u}}^{(1),m_{\bar{t}}}_{R_1}$ or $\Psi^{(1),r_1}_{\mathbf{u}} \tilde{\mathbf{u}}^{(1),m_{\bar{t}}}_{r_1}$, regardless of the specific values of R_1 and r_1. We follow the same approach for the

variable $\mathbf{v}$. This choice ensures that the initial conditions for the second time interval remain sufficiently close to the full-order solution, thereby preserving the accuracy of the simulation.

In Figure 30, the results of the adaptive methods are visualized. The final-time relative Frobenius error (4.11) is shown for the reactivity zone $\mathcal{I}_1$ ($r = 1, \ldots, 41$) in the left panel and for the stability zone $\mathcal{I}_2$ ($r = 1, \ldots, 150$) in the right panel. In both zones, the POD and the POD-DEIM error behavior is completely unstable, showing no monotonic error decay. The stabilized methods, however, present a different situation. Both the PODc and the POD-DEIMc method in the reactivity zone $\mathcal{I}_1$ demonstrate an initial monotonic error decay. While the error then slightly increases for $r \geq 5$, it nonetheless stabilizes on the order of magnitude around 10^{-3}. Clearer differences between the two methods emerge in the stability zone $\mathcal{I}_2$. While the POD-DEIMc method initially exhibits a monotonic error decay before stagnating around 10^{-3} for $r \gtrapprox 30$, the PODc method shows a consistently monotonic error decay, reaching an error on the order of 10^{-6} at $r = 97$. Hence, the adaptive methods show a significant improvement in error behavior (cf. Figure 29).

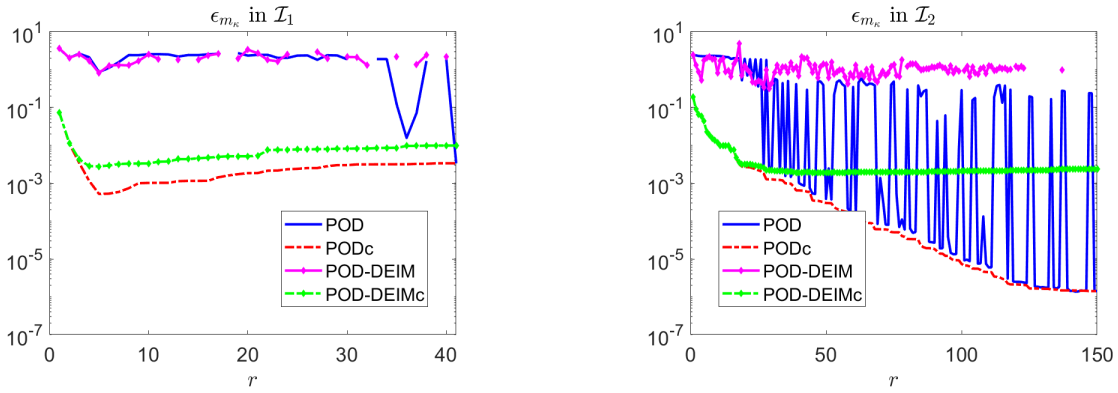


Figure 30: Relative Frobenius error (4.11) of u at the final time T of the reactivity zone $\mathcal{I}_1$ (left, $r = 1, \ldots, 41$) and the stability zone $\mathcal{I}_2$ (right, $r = 1, \ldots, 150$), where u is reconstructed using the POD, POD-DEIM, PODc and POD-DEIMc method.

Table 4 emphasizes that the adaptive approach significantly speeds up the offline phase, primarily due to the faster calculation of the POD solutions $\tilde{\mathbf{u}}_R$ and $\tilde{\mathbf{v}}_R$. Specifically, the cost for the entire time interval $[0, T]$ is 1729.2 seconds, compared to 11.8 seconds for the reactivity zone $\mathcal{I}_1$ and 1237.7 seconds for the stability zone $\mathcal{I}_2$.

Table 4: Offline phase running times for the POD computation of $\tilde{\mathbf{u}}_R$ and $\tilde{\mathbf{v}}_R$ over the entire time interval $[0, T]$, the reactivity zone $\mathcal{I}_1$, and the stability zone $\mathcal{I}_2$.

	R	Running time $[s]$
$[0, T]$	200	1729.2
$\mathcal{I}_1$	41	11.8
$\mathcal{I}_2$	150	1237.7

6 Parameter Identification

This section is based on the work of [5], [13] and [24].

We propose a parameter identification approach within an optimization framework, where the goal is to minimize an appropriate *cost functional* subject to a semilinear parabolic PDE and a box constraint on the parameter.
After establishing the mathematical foundation, including assumptions ensuring the existence and uniqueness of solutions, the *projected Barzilai-Borwein (PBB) method* is introduced as an efficient technique for solving the resulting constrained optimization problems. To determine the derivative of the cost functional, we employ a *sensitivity approach*. As a concrete example, we consider a one-dimensional reaction-diffusion equation as a representative example of a semilinear parabolic PDE constraint.
In contrast to the sections before, the *finite element method (FEM)* is used for the spatial discretization.

Let $\Omega \subset \mathbb{R}^{d_\Omega}$, with $d_\Omega \in \{1, 2\}$, be a non-empty, bounded domain with Lipschitz-continuous boundary $\Gamma := \partial\Omega$. Define $V := H^1(\Omega)$, $H := L^2(\Omega)$ and

$$H^{d_\Omega} := \underbrace{H \times \cdots \times H}_{d_\Omega \text{ times}}.$$

Note that H^{d_Ω} is a Hilbert space equipped with the inner product

$$\langle u, v \rangle_{H^{d_\Omega}} := \sum_{i=1}^{d_\Omega} \langle u_i, v_i \rangle_H, \tag{6.1}$$

where $u := \begin{bmatrix} u_1 & \cdots & u_{d_\Omega} \end{bmatrix}^T$, $v := \begin{bmatrix} v_1 & \cdots & v_{d_\Omega} \end{bmatrix}^T \in H^{d_\Omega}$. For $T > 0$, define

$$\Omega_T := (0, T] \times \Omega$$

and

$$\Gamma_T := [0, T] \times \Gamma.$$

Moreover, the compact and convex set of admissible parameters is given as

$$\begin{aligned} \mathcal{P}_{\mathsf{ad}} &:= [\mu^-, \mu^+] \\ &:= \{\mu \in \mathcal{P} \,|\, \mu^- \leq \mu \leq \mu^+\} \end{aligned}$$

with $\mathcal{P} := \mathbb{R}_{>0}$ and $\mu^-, \mu^+ \in \mathcal{P}$. The objective is to minimize the cost functional

$$J \colon W(0, T; V, V') \times \mathcal{P} \to \mathbb{R},$$

where $J(u, \mu)$ is defined as

$$J(u, \mu) := \frac{1}{2} \int_0^T \int_\Omega |u - u_d|^2 \, dx \, dt + \frac{\sigma}{2} |\mu|^2, \tag{6.2}$$

subject to $\mu \in \mathcal{P}_{\mathsf{ad}}$ and such that the real-valued function $u \colon \Omega_T \to \mathbb{R}$ is the weak solution to the *state equation*

$$u_t(t, x) - \mu \Delta u((t, x)) + d(t, x, u(t, x)) = 0 \quad \text{f.a.a. } (t, x) \in \Omega_T. \tag{6.3a}$$

The given function $u_d \in L^2(\Omega_T)$ represents the *desired state* that we aim to achieve. The term $\sigma/2|\mu|^2$ is referred to as the *regularization term*, with $\sigma \geq 0$ denoting the corresponding *regularization parameter*. We emphasize that, in particular, setting $\sigma = 0$ is permissible, since the set of admissible parameters $\mathcal{P}_{\mathsf{ad}}$ is compact in $\mathbb{R}$. Moreover, $d\colon \Omega_T \times \mathbb{R} \to \mathbb{R}$ is a given nonlinear function.

Remark 6.1. We use the notation $d(u)$ instead of $d(\cdot, \cdot, u(\cdot, \cdot))$, and $d(t, u(t))$ instead of $d(t, \cdot, u(t, \cdot))$, if the context is clear.

We suppose homogeneous Neumann boundary conditions

$$\nabla u \cdot n = 0 \quad \text{on } \Gamma_T. \tag{6.3b}$$

Finally, u satisfies

$$u(0) = u_0 \quad \text{in } \Omega \tag{6.3c}$$

for a given initial function $u_0 \in H$. The constraint of the optimization problem is thus formulated by the semilinear parabolic IBVP defined in (6.3).

Definition 6.1 (cf. [24, Chapter 5])**.** A function $u \in L^2(0, T; V) \cap L^\infty(\Omega_T)$ is called a *weak solution* to the IBVP (6.3) if it satisfies the variational formulation

$$-\int_0^T \int_\Omega u\varphi_t \, dx \, dt + \int_0^T \int_\Omega \mu \nabla u \cdot \nabla \varphi + d(u)\varphi \, dx \, dt = \int_\Omega u_0 \varphi(0) \, dx \tag{6.4}$$

for all $\varphi \in W(0, T; V, H)$ with $\varphi(T) = 0$ in Ω.

Integration by parts of the first term in the variational formulation yields

$$\begin{aligned}\int_0^T \int_\Omega u\varphi_t \, dx \, dt &= \int_\Omega [u\varphi]_0^T \, dx - \int_0^T \langle u_t(t), \varphi(t) \rangle_{V',V} \, dt \\ &= \int_\Omega u(T)\varphi(T) \, dx - \int_\Omega u(0)\varphi(0) \, dx - \int_0^T \langle u_t(t), \varphi(t) \rangle_{V',V} \, dt,\end{aligned}$$

where u_t stands for the distributional derivative of u with respect to t. Since $\varphi(T) = 0$ and $u(0) = u_0$ in Ω, we obtain

$$\int_0^T \int_\Omega u\varphi_t \, dx \, dt = -\int_\Omega u_0 \varphi(0) \, dx - \int_0^T \langle u_t(t), \varphi(t) \rangle_{V',V} \, dt. \tag{6.5}$$

By inserting (6.5) into (6.4) and subtracting $\int_\Omega u_0 \varphi(0) \, dx$ from both sides, we have

$$\int_0^T \langle u_t(t), \varphi(t) \rangle_{V',V} \, dt + \int_0^T \int_\Omega \mu \nabla u \cdot \nabla \varphi + d(u)\varphi \, dx \, dt = 0. \tag{6.6}$$

This leads to a second, more convenient definition of a weak solution.

Definition 6.2. A function $u \in W(0, T; V, V') \cap L^\infty(\Omega_T)$ is called a *weak solution* to the IBVP (6.3) if it satisfies the variational formulation

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\langle u(t), \varphi \rangle_H + \mu \langle \nabla u(t), \nabla \varphi \rangle_{H^{d_\Omega}} + \langle d(t, u(t)), \varphi \rangle_H &= 0 \\ \langle u(0), \varphi \rangle_H &= \langle u_0, \varphi \rangle_H\end{aligned} \tag{6.7}$$

for all $\varphi \in V$ and almost all $t \in (0, T]$.

The concept of a weak solution as defined in Definition 6.2 is more regular than that in Definition 6.1, meaning that any solution satisfying Definition 6.2 also meets the criteria of Definition 6.1. From this point on, we restrict our attention to weak solutions as defined in Definition 6.2 and simply refer to them as weak solutions to the IBVP (6.3).

To derive an existence and uniqueness statement for a weak solution, we need to make some assumptions.

Assumption 6.1. The function $d\colon \Omega_T \times \mathbb{R} \to \mathbb{R}$ satisfies the following properties:

- the mapping $(t,x) \mapsto d(t,x,u)$ is measurable f.a.a. $(t,x) \in \Omega_T$ and any fixed $u \in \mathbb{R}$.
- the function $u \mapsto d(t,x,u)$ is monotonically increasing f.a.a. $(t,x) \in \Omega_T$.
- there is a constant $K > 0$ such that $|d(t,x,0)| \leq K$ holds f.a.a. $(t,x) \in \Omega_T$.
- the function $u \mapsto d(t,x,u)$ is locally Lipschitz-continuous f.a.a. $(t,x) \in \Omega_T$, i.e., for any $M > 0$ there is a constant $L(M) > 0$ such that all $u_1, u_2 \in \mathbb{R}$ with $|u_1|, |u_2| \leq M$ satisfy

$$|d(t,x,u_1) - d(t,x,u_2)| \leq L(M)|u_1 - u_2| \quad \text{f.a.a. } (t,x) \in \Omega_T.$$

This assumption enables us to formulate the next statement.

Theorem 6.1 (cf. [24, Chapter 5])**.** Let Assumption 6.1 hold. Then the semilinear parabolic IBVP (6.3) has a unique weak solution $u \in W(0,T;V,V') \cap L^\infty(\Omega_T)$ for any $\mu \in \mathcal{P}_{\mathsf{ad}}$ and any $f \in L^r(\Omega_T)$, $u_0 \in C(\overline{\Omega})$ with $r > {}^d\!/_2 + 1$. The solution is continuous on $\overline{\Omega}_T$ and there exists a constant $c_\infty > 0$, indenpendent of d and u_0, such that

$$\|u\|_{W(0,T;V,V')} + \|u\|_{C(\overline{\Omega}_T)} \leq c_\infty \left(\|d(0)\|_{L^r(\Omega_T)} + \|u_0\|_{C(\overline{\Omega})} \right).$$

In the following, we suppose that Assumption 6.1 holds. Thus, for each $\mu \in \mathcal{P}_{\mathsf{ad}}$ the IBVP (6.3) has a unique weak solution $u[\mu] \in W(0,T;V,V')$ by Theorem 6.1. This allows for the definition of the *solution operator*

$$S\colon \mathcal{P}_{\mathsf{ad}} \to W(0,T;V,V'), \quad S(\mu) := u[\mu].$$

Using this operator, the *reduced cost functional* is defined as

$$\hat{J}\colon \mathcal{P}_{\mathsf{ad}} \to \mathbb{R}, \quad \hat{J}(\mu) := J(S(\mu), \mu),$$

which leads to the *reduced problem*

$$\min_{\mu \in \mathcal{P}_{\mathsf{ad}}} \hat{J}(\mu). \tag{6.8}$$

In (6.8) the *state variable* u is eliminated.

Throughout the following, the reduced cost functional $\hat{J}$ is assumed to be continuously differentiable.

6.1 Projected Barzilai-Borwein Method

Gradient descent methods have long been used to address smooth unconstrained nonlinear optimization problems. The core concept is as follows: consider a continuously differentiable function $\hat{J}\colon \mathbb{R}^n \to \mathbb{R}$ with $n \in \mathbb{N}$. The goal is to find a local minimizer $\bar{\mu} \in \mathbb{R}^n$ of $\hat{J}$. Starting from an initial guess $\mu^{-1} \in \mathbb{R}^n$, gradient descent methods iteratively update the current iterate μ^k by moving in the direction of the negative gradient of $\hat{J}$ at μ^k

$$\mu^{k+1} := \mu^k + \alpha_k d^k \quad \text{for } k = -1, 0, 1, 2, \ldots,$$

where $d^k := -\nabla \hat{J}(\mu^k)$ and $\alpha_k > 0$ represents the step size. Although the iteration index k conventionally starts at 0, we begin at $k = -1$ in this case to allow for the use of two starting points in the following method. Furthermore, we remark that we use the standard notation d for the descent direction, which also appeared earlier to denote the nonlinearity in (6.3). In each case, the meaning should be clear from the context. The classic *steepest descent (SD) method* sets

$$\alpha_k := \arg\min_{\alpha > 0} \hat{J}(\mu^k + \alpha d^k). \tag{6.9}$$

Note that (6.9) requires solving an optimization problem at each iteration. In practice, this can lead to high computational costs if $\hat{J}$ or $\nabla \hat{J}$ are expensive to evaluate. Furthermore, the rate of convergence of the SD method is only linear and it is severely affected by problems that are badly conditioned. To address this issue, Barzilai and Borwein (see [4]) introduced in 1988 a line search method that selects the step size based on previously calculated evaluations of the gradient and the current iterates to improve the convergence rate of the algorithm:

$$\alpha_k^{\mathrm{BB1}} := \frac{s_{k-1}^\top d_{k-1}}{d_{k-1}^\top d_{k-1}} \quad \text{for } k = 0, 2, 4, \ldots,$$
$$\alpha_k^{\mathrm{BB2}} := \frac{s_{k-1}^\top s_{k-1}}{s_{k-1}^\top d_{k-1}} \quad \text{for } k = 1, 3, 5, \ldots,$$

where

$$\begin{aligned} s_{k-1} &:= \mu^k - \mu^{k-1}, \\ d_{k-1} &:= \nabla \hat{J}(\mu^k) - \nabla \hat{J}(\mu^{k-1}), \\ \mu^0 &:= \nabla \hat{J}(\mu^{-1}). \end{aligned}$$

It is worth noting that, in the one-dimensional case, the step size reduces to

$$\alpha_k^{\mathrm{BB}} = \frac{s_{k-1}}{d_{k-1}} = \frac{\mu^k - \mu^{k-1}}{\nabla \hat{J}(\mu^k) - \nabla \hat{J}(\mu^{k-1})}$$

and the update formula becomes

$$\mu^{k+1} = \mu^k - \alpha_k^{\mathrm{BB}} d^k = \mu^k - \frac{\mu^k - \mu^{k-1}}{\nabla \hat{J}(\mu^k) - \nabla \hat{J}(\mu^{k-1})} \nabla \hat{J}(\mu^k),$$

which is the *secant* and therefore a *quasi-Newton method* with locally superlinear convergence properties. The advantage of the secant method or quasi-Newton methods over the Newton method is that the hessian matrix does not need to be computed.

Remark 6.2.

(i) According to the Cauchy-Schwarz inequality, we have

$$(s_{k-1}^\top d_{k-1})^2 \leq (s_{k-1}^\top s_{k-1})(d_{k-1}^\top d_{k-1}).$$

Assuming $s_{k-1}^\top d_{k-1} > 0$, we can divide both sides by $(s_{k-1}^\top d_{k-1})(d_{k-1}^\top d_{k-1})$ to obtain

$$\alpha_k^{\mathrm{BB1}} = \frac{s_{k-1}^\top d_{k-1}}{d_{k-1}^\top d_{k-1}} \leq \frac{s_{k-1}^\top s_{k-1}}{s_{k-1}^\top d_{k-1}} = \alpha_k^{\mathrm{BB2}}.$$

Thus, the step size α_k^{BB1} is referred to as the *short BB step size* and α_k^{BB2} as the *long BB step size* (see [14]).

(ii) The short BB step size α_k^{BB1} minimizes

$$\min_{\alpha>0} \|s_{k-1} - \alpha d_{k-1}\|_2^2,$$

while the long BB step size α_k^{BB2} minimizes

$$\min_{\alpha>0} \|\alpha s_{k-1} - d_{k-1}\|_2^2.$$

For more details, we refer to [4].

(iii) The BB method is only well-defined, if the denominator of α_k^{BB} is not zero for all k.

(iv) The BB method has been shown to exhibit superlinear convergence when applied to the minimization of strongly convex quadratic functions in two-dimensional settings. In the more general case of n-dimensional problems, the method exhibits global convergence with a linear rate. Although the BB method does not guarantee a strictly decreasing objective function value at iteration, numerous computational experiments have demonstrated that it significantly outperforms the SD method (cf. [14]).

Next, we extend the BB method to the case of the box-constrained optimization problems (6.8). The starting points μ^{-1} and μ^0 are chosen such that they satisfy the box constraints

$$\begin{aligned} \mu^{-1} &\in \mathcal{P}_{\mathsf{ad}}, \\ \mu^0 := \mathcal{P}(\nabla \hat{J}(\mu^{-1})) &\in \mathcal{P}_{\mathsf{ad}}, \end{aligned}$$

where $\mathcal{P}\colon \mathbb{R} \to \mathcal{P}_{\mathsf{ad}}$ denotes the *projection operator* onto the feasible set, defined by

$$\mathcal{P}(\mu) := \max\left\{\mu^-, \min\{\mu^+, \mu\}\right\}.$$

The iterative process follows the rule

$$\mu^{k+1} = \mathcal{P}(\mu^k - \alpha_k \nabla \hat{J}(\mu^k)), \quad k = 0, 1, 2, \ldots,$$

where the derivative of the reduced cost function $\hat{J}$ at the point μ^k is denoted by $\nabla\hat{J}(\mu^k)$ and can be computed by the sensitivity approach introduced in the following Section 6.2. Finally, the iteration process terminates when a maximum number of iterations $k_{\max} := 200$ is reached or when

$$\left|\mu^k - \mathcal{P}(\mu^k - \alpha_{k-1}\nabla\hat{J}(\mu^k))\right| < \varepsilon$$

is satisfied for a given tolerance $\varepsilon := 10^{-8} > 0$. Algorithm 6 provides an overview of the PBB method.

Algorithm 6 Projected Barzilai-Borwein (PBB) gradient method

INPUT $\mu^{-1} \in \mathcal{P}_{\mathsf{ad}}$, $\varepsilon > 0$, $k_{\max} \in \mathbb{N}$, $\hat{J}$, $\nabla\hat{J}$
Compute $\hat{J}(\mu^{-1})$, $\nabla\hat{J}(\mu^{-1})$
Set $k = 0$
Compute $\mu^k = \mathcal{P}(\nabla\hat{J}(\mu^{k-1}))$, $\hat{J}(\mu^k)$, $\nabla\hat{J}(\mu^k)$
Compute residual $r^k = |\mu^k - \mathcal{P}(\mu^k - \nabla\hat{J}(\mu^k))|$
while $k < k_{\max}$ and $r^k > \varepsilon$ **do**
    Set $s_{k-1} = \mu^k - \mu^{k-1}$
    Set $d_{k-1} = \nabla\hat{J}(\mu^k) - \nabla\hat{J}(\mu^{k-1})$
    if k is even **then**
        Set $\alpha_k = \frac{s_{k-1}^\top d_{k-1}}{d_{k-1}^\top d_{k-1}}$
    else
        Set $\alpha_k = \frac{s_{k-1}^\top s_{k-1}}{s_{k-1}^\top d_{k-1}}$
    end if
    Update $\mu^{k+1} = \mathcal{P}(\mu^k - \alpha_k\nabla\hat{J}(\mu^k))$
    Compute $\hat{J}(\mu^{k+1})$, $\nabla\hat{J}(\mu^{k+1})$
    Compute residual $r^{k+1} = |\mu^{k+1} - \mathcal{P}(\mu^{k+1} - \alpha_k\nabla\hat{J}(\mu^{k+1}))|$
    Update $k = k + 1$
end while
RETURN $\mu^k, \hat{J}(\mu^k) \in \mathbb{R}$, $k \in \mathbb{N}$

The PBB method benefits from the efficiency of the BB step sizes while ensuring constraint satisfaction through projection. This approach has been successfully applied in PDE-constrained optimization (see [3]) and machine learning problems, where it demonstrates improved convergence properties compared to classical gradient methods.

6.2 Sensitivity Approach

To apply the sensitivity approach, the derivative of the reduced cost functional $\hat{J}$ needs to be computed in $\mu \in \mathcal{P}_{\mathsf{ad}}$. For that purpose, we assume that the solution operator S is continuously Fréchet differentiable and that the mapping $u \mapsto d(u)$ is continuously differentiable as well. Under these assumptions, we evaluate the directional derivative (or *sensitivity*) of $\hat{J}(\mu)$ in any direction $h \in \mathbb{R}$ such that $\mu + \tau h \in \mathcal{P}_{\mathsf{ad}}$ for all $\tau \in (0, \tau_h]$, for some $\tau_h > 0$.

Remark 6.3.

(i) Since S is continuously Fréchet differentiable and $u \mapsto d(u)$ is continuously differentiable, we employ the first-order Taylor expansions

$$\begin{aligned} S(\mu+\tau h) &= S(\mu) + S'(\mu)\tau h + \mathcal{o}(\tau) && \text{for } \tau \to 0, \\ |\mu+\tau h|^2 &= |\mu|^2 + 2\mu\tau h + \mathcal{o}(\tau) && \text{for } \tau \to 0, \\ d(S(\mu+\tau h)) &= d(S(\mu)) + d_u(S(\mu))S'(\mu)\tau h + \mathcal{o}(\tau) && \text{for } \tau \to 0, \end{aligned}$$

where the first identity is understood in Ω_T. Moreover, we denote the partial derivative of d with respect to u by d_u (cf. Remark 6.1) and emphasize that $S'(\mu)$ and $d_u(S(\mu))$ are linear mappings.

(ii) Since $S(\mu) \in W(0,T;V,V')$ for $\mu \in \mathcal{P}_{\mathsf{ad}}$, the weak time derivative $S(\mu)_t$ lies in $L^2(0,T;V')$ and the weak (spatial) derivative $\nabla S(\mu)$ belongs to $L^2(0,T;H^{d_\Omega})$.

Our first goal is to derive a way to compute the sensitivity $S'(\mu)h$. Let

$$\hat{e}\colon \mathcal{P}_{\mathsf{ad}} \to L^2(0,T;V') \times H$$

be the function defined by

$$\begin{aligned} &\hat{e}(\mu) \\ :=& e(S(\mu),\mu) \\ :=& \begin{bmatrix} e_1(S(\mu),\mu) \\ e_2(S(\mu),\mu) \end{bmatrix} \\ :=& \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t}\langle S(\mu)(\cdot),\cdot\rangle_H + \mu\langle \nabla S(\mu)(\cdot), \nabla\cdot\rangle_{H^{d_\Omega}} + \langle d(\cdot, S(\mu)(\cdot)),\cdot\rangle_H \\ \langle S(\mu)(0) - u_0, \cdot\rangle_H \end{bmatrix}. \end{aligned}$$

The variational formulation (6.2) can then be expressed as

$$0 = \hat{e}(\mu) \quad \text{in } L^2(0,T;V') \times H.$$

Differentiating in any direction $h \in \mathbb{R}$ yields

$$\begin{aligned} 0 &= \hat{e}'(\mu)h \\ &= \lim_{\tau \to 0} \frac{1}{\tau}\left(\hat{e}(\mu+\tau h) - \hat{e}(\mu)\right) \end{aligned} \tag{6.10}$$

in $L^2(0,T;V') \times H$. Considering (6.10) component-wise, we formally obtain for the

first component

$$
\begin{aligned}
0 = \lim_{\tau\to 0}\frac{1}{\tau}\Bigg(&\frac{\mathrm{d}}{\mathrm{d}t}\langle S(\mu+\tau h)(t),\varphi\rangle_H + (\mu+\tau h)\langle\nabla S(\mu+\tau h)(t),\nabla\varphi\rangle_{H^{d_\Omega}}\\
&+\langle d(t,S(\mu+\tau h)(t)),\varphi\rangle_H\\
&-\frac{\mathrm{d}}{\mathrm{d}t}\langle S(\mu)(t),\varphi\rangle_H - \mu\langle\nabla S(\mu)(t),\nabla\varphi\rangle_{H^{d_\Omega}} - \langle d(t,S(\mu)(t)),\varphi\rangle_H\Bigg)\\
= \lim_{\tau\to 0}\frac{1}{\tau}\Bigg(&\frac{\mathrm{d}}{\mathrm{d}t}\langle (S(\mu+\tau h)-S(\mu))(t),\varphi\rangle_H + \mu\langle\nabla\left(S(\mu+\tau h)-S(\mu)\right)(t),\nabla\varphi\rangle_{H^{d_\Omega}}\\
&+\tau h\langle\nabla S(\mu+\tau h)(t),\nabla\varphi\rangle_{H^{d_\Omega}}\\
&+\langle d(t,S(\mu+\tau h)(t)) - d(t,S(\mu)(t)),\varphi\rangle_H\Bigg)\\
= \lim_{\tau\to 0}\frac{1}{\tau}\Bigg(&\frac{\mathrm{d}}{\mathrm{d}t}\langle (S'(\mu)\tau h + o(\tau)(t))(t),\varphi\rangle_H + \mu\langle\nabla\left(S'(\mu)\tau h + o(\tau)\right)(t),\nabla\varphi\rangle_{H^{d_\Omega}}\\
&+\tau h\langle\nabla\left(S(\mu)+S'(\mu)\tau h + o(\tau)\right)(t),\nabla\varphi\rangle_{H^{d_\Omega}}\\
&+\langle d_u(S(\mu))(t,(S'(\mu)\tau h)(t)) + o(\tau)(t),\varphi\rangle_H\Bigg)\\
= \lim_{\tau\to 0}\frac{1}{\tau}\Bigg(&\tau\frac{\mathrm{d}}{\mathrm{d}t}\left\langle\left(S'(\mu)h+\frac{o(\tau)}{\tau}\right)(t),\varphi\right\rangle_H + \mu\tau\left\langle\nabla\left(S'(\mu)h+\frac{o(\tau)}{\tau}\right)(t),\nabla\varphi\right\rangle_{H^{d_\Omega}}\\
&+\tau h\langle\nabla S(\mu)(t),\nabla\varphi\rangle_{H^{d_\Omega}} + \tau^2 h\left\langle\nabla\left(S'(\mu)h+\frac{o(\tau)}{\tau}\right)(t),\nabla\varphi\right\rangle_{H^{d_\Omega}}\\
&+\tau\langle d_u(S(\mu))(t,(S'(\mu)h)(t)) + \frac{o(\tau)(t)}{\tau},\varphi\rangle_H\Bigg)\\
= &\frac{\mathrm{d}}{\mathrm{d}t}\langle (S'(\mu)h)(t),\varphi\rangle_H + \mu\langle\nabla\left(S'(\mu)h\right)(t),\nabla\varphi\rangle_{H^{d_\Omega}} + h\langle\nabla S(\mu)(t),\nabla\varphi\rangle_{H^{d_\Omega}}\\
&+\langle d_u(S(\mu))(t,(S'(\mu)h)(t),\varphi\rangle_H,
\end{aligned}
$$

for almost all $t\in(0,T]$ and for all $\varphi\in V$. This equation is called the (weak) *linearized state equation*. The second component yields

$$
\begin{aligned}
0 &= \lim_{\tau\to 0}\frac{1}{\tau}\Big(\langle S(\mu+\tau h)(0)-u_0,\varphi\rangle_H - \langle S(\mu)(0)-u_0,\varphi\rangle_H\Big)\\
&= \lim_{\tau\to 0}\frac{1}{\tau}\Big(\langle (S(\mu+\tau h)-S(\mu))(0),\varphi\rangle_H\Big)\\
&= \lim_{\tau\to 0}\frac{1}{\tau}\Big(\langle (S'(\mu)\tau h + o(\tau))(0),\varphi\rangle_H\Big)\\
&= \lim_{\tau\to 0}\frac{1}{\tau}\left(\tau\left\langle\left(S'(\mu)h+\frac{o(\tau)}{\tau}\right)(0),\varphi\right\rangle_H\right)\\
&= \langle (S'(\mu)h)(0),\varphi\rangle_H
\end{aligned}
\tag{6.11}
$$

for all $\varphi \in H$. Note that it is equivalent to require (6.11) for all $\varphi \in V$, since the embedding $V \hookrightarrow H$ is continuous and dense. Therefore, every element of H can be approximated by a sequence in V.
Equation (6.10) yields the weak formulation of an IBVP that allows us to compute the sensitivity $S'(\mu)h$.

Based on this sensitivity $S'(\mu)h$, the sensitivity $\nabla \hat{J}(\mu)h$ can be determined. It satisfies

$$
\begin{aligned}
\nabla \hat{J}(\mu)h &= \lim_{\tau \to 0} \frac{1}{\tau} \left(\hat{J}(\mu + \tau h) - \hat{J}(\mu) \right) \\
&= \lim_{\tau \to 0} \frac{1}{2\tau} \left(\int_0^T \int_\Omega |S(\mu + \tau h) - u_d|^2 \, dx \, dt + \sigma |\mu + \tau h|^2 \right. \\
&\qquad \left. - \int_0^T \int_\Omega |S(\mu) - u_d|^2 \, dx \, dt - \sigma |\mu|^2 \right) \\
&= \lim_{\tau \to 0} \frac{1}{2\tau} \left(\int_0^T \int_\Omega |S(\mu) + S'(\mu)\tau h + \mathcal{O}(\tau) - u_d|^2 - |S(\mu) - u_d|^2 \, dx \, dt \right) \\
&\quad + \lim_{\tau \to 0} \frac{\sigma}{2\tau} \left(|\mu|^2 + 2\mu\tau h + \mathcal{O}(\tau) - |\mu|^2 \right) \\
&= \lim_{\tau \to 0} \frac{1}{2\tau} \left(\int_0^T \int_\Omega |S(\mu) - u_d|^2 + 2(S(\mu) - u_d)(S'(\mu)\tau h + \mathcal{O}(\tau)) \right. \\
&\qquad \left. + |S'(\mu)\tau h + \mathcal{O}(\tau)|^2 - |S(\mu) - u_d|^2 \, dx \, dt \right) \\
&\quad + \lim_{\tau \to 0} \frac{\sigma}{2\tau} \left(2\mu\tau h + \mathcal{O}(\tau) \right) \\
&= \lim_{\tau \to 0} \frac{1}{2\tau} \left(\int_0^T \int_\Omega 2(S(\mu) - u_d)(S'(\mu)\tau h + \mathcal{O}(\tau)) + |S'(\mu)\tau h + \mathcal{O}(\tau)|^2 \, dx \, dt \right) \\
&\quad + \sigma \mu h \\
&= \lim_{\tau \to 0} \frac{1}{2\tau} \left(\int_0^T \int_\Omega 2\tau (S(\mu) - u_d) \left(S'(\mu)h + \frac{\mathcal{O}(\tau)}{\tau} \right) \right. \\
&\qquad \left. + \tau^2 |S'(\mu)h + \frac{\mathcal{O}(\tau)}{\tau}|^2 \, dx \, dt \right) + \sigma \mu h \\
&= \int_0^T \int_\Omega (S(\mu) - u_d) S'(\mu)h \, dx \, dt + \sigma \mu h.
\end{aligned}
$$

Hence, to determine the sensitivity $\nabla \hat{J}(\mu)h$ via the sensitivity approach, the following steps are required:

1. Compute $u_\mu^h[\mu] := S'(\mu)h$ by solving the linearized state equation

$$\begin{aligned} 0 &= \frac{\mathrm{d}}{\mathrm{d}t}\langle u_\mu^h[\mu](t), \varphi\rangle_H + \mu\langle \nabla u_\mu^h[\mu](t), \nabla\varphi\rangle_{H^{d_\Omega}} + h\langle \nabla S(\mu)(t), \nabla\varphi\rangle_{H^{d_\Omega}} \\ &\quad + \langle d_u(S(\mu))(t, u_\mu^h[\mu](t), \varphi\rangle_H, \\ 0 &= \langle u_\mu^h[\mu](0), \varphi\rangle_H \end{aligned} \tag{6.12}$$

   for all $\varphi \in V$ and almost all $t \in (0, T]$.

2. Determine the sensitivity $\nabla\hat{J}(\mu)h$ by using the sensitivity $u_\mu^h[\mu]$

$$\nabla\hat{J}(\mu)h = \int_0^T \int_\Omega (S(\mu) - u_d)u_\mu^h[\mu]\,dx\,dt + \sigma\mu h. \tag{6.13}$$

It is important to highlight that the goal is the computation of the derivative $\nabla\hat{J}(\mu)$, which can be readily obtained for $h = 1$ in the above expressions. However, in the case $\mu = \mu^+$, choosing $h = 1$ is not admissible, as this would result in $\mu + \tau h \notin \mathcal{P}_{\mathsf{ad}}$ for all $\tau > 0$. Instead, the derivative is computed in the direction $h = -1$, yielding the negative derivative. The desired value $\nabla\hat{J}(\mu)$ is then obtained by multiplying this result by -1.
With this derivative of $\hat{J}$ at μ at our disposal, we can fully implement the PBB method to address our optimization problem. The procedure can be summarized as follows:

1. Initialize Algorithm 6 with an initial guess $\mu^{-1} \in \mathcal{P}_{\mathsf{ad}}$.
2. Determine the weak solution of the state equation (6.3) in order to obtain $S(\mu^{-1})$.
3. Compute $\hat{J}(\mu^{-1})$ and $\nabla\hat{J}(\mu^{-1})$ using the sensitivity approach given by (6.12) and (6.13). Set $k = 0$.
4. Since $k = 0$, update the iterate as

$$\mu^k = \mathcal{P}(\nabla\hat{J}(\mu^{k-1}))$$

   and determine the weak solution of the state equation (6.3) in order to obtain $S(\mu^k)$.
5. Compute $\hat{J}(\mu^k)$ and $\nabla\hat{J}(\mu^k)$ following (6.12) and (6.13).
6. Compute the residual

$$r^k = \left|\mu^k - \mathcal{P}(\mu^k - \nabla\hat{J}(\mu^k))\right|.$$

7. If the stopping criterion $r^k \leq \varepsilon$ is satisfied or a maximum number of iterations $k_{\max}$ is reached, terminate the algorithm and return the iterate μ^k, the cost $\hat{J}(\mu^k)$ and the iteration counter k. Otherwise, compute the BB step size α_k and update the iterate as

$$\mu^{k+1} = \mathcal{P}(\mu^k - \alpha_k\nabla\hat{J}(\mu^k)).$$

8. Determine the weak solution of the state equation (6.3) in order to obtain $S(\mu^{k+1})$.

9. Compute $\hat{J}(\mu^{k+1})$ and $\nabla\hat{J}(\mu^{k+1})$ following (6.12) and (6.13).

10. Compute the residual

$$r^{k+1} = \left|\mu^{k+1} - \mathcal{P}(\mu^{k+1} - \alpha_k \nabla\hat{J}(\mu^{k+1}))\right|,$$

update $k = k + 1$ and return to Step 7.

6.3 Example: Semilinear Reaction-Diffusion Equation

We consider the following IBVP

$$\begin{aligned} u_t - \mu\Delta u + d(u) &= 0 \quad \text{in } \Omega_T, \\ (\nabla u)n &= 0 \quad \text{on } \Gamma_T, \\ u(0) &= u_0 \quad \text{in } \Omega. \end{aligned} \tag{6.14}$$

The domain is given by $\Omega := (0, 1)$, i.e., $d_\Omega := 1$, and the final time is set to $T := 1$. Moreover, the nonlinear function d is defined as

$$d\colon \mathbb{R} \to \mathbb{R}, \quad d(u) := u^3.$$

Its derivative is given by

$$d_u\colon \mathbb{R} \to \mathbb{R}, \quad d_u(u) := 3u^2.$$

To remain consistent with the notation in Section 6.2, we denote the derivative as d_u rather than d', despite d being a function of u only. Furthermore, the initial condition is

$$u_0(x) := 0.1(x - 0.5)^2 \quad \text{f.a.a. } x \in \Omega.$$

In contrast to the previous sections, we employ the FEM instead of the FDM for the spatial discretization of (6.14). To this end, we triangulate the domain Ω into a *finite element (FE)* mesh with $n_x := 1024$ nodes. Let $\{\varphi_i\}_{i=1}^{n_x} \subset V$ be a set of piecewise linear FE basis functions and define

$$V^{\mathrm{FE}} := \operatorname{span}\{\varphi_1, \ldots, \varphi_{n_x}\} \subset V.$$

More precisely, each basis function $\varphi_i \in V^{\mathrm{FE}}$ is defined by

$$\varphi_i(x_j) := \delta_{ij} \quad \text{for } 1 \le i, j \le n_x,$$

where x_j denotes the j-th node of the mesh and $\delta_{i,j}$ is the *Kronecker delta* given by

$$\delta_{i,j} := \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

In terms of the FE basis functions, the (spatially) discrete weak solution u of (6.7) can be expressed as

$$u^{\mathrm{FE}}(t) := \sum_{i=1}^{n_x} \mathfrak{u}_i(t)\varphi_i \in V^{\mathrm{FE}} \quad \text{f.a.a. } t \in [0, T],$$

with time-dependent coefficients

$$\mathfrak{u}_i \colon [0,T] \to \mathbb{R}, \quad 1 \le i \le n_x.$$

Hence, the FE Galerkin scheme takes the form

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\langle u^{\mathrm{FE}}(t), \varphi\rangle_H + \mu\langle \nabla u^{\mathrm{FE}}(t), \nabla\varphi\rangle_{H^{d_\Omega}} + \langle d(u^{\mathrm{FE}}(t)), \varphi\rangle_H = 0, \\ \langle u^{\mathrm{FE}}(0), \varphi\rangle_H = \langle u_0, \varphi\rangle_H \end{aligned}$$

for all $\varphi \in V^{\mathrm{FE}}$ and almost all $t \in (0,T]$.

Finally, the full discretization is obtained by applying the IMEX Euler method. To this end, the time interval $[0,T]$ is partitioned using an equidistant temporal grid

$$t_k := k\Delta t \quad \text{for } k = 0, \dots, m$$

with $m := {}^{T}/_{\Delta t}$. In particular, we choose $\Delta t := 10^{-3}$. Using the notation

$$u^k := \sum_{i=1}^{n_x} \mathfrak{u}_i^k \varphi_i := u^{\mathrm{FE}}(t_k) \quad \text{for } k = 0, \dots, m,$$

the fully discrete FE scheme takes the form

$$\begin{aligned} \left\langle \frac{u^k - u^{k-1}}{\Delta t}, \varphi \right\rangle_H + \mu\langle \nabla u^k, \nabla\varphi\rangle_{H^{d_\Omega}} + \langle d(u^k), \varphi\rangle_H = 0, \\ \langle u^0, \varphi\rangle_H = \langle u_0, \varphi\rangle_H \end{aligned} \tag{6.15}$$

for all $\varphi \in V^{\mathrm{FE}}$ and $k = 1, \dots, m$.

Let the mass matrix $\mathrm{M} \in \mathbb{R}^{n_x \times n_x}$ be defined by

$$\mathrm{M}_{ij} := \langle \varphi_i, \varphi_j \rangle_H \tag{6.16}$$

and the stiffness matrix $\mathrm{S} \in \mathbb{R}^{n_x \times n_x}$ (not to be confused with the solution operator S) by

$$\mathrm{S}_{ij} := \langle \nabla\varphi_i, \nabla\varphi_j \rangle_{H^{d_\Omega}}. \tag{6.17}$$

Furthermore, define the vectors corresponding to the initial condition and the nonlinear term by

$$\mathrm{u}_0 := \begin{bmatrix} \langle u_0, \varphi_1\rangle_H \\ \vdots \\ \langle u_0, \varphi_{n_x}\rangle_H \end{bmatrix} \in \mathbb{R}^{n_x},$$

$$\mathrm{d}^k := \begin{bmatrix} \langle d(u^k), \varphi_1\rangle_H \\ \vdots \\ \langle d(u^k), \varphi_{n_x}\rangle_H \end{bmatrix} \in \mathbb{R}^{n_x}.$$

Testing (6.15) with the n_x basis functions φ_i for $i = 1, \dots, n_x$ yields the linear system

$$\begin{aligned} (\mathrm{M} + \Delta t\mu\mathrm{S})\,\mathfrak{u}^k = \mathrm{M}\mathfrak{u}^{k-1} - \Delta t\mathrm{d}^{k-1} \quad \text{for } k = 1, \dots, m, \\ \mathrm{M}\mathfrak{u}^0 = \mathrm{u}_0, \end{aligned} \tag{6.18}$$

where for each $k = 0, \dots, m$, the coefficient vector $\mathfrak{u}^k$ is given by

$$\mathfrak{u}^k := \begin{bmatrix} \mathfrak{u}_1^k & \cdots & \mathfrak{u}_{n_x}^k \end{bmatrix}^\top.$$

We solve the linear system (6.18) for the diffusion parameter $\mu^* = 5.0 \cdot 10^{-2}$ and take the resulting solution as the FE approximation of the desired state, i.e., $u_d := S(\mu^*)$. This yields the approximation

$$u_d(t_k) \approx u^k \quad \text{for } k = 0, \dots, m.$$

Next, we define the compact and convex parameter set

$$\begin{aligned} \mathcal{P}_{\mathsf{ad}} &:= [\mu^-, \mu^+] \\ &:= [10^{-3}, 10^{-1}] \subset \mathbb{R}_{>0}. \end{aligned}$$

The objective is to minimize the reduced cost functional

$$\hat{J}(\mu) := \frac{1}{2} \int_0^T \int_\Omega |S(\mu) - u_d|^2 \, dx \, dt + \frac{\sigma}{2} |\mu|^2$$

subject to $\mu \in \mathcal{P}_{\mathsf{ad}}$ and the constraint that $S(\mu)$ satisfies (6.14). Given $\mu \in \mathcal{P}_{\mathsf{ad}}$, the corresponding state $S(\mu)$ is approximated at time t_k by

$$S(\mu)(t_k) \approx S(\mu)^k := \sum_{i=1}^{n_x} \mathfrak{c}_i^k \varphi_i := S(\mu)^{\mathrm{FE}}(t_k) := \sum_{i=1}^{n_x} \mathfrak{c}_i(t_k) \varphi_i$$

for some computed coefficients $\mathfrak{c}_i^k$, with $i = 1, \dots, n_x$ and $k = 0, \dots, m$. Furthermore, define the coefficient vector at time step k as

$$\mathfrak{c}^k := \begin{bmatrix} \mathfrak{c}_1^k & \cdots & \mathfrak{c}_{n_x}^k \end{bmatrix}^\top \quad \text{for } k = 0, \dots, m.$$

The linear system for the sensitivity $u_\mu^h[\mu]$, which is required to compute the derivative $\nabla \hat{J}(\mu)$ via the sensitivity approach, can be derived analogously to (6.18) from (6.12). We approximate the sensitivity at each time step t_k by

$$u_\mu^h[\mu](t_k) \approx u_\mu^h[\mu]^k := \sum_{i=1}^{n_x} \mathfrak{e}_i^k \varphi_i := u_\mu^h[\mu]^{\mathrm{FE}}(t_k) := \sum_{i=1}^{n_x} \mathfrak{e}_i(t_k) \varphi_i$$

for $k = 0, \dots, m$. This yields the fully discrete sensitivity system

$$\begin{aligned} (\mathrm{M} + \Delta t \mu \mathrm{S}) \, \mathfrak{e}^k &= \mathrm{M} \mathfrak{e}^{k-1} - \Delta t \mathrm{r}^{k-1} - \Delta t h \mathrm{S} \mathfrak{c}^k \quad \text{for } k = 1, \dots, m, \\ \mathrm{M} \mathfrak{e}^0 &= 0_{n_x}, \end{aligned}$$

where $\mathfrak{e}^k := \begin{bmatrix} \mathfrak{e}_1^k & \cdots & \mathfrak{e}_{n_x}^k \end{bmatrix}^\top$ for $k = 0, \dots, m$ and

$$\mathrm{r}^k := \begin{bmatrix} \langle d_u(S(\mu)^{k+1}) u_\mu^1[\mu]^k, \varphi_1 \rangle_H \\ \vdots \\ \langle d_u(S(\mu)^{k+1}) u_\mu^1[\mu]^k, \varphi_{n_x} \rangle_H \end{bmatrix} \in \mathbb{R}^{n_x}$$

for $k = 0, \dots, m-1$.

In addition, to apply the PBB method, both $\hat{J}(\mu)$ and its derivative $\nabla \hat{J}(\mu)$ need to be evaluated for any $\mu \in \mathcal{P}_{\mathsf{ad}}$. Therefore, it is necessary to determine the integrals

$$\int_0^T \int_\Omega |S(\mu) - u_d|^2 \, dx \, dt,$$
$$\int_0^T \int_\Omega (S(\mu) - u_d) u_\mu^h[\mu] \, dx \, dt.$$

Using the previously introduced notation, we approximate the first integral at a fixed time point by

$$\begin{aligned}
\int_\Omega |S(\mu) - u_d|^2 \, dx &\approx \int_\Omega |S(\mu)^{\mathrm{FE}} - {u_d}^{\mathrm{FE}}|^2 \, dx \\
&= \int_\Omega \left(\sum_{i=1}^{n_x} \mathfrak{c}_i \varphi_i - \sum_{j=1}^{n_x} \mathfrak{u}_j \varphi_j \right) \left(\sum_{k=1}^{n_x} \mathfrak{c}_k \varphi_k - \sum_{l=1}^{n_x} \mathfrak{u}_l \varphi_l \right) dx \\
&= \int_\Omega \left(\sum_{i=1}^{n_x} (\mathfrak{c}_i - \mathfrak{u}_i) \varphi_i \right) \left(\sum_{j=1}^{n_x} (\mathfrak{c}_j - \mathfrak{u}_j) \varphi_j \right) dx \\
&= \sum_{i=1}^{n_x} \sum_{j=1}^{n_x} (\mathfrak{c}_i - \mathfrak{u}_i)(\mathfrak{c}_j - \mathfrak{u}_j) \int_\Omega \varphi_i \varphi_j \, dx \\
&= \sum_{i=1}^{n_x} (\mathfrak{c}_i - \mathfrak{u}_i) \sum_{j=1}^{n_x} (\mathfrak{c}_j - \mathfrak{u}_j) \mathrm{M}_{ij} \\
&= \sum_{i=1}^{n_x} (\mathfrak{c}_i - \mathfrak{u}_i) (\mathrm{M}(\mathfrak{c} - \mathfrak{u}))_{i,:} \\
&= (\mathfrak{c} - \mathfrak{u})^\top \mathrm{M} (\mathfrak{c} - \mathfrak{u}),
\end{aligned}$$

where $\mathfrak{c} := \begin{bmatrix} \mathfrak{c}_1 & \cdots & \mathfrak{c}_{n_x} \end{bmatrix}^\top$ and $\mathfrak{d} := \begin{bmatrix} \mathfrak{d}_1 & \cdots & \mathfrak{d}_{n_x} \end{bmatrix}^\top$. Analogously, we obtain the approximation

$$\int_\Omega (S(\mu) - u_d) u_\mu^h[\mu] \, dx \approx (\mathfrak{c} - \mathfrak{u})^\top \mathrm{M} \mathfrak{e},$$

where $\mathfrak{e} := \begin{bmatrix} \mathfrak{e}_1 & \cdots & \mathfrak{e}_{n_x} \end{bmatrix}^\top$. To approximate the time integrals, we apply the trapezoidal rule, resulting in

$$\begin{aligned}
\int_0^T \int_\Omega |S(\mu) - u_d|^2 \, dx \, dt &\approx \int_0^T (\mathfrak{c} - \mathfrak{u})^\top \mathrm{M} (\mathfrak{c} - \mathfrak{u}) \, dt \\
&\approx \frac{\Delta t}{2} \sum_{i=1}^{m} (\mathfrak{c}^{i-1} - \mathfrak{u}^{i-1})^\top \mathrm{M} (\mathfrak{c}^{i-1} - \mathfrak{u}^{i-1}) \\
&\qquad + (\mathfrak{c}^i - \mathfrak{u}^i)^\top \mathrm{M} (\mathfrak{c}^i - \mathfrak{u}^i)
\end{aligned}$$

and

$$\int_0^T \int_\Omega (S(\mu) - u_d) u_\mu^h[\mu] \, dx \, dt \approx \int_0^T (\mathfrak{c} - \mathfrak{u})^\top \mathrm{M} \mathfrak{e} \, dt$$

$$\approx \frac{\Delta t}{2} \sum_{i=1}^{m} (\mathfrak{e}^{i-1} - \mathfrak{u}^{i-1})^\top \mathrm{M} \mathfrak{e}^{i-1} + (\mathfrak{e}^{i} - \mathfrak{u}^{i})^\top \mathrm{M} \mathfrak{e}^{i}.$$

With all required components in place, we now ready to apply the PBB method to the given optimization problem. The algorithm is initialized with the initial guess $\mu^{-1} := 9.0 \cdot 10^{-2} \in \mathcal{P}_{\mathsf{ad}}$, chosen just below the upper bound μ^+ of the admissible parameter set $\mathcal{P}_{\mathsf{ad}}$. It is worth noting that simulations are conducted for various initial guesses within the interval $\mathcal{P}_{\mathsf{ad}}$, consistently yielding similar results. Therefore, only the simulation corresponding to a representative initial guess is presented here.
To analyze the performance of the PBB method, it is executed for a range of values of the regularization parameter σ, specifically

$$\sigma = 0, 10^{-7}, 10^{-5}, 10^{-3}, 10^{-2}.$$

The corresponding results are summarized in Table 5, as well as in Figures 31, 32 and 33.

Table 5: Computed optimal parameter $\bar{\mu}$, corresponding objective value $\hat{J}(\bar{\mu})$ and number of iterations k_{it} required by the PBB method for various values of the regularization parameter σ. The threshold for the PBB method is set to $\varepsilon = 10^{-8}$.

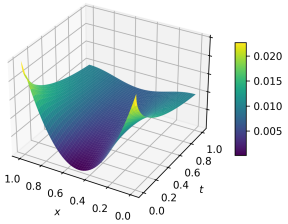
σ	0	10^{-7}	10^{-5}	10^{-3}	10^{-2}
$\bar{\mu}$	$5.0 \cdot 10^{-2}$	$4.998 \cdot 10^{-2}$	$4.801 \cdot 10^{-2}$	$1.770 \cdot 10^{-2}$	$2.851 \cdot 10^{-3}$
$\hat{J}(\bar{\mu})$	$0.0 \cdot 10^{0}$	$1.254 \cdot 10^{-10}$	$1.567 \cdot 10^{-8}$	$2.571 \cdot 10^{-6}$	$8.753 \cdot 10^{-6}$
k_{it}	10	10	9	4	5

Table 5 offers several key observations. Most notably, the algorithm consistently converges to the optimal parameter $\bar{\mu}$ within just a few iterations across all tested values of σ. The highest number of iterations, namely 10, is required for the two smallest values $\sigma = 0$ and $\sigma = 10^{-7}$. These results highlight the high efficiency and robustness of the PBB method in solving the considered optimization problem.
Moreover, for $\sigma = 0$, the algorithm yields an optimal solution of $\bar{\mu} = 5.0 \cdot 10^{-2}$, which exactly matches the target value μ^*. Consequently, the corresponding function value at this point is $\hat{J}(\bar{\mu}) = 0$. This result aligns with theoretical expectations, as for $\sigma = 0$, the regularization term is absent, allowing the desired state to be attained precisely.
Additionally, as σ decreases, the optimum $\bar{\mu}$ exhibits a monotonic increase. This trend can be explained by the role of the regularization term: for larger values of σ, higher values of μ are more strongly penalized, leading to smaller optimal values. As a consequence, the function value $\hat{J}(\bar{\mu})$ at the optimal solution also decreases monotonically with decreasing σ, since the influence of the regularization term diminishes and the primary objective term becomes dominant.

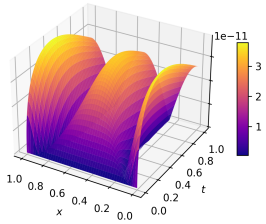
Figure 31 further supports these observations. The top-left panel displays the desired state u_d, while the remaining subplots show the absolute error

$$|S(\bar{\mu}) - u_d|$$

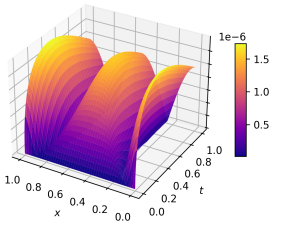
for the optimal parameters $\bar{\mu}$ obtained by the PBB method for various values of σ. For $\sigma = 0$, the desired state is reproduced almost exactly with an error on the order of 10^{-11}. As σ increases, the corresponding errors grow, illustrating the influence of the regularization term. In fact, for $\sigma = 10^{-2}$, the optimum value $\bar{\mu}$ is so small that the corresponding solution $S(\bar{\mu})$ exhibits only slightly diffusion for larger values of t.



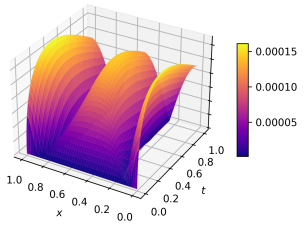
(a) u_d



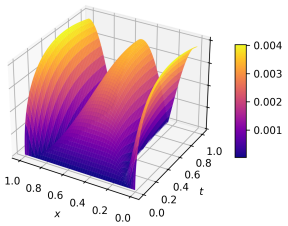
(b) $S(\bar{\mu})$ for $\sigma = 0$



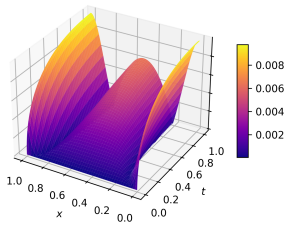
(c) $S(\bar{\mu})$ for $\sigma = 10^{-7}$



(d) $S(\bar{\mu})$ for $\sigma = 10^{-5}$



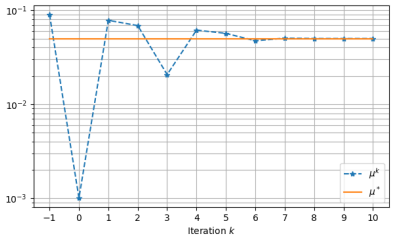
(e) $S(\bar{\mu})$ for $\sigma = 10^{-3}$



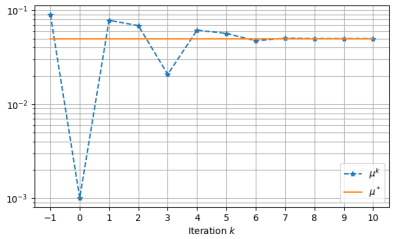
(f) $S(\bar{\mu})$ for $\sigma = 10^{-2}$

Figure 31: The top-left panel shows the desired state u_d. The remaining subfigures display the absolute error $|S(\bar{\mu}) - u_d|$ for the optimal values $\bar{\mu}$ obtained by the PBB method for $\sigma = 0, 10^{-7}, 10^{-5}, 10^{-3}, 10^{-2}$.

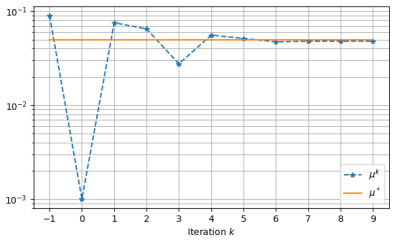
The convergence behavior of the iterates μ^k is reported in Figure 32. As shown, the sequence of iterates converges to the optimal value $\bar{\mu}$ within only a few iterations for all tested values of σ. The initialization is fixed as $\mu^{-1} = 5.250 \cdot 10^{-2}$, followed by $\mu^0 = \mathcal{P}(\nabla \hat{J}(\mu^{-1}))$. Since the derivative $\nabla \hat{J}(\mu^{-1})$ consistently lies below the lower bound $\mu^- = 10^{-3}$, the projection operator maps to this lower bound, resulting in the starting value $\mu^0 = 10^{-3}$. Starting from $k = 1$, the iterates already lie in close proximity to μ^*.



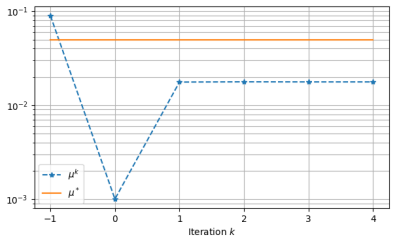
(a) $\sigma = 0$



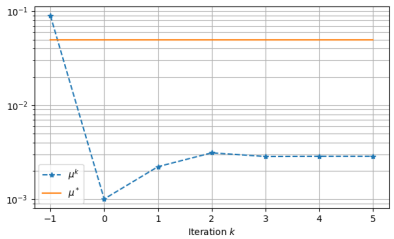
(b) $\sigma = 10^{-7}$



(c) $\sigma = 10^{-5}$



(d) $\sigma = 10^{-3}$



(e) $\sigma = 10^{-2}$

Figure 32: The subfigures show the history of the iterates μ^k during the PBB method for $\sigma = 0, 10^{-7}, 10^{-5}, 10^{-3}, 10^{-2}$, along with the value μ^*.

Finally, Figure 33 does not introduce new information but instead highlights, in a single plot, a comparative visualization of the evolution of the iterates for all considered values of σ. As expected, for smaller values of σ, the iterates tend to converge more closely to the target value μ^*, further confirming the theoretical predictions.

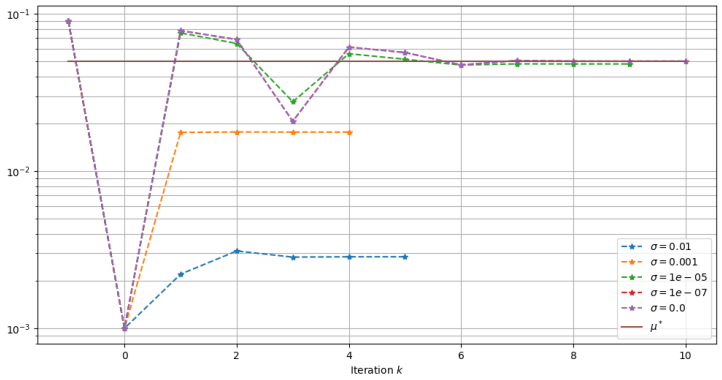


Figure 33: The figure shows the history of the iterates μ^k during the PBB method for $\sigma = 0, 10^{-7}, 10^{-5}, 10^{-3}, 10^{-2}$, along with the value μ^*.

7 Modeling of Gene Regulation via the Activating Hill Function

For the biological foundation of this section, we refer to [19] and [20].

Here, we investigate another RD-PDE system in which the nonlinearity is in the so-called *activating Hill function*. We begin by discretizing the system in space using the FEM, followed by a temporal discretization based on a variant of the *Theta method*. The numerical analysis focuses on exploring how different parameters influence the system's behavior. Finally, we revisit the methods introduced in Section 3 and Section 5 by applying *rDMD* and *POD* to the second component of the system to evaluate their performance in this new setting.

7.1 Mathematical Model Formulation

The so-called activating Hill function serves as a fundamental mathematical tool for capturing cooperative binding behavior in biochemical systems. It plays a key role in synthetically engineered gene networks, where it enables a realistic representation of the production rate of a gene product (e.g., a protein), when the gene, that encodes this product, is regulated by transcription factors (proteins that can bind to specific DNA sequences and activate or inhibit promotors).

Mathematically, the activating Hill function is defined as

$$H(u_1, u_2, K, H_c, k_5) := u_1 \left(k_5 + \frac{u_2^{H_c}}{K^{H_c} + u_2^{H_c}} \right). \tag{7.1}$$

The constant K indicates the concentration of u_2 at which the function attains half of its maximum value. This is called the *Hill activation constant* or *half-saturation constant*. Moreover, the *Hill coefficient* H_c is a measure for the ultrasensitivity of the system response (i.e., how steep is the response curve) and the *leak coefficient* k_5 allows a basal activity of the system, even if $u_2 = 0$.

Based on this function, we introduce a mathematical model to describe the dose-response relationship of u_2. The model is given by a PDE system that captures the temporal and spatial dynamics of three biologically relevant quantities. To formulate the model precisely, the time-space cylinder Ω_T is defined as

$$\Omega_T := (0, T] \times \Omega$$

with domain $\Omega := (0, 1)$ and final time $T := 2$. Furthermore, we set

$$\Gamma_T := [0, T] \times \Gamma,$$

where $\Gamma := \{0, 1\}$. Set $V := H^1(\Omega)$, $H := L^2(\Omega)$ and

$$H^{d_\Omega} := \underbrace{H \times \cdots \times H}_{d_\Omega \text{ times}}.$$

Let H^{d_Ω} again be equipped with the inner product defined in (6.1). We seek for a vector-valued function

$$u := \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^\top : \Omega_T \to \mathbb{R}^3$$

that satisfies the following IBVP:

$$\begin{aligned} (u_1)_t + d_{u_1}(u) &= 0 \quad \text{in } \Omega_T, \\ (u_2)_t - \mu_{u_2}\Delta u_2 + d_{u_2}(u) &= 0 \quad \text{in } \Omega_T, \\ (u_3)_t + d_{u_3}(u) &= 0 \quad \text{in } \Omega_T, \end{aligned} \tag{7.2a}$$

where the real-valued functions $d_{u_1}, d_{u_2}, d_{u_3}: \mathbb{R}^3 \to \mathbb{R}$ are defined as

$$\begin{aligned} d_{u_1}(u) &:= k_1 u_1 - k_2, \\ d_{u_2}(u) &:= k_3 u_2, \\ d_{u_3}(u) &:= -k_4 H(u_1, u_2, K, H_c, k_5) + k_6 u_3. \end{aligned}$$

The positive reaction constants $k_1, k_2, k_3, k_4, k_5, k_6$ are chosen as follows:

k_1	k_2	k_3	k_4	k_5	k_6
$2.8 \cdot 10^{-3}$	$2.0 \cdot 10^{-1}$	$1.1 \cdot 10^{-3}$	$2.0 \cdot 10^{-2}$	$5.0 \cdot 10^{-2}$	$1.8 \cdot 10^{-2}$

Moreover, the constant

$$\mu_{u_2} \in \{0, 3.6 \cdot 10^{-2}\}$$

models the diffusion rate of u_2. We define the parameter choices

$$\begin{aligned} K &:= 50, \\ H_c &:= 2. \end{aligned}$$

Furthermore, homogeneous Neumann boundary conditions are applied on the entire boundary Γ_T:

$$\begin{aligned} \nabla u_1 \cdot n &= 0 \quad \text{on } \Gamma_T, \\ \nabla u_2 \cdot n &= 0 \quad \text{on } \Gamma_T, \\ \nabla u_3 \cdot n &= 0 \quad \text{on } \Gamma_T. \end{aligned} \tag{7.2b}$$

The initial functions are defined as

$$\begin{aligned} u_1(0) &:= 5.0 && \text{in } \Omega, \\ u_2(0, x) &:= 10^{-2} + a \cdot \exp\left(-\frac{(x - x_0)^2}{2\sigma^2}\right) && \text{f.a.a. } x \in \Omega, \\ u_3(0) &:= 1.0 && \text{in } \Omega. \end{aligned} \tag{7.2c}$$

The initial condition for u_2 is chosen as a bell-shaped distribution modeled by a Gaussian function. This input represents a locally increased concentration of the diffusing concentration within the domain and serves as an initial stimulus to activate the system. The Gaussian function is characterized by the parameters

$$b := 10^{-2},$$

$$
\begin{aligned}
a &:= 2.0 \cdot 10^{-1}, \\
x_0 &:= 5.0 \cdot 10^{-1}, \\
\sigma &:= 5.0 \cdot 10^{-1}.
\end{aligned}
$$

Here, a models the amplitude of the Gaussian bell, i.e., the peak height above the baseline b. The parameter x_0 specifies the location of the peak within the interval Ω, while standard deviation σ controls the width of the distribution and thus determines the spatial extent of the initial signal. Moreover, the baseline b ensures a small, biologically plausible background level of the concentration u_2 throughout the domain.

Note that – due to its structure – (7.2) is a coupled PDE-ODE system. Only the second equation is a PDE, while the first and third equations are ODEs, which are defined for almost all $x \in \Omega$.

7.2 Finite Element Discretization

First, for the spatial discretization, we triangulate the domain Ω into a FE mesh with $n_x := 1024$ nodes. We then introduce the FE product space

$$
\mathcal{V}^{\mathrm{FE}} := V^{\mathrm{FE}} \times V^{\mathrm{FE}} \times V^{\mathrm{FE}},
$$

where V^{FE} denotes the FE space spanned by the standard piecewise linear basis functions $\varphi_i \in V$ for $i = 1, \ldots, n_x$. The associated $3n_x$ basis functions of the product space $\mathcal{V}^{\mathrm{FE}}$ are defined as

$$
\begin{aligned}
\phi_1 = \begin{bmatrix} \varphi_1 \\ 0 \\ 0 \end{bmatrix}, \quad \ldots, \quad \phi_{n_x} = \begin{bmatrix} \varphi_{n_x} \\ 0 \\ 0 \end{bmatrix}, \\
\phi_{n_x+1} = \begin{bmatrix} 0 \\ \varphi_1 \\ 0 \end{bmatrix}, \quad \ldots, \quad \phi_{2n_x} = \begin{bmatrix} 0 \\ \varphi_{n_x} \\ 0 \end{bmatrix}, \\
\phi_{2n_x+1} = \begin{bmatrix} 0 \\ 0 \\ \varphi_1 \end{bmatrix}, \quad \ldots, \quad \phi_{3n_x} = \begin{bmatrix} 0 \\ 0 \\ \varphi_{n_x} \end{bmatrix}.
\end{aligned}
$$

The semi-discrete system is obtained by applying the Galerkin projection to the continuous model (analogously to the derivation in Section 6), leading to the following system of ODEs:

$$
\begin{aligned}
\mathrm{M}\dot{\mathrm{u}}_1 + k_1 \mathrm{M}\mathrm{u}_1 - k_2 \mathrm{M}1_{n_x} &= 0_{n_x}, \\
\mathrm{M}\dot{\mathrm{u}}_2 + \mu_{u_2} \mathrm{S}\mathrm{u}_2 + k_3 \mathrm{M}\mathrm{u}_2 &= 0_{n_x}, \\
\mathrm{M}\dot{\mathrm{u}}_3 - k_4 k_5 \mathrm{M}\mathrm{u}_1 - k_4 \mathrm{r}_{\mathrm{nl}} + k_6 \mathrm{M}\mathrm{u}_3 &= 0_{n_x}.
\end{aligned} \tag{7.3}
$$

Here, M and S denotes the mass and stiffness matrix, respectively (cf. (6.16) and (6.17)). The vector of ones with n_x entries is denoted by 1_{n_x}.

The term $-k_2 \mathrm{M}1_{n_x}$ in the first equation of (7.3) is the result of projecting the constant function $-k_2$ onto the FE space: the projection yields the corresponding FE function

$$
-k_2 \sum_{i=1}^{n_x} \varphi_i,
$$

which satisfies $-k_2 \sum_{i=1}^{n_x} \varphi_i(x_i) = -k_2$ at every grid point x_i. A direct consequence of the property that the piecewise linear basis functions φ_i evaluate to 1 at node x_i and 0 elsewhere. Testing with the j-th basis function φ_j results in

$$\langle -k_2 \sum_{i=1}^{n_x} \varphi_i, \varphi_j \rangle_H = -k_2 \sum_{i=1}^{n_x} \langle \varphi_i, \varphi_j \rangle_H,$$

which corresponds precisely to the j-th component of the vector $-k_2 \mathrm{M} 1_{n_x}$.

The term $\mathrm{r_{nl}}$ describes the nonlinear coupling

$$u_1 \frac{u_2^{H_c}}{K^{H_c} + u_2^{H_c}}$$

induced by the Hill function (7.1). For $j = 1, \ldots, n_x$, it is given by

$$[\mathrm{r_{nl}}]_j := \left\langle \frac{\left(\sum_{i=1}^{n_x} \mathrm{u}_{1,i} \varphi_i\right) \left(\sum_{l=1}^{n_x} \mathrm{u}_{2,l} \varphi_l\right)^{H_c}}{K^{H_c} + \left(\sum_{l=1}^{n_x} \mathrm{u}_{2,l} \varphi_l\right)^{H_c}}, \varphi_j \right\rangle_H .$$

To obtain the fully discretized system, we employ a variant of the Theta method. Here, the nonlinear term is treated only explicitly, whereas the remaining parts of the equation are handled either implicitly or explicitly, depending on the chosen value of $\theta \in [0, 1]$.

7.3 Time Discretization Using the Theta Method

The one-step Theta method provides a unified framework for various integration schemes by introducing a parameter $\theta \in [0, 1]$. Consider an ODE of the form

$$\frac{\mathrm{d}}{\mathrm{d}t} u = f(u), \quad u(0) = u_0.$$

The Theta method approximates the solution at equidistant time points $t_k := k\Delta t$, where $k = 0, \ldots, m$ and $m := {}^{T}/_{\Delta t}$. It computes $u^k \approx u(t_k)$ by solving

$$\frac{u^k - u^{k-1}}{\Delta t} = \theta f(u^k) + (1 - \theta) f(u^{k-1}).$$

Choosing $\theta = 0$ yields the *explicit Euler method*, $\theta = 1$ the *implicit Euler method* and $\theta = {}^1/_2$ the *Crank-Nicolson method*. The explicit Euler method is the simplest of the three methods, but it is only conditionally stable unless very small time steps are used. In contrast, the implicit Euler method offers strong stability properties, but suffers from only first-order accuracy. While the Crank-Nicolson method offers both stability and second-order accuracy, it may exhibit oscillatory behavior in the presence of strong nonlinearities and can require additional stabilization.

However, it is worth noting that the second-order accuracy of the Crank-Nicolson method cannot be fully exploited for system (7.3) due to the explicit treatment of the nonlinear term. As a result, only first-order accuracy $\mathcal{O}(\Delta t)$ is achieved. Therefore, the implicit Euler method is the most reasonable choice.

For system (7.3), the Theta method yields the following fully discretized formulation:

$$\begin{aligned}
\mathrm{M}\frac{\mathrm{u}_1^k - \mathrm{u}_1^{k-1}}{\Delta t} + \theta k_1 \mathrm{M}\mathrm{u}_1^k &= -(1-\theta)k_1\mathrm{M}\mathrm{u}_1^{k-1} + k_2\mathrm{M}1_{n_x},\\
\mathrm{M}\frac{\mathrm{u}_2^k - \mathrm{u}_2^{k-1}}{\Delta t} + \theta(\mu_{u_2}\mathrm{S}\mathrm{u}_2^k + k_3\mathrm{M}\mathrm{u}_2^k) &= -(1-\theta)(\mu_{u_2}\mathrm{S}\mathrm{u}_2^{k-1} + k_3\mathrm{M}\mathrm{u}_2^{k-1}),\\
\mathrm{M}\frac{\mathrm{u}_3^k - \mathrm{u}_3^{k-1}}{\Delta t} + \theta(-k_4k_5\mathrm{M}\mathrm{u}_1^k + k_6\mathrm{M}\mathrm{u}_3^k) &= -(1-\theta)(-k_4k_5\mathrm{M}\mathrm{u}_1^{k-1} + k_6\mathrm{M}\mathrm{u}_3^{k-1}) + k_4\mathrm{r}_{\mathrm{nl}}^{k-1}.
\end{aligned}$$

As previously mentioned, the nonlinear term r_{nl} is treated only explicitly, i.e., it is only evaluated at the previous time step t_{k-1}. This avoids the need for a nonlinear solver but imposes restrictions on the maximum allowable time step size Δt in order to ensure stability.

Collecting all terms at time t_k on the left-hand side results in a linear system of the form:

$$A\mathbf{u}^k = \mathbf{b}^{k-1}, \tag{7.4}$$

where A is the system matrix, $\mathbf{u}^k = \begin{bmatrix} \mathrm{u}_1^k & \mathrm{u}_2^k & \mathrm{u}_3^k \end{bmatrix}^\top$ is the vector of unknowns at time t_k and $\mathbf{b}^{k-1}$ is the right-hand side vector containing the explicit contributions from the previous time step t_{k-1}. The matrix A has block structure

$$A := \begin{bmatrix} \left(\frac{1}{\Delta t} + \theta k_1\right)\mathrm{M} & 0_{n_x\times n_x} & 0_{n_x\times n_x} \\ 0_{n_x\times n_x} & \left(\frac{1}{\Delta t} + \theta k_3\right)\mathrm{M} + \theta\mu_{u_2}\mathrm{S} & 0_{n_x\times n_x} \\ -\theta k_4k_5\mathrm{M} & 0_{n_x\times n_x} & \left(\frac{1}{\Delta t} + \theta k_6\right)\mathrm{M} \end{bmatrix}.$$

For the right-hand side vector $\mathbf{b}^{k-1}$, we have

$$\mathbf{b}^{k-1} := \begin{bmatrix} \left(\frac{1}{\Delta t} - (1-\theta)k_1\right)\mathrm{M}\mathrm{u}_1^{k-1} + k_2\mathrm{M}1_{n_x} \\ \left(\frac{1}{\Delta t} - (1-\theta)k_3\right)\mathrm{M}\mathrm{u}_2^{k-1} - (1-\theta)\mu_{u_2}\mathrm{S}\mathrm{u}_2^{k-1} \\ \left(\frac{1}{\Delta t} - (1-\theta)k_6\right)\mathrm{M}\mathrm{u}_3^{k-1} + (1-\theta)k_4k_5\mathrm{M}\mathrm{u}_1^{k-1} + k_4\mathrm{r}_{\mathrm{nl}}^{k-1} \end{bmatrix}.$$

Solving system (7.4) yields the approximated solution components at the new time step t_k. The block structure of A and the sparsity of matrices M and S allow for efficient numerical implementation.

To efficiently solve large sparse linear systems, we employ the `scipy.sparse.linalg.factorized` function from the `SciPy` library. This function performs a one-time LU *factorization* of the matrix A and returns a callable object that can quickly solve the system for different right-hand sides $\mathbf{b}^{k-1}$. This is particularly advantageous because the matrix A remains constant across all time steps. Unlike general solvers, which analyze and factorize the system from scratch on each call, `factorized` allows preprocessing of the matrix structure, substantially reducing computation time.

7.4 Numerical Results

In this section, we present a variety of numerical results regarding system (7.4).

Figure 34 shows the development of u_1, u_2 and u_3 at five consecutive time points $t = 0, 0.5, 1, 1.5, 2$ – from top to bottom. The results are depicted for two diffusion

scenarios, $\mu_{u_2} = 0$ and $\mu_{u_2} = 3.6 \cdot 10^{-2}$. Over time, the concentration of u_1 increases, whereas that of u_3 decreases. This behavior is expected given the model parameters and reaction terms. Moreover, u_1 and u_3 show nearly identical trajectories for the two different diffusion constants – differences are at most numerical and barely visible. In contrast, u_2 differs significantly between the following two cases: while for $\mu_{u_2} = 0$ there is almost no change over time, for $\mu_{u_2} = 3.6 \cdot 10^{-2}$, a clear diffusion of the initial Gaussian bell is oberseved.

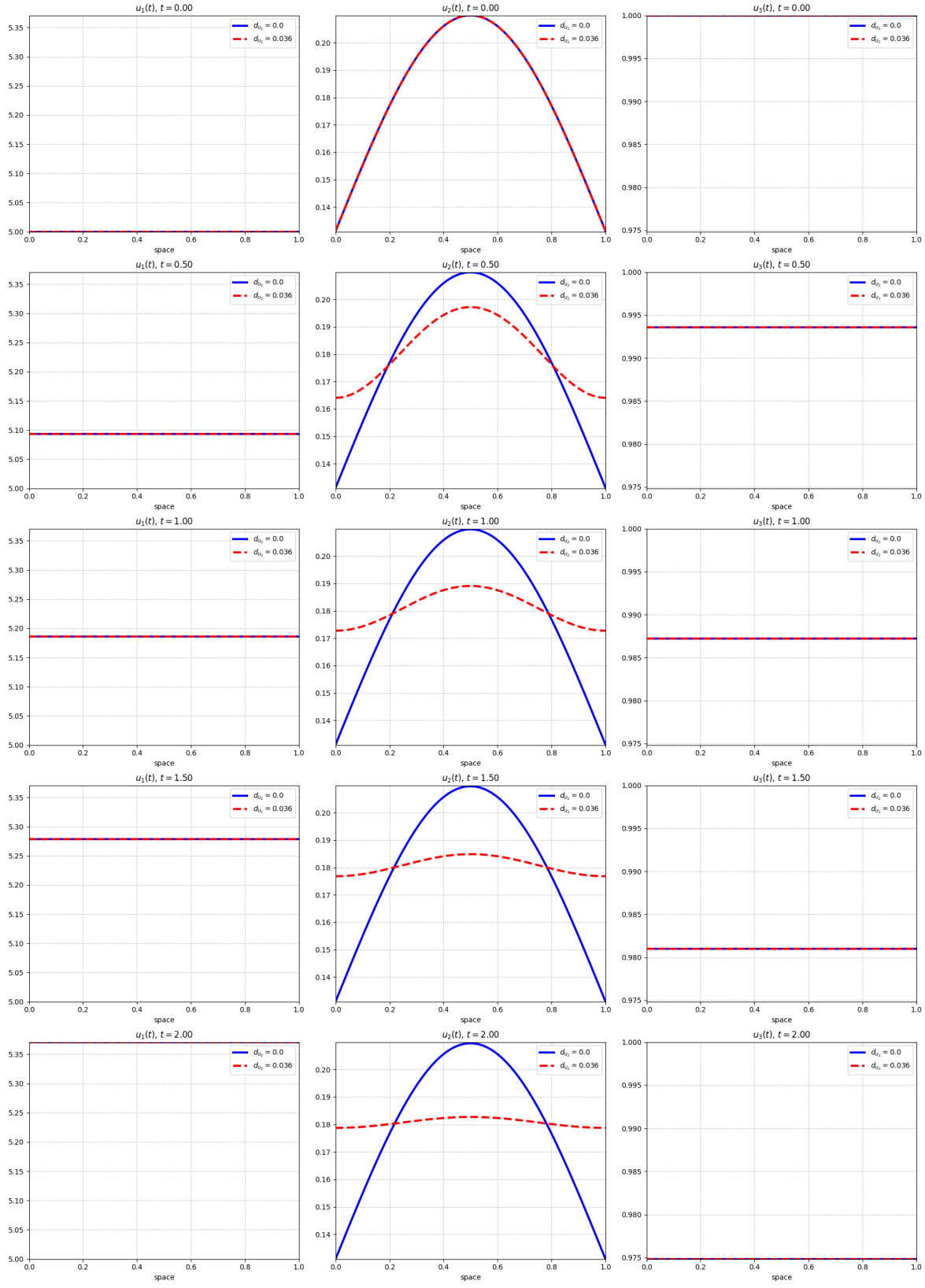


Figure 34: Snapshots of u_1 (left), u_2 (middle) and u_3 (right) at five consecutive time points $t = 0, 0.5, 1, 1.5, 2$ for the two diffusion constants $\mu_{u_2} = 0$ (blue) and $\mu_{u_2} = 3.6 \cdot 10^{-2}$ (red).

This observation is supported in detail by Figure 35. Here, we compare u_2 in the

diffusion-free scenario ($\mu_{u_2} = 0$) at the initial time $t = 0$ and the final time $t = T$. The initial Gaussian bell remains almost unchanged. Note that the previous simulations are carried out for $\theta = 1$.

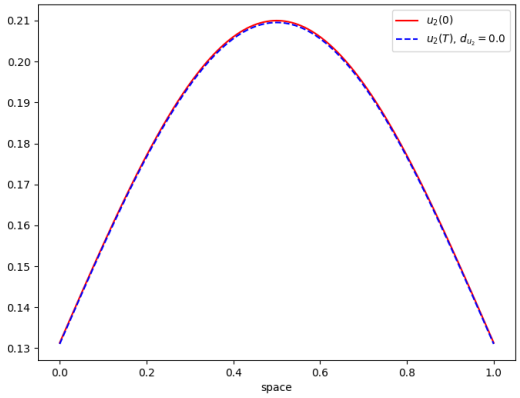


Figure 35: Initial profile and profile at the final time T of u_2 for $\mu_{u_2} = 0$.

In Figure 36, the final-time profiles of u_1, u_2 and u_3 for three different values of θ ($\theta = 0, 0.5, 1$) and two diffusion constants ($\mu_{u_2} = 0, 3.6 \cdot 10^{-2}$) are illustrated. The solutions obtained from all three time-stepping methods are very similar in the diffusion-free scenario, with differences on the order of magnitude 10^{-5}. However, for $\mu_{u_2} = 3.6 \cdot 10^{-2}$, the explicit Euler method fails completely: only `NaN` values are returned, indicating numerical instability. Consequently, no profiles are shown for $\theta = 0$. In contrast, the two implicit methods remain stable and yield consistent results.

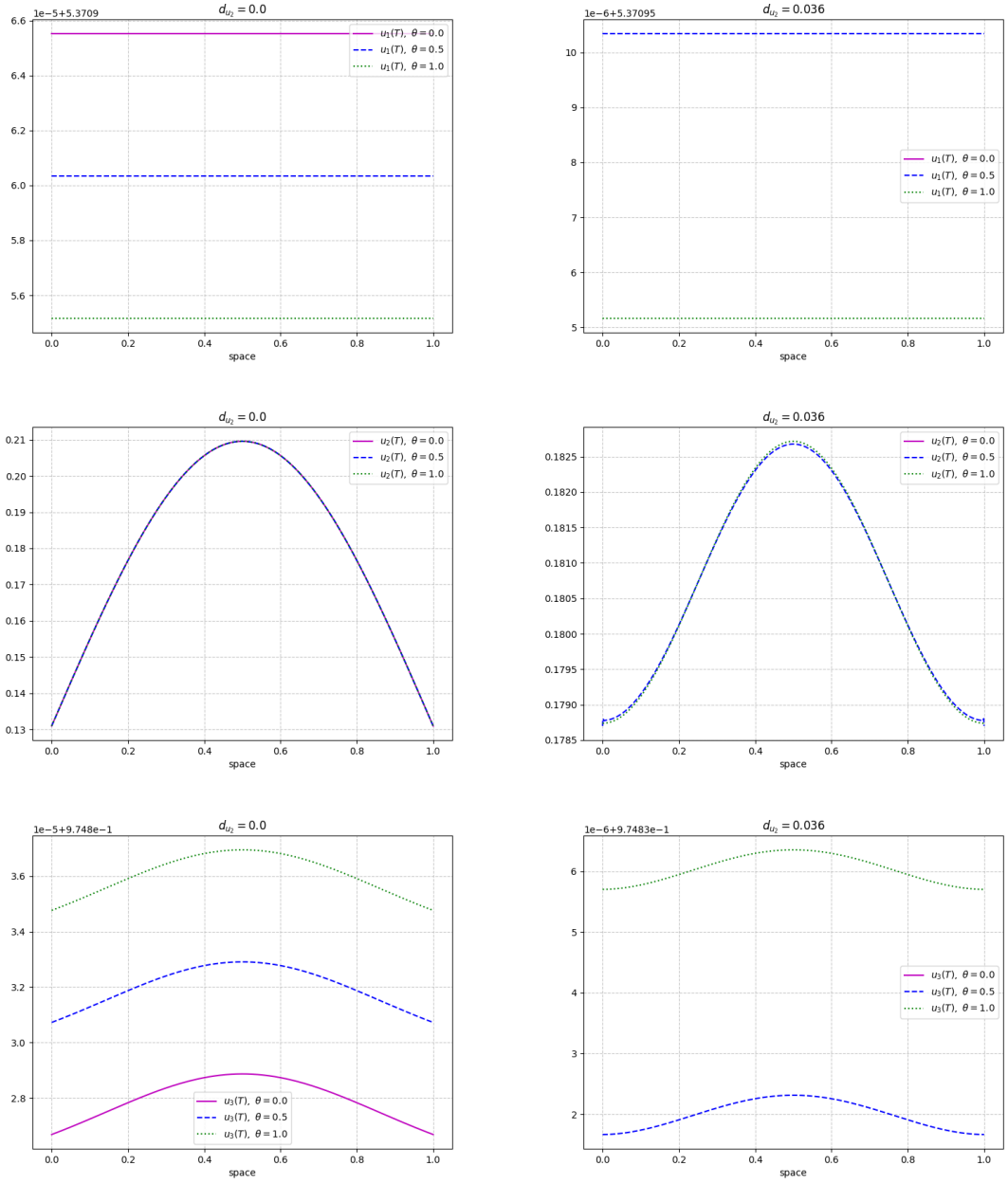


Figure 36: Profiles of u_1, u_2 and u_3 at the final time T for $\theta = 0, 0.5, 1$. Left: $\mu_{u_2} = 0$. Right: $\mu_{u_2} = 3.6 \cdot 10^{-2}$.

Next, Figure 37 illustrates the influence of different constant initial values for u_2. The simulations are carried out using the implicit Euler method, where $u_2(0) = c$ with $c = 0, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}$.
Since the profiles of u_1 at the final time T are identical for all tested initial values $u_2(0)$, only the solution corresponding to $u_2(0) = 10^{-2}$ is shown for clarity. In contrast, the profiles of u_2 and u_3 differ significantly in magnitude at the final time T, depending on the chosen initial value $u_2(0)$.

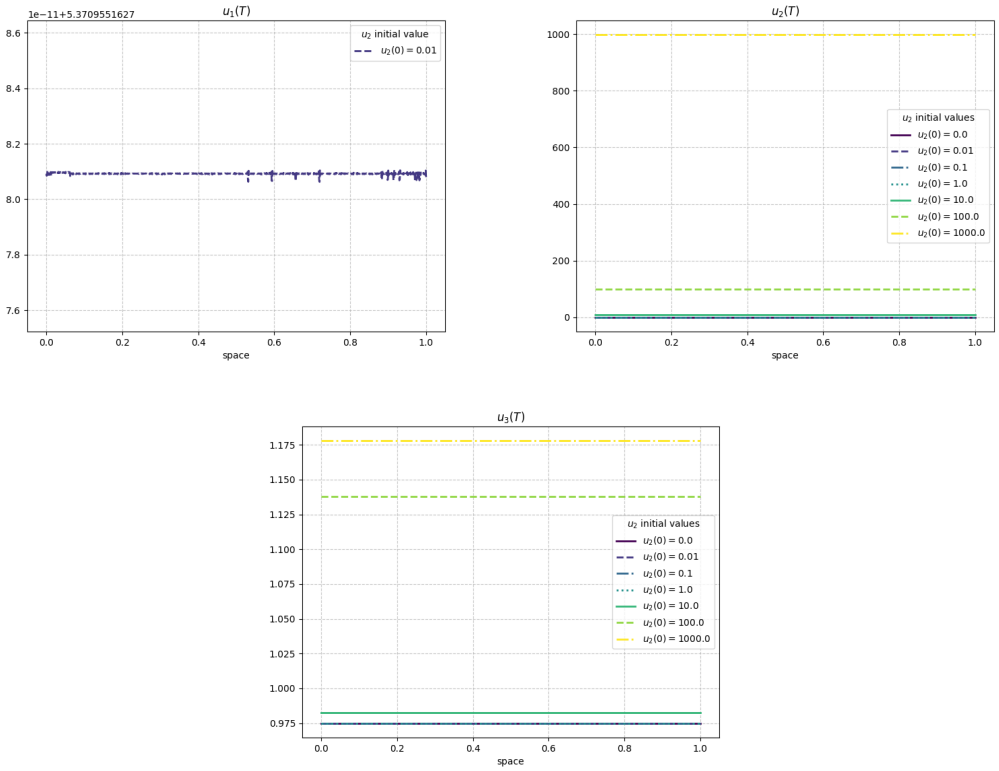


Figure 37: Profiles of u_1, u_2 and u_3 at the final time T for varying constant initial values of u_2. Simulations are performed for $\theta = 1$ and $\mu_{u_2} = 3.6 \cdot 10^{-2}$.

We also perform simulations for different Hill coefficients H_c (e.g., $H_c = 0, 1, 2, 3$), but observe no significant influence on the results. Therefore, these findings are not presented here.

Finally, we establish a connection to Section 3 and Section 5. To this end, we consider the snapshot matrix generated using the implicit Euler method for the parameter value $\mu_{u_2} = 3.6 \cdot 10^{-2}$ (cf. Figure 34, middle), which serves as the reference dataset. The subdataset corresponding to u_1, u_2 and u_3 are denoted by U_1, U_2 and $U_3 \in \mathbb{R}^{n_x \times (m+1)}$, respectively. First, we apply the rDMD method to the reference dataset U_2. Then, we solve the POD-ROM corresponding to (7.3) and use the resulting reduced-order solution corresponding to u_2 for evaluation.

MATLAB's `rank` function reports a numerical rank of 17 for the snapshot matrix U_2. The economy-sized SVD for all three subdatasets yields the left singular vectors $\Psi_1, \Psi_2, \Psi_3 \in \mathbb{R}^{n_x \times (m+1)}$ and the singular values $(\sigma_1)_i, (\sigma_2)_i, (\sigma_3)_i$ (see Figure 38, left). Based on the decay of the singular values $(\sigma_2)_i$, we select the target ranks $r = 1, \ldots, 21$ for both rDMD and POD. A closer look at the POD basis (see Figure 38, right) further supports this choice: the first 21 POD basis vectors capture the essential features of the dataset. The remaining POD basis vectors exhibit fine-scale oscillations, indicating that they primarily represent noise rather than meaningful structure.

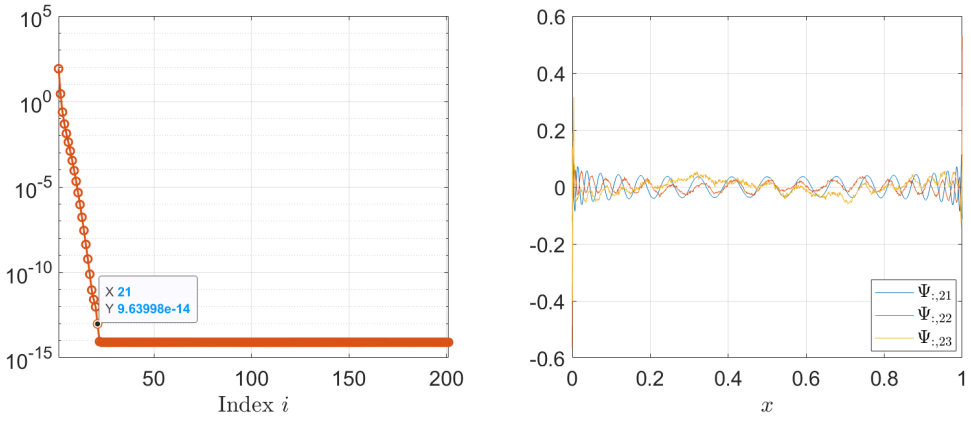


Figure 38: Left: singular values $(\sigma_2)_i$ of the data matrix U_2. Right: POD basis vectors $(\Psi_2)_{:,21}$, $(\Psi_2)_{:,22}$ and $(\Psi_2)_{:,23}$.

For rDMD, we use the parameters $p := 0$ and $q := 2$ and denote the reconstructed matrix by $\tilde{U}_2^{\mathrm{rDMD}} \in \mathbb{R}^{n_x \times (m+1)}$. The rank-$r$ POD-ROM corresponding to (7.3) is given by

$$\begin{aligned}
(\Psi_1^r)^\top \mathrm{M} \Psi_1^r \dot{\tilde{\mathrm{u}}}_1 + k_1 (\Psi_1^r)^\top \mathrm{M} \Psi_1^r \tilde{\mathrm{u}}_1 - k_2 (\Psi_1^r)^\top \mathrm{M} 1_{n_x} &= 0_r, \\
(\Psi_2^r)^\top \mathrm{M} \Psi_2^r \dot{\tilde{\mathrm{u}}}_2 + \mu_{u_2} (\Psi_2^r)^\top \mathrm{S} \Psi_2^r \tilde{\mathrm{u}}_2 + k_3 (\Psi_2^r)^\top \mathrm{M} \Psi_2^r \tilde{\mathrm{u}}_2 &= 0_r, \\
(\Psi_3^r)^\top \mathrm{M} \Psi_3^r \dot{\tilde{\mathrm{u}}}_3 - k_4 k_5 (\Psi_3^r)^\top \mathrm{M} \Psi_1^r \tilde{\mathrm{u}}_1 - k_4 (\Psi_3^r)^\top \tilde{\mathrm{r}}_{\mathrm{nl}} + k_6 (\Psi_3^r)^\top \mathrm{M} \Psi_3^r \tilde{\mathrm{u}}_3 &= 0_r.
\end{aligned} \tag{7.5}$$

where $\Psi_1^r, \Psi_2^r, \Psi_3^r \in \mathbb{R}^{n_x \times r}$ consist of the first r columns of Ψ_1, Ψ_2, Ψ_3, respectively. Here, the vector $\tilde{\mathrm{r}}_{\mathrm{nl}}$ is defined as

$$[\tilde{\mathrm{r}}_{\mathrm{nl}}]_j := \left\langle \frac{\left(\sum_{i=1}^{n_x} (\Psi_1^r \tilde{\mathrm{u}}_1)_i \varphi_i\right) \left(\sum_{l=1}^{n_x} (\Psi_2^r \tilde{\mathrm{u}}_2)_l \varphi_l\right)^{H_c}}{K^{H_c} + \left(\sum_{l=1}^{n_x} (\Psi_2^r \tilde{\mathrm{u}}_2)_l \varphi_l\right)^{H_c}}, \varphi_j \right\rangle_H .$$

We solve (7.5) by using the implicit Euler method. Then, the full-order solution corresponding to u_2 is obtained via

$$\tilde{U}_2^{\mathrm{POD}} := \Psi_2^r \begin{bmatrix} \tilde{\mathrm{u}}_2^0 & \cdots & \tilde{\mathrm{u}}_2^m \end{bmatrix} \in \mathbb{R}^{n_x \times (m+1)}.$$

The reference data U_2 is depicted in Figure 39 (left), whereas the relative Frobenius errors (3.20) resulting from rDMD and POD are reported in the right panel of Figure 39 for $r = 1, \ldots, 21$. We observe that both methods yield good approximations. Up to $r = 14$, the error values decrease similarly for both methods. However, the error curve corresponding to rDMD attains its minimum value of $1.213 \cdot 10^{-10}$ at $r = 14$. After this point, the curve rises again, reaching a value of $3.842 \cdot 10^{-6}$ at $r = 21$. In contrast, the POD error decreases monotonically, reaching a plateau for $r \geq 16$ with an order of magnitude of 10^{-12}.

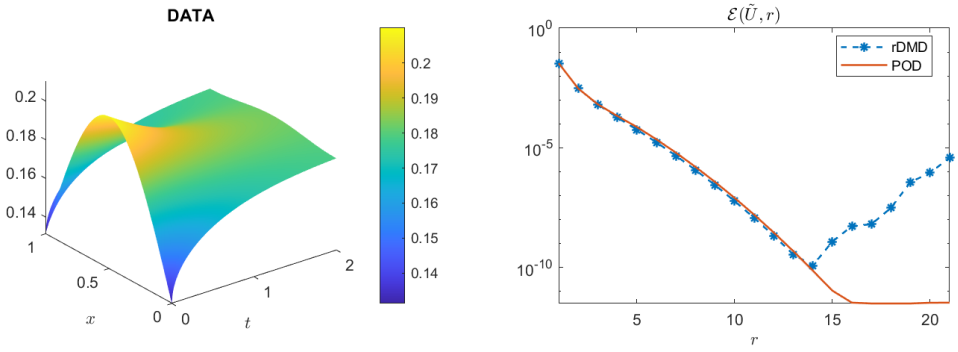


Figure 39: Left: the reference data U_2. Right: relative Frobenius errors (3.20) for rDMD and POD for $r = 1, \ldots, 21$. The rDMD method is applied with parameters $p = 0$ and $q = 2$.

The quality of the reconstruction even for small target ranks r is illustrated in Figures 40. Shown are the POD reconstructions for $r = 1, \ldots, 3$. While the reconstruction for $r = 1$ does not yet provide a good approximation over the entire time interval $[0, T]$, the reconstructions for $r = 2, 3$ already match the reference data very closely (cf. Figure 39, left).

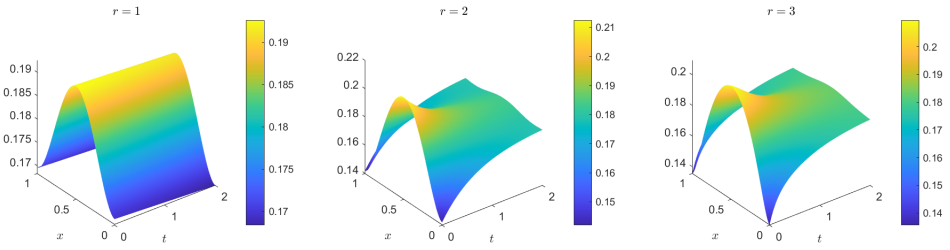


Figure 40: The POD reconstructions for $r = 1, \ldots, 3$.

The corresponding absolute errors of the POD reconstructions are shown in Figure 41, confirming that the reconstructions for $r = 2, 3$ are already very accurate, deviating from the reference data only on the order of 10^{-3}.

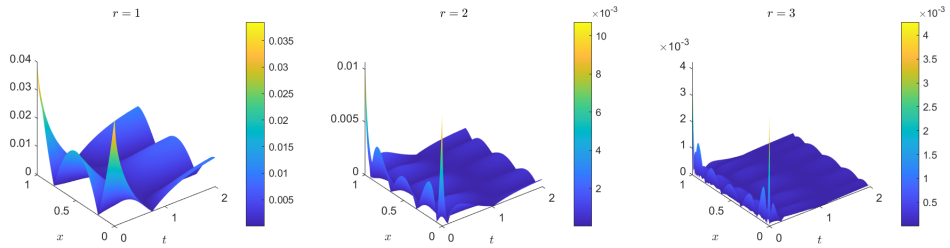


Figure 41: Absolute errors of the POD reconstructions for $r = 1, \ldots, 3$.

8 Conclusion and Outlook

In this thesis, we demonstrated the effectiveness and accuracy of data-driven MOR techniques for analyzing the dynamics of RD-PDE systems, particularly those exhibiting periodic behavior and Turing instabilities.
To address the challenges posed by high-dimensional data and long simulation times, we introduced a randomized DMD variant as well as a localized, piecewise application of DMD. These adaptations significantly improved computational efficiency and reconstruction quality in settings where the standard DMD approach failed.
Likewise, POD and POD-DEIM provided an effective framework for approximating the dynamics of RD-PDE systems. Nevertheless, we identified scenarios in which the standard POD and POD-DEIM approximations were insufficient in terms of accuracy. In response, correction-based enhancements were developed and an adaptive strategy was proposed to further exploit structural properties in the data. Both robustness and convergence speed were significantly improved as a result.

Beyond MOR, we addressed the problem of parameter identification. An optimization framework was developed to estimate diffusion parameters by minimizing a cost functional subject to a RD-PDE constraint and a box constraints on the diffusion parameter. To determine the necessary gradients of the cost functional, we employed a sensitivity-based approach. These gradients were integrated into a PBB optimization method. The method yields fast convergence within only a few iterations and produced expected results across various regularization parameters.

The findings in this thesis suggest several promising directions for further investigation. One natural extension involves the treatment of more complex spatial domains and boundary conditions. In addition, the pDMD approach could be enhanced by incorporating adaptive segmentation strategies based on error indicators or dynamic regime changes.
Similarly, combining reduced-order modeling with machine learning techniques could open up new possibilities for surrogate modeling.
Lastly, extending the methodology to systems with time-dependent or spatially varying coefficients, or introducing stochastic elements into the model, would further increase its applicability to real-world scenarios – particularly in the modeling of biological or chemical systems where uncertainties are inherent.

References

[1] A. Alla, A. Monti, and I. Sgura. "Adaptive POD-DEIM Correction for Turing Pattern Approximation in Reaction-Diffusion PDE Systems". In: *Journal of Numerical Mathematics* 31.3 (2023), pp. 205–229. DOI: 10.1515/jnma-2022-0025.

[2] A. Alla, A. Monti, and I. Sgura. "Piecewise DMD for Oscillatory and Turing Spatio-Temporal Dynamics". In: *Computers and Mathematics with Applications* 160 (2024), pp. 108–124. DOI: 10.1016/j.camwa.2024.02.022.

[3] B. Azmi and K. Kunisch. "Analysis of the Barzilai-Borwein Step-Sizes for Problems in Hilbert Spaces". In: *Journal of Optimization Theory and Applications* 185 (2020), pp. 819–844. DOI: 10.1007/s10957-020-01677-y.

[4] J. Barzilai and J. M. Borwein. "Two-Point Step Size Gradient Methods". In: *IMA Journal of Numerical Analysis* 8.1 (1988), pp. 141–148. DOI: 10.1093/imanum/8.1.141.

[5] D. P. Bertsekas. *Nonlinear Programming*. 2nd ed. Athena scientific optimization and computation series. Athena Scientific, 1999.

[6] E. Casas. "Pontryagin's Principle for State-Constrained Boundary Control Problems of Semilinear Parabolic Equations". In: *SIAM Journal on Control and Optimization* 35.4 (1997), pp. 1297–1327. DOI: 10.1137/S0363012995283637.

[7] M. C. D'Autilia, I. Sgura, and V. Simoncini. "Matrix-Oriented Discretization Methods for Reaction-Diffusion PDEs: Comparisons and Applications". In: *Computers and Mathematics with Applications* 79.7 (2020), pp. 2067–2085. DOI: 10.1016/j.camwa.2019.10.020.

[8] W. Dahmen and A. Reusken. *Numerik für Ingenieure und Naturwissenschaftler*. Second edition. Springer Berlin, Heidelberg, 2008. DOI: 10.1007/978-3-540-76493-9.

[9] M. Dobrowolski. *Angewandte Funktionalanalysis: Funktionalanalysis, Sobolev-Räume und elliptische Differentialgleichungen*. 2nd ed. Springer Berlin, Heidelberg, 2010. DOI: 10.1007/978-3-642-15269-6.

[10] N. B. Erichson et al. "Randomized Dynamic Mode Decomposition". In: *SIAM Journal on Applied Dynamical Systems* 18.4 (2019), pp. 1867–1891. DOI: 10.1137/18M1215013.

[11] A. Gerisch and M. A. J. Chaplain. "Robust Numerical Methods for Taxis-Diffusion-Reaction Systems: Applications to Biomedical Problems". In: *Mathematical and Computer Modelling* 43.1 (2006), pp. 49–75. DOI: 10.1016/j.mcm.2004.05.016.

[12] M. Gubisch and S. Volkwein. "Chapter 1: Proper Orthogonal Decomposition for Linear-Quadratic Optimal Control". In: *Model Reduction and Approximation*, pp. 3–63. DOI: 10.1137/1.9781611974829.ch1.

[13] M. Hinze et al. *Optimization with PDE Constraints*. 1st ed. Philadelphia, PA: Springer Dordrecht, 2008. DOI: 10.1007/978-1-4020-8839-1.

[14] Y. Huang et al. "On the Acceleration of the Barzilai-Borwein Method". In: *Computational Optimization and Applications* 81 (2022), pp. 717–740. DOI: 10.1007/s10589-022-00349-z.

[15] J. N. Kutz et al. *Dynamic Mode Decomposition*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2016. DOI: 10.1137/1.9781611974508.

[16] D. Lacitignola et al. "Turing Pattern Formation on the Sphere for a Morphochemical Reaction-Diffusion Model for Electrodeposition". In: *Communications in Nonlinear Science and Numerical Simulation* 48 (2017), pp. 484–508. DOI: 10.1016/j.cnsns.2017.01.008.

[17] A. N. Langville and W. J. Stewart. "The Kronecker Product and Stochastic Automata Networks". In: *Journal of Computational and Applied Mathematics* 167.2 (2004), pp. 429–447. DOI: 10.1016/j.cam.2003.10.010.

[18] J. Lefèvre and J.-F. Mangin. "A Reaction-Diffusion Model of Human Brain Development". In: *PLOS Computational Biology* 6.4 (2010), pp. 1–10. DOI: 10.1371/journal.pcbi.1000749.

[19] J. D. Murray. *Mathematical Biology I. An Introduction*. Third edition. Springer New York, NY, 2002. DOI: 10.1007/b98868.

[20] J. D. Murray. *Mathematical Biology II. Spatial Models and Biomedical Applications*. Third edition. Springer New York, NY, 2003. DOI: 10.1007/b98869.

[21] J.-P. Raymond and H. Zidani. "Hamiltonian Pontryagin's Principles for Control Problems Governed by Semilinear Parabolic Equations". In: *Applied Mathematics and Optimization* 39 (1999), pp. 143–177. DOI: 10.1007/s002459900102.

[22] S. Salsa. *Partial Differential Equations in Action*. 3rd ed. Springer Cham, 2016. DOI: 10.1007/978-3-319-31238-5.

[23] P. Schmid and J. Sesterhenn. "Dynamic Mode Decomposition of Numerical and Experimental Data". In: *61st Annual Meeting of the APS Division of Fluid Dynamics*. Vol. 53. 15. San Antonio, Texas: American Physical Society, Nov. 2008.

[24] F. Tröltzsch. *Optimal Control of Partial Differential Equations: Theory, Methods and Applications*. Vol. 112. Graduate studies in mathematics Applied mathematics. American Mathematical Society, 2010. DOI: 10.1090/gsm/112.

[25] J. H. Tu et al. "On Dynamic Mode Decomposition: Theory and Applications". In: *Journal of Computational Dynamics* 1.2 (2014), pp. 391–421. DOI: 10.3934/jcd.2014.1.391.

[26] C. F. Van Loan. "The Ubiquitous Kronecker Product". In: *Journal of Computational and Applied Mathematics* 123.1 (2000), pp. 85–100. DOI: 10.1016/S0377-0427(00)00393-9.

[27] A. Viguerie et al. "Coupled and Uncoupled Dynamic Mode Decomposition in Multi-Compartmental Systems with Applications to Epidemiological and Additive Manufacturing Problems". In: *Computer Methods in Applied Mechanics and Engineering* 391 (2022). DOI: 10.1016/j.cma.2022.114600.