Model Order Reduction by Proper Orthogonal Decomposition

Michael Hinze
Stefan Volkwein

Konstanzer Schriften in Mathematik
Nr. 380, Februar 2019
ISSN 1430-3558
Model Order Reduction by Proper Orthogonal Decomposition

Received February 5, 2019; accepted February 5, 2019

Abstract: We provide an introduction to POD-MOR with focus on (nonlinear) parametric PDEs and (nonlinear) time-dependent PDEs, and PDE constrained optimization with POD surrogate models as application. We cover the relation of POD and SVD, POD from the $\infty$-dimensional perspective, Reduction of nonlinearities, Certification with a priori and a posteriori error estimates, Spatial and temporal adaptivity, Input dependency of the POD surrogate model, POD basis update strategies in optimal control with surrogate models, and sketch related algorithmic frameworks. The perspective of the method is demonstrated with several numerical examples.

Keywords: POD model order reduction, (discrete) empirical interpolation, adaptivity, parametric PDEs, evolutionary PDEs, certification with error analysis

MSC 2010: 35B30, 37M99, 41A05, 65K99, 93A15, 93C05

Communicated by: February 5, 2019

1 Introduction

Proper orthogonal decomposition (POD) is a method which comprises the essential information contained in data sets. Data sets may have their origin in various sources, like, e.g., (uncertain) measurements of geophysical processes, numerical simulations of (parameter-dependent) complex physical problems, or (dynamical) imaging. In order to illustrate the POD-idea of information extraction let $\{y_1, \ldots, y_n\} \subset \mathbb{R}^m$ denote a vector cloud (which here serves as our data set), where we suppose at least one of the vectors $y_j$ is nonzero. Let us collect the vectors $y_j$'s in the data matrix

$$Y = [y_1 \mid \ldots \mid y_n] \in \mathbb{R}^{m \times n}.$$
Then we have \( r = \text{rank} \, Y \in \{1, \ldots, \min(m, n)\} \). Our aim now is to find a unit vector \( \tilde{\psi} \in \mathbb{R}^m \) which carries as much information of this vector cloud as possible. Of course we here have to specify what \textit{information} in this context means. For this purpose we equip \( \mathbb{R}^m \) with some inner product \( \langle \cdot, \cdot \rangle \) and induced norm \( \| \cdot \| \).

We define the information content of vector \( y_\ell \) with respect to some unit vector \( \bar{\psi} \) by the quantity 
\[
| \langle y, \bar{\psi} \rangle |_{\mathbb{R}^m}.
\]
Then we determine the special vector \( \tilde{\psi} \in \mathbb{R}^m \) by solving the maximization problem
\[
\tilde{\psi} = \arg \max \left\{ \sum_{j=1}^n | \langle y_j, \psi \rangle |^2 \mid \psi \in \mathbb{R}^m \text{ with } \| \psi \| = 1 \right\}.
\]

Let us label the vector \( \tilde{\psi} \) by \( \psi_1 \). We now iterate this procedure. Suppose that for \( 2 \leq \ell \leq r \) we have already computed the \( \ell - 1 \) orthonormal vectors \( \{\psi_i\}_{i=1}^{\ell-1} \) seek a unit vector \( \psi_\ell \in \mathbb{R}^m \) which is perpendicular to the \( (\ell - 1) \)-dimensional subspace
\[
\mathcal{V}_{\ell-1} = \text{span} \{\psi_1, \ldots, \psi_{\ell-1}\} \subset \mathbb{R}^m
\]
and carries as much information of our vector cloud as possible, i.e. satisfies
\[
\psi_\ell = \arg \max \left\{ \sum_{j=1}^n | \langle y_j, \psi \rangle |^2 \mid \psi \in \mathbb{R}^m \text{ with } \| \psi \| = 1 \text{ and } \psi \perp \mathcal{V}_{\ell-1} \right\}.
\]

It is now straightforward to see that the vectors \( \{\psi_i\}_{i=1}^\ell \) solve the eigenvalue problem (cf. [38, 49])
\[
\tilde{Y} \tilde{Y}^\top \psi_i = \lambda_i \psi_i, \quad i = 1, \ldots, d \text{ and } \lambda_1 \geq \ldots \geq \lambda_r > 0,
\]
where \( \tilde{Y} = W^{1/2}Y \in \mathbb{R}^{m \times n} \) with the symmetric, positive definite (weighting) matrix
\[
W = \left( ((e_i, e_j)) \right)_{1 \leq i, j \leq m}.
\]
(1)

Here, \( e_i \) denotes the \( i \)-th unit vector in \( \mathbb{R}^m \). The modes \( \{\psi_i\}_{i=1}^\ell \) obtained in this way are called \textit{POD Modes} or \textit{Principal Components} of our data cloud.

If now \( m \gg n \) it is advantageous to consider the eigenvalue problem
\[
\tilde{Y}^\top \tilde{Y} \phi_i = \lambda_i \phi_i, \quad i = 1, \ldots, d \text{ and } \lambda_1 \geq \ldots \geq \lambda_r > 0,
\]
which admits the same eigenvalues \( \lambda_i \) as before. The modes \( \psi_i \) and \( \phi_i \), \( i = 1, \ldots, r \), are related by \textit{singular value decomposition} (SVD):
\[
\psi_i = \frac{1}{\sigma_i} Y \phi_i, \quad i = 1, \ldots, r,
\]
and \( \sigma_i = \sqrt{\lambda_i} > 0 \) is the \( i \)-th singular value of the weighted data matrix \( \tilde{Y} \).
It is now clear that a vector cloud also could be replaced by a function cloud \( \{ y(\mu_j) \mid j = 1, \ldots, n \} \) in some Hilbert space \((X, \langle \cdot, \cdot \rangle_X)\), where \( \{ \mu_j \}_{j=1}^n \) are parameters which may refer to, e.g., time instances of a dynamic process, or stochastic variables, and the concept of information extraction by the above maximization problems directly carries over to this situation. As it is shown in the next section we can even extend this concept to general Hilbert spaces. This will be formalized in Section 2.1 below. From the considerations above it also becomes clear that POD is closely related to SVD. This is outlined in Section 2.2. The POD method for abstract nonlinear evolution problems is explained in Section 2.3. The Hilbert space perspective also allows us to treat spatially discrete evolution equations, which include adaptive concepts for the spatial discretization. This is outlined in Section 2.4. We support our analytical exposition by several numerical experiments which give an impression of the power of the approach.

POD is one of the most successfully used model reduction technique for nonlinear dynamical systems; see, e.g., [21, 38, 49, 70, 85] and the references therein. It is applied in a variety of fields including fluid dynamics, coherent structures [4, 9] and inverse problems [12]. Moreover in [11] POD is successfully applied to compute reduced-order controllers. The relationship between POD and balancing was considered in [57, 77, 94]. An error analysis for nonlinear dynamical systems in finite dimensions were carried out in [73] and a missing point estimation in models described by POD was studied in [10].

2 Proper Orthogonal Decomposition (POD)

In this section we introduce a discrete variant of the POD method, where we follow partially [38, Section 1.2.1]. For a continuous variant of the POD method and its relationship to the discrete one we refer the reader to [54] and [38, Sections 1.2.2 and 1.2.3].

2.1 The POD method

Suppose that \( K, n_1, \ldots, n_K \) are fixed natural numbers. Let the so-called snapshot ensembles \( \{ y_j^k \}_{j=1}^{n_k} \subset X \) be given for \( 1 \leq k \leq K \), where \( X \) is a separable real Hilbert space. We set \( n = n_1 + \ldots + n_K \). To avoid a trivial case we suppose that at least one of the \( y_j^k \)'s is nonzero. Then we introduce the finite dimensional,
linear snapshot space

\[ V = \text{span}\{y_j^k \mid 1 \leq j \leq n_k \text{ and } 1 \leq k \leq K\} \subset X \]  \hspace{1cm} (2)

with finite dimension \(d \leq n\). We distinguish two cases:

1) The separable Hilbert space \(X\) has finite dimension \(m\): Then \(X\) is isomorphic to \(\mathbb{R}^m\); see, e.g., [76, p. 47]. We define the finite index set \(I = \{1, \ldots, m\}\). Clearly, we have \(1 \leq r \leq \min(n, m)\). Especially in case of \(X = \mathbb{R}^m\), the snapshots \(y_j^k = (y_{ij}^k)_{1 \leq i \leq m}\) are vectors in \(\mathbb{R}^m\) for \(k = 1, \ldots, K\).

2) \(X\) is infinite-dimensional: Since \(X\) is separable, each orthonormal basis of \(X\) has countably many elements. In this case \(X\) is isomorphic to the set \(\ell_2\) of sequences \(\{x_i\}_{i \in \mathbb{N}}\) of complex numbers which satisfy \(\sum_{i=1}^{\infty} |x_i|^2 < \infty\); see [76, p. 47], for instance. The index set \(I\) is now the countable, but infinite set \(\mathbb{N}\).

The POD method consists in choosing a complete orthonormal basis \(\{\psi_i\}_{i \in I}\) in \(X\) such that for every \(\ell \in \{1, \ldots, r\}\) the information content of the given snapshots \(y_j^k\) is maximized in the following sense:

\[
\begin{cases}
\max_{\ell} \sum_{i=1}^{\ell} \sum_{k=1}^{K} \sum_{j=1}^{n_k} \alpha_j^k \left| \langle y_j^k, \psi_i \rangle_X \right|^2 \\
\text{s.t. } \{\psi_i\}_{i=1}^{\ell} \subset X \text{ and } \langle \psi_i, \psi_j \rangle_X = \delta_{ij}, \ 1 \leq i, j \leq \ell
\end{cases}
\]  \hspace{1cm} (P\ell)

with positive weighting parameters \(\alpha_j^k, j = 1, \ldots, n_k\) and \(k = 1, \ldots, K\). Here, the symbol \(\delta_{ij}\) denotes the Kronecker symbol satisfying \(\delta_{ii} = 1\) and \(\delta_{ij} = 0\) for \(i \neq j\).

An optimal solution \(\{\Psi_i\}_{i=1}^{\ell}\) to (P\ell) is called a POD basis of rank \(\ell\). It is proved in [38, Theorem 1.8] that for every \(\ell \in \{1, \ldots, r\}\) a solution \(\{\Psi_i\}_{i=1}^{\ell}\) to (P\ell) is characterized by the eigenvalue problem

\[ \mathcal{R}\Psi_i = \lambda_i \Psi_i \quad \text{for } 1 \leq i \leq \ell, \]  \hspace{1cm} (3)

where \(\lambda_1 \geq \ldots \geq \lambda_r > 0\) denote the largest eigenvalues of the linear, bounded, nonnegative and self-adjoint operator \(\mathcal{R} : X \to X\) given as

\[ \mathcal{R}\Psi = \sum_{k=1}^{K} \sum_{j=1}^{n_k} \alpha_j^k \langle \Psi, y_j^k \rangle_X y_j^k \quad \text{for } \Psi \in X. \]  \hspace{1cm} (4)

Moreover, the operator \(\mathcal{R}\) can be presented in the form

\[ \mathcal{R} = \mathcal{Y}\mathcal{Y}^* \]  \hspace{1cm} (5)
with the mapping
\[ \mathcal{Y} : \mathbb{R}^n \to X, \quad \mathcal{Y}(\Phi) = \sum_{k=1}^{K} \sum_{j=1}^{n_k} \sqrt{\alpha_j^k} \phi_j^k y_j^k \quad \text{for} \quad \Phi = (\phi_1^1, \ldots, \phi_K^{n_K}) \in \mathbb{R}^n, \]
where \( \mathcal{Y}^* : X \to \mathbb{R}^n \) denotes the Hilbert space adjoint of \( \mathcal{Y} \), whose action is given by
\[ \mathcal{Y}^* (\Psi) = \left( \left\langle \Psi, \sqrt{\alpha_1^1 y_1^1} \right\rangle_X, \ldots, \left\langle \Psi, \sqrt{\alpha_K^{n_K} y_K^{n_K}} \right\rangle_X \right)^\top \quad \text{for} \quad \Psi \in X. \]

The operator \( \mathcal{K} : \mathbb{R}^n \to \mathbb{R}^n, \quad \mathcal{K} := \mathcal{Y}^* \mathcal{Y} \) then admits the same nonzero eigenvalues \( \lambda_1 \geq \ldots \geq \lambda_r > 0 \) with corresponding eigenvectors \( \Phi_1, \ldots, \Phi_r \), and its action is given by
\[ \mathcal{K} \Phi = \sum_{k=1}^{K} \sum_{j=1}^{n_k} \left( \sqrt{\alpha_1^1 \alpha_j^k \langle y_j^k, y_1^1 \rangle_X}, \ldots, \sqrt{\alpha_K^{n_K} \alpha_j^k \langle y_j^k, y_K^{n_K} \rangle_X} \right)^\top \quad \text{(6)} \]
with the vector \( \Phi = (\phi_1^1, \ldots, \phi_K^{n_K}) \in \mathbb{R}^n \). For the eigensystems of \( \mathcal{R} \) and \( \mathcal{K} \) there holds the relation
\[ \Phi_i = \frac{1}{\sqrt{\lambda_i}} \mathcal{Y}^* \Psi_i, \quad \text{and} \quad \Psi_i = \frac{1}{\sqrt{\lambda_i}} \mathcal{Y} \Phi_i, \quad \text{for} \quad i = 1, \ldots, r. \quad \text{(7)} \]
Furthermore, we obtain
\[ \sum_{i=1}^{\ell} \sum_{k=1}^{K} \sum_{j=1}^{n_k} \alpha_j^k \left| \langle y_j^k, \Psi_i \rangle_X \right|^2 = \sum_{i=1}^{\ell} \lambda_i, \]
and for the POD projection error we get
\[ \sum_{k=1}^{K} \sum_{j=1}^{n_k} \alpha_j^k \left\| y_j^k - \sum_{i=1}^{\ell} \sum_{k=1}^{K} \sum_{j=1}^{n_k} \langle y_j^k, \Psi_i \rangle_X \Psi_i \right\|_X^2 = \sum_{i=\ell+1}^{r} \lambda_i. \quad \text{(8)} \]
Thus, the decay of the positive eigenvalues \( \{\lambda_i\}_{i=1}^r \) plays an essential role for a successful application of the POD method. In general, one has to utilize the complete orthonormal basis \( \{\Psi_i\}_{i \in I} \subset X \) to represent elements in the snapshot space \( \mathcal{V} \) by their Fourier sum. This leads to a high-dimensional or even infinite-dimensional approximation scheme. Nevertheless, if the term \( \sum_{i=\ell+1}^{r} \lambda_i \) is sufficiently small for a not too large \( \ell \), elements in the subspace \( \mathcal{V} \) can be approximated by a linear combination of the few basis elements \( \{\Psi_i\}_{i=1}^{\ell} \). This offers the chance to reduce the number of terms in the Fourier series using the
POD basis of rank \( \ell \), as shown in the following examples. For this reason it is useful to define information content of the basis \( \{ \Psi_i \}_{i=1}^{\ell} \) in \( V \) by the quantity

\[
\mathcal{E}(\ell) = \frac{\sum_{i=1}^{\ell} \lambda_i}{\sum_{i=1}^{r} \lambda_i} \in [0, 1].
\]

(9)

It can e.g. be utilized to determine a basis of length \( \ell \in \{1, \ldots, r\} \) containing \( \approx 99\% \) of the information contained in \( V \) by requiring \( \mathcal{E}(\ell) \approx 99\% \). Now it is shown in [38, Section 1.2.1] that

\[
\sum_{i=1}^{r} \lambda_i = \sum_{k=1}^{K} \sum_{j=1}^{n_{k_j}} \alpha_j^k \left\| y_j^k \right\|_X^2
\]

holds true. This implies

\[
\mathcal{E}(\ell) = \frac{\sum_{i=1}^{\ell} \lambda_i}{\sum_{k=1}^{K} \sum_{j=1}^{n_{k_j}} \alpha_j^k \left\| y_j^k \right\|_X^2} \in [0, 1],
\]

so that the quantity \( \mathcal{E}(\ell) \) can be computed without knowing the eigenvalues \( \lambda_{\ell+1}, \ldots, \lambda_r \).

### 2.2 Singular value decomposition and POD

To investigate the relationship between singular value decomposition (SVD) and POD let us discuss the POD method for the specific case \( X = \mathbb{R}^m \). Then we define the matrices

\[
D^k = \begin{pmatrix} 
\alpha_1^k & 0 \\
\vdots & \ddots \\
0 & \alpha_{n_k}^k
\end{pmatrix} \in \mathbb{R}^{n_k \times n_k} \quad \text{for } 1 \leq k \leq K,
\]

\[
D = \begin{pmatrix} 
D^1 & 0 \\
\vdots & \ddots \\
0 & D^K
\end{pmatrix} \in \mathbb{R}^{n \times n},
\]

\[
Y^k = \begin{bmatrix} 
y_1^k | \ldots | y_{n_{k_j}}^k
\end{bmatrix} \in \mathbb{R}^{m \times n_k} \quad \text{for } 1 \leq k \leq K,
\]

\[
Y = \begin{bmatrix} 
Y^1 | \ldots | Y^K
\end{bmatrix} \in \mathbb{R}^{m \times n}, \quad \bar{Y} = W^{1/2}YD^{1/2} \in \mathbb{R}^{m \times m},
\]

where we have introduced the weighting matrix \( W \in \mathbb{R}^{m \times m} \) in (1).

**Remark 1.** Let us mention that \( \bar{Y} = Y \) holds true provided all \( \alpha_j^k \) are equal to one (i.e., \( D \) is the identity matrix) and the inner product in \( X \) is given by the Euclidean inner product (i.e., \( W \) is the identity matrix).  \( \diamond \)
Now (3) is equivalent to the $m \times m$ eigenvalue problem
\[ \tilde{Y} \tilde{Y}^\top \Psi_i = \lambda_i \Psi_i \quad \text{for } 1 \leq i \leq \ell \] (10)
with $\Psi_i = W^{-1/2} \bar{\Psi}_i$ and the $n \times n$ eigenvalue problem
\[ \tilde{Y}^\top \tilde{Y} \Phi_i = \lambda_i \Phi_i \quad \text{for } 1 \leq i \leq \ell \] (11)
with $\Phi_i = YD^{1/2} \bar{\Phi}_i / \sqrt{\lambda_i}$. If $m \gg n$ holds, we solve (10). However, we have to solve the linear system $W^{1/2} \Psi_i = \bar{\Psi}_i$ for any $i = 1, \ldots, \ell$ in order to get the POD basis $\{ \Psi_i \}_{i=1}^\ell$. Thus, if $n \leq m$ holds, we will compute the solution $\{ \Phi_i \}_{i=1}^\ell$ to (11) and get the POD basis by the formula $\Psi_i = YD^{1/2} \bar{\Phi}_i / \sqrt{\lambda_i}$. In that case we also have $\tilde{Y}^\top \tilde{Y} = Y^\top W Y$ so that we do not have to compute the square root matrix $W^{1/2}$. On the other hand, the diagonal matrix $D^{1/2}$ can be computed easily. The relationship between (10) and (11) is given by SVD: There exist real numbers $\sigma_1 \geq \ldots \geq \sigma_r > 0$ and orthogonal matrices $\Psi \in \mathbb{R}^{m \times m}$, $\Phi \in \mathbb{R}^{n \times n}$ with column vectors $\{ \bar{\Psi}_i \}_{i=1}^m$, $\{ \bar{\Phi}_i \}_{i=1}^n$ such that
\[ \Psi^\top \tilde{Y} \Phi = \begin{pmatrix} \Sigma' & 0 \\ 0 & 0 \end{pmatrix} =: \Sigma \in \mathbb{R}^{m \times n}, \] (12)
where $\Sigma' = \text{diag}(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r \times r}$ and the zeros in (12) denote matrices of appropriate dimensions. Moreover, the vectors $\{ \bar{\Psi}_i \}_{i=1}^r$ and $\{ \bar{\Phi}_i \}_{i=1}^r$ are eigenvectors of $\tilde{Y} \tilde{Y}^\top$ and $\tilde{Y}^\top \tilde{Y}$, respectively, with eigenvalues $\lambda_i = (\sigma_i)^2 > 0$ for $i = 1, \ldots, r$. The vectors $\{ \bar{\Psi}_i \}_{i=r+1}^m$ and $\{ \bar{\Phi}_i \}_{i=r+1}^n$ (if $r < m$ respectively $r < n$) are eigenvectors of $\tilde{Y} \tilde{Y}^\top$ and $\tilde{Y}^\top \tilde{Y}$ with eigenvalue 0. We summarize the computation of the POD basis in the pseudo code function $[\Psi, \Lambda] = \text{POD}(Y, W, D, \ell, \text{flag})$. 
function \( [\Psi, \Lambda] = \text{POD}(Y, W, D, \ell, \text{flag}) \)

**Require:** Snapshots matrix \( Y = [Y^1, \ldots, Y^K] \) with rank \( r \), weighting matrices \( W, D \), number \( \ell \) of POD functions and flag for the solver;

1: if \( \text{flag} = 0 \) then
2: Set \( \bar{Y} = W^{1/2} Y D^{1/2} \);
3: Compute singular value decomposition \( [\Psi, \Sigma, \Phi] = \text{svd}(\bar{Y}) \);
4: Define \( \bar{\Psi}_i \) as the \( i \)-th column of \( \Psi \) and \( \sigma_i = \Sigma_{ii} \) for \( 1 \leq i \leq \ell \);
5: Set \( \Psi_i = W^{-1/2} \bar{\Psi}_i \) and \( \lambda_i = \sigma_i^2 \) for \( i = 1, \ldots, \ell \);
6: else if \( \text{flag} = 1 \) then
7: Compute eigenvalue decomposition \( [\Psi, \Lambda] = \text{eig}(\bar{Y} \bar{Y}^T) \);
8: Define \( \bar{\Psi}_i \) as the \( i \)-th column of \( \Psi \) and \( \lambda_i = \Lambda_{ii} \) for \( 1 \leq i \leq \ell \);
9: Set \( \Psi_i = W^{-1/2} \bar{\Psi}_i \) for \( i = 1, \ldots, \ell \);
10: else if \( \text{flag} = 2 \) then
11: Compute eigenvalue decomposition \( [\Phi, \Lambda] = \text{eig}(\bar{Y}^T \bar{Y}) \);
12: Define \( \bar{\Phi}_i \) as the \( i \)-th column of \( \Phi \) and \( \lambda_i = \Lambda_{ii} \) for \( 1 \leq i \leq \ell \);
13: Set \( \Psi_i = Y D^{1/2} \bar{\Phi}_i / \sqrt{\lambda_i} \) for \( i = 1, \ldots, \ell \);
14: end if
15: return \( \Psi = [\Psi_1|\ldots|\Psi_\ell] \) and \( \Lambda = [\lambda_1|\ldots|\lambda_\ell] \);

### 2.3 The POD method for nonlinear evolution problems

In this subsection we explain the POD method for abstract nonlinear evolution problems. We focus on the numerical realization. For detailed theoretical investigations we refer the reader to \([38, 46, 47, 53, 54]\); for instance.

#### 2.3.1 The nonlinear evolution problems

Let us formulate the nonlinear evolution problem. For that purpose we suppose the following hypotheses.

**Assumption 1.** Suppose that \( T > 0 \) holds.

1) \( V \) and \( H \) are real, separable Hilbert spaces and suppose that \( V \) is dense in \( H \) with compact embedding. By \( \langle \cdot, \cdot \rangle_H \) and \( \langle \cdot, \cdot \rangle_V \) we denote the inner products in \( H \) and \( V \), respectively. We identify \( H \) with its dual (Hilbert) space \( H' \) by the Riesz isomorphism so that we have the Gelfand triple

\[
V \hookrightarrow H \simeq H' \hookrightarrow V',
\]
where each embedding is continuous and dense. The last embedding is understood as follows: For every element \( h \in H' \) and \( v \in V \), we also have \( v \in H \) by the embedding \( V \hookrightarrow H \), so we can define \( \langle h', v \rangle_{V',V} = \langle h', v \rangle_{H',H} \).

2) \( U \) is a real Hilbert space with inner product \( \langle \cdot, \cdot \rangle_U \) and induced norm \( \| \cdot \|_U = \langle \cdot, \cdot \rangle_U^{1/2} \). Furthermore, \( U_{ad} \) is a bounded, closed and convex subset of \( U \).

3) For almost all \( t \in [0, T] \) we define a time-dependent bilinear form \( a(t; \cdot, \cdot) : V \times V \rightarrow \mathbb{R} \) satisfying

\[
|a(t; \varphi, \phi)| \leq \gamma \| \varphi \|_V \| \phi \|_V \quad \text{for all } \varphi, \phi \in V, \ t \in [0, T] \ a.e., \quad (13a)
\]

\[
a(t; \varphi, \varphi) \geq \gamma_1 \| \varphi \|_V^2 - \gamma_2 \| \varphi \|_H^2 \quad \text{for all } \varphi \in V, \ t \in [0, T] \ a.e., \quad (13b)
\]

for time-independent constant \( \gamma, \gamma_2 \geq 0, \gamma_1 > 0 \) and where „a.e.“ stands for „almost everywhere“.

4) Assume that \( y_0 \in V, \ f \in L^2(0, T; H) \) and that \( \mathcal{B} : U \rightarrow L^2(0, T; H) \) is a continuous, linear operator. Here we refer to [25, pp. 469-472] for vector-valued function spaces.

Recall the function space

\[ W(0, T) = \{ \varphi \in L^2(0, T; V) \mid \varphi_t \in L^2(0, T; V') \} \]

which is a Hilbert space endowed with the standard inner product; cf. [25, pp. 472-479]. Furthermore, we have

\[
\frac{d}{dt} \langle \varphi(t), \phi \rangle_H = \langle \varphi_t(t), \phi \rangle_{V',V} \quad \text{for } \varphi \in W(0, T), \ \phi \in V
\]

in the sense of distributions in \([0, T]\). Here, \( \langle \cdot, \cdot \rangle_{V',V} \) stands for the dual pairing between \( V \) and its dual \( V' \).

Now the evolution problem is given as follows: For fixed input \( u \in U_{ad} \) find the state \( y \in W(0, T) \cap C([0, T]; V) \) such that

\[
\frac{d}{dt} \langle y(t), \varphi \rangle_H + a(t; y(t), \varphi) + \langle \mathcal{N}(y(t)), \varphi \rangle_{V',V} = \langle (f + \mathcal{B}u)(t), \varphi \rangle_H \quad \forall \varphi \in V, \ t \in (0, T] \ a.e., \quad (14)
\]

\[
\langle y(0), \varphi \rangle_H = \langle y_0, \varphi \rangle_H \quad \forall \varphi \in H.
\]

Throughout we assume that (14) admits a unique solution \( y \in W(0, T) \cap C([0, T]; V) \). Of course, this requires some properties for the nonlinear mapping \( \mathcal{N} \) which we will not specify here.

**Example 2** (Semilinear heat equation). Let \( \Omega \subset \mathbb{R}^d, \ d \in \{2, 3\} \) be a bounded open domain with Lipschitz continuous boundary \( \partial \Omega \) and let \( T > 0 \) be a fixed end
time. We set \( Q := (0, T) \times \Omega \) and \( \Sigma := (0, T) \times \partial \Omega \) and \( c \geq 0 \). For a given forcing term \( f \in L^2(Q) \) and initial condition \( y_0 \in L^2(\Omega) \), we consider the semilinear heat equation with homogeneous Dirichlet boundary condition:

\[
\begin{aligned}
& y_t(t, x) - \Delta y(t, x) + cy^3(t, x) = f(t, x) \quad \text{in } Q, \\
& y(t, x) = 0 \quad \text{on } \Sigma, \\
& y(0, x) = y_0(x) \quad \text{in } \Omega.
\end{aligned}
\tag{15}
\]

The existence of a unique solution to (15) is proved in [78], for example. We can write (15) as an abstract evolution problem of type (14) by deriving a variational formulation for (15) with \( V = H^1_0(\Omega) \) as the space of test functions, \( H = L^2(\Omega) \) and integrating over the space \( \Omega \). The bilinear form \( a : V \times V \to \mathbb{R} \) is introduced by

\[
a(\varphi, \phi) = \int_{\Omega} \nabla \varphi \cdot \nabla \phi \, dx \quad \text{for } \varphi, \phi \in V
\]

and the operator \( \mathcal{N} : V \to V' \) is defined as \( \mathcal{N}(\varphi) = \varphi^3 \) for \( \varphi \in V \). For \( c \equiv 0 \), the heat equation (15) is linear.

**Example 3** (Cahn-Hilliard equations). Let \( \Omega, T, Q \) and \( \Sigma \) be defined as in Example 2. The Cahn-Hilliard system was proposed in [19] as a model for phase separation in binary alloys. Introducing the chemical potential \( w \), the Cahn-Hilliard equations can be formulated in the common setting as a coupled system for the phase field \( c \) and the chemical potential \( w \):

\[
\begin{aligned}
c_t(t, x) + y \cdot \nabla c(t, x) &= m \Delta w(t, x) \quad \text{in } Q, \\
w(t, x) &= -\sigma \varepsilon \Delta c(t, x) + \frac{\sigma}{\varepsilon} W'(c(t, x)) \quad \text{in } Q, \\
\nabla c(t, x) \cdot \nu_{\Omega} &= \nabla w(t, x) \cdot \nu_{\Omega} = 0 \quad \text{on } \Sigma, \\
c(0, x) &= c_0(x) \quad \text{in } \Omega.
\end{aligned}
\tag{16}
\]

By \( \nu_{\Omega} \) we denote the outward normal on \( \partial \Omega \), \( m \geq 0 \) is a constant mobility, \( \sigma > 0 \) denotes the surface tension and \( 0 < \varepsilon \ll 1 \) represents the interface parameter. Note that the convective term \( y \cdot \nabla c \) describes the transport with (constant) velocity \( y \). The transport term represents the coupling to the Navier-Stokes equations in the context of multiphase flow, see e.g. [48] and [2]. The phase field function \( c \) describes the phase of a binary material with components \( A \) and \( B \) and takes the values \( c \equiv -1 \) in the pure \( A \)-phase and \( c \equiv +1 \) in the pure \( B \)-phase. The interfacial region is described by \( c \in (-1, 1) \) and admits a thickness of order \( O(\varepsilon) \). The function \( W(c) \) represents the free energy and is of double well-type. A typical choice for \( W \) is the polynomial free energy function

\[
W^p(c) = (1 - c^2)^2 / 4
\]

(17)
with two minima at $c = \pm 1$, which describe the energetically favorable states. It is infinitely often differentiable. Another choice for $W$ is the $C^1$ relaxed double obstacle free energy

$$W^\text{rel}_s(c) = \frac{1}{2}(1 - c^2) + \frac{s}{2}(\max(c - 1, 0)^2 + \min(c + 1, 0)^2),$$

with relaxation parameter $s \gg 0$, which is introduced in [39] as the Moreau-Yosida relaxation of the double obstacle free energy

$$W^\infty(c) = \begin{cases} 
\frac{1}{2}(1 - c^2), & \text{if } c \in [-1, 1], \\
+\infty, & \text{otherwise.}
\end{cases}$$

The energies $W^p(c)$ and $W^\text{rel}_s(c)$ later will be used to compare the performance of POD on systems with smooth and less smooth nonlinearities. For more details on the choices for $W$ we refer to [1] and [17], for example. Concerning existence, uniqueness and regularity of a solution to (16), we refer to [17]. In order to derive a variational form of type (14), we rewrite (16) as a single fourth-order parabolic equation for $c$ by

$$c_t(t, x) + y \cdot \nabla c(t, x) = m\Delta \left( -\sigma \varepsilon \Delta c(t, x) + \frac{\sigma W'(c(t, x))}{\varepsilon} \right) \quad \text{in } Q,$$

$$0 = \nabla c(t, x) \cdot \nu_\Omega = \nabla \left( -\sigma \varepsilon \Delta c(t, x) + \frac{\sigma W'(c(t, x))}{\varepsilon} \right) \cdot \nu_\Omega \quad \text{on } \Sigma,$$

$$c(0, x) = c_0(x) \quad \text{in } \Omega.$$  

We choose $V = H^2(\Omega) \cap H^1_0(\Omega)$, $H = L^2(\Omega)$ and introduce the bilinear form

$$a(\varphi, \phi) = \int_\Omega m\sigma \varepsilon \Delta \varphi \cdot \Delta \phi + y \cdot \nabla \varphi \cdot \phi \, dx \quad \text{for } \varphi, \phi \in V$$

Moreover, we define the nonlinear operator $\mathcal{N}$ by $\mathcal{N}(c) = -\sigma \Delta W'(c)/\varepsilon$. 

Further examples include the two-dimensional incompressible Navier-Stokes equations, see e.g. [87, p. 111].

2.3.2 Temporal discretization and POD method

Let $0 = t_1 < \ldots < t_{n_t} = T$ be a given time grid with step sizes $\delta t_j = t_j - t_{j-1}$ for $j = 2, \ldots, n_t$. Suppose that for any $j \in \{1, \ldots, n_t\}$ the elements $y_j \in V \subset H$ is an approximation of $y(t_j)$ computed by applying a temporal integration method (e.g., the implicit Euler method) to (14). Then we consider the snapshot ensemble

$$V = \text{span} \{y_j \mid 1 \leq j \leq n\} \subset V \subset H$$
with \( n = n_t \) and \( r = \dim V \leq n \). In the context of \((P^\ell)\) we choose \( K = 1 \) and \( n = n_1 = n_t \). Moreover, \( X \) can be either \( V \) or \( H \). For the weighting parameters we take the trapezoidal weights

\[
\alpha_1 = \frac{\delta t_1}{2}, \quad \alpha_j = \frac{\delta t_j + \delta t_{j-1}}{2} \quad \text{for } j = 2, \ldots, n_t - 1, \quad \alpha_{n_t} = \frac{\delta t_{n_t}}{2}. \tag{20}
\]

Of course, other quadrature weights are also possible. Now, instead of \((P^\ell)\) we consider the minimization problem

\[
\begin{align*}
\max_{\ell} & \sum_{i=1}^{\ell} \sum_{j=1}^{n} \alpha_j \left| \langle y_j, \Psi_i \rangle_X \right|^2 \\
\text{s.t.} & \{\Psi_i\}_{i=1}^{\ell} \subset X \text{ and } \langle \Psi_i, \Psi_j \rangle_X = \delta_{ij}, \ 1 \leq i, j \leq \ell
\end{align*}
\tag{21}
\]

with either \( X = V \) or \( X = H \).

**Remark 4.** In [38, Sections 1.2.2 and 1.3.2] a continuous variant of the POD method is considered. In that case the trapezoidal approximation in (21) is replaced by integrals over the time interval \([0, T]\). More precisely, we consider

\[
\begin{align*}
\max_{\ell} & \sum_{i=1}^{\ell} \int_0^T \left| \langle y(t), \Psi_i \rangle_X \right|^2 dt \\
\text{s.t.} & \{\Psi_i\}_{i=1}^{\ell} \subset X \text{ and } \langle \Psi_i, \Psi_j \rangle_X = \delta_{ij}, \ 1 \leq i, j \leq \ell
\end{align*}
\tag{22}
\]

with either \( X = V \) or \( X = H \). The relationship between solutions to (21) and to (22) we refer to [54] and [38, Section 1.2.3].

To compute the POD basis \( \{\Psi_i\}_{i=1}^{\ell} \) of rank \( \ell \) we have to evaluate the inner products \( \langle y_j, \Psi_i \rangle_X \), where either \( X = V \) or \( X = H \) hold. In typical applications the space \( X \) is usually infinite dimensional. Therefore, a discretization of \( X \) is required in order to get a POD method that can be realized on a computer. This is the topic of the next subsection.

### 2.3.3 Galerkin discretization

We discretize the state equation by applying any spatial approximation scheme. Let us consider here a Galerkin scheme for (14). For this reason we are given linearly independent elements \( \varphi_1, \ldots, \varphi_m \in V \) and define the \( m \)-dimensional subspace

\[
V^h = \text{span} \{ \varphi_1, \ldots, \varphi_m \} \subset V
\]
endowed with the $V$ topology. Then a Galerkin scheme for (14) is given as follows: find $y^h \in W(0, T) \cap C([0, T]; V^h)$ satisfying
\[
\frac{d}{dt} \langle y^h(t), \varphi_h \rangle_H + a(t; y^h(t), \varphi^h) + \langle N(y^h(t)), \varphi \rangle_{V', V}
= \langle (f + Bu)(t), \varphi^h \rangle_H \quad \forall \varphi^h \in V^h, \ t \in (0, T) \text{ a.e.,}
\]
\[
\langle y^h(0), \varphi^h \rangle_H = \langle y_0, \varphi^h \rangle_H \quad \forall \varphi^h \in V^h.
\]
Inserting the representation $y^h(t) = \sum_{i=1}^{m} v_i(t) \varphi_i$ in (23) and choosing $\varphi^h = \varphi_i$ for $i = 1, \ldots, m$ we derive the following $m$-dimensional initial value problem
\[
M^h \ddot{y}^h(t) + A^h(t)y^h(t) + N^h(y^h(t)) = F^h(t; u) \quad t \in (0, T],
\]
M^h y^h(0) = y^h_0,
\]
where we have used the matrices and vectors
\[
M^h = \left( (\langle \varphi_j, \varphi_i \rangle_H)_{1 \leq i, j \leq m} \right), \quad y^h(t) = (y^h_i)_{1 \leq i \leq m} \text{ for } t \in [0, T) \text{ a.e.,}
\]
\[
A^h(t) = \left( (a(t; \varphi_j, \varphi_i))_{1 \leq i, j \leq m} \right), \quad y^h_0 = (\langle y_0, \varphi_i \rangle_H)_{1 \leq i \leq m},
\]
\[
N^h(v) = \left( \langle N(\sum_{j=1}^{m} v_j \varphi_j), \varphi_i \rangle_{V', V} \right)_{1 \leq i \leq m} \text{ for } v = (v_j)_{1 \leq j \leq m},
\]
\[
F^h(t; u) = \left( (f + Bu)(t, \varphi_i) \right)_{1 \leq i \leq m} \text{ for } t \in [0, T) \text{ a.e. and } u \in U_{ad}.
\]

Let us introduce the matrices $A_j^h = A^h(t_j) \in \mathbb{R}^{m \times m}$ for $j = 2, \ldots, n_t$. In the pseudo code function [Y]=StateSol($u, \{t_j\}_{j=1}^{n_t}, M^h, \{A_j^h\}_{j=2}^{n_t}, \{N^h, F^h, y^h_0\}$) we present a solution method for (24) using the implicit Euler method.

```
function [Y]=StateSol($u, \{t_j\}_{j=1}^{n_t}, M^h, \{A_j^h\}_{j=2}^{n_t}, \{N^h, F^h, y^h_0\}$)

Require: input $u$, time grid $\{t_j\}_{j=1}^{n_t}$, matrices $M^h$, $\{A_j^h\}_{j=2}^{n_t}$, vectors $N^h$, $F^h$, $y^h_0$;
1: Compute $y_1^h \in \mathbb{R}^m$ solving $M^h y_1^h = y^h_0$;
2: for $j = 2$ to $n_t$ do
3: Solve (e.g., by applying Newton’s method) for $y_j^h \in \mathbb{R}^m$
\[
(M^h + \delta t_j A_j^h) y_j^h + N^h(y_j^h) = M^h y_{j-1}^h + \delta t_j F^h(u, t_j);
\]
4: end for
5: return matrix $Y = [y_1^h | \ldots | y_{n_t}^h] \in \mathbb{R}^{m \times n_t}$;
```

In the next subsection we discuss how a POD basis $\{\Psi_j\}_{j=1}^{\ell}$ of rank $\ell \leq r$ can be computed from numerical approximations for the solution $y$ to (24).
2.3.4 POD method for the fully discretized nonlinear evolution problem

Recall that we have introduced the temporal grid \( \{ t_j \}_{j=1}^{n_t} \subset [0,T] \) and set \( n = n_t \). Let \( y^h_1, \ldots, y^h_n \in V^h \) be numerical approximations to the solution \( y(t) \) to (24) at time instances \( t = t_j, j = 1 \ldots, n_t \). Then, a coefficient matrix \( Y \in \mathbb{R}^{m \times n} \) is defined by the elements \( Y_{ij} \) given by

\[
y^h_j = \sum_{i=1}^{m} Y_{ij} \varphi_i \in V^h \quad \text{for} \ 1 \leq j \leq n.
\]

The \( j \)-th column of \( Y \) (denoted by \( y_j = Y_{*,j} \)) contains the Galerkin coefficients of the snapshot \( y^h_j \in V^h \). We set \( r = \text{rank}(Y) \leq \min(m,n) \) and

\[
\mathcal{V}^h = \text{span} \{ y^h_j \mid 1 \leq j \leq n \} \subset V^h.
\]

Due to \( \mathcal{V}^h \subset V^h \) we have \( \Psi_j \in V^h \) for \( 1 \leq j \leq \ell \). Therefore, there exists a coefficient matrix \( \Psi \in \mathbb{R}^{m \times \ell} \) is defined by the elements \( \Psi_{ij} \) satisfying

\[
\Psi_j = \sum_{i=1}^{m} \Psi_{ij} \varphi_i \in V^h \quad \text{for} \ 1 \leq j \leq \ell.
\]

where the \( j \)-th column \( \Psi_{*,j} \) of the matrix \( \Psi \) consists of the Galerkin coefficients of the element \( \Psi_j \). Notice that

\[
\langle v^h, w^h \rangle_H = (v^h)^\top M^h w^h, \quad \langle v^h, w^h \rangle_H = (v^h)^\top S^h w^h
\]

hold for \( v^h = \sum_{i=1}^{m} v^h_i \varphi_i, w^h = \sum_{i=1}^{m} w^h_i \varphi_i \in V^h \) and for the symmetric, positive definite stiffness matrix

\[
S^h = \left( (\langle \varphi_j, \varphi_i \rangle_V) \right)_{1 \leq i,j \leq m}.
\]

Then, we have for \( X = H \)

\[
\langle y^h_j, \Psi_i \rangle_X = y^\top \psi \Psi_i, y^h_j = \sum_{k=1}^{m} \langle \varphi_j, \varphi_k \rangle_V \Psi_i, \quad \text{for} \ 1 \leq j \leq n, \ 1 \leq i \leq \ell
\]

and for \( X = V \)

\[
\langle y^h_j, \Psi_i \rangle_X = y^\top \psi \Psi_i, y^h_j = \sum_{k=1}^{m} \langle \varphi_j, \varphi_k \rangle_V \Psi_i, \quad \text{for} \ 1 \leq j \leq n, \ 1 \leq i \leq \ell
\]

Thus, we can apply the approach presented in Section 2.2 choosing \( W = M^h \) for \( X = H \) and \( W = S^h \) for \( X = V \). Moreover, we set \( K = 1, n_1 = n_t = n \) and \( \alpha_j = \alpha_j \) defined in (20). Now a POD basis of rank \( \ell \) for (24) can be computed by the pseudo code function \( [Y, \Psi] = \text{PODState}(u, \{ t_j \}_{j=1}^{n_t}, M^h, \{ A^h_j \}_{j=2}^{n_t}, N^h, P^h, y^h_0, W, D, \ell, \text{flag}) \).
\begin{function}
\text{function} \ [Y,\Psi] = \text{PODState}(u,\{t_j\}_{j=1}^{n_t},M^h,\{A_j^h\}_{j=2}^{n_t},N^h,F^h,y^o,W,D,\ell,\text{flag})
\end{function}

\textbf{Require:} input \(u\), time grid \(\{t_j\}_{j=1}^{n_t}\), matrices \(M^h, \{A_j^h\}_{j=2}^{n_t}\), vectors \(N^h, F^h, y^h\), weighting matrices \(W, D\), number \(\ell\) of POD functions and \text{flag} for the solver;

1: Call \([Y] = \text{StateSol}(u, \{t_j\}_{j=1}^{n_t},M^h,\{A_j^h\}_{j=2}^{n_t},N^h,F^h,y^o);\)

2: Call \([\Psi,\Lambda] = \text{POD}(Y,W,D,\ell,\text{flag});\)

3: return \(Y\) and \(\Psi\);

In the next subsection we will discuss in detail how the POD method has to be applied in that case if we have – instead of \(V^h\) – different spaces \(V^{h_j}\) for each \(j = 2,\ldots,n\).

### 2.4 The POD method with adaptively generated snapshots

In practical applications it often is desirable to provide POD models for time-dependent PDE systems, whose numerical treatment requires adaptive numerical techniques in space and/or time. Snapshots generated by those methods are not directly amenable to the POD procedure described in Section 2.3.4, since the application of e.g. spatial adaptivity means, that the snapshots at each time instance may have different lengths due to their different spatial resolutions. In fact, there is no common discrete Galerkin space \(X^h\) for all snapshots generated by the fully discrete evolution, but at every time instance \(t_j\) the adaptive procedure generates a discrete Galerkin space \(X^{h_j} \subset X\), so that in this case \(y^h_j \equiv y^{h_j} \in X^{h_j}\). For this reason, no snapshot matrix \(Y\) can be formed with columns containing the basis coefficient vectors of the snapshots.

To obtain also a POD basis in this situation we inspect the operator \(\mathcal{K}\) of (6) and observe that its action can be computed if the inner products \(\langle y^k_j, y^l_i \rangle_X\) can be evaluated for all \(1 \leq i \leq n_l, 1 \leq j \leq n_k\) and \(1 \leq k, l \leq K\).

Let us next demonstrate how to compute a POD basis for snapshots residing in arbitrary finite element spaces. To begin with we drop the superindex \(h\) and set \(X_j := X^{h_j}\). For each time instant \(j = 1,\ldots,n\) of our time discrete PDE system the snapshots \(\{y^j\}_{j=1}^{n} \) are taken from different finite element spaces \(X_j \subseteq X\) \((j = 1,\ldots,n)\) where \(X\) denotes a common Hilbert space. Let \(X_j = \text{span}\{\varphi^j_1,\ldots,\varphi^j_{m_j}\}\). Then we have the expansions

\[ y_j = \sum_{i=1}^{m_j} y^j_i \varphi^j_i \quad \text{for} \ j = 1,\ldots,n \] (25)
with coefficient vectors

\[ y_j = (y^j_i) \in \mathbb{R}^{m_j} \quad \text{for } j = 1, \ldots, n. \]

The inner product of the associated functions can thus be computed as

\[ \langle y_i, y_j \rangle_X = \sum_{k=1}^{m_i} \sum_{l=1}^{m_j} y^k_i y^l_j \langle \varphi^k_i, \varphi^l_j \rangle_X \quad \text{for } i, j = 1, \ldots, n, \]

so that the evaluation of the action \( K\Phi \) only relies on the evaluation of the inner products \( \langle \varphi^k_i, \varphi^l_j \rangle_X \) (1 ≤ i, j ≤ n, 1 ≤ k ≤ m_i, 1 ≤ l ≤ m_j). In other words, once we are able to compute those inner products we are in the position to set up the eigensystem \( \{ (\lambda_i, \Phi_i) \}_{i=1}^r \) of \( K \). The POD modes \( \{ \Psi_i \}_{i=1}^r \) can then be computed according to (7) by

\[ \Psi_i = \frac{1}{\sqrt{\lambda_i}} y^i \Phi_i \quad \text{for } i = 1, \ldots, r. \]

Details on this procedure can be found in [58, 36].

To illustrate how this procedure can be implemented we summarize Examples 6.1-6.3 from [36], which deal with \( h \)- and \( r \)-adaptivity. All coding was done by Carmen Gräßle in \text{C++} with using FEniCS [8, 62] for the solution of the differential equations and ALBERTA [82] for dealing with hierarchical meshes. The numerical tests were run on a compute server with 24 CPU kernels and 512 GB RAM.

**Run 1** ([36, Example 6.1]). We consider the Example 2 with homogeneous Dirichlet boundary condition and vanishing nonlinearity, i.e. we set \( c \equiv 0 \) so that the equation becomes linear. The spatial domain is chosen as \( \Omega = (0,1) \times (0,1) \subset \mathbb{R}^2 \), the time interval is \( [0,T] = [0,1.57] \). For the temporal discretization we introduce the uniform time grid by

\[ t_j = j \Delta t \quad \text{for } j = 0, \ldots, n_t = 1570 \]

with \( \Delta t = 0.001 \). Snapshots of the analytical solution at three different time points are shown in Figure 1.
Due to the steep gradients in the neighborhood of the minimum and maximum, respectively, the use of an adaptive finite element discretization is justified. The resulting computational meshes as well as the corresponding finest mesh (reference mesh at the end of the simulation which is the union of all adaptive meshes generated during the simulation) are shown in Figure 2.

The number of nodes of the adaptive meshes varies between 3637 and 7071 points. The finest mesh has 18628 nodes. A uniform mesh with grid size of order of the diameter of the smallest triangles in the adaptive grids ($h_{\text{min}} = 0.0047$) would have 93025 nodes. This clearly reveals the benefit of using adaptive meshes for snapshot generation which is also well reflected in the comparison of the computational times needed for the the snapshot generation on the adaptive
mesh taking 944 seconds compared to 8808 seconds on the uniform mesh, see Table 3) for the speedup factors obtained by spatial adaptation. In Figure 3, the resulting normalized eigenspectrum of the correlation matrix for uniform spatial discretization (“uniform FE mesh”), the normalized eigenspectrum of the matrix $K$ (27) without interpolation (“infPOD”) as well as the normalized eigenspectrum of the correlation matrix using snapshots interpolated onto the finest mesh (“adaptive FE mesh”) is shown.

![Fig. 3: Example 1: Comparison of the normalized eigenvalues using an adaptive and a uniform spatial mesh, respectively. Left: all eigenvalues, middle: first 200 largest eigenvalues, right: first 200 largest eigenvalues with different error tolerances for the adaptivity (1.5 times bigger and smaller error tolerances, respectively).](image)

We observe that the eigenvalues for both adaptive approaches coincide. This numerically validates what we expect from theory: the information content which is contained in the matrix $K$ when we explicitly compute the entries without interpolation is the same as the information content contained within the eigenvalue problem which is formulated when using the finest mesh. No information is added or lost. Moreover, we recognize that about the first 28 eigenvalues computed corresponding to the adaptive simulation coincide with the simulation on a uniform mesh. From index 29 on, the methods deliver different results: for the uniform discretizations, the normalized eigenvalues fall below machine precision at around index 100 and stagnate. In contrary, the normalized eigenvalues for both adaptive approaches flatten in the order around $10^{-10}$. If the error tolerance for the spatial discretization error is chosen larger (or smaller), the stagnation of the eigenvalues in the adaptive method takes place at a higher (or lower) order (see Figure 3, right). Concerning dynamical systems, the magnitude of the eigenvalue corresponds to the characteristic properties of the underlying dynamical system: the larger the eigenvalue, the more information is contained in the corresponding eigenfunction. Since all adaptive meshes are contained in the uniform mesh, the difference in the amplitude of the eigenvalues is due to the interpolation errors during refinement and coarsening. This is the price we
have to pay for faster snapshot generation using adaptive methods. A further aspect gained from the decay behaviour of the eigenvalues in the adaptive case is the following; the adaptive approach filters out the noise in the system which is related to the modes corresponding to the singular values that are not not matched by the eigenvalues of the adaptive approach. This in the language of frequencies means that the overtones in the systems which get lost in the adaptive computations live in the space which is neglected by the POD method based on adaptive finite element snapshots. From this point of view, adaptivity can be interpreted as a smoother.

The first, second and fifth POD modes of Run 1 obtained by the adaptive approach are depicted in Figure 4. We observe the classical appearance of the basis functions. The initial condition is reflected by the first POD basis function. The next basis functions admit a number of minima and maxima corresponding to the index in the basis: $\Psi_2$ has two minima and two maxima etc. This behaviour is similar to the increasing oscillations in higher frequencies in trigonometric approximations. The POD basis functions corresponding to the uniform spatial discretization have a similar appearance.

![Fig. 4: Example 1: Surface plot (top) and view from above (bottom) of the POD basis functions $\Psi_1$ (left), $\Psi_2$ (middle) and $\Psi_5$ (right).](image)

Run 2 ([36, Example 6.2]). *Cahn-Hilliard system.* We consider Example 3 in the form (16) with $\Omega = (0,1.5) \times (0,0.75)$, $T = 0.025$, constant mobility $m \equiv 0.00002$, and constant surface tension $\sigma \equiv 24.5$. The interface parameter
\( \varepsilon \) is set to \( \varepsilon = 0.02 \), with resulting interface thickness \( \pi \cdot \varepsilon \approx 0.0628 \). We use the relaxed double obstacle free energy \( W_s^{\text{rel}} \) from (18) with \( s = 10^4 \). As initial condition, we choose a circle with radius \( r = 0.25 \) and center \((0.375, 0.375)\). The initial condition is transported horizontally with constant velocity \( v = (30, 0)^T \). Furthermore, we choose \( V = H_0^1(\Omega) \) and \( H = L^2(\Omega) \). We set

\[
t_j = j \Delta t \quad \text{for } j = 1, \ldots, n_t = 1000,
\]

so that \( \Delta t = 2.5 \cdot 10^{-5} \). The numerical computations are performed with the semi-implicit Euler scheme. For this purpose let \( c^{j-1} \in V \) and \( c^j \in V \) denote the time-discrete solution at \( t_{j-1} \) and \( t_j \), respectively. The time-discrete Cahn-Hilliard system then takes the form: given \( c^{j-1} \), find \( c^j, w^j \) solving

\[
\begin{align*}
\frac{1}{\Delta t} \langle c^j - c^{j-1}, v_1 \rangle_H + \langle v \cdot \nabla c^{j-1}, v_1 \rangle_H + m \langle \nabla w^j, \nabla v_1 \rangle_H &= 0, \\
-\langle w^j, v_2 \rangle_H + \sigma \varepsilon \langle \nabla c^j, \nabla v_2 \rangle_H + \frac{\sigma}{\varepsilon} \langle W'_+(c^j) + W'_-(c^{j-1}), v_2 \rangle_H &= 0,
\end{align*}
\]

for all \( v_1, v_2 \in V \) and \( j = 2, \ldots, n_t \), where \( c^1 = c_0 \). Note that the free energy function \( W \) is split into a convex part \( W_+ \) and a concave part \( W_- \), such that \( W = W_+ + W_- \) and \( W'_+ \) is treated implicitly, whereas \( W'_- \) is treated explicitly with respect to time. This leads to an unconditionally energy stable time marching scheme, compare \([31]\). The system (26) is discretized in space using piecewise linear and continuous finite elements. The resulting nonlinear equation systems are solved using a semi-smooth Newton method.

Figures 5 shows the phase field (left) and the chemical potential (right) for the finite element simulation using adaptive meshes. The initial condition \( c_0 \) is transported horizontally with constant velocity.
Fig. 5: Example 3: Phase field $c$ (left) and chemical potential $\psi$ (right) computed on adaptive finite element meshes at $t = t_1$ (top), $t = T/2$ (middle) and $t = T$ (bottom).

The adaptive finite element meshes as well as the finest mesh which is generated during the adaptive finite element simulation are shown in Figure 6. The number of degrees of freedom in the adaptive meshes varies between 6113 and 8795. The finest mesh (overlay of all adaptive meshes) has 54108 degrees of freedom, whereas a uniform mesh with discretization fineness as small as the smallest triangle in the adaptive meshes has 88450 degrees of freedom.
Figure 7 shows the first, second and fifth POD mode for the phase field $c$ and the chemical potential $w$. Analogously to Run 1, we observe a periodicity in the POD basis functions corresponding to their basis index numbers.

**Fig. 6:** Example 3: *Adaptive finite element meshes and finest mesh.*

**Fig. 7:** Example 3: *First, second and fifth POD modes for $c$ (left) and $w$ (right).*
In the present example we only compare the POD procedure for two kinds of snapshot discretizations, namely the adaptive approach with using a finest mesh, and the uniform mesh approach, where the gridsize is chosen to be of the same size as the smallest triangle in the adaptive meshes. We choose \( X = L^2(\Omega) \) and compute a separate POD basis for each of the variables \( c \) and \( w \).

In Figure 8, a comparison is visualized concerning the normalized eigenspectrum for the phase field \( c \) and the chemical potential \( w \) using uniform and adaptive finite element discretization. We note for the phase field \( c \) that about the first 180 eigenvalues computed corresponding to the adaptive simulation coincide with the eigenvalues of the simulation on the finest mesh. Then, the eigenvalues corresponding to the uniform simulation decay faster. Similar observations apply for the chemical potential \( w \).

In order to control the accuracy of the POD-ROM, we use the criterium (9) to determine the basis length \( \ell \) which is required to represent a prescribed information content with the respective POD space. We will choose the POD basis length \( \ell_c \) for the phase field \( c \) and the number of POD modes \( \ell_w \) for the chemical potential, such that

\[
\ell_{\min} = \arg \min \{ \mathcal{E}(\ell) : \mathcal{E}(\ell) > 1 - p \}, \quad \text{with } \ell = \ell_c \text{ and } \ell_w,
\]

for a given value \( p \) representing the loss of information. Alternatively, the POD basis length could be chosen in alignment of the POD projection error (8) with the expected spatial and/or temporal discretization error, compare e.g. [36, Theorem 5.1]. Table 1 summarizes how to choose \( \ell_c \) and \( \ell_w \) in order to capture a desired amount of information. Moreover, it tabulates the POD projection error (8) depending on the POD basis length, where \( \lambda_i^c \) and \( \lambda_i^w \) denote the eigenvalues.

---

**Fig. 8:** Example 3: Comparison of the normalized eigenvalues for the phase field \( c \) (left) and the chemical potential \( w \) (right) using an adaptive and a uniform spatial mesh, respectively.
for the phase field \( c \) and the chemical potential \( w \), respectively. The results in Table 1 agree with our expectations: the smaller the loss of information \( p \) is, the more POD modes are needed and the smaller is the POD projection error.

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
p & \ell_c^{ad} & \sum_{i>\ell} \lambda_i^c & \ell_w^{ad} & \sum_{i>\ell} \lambda_i^w & \ell_c^{uni} & \sum_{i>\ell} \lambda_i^c \\
\hline
10^{-1} & 3 & 2.0 \cdot 10^{-3} & 4 & 156.9 \cdot 10^0 & 3 & 2.0 \cdot 10^{-3} \\
10^{-2} & 10 & 2.1 \cdot 10^{-4} & 13 & 15.8 \cdot 10^0 & 10 & 2.1 \cdot 10^{-4} \\
10^{-3} & 19 & 2.5 \cdot 10^{-5} & 26 & 1.8 \cdot 10^0 & 19 & 2.5 \cdot 10^{-5} \\
10^{-4} & 29 & 2.0 \cdot 10^{-6} & 211 & 1.8 \cdot 10^{-1} & 28 & 2.6 \cdot 10^{-6} \\
10^{-5} & 37 & 2.5 \cdot 10^{-7} & 644 & 1.1 \cdot 10^{-2} & 37 & 2.4 \cdot 10^{-7} \\
\hline
\end{array}
\]

Table 1: Example 3: Number of needed POD bases in order to achieve a loss of information below the tolerance \( p \) using adaptive finite element meshes (columns 2-5) and uniform finite element discretization (columns 6-9) and POD projection error.

**Run 3** ([36, Example 6.3]). *(Linear heat equation revisited).* We again consider again Example 2 with \( c \equiv 0 \). The purpose of this example is to confirm that our POD approach also is applicable in the case of \( r \)-adaptivity. We set up the matrix \( K \) for snapshots generated on sequences of non-nested spatial discretizations. This requires the integration over cut elements, see [36]. We choose \( \Omega = (0,1) \times (0,1) \subset \mathbb{R}^2 \), \([0,T] = [0,1] \), and we apply a uniform temporal discretization with time step size \( \Delta t = 0.01 \). The analytical solution in the present example is given by

\[
y(t,x) = \sin(\pi x_0) \cdot \sin(\pi x_1) \cdot \cos(2\pi t x_0),
\]

with source term \( f \) and the initial condition \( g \) chosen accordingly. The initial condition is discretized using piecewise linear and continuous finite elements on a uniform spatial mesh which is shown in Figure 9 (left). Then, at each time step, the mesh is disturbed by relocating each mesh node according to the assignment

\[
x_0 \leftarrow x_0 + \theta \cdot x_0 \cdot (x_0 - 1) \cdot (\Delta t/10), \\
x_1 \leftarrow x_1 + \theta \cdot 0.5 \cdot x_1 \cdot (x_1 - 1) \cdot (\Delta t/10),
\]

where \( \theta \in \mathbb{R}_+ \) is sufficiently small such that all coordinates of the interior nodes fulfill \( 0 < x_0 < 1 \) and \( 0 < x_1 < 1 \). After relocating the mesh nodes, the heat equation is solved on this mesh for the next time instance. We use Lagrange interpolation to transfer the finite element solution of the previous time step onto the new mesh. The disturbed meshes at \( t = 0.5 \) and \( t = 1.0 \) as well as an overlap of two meshes are shown in Figure 9. To compute the matrix \( K \) from (27) we
have to evaluate the corresponding inner products of the snapshots, where we need to integrate over cut elements.

![Fig. 9: Example 3: Uniform mesh (left), disturbed meshes at \( t = 0.5 \) and \( t = 1.0 \) (middle left, middle right), overlap of the mesh at \( t = 0 \) with the mesh at \( t = 1.0 \) (right). Here, we use \( \theta = 10 \).](image)

We compute the eigenvalue decomposition of \( K \) for different values of \( \theta \) and compare the results with a uniform mesh (i.e. \( \theta = 0 \)) in Figure 10. We note that the eigenvalues of the disturbed mesh are converging to the eigenvalues of the uniform mesh for \( \theta \to 0 \). As expected, the eigenvalue spectrum depends only weakly on the underlying mesh given that the mesh size is sufficiently small. The construction of the POD-ROM is done according to Examples 1 and 3.

Concerning the computational complexity of POD with r-adaptivity let us note that solving the heat equation takes 2.1 seconds on the disturbed meshes and 1.8 seconds on the uniform mesh. The computational time needed to compute each entry of the matrix \( K \) is 0.022 seconds and computing the eigenvalue decomposition for \( K \) takes 0.0056 seconds. Note that the cut element integration

![Fig. 10: Run 3: Decay of eigenvalues of matrix \( K \) with different meshes.](image)
problem for each matrix entry takes a fraction of time required to solve the finite element problem.

3 The POD Galerkin procedure

Once the POD basis is generated is can be used to set up a POD-Galerkin approximation of the original dynamical system. This is discussed in the present section.

3.1 The POD Galerkin procedure

Suppose that for given snapshots \( y^h_j \in V^h_j \subset X \), \( 1 \leq j \leq n \), we have computed the symmetric matrix

\[
K = \left( \sqrt{\alpha_i} \sqrt{\alpha_j} \langle y^h_i, y^h_j \rangle_X \right)_{1 \leq i, j \leq n}
\]

with rank \( K = r \leq n \) (27)

associated to the operator \( K \) from (6) together with its eigensystem. Its \( \ell \in \{1, \ldots, r\} \) largest eigenvalues are \( \{\lambda_i\}_{i=1}^\ell \) with corresponding eigenvectors \( \{\Phi_i\}_{i=1}^\ell \subset \mathbb{R}^n \). The POD basis \( \{\Psi_i\}_{i=1}^\ell \) is then given by (7), i.e.,

\[
\Psi_i = \frac{1}{\sqrt{\lambda_i}} \mathcal{Y}\Phi_i \quad \text{for } i = 1, \ldots, \ell.
\]

This POD basis is utilized in order to compute a reduced-order model for (14) along the lines of Section 2.3.3, where the space \( V^h \) is replaced by the space \( V^\ell = \text{span} \{\Psi_1, \ldots, \Psi_\ell\} \subset V \). More precisely we make the POD Galerkin ansatz

\[
y^\ell(t) = \sum_{i=1}^\ell \eta_i(t) \Psi_i = \sum_{i=1}^\ell \eta_i(t) \frac{1}{\sqrt{\lambda_i}} \mathcal{Y}\phi_i \quad \text{for all } t \in [0, T],
\]

as an approximation for \( y(t) \), with the Fourier coefficients

\[
\eta_i(t) = \langle y^\ell(t), \Psi_i \rangle_X = \left\langle y^\ell(t), \frac{1}{\sqrt{\lambda_i}} \mathcal{Y}\phi_i \right\rangle_X \quad \text{for } 1 \leq i \leq \ell.
\]

Inserting \( y^\ell \) into (14) and choosing \( V^\ell \subset V \) as the test space leads to the system

\[
\begin{align*}
\frac{d}{dt} \langle y^\ell(t), \Psi \rangle_H + a(y^\ell(t), \Psi) + \langle \mathcal{N}(y^\ell(t)), \Psi \rangle_{V', V} = \langle f(t), \Psi \rangle_{V', V} \\
\langle y^\ell(0), \Psi \rangle_H = \langle y_0, \Psi \rangle_H
\end{align*}
\]

(29)
for all $\Psi \in V^\ell$ and for almost all $t \in (0, T]$, using the ansatz (28), we can write (29) as an $\ell$-dimensional ordinary differential equation system for the POD mode coefficients $\eta(t) = (\eta_i(t))_{1 \leq i \leq \ell}$, $t \in (0, T]$, as follows:

$$\begin{aligned}
\sum_{j=1}^{\ell} \dot{\eta}_j(t) \langle \Psi_i, \Psi_j \rangle_H + \sum_{j=1}^{\ell} \eta_j(t) a(\Psi_j, \Psi_i) = \langle f(t) - \mathcal{N}(y^\ell(t)), \Psi_i \rangle_{V', V} \\
\sum_{j=1}^{\ell} \eta_j(0) \langle \Psi_i, \Psi_j \rangle_H = \langle y_\circ, \Psi_i \rangle_H
\end{aligned} \quad (30)$$

for $i = 1, \ldots, \ell$. Note that $\langle \Psi_i, \Psi_j \rangle_H = \delta_{ij}$ if we choose $X = H$ in the context of Section 2.3. In a next step we rewrite this system using relation between $\Psi_i$ and $\Phi_i$ given in (7). This leads to

$$\begin{aligned}
\sum_{j=1}^{\ell} \dot{\eta}_j(t) \frac{\langle \mathcal{Y} \Phi_i, \mathcal{Y} \Phi_j \rangle_H}{\sqrt{\lambda_i \lambda_j}} + \sum_{j=1}^{\ell} \eta_j(t) \frac{a(\mathcal{Y} \Phi_j, \mathcal{Y} \Phi_i)}{\sqrt{\lambda_i \lambda_j}} = \frac{\langle f(t) - \mathcal{N}(y^\ell(t)), \mathcal{Y} \Phi_i \rangle_{V', V}}{\sqrt{\lambda_i}} \\
\sum_{j=1}^{\ell} \eta_j(0) \frac{\langle \mathcal{Y} \Phi_i, \mathcal{Y} \Phi_j \rangle_H}{\sqrt{\lambda_i \lambda_j}} = \frac{\langle y_\circ, \mathcal{Y} \Phi_i \rangle_H}{\sqrt{\lambda_i}}
\end{aligned} \quad (31)$$

for $i = 1, \ldots, \ell$. In order to write (31) in a compact matrix-vector form, let us introduce the diagonal matrix

$$\Lambda \in \mathbb{R}^{\ell \times \ell} \text{ with } \Lambda = \text{diag} \left( \frac{1}{\sqrt{\lambda_1}}, \ldots, \frac{1}{\sqrt{\lambda_\ell}} \right) \in \mathbb{R}^{\ell \times \ell}.$$

From the first $\ell$ eigenvectors $\{\Phi_i\}_{i=1}^\ell$ of $\mathcal{K}$ we build the matrix

$$\Phi = [\Phi_1 | \ldots | \Phi_\ell] \in \mathbb{R}^{n \times \ell}.$$ 

Then, the system (31) can be written as the system

$$\begin{aligned}
(\Lambda \Phi^T \mathcal{K} \Phi \Lambda) \dot{\eta}(t) + (\Lambda \Lambda^\ell \Lambda) \eta(t) + \Lambda \mathcal{N}^\ell(\eta(t)) = \Lambda F^\ell(t) \quad \text{for } t \in (0, T], \\
\Lambda \Phi^T \mathcal{K} \Phi \Lambda \eta(0) = \Lambda \eta_\circ
\end{aligned} \quad (32)$$

for the vector-valued mapping $\eta = (\eta_1, \ldots, \eta_\ell)^T : [0, T] \rightarrow \mathbb{R}^\ell$, for the nonlinearity $\mathcal{N}^\ell = (\mathcal{N}_i^\ell(\cdot))_{1 \leq i \leq \ell} : \mathbb{R}^\ell \rightarrow \mathbb{R}^\ell$ with

$$\mathcal{N}_i^\ell(v) = \left\langle \mathcal{N}\left(\sum_{j=1}^{\ell} v_j \Psi_j\right), \varphi_i \right\rangle_{V', V} = \left\langle \mathcal{N}\left(\sum_{j=1}^{\ell} v_j \mathcal{Y} \Phi_j / \sqrt{\lambda_j}\right), \varphi_i \right\rangle_{V', V}$$

and for the stiffness matrix $A^\ell = ((A^\ell_{ij})) \in \mathbb{R}^{\ell \times \ell}$ given as

$$A^\ell_{ij} = a(\mathcal{Y} \Phi_j, \mathcal{Y} \Phi_i) \quad \text{for } 1 \leq i, j \leq \ell.$$
Note that the right hand side \( F^\ell(t) = (F^\ell_i(t))_{1 \leq i \leq \ell} \) and the initial condition \( \eta_0 = (\eta_{0i})_{1 \leq i \leq \ell} \) are given by

\[
F^\ell_i(t) = \langle f(t), Y\Phi_i \rangle_{V',V} = \langle Y^* f(t), \Phi_i \rangle_{\mathbb{R}^n}, \quad t \in [0,T] \text{ a.e.,}
\]

and

\[
\eta_{0i} = \langle y_0, Y\Phi_i \rangle_H = \langle Y^* y_0, \Phi_i \rangle_{\mathbb{R}^n},
\]

for \( i = 1, \ldots, \ell \), respectively. Their calculation can be done explicitly for any arbitrary finite element discretization. For a given function \( w \in V \) (for example \( w = f(t) \) or \( w = y_0 \)) with finite element discretization \( w = \sum_{i=1}^{m_w} w_i \chi_i \), nodal basis \( \{\chi_i\}_{i=1}^{m_w} \subset V \) and appropriate mode coefficients \( \{w_i\}_{i=1}^{m_w} \) we can compute

\[
\langle Y^* w \rangle_j = \langle w, y_j \rangle_X = \left\langle \sum_{i=1}^{m_w} w_i \chi_i, \sum_{k=1}^{m_j} y^j_k \varphi^j_k \right\rangle_X = \sum_{i=1}^{m_w} \sum_{k=1}^{m_j} w_i y^j_k \langle \chi_i, \varphi^j_k \rangle_X
\]

for \( j = 1, \ldots, n \) where \( y_j = \sum_{k=1}^{m_j} y^j_k \varphi^j_k \in V_j \) denotes the \( j \)-th snapshot. Again, for any \( i = 1, \ldots, m_w \) and \( k = 1, \ldots, m_j \), the computation of the inner product \( \langle \chi_i, \varphi^j_k \rangle_X \) can be done explicitly.

Obviously, for linear evolution equations the POD reduced-order model (32) can be set up and solved using snapshots with arbitrary finite element discretizations. The computation of the nonlinear component \( N^\ell(\eta(t)) \) needs particular attention. In Section 3.3 we discuss the options to treat the nonlinearity.

### 3.2 Time-discrete reduced-order model

In order to solve the reduced-order system (29) numerically, we apply the implicit Euler method for time discretization and use for simplicity the same temporal grid \( \{t_j\}_{j=0}^n \) as for the snapshots. It is also possible to use a different time grid, cf. [54]. The time-discrete reduced-order model reads

\[
\begin{aligned}
\frac{\langle y^\ell_j - y^\ell_{j-1}, \Psi \rangle_H}{\Delta t_j} + a(y^\ell_j, \Psi) + \langle N(y^\ell_j), \Psi \rangle_{V',V} &= \int_{t_{j-1}}^{t_j} \frac{\langle f(\tau), \Psi \rangle_{V',V}}{\Delta t_j} \, d\tau, \\
\langle y^\ell_1, \Psi \rangle_H &= \langle y_0, \Psi \rangle_H
\end{aligned}
\]

for all \( \Psi \in V^\ell \) and \( j = 2, \ldots, n \). Equivalently the following system holds for the coefficient vector \( \eta(t) \in \mathbb{R}^\ell \) (cf. (32)):

\[
\begin{aligned}
(\Lambda \Phi^\top K \Phi \Lambda) \left( \frac{\eta^j - \eta^{j-1}}{\Delta t_j} \right) + (\Lambda A^\ell \Lambda) \eta^j + \Lambda N^\ell(\eta^j) &= \Lambda F^\ell_j, \quad j = 2, \ldots, n \\
\Lambda \Phi^\top K \Phi \Lambda \eta^1 &= \Lambda \eta_0
\end{aligned}
\]

(34)
with the inhomogeneity $F^\ell_j = (F^\ell_{ji})_{1 \leq i \leq \ell}$, $j = 2, \ldots, n$, given as

$$F^\ell_{ji} = \int_{t_{j-1}}^{t_j} \frac{\langle f(\tau), Y\Phi_i \rangle_{V',V}}{\Delta t_j} d\tau = \int_{t_{j-1}}^{t_j} \frac{\langle y^*(\tau), \Phi_i \rangle_{\mathbb{R}^n}}{\Delta t_j} d\tau.$$ 

3.3 Discussion of the computation of the nonlinear term

Let us now consider the computation of the nonlinear term $\Lambda N^\ell(\eta^j) \in \mathbb{R}^\ell$ of the POD-ROM (32). It holds true

$$\left(\Lambda N^\ell(\eta^j)\right)_k = \langle N(y^\ell(t)), \Psi_k \rangle_{V',V} = \left\langle N(\sum_{i=1}^\ell \eta_i(t) \Psi_i), \Psi_k \right\rangle_{V',V}$$

for $k = 1, \ldots, \ell$. It is well-known that the evaluation of nonlinearities in the reduced-order model context in computationally expensive. To make this clear, let us assume, we are given a uniform finite element discretization with $m$ degrees of freedom. Then, in the fully discrete setting, the nonlinear term has the form

$$\Psi^T W N^h(\Psi(t)) \in \mathbb{R}^\ell, \quad t \in [0, T] \text{ a.e.},$$

where $\Psi = [\Psi_1 | \ldots | \Psi_\ell] \in \mathbb{R}^{m \times \ell}$ is the matrix in which the POD modes are stored columnwise and $W \in \mathbb{R}^{m \times m}$ is a weighting matrix related to the utilized inner product (cf. (1)). Hence, the treatment of the nonlinearity requires the expansion of $\Psi(t) \in \mathbb{R}^m$ in the full space for $t \in [0, T]$ a.e. Then the nonlinearity can be evaluated and finally the result is projected back to the POD space. Obviously, this means that the reduced-order model is not fully independent of the high-order dimension $m$ and efficient simulation cannot be guaranteed. Therefore, it is convenient to seek for hyper reduction, i.e., for a treatment of the nonlinearity, where the model evaluation cost is related to the low dimension $\ell$. Common choices are empirical interpolation methods like, e.g., EIM ([13]), DEIM ([22]), and QDEIM ([29]). Another option is dynamic mode decomposition for nonlinear model order reduction, see e.g. [7]. Furthermore, in [93] nonlinear model reduction is realized by replacing the nonlinear term by its interpolation in the finite element space. An alternative approach for the treatment of the nonlinearity is missing point estimation [10], or best points interpolation [65]. Most of these methods need a common reference mesh for the computations. To overcome this restriction we propose different paths which allow for more general discrete settings like $r$-adaptivity discussed in Run 3.

One option is to use EIM [13], Alternatively, we can linearize and project the nonlinearity onto the POD space. For this approach, let us consider the linear
reduced-order system for a fixed given state $\tilde{y}$, which takes the form
\[
\begin{aligned}
\frac{d}{dt} \langle y^\ell(t), \Psi \rangle_H + a(y^\ell(t), \Psi) + \langle \mathcal{N}(\tilde{y}(t)), \Psi \rangle_{V', V} &= \langle f(t), \Psi \rangle_{V', V}, \\
\langle y^\ell(0), \Psi \rangle_H &= \langle y_0, \Psi \rangle_H
\end{aligned}
\]
for all $\Psi \in V^\ell$ and for almost all $t \in (0, T]$. The linear evolution problem (35) can be set up and solved explicitly without spatial interpolation. In the numerical examples in Section 6, we take the finite element solution as given state in each time step, i.e., $\tilde{y}(t_j) = y_j$ for $j = 2, \ldots, n$.

Furthermore, the linearization of the reduced-order model (29) can be considered:
\[
\begin{aligned}
\frac{d}{dt} \langle y^\ell(t), \Psi \rangle_H + a(y^\ell(t), \Psi) + \langle \mathcal{N}'(\tilde{y}(t))y^\ell(t), \Psi \rangle_{V', V}
\end{aligned}
\]
\[
\begin{aligned}
= \langle f(t) - \mathcal{N}(\tilde{y}(t)) + \mathcal{N}'(\tilde{y}(t))\tilde{y}(t), \Psi \rangle_{V', V}
\end{aligned}
\]
\[
\begin{aligned}
\langle y^\ell(0), \Psi \rangle_H &= \langle y_0, \Psi \rangle_H
\end{aligned}
\]
for all $\Psi \in V^\ell$ and for almost all $t \in (0, T]$, where $\mathcal{N}'$ denotes the Fréchet derivative of the nonlinear operator $\mathcal{N}$. This linearized problem is of interest e.g. in the context of optimal control, where it occurs in each iteration level within sequential quadratic programming (SQP) methods; see [45], for example. Choosing the finite element solution as given state in each time instance and using (7) leads to
\[
\begin{aligned}
\langle \mathcal{N}(y_j), \Psi_i \rangle_{V', V} &= \frac{1}{\sqrt{\lambda_i}} \sum_{k=1}^n \sqrt{\alpha_k(\Phi_i)_k} \langle \mathcal{N}(y_j), y_k \rangle_{V', V}, \\
\langle \mathcal{N}'(y_j)y^\ell(t_j), \Psi_i \rangle_{V', V} &= \left\langle \mathcal{N}'(y_j)\left( \sum_{k=1}^\ell \eta_k(t_j)\Psi_k \right), \Psi_i \right\rangle_{V', V}, \\
&= \sum_{k=1}^\ell \eta_k(t_j) \frac{1}{\sqrt{\lambda_k\lambda_i}} \sum_{\nu=1}^n \sum_{\mu=1}^n \sqrt{\alpha_\nu\alpha_\mu}(\Phi_k)_\nu(\Phi_i)_\mu \langle \mathcal{N}'(y_j)y_{\nu\mu}, y_{\nu\mu} \rangle_{V', V}, \\
\langle \mathcal{N}'(y_j)y_j, \Psi_i \rangle_{V', V} &= \frac{1}{\sqrt{\lambda_i}} \sum_{k=0}^n \sqrt{\alpha_k(\Phi_i)_k} \langle \mathcal{N}'(y_j)y_j, y_k \rangle_{V', V}
\end{aligned}
\]
for $j = 2, \ldots, n$ and $i = 1, \ldots, \ell$. Finally, we approximate the nonlinearity $\mathcal{N}^\ell(\eta^j) \in \mathbb{R}^\ell$ in (34) by
\[
(\mathcal{N}^\ell(\eta^j))_i \approx \langle \mathcal{N}(y_j) + \mathcal{N}'(y_j)(y^\ell(t_j) - y_j), \Psi_i \rangle_{V', V}
\]
for $j = 2, \ldots, n$ and $i = 1, \ldots, \ell$, which can be written as
\[
\mathcal{N}^\ell(\eta^j) \approx \Lambda\Phi^\top N^j + \Lambda\Phi^\top N^j_y\Phi \Lambda\eta^j - \Lambda\Phi^\top N^j_y
\]
where
\[ \mathbf{N}_j = \begin{pmatrix} \langle \mathcal{N}(y_j), \sqrt{\alpha_1} y_1 \rangle_{V',V} \\ \vdots \\ \langle \mathcal{N}(y_j), \sqrt{\alpha_n} y_n \rangle_{V',V} \end{pmatrix} \in \mathbb{R}^n, \quad \mathbf{N}_y = \begin{pmatrix} \langle \mathcal{N}'(y_j)y_j, \sqrt{\alpha_1} y_1 \rangle_{V',V} \\ \vdots \\ \langle \mathcal{N}'(y_j)y_j, \sqrt{\alpha_n} y_n \rangle_{V',V} \end{pmatrix} \in \mathbb{R}^n, \]

and with \( \tilde{y}_j = \sqrt{\alpha_j} y_j, \ j = 1, \ldots, n, \)
\[ \mathbf{N}^j_y = \begin{pmatrix} \langle \mathcal{N}'(y_j)\tilde{y}_1, \tilde{y}_1 \rangle_{V',V} & \ldots & \langle \mathcal{N}'(y_j)\tilde{y}_n, \tilde{y}_1 \rangle_{V',V} \\ \vdots & \ddots & \vdots \\ \langle \mathcal{N}'(y_j)\tilde{y}_1, \tilde{y}_n \rangle_{V',V} & \ldots & \langle \mathcal{N}'(y_j)\tilde{y}_n, \tilde{y}_n \rangle_{V',V} \end{pmatrix} \in \mathbb{R}^{n \times n}. \]

For weakly nonlinear systems this approximation may be sufficient, depending on the problem and its goal. A great advantage of linearizing the semilinear partial differential equation is that only linear equations need to be solved which leads to a further speedup, see Table 5. However, if a more precise approximation is desired or necessary, we can think of approximations including higher order terms, like quadratic approximation, see, e.g., [23] and [79], or Taylor expansions, see, e.g., [68, 69] and [33]. Nevertheless, the efficiency of higher order approximations is limited due to growing memory and computational costs.

### 3.4 Expressing the POD solution in the full spatial domain

Having determined the solution \( \eta(t) \) to (32), we can set up the reduced solution \( y^\ell(t) \) in a continuous framework:
\[ y^\ell(t) = \sum_{i=1}^{\ell} \eta_i(t) \left( \frac{1}{\sqrt{\lambda_i}} \sum_{j=1}^{n} \sqrt{\alpha_j} (\Phi_i)_j y_j \right). \tag{37} \]

Now, let us turn to the fully discrete formulation of (37). For a time-discrete setting, we introduce for simplicity the same temporal grid \( \{t_j\}_{j=1}^{n} \) as for the snapshots. The snapshots (25) admit the expansion
\[ y_j = \sum_{i=1}^{m_j} y_j^j \varphi_i \quad \text{for } j = 1, \ldots, n. \]

Let \( \{Q_j^i\}_{r=1}^{l_j} \) denote an arbitrary set of grid points for the reduced system at time level \( t_j \). The fully discrete POD solution can be computed by evaluation:
\[ y^\ell(t_j, Q_j^i) = \sum_{i=1}^{\ell} \eta_i(t_j) \left( \frac{1}{\sqrt{\lambda_i}} \sum_{\nu=1}^{n} \sqrt{\alpha_{\nu}} (\Phi_i)_\nu \left( \sum_{k=1}^{m_\nu} y_k^\nu \varphi_k(Q_j^i) \right) \right) \tag{38} \]
for $r = 1,\ldots,l_j$ and $j = 1,\ldots,n$. This allows us to use any grid for expressing the POD solution in the full spatial domain. For example, we can use the same nodes at time level $j$ for the POD simulation as we have used for the snapshots, i.e., for $j = 1,\ldots,n$ it holds $l_j = m_j$ and $Q_r^j = P_r^j$ for all $r, k = 1,\ldots,m_j$. Another option can be to choose

$$\{Q_r^j\}_{r=1}^{l_j} = \bigcup_{j=1}^{m_j} \bigcup_{k=1}^{m_j} \{P_r^j\}$$

i.e., the common finest grid. Obviously, a special and probably the easiest case concerning the implementation is to choose snapshots which are expressed with respect to the same finite element basis functions and utilize the common finest grid for the simulation of the reduced-order system, which is proposed by [90]. After expressing the adaptively sampled snapshots with respect to a common finite element space, the subsequent steps coincide with the common approach of taking snapshots which are generated without adaptivity. Then, expression (38) simplifies to

$$y^j(t_j, P_r) = \sum_{i=1}^{\ell} \eta_i(t_j) \left( \frac{1}{\sqrt{\lambda_i}} \sum_{\nu=1}^{n} \sqrt{\alpha_{\nu}(\Phi_i)_{\nu} y_{\nu}} \right)$$

for $j = 1,\ldots,n$, (39)

where $\{P_r\}_{r=1}^{m}$ are the nodes of the common finite element space.

**Run 4** ([36, Example 6.1]). Let us revisit Run 1 and consider its POD Galerkin solutions. The POD solutions for $\ell = 10$ and $\ell = 50$ POD basis functions using spatial adaptive snapshots which are interpolated onto the finest mesh are shown in Figure 11. As expected, the more POD basis functions we use (until stagnation of the corresponding eigenvalues), the less oscillations appear in the POD solution and the better the approximation is.
Table 2 compares the approximation quality of the POD solution using adaptively generated snapshots which are interpolated onto the finest mesh with snapshots of uniform spatial discretization depending on different POD basis lengths. Then, for $\ell = 20$ we obtain a relative $L^2(0, T; L^2(\Omega))$-error between the POD solution and the finite element solution of size $\varepsilon_{\text{FE}} = 3.07 \cdot 10^{-2}$, and a relative $L^2(0, T; L^2(\Omega))$-error between the POD solution and the true solution of size $\varepsilon_{\text{true}} = 2.16 \cdot 10^{-2}$.

<table>
<thead>
<tr>
<th>$\ell$</th>
<th>$\varepsilon_{\text{ad}}$</th>
<th>$\varepsilon_{\text{uni}}$</th>
<th>$\varepsilon_{\text{ad}}$</th>
<th>$\varepsilon_{\text{uni}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1.30 \cdot 10^0$</td>
<td>$1.30 \cdot 10^0$</td>
<td>$1.28 \cdot 10^0$</td>
<td>$1.30 \cdot 10^0$</td>
</tr>
<tr>
<td>3</td>
<td>$7.49 \cdot 10^{-1}$</td>
<td>$7.58 \cdot 10^{-1}$</td>
<td>$7.46 \cdot 10^{-1}$</td>
<td>$7.60 \cdot 10^{-1}$</td>
</tr>
<tr>
<td>5</td>
<td>$4.39 \cdot 10^{-1}$</td>
<td>$4.45 \cdot 10^{-1}$</td>
<td>$4.39 \cdot 10^{-1}$</td>
<td>$4.46 \cdot 10^{-1}$</td>
</tr>
<tr>
<td>10</td>
<td>$1.37 \cdot 10^{-1}$</td>
<td>$1.37 \cdot 10^{-1}$</td>
<td>$1.36 \cdot 10^{-1}$</td>
<td>$1.38 \cdot 10^{-1}$</td>
</tr>
<tr>
<td>20</td>
<td>$3.08 \cdot 10^{-2}$</td>
<td>$1.56 \cdot 10^{-2}$</td>
<td>$2.17 \cdot 10^{-2}$</td>
<td>$1.60 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>30</td>
<td>$2.59 \cdot 10^{-2}$</td>
<td>$2.04 \cdot 10^{-3}$</td>
<td>$1.49 \cdot 10^{-2}$</td>
<td>$3.00 \cdot 10^{-3}$</td>
</tr>
<tr>
<td>50</td>
<td>$2.63 \cdot 10^{-2}$</td>
<td>$5.67 \cdot 10^{-5}$</td>
<td>$1.41 \cdot 10^{-2}$</td>
<td>$2.07 \cdot 10^{-3}$</td>
</tr>
<tr>
<td>100</td>
<td>$2.63 \cdot 10^{-2}$</td>
<td>$6.48 \cdot 10^{-8}$</td>
<td>$1.40 \cdot 10^{-2}$</td>
<td>$2.06 \cdot 10^{-3}$</td>
</tr>
<tr>
<td>150</td>
<td>$2.63 \cdot 10^{-2}$</td>
<td>$8.13 \cdot 10^{-7}$</td>
<td>$1.39 \cdot 10^{-2}$</td>
<td>$2.07 \cdot 10^{-3}$</td>
</tr>
</tbody>
</table>

Table 2: Run 4: Relative $L^2(0, T; L^2(\Omega))$-error between the POD solution and the finite element solution (columns 2-3) and the true solution (columns 4-5), respectively, using adaptive finite element snapshots which are interpolated onto the finest mesh and using a uniform mesh.
We note that $\varepsilon_{\text{FE}}$ decays down to $10^{-8}$ ($\ell = 100$) and then stagnates if using a uniform mesh. This behaviour is clear, since the more POD basis we include (up to stagnation of the corresponding eigenvalues), the better is the POD solution an approximation for the finite element solution. On the other hand, $\varepsilon_{\text{true}}$ starts to stagnate after $\ell = 29$. This is due to the fact that at this point the spatial discretization error dominates the modal error. This is in accordance to the decay of the eigenvalues shown in Figure 3. However, further error contributions like temporal and spatial discretization errors affect the error between true the state and POD solution. This explains the stagnation of the error in Table 2, columns 4 and 5. This is accounted for e.g. in the error estimation presented in [36, Theorem 5.1].

The computational times for the full and the low order simulation using uniform finite element discretizations and adaptive finite element snapshots, which are interpolated onto the finest mesh, respectively, are listed in Table 3.

<table>
<thead>
<tr>
<th></th>
<th>adaptive FE mesh</th>
<th>uniform FE mesh</th>
<th>speedup factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE simulation</td>
<td>944 sec</td>
<td>8808 sec</td>
<td>9.3</td>
</tr>
<tr>
<td>POD offline computations</td>
<td>264 sec</td>
<td>1300 sec</td>
<td>4.9</td>
</tr>
<tr>
<td>POD simulation</td>
<td>187 sec</td>
<td></td>
<td>–</td>
</tr>
<tr>
<td>speedup factor</td>
<td>5.0</td>
<td>47.1</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 3: Run 4: CPU times for FE and POD simulation using uniform finite element meshes and adaptive finite element snapshots which are interpolated onto the finest mesh, respectively, and using $\ell = 50$ POD modes.

Once the POD basis is computed in the offline phase, the POD simulation corresponding to adaptive snapshots is 5 times faster than the FE simulation using adaptive finite element meshes. This speedup factor is important when one considers e.g. optimal control problems with time-dependent PDEs, where the POD-ROM can be used as surrogate model in repeated solution of the underlying PDE model. In the POD offline phase, the most expensive task is to express the snapshots with respect to the common finite element space, which takes 226 seconds. Since $K$ is symmetric, it suffices to calculate the entries on and above the diagonal, which are $\sum_{k=1}^{n} k = (n^2 + n)/2$ entries. Thus, the computation of each entry in the correlation matrix $K$ using a common finite element space takes around 0.00018 seconds. We note that in the approach explained in Section 3 and 4, the computation of the matrix $K$ (27) is expensive. For each entry the calculation time is around 0.03 seconds, which leads to a computation time
of around $36997$ seconds for the matrix $K$. The same effort is needed to build $\mathcal{V}^* A \mathcal{V}$. In this case, the offline phase takes therefore around $88271$ seconds. For this reason, the approach to interpolate the adaptive generated snapshots onto the finest mesh is computationally more favorable. But since the computation of $K$ can be parallelized, the offline computation time can be reduced provided that the appropriate hardware is available.

**Run 5** (Cahn-Hilliard equations). Now let us revisit Run 3, where we in the following run the numerical simulations for different combinations of numbers for $\ell_c$ and $\ell_w$ of Table 1. The approximation quality of the POD solution using adaptive meshes is compared to the use of a uniform mesh in Table 4. As expected, Table 4 shows that the error between the POD surrogate solution and the high-fidelity solution gets smaller for an increasing number of utilized POD basis functions. Moreover, a larger number of POD modes is needed for the chemical potential $w$ than for the phase field $c$ in order to get an error in the same order which is in accordance to the fact that the decay of the eigenvalues for $w$ is slower than for $c$ as seen in Figure 8.

<table>
<thead>
<tr>
<th>$\ell_c$</th>
<th>$\ell_w$</th>
<th>$c : \varepsilon_{\text{ad}}^{\text{FE}}$</th>
<th>$w : \varepsilon_{\text{ad}}^{\text{FE}}$</th>
<th>$c : \varepsilon_{\text{uni}}^{\text{FE}}$</th>
<th>$w : \varepsilon_{\text{uni}}^{\text{FE}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>$8.44 \cdot 10^{-3}$</td>
<td>$3.00 \cdot 10^{0}$</td>
<td>$8.44 \cdot 10^{-3}$</td>
<td>$3.75 \cdot 10^{0}$</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>$3.30 \cdot 10^{-3}$</td>
<td>$3.77 \cdot 10^{-1}$</td>
<td>$3.30 \cdot 10^{-3}$</td>
<td>$4.32 \cdot 10^{-1}$</td>
</tr>
<tr>
<td>19</td>
<td>26</td>
<td>$1.57 \cdot 10^{-3}$</td>
<td>$2.12 \cdot 10^{-1}$</td>
<td>$1.57 \cdot 10^{-3}$</td>
<td>$2.39 \cdot 10^{-1}$</td>
</tr>
<tr>
<td>29</td>
<td>26</td>
<td>$7.34 \cdot 10^{-4}$</td>
<td>$1.09 \cdot 10^{-1}$</td>
<td>$7.32 \cdot 10^{-4}$</td>
<td>$1.16 \cdot 10^{-1}$</td>
</tr>
<tr>
<td>37</td>
<td>26</td>
<td>$3.57 \cdot 10^{-4}$</td>
<td>$4.82 \cdot 10^{-2}$</td>
<td>$3.55 \cdot 10^{-4}$</td>
<td>$5.04 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>$1.88 \cdot 10^{-4}$</td>
<td>$2.17 \cdot 10^{-2}$</td>
<td>$1.86 \cdot 10^{-4}$</td>
<td>$2.33 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>65</td>
<td>26</td>
<td>$9.74 \cdot 10^{-5}$</td>
<td>$1.11 \cdot 10^{-2}$</td>
<td>$9.56 \cdot 10^{-5}$</td>
<td>$1.15 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>$3.37 \cdot 10^{-5}$</td>
<td>$3.56 \cdot 10^{-3}$</td>
<td>$3.22 \cdot 10^{-5}$</td>
<td>$3.42 \cdot 10^{-3}$</td>
</tr>
</tbody>
</table>

**Table 4:** Run 5: Relative $L^2(0,T; L^2(\Omega))$-error between the POD solution and the finite element solution using adaptive meshes (columns 3-4) and using a uniform mesh (columns 5-6), respectively.

We now discuss the treatment of the nonlinearity and also investigate the influence of non-smoothness of the model equations to the POD procedure. Using the convex-concave splitting for $W$, we obtain for the Moreau-Yosida relaxed double obstacle free energy the concave part $W_\text{rel}^- (c) = \frac{1}{2} (1 - c^2)$ and the convex part $W_\text{rel}^+ (c) = \frac{4}{3} (\max(c - 1,0)^2 + \min(c + 1,0)^2)$. This means that the first derivative of the concave part is linear with respect to the phase field variable $c$. The challenging part is the convex term with nonsmooth first derivative. For a
comparison, we consider the smooth polynomial free energy with concave part $W^p_-(c) = \frac{1}{4}(1 - 2c^2)$ and convex part $W^p_+(c) = \frac{1}{4}c^4$.

Figure 12 shows the decay of the normalized eigenspectrum for the phase field $c$ (left) and the first derivative of the convex part $W'_+(c)$ (right) for the polynomial and the relaxed double obstacle free energy. Obviously, in the non-smooth case more POD modes are needed for a good approximation than in the smooth case. This behaviour is similar to the decay of the Fourier coefficients in the context of trigonometric approximation, where the decay of the Fourier coefficients depends on the smoothness of the approximated object.

![Normalized eigenspectrum for polynomial and relaxed double obstacle free energy](image)

**Fig. 12:** Run 5: Comparison of the normalized eigenvalues for $c$ (left) and the first derivative of the convex part $W'_+(c)$ of the free energy (right) using polynomial and relaxed double obstacle energy, respectively.

Table 5 summarizes computational times for different finite element runs as well as reduced-order simulations using the polynomial and the relaxed double obstacle free energy, respectively. In addition, the approximation quality is compared. The computational times are rounded averages from various test runs. It turns out that the finite element simulation (row 1) using the smooth potential is around two times faster than using the non-smooth potential. This is due to the fact that in the smooth case, 2-3 Newton steps are needed for convergence in each time step, whereas in the non-smooth case 6-8 iterations are needed in the semismooth Newton method.
Using the smooth polynomial free energy, the reduced-order simulation is 8-9 times faster than the finite element simulation, whereas using the relaxed double obstacle free energy only delivers a very small speedup. The inclusion of DEIM (we use $\ell_{\text{deim}} = \ell_c$) in the reduced-order model leads to immense speedup factors for both free energy functions (row 8). This is due to the fact that the evaluation of the nonlinearity in the reduced-order model is still dependent on the full spatial dimension and hyper reduction methods are necessary for useful speedup factors. Note that the speedup factors are of particular interest in the context of optimal control problems. At the same time, the relative $L^2(Q)$-error between the finite element solution and the ROM-DEIM solution is close to the quality of the reduced-order model solution (row 10-11).

However, in the case of the non-smooth free energy function using $\ell_c = 19$ POD modes for the phase field and $\ell_w = 26$ POD modes for the chemical potential, the inclusion of DEIM has the effect that the semismooth Newton method does not converge. For this reason, we treat the nonlinearity by applying the technique explained in Section 3.1, i.e. we project the finite element snapshots for $W'_+(c)$ (which are interpolated onto the finest mesh) onto the POD space. Since this leads to linear systems, the computational times are very small (row 6). The error between the finite element solution and the reduced-order solution using projection of the nonlinearity is of the magnitude $10^{-02}/10^{-03}$. Depending on the motivation, this approximation quality might be sufficient. Nevertheless, we
note that for large numbers of POD modes, using the projection of the nonlinearity onto the POD space leads to a large increase of the error. ♦

To summarize, a POD reduced-order model construction approach is proposed which can be set up and solved for snapshots originating from arbitrary finite element (and also other) spaces. The method is applicable for $h$-, $p$- and $r$-adaptive finite elements. It is motivated from an infinite-dimensional perspective. Using the method of snapshots we are able to set up the correlation matrix $K$ from (27) by evaluating the inner products of snapshots which live in different finite element spaces. For non-nested meshes, this requires the detection of cell collision and integration over cut finite elements. A numerical strategy how to implement this practically is elaborated and numerically tested. Using the eigenvalues and eigenvectors of this correlation matrix, we are able to set up and solve a POD surrogate model that does not need the expression of the snapshots with respect to the basis of a common finite element space or the interpolation onto a common reference mesh. Moreover, an error bound for the error between the true solution and the solution to the POD-ROM using spatially adapted snapshots is available in [36, Theorem 5.1]. The numerical tests show that the POD projection error decreases if the number of utilized POD basis functions is increased. However, the error between the POD solution and the true solution stagnates when the spatial discretization error dominates. Moreover, the numerics show that using the correlation matrix calculated explicitly without interpolation in order to build a POD-ROM gives the same results as the approach where the snapshots are interpolated onto the finest mesh. From a computational point of view, sufficient hardware should be available in order to compute the correlation matrix in parallel and make the offline computational time competitive. For semilinear evolution problems, the nonlinearity is treated by linearization. This is of interest in view of optimal control problems, in which a linearized state equation has to be solved in each SQP iteration level. An appropriate treatment of the nonlinearity in our applications gains significant speedup of the ROM in computational times when compared to the full simulations. This makes POD-MOR with adaptive finite elements an ideal approach for the construction of surrogate models in e.g. optimal control with nonlinear PDE systems as they arise e.g. in the context of multi-phase flow control problems.
4 Certification with a priori and a posteriori error estimates

As we have seen in Section 3 POD provides a method for deriving low order models of dynamical systems. It can be thought of as a Galerkin approximation in the spatial variable, built from functions corresponding to the solution of the physical system at prespecified time instances. After carrying out a singular value decomposition the leading $\ell$ generalized eigenfunctions are chosen as the POD basis $\{\Psi_j\}_{j=1}^\ell$ of rank $\ell$. As soon as one uses POD, questions concerning the quality of the approximation properties, convergence, and rate of convergence become relevant. Let us refer, e.g., to the literature [20, 38, 52, 53, 80, 83, 84, 75] for a-priori error analysis for POD Galerkin approximations. It turns out that the error depends on the decay of the sum $\sum_{i>\ell} \lambda_i$, the error $\Delta t^\beta$ (with an appropriate $\beta \geq 1$) due to the used time integration method, the used Galerkin spaces $\{V^h\}_{j=1}^n$ and the choice $X = H$ or $X = V$. In particular, best approximation properties hold provided the time differences $\dot{y}^h(t_j)$ (or the finite difference discretizations) are included in the snapshot ensembles; cf. [52, 54, 84].

Let us recall numerical test examples from [38, Section 1.5]. The programs are written in MATLAB using the Partial Differential Equation Toolbox for the computation of the piecewise linear finite element (FE) discretization. For the temporal integration the implicit Euler method is applied based on the equidistant time grid $t_j = (j-1)\Delta t$, $j = 1, \ldots, n$ and $\Delta t = T/(n-1)$.

Run 6 (POD for the heat equation; cf. [38, Run 1]). We choose the final time $T = 3$, the spatial domain $\Omega = (0, 2) \subset \mathbb{R}$, the Hilbert spaces $H = L^2(\Omega)$, $V = H^1_0(\Omega)$, the source term $f(t, x) = t^3 - x^2$ for $(t, x) \in Q = (0, T) \times \Omega$ and the discontinuous initial value $y_0(x) = \chi_{(0,5,1,0)} - \chi_{(1,1,5)}$ for $x \in \Omega$, where, e.g., $\chi_{(0.5,1)}$ denotes the characteristic function on the subdomain $(0.5, 1) \subset \Omega$, $\chi_{(0.5,1)}(x) = 1$ for $x \in (0.5, 1)$ and $\chi_{(0.5,1)}(x) = 0$ otherwise. We consider a discretization of the linear heat equation

$$
y_t(t, x) - \Delta y(t, x) = f(t, x) \quad \text{for } (t, x) \in Q, \\
y(t, x) = 0 \quad \text{for } (t, x) \in \Sigma = (0, T) \times \partial \Omega, \\
y(0, x) = y_0(x) \quad \text{for } x \in \Omega.
$$

To obtain an accurate approximation of the exact solution we choose $n = 4000$ so that $\Delta t \approx 7.5 \cdot 10^{-4}$ holds. For the FE discretization we choose $m = 500$ spatial grid points and the equidistant mesh size $h = 2/(m + 1) \approx 4 \cdot 10^{-3}$. Thus, the FE error – measured in the $H$-norm – is of the order $10^{-4}$. In the left graphic of Figure 13, the FE solution $y^h$ to the state equation (40) is visualized. To
compute a POD basis \( \{ \Psi_i \}_i^{\ell} \) of rank \( \ell \) we utilize the multiple discrete snapshots \( y_j^1 = y^h(t_j) \) for \( 1 \leq j \leq n_t \) as well \( y_j^2 = (y^h(t_j) - y^h(t_{j-1}))/\Delta t, \) \( j = 2, \ldots, n_t, \) i.e., we include the temporal difference quotients in the snapshot ensemble and \( K = 2, n_1 = n_2 = n_t \). We choose \( X = H \) and utilize the (stable) singular value decomposition to determine the POD basis of rank \( \ell \); compare Section 2.2. We address this issue in a more detail in Run 9. Since the snapshots are FE functions, the POD basis elements are also FE functions. In the right plot of Figure 13, the projection and reduced-order error given by

\[
\text{PROJ Error}(\ell) = \left( \sum_{j=1}^{n_t} \alpha_j \left\| y^h(t_j) - \sum_{i=1}^{\ell} \langle y^h(t_j), \psi_i \rangle_H \psi_i \right\|_H^2 \right)^{1/2},
\]

\[
\text{ROM Error}(\ell) = \left( \sum_{j=1}^{n_t} \alpha_j \left\| y^h(t_j) - y^\ell(t_j) \right\|_H^2 \right)^{1/2}
\]

are plotted for different POD basis ranks \( \ell \). The chosen trapezoidal weights \( \alpha_j \) have been introduced in (20). We observe that both errors decay rapidly and coincide until the accuracy \( 10^{-12} \), which is already significant smaller than the FE discretisation error. This numerical results reflects the a-priori error estimates presented in [38, Theorem 1.29].

\( \Box \)

**Run 7** (POD for a convection dominated heat equation; cf. [38, Run 2]). Now we consider a more challenging example. We study a convection-reaction-diffusion equation with a source term which is close to be singular: Let \( T, \Omega, y_\circ, H \) and \( V \) be given as in Run 6. The parabolic problem reads as follows

\[
y_t(t, x) - cy_{xx}(t, x) + \beta y_x(t, x) + ay(t, x) = f(t, x) \quad \text{for } (t, x) \in Q,
\]

\[
y(t, x) = 0 \quad \text{for } (t, x) \in \Sigma,
\]

\[
y(0, x) = y_\circ(x) \quad \text{for } x \in \Omega.
\]
We choose the diffusivity $c = 0.025$, the velocity $\beta = 1.0$ that determines the speed in which the initial profile $y_0$ is shifted to the boundary and the reaction rate $a = -0.001$. Finally, $f(t, x) = \mathbb{P}(\frac{1}{1-t}) \cos(\pi x)$ for $(t, x) \in Q$, where $(\mathbb{P}z)(t) = \min(+l, \max(-l, z(t)))$ restricts the image of $z$ on a bounded interval. In this situation, the state solution $y$ develops a jump at $t = 1$ for $l \to \infty$; see the left plot of Figure 14. The right plot of Figure 14 demonstrates that in this case, the decay of the reconstruction residuals and the decay of the errors are much slower as in the right plot of Figure 13. The manifold dynamics of the state solution require an inconvenient large number of POD basis elements. Since the supports of these ansatz functions in general cover the whole domain $\Omega$, the corresponding system matrices of the reduced model are not sparse in contrast to the matrices arising in the finite element Galerkin framework, so the model order reduction cannot be provided efficiently for this example if a good accuracy of the solution function $y^\ell$ is required.

Run 8 (True and exact approximation error; cf. [38, Run 3]). We consider the setting introduced in Run 6 again. The exact solution to (40) does not possess a representation by elementary functions. Hence, the presented reconstruction and reduction errors actually are the residuals with respect to a high-order finite element solution $y^h$. To compute an approximation $y$ of the exact solution $y_{ex}$ we apply a Crank-Nicolson method (with Rannacher smoothing [72]) ensuring $\|y - y_{ex}\|_{L^2(0, T; H)} = \mathcal{O}(\Delta t^2 + h^2) \approx 10^{-5}$. In the context of model reduction, such a state is sometimes called the “true” solution. To compute the FE state $y^h$ we apply the Euler method. In the left plot of Figure 15 we compare the true solution $y_{ex}$ with the associated POD approximation for different values $n_t \in \{64, 128, 256, ..., 8192\}$ of the time integration and for the spatial mesh size $h = 4 \cdot 10^{-3}$. For the norm we apply a discrete $L^2(0, T; H)$-norm as in Run 6. Let us mention that we compute for every $n_t$ a corresponding FE solution $y^h$. We observe that the residuals ignore the errors arising by the application of time and
We can observe that the exact residuals do not decay below a limit of the order \( \ell \). Therefore, the spatial error is dominated by the time error for all values of \( \ell \). We can observe that the exact residuals do not decay below a limit of the order \( \Delta t \). One can observe that for fixed POD basis rank \( \ell \), the residuals with respect to the true solution increase if the high-order accuracy is improved by enlarging \( n \), since the reduced order model has to approximate a more complex system in this case, where the residuals with respect to the exact solution decrease due to the lower limit of stagnation \( \Delta t = 3/(n - 1) \).

Run 9 (Different strategies for a POD basis computation; cf. [38, Run 4]). As we have explained in Section 2.2, let \( Y \in \mathbb{R}^{m \times n} \) denote the matrix of snapshots with rank \( r \), \( W \in \mathbb{R}^{m \times m} \) be the (sparse) spatial weighting matrix consisting of the elements \( \langle \varphi_j, \varphi_i \rangle_X \) (introduced Section 2.3.3) and \( D \in \mathbb{R}^{n \times n} \) be the diagonal matrix containing the nonnegative weighting parameters \( \alpha_j^k \). As we have explained in Section 2.2, the POD basis \( \{\Psi_i\}_{i=1}^\ell \) of rank \( \ell \leq r \) can be determined by providing an eigenvalue decomposition of the matrix \( \tilde{Y} \tilde{Y}^\top = W^{1/2} Y D Y^\top W^{1/2} \in \mathbb{R}^{m \times m} \), one of \( \tilde{Y} \tilde{Y}^\top = D^{1/2} Y^\top W Y D^{1/2} \in \mathbb{R}^{n \times n} \), or a singular value decomposition of \( \tilde{Y} = W^{1/2} Y D^{1/2} \in \mathbb{R}^{m \times n} \). Since \( n \gg m \) in Runs 6-8, the first variant is the cheapest one from a computational point of view. In case of multiple space dimensions or if a second-order time integration scheme such as some Crank-Nicolson technique is applied, the situation is converse. On the other hand, a singular value decomposition is more accurate and stable than an eigenvalue decomposition if the POD elements corresponding to eigenvalues/singular values which are close to zero are taken into account: Since \( \lambda_i = \sigma_i^2 \) holds for all eigenvalues \( \lambda_i \) and
Fig. 16: Run 9 (cf. [38, Figure 1.4]): Singular values $\sigma_i$ using the SVD (SVD Vals) or the eigenvalue decomposition (EIG Vals) and the associated ROM errors (SVD error and EIG Error, respectively) (left); ROM errors for different the choices for $X$, the error norm and the snapshot ensembles (right).

The singular values $\sigma_i$, the singular values are able to decay to machine precision, where the eigenvalues stagnate significantly above. This is illustrated in the left graphic of Figure 16. Indeed, for $\ell > 20$ the EIG-ROM system matrices become singular due to the numerical errors in the eigenfunctions and the reduced order system is ill-posed in this case while the SVD-ROM model remains stable. In the right plot of Figure 16 POD elements are constructed with respect to different scalar products and the resulting ROM errors are compared: $\| \cdot \|_H$-residuals for $X = H$ (denoted by POD(H)), $\| \cdot \|_V$-residuals for $X = V$ (denoted by POD(V)), $\| \cdot \|_V$-residuals for $X = H$ (denoted by POD(H,V)), which also works quite well, the consideration of time derivatives in the snapshot sample (denoted by POD(H,dt)) which allows to apply the a priori error estimate given in [38, Theorem 1.29-2]) and the corresponding sums of singular values (denoted by SV(H,dt)) corresponding to the unused eigenfunctions in the latter case which indeed nearly coincide with the ROM errors.

Notice that in many applications, the quality of the reduced order model does not vary significantly if the weights matrix $W$ refers to the space $X = H$ or $X = V$ and if time derivatives of the used snapshots are taken into account or not. Especially, the ROM residual decays with the same order as the sum over the remaining singular values, independent of the chosen geometrical framework.
5 Optimal snapshot location for computing POD basis functions

The construction of reduced-order models for nonlinear dynamical systems using proper orthogonal decomposition (POD) is based on the information carried of the so-called snapshots. These provide the spatial distribution of the nonlinear system at discrete time instances. Thus, we are interested in optimizing the choice of these time instances in such a manner that the error between the POD-solution and the trajectory of the dynamical system is minimized. This approach was suggested in [55] and was extended in [60] to parametrized elliptic problems. Let us briefly mention some related issues of interest. In [24, 30] the situation of missing snapshot data is investigated and gappy POD is introduced for their reconstruction. An important alternative to POD model reduction is given by reduced-basis approximations; we refer to [67] and references given there. In [35] a reduced model is constructed for a parameter dependent family of large scale problems by an iterative procedure that adds new basis variables on the basis of a greedy algorithm. In the Ph.D thesis [18] a model reduction is sought of a class for a family of models corresponding to different operating stages.

Suppose that we are given the \( n_t \) snapshots \( \{ y(t_j) \} \) \( n_t \subset V \subset X \). The goal is to determine additional \( k \) snapshots at time instances \( \tau = (\tau_1, \ldots, \tau_k) \) with \( 0 \leq \tau_j \leq T, j = 1, \ldots, k \). In [55] we propose to determine \( \tau = (\tau_1, \ldots, \tau_k) \) by solving the optimization problem

\[
\min_{0 \leq \tau_1, \ldots, \tau_k \leq T} \int_0^T \| y(t) - y^f(t) \|^2_V dt, \tag{41}
\]

where \( y \) and \( y^f \) are the solutions to (14) and its POD Galerkin approximation, respectively. Clearly, the definition of the operator \( \mathcal{R} \) given in (4) has to be modified as follows:

\[
\mathcal{R}^\tau \Psi = \sum_{j=1}^{n_t} \alpha_j^\tau \langle y(t_j), \Psi \rangle_X y(t_j) + \sum_{j=1}^{k} \alpha_{n_t+j}^\tau \langle y(\tau_j), \Psi \rangle_X y(\tau_j)
\]

with appropriately modified (trapezoidal) weights \( \alpha_j^\tau, j = 1, \ldots, k + n_t \). Consequently, (41) becomes an optimization problem subject to the equality constraint

\[
\mathcal{R}^\tau \Psi_i = \lambda_i \Psi_i, \quad 1 = 1, \ldots, \ell.
\]

Note that no precautions are made in (41) to avoid multiple appearance of a snapshot. In fact, this would simply imply that a specific snapshot location
should be given a higher weight than others. While the presented approach shows how to choose optimal snapshots in evolution equations, a similar strategy is applicable in the context of parameter dependent systems.

It turns out in our numerical tests carried out in [55] that the proposed criterion is sensitive with respect to the choice of the time instances. Moreover, the tests demonstrate the feasibility of the method in determining optimal snapshot locations for concrete diffusion equations.

**Run 10** (cf. [55, Run 1]). For $T = 1$ let $Q = (0, T) \times \Omega$ and $\Omega = (0, 1) \times (0, 1) \subset \mathbb{R}^2$. For the finite element (FE) triangulation we choose a uniform grid with mesh size $h = 1/40$, i.e., we have 900 degrees of freedoms for the spatial discretization. Then, we consider

$$
y_t(t, x) - c \Delta y(t, x) + \beta \cdot \nabla y(t, x) + y(t, x) = f(x) \quad \text{for} \ (t, x) \in Q,
$$

$$
c \frac{\partial y}{\partial t}(t, x) + q(x)y(t, x) = g(x) \quad \text{for} \ (t, x) \in \Sigma,
$$

$$
y(0, x) = y_\circ(x) \quad \text{for} \ x \in \Omega,
$$

where $c = 0.1$, $\beta = (0.1, -10)^T \in \mathbb{R}^2$,

$$
f(x) = \begin{cases} 4 & \text{for all} \ x = (x_1, x_2) \text{ with } (x_1 - 0.25)^2 + (x_2 - 0.65)^2 \leq 0.05, \\ 0 & \text{otherwise}, \end{cases}
$$

and $y_\circ(x) = \sin(\pi x_1) \cos(\pi x_2)$ for $x = (x_1, x_2) \in \Omega$ (see Figure 17, left plot). Furthermore, we have
\[ q(x) = \begin{cases} 
1 & \text{for } x = (x_1, 1) \text{ with } 0 < x_1 < 1, \\
2 & \text{for } x = (1, x_2) \text{ with } 0 < x_2 < 1, \\
-2 & \text{for } x = (x_1, 0) \text{ with } 0 < x_1 < 1, \\
0 & \text{for } x = (0, x_2) \text{ with } 0 < x_2 < 1. 
\end{cases} \]

\[ g(x) = \begin{cases} 
1 & \text{for } x = (x_1, 1) \text{ with } 0 < x_1 < 1, \\
0 & \text{for } x = (1, x_2) \text{ with } 0 < x_2 < 1, \\
-1 & \text{for } x = (x_1, 0) \text{ with } 0 < x_1 < 1. 
\end{cases} \]

We utilize piecewise linear FE functions. The FE solutions \( y^h = y^h(t, x) \) for \( t = 0.15 \) and \( t = T \) are shown in Figure 17. Next we take snapshots on the fixed uniform time grid \( t_j = j \Delta t, 0 \leq j \leq m, \) with \( m = 10 \) and \( \Delta t = T/m = 0.1. \) The goal is to determine four additional time instances \( \tilde{t} = (\tilde{t}_1, \ldots, \tilde{t}_4) \in [0, T] \) based on a FE approximation for (41). Since the behavior of the solution exhibits more change during the initial time interval \([0, 0.3]\) than later on, we initialize our Quasi-Newton method by the starting value \( \tau^0 = (0.05, 0.15, 0.25, 0.35) \in [0, T]. \) The number of POD ansatz functions is fixed to be \( \ell = 3. \) The corresponding value of the ROM error is approximately 0.1093. The optimal solution is given as \( \bar{\tau} = (0.0092, 0.0076, 0.1336, 0.2882) \in [0, T], \) while the associated ROM error is approximately 0.0165, which is a reduction of about 85%. In Figure 18 we can see that the shapes of the three POD bases are changes significantly from the initial time instances \( \tau^0 \in \mathbb{R}^4 \) to the optimal ones \( \bar{\tau} \in \mathbb{R}^4. \)

POD model order reduction can also be applied to provide surrogate models for high-fidelity components in networks.

### 6 Optimal control with POD surrogate models

Reduced order models are used in PDE-constrained optimization in various ways; see, e.g., [46, 81] for a survey. In optimal control problems it is sometimes necessary to compute a feedback control law instead of a fixed optimal control. In the implementation of these feedback laws models of reduced-order can play an important and very useful role, see [11, 56, 61, 74]. Another useful application is the use in optimization problems, where a PDE solver is part of the function evaluation. Obviously, thinking of a gradient evaluation or even a step-size rule in the optimization algorithm, an expensive function evaluation leads to an enormous amount of computing time. Here, the reduced-order model can replace the system given by a PDE in the objective function. It is quite common that a PDE can be replaced by a five- or ten-dimensional system of ordinary differential equations. This results computationally in a very fast method for optimization.
compared to the effort for the computation of a single solution of a PDE. There is a large amount of literature in engineering applications in this regard, we mention only the papers [63, 66]. Recent applications can also be found in finance using the RB model [71] and the POD model [80, 83] in the context of calibration for models in option pricing.

We refer to the survey article [38], where a linear-quadratic optimal control problem in an abstract setting is considered. Error estimates for the POD Galerkin approximations of the optimal control are proved. This is achieved by combining techniques from [26, 27, 40] and [52, 54]. For nonlinear problems we refer the reader to [46, 70, 81]. However, unless the snapshots are generating a sufficiently rich state space or are computed from the exact (unknown) optimal controls, it is not a-priori clear how far the optimal solution of the POD problem is from the exact one. On the other hand, the POD method is a universal tool that is applicable also to problems with time-dependent coefficients or to nonlinear equations. Moreover, by generating snapshots from the real (large) model, a space is constructed that inhibits the main and relevant physical properties of the state system. This, and its ease of use makes POD very competitive in practical use, despite of a certain heuristic flavor. In this context results for a POD a-posteriori analysis are important. see e.g., [89] and [37, 50, 51, 86, 88, 91, 92]. Using a fairly standard perturbation method it is deduced how far the suboptimal control,
computed on the basis of the POD model, is from the (unknown) exact one. This idea turned out to be very efficient in our examples. It is able to compensate for the lack of a priori analysis for POD methods. Let us also refer to the papers [28, 34, 64], where a-posteriori error bounds are computed for linear-quadratic optimal control problems approximated by the reduced basis method.

Data- and/or simulation-based POD models depend on the data (e.g. initial values, right hand sides, boundary conditions, observations, etc.) which is used to generate the snapshots. If those models are used as surrogates in e.g. optimization problems with PDE constraints the algorithmical framework has to account for this fact with providing mechanisms for accordingly updating the surrogate model during the solution process. Strategies proposed in this context for optimal flow control can be found in e.g. [3, 4, 9, 32, 16]. One of the most mature methods developed in this context is Trust-Region POD proposed in [9], which since then has successfully been applied in many applications.

The quality of the surrogate model highly depends on its information basis, which for snapshot-based methods is given by the snapshot set, compare Section 6. The location of snapshots and also the choice of the initial control in surrogate based optimal control is discussed in [5]. There, techniques from time-adaptive schemes for optimality systems of parabolic optimal control problems are adjusted to compute optimal time locations for snapshots generation in POD surrogate modeling for parabolic optimal control problems.

Concepts for the construction and use of POD surrogate modeling in robust optimal control of electrical machines are presented in [59, 6]. Those problems are governed by nonlinear partial differential equations with uncertain parameters, so that robustness can be achieved by considering a worst case formulation. The resulting optimization problem then is of bilevel structure and POD reduced order models in combination with a posteriori error estimators are used to speed up the numerical computations.

7 Miscellaneous

POD model order reduction can also be applied to provide surrogate models for high-fidelity components in networks. The general perspective is discussed in e.g. [44]. Related research for MOR of electrical networks is reported in e.g. [15, 42, 43]. The basic idea here consists in a decoupling of MOR approaches for the network and high-fidelity components which in general are modeled by PDE systems. For the latter, simulation based POD MOR techniques are used to construct surrogate models which then are stamped back into the (reduced)
electrical network. Details and performance tests are reported e.g. in [41, 43]. A short lecture series with related topics is presented under Hinze-Pilsen. Further contributions to this topic can be found in [14].

Recent trends in data-driven and nonlinear MOR methods are discussed within a YouTube lecture series under Carlberg-YouTube.

References


[66] B. Noack, K. Afanasiev, M. Morzynski, G. Tadmor, and F. Thiele. A


