Model Reduction for Nonlinear Systems: Kernel Methods and Error Estimation

Von der Fakultät für Mathematik und Physik der Universität Stuttgart
zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Daniel Wirtz

aus Minden, Westfalen

Hauptberichter:	Jun.-Prof. Dr. Bernard Haasdonk
Mitberichter:	Prof. Dr. Guido Schneider
Mitberichter:	Prof. Dr. Danny Sorensen

Tag der mündlichen Prüfung: 08. Oktober 2013

Institut für Angewandte Analysis und Numerische Simulation,
Universität Stuttgart, 2013

D 93

Universität Stuttgart

mathematik
physik fakultät8

Impressum
Copyright: © 2013 Daniel Wirtz
All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the permission of the author.

Printed by F.U.T Müllerbader GmbH
Filderstadt, Germany 2013
`http://www.muellerbader.de`

This work is available via Print-On-Demand at
epubli GmbH, Berlin
`http://www.epubli.de`
ISBN 978-3-8442-7671-8

Preface

I always thought writing my thesis would result in a nice linear story and considerably easy to understand as the results built up on each other as the project went along. Unfortunately, merging the fields of model order reduction for nonlinear systems with kernel methods provided too many relevant aspects as could be covered within the three years time. Consequently, a focus had to be set and coincidence played its role towards which topic was dealt with next.

In the beginning, I started with the combination proposed in the project key paper from Philips et al. [131], and focused on a-posteriori error estimates. Due to enough possible choices for itself, those estimators had been developed without including the kernel approximation error in a practical way. Thus, I started working on kernel approximation algorithms and convergence rates, but shortly afterwards the scientific exchange at Rice University opened the perspective to the discrete empirical interpolation method (DEIM) as an alternative for nonlinear approximation. As the previously developed error estimator structure was applicable, *including* proper means of assessing the nonlinear approximation error, my focus shifted over to the alternative combination. Additionally, collaborations within the SimTech cluster of excellence at the University of Stuttgart led me to investigate alternative applications of the considered kernel approximation methods. In the context of multiscale methods, this turned out to be a promising approach in itself.

In the end, I fell upon two general types of models that I had reduction concepts for and also two different classes of nonlinear approximation techniques. This led me to organize this thesis as follows: At first, I introduce the nonlinear approximation techniques in a context separated from dynamical systems. Secondly, the MOR techniques will be introduced independent from each other, both using a suitable sub-part of the previously introduced nonlinear approximation schemes. Lastly, the error estimation results for dynamical systems are presented in consideration of both different nonlinear approximations using kernels and DEIM. My aim is to clarify the connections and ease the navigation inside this work.

Acknowledgements

First of all I have to thank Jun.-Prof. Dr. Bernard Haasdonk for taking me up as his PhD-Student. Over the last three years I enjoyed an uncomplicated working environment with a lot of personal freedom for own ideas and small projects. I also could always rely on constructive feedback regarding publications or presentations.
Thank you Bernard!

I also owe Prof. Dr. Danny Sorensen from Rice University, Houston, Texas a great deal. Not only did I personally enjoy my research visit with him at Rice a lot, but he also was a very constructive adviser and showed me how to neatly write things up. Thanks Danny!

Furthermore, I like to thank for cooperation and constructive feedback:

- R. Schaback from Göttingen for very helpful discussions regarding the kernel greedy algorithms and connections to existing work.
- N. Karajan from the Institute of Applied Mechanics (now working at DynaMore) for fruitful discussions and our work on the multiscale model reduction. I also thank him for assisting me in writing up the introductory part of the human spine model.
- F. Kissling from IANS regarding the porous media applications and experimental data.
- D. Otto from the Institute of Applied Mechanics (CE) for contributions to the IVD experiments and discussions.
- All the proof readers (I. Maier, S. Kaulmann, M. Dihlmann, S. Steinig, J. Neusser, F. Gaspoz, A. Hoang) that made me feel a lot more comfortable handing in my manuscript.
- B. Pferdmenges from the SimTech administration, who was a very patient and resourceful conversation partner regarding anything inside and around SimTech and the graduate school.

Finally, I express my gratitude towards my parents for providing me with everything necessary that no university could ever teach me. Without that, the path to where I am now would have been a lot harder if not impossible.
Thank you!

For financial support, I also would like to thank the German Research Foundation (DFG), the SimTech Excellence Cluster (EXC 310/1) and the Baden-Württemberg Stiftung gGmbH.

Contents

Chapter 1

Introduction

Modeling and simulation of natural or technical processes has developed an important role in both research and industry. Not only can we choose the appropriate clothing depending on the latest weather forecast, simulations have also helped designing new modern cities and traffic guidance systems. They also offer a wide range of applications in engineering, medicine and are even applied at the stock market. This is mainly made possible by the huge increase in computational capacity over the last decades, which allow simulations in previously unthinkable orders of magnitude.

Typically, the considered problems are *parameterized* in nature, e.g. shape, material, diffusivity or initial conditions, yet, as the computational power increases, so do the demands of science and industry for even more complex and detailed simulations. Consequently, there will always be applications that exceed the capacity of current hardware. These applications do not necessarily only comprise of the so called *forward simulations*, where a single (physical) process is simulated for prescribed conditions and parameters. More often, the actual problems of interest are higher-level applications like optimization, uncertainty quantification or inverse problems, which in turn involve many expensive forward simulations for different parameters. Hence, if expensive simulations are repeatedly executed in such a *many-query* context, the overall scientific task might be practically unsolvable due to time or monetary constraints. Furthermore, another difficulty regarding expensive simulations arises in the *real time* context. For example, to increase productivity or efficiency of an industrial process or machine, it can be essential if a live simulation with the current real world data can be run on the fly to provide optimal control parameters. In this setting it is crucial to ensure a rapid response of the simulation at limited hardware capacity.

Before we start with the key concepts, please note that the scientific contexts will be established in their individual chapters and sections.

Facing the above mentioned difficulties, the field of "*model order reduction*" (MOR) has developed along with the increasing model complexity. We remark that in this work, MOR is discussed from a mathematical perspective. Of course, models can also be reduced in complexity conceptionally by, for example, leaving out non-essential parts of the original formulation. However, this approach contradicts the overall paradigm of increasing model complexity to obtain more detailed and realistic results. Instead, mathematical MOR concepts aim at reducing complexity without changing the initial problem setup. They mostly rely on approximation of model components based on analytical features and/or *snapshots*. These are simulations of the full model or subparts thereof, for certain presumably "good" prescribed configurations.

As there are different types of models, there are also different types of MOR approaches for each (class) of them. In this thesis, we shall consider "*dynamical systems*" and "*multiscale models*" with an emphasis on the former. Dynamical systems are a very general class of models, which essentially describe the evolution of a process over time. Dynamical systems themselves can be divided into many different types and those relevant to our context will be introduced in Section 1.3, whereas MOR techniques will be discussed in Chapter 3. Multiscale models, on the other hand, describe (physical) phenomena which involve scales that may differ over several orders of magnitude in time and/or space. The existence of own models for each scale naturally leads to the many-query context discussed above, this and the involved concepts will be introduced in Chapter 4.

Another inevitable difficulty is that many processes or phenomena are nonlinear in nature. Even though it might be possible to describe the effects of interest using suitable nonlinear terms, it is a fundamental problem for MOR techniques to be *efficient* for multiple nonlinear problems as well. Frequently, this problem is approached by "*nonlinear approximation*" techniques, which provide surrogates for nonlinear model components and also satisfy requirements for efficient application of other MOR techniques, cf. Section 3.2.

One such approach was outlined in the work of Philips et al. [131], which considers "*kernel methods*" for nonlinear approximation in the context of MOR for a nonlinear resistor-capacitor model. Unfortunately, the work on this combined topic has not been continued since, which led us to investigate where kernel methods can successfully contribute to MOR of dynamical systems and multiscale models. A first result is an extension of the list of applicable kernels suitable for the proposed method. As a wide range of techniques and applications regarding kernel methods has already been developed, we shall establish the core concepts of kernels in Sections 1.1 and 1.2. Then, Chapter 2 details selected new and existing applications.

Another yet unmentioned aspect is of paramount importance as soon as MOR techniques are applied: Error estimation. Without proper means to assess the approximated model qualitatively and/or quantitatively, there is little use of MOR when it comes to incorrect results or ultimately to liability issues. Consequently, we put an emphasis on (a-posteriori) error estimation of the considered MOR schemes. The results in this area will be discussed in Chapter 5.

Lastly, would like to establish some frequently used notation and complete the introduction by establishing core concepts used throughout this work. Let $\Omega \subseteq \mathbb{R}^d$ be a closed domain unless said otherwise. Also, let $\boldsymbol{G} \in GL_d(\mathbb{R})$ be a symmetric positive definite matrix which induces the scalar product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_G := \boldsymbol{x}^T \boldsymbol{G} \boldsymbol{y}$ and norm $||\cdot||_G$ on $\mathbb{R}^d$. If omitted, we assume the standard

Euclidean case $\boldsymbol{G} = \boldsymbol{I}_d$, where $\boldsymbol{I}_d$ denotes the d-dimensional identity matrix. Further, we shall use MatLab-style indexing in order to refer to subparts or blocks of matrices, e.g. $\boldsymbol{A}[:, 1{:}k]$ refers to the first k columns of a matrix $\boldsymbol{A}$ or $\boldsymbol{A}[1{:}l, 1{:}l]$ denotes the upper left $l \times l$ block of $\boldsymbol{A}$. For any square matrix $\boldsymbol{A}$, we denote the spectrum of $\boldsymbol{A}$ by $\sigma(\boldsymbol{A})$. Similar to MatLab, we will write $[a, b] = f(c)$ for an operation f that returns two values a, b, e.g. $[\boldsymbol{v}, \lambda] = \max \sigma(\boldsymbol{A})$ for the eigenvector $\boldsymbol{v}$ corresponding to the largest eigenvalue λ of a matrix $\boldsymbol{A}$. Finally, we shall use normal lower case letters a to denote scalar values, bold lower case $\boldsymbol{a}$ for vectorial and bold upper case $\boldsymbol{A}$ for matrix-valued quantities throughout this work. Sections 1.1 and 1.2 detail the concepts of kernels and their associated Hilbert spaces and Section 1.3 introduces the considered classes of dynamical systems.

This thesis contains contents which are already published or have been submitted for publication in a very similar form.

Section 2.2 is based upon joint work with B. Haasdonk which has already been published in [183].

Chapter 4 is based upon joint work with N. Karajan and B. Haasdonk which is available as submitted pre-print [185].

Section 5.1 is based upon joint work with B. Haasdonk which has already been published in [182]. The results from Subsection 5.1.3 are based upon joint work with B. Haasdonk which have already been published in [181].

Section 5.2 is based upon joint work with D. C. Sorensen and B. Haasdonk, whose manuscript [186] has recently been accepted.

1.1 Kernels

Basically, any function $K : \Omega \times \Omega \to \mathbb{R}$ can be called a "*kernel*", where the original naming stems from the use of such functions in integral operators

[78]. Relevant in this thesis' context are real-valued kernels operating on real-valued vector spaces, albeit the kernel theory can be transferred to more general settings. Moreover, kernels become more interesting in a wide range of applications if additional properties are assumed. The most commonly assumed property is symmetry, which is given if $K(\boldsymbol{x}, \boldsymbol{y}) = K(\boldsymbol{y}, \boldsymbol{x}) \, \forall \, \boldsymbol{x}, \boldsymbol{y} \in \Omega$. Furthermore, the concept of positive definiteness [19, 178] is crucial when it comes to function spaces.

Definition 1.1.1 (Positive definite kernels)**.** A kernel K is called "*positive definite*", if for all $n \in \mathbb{N}$, $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \subset \Omega$ pairwise distinct and $\boldsymbol{c} \in \mathbb{R}^n \backslash \{0\}$ we have

$$\sum_{i,j=1}^{n} c_i c_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \;>\; 0. \tag{1.1}$$

A kernel is called "*positive semi-definite*", if only ≥ 0 holds in (1.1).

Remark 1.1.2. For the definition of positive definiteness exist two slightly different points of view in literature, which can cause confusion if not handled carefully. In e.g. [13, 152], equation (1.1) is relaxed to $\geq$ instead of $>$, but is still called positive definite (with $>$ being strictly positive definite). With Definition 1.1.1 we follow [178], where the relaxed condition $\geq$ amounts to positive semi-definiteness. While the consequences of different definitions are of mostly technical nature, we chose to stick with positive definiteness using the strict condition, as it is in agreement with the induced properties of kernel matrices, see Definition 1.1.5.

There are two commonly used classes of kernels, which will also be of relevance in this work.

Definition 1.1.3 (Inner product kernels)**.** A kernel K is called "*inner product kernel*", if

$$K(\boldsymbol{x}, \boldsymbol{y}) := \phi\big(\langle \boldsymbol{x}, \boldsymbol{y} \rangle_G\big), \qquad \forall \, \boldsymbol{x}, \boldsymbol{y} \in \Omega,$$

for some scalar function $\phi : \mathbb{R} \to \mathbb{R}$.

Definition 1.1.4 (Translation- and rotation invariant kernels)**.** A kernel K is called "*translation- and rotation invariant*", if

$$K(\boldsymbol{x}, \boldsymbol{y}) := \phi\big(||\boldsymbol{x} - \boldsymbol{y}||_G\big), \quad \forall\, \boldsymbol{x}, \boldsymbol{y} \in \Omega,$$

for some scalar function $\phi : \mathbb{R}_+ \to \mathbb{R}$. Note that rotations w.r.t. the induced $\boldsymbol{G}$-norm ($\boldsymbol{G}$-orthogonal transformations) are implied here.

The latter type of kernel is also commonly referred to as "*radial basis function*" (RBF) kernel, where one argument is considered the *center* (say $\boldsymbol{x}$) and the resulting function $K(\boldsymbol{x}, \cdot) : \Omega \to \mathbb{R}$ is radial-symmetric. For both classes of kernels the symmetry is obvious, and for the RBF case the (semi-) positive definiteness is given if a similar property holds for the inducing ϕ. See e.g. [19, 13] or [178, §6] for more complete characterizations of RBF and positive definite functions.

Definition 1.1.5 (Kernel matrices and vectors)**.** Let K be a kernel. For $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \Omega$ we denote by

$$\boldsymbol{K}_X := \big(K(\boldsymbol{x}_i, \boldsymbol{x}_j)\big)_{ij}, \quad i, j = 1 \ldots n,$$

the *kernel matrix* of K with respect to X. Further we denote for $\boldsymbol{x} \in \Omega$ by

$$\boldsymbol{k}_X(\boldsymbol{x}) := \Big(K(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, K(\boldsymbol{x}_n, \boldsymbol{x})\Big)^T \in \mathbb{R}^n, \tag{1.2}$$

the *kernel vector* of K at $\boldsymbol{x}$ with respect to X. For ease of reading, we will omit the subindices X in $\boldsymbol{K}_X, \boldsymbol{k}_X$ whenever it is clear from context.

From (1.1) it is clear that symmetry or positive (semi-)definiteness of the kernel directly transfers to the respective properties of the kernel matrix (for pairwise distinct points X).

1.1.1 Kernel examples

Typical examples for positive semi-definite inner product kernels are

$$
\begin{aligned}
K_l(\boldsymbol{x}, \boldsymbol{y}) &= \langle \boldsymbol{x}, \boldsymbol{y} \rangle_G && \text{linear kernel,} \\
K_p(\boldsymbol{x}, \boldsymbol{y}) &= (1 + \langle \boldsymbol{x}, \boldsymbol{y} \rangle_G)^p && \text{polynomial kernel of degree } p \in \mathbb{N}.
\end{aligned}
$$

Famous examples for positive definite RBF kernels on $\mathbb{R}^d$ are

$$
\begin{aligned}
K_g(\boldsymbol{x}, \boldsymbol{y}) &= \phi_g\left(\frac{||\boldsymbol{x} - \boldsymbol{y}||_G}{\gamma}\right) && \text{“Gaussian”,} \qquad (1.3)\\
K_w^{d,k}(\boldsymbol{x}, \boldsymbol{y}) &= \phi_w^{d,k}\left(\frac{||\boldsymbol{x} - \boldsymbol{y}||_G}{\gamma}\right) && \text{“Wendland”,} \qquad (1.4)
\end{aligned}
$$

where $\gamma > 0$ is a *dilation* hyperparameter or *kernel width*. They are induced by the RBFs

$$
\phi_g(r) = e^{-r^2}, \qquad \phi_w^{d,k}(r) = (1 - r)_+^{l+k} p_{d,k}(r),
$$

where for $k \in \{1, 2, 3\}$ we have the definitions $l := \lfloor d/2 \rfloor + k + 1$ and

$$
\begin{aligned}
p_{d,1}(r) &= (l + 1)r + 1, \\
p_{d,2}(r) &= \frac{1}{3}(l^2 + 4l + 3)r^2 + (l + 2)r + 1, \\
p_{d,3}(r) &= \frac{1}{15}\big((l^3 + 9l^2 + 23l + 15)r^3 \\
&\quad + (6l^2 + 36l + 45)r^2\big) + (l + 3)r + 1.
\end{aligned}
$$

Here, k is a hyperparameter for *smoothness* in the sense of $K_w^{d,k} \in C^{2k}(\Omega)$ if $\Omega \subseteq \mathbb{R}^{\hat{d}}$, $\hat{d} \leq d$. Note that the Gaussian kernel is positive definite for any $d \in \mathbb{N}$, whereas the polynomials $p_{p,k}$ of the Wendland kernel explicitly depend on d. We refer to e.g. [178, §9] for more details. Figure 1.1 shows different kernels for varying $\boldsymbol{x}$ at $\boldsymbol{y} = 0$. Moreover, there are many more kernels like inverse multiquadrics, thin plate splines [177] or Matern-Kernels [115, 17], where the latter have been getting more attention recently as they are

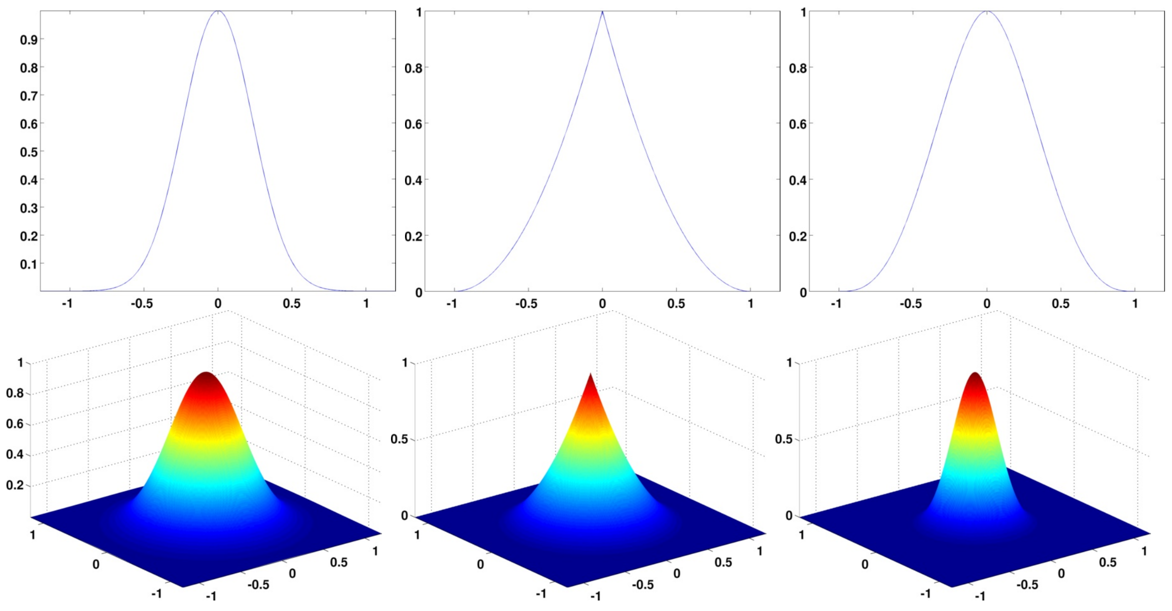


Figure 1.1: Kernel examples. Top row with $d = 1$: $K_g(\gamma = 0.33)$, $K_w^{2,0}$ and $K_w^{1,1}$. Bottom row with $d = 2$: $K_g(\gamma = 0.49)$, $K_w^{2,0}$ and $K_w^{3,3}$. We use $\gamma = 1$ for all K_w.

the reproducing kernels for certain Sobolev spaces and have controllable smoothness like the Wendland kernels. Actually, the correct choice of kernel is often crucial for the successful application of the numerical method they are applied within. Unfortunately, in many cases a satisfying answer about the "best" kernel choice cannot be readily given. For more examples and details on kernels we refer to e.g. [62, 58, 147, 178, 152, 13].

Remark 1.1.6. Despite the "direct" kernel definitions as e.g. radial or scalar product kernels, basic algebraic combinations (sums with positive weights, products) of kernels are again kernels. We will not go into detail here, however, we will utilize this by using a product of three possibly different kernels in the context of time- and parameter-dependent functions, cf. Section 1.3.2 and Section 3.2.1.

1.2 Reproducing kernel Hilbert spaces

In this work we will not only deal with kernels but also with certain Hilbert spaces that are strongly connected to them. In the following we shall shortly recall the essential aspects along the lines of [178, §10] and refer to this work for more background. The type of Hilbert spaces we consider are called reproducing kernel Hilbert spaces (RKHS). They have a characteristic relation to a "*reproducing kernel*" [8], which we define next.

Definition 1.2.1 (Reproducing kernel)**.** Let $K : \Omega \times \Omega \to \mathbb{R}$ be a kernel and $\mathcal{H}$ a real Hilbert space of functions $f : \Omega \to \mathbb{R}$ with scalar product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. Then K is called a "*reproducing kernel*" if

$$K(\boldsymbol{x}, \cdot) \in \mathcal{H} \quad \forall\, \boldsymbol{x} \in \Omega, \tag{1.5}$$

$$f(\boldsymbol{x}) = \langle f, K(\boldsymbol{x}, \cdot) \rangle_{\mathcal{H}} \quad \forall\, f \in \mathcal{H}, \boldsymbol{x} \in \Omega. \tag{1.6}$$

The elements $K(\boldsymbol{x}, \cdot)$ are also known as *kernel translates* and the property (1.6) is called "*reproducing property*" of K in $\mathcal{H}$. If a reproducing kernel K exists for $\mathcal{H}$, it is unique and positive semi-definite, cf. [178].

Note that the kernel translates $K(\boldsymbol{x}, \cdot)$ are the Riesz-representative of the point-evaluation functionals $\delta_{\boldsymbol{x}} \in \mathcal{H}^*$.

However, as explicit representations of reproducing kernels or even proof of existence are not available for Hilbert spaces in general, one is interested in how to find the associated Hilbert space for a given kernel. In fact, for any symmetric, positive definite (s.p.d.) kernel there exists such a Hilbert space, also referred to as the "*native space*" of a kernel K. At first we will look at linear vector spaces, as they will play an important role in subsequent parts of this work.

Definition 1.2.2 (Kernel vector spaces)**.** Let $K : \Omega \times \Omega \to \mathbb{R}$ be a kernel and $X \subset \Omega$. Then we denote by

$$\mathcal{H}^X := \langle \{ K(\boldsymbol{x}, \cdot) \mid \boldsymbol{x} \in X \} \rangle$$

the $\mathbb{R}$-vector space spanned by all translates $K(\boldsymbol{x}, \cdot)$, where $\langle \cdot \rangle$ is a shorthand for the span operation. The elements or functions

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} c_i K(\boldsymbol{x}_i, \boldsymbol{x}) \in \mathcal{H}^X, \quad c_i \in \mathbb{R}, \tag{1.7}$$

from $\mathcal{H}^X$ are called *kernel expansions* and c_i *expansion coefficients.* For a finite $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, $\mathcal{H}^X$ is an at most n-dimensional $\mathbb{R}$-vector space spanned by K over X.

Note that, for general $X \subset \Omega$ with $|X| = n$, we have $\dim(\mathcal{H}^X) = n$ only if K is positive definite, as kernel translates $K(\boldsymbol{x}, \cdot), K(\boldsymbol{y}, \cdot)$ for $\boldsymbol{x} \neq \boldsymbol{y}$ can be linearly dependent for semi-definite kernels. Now, equipping $\mathcal{H}^\Omega$ with a suitable scalar product and considering its completion, we can construct the native space for K.

Lemma 1.2.3 (Kernel pre-Hilbert spaces). *Let $K : \Omega \times \Omega \to \mathbb{R}$ be a s.p.d. kernel. Further define on $\mathcal{H}^\Omega$ the bilinear form*

$$\left\langle \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}_i, \cdot), \sum_{i=1}^{m} \beta_i K(\boldsymbol{y}_i, \cdot) \right\rangle_{\mathcal{H}^\Omega} := \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j K(\boldsymbol{x}_i, \boldsymbol{y}_j),$$

where $\boldsymbol{x}_i, \boldsymbol{y}_j \in \Omega$ and $\alpha_i, \beta_j \in \mathbb{R}, i = 1 \ldots n, j = 1 \ldots m$, define two functions from $\mathcal{H}^\Omega$. Then $\langle \cdot, \cdot \rangle_{\mathcal{H}^\Omega}$ is an inner product and $\mathcal{H}^\Omega$ a pre-Hilbert space with reproducing kernel K.

Proof. The symmetry is obvious and for any $f = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}_i, \cdot) \neq 0$ we have

$$\langle f, f \rangle_{\mathcal{H}} = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) > 0$$

by (1.1, p.9) and $f \equiv 0$ implies $\langle f, f \rangle_{\mathcal{H}} = 0$. Moreover,

$$\langle f, K(\boldsymbol{y}, \cdot) \rangle_{\mathcal{H}} = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{y}) = f(\boldsymbol{y})$$

establishes the reproducing property of K on $\mathcal{H}^\Omega$. □

Remark 1.2.4. The construction in Lemma 1.2.3 also works for semi-definite kernels K. However, in that case the representation of any $f \in \mathcal{H}^\Omega$ is not unique anymore and the proof would get more technical.

Definition 1.2.5 (Reproducing kernel Hilbert spaces)**.** Let $K : \Omega \times \Omega \to \mathbb{R}$ be a s.p.d. kernel. We then denote by

$$\mathcal{H} = \overline{\mathcal{H}^\Omega} = \overline{\langle \{K(\boldsymbol{x}, \cdot) \mid \boldsymbol{x} \in \Omega\} \rangle}$$

the "*reproducing kernel Hilbert space*" or native space of K induced over Ω.

Note that function evaluations for elements of the completion can be defined using the reproducing property and continuity of the scalar product; again we refer to [178, §10] for more details. The following definition introduces the vectorial RKHS we will be dealing with later.

Definition 1.2.6 (Vectorial Hilbert Spaces)**.** Let $q \in \mathbb{N}$. Then we denote by

$$\mathcal{H}^q := \{\boldsymbol{f} : \Omega \to \mathbb{R}^q \mid f_j \in \mathcal{H}, j = 1 \dots q\}$$

the function space of vectorial functions from $\mathcal{H}$ which we equip with the canonical scalar product and norm

$$\langle \boldsymbol{f}, \boldsymbol{g} \rangle_{\mathcal{H}^q} := \sum_{j=1}^{q} \langle f_j, g_j \rangle_{\mathcal{H}}, \qquad ||\boldsymbol{f}||_{\mathcal{H}^q} = \sqrt{\langle \boldsymbol{f}, \boldsymbol{f} \rangle_{\mathcal{H}^q}} = \sqrt{\sum_{j=1}^{q} ||f_j||^2_{\mathcal{H}}}.$$

Finally, we will show some properties of RKHS functions that will be relevant at a later stage. The following Lemma shows how smoothness of a kernel inherits to the RKHS functions. The proof is along the lines of [178, §10.6] and/or [160, Lemma 4.34].

Lemma 1.2.7 (Derivatives of RKHS functions)**.** *If* $K \in C^2(\Omega \times \Omega)$*, then* $\frac{\partial K}{\partial x_i}(\boldsymbol{x}, \cdot) \in \mathcal{H} \; \forall \, \boldsymbol{x} \in \Omega, i = 1 \ldots d$ *and we have*

$$\frac{\partial f}{\partial x_i}(\boldsymbol{x}) = \left\langle f, \frac{\partial K}{\partial x_i}(\boldsymbol{x}, \cdot) \right\rangle_{\mathcal{H}} \qquad \forall \, \boldsymbol{x} \in \Omega, \; i = 1 \ldots d.$$

For the remainder of this work, let K be a s.p.d. and normalized ($K(\boldsymbol{x}, \boldsymbol{x}) = 1 \; \forall \, \boldsymbol{x} \in \Omega$) kernel on Ω with native spaces $\mathcal{H}$ or $\mathcal{H}^q$ unless defined otherwise.

1.3 Dynamical systems

In the following we shall shortly establish the notion of dynamical systems and detail the variants used in this work. Basically, a dynamical system is a concept describing the evolution of a state or position of a considered object over time. The key feature is that the governing law is given by a relation that only tells the *change* of the state given a current state. Hence, knowing an "initial" state, the dynamical system can be "solved" by repeatedly evaluating the relation and advancing in time. While there are discrete or infinitely dimensional versions of dynamical systems coming with each their own field of research, our focus shall be on finite dimensional time-continuous dynamical systems. Mathematically, this comes down to an "*ordinary differential equation*" (ODE) or a system thereof, depending on the size or "*degrees of freedom*" (DoF) of the considered system. The most basic dynamical system is given by the equations

$$\boldsymbol{x}'(t) = \boldsymbol{A}\boldsymbol{x}(t), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0, \qquad t \in [0, T]. \tag{1.8}$$

Here, $\boldsymbol{x}(t) \in \mathbb{R}^d$ denotes the *system state* at time t and $d \in \mathbb{N}$ denote the number of DoFs or *system dimension*. The system starts at time $t = 0$ at the *initial state* $\boldsymbol{x}_0$ and runs until some time $t = T$ is reached. An entire solution $\boldsymbol{x}(t), t \in [0, T]$ is called *trajectory* of the dynamical system. The key element of the above system is the matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$, which determines the system's dynamics or behaviour. Since $\boldsymbol{A}$ corresponds to a linear map, the system (1.8) is called a "*linear dynamical system*". However, the dynamical systems arising in applications are usually more complicated. We shall introduce the most common forms that frequently appear and are also of interest in this work.

In applications like mechanics or control [77], frequently the following type of linear systems is considered:

$$\boldsymbol{x}'(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0, \tag{1.9a}$$

$$\boldsymbol{w}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t). \tag{1.9b}$$

This linear system is augmented by an *input* or *control* term $\boldsymbol{B}\boldsymbol{u}(t)$ with $\boldsymbol{u}(t) : [0, T] \to \mathbb{R}^l$ and $\boldsymbol{B} \in \mathbb{R}^{d\times l}$. Further we have the "*system output*" $\boldsymbol{w}(t) \in \mathbb{R}^k$ composed by an output mapping $\boldsymbol{C}\boldsymbol{x}(t), \boldsymbol{C} \in \mathbb{R}^{k\times d}$ and a control forwarding term $\boldsymbol{D}\boldsymbol{u}(t), \boldsymbol{D} \in \mathbb{R}^{k\times l}$. The input $\boldsymbol{u}(t)$ can be thought of as an "external influence" on the system and the output could be the values of interest in the application at hand. However, in this work we will focus mainly on the part (1.9a) and will thus omit the control forwarding term $\boldsymbol{D}$ and mention any output mapping $\boldsymbol{C}$ only where appropriate.

1.3.1 Nonlinear dynamical systems

Even though many processes can be described by linear systems, the vast majority of real-world problems leads to systems that involve nonlinear dynamics. Basically, a nonlinear dynamical system can be written as

$$\boldsymbol{x}'(t) = \boldsymbol{f}(\boldsymbol{x}(t)) + \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0. \tag{1.10}$$

Here, $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^d$ is a general nonlinear function that describes the system's inner dynamics. Note here that (1.10) could of course also be merged notationally into a setting only involving a function $\boldsymbol{g}$ by considering

$$\boldsymbol{g}(\boldsymbol{x},t) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{f}(\boldsymbol{x},t) + \boldsymbol{B}\boldsymbol{u}(t)$$

as one function. However, the refined structure allows for more insight and thus numerical methods (e.g. solvers) to treat it.

At this stage no further assumptions on $\boldsymbol{f}$ are made to allow for an as general description as possible. Of course, subsequent chapters will impose more structure or regularity on $\boldsymbol{f}$ in order to allow error estimation, for example.

Remark 1.3.1. Since the context of this work is model reduction, we silently assume existence and uniqueness is given for the solutions of the considered systems within the scope of their definition. Of course, most nonlinear functions $\boldsymbol{f}$ will at least be Lipschitz-continuous on Ω in order to guarantee existence and uniqueness via the Picard-Lindelöf theorem, for example.

1.3.2 Affine-parametric nonlinear dynamical systems

Finally, we introduce the most general nonlinear dynamical system structure considered in this work. In addition, here the system components may be time- and parameter dependent for parameters $\boldsymbol{\mu} \in \mathcal{P}$ of some parameter domain $\mathcal{P} \subset \mathbb{R}^p$. In detail, the matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and the input $\boldsymbol{x}_0$ are assumed to be affine-linear in the sense of

$$\boldsymbol{A}(t,\boldsymbol{\mu}) = \sum_{i=1}^{Q_A} \theta_i^A(t,\boldsymbol{\mu})\boldsymbol{A}_i, \tag{1.11}$$

with $Q_A \in \mathbb{N}$, matrices $\boldsymbol{A}_i \in \mathbb{R}^{d\times d}$ and scalar coefficient functions $\theta_i^A : [0,T]\times\mathcal{P} \to \mathbb{R}$ as proposed in [74]. Analogous definitions hold for the components $\boldsymbol{B}(t,\boldsymbol{\mu}), \boldsymbol{C}(t,\boldsymbol{\mu})$ and $\boldsymbol{x}_0(\boldsymbol{\mu})$ with $\boldsymbol{B}_i \in \mathbb{R}^{d\times l}, i = 1\ldots Q_B, \boldsymbol{C}_i \in \mathbb{R}^{k\times d}, i = 1\ldots Q_C$ and $\boldsymbol{x}_i^0 \in \mathbb{R}^d, i = 1\ldots Q_0$, respectively, where in the latter case only parameter-dependent functions $\theta_i^0(\boldsymbol{\mu})$ are used. The complete nonlinear *affine-parametric* dynamical system then reads as

$$\boldsymbol{x}'(t) = \boldsymbol{f}(\boldsymbol{x}(t), t, \boldsymbol{\mu}) + \boldsymbol{A}(t, \boldsymbol{\mu})\boldsymbol{x}(t) + \boldsymbol{B}(t, \boldsymbol{\mu})\boldsymbol{u}(t), \tag{1.12a}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_0(\boldsymbol{\mu}), \tag{1.12b}$$

$$\boldsymbol{w}(t) = \boldsymbol{C}(t, \boldsymbol{\mu})\boldsymbol{x}(t), \tag{1.12c}$$

for $\boldsymbol{f} : \mathcal{D} \rightarrow \Omega$ with $\mathcal{D} := \Omega \times [0, T] \times \mathcal{P}$. The key idea behind this affine structure is that an efficient model reduction scheme [73] can be devised, which allows to split the reduction process into a so-called "offline" and "online" phase. Therein, the quantities Q_A etc. are assumed to be of small/moderate size and the coefficient functions of simple (e.g. algebraic) complexity, allowing a fast reduced simulation. We will detail this process later in Section 3.3.

Chapter 2

Nonlinear approximation

In this second chapter we will deal with methods on how to approximate nonlinear functions. The intention behind this is to provide means to approximate the nonlinear parts $\boldsymbol{f}(\boldsymbol{x})$ or $\boldsymbol{f}(\boldsymbol{x}, t, \boldsymbol{\mu})$ occurring in the dynamical systems (1.10, p.17) or (1.12a, p.19), respectively. Ultimately, the goal is an accelerated simulation time of the resulting reduced order models. However, as the methodology can be applied independently of this setting, we shall leave the domain of dynamical systems for this chapter. The only requirement we shall carry over from the dynamical systems setting is the need for *vectorial approximation*. This can of course always be obtained via a component-wise approximation, but truly vectorial approximation methods have their own advantages, e.g. regarding overall sparsity. Moreover, we shall require *sampling based* approximation methods, whose choice can be motivated as follows: Quite frequently, $\boldsymbol{f}$ has to be treated as a black-box, as further analytical insight is impossible due to a high complexity, or even more trivially, because $\boldsymbol{f}$ is provided by a third party software or tool. Hence, sampling based methods assume to only have the possibility of evaluating $\boldsymbol{f}$ for a given $\boldsymbol{x} \in \Omega$ or, even worse, only have a set of inputs and evaluations of $\boldsymbol{f}$ to begin with. Mathematically, this means to compute or have $n \in \mathbb{N}$ samples of *training inputs* $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \subseteq \Omega$ and *training outputs* $Y = \{\boldsymbol{y}_i := \boldsymbol{f}(\boldsymbol{x}_i) \mid i = 1 \dots n\} \subseteq \mathbb{R}^q$, respectively. Note that for the dynamical system setting we have $q = d = \dim(\Omega)$, but we consider independent input and output dimensions in the following. The goal then is to learn the behaviour of $\boldsymbol{f}$ only using the *training data* $D := \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \mid i = 1 \dots n\}$.

Consequently, the considered approximation methods can be divided into two stages. At first, a computationally expensive "offline" stage, where X and Y are computed and the training algorithms are executed. Second, a fast "online" stage, where the resulting approximant is evaluated. We will not go into details regarding training data generation, but instead refer to Section 3.3 for a general description of the offline/online concept in the context of model reduction. However, we would like to point out that it is vital to have large enough X, Y to sufficiently describe $\boldsymbol{f}$, i.e. X and Y are *statistically representative*. Even the best learning algorithm will not perform well on poor training data.

Now, of course, there is a vast range of techniques for approximation of nonlinear functions. Most of them adhere to the same basic principle, where the functional form of the approximant is pre-selected with some yet undetermined coefficients. The most basic and well known approaches are (multivariate) polynomial interpolation or splines [165, 29], where, among others, the polynomial chaos expansion (PCE) [180, 123] is a popular relative. However, as will become clear later in Section 3.2, we have some additional structural requirements for the approximants regarding the efficient projectability. We shall introduce two different approaches of (vectorial) nonlinear approximation that will allow just that.

At first, as mentioned in the introduction, we consider "*kernel methods*" for nonlinear approximation. Kernel methods are probably best-known from the field of machine learning [152, 14], especially in the context of pattern recognition [157]. Other well-known applications are scattered data approximation [178, 109] or orthogonal least squares [25, 27]. As the range of applications and amount of work done in this area is huge, we refer to [149] for a practical guide on kernel methods and their applications and [58] for an excellent current review of approximation methods using (positive definite) kernels. Now, the kernel methods we consider are support vector machines (SVM) in Section 2.1 and vectorial kernel-greedy algorithms in Section 2.2, where we will provide the scientific context in the respective sections.

As second approach, we will shortly recall the "*Discrete empirical interpolation method*" (DEIM) in Section 2.3, which was introduced in [23] and goes back to [10].

2.1 Support vector regression

To clarify terminology, the general type of algorithm behind support vector regression (SVR) is a SVM, which can actually be used to perform classification or regression. It is often used synonymously in the context of classification, but dependent on the usage of the result it is either called support vector classification or SVR.

The classic way of motivating support vector machines comes from linear regression, which is done for example in [152, §9.1] or [32]. Those formulations contain an offset term b, which is inconvenient e.g. regarding analysis in $\mathcal{H}$. More modern approaches like in [160, §11.1] formulate SVMs without the offset term and take a statistical learning theory point of view, which we will use in this thesis, too. After introducing the background of SVMs and the standard solution approach, we will detail an adopted formulation without offset for the regression case along the lines of [162].

Even though we ultimately seek true vectorial approximations, the current SVM theory focuses on cases where one might have vectorial input spaces, but considers scalar output. As the theory is already involved enough for the scalar case, we will detail our theory for this case and consider component-wise extensions for the vectorial setting. Consequently, we shall assume $f : \Omega \to \mathbb{R}$ for this section.

2.1.1 SVM principles

In the following we shall give a short informal introduction to statistical learning theory. Most importantly, in statistical learning theory we step back from the deterministic relation $y = f(\boldsymbol{x})$ for some unknown function f, but instead consider the relation $\boldsymbol{x} \to y$ between an input $\boldsymbol{x}$ and output y to be defined by a probability distribution $P(\boldsymbol{x}, y)$. Now, the "*learning goal*" is to find a function $\hat{f}$ so that $\hat{f}(\boldsymbol{x}) \approx y$ for arbitrary $\boldsymbol{x}$. This perspective has two imminent consequences. At first, as $\boldsymbol{x}$ is also generated by the unknown P, we assume to have no control or influence on the way the input values are chosen, which makes this approach applicable for a very general sampling setting. Second, the fact that the response y to a certain $\boldsymbol{x}$ is given by a probability $P(\cdot|\boldsymbol{x})$ takes into account that the relation might not be deterministic in the sense that the output does not solely depend on a single input $\boldsymbol{x}$. Thus, this approach enables the assignment $\boldsymbol{x} \to \hat{f}(\boldsymbol{x})$ to depend on more data than just the input $\boldsymbol{x}$, even though in the end, the result will be a deterministic $\hat{f}$ to describe $\boldsymbol{x} \to y$.

Of course, we need some means to assess the quality of any $\hat{f}$ with respect to the distribution P. These means are given by the concept of "*loss functions*"

$$\begin{aligned} L : \Omega \times \mathbb{R} \times \mathbb{R} &\to \mathbb{R}_+, \\ (\boldsymbol{x}, y, t) &\mapsto L(\boldsymbol{x}, y, t), \end{aligned}$$

which measure the discrepancy between an output y and the prediction $t = \hat{f}(\boldsymbol{x})$ for a given input $\boldsymbol{x}$. Then, given a loss function L, statistical learning methods target to minimize the *expected risk*

$$\hat{f} = \arg\min_f R_{L,P}(f), \qquad R_{L,P}(f) := \int\limits_{\Omega\times\mathbb{R}} L(\boldsymbol{x}, y, f(\boldsymbol{x}))dP(\boldsymbol{x}, y),$$

where $R_{L,P}$ denotes the dependency on the loss function L and the distribution P. As we only have the training data D representing P, the *empirical* probability distribution

$$P_D(\boldsymbol{x}, y) = \frac{1}{n} \sum_{i=1}^{n} \delta_{(\boldsymbol{x}_i, y_i)}(\cdot, \cdot)$$

is used to obtain the "*expected empirical risk*"

$$R_{L,P_D}(f) := \int\limits_{\Omega \times \mathbb{R}} L(\boldsymbol{x}, y, f(\boldsymbol{x})) dP_D(\boldsymbol{x}, y) = \frac{1}{n} \sum_{i=1}^{n} L(\boldsymbol{x}_i, y_i, f(\boldsymbol{x}_i)). \quad (2.1)$$

2.1.1.1 Applied loss function

In our applications, for some $\epsilon > 0$, we shall employ the "*ϵ-insensitive loss*"

$$L(y, f(\boldsymbol{x})) = |y - f(\boldsymbol{x})|_\epsilon := \max\{0, |y - f(\boldsymbol{x})| - \epsilon\}.$$

Figure 2.1 shows an example for the ϵ-insensitive loss using $\epsilon = 0.2$. This

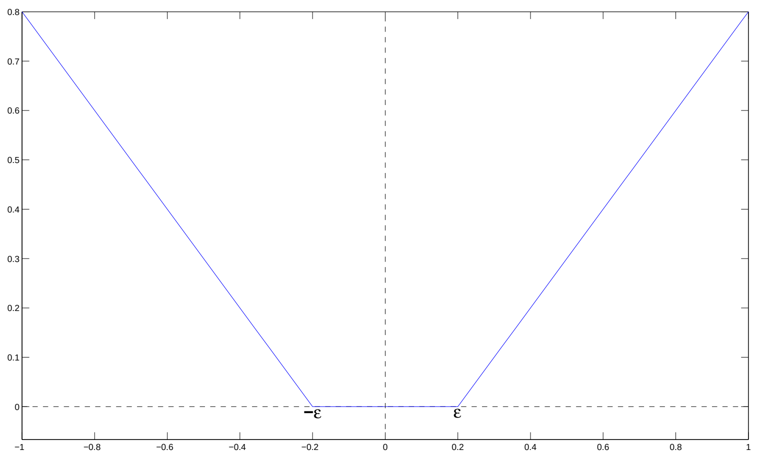


Figure 2.1: Illustration of the ϵ-insensitive loss function for $\epsilon = 0.2$

choice causes SVM solutions to be sparse in a certain sense, which we will discuss later. The resulting SVM is also known as called "*ϵ-SVR*". Other popular loss functions are e.g. the Hinge-Loss $L(\boldsymbol{x}, y, f(\boldsymbol{x})) = \max\{0, 1 - yf(\boldsymbol{x})\}$ for functions $f : \Omega \to [-1, 1]$ or the Least-Squares loss $L(\boldsymbol{x}, y, f(\boldsymbol{x})) = (y - f(\boldsymbol{x}))^2$.

Remark 2.1.1. Note that the ϵ-insensitive loss does not depend directly on the input $\boldsymbol{x}$. In fact, there are different ways of designing loss functions, where we distinguish between supervised and unsupervised learning in general. Supervised learning includes the output y in the loss function, which makes the SVM aware of the desired output. Regression and classification belong to supervised learning methods. The classical example for unsupervised learning are clustering algorithms, where no output label y is known a-priori and the relation f has to be learned only using $\boldsymbol{x}$ and $f(\boldsymbol{x})$. We refer to e.g. [160, 152] for more background.

Now, there is one more aspect to take into account. Most obviously, we somehow need to decide which function space $\hat{f}$ shall come from. There are many possibilities, of course, but in this work we use the RKHS $\mathcal{H}$ with associated kernel K as introduced in Section 1.2. Moreover, the following section motivates why we should impose additional requirements to the empirical risk minimizers $\hat{f}$, even when a presumably suitable function space $\mathcal{H}$ is chosen.

2.1.1.2 Regularization

There is one imminent danger hidden in minimizing (2.1) directly. Assume $\mathcal{H}$ allows to choose a (probably fairly degenerated) $\hat{f} \in \mathcal{H}$ which reproduces the training data D and is nearly zero anywhere else. This $\hat{f}$ will likely be a candidate for a minimal empirical risk, but will obviously be a bad generalization with respect to P. This means, for $(\boldsymbol{x}, y) \notin D$, we have to expect high loss $L(\boldsymbol{x}, y, \hat{f}(\boldsymbol{x}))$ or, in other words, the probability $P\big(\hat{f}(\boldsymbol{x})\big|\boldsymbol{x}\big)$ will most likely be small. So if we do not restrict the possible functions somehow, we will encounter a phenomenon called *overfitting*.

Hence, SVM's actually solve the "*regularized empirical risk minimization*" problem

$$\hat{f} = \arg\min_{f\in\mathcal{H}} R_{L,P_D,\lambda}(f), \qquad R_{L,P_D,\lambda}(f) := \lambda\,||f||_{\mathcal{H}}^2 + R_{L,P_D}(f), \tag{2.2}$$

for some *regularization parameter* $\lambda > 0$. Note that, additionally and convex loss functions to ensure solution uniqueness). In the following we shortly motivate how this technique influences the number of admissible minimizing functions analogous to [160, §1.2]. Suppose $\hat{f}$ is a minimizer of (2.2). Then for $g \equiv 0$ we obtain with ϵ-insensitive loss

$$R_{L,P_D}(g) = \frac{1}{n}\sum_{i=1}^{n} \max\{0, |y_i - g(\boldsymbol{x}_i)| - \epsilon\} \leq \frac{1}{n}\sum_{i=1}^{n} |y_i| =: C.$$

This gives

$$\begin{aligned}\lambda \left|\left|\hat{f}\right|\right|_{\mathcal{H}}^2 &\leq \lambda \left|\left|\hat{f}\right|\right|_{\mathcal{H}}^2 + R_{L,P_D}(\hat{f}) = R_{L,P_D,\lambda}(\hat{f}) \\ &\leq R_{L,P_D,\lambda}(g) \leq \lambda \left|\left|g\right|\right|_{\mathcal{H}}^2 + R_{L,P_D}(g) \leq C,\end{aligned}$$

which means $\|\hat{f}\|_{\mathcal{H}} \leq \sqrt{\frac{C}{\lambda}}$. Consequently, $\hat{f}$ is also a solution to the minimization (2.2) when replacing $\mathcal{H}$ by the subspace

$$\mathcal{H}_\lambda := \left\{ f \in \mathcal{H} \;\middle|\; ||f||_{\mathcal{H}} \leq C^{\frac{1}{2}}\lambda^{-\frac{1}{2}} \right\} \subset \mathcal{H}.$$

So varying λ in fact controls the amount of effectively considered functions used within the minimization, or equivalently, functions with $||f||_{\mathcal{H}} > C^{\frac{1}{2}}\lambda^{-\frac{1}{2}}$ are never solutions of (2.2). Informally speaking, functions like the "point-wise approximation" of D tend to have higher norms (w.r.t. the considered RKHS norms) than more smooth ones, which allow better generalization. In a way, finding the right λ corresponds to finding the best trade-off between approximation quality on D and ability of generalization.

2.1.2 Optimization problem for ϵ-SVR

In this section we will derive the necessary notation and recall already established means to compute a solution of the ϵ-SVR or minimization problem (2.2). At first, so-called *slack variables* are introduced to obtain an equivalent, but algorithmically more convenient formulation. Proposition 2.1.2

states this equivalent formulation also employed in e.g. [152].

Proposition 2.1.2 (Equivalent constrained minimization). *Let $\lambda, \epsilon > 0$ and define $C := \frac{1}{2\lambda n}$. Then the unconstrained regularized empirical risk minimization problem (2.2, p.26)*

$$\min_{f\in\mathcal{H}} R_{L,P_D,\lambda}(f) = \min_{f\in\mathcal{H}} \lambda \,||f||^2_{\mathcal{H}} + \frac{1}{n}\sum_{i=1}^{n} |y_i - f(\boldsymbol{x}_i)|_\epsilon$$

is equivalent to the constrained minimization problem

$$\begin{aligned} \min_{f\in\mathcal{H},\boldsymbol{\xi}^\pm\in\mathbb{R}^n} & \frac{1}{2}\,||f||^2_{\mathcal{H}} + C\sum_{i=1}^{n}\left(\xi_i^+ + \xi_i^-\right) \\ \text{s.t.} \quad & 0 \geq -\boldsymbol{\xi}^\pm \\ & 0 \geq y_i - f(\boldsymbol{x}_i) - \epsilon - \xi_i^+, \quad i = 1\dots n, \\ & 0 \geq -(y_i - f(\boldsymbol{x}_i)) - \epsilon - \xi_i^-, \quad i = 1\dots n, \end{aligned} \tag{2.3}$$

where $\boldsymbol{\xi}^\pm \in \mathbb{R}^n$ are slack variables *[152].*

This formulation "transfers" the empirical risk term into the constraints and instead tries to keep deviations or violations from that constraint as small as possible. Now, the primal problem (2.3) is classically transformed into a dual problem, which is carried out in the next Proposition. The LaGrangian formalism used therein is standard [32, 152, 160], and we state it here for completeness due to the modified variant without offset.

Proposition 2.1.3 (Dual problem solution of ϵ-SVR). *Let the conditions of Proposition 2.1.2 hold, let K denote the reproducing kernel inducing $\mathcal{H}$ and $\boldsymbol{K} = \boldsymbol{K}_X$ be the kernel matrix for the training inputs X. Then the minimizer $\hat{f}$ of the primal problem* (2.3) *is given by the expansion*

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n}(\alpha_i^+ - \alpha_i^-)K(\boldsymbol{x}_i, \boldsymbol{x}), \tag{2.4}$$

where $\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^- \in \mathbb{R}^n$ are the solution of

$$\max_{\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^- \in \mathbb{R}^n} \quad W(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) \tag{2.5a}$$

$$\textit{s.t.} \quad 0 \le \alpha_i^{+,-} \le C, \quad i = 1 \dots n, \tag{2.5b}$$

$$0 = \alpha_i^+ \alpha_i^-, \quad i = 1 \dots n, \tag{2.5c}$$

with

$$\begin{aligned} W(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) = &- \frac{1}{2} \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)^T \boldsymbol{K} \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) \\ &- \epsilon \sum_{i=1}^n (\alpha_i^+ + \alpha_i^-) + y^T \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right). \end{aligned} \tag{2.6}$$

Proof. We can write $f(\boldsymbol{x}_i) = \langle f, K(\boldsymbol{x}_i, \cdot)\rangle_{\mathcal{H}}$ due to the reproducing properties of $\mathcal{H}$ and establish the Lagrangian as

$$\begin{aligned} &\mathcal{L}(f, \boldsymbol{\xi}^+, \boldsymbol{\xi}^-, \boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-, \boldsymbol{\eta}^+, \boldsymbol{\eta}^-) \\ &= \frac{1}{2} ||f||_{\mathcal{H}}^2 + C \sum_{i=1}^n \left(\xi_i^+ + \xi_i^-\right) + \sum_{i=1}^n \alpha_i^+ \left(y_i - \langle f, K(\boldsymbol{x}_i, \cdot)\rangle_{\mathcal{H}} - \epsilon - \xi_i^+\right) \\ &+ \sum_{i=1}^n \alpha_i^- \left(-(y_i - \langle f, K(\boldsymbol{x}_i, \cdot)\rangle_{\mathcal{H}}) - \epsilon - \xi_i^-\right) - \sum_{i=1}^n \left(\eta_i^+ \xi_i^+ + \eta_i^- \xi_i^-\right), \end{aligned}$$

with Lagrange multipliers $\boldsymbol{\alpha}^\pm, \boldsymbol{\eta}^\pm \ge 0$. According to the Karush-Kuhn-Tucker (KKT) conditions, the derivatives of $\mathcal{L}$ with respect to the primary variables have to vanish at any extrema, leading to

$$\frac{\partial \mathcal{L}}{\partial f} = f + \sum_{i=1}^n (\alpha_i^- - \alpha_i^+) K(\boldsymbol{x}_i, \cdot) \overset{!}{=} 0, \tag{2.7}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i^+} = C - \alpha_i^+ - \eta_i^+ \overset{!}{=} 0, \quad i = 1 \dots n, \tag{2.8}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i^-} = C - \alpha_i^- - \eta_i^- \overset{!}{=} 0, \quad i = 1 \dots n, \tag{2.9}$$

where (2.7) directly yields (2.4). The last two conditions together with the positivity constraints on $\boldsymbol{\alpha}^{\pm}$ and $\boldsymbol{\eta}^{\pm}$ also directly show (2.5b). Moreover, (2.7) also allows to write

$$\begin{aligned}\left|\left|\hat{f}\right|\right|_{\mathcal{H}}^2 = \left\langle \hat{f}, \hat{f} \right\rangle_{\mathcal{H}} &= \sum_{i,j}^{n}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)K(\boldsymbol{x}_i, x_j) \\ &= \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)^T \boldsymbol{K}\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right), \\ \hat{f}(\boldsymbol{x}_i) = \left\langle \hat{f}, K(\boldsymbol{x}_i, \cdot)\right\rangle &= \sum_{j=1}^{n}(\alpha_j^+ - \alpha_j^-)K(x_j, x_i) = \boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right),\end{aligned}$$

where $\boldsymbol{K}_i = \boldsymbol{k}_X(\boldsymbol{x}_i)^T$ denotes the transposed kernel vector at $\boldsymbol{x}_i$, i.e. the i-th row of $\boldsymbol{K}$. Hence, additionally using the conditions (2.8), (2.9) we can reduce $\mathcal{L}$ to be only dependent on $\boldsymbol{\alpha}^{\pm}$:

$$\begin{aligned}& L\left(\hat{f}, \xi^+, \xi^-, \boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-, \eta^+, \eta^-\right) \\ = \; & \frac{1}{2}\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)^T \boldsymbol{K}\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) + C\sum_{i=1}^{n}\left(\xi_i^+ + \xi_i^-\right) - \epsilon\sum_{i=1}^{n}(\alpha_i^+ + \alpha_i^-) \\ & + \sum_{i=1}^{n}(\alpha_i^+ - \alpha_i^-)y_i + \sum_{i=1}^{n}\alpha_i^+\left(-\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) - \xi_i^+\right) \\ & + \sum_{i=1}^{n}\alpha_i^-\left(\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) - \xi_i^-\right) - \sum_{i=1}^{n}\left(\left(C - \alpha_i^+\right)\xi_i^+ + \left(C - \alpha_i^-\right)\xi_i^-\right) \\ = \; & \frac{1}{2}\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)^T \boldsymbol{K}\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) - \epsilon\sum_{i=1}^{n}(\alpha_i^+ + \alpha_i^-) + y^T\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) \\ & + C\sum_{i=1}^{n}\left(\xi_i^+ + \xi_i^-\right) - \sum_{i=1}^{n}\alpha_i^+\left(\xi_i^+ + \xi_i^-\right) - C\sum_{i=1}^{n}\left(\xi_i^+ + \xi_i^-\right) \\ & + \sum_{i=1}^{n}\alpha_i^+\left(\xi_i^+ + \xi_i^-\right) - \sum_{j=1}^{n}(\boldsymbol{\alpha}_j^+ - \boldsymbol{\alpha}_j^-)\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) \\ = \; & -\frac{1}{2}\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)^T \boldsymbol{K}\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) - \epsilon\sum_{i=1}^{n}(\alpha_i^+ + \alpha_i^-) + y^T\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) \\ = \; & W(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-).\end{aligned}$$

Finally, for $\boldsymbol{\alpha}^{\pm}$ we have the complementary conditions

$$0 = \alpha_i^+ \left(y_i - \left\langle \hat{f}, K(\boldsymbol{x}_i, \cdot) \right\rangle_{\mathcal{H}} - \epsilon - \xi_i^+ \right), \quad i = 1 \dots n, \tag{2.10}$$

$$0 = \alpha_i^- \left(-(y_i - \left\langle \hat{f}, K(\boldsymbol{x}_i, \cdot) \right\rangle_{\mathcal{H}}) - \epsilon - \xi_i^- \right), \quad i = 1 \dots n. \tag{2.11}$$

Suppose $\alpha_i^+ \neq 0$, then we have by (2.10) that $\xi_i^+ = y_i - \left\langle \hat{f}, K(\boldsymbol{x}_i, \cdot) \right\rangle_{\mathcal{H}}$ and hence $\alpha_i^-(-2\epsilon - \xi_i^+ - \xi_i^-) = 0$ by (2.11). But as $\xi_i^+, \xi_i^- \geq 0$ this means $\alpha_i^- = 0$. By symmetry we obtain the second set of constraints (2.5c, p.29). □

It is remarkable that we did not impose any structure on $\hat{f} \in \mathcal{H}$ a-priori, but it turned out that the optimal solution is really contained in $\mathcal{H}^X$. This actually holds true for SVM solutions in general and is known as the "*Representer Theorem*", see e.g. [160, 152]. Now, the problem (2.5, p.29) states a "*quadratic program*" (QP), which can readily be solved by any QP solver like MatLab's `quadprog`. However, standard QP procedures require to store the whole kernel matrix $\boldsymbol{K}$ in memory or at least have to access all entries in each iteration. Despite the possibly high memory requirements for large data sets X, those QP solvers ofttimes are also too slow in practice. This can be computationally infeasible, which is why alternative solution strategies have been investigated since.

2.1.3 Sequential minimal optimization

In this section we introduce a solution strategy called "*sequential minimal optimization*" (SMO) and present an adopted version following the new ideas in [162]. SMO goes back to [133] and is discussed with more overview in [152, §10.5] or [32]. However, the "classic" approach favors the use of an additional *offset term* $b \in \mathbb{R}$ in the resulting kernel expansions, and it is shown in [162], that SVMs without offset not only perform as well as their counterpart, but, using some new ideas, enjoy significantly shorter run-times. The authors of [162] use the Hinge-loss formulation for classification, and we shall present an adopted formulation using the ϵ-loss for regression in the following.

The key idea of SMO in comparison to global methods is to only change a fixed amount of unknowns or points per iteration, while the selection of those points is based on the optimal *analytical* increase in W. In [162] this is carried out for simultaneous selection of one or two points, however, due to the more technical formulation due to two dual variables we will only shortly convey the main ideas for the "1D" case and provide full details for the "2D" case. We begin to introduce the clipping concept also used in [162], which will play an important role in satisfying the constraints (2.5b, p.29).

Definition 2.1.4 (Clipping operation)**.** For real numbers $a, b, c \in \mathbb{R}$ we define by

$$[a]_b^c := \min\{c, \max\{b, a\}\}$$

the *clipping operation* of b, c on a, which limits the value of a to the interval $[b, c]$.

Now we seek to find the direction or index i, whose update yields the highest *gain* with respect to W. Therefore, we define

$$\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) := \frac{\partial W}{\partial \alpha_i^+}\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) = -\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) + y_i - \epsilon, \tag{2.12a}$$

$$\nabla W_i^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) := \frac{\partial W}{\partial \alpha_i^-}\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) = \boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) - y_i - \epsilon \tag{2.12b}$$

as shorthand for the derivatives of W in (2.6, p.29). We look at the analytic maxima of $\omega^+(r) \mapsto W(\boldsymbol{\alpha}^+ + re_i, \boldsymbol{\alpha}^-)$ and $\omega^-(r) \mapsto W(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^- + re_i)$ for all i, which will give us optimal updates $r_i^\pm$ for $\alpha_i^\pm$, respectively. As example, for ω^+ we have

$$\begin{aligned}\omega^+(r) &= W(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - r\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) - \frac{1}{2}r^2 - r\epsilon + ry_i,\\ \frac{\partial \omega^+}{\partial r}(r) &= -\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) + y_i - \epsilon - r = \nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - r.\end{aligned}$$

We identify the global maximum at $\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)$, due to the concavity of ω^+. But in order to satisfy the constraints (2.5b, p.29), we actually need to set

$$r_i^+ := \left[\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)\right]_{-\alpha_i^+}^{C-\alpha_i^+} . \tag{2.13}$$

With this update we can compute the gain in W easily via

$$\begin{aligned} \omega^+(r_i^+) - W(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) &= r_i^+ \left(-\boldsymbol{K}_i \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) - \frac{1}{2} r_i^+ - \epsilon + y_i\right) \\ &= r_i^+(\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \frac{1}{2} r_i^+). \end{aligned}$$

Applying the same procedure to obtain r_i^-, we can now choose the index with the largest gain. However, the conditions $\alpha_i^+ \alpha_i^- = 0$ have not been taken care of yet. They allow to reduce the set of possible indices to search for maxima, as α_i^+ or α_i^- may only be changed if $\alpha_i^- = 0$ or $\alpha_i^+ = 0$, respectively. Thus, let $I^+ := \{i \mid \alpha_i^- = 0\}$ and $I^- := \{i \mid \alpha_i^+ = 0\}$ and

$$g(i) := \begin{cases} r_i^+(\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \frac{1}{2} r_i^+), & i \in I^+, \\ r_i^-(\nabla W_i^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \frac{1}{2} r_i^-), & i \in I^-. \end{cases}$$

Then the index with the largest gain is

$$i^* = \arg \max_{i \in I^+ \cup I^-} g(i),$$

which can be computed (in the worst case for $\boldsymbol{\alpha}^+ = \boldsymbol{\alpha}^- = 0$) in $\mathcal{O}(2n)$ time as $|I^+ \cup I^-| \leq 2n$.

Notice that we can write changes in the j-th coordinate as

$$\nabla W_i^+(\boldsymbol{\alpha}^+ + r\boldsymbol{e}_j, \boldsymbol{\alpha}^-) = \nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - rK_{ij}, \tag{2.14a}$$

$$\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^- + r\boldsymbol{e}_j) = \nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + rK_{ij}, \tag{2.14b}$$

$$\nabla W_i^-(\boldsymbol{\alpha}^+ + r\boldsymbol{e}_j, \boldsymbol{\alpha}^-) = \nabla W_i^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + rK_{ij}, \tag{2.14c}$$

$$\nabla W_i^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^- + r\boldsymbol{e}_j) = \nabla W_i^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - rK_{ij}, \tag{2.14d}$$

where $\boldsymbol{e}_j \in \mathbb{R}^n$ denotes the j-th unit vector and $K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ the i, j-th entry of the kernel matrix $\boldsymbol{K} = \boldsymbol{K}^X$. This allows for a very efficient algorithm, as both $\boldsymbol{\alpha}^{\pm}$ and $\nabla W^{\pm}(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)$ can be kept as single vectors and updated in-place via,

$$\begin{aligned}
\boldsymbol{\alpha}_{i^*}^+ &:= \boldsymbol{\alpha}_{i^*}^+ + \left[\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)\right]_{-\alpha_i^+}^{C-\alpha_i^+}, \\
\nabla W_j^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) &:= \nabla W_j^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - r_{i^*}^+ K_{ji^*}, \quad j = 1 \dots n, \\
\nabla W_j^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) &:= \nabla W_j^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + r_{i^*}^+ K_{ji^*}, \quad j = 1 \dots n,
\end{aligned}$$

for $i^* \in I^+$ and an analogous procedure for $i^* \in I^-$. This can be iterated until the maximum count of iterations is reached or a suitable stopping criteria is met. However, we will detail those aspects in the next section and conclude with a pseudo-code implementation for the 1D-SMO variant shown in Algorithm 1.

2.1.3.1 Double step (2D) optimization

A more advanced technique is to update two variables simultaneously. This might cause the algorithm to finish after possibly less iterations, but also comes at additional computational cost. As in this case update directions i, j have to be chosen, the analytic solution has to consider many different cases. However, as already pointed out in [162], the search for a “good” index pair i, j does not have to be of quadratic complexity $\mathcal{O}\left(n^2\right)$, cf. “Working set selection” paragraph below.

Analytic maxima As before, we fix two $i \neq j$ and look at the analytic maxima of

$$\begin{aligned}
\omega^{++} &: (r, s) \mapsto W(\boldsymbol{\alpha}^+ + re_i + se_j, \boldsymbol{\alpha}^-), \\
\omega^{+-} &: (r, s) \mapsto W(\boldsymbol{\alpha}^+ + re_i, \boldsymbol{\alpha}^- + se_j), \\
\omega^{--} &: (r, s) \mapsto W(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^- + re_i + se_j),
\end{aligned}$$

Algorithm 1 : $[\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-] = \epsilon$-SVR-1D-SMO$(\boldsymbol{K}, \boldsymbol{y}, \epsilon, \lambda, \boldsymbol{a}^{\pm}_{init}, \delta)$

```
1:  $C \leftarrow \frac{1}{2N\lambda} \quad (\boldsymbol{y} \in \mathbb{R}^N)$
2:  $\boldsymbol{\alpha}^+ \leftarrow \boldsymbol{\alpha}^+_{init}, \quad \boldsymbol{\alpha}^- \leftarrow \boldsymbol{\alpha}^-_{init}$
3:  $\nabla \boldsymbol{w}^+ \leftarrow -\boldsymbol{K}(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-) + \boldsymbol{y} - \epsilon$
4:  $\nabla \boldsymbol{w}^- \leftarrow \boldsymbol{K}(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-) - \boldsymbol{y} - \epsilon$
5:  $E \leftarrow C \sum_{i=1}^{n} \max \left\{ \left[\nabla w_i^+\right]_0^{2-\epsilon}, \left[\nabla w_i^-\right]_0^{2-\epsilon} \right\}$    ▷ Initialize stopping condition values
6:  $T \leftarrow -\langle \boldsymbol{\alpha}^+, \nabla \boldsymbol{w}^+ \rangle - \langle \boldsymbol{\alpha}^-, \nabla \boldsymbol{w}^- \rangle$
7:  while $T + E > \delta N C$ do
8:      $\boldsymbol{r}^+ \leftarrow [\boldsymbol{w}^+]_{-\boldsymbol{\alpha}^+}^{C-\boldsymbol{\alpha}^+}, \quad \boldsymbol{r}^- \leftarrow [\boldsymbol{w}^-]_{-\boldsymbol{\alpha}^-}^{C-\boldsymbol{\alpha}^-}$    ▷ Compute constraint-satisfying updates for each point
9:      $I^+ \leftarrow \mathit{find}(\boldsymbol{\alpha}^- = 0), \quad I^- \leftarrow \mathit{find}(\boldsymbol{\alpha}^+ = 0)$    ▷ Identify modifiable indices
10:     $g(i) := \begin{cases} \boldsymbol{r}_i^+(\boldsymbol{w}^+ - \frac{1}{2}\boldsymbol{r}_i^+), & i \in I^+ \\ \boldsymbol{r}_i^-(\boldsymbol{w}^- - \frac{1}{2}\boldsymbol{r}_i^-), & i \in I^- \end{cases}$    ▷ Compute gain vector for modifiable indices
11:     $i^* \leftarrow \arg\max_{i \in I^+ \cup I^-} g(i)$    ▷ Find index with largest gain
12:     if $i^* \in I^+$ then
13:         $\alpha_{i^*}^+ \leftarrow \alpha_{i^*}^+ + r_{i^*}^+$    ▷ Update coefficients
14:         $\nabla \boldsymbol{w}^+ \leftarrow \nabla \boldsymbol{w}^+ - r_{i^*}^+ \boldsymbol{K}(i^*, :), \quad \nabla \boldsymbol{w}^- \leftarrow \nabla \boldsymbol{w}^- + r_{i^*}^+ \boldsymbol{K}(i^*, :)$    ▷ Update $W$-gradients
15:         $T \leftarrow T + r_{i^*}^+ \left(r_{i^*}^+ - 2\nabla w_{i^*}^+ - \epsilon + y_{i^*}\right)$    ▷ Update stopping conditions
16:     else
17:         $\alpha_{i^*}^- \leftarrow \alpha_{i^*}^- + r_{i^*}^-$
18:         $\nabla \boldsymbol{w}^+ \leftarrow \nabla \boldsymbol{w}^+ + r_{i^*}^- \boldsymbol{K}(i^*, :), \nabla \boldsymbol{w}^- \leftarrow \nabla \boldsymbol{w}^- - r_{i^*}^- \boldsymbol{K}(i^*, :)$
19:         $T \leftarrow T + r_{i^*}^- \left(r_{i^*}^- - 2\nabla w_{i^*}^- - \epsilon - y_{i^*}\right)$
20:     end if
21:     $E \leftarrow C \sum_{i=1}^{n} \max \left\{ \left[w_i^+\right]_0^{2-\epsilon}, \left[w_i^-\right]_0^{2-\epsilon} \right\}$    ▷ Recompute stopping condition values
22: end while
23: return $(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)$
```

which will give us two optimal updates for $\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-$, named r_i^+, r_i^-, s_j^+ or s_j^-, respectively. For the 2D case the functions and derivatives are given as

$$\begin{aligned}
\omega^{++}(r,s) &= W(\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-) - r\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) - sK_j\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) \\
&\quad - \frac{1}{2}(r^2+s^2) - rsK_{ij} - \epsilon(r+s) + y_i(r+s), \\
\frac{\partial\omega^{++}}{\partial r}(r,s) &= -\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) + y_i - \epsilon - r - sK_{ij} \\
&= \nabla W_i^+(\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-) - r - sK_{ij}, \\
\frac{\partial\omega^{++}}{\partial s}(r,s) &= -\boldsymbol{K}_j\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) + y_j - \epsilon - s - rK_{ij} \\
&= \nabla W_j^+(\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-) - s - rK_{ij}, \\
\omega^{+-}(r,s) &= W(\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-) - r\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) + s\boldsymbol{K}_j\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) \\
&\quad - \frac{1}{2}(r^2+s^2) + rsK_{ij} - \epsilon(r+s) + y_i(r-s), \\
\frac{\partial\omega^{+-}}{\partial r}(r,s) &= \nabla W_i^+(\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-) - r + sK_{ij}, \\
\frac{\partial\omega^{+-}}{\partial s}(r,s) &= \nabla W_j^-(\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-) - s + rK_{ij}, \\
\omega^{--}(r,s) &= W(\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-) + r\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) + s\boldsymbol{K}_j\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) \\
&\quad - \frac{1}{2}(r^2+s^2) - rsK_{ij} - \epsilon(r+s) - y_i(r+s), \\
\frac{\partial\omega^{--}}{\partial r}(r,s) &= \nabla W_i^-(\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-) - r - sK_{ij}, \\
\frac{\partial\omega^{--}}{\partial s}(r,s) &= \nabla W_j^-(\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-) - s - rK_{ij}.
\end{aligned}$$

In the following we will focus on ω^{++}. We see that the Hessian is negative definite and we thus have a global maximum at $\nabla\omega^{++} \equiv 0$. This condition may be expressed as a system of linear equations

$$\begin{pmatrix} -1 & -K_{ij} \\ -K_{ij} & -1 \end{pmatrix} \begin{pmatrix} r \\ s \end{pmatrix} = \begin{pmatrix} -\nabla W_i^+(\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-) \\ -\nabla W_j^+(\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-) \end{pmatrix},$$

whose solution yields the analytic maxima

$$\begin{aligned} r_i^{++} &:= \frac{\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - K_{ij} \nabla W_j^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)}{1 - K_{ij}^2}, \\ s_j^{++} &:= \frac{\nabla W_j^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - K_{ij} \nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)}{1 - K_{ij}^2}. \end{aligned}$$

Recall here that $K_{ij} < 1 \,\forall\, i \neq j$ as K is a normalized, positive definite kernel and the training inputs X are pairwise distinct. Since only two different directions $i \neq j$ make sense for simultaneous optimization, we also do not need to consider the case $i = j$.

Updates As in the 1D case, the updates might conflict with the constraints (2.5b, p.29). Unfortunately, we now count nine possible cases where r_i^{++} or s_j^{++} updates may violate the constraints, as depicted in Figure 2.2.

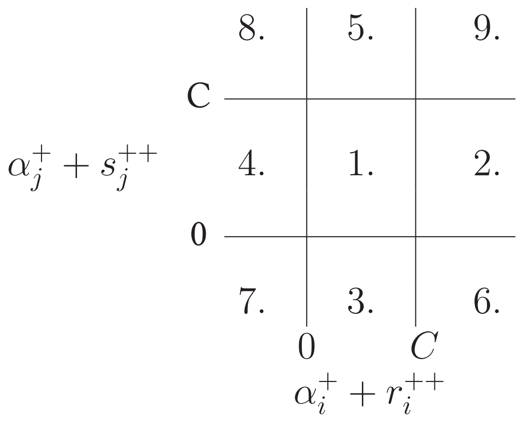


Figure 2.2: Possible constraint cases for either or both r_i^{++} and s_j^{++}.

1. $\alpha_i^+ + r_i^{++} \in [0, C], \alpha_j^+ + s_j^{++} \in [0, C]$
   Both updates are valid and can be added to α_i^+ and α_j^+, respectively.

2. $\alpha_i^+ + r_i^{++} > C, \alpha_j^+ + s_j^{++} \in [0, C]$
   The maximum lies on $\{C\} \times [0, C]$. Set $r_i^{++} := C - \alpha_i^+$ and perform a 1D update in the jth coordinate. From (2.14a, p.33) we obtain a temporary gradient $\nabla W_j^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - (C - \alpha_i^+)K_{ij}$ and thus have given

the clipped 1D update (2.13, p.33) for the jth coordinate

$$s_j^{++} := \left[\nabla W_j^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - (C - \alpha_i^+)K_{ij}\right]_{-\alpha_j^+}^{C-\alpha_j^+} .$$

3. $\alpha_i^+ + r_i^{++} \in [0, C], \alpha_j^+ + s_j^{++} < 0$
   The maximum lies on $[0, C] \times \{0\}$. Set $s_j^{++} := -\alpha_j^+$ and perform a 1D update in the ith coordinate. From (2.14a, p.33) we obtain a temporary gradient $\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + \alpha_j^+ K_{ij}$ and thus have given a clipped 1D update (2.13, p.33) for the ith coordinate

$$r_i^{++} := \left[\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + \alpha_j^+ K_{ij}\right]_{-\alpha_i^+}^{C-\alpha_i^+} .$$

4. $\alpha_i^+ + r_i^{++} < 0, \alpha_j^+ + s_j^{++} \in [0, C]$
   This is case 3. with interchanged roles, so by symmetry we obtain the updates

$$r_i^{++} = -\alpha_i^+, \quad s_j^{++} := \left[\nabla W_j^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + \alpha_i^+ K_{ij}\right]_{-\alpha_j^+}^{C-\alpha_j^+} .$$

5. $\alpha_i^+ + r_i^{++} \in [0, C], \alpha_j^+ + s_j^{++} > C$
   This is case 2. with interchanged roles, so by symmetry we obtain the updates

$$r_i^{++} = \left[\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - (C - \alpha_j^+)K_{ij}\right]_{-\alpha_i^+}^{C-\alpha_i^+} , \quad s_j^{++} := C - \alpha_j^+.$$

6. $\alpha_i^+ + r_i^{++} > C, \alpha_j^+ + s_j^{++} < 0$
   Here the maximum lies somewhere on the set $\{C\} \times [0, C] \cup [0, C] \times \{0\}$. For this case we assume each update to be bounded and perform an 1D update for the other coordinate, respectively. This equals the cases 2. and 3., and we choose the coordinate pair with the largest gain.

7. $\alpha_i^+ + r_i^{++} < 0, \alpha_j^+ + s_j^{++} < 0$
   Similar to 6., we choose the coordinate pair with maximum gain from

cases 3. and 4.

8. $\alpha_i^+ + r_i^{++} < 0, \alpha_j^+ + s_j^{++} > C$
   Choose the best maximum gain coordinates from cases 4. and 5.

9. $\alpha_i^+ + r_i^{++} > C, \alpha_j^+ + s_j^{++} > C$
   Choose the best maximum gain coordinates from cases 5. and 2.

Consequently, for ω^{+-} and ω^{--} we obtain the analytical maxima by an analogous derivation

$$\begin{aligned}
r_i^{+-} &:= \frac{\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + K_{ij}\nabla W_j^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)}{1 - K_{ij}^2}, \\
s_j^{+-} &:= \frac{\nabla W_j^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + K_{ij}\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)}{1 - K_{ij}^2}, \\
r_i^{--} &:= \frac{\nabla W_i^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - K_{ij}\nabla W_j^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)}{1 - K_{ij}^2}, \\
s_j^{--} &:= \frac{\nabla W_j^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - K_{ij}\nabla W_i^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)}{1 - K_{ij}^2},
\end{aligned}$$

and also have to consider the constrained cases and resulting updates as explained above for ω^{++}.

2D Gain With the updates above we can compute the gain for any directions i, j in W:

$$\begin{aligned}
\omega^{++}(r_i^{++}, s_j^{++}) - W(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) &= r_i^{++}(\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \frac{1}{2}r_i^{++}) \\
&\quad + s_j^{++}(\nabla W_j^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \frac{1}{2}s_j^{++}) - r_i^{++}s_j^{++}K_{ij}, \\
\omega^{+-}(r_i^{+-}, s_j^{+-}) - W(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) &= r_i^{+-}(\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \frac{1}{2}r_i^{+-}) \\
&\quad + s_j^{+-}(\nabla W_j^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \frac{1}{2}s_j^{+-}) + r_i^{+-}s_j^{+-}K_{ij}, \\
\omega^{--}(r_i^{--}, s_j^{--}) - W(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) &= r_i^{--}(\nabla W_i^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \frac{1}{2}r_i^{--})
\end{aligned}$$

$$+ s_j^{--}(\nabla W_j^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \frac{1}{2}s_j^{--}) - r_i^{--}s_j^{--}K_{ij}.$$

Working set selection Recall that we have the conditions $\alpha_i^+\alpha_i^- = 0$ and $I^+ = \{i \mid \alpha_i^- = 0\}$ and $I^- = \{i \mid \alpha_i^+ = 0\}$, which allows to limit the analytical gains that have to be computed. Nevertheless, finding the optimal 2D gain in general involves a search of quadratic complexity $\mathcal{O}\left(n^2\right)$. The authors of [162] introduce alternative strategies and demonstrate in experiments to obtain the same quality of results in only $\mathcal{O}\left(kn\right)$ complexity for $k \ll n$. We list a short description of the most important selection strategies and refer to the original work for details.

1. "WSS0": Choose the indices i, j with optimal gain by brute force search ($\mathcal{O}\left(n^2\right)$)
2. "WSS1": Choose 1D maximal gain and previous 1D direction ($\mathcal{O}\left(n\right)$)
3. "WSS2": Maximum 1D gain for two subsets ($\mathcal{O}\left(n\right)$)
4. "WSS4": Maximum 1D gain and second index from $k \in \mathbb{N}$ neighbors ($\mathcal{O}\left(kn\right)$)
5. Combinations of the above strategies and others.

Updates Once two suitable indices $i^*, j^* \in I^+ \cup I^-$ are determined, the updates to $\boldsymbol{\alpha}^\pm$ and $\nabla W^\pm$ can be obtained from (2.14, p.33) by consecutive application. For $i^* \in I^+, j^* \in I^+$ we set

$$\begin{aligned}
\boldsymbol{\alpha}_{i^*}^+ &:= \boldsymbol{\alpha}_{i^*}^+ + r_{i^*}^{++}, \quad \boldsymbol{\alpha}_{j^*}^+ := \boldsymbol{\alpha}_{j^*}^+ + s_{j^*}^{++}, \\
\nabla W_k^+ &:= \nabla W_k^+ - r_{i^*}^{++}K_{ki^*} - s_{j^*}^{++}K_{kj^*}, \quad k = 1 \dots n, \\
\nabla W_k^- &:= \nabla W_k^- + r_{i^*}^{++}K_{ki^*} + s_{j^*}^{++}K_{kj^*}, \quad k = 1 \dots n.
\end{aligned}$$

For $i^* \in I^+, j^* \in I^-$ we set

$$\begin{aligned}
\boldsymbol{\alpha}_{i^*}^+ &:= \boldsymbol{\alpha}_{i^*}^+ + r_{i^*}^{+-}, \quad \boldsymbol{\alpha}_{j^*}^- := \boldsymbol{\alpha}_{j^*}^- + s_{j^*}^{+-}, \\
\nabla W_k^+ &:= \nabla W_k^+ - r_{i^*}^{+-} K_{ki^*} + s_{j^*}^{+-} K_{kj^*}, \quad k = 1 \dots n, \\
\nabla W_k^- &:= \nabla W_k^- + r_{i^*}^{+-} K_{ki^*} - s_{j^*}^{+-} K_{kj^*}, \quad k = 1 \dots n.
\end{aligned}$$

For $i^* \in I^-, j^* \in I^-$ we set

$$\begin{aligned}
\boldsymbol{\alpha}_{i^*}^- &:= \boldsymbol{\alpha}_{i^*}^- + r_{i^*}^{--}, \quad \boldsymbol{\alpha}_{j^*}^- := \boldsymbol{\alpha}_{j^*}^- + s_{j^*}^{--}, \\
\nabla W_k^+ &:= \nabla W_k^+ + r_{i^*}^{--} K_{ki^*} + s_{j^*}^{--} K_{kj^*}, \quad k = 1 \dots n, \\
\nabla W_k^- &:= \nabla W_k^- - r_{i^*}^{--} K_{ki^*} - s_{j^*}^{--} K_{kj^*}, \quad k = 1 \dots n.
\end{aligned}$$

2.1.3.2 Stopping conditions

So far we have not detailed suitable stopping conditions for this SMO variant. We introduce them before discussing initial value choices, as some values regarding the stopping criteria also have to be initialized. We follow the work [162, §2.2] but adopt the considered values to the ϵ-SVR case and allow arbitrary function values outside the range $[-1, 1]$. Basically, the idea is to monitor the "*duality gap*", see e.g. [32], but in a clipped version.

Theorem 2.1.5 (δ-approximation of optimum for ϵ-SVR)**.** *In the context of Proposition 2.1.3 let* $\delta > 0, b := ||\boldsymbol{y}||_\infty$ *for* $\boldsymbol{y} = (y_1, \dots, y_n) \in \mathbb{R}^n$ *and define*

$$\begin{aligned}
T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) := &\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)^T \boldsymbol{K} \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) \\
&\qquad + \epsilon \sum_{i=1}^n (\alpha_i^+ + \alpha_i^-) - y^T \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right), \\
E(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) := &\, C \sum_{i=1}^n \left| y_i - \left[\boldsymbol{K}_i \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)\right]_{-b}^b \right|_\epsilon .
\end{aligned}$$

Then for any $\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-$ satisfying $T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + E(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) \leq \frac{\delta}{2\lambda}$ we have

$$\lambda \left|\left|\hat{f}\right|\right|_{\mathcal{H}}^2 + R_{L,P_D}\left(\left[\hat{f}\right]_{-b}^{b}\right) \leq \min_{g \in \mathcal{H}} \lambda \left|\left|g\right|\right|_{\mathcal{H}}^2 + R_{L,P_D}(g) + \delta,$$

with $\hat{f}$ as in (2.4, p.28).

Proof. Let $\hat{f}, \xi^{+*}, \xi^{-*}$ be the solution of the primary problem (2.3, p.28), where we denote the objective function of (2.3, p.28) by $P(\hat{f}, \xi^{+*}, \xi^{-*})$. By construction of the slack variables in Proposition 2.1.2 we know that

$$C \sum_{i=1}^{n} \left(\xi_i^{+*} + \xi_i^{-*}\right) = C \sum_{i=1}^{n} \left|y_i - \hat{f}(x_i)\right|_{\epsilon}.$$

Hence, by duality (Maximal W corresponds to minimal P) we see that

$$W(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) \leq P(\hat{f}, \xi^{+*}, \xi^{-*}) = \frac{1}{2}\left|\left|\hat{f}\right|\right|_{\mathcal{H}}^2 + C \sum_{i=1}^{n} \left|y_i - \hat{f}(x_i)\right|_{\epsilon},$$

for any $\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-$ satisfying the constraints of the dual problem. We then deduce

$$\begin{aligned}
&\lambda \left|\left|f\right|\right|_{\mathcal{H}}^2 + R_{L,P_D}([f]_{-b}^{b}) \\
&= 2\lambda \left(\frac{1}{2}\left|\left|f\right|\right|_{\mathcal{H}}^2 + C \sum_{i=1}^{n} \left|y_i - [f(\boldsymbol{x}_i)]_{-b}^{b}\right|_{\epsilon}\right) \\
&= 2\lambda \left(\frac{1}{2}\left|\left|f\right|\right|_{\mathcal{H}}^2 - W(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + W(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)\right. \\
&\quad \left. + C \sum_{i=1}^{n} \left|y_i - \left[\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)\right]_{-b}^{b}\right|_{\epsilon}\right) \\
&\leq 2\lambda \left(\frac{1}{2}\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)^T \boldsymbol{K}\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) - W(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)\right. \\
&\quad \left. + P(\hat{f}, \xi^{+*}, \xi^{-*}) + E(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)\right) \\
&= 2\lambda \left(P(\hat{f}, \xi^{+*}, \xi^{-*}) + T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + E(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)\right)
\end{aligned}$$

$$\leq \min_{g\in\mathcal{H}} \lambda \left|\left|g\right|\right|_{\mathcal{H}}^2 + R_{L,P_D}(g) + \delta.$$

□

Theorem 2.1.5 provides direct control over the accuracy of the solution by checking $T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + E(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) \leq \frac{\delta}{2\lambda}$ for given $\delta > 0$.

Now, Lemma 2.1.6 and 2.1.7 show how to efficiently update and compute the T and E terms after each SMO step using the $\nabla W^{\pm}$ gradients.

Lemma 2.1.6 (Changes in $T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)$ for single and double step optimization)**.** *Let T be given as in Theorem 2.1.5. Then for $r \in \mathbb{R}$ and $i \in \{1, \dots n\}$ we have*

$$\begin{aligned}
T(\boldsymbol{\alpha}^+ + re_i, \boldsymbol{\alpha}^-) &= T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + r\left(r - 2\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \epsilon + y_i\right), \\
T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^- + re_i) &= T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + r\left(r - 2\nabla W_i^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \epsilon - y_i\right).
\end{aligned}$$

Further, for $r, s \in \mathbb{R}$ and coordinate pairs i, j, changes in T are given by

$$\begin{aligned}
T(\boldsymbol{\alpha}^+ + re_i + se_j, \boldsymbol{\alpha}^-) &= T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + r\left(r - 2\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \epsilon + y_i\right) \\
&\quad + s\left(s - 2\nabla W_j^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \epsilon + y_j\right) + 2rsK_{ij}, \\
T(\boldsymbol{\alpha}^+ + re_i, \boldsymbol{\alpha}^- + se_j) &= T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + r\left(r - 2\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \epsilon + y_i\right) \\
&\quad + s\left(s - 2\nabla W_j^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \epsilon - y_j\right) - 2rsK_{ij}, \\
T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^- + re_i + se_j) &= T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + r\left(r - 2\nabla W_i^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \epsilon - y_i\right) \\
&\quad + s\left(s - 2\nabla W_j^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \epsilon - y_j\right) + 2rsK_{ij}.
\end{aligned}$$

Proof. For an update of $\boldsymbol{\alpha}^+$ we have

$$\begin{aligned}
&T(\boldsymbol{\alpha}^+ + re_i, \boldsymbol{\alpha}^-) \\
&= \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)^T \boldsymbol{K}\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) + 2r\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) + r^2 \\
&\quad + \epsilon\sum_{i=1}^n(\alpha_i^+ + \alpha_i^-) + r\epsilon - y^T\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) - ry_i \\
&= T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + r\left(2\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) + r + \epsilon - y_i\right)
\end{aligned}$$

$$
\begin{aligned}
&= T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + r\left(2\left[-\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \epsilon + y_i\right] + r + \epsilon - y_i\right) \\
&= T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + r\left(r - 2\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \epsilon + y_i\right).
\end{aligned}
$$

Similarly, we obtain for $\boldsymbol{\alpha}^-$ that

$$
T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^- + re_i) = T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + r\left(r - 2\nabla W_i^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) - \epsilon - y_i\right).
$$

Consecutive application of the above for pairs i, j together with (2.14, p.33) directly yields the results for double step updates. $\square$

Lemma 2.1.7 (Computation of $E(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)$). *The term E from Theorem 2.1.5 can be written as*

$$
E(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) = C\sum_{i=1}^{n}\left[\max\left\{\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-), \nabla W_i^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)\right\}\right]_0^{2b-\epsilon}.
$$

Proof. Consider

$$
\begin{aligned}
\left[y_i - \boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)\right]_{-b}^{b} - \epsilon &= \left[y_i - \boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) - \epsilon\right]_{-b-\epsilon}^{b-\epsilon} \\
&= \left[\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)\right]_{-b-\epsilon}^{b-\epsilon}.
\end{aligned}
$$

Similarly, we have

$$
\left[\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) - y_i\right]_{-b}^{b} - \epsilon = \left[\nabla W_i^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)\right]_{-b-\epsilon}^{b-\epsilon}.
$$

Hence

$$
\begin{aligned}
&E(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) \\
&= C\sum_{i=1}^{n}\left|y_i - \left[\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)\right]_{-b}^{b}\right|_{\epsilon} \\
&= C\sum_{i=1}^{n}\max\left\{0, \left|\left[y_i - \boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)\right]_{-2b}^{2b}\right| - \epsilon\right\} \\
&= C\sum_{i=1}^{n}\max\left\{0, \max\left\{\left[y_i - \boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)\right]_{-2b}^{2b} - \epsilon, \left[\boldsymbol{K}_i\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) - y_i\right]_{-2b}^{2b} - \epsilon\right\}\right\} \\
&= C\sum_{i=1}^{n}\max\left\{0, \max\left\{\left[\nabla W_i^+(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)\right]_{-2b-\epsilon}^{2b-\epsilon}, \left[\nabla W_i^-(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)\right]_{-2b-\epsilon}^{2b-\epsilon}\right\}\right\}
\end{aligned}
$$

$$
\begin{aligned}
&= C\sum_{i=1}^{n} \max\left\{\left[\nabla W_i^+(\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-)\right]_0^{2b-\epsilon}, \left[\nabla W_i^-(\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-)\right]_0^{2b-\epsilon}\right\} \\
&= C\sum_{i=1}^{n} \left[\max\left\{\nabla W_i^+(\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-), \nabla W_i^-(\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-)\right\}\right]_0^{2b-\epsilon}.
\end{aligned}
$$

$\square$

2.1.3.3 Initial conditions

Finally, every optimization problem requires initial conditions to be given. The two most common choices are

1. **Cold start**. Simply choose $\boldsymbol{\alpha}^+ = \boldsymbol{\alpha}^- \equiv 0$. This satisfies the constraints and has the maximum number of support vectors equal to zero. However, it may lead to many iterations.

2. **Kernel rule**. Recall $C = \frac{1}{2\lambda n}$ and initialize

$$
\alpha_i^+ = \begin{cases} [y_i]_0^C, & y_i > 0 \\ 0, & y_i <= 0 \end{cases}, \qquad \alpha_i^- = \begin{cases} 0, & y_i >= 0 \\ [-y_i]_0^C, & y_i < 0 \end{cases}.
$$

   This approach likely leads to a smaller initial training error and goes back to [42] and is modified in [162].

Note that, of course, other initial conditions like recycling might be suitable in different settings, see e.g. [162, 152, 42] for some more alternatives. Finally, once $\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-$ are chosen the ∇W-gradients can be initialized according to (2.12, p.32), which writes in vectorized form as

$$
\begin{aligned}
\nabla \boldsymbol{W}^+\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) &= -\boldsymbol{K}\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) + \boldsymbol{y} - \epsilon\mathbf{1}, \\
\nabla \boldsymbol{W}^-\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) &= \boldsymbol{K}\left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) - \boldsymbol{y} - \epsilon\mathbf{1},
\end{aligned}
$$

for $\mathbf{1} = (1 \dots 1)^T \in \mathbb{R}^n$. Regarding the stopping conditions, the E term is given explicitly and we can express $T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)$ using $\nabla \boldsymbol{W}^\pm$ via

$$\begin{aligned}
T(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) &= \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)^T \boldsymbol{K} \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) + \epsilon \sum_{i=1}^n (\alpha_i^+ + \alpha_i^-) - y^T \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) \\
&= \sum_{i=1}^n (\alpha_i^+ - \alpha_i^-) \boldsymbol{K}_i \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) + \epsilon \sum_{i=1}^n (\alpha_i^+ + \alpha_i^-) - \sum_{i=1}^n y_i (\alpha_i^+ - \alpha_i^-) \\
&= \sum_{i=1}^n \alpha_i^+ (\boldsymbol{K}_i \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) + \epsilon - y_i) + \alpha_i^- (-\boldsymbol{K}_i \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right) + \epsilon + y_i) \\
&= -\sum_{i=1}^n \alpha_i^+ \nabla W_i^+ (\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) + \alpha_i^- \nabla W_i^- (\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) \\
&= (\nabla \boldsymbol{W}^-)^T \boldsymbol{\alpha}^- - (\nabla \boldsymbol{W}^+)^T \boldsymbol{\alpha}^+.
\end{aligned}$$

Remark 2.1.8. The computation of $\nabla \boldsymbol{W}^\pm \left(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-\right)$ actually does require $\mathcal{O}\left(n^2\right)$ operations, as the complete kernel matrix $\boldsymbol{K}$ is involved. This is in fact the only time the *whole* matrix is needed, which is why the cold start initialization can be a feasible choice to this end. However, the extra initialization cost might yet pay off due to a smaller amount of iterations needed to achieve the prescribed accuracy. As this of course depends on the application at hand, we will leave the level of investigation at this remark.

We refrain here from stating a pseudo-code algorithm for the 2D-SMO variant due to the more involved update cases. However, the implementation can be done straightforwardly following the corresponding sections linearly. A MatLab implementation of the 2D SMO variant is also available in the software framework *KerMor* [184].

2.2 Vectorial kernel orthogonal greedy algorithms

In difference to the SVR technique introduced in Section 2.1, the algorithm discussed in the following will be formulated in a vectorial setting from the start. It belongs to the family of "*greedy algorithms*" which build on the concept of m-term approximation [153, 41]. This formalism is basically a measure of how well a function from a given function space can be approximated by linearly combining m functions out of a given set from the same space. However, direct computation of *the best* m-term approximation is not possible in practice, as its computation has combinatorial complexity dependent on the number of considered elements. Hence, the challenge is to find methods and algorithms that provide near-best m-term approximations, and greedy algorithms have been proven to satisfy this requirement under various conditions, see e.g. [167, 40, 41, 169]. Their "greedy" nature has its foundation in a *greedy step*, which determines the next function to be added to an existing m-term approximant according to certain maximizing criteria, mostly involving residuals. The considered function set can either be a basis of the containing space or a redundant (dense) system of functions, where the latter is also called a "*dictionary*" and is also used in this thesis.

Well known criteria roughly distinguish between pure and weak greedy algorithms, where the true or almost true maximum approximation "gain" is considered, respectively. An extensive survey of greedy algorithms can be found in [169], however, greedy approximation methods appear in the literature in different facets like (orthogonal) matching pursuit [114, 128, 35] or greedy pursuit [172] and others. So far, approximation and convergence results have been established for quite general spaces, e.g. Hilbert [41, 169] or Banach spaces [96, 104]. In this work we will use the RKHS introduced in Section 1.2 as they readily allow to apply the greedy approximation scheme described above. Note that greedy algorithms have been already formulated in the context of RKHS in e.g. [148, 117], also including some results on convergence.

It is also evident that the selection criteria for subsequent new dictionary elements depends on the way the m-term approximant in any current linear subspace is computed. The most natural approach is to use orthogonal projection with respect to the native RKHS scalar product, which guarantees the best possible approximation in each subspace. We shall regard this approach here, however, there are more choices e.g. using least squares [175] or orthogonal least squares [25, 27].

In this thesis, we will focus on a vectorial variant of the orthogonal kernel greedy algorithm, which has intermediately been published in [183] in similar form. It states a vectorial extension of the so-called f/P-Greedy algorithm from [117, 3.1.1] in the spirit of [104] and is characterized by a native-norm maximizing selection criteria. We will investigate this criteria more closely and show that an improved error bound and a-posteriori bounds can be obtained for the considered vectorial kernel orthogonal greedy algorithm (VKOGA). We also show that, under certain conditions, adding the same data points twice yields a Hermite-type interpolation in the limit. For related work on vectorial greedy algorithms see e.g. [107, 103].

In the following we will establish necessary concepts and show their relations to existing work in Section 2.2.1.

2.2.1 Preliminaries

In the following we will introduce the core principles regarding greedy algorithms, where we will at first consider the scalar case $q = 1$.

As mentioned in the introductory part, we will focus on projection of functions into subspaces to obtain approximants.

Definition 2.2.1 (Orthogonal projection operator). Let $S \subseteq \mathcal{H}$ be a linear subspace of $\mathcal{H}$. Then the orthogonal projection operator is denoted by

$$\begin{aligned} P_S : \mathcal{H} &\to S \\ f &\mapsto P_S[f], \end{aligned}$$

such that

$$\langle f - P_S[f], g\rangle_{\mathcal{H}} = 0 \,\forall\, g \in S. \tag{2.15}$$

Next we will show some frequently used properties of projections.

Lemma 2.2.2. *Let $S \subseteq \mathcal{H}$ be a linear subspace of $\mathcal{H}$. Then*

$$||\mathcal{P}_S[f]||_{\mathcal{H}}^2 = \langle f, \mathcal{P}_S[f]\rangle_{\mathcal{H}} \;\forall\, f \in \mathcal{H}.$$

If further $S = \mathcal{H}^X$ for some $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \subset \Omega$ so that $\boldsymbol{K}$ is non-singular, we have

$$\mathcal{P}_S[f] = \sum_{i=1}^{n} c_i K(\boldsymbol{x}_i, \cdot), \quad \boldsymbol{c} = \boldsymbol{K}^{-1}\boldsymbol{f}, \quad f_i = f(\boldsymbol{x}_i),\; i = 1 \dots n. \tag{2.16}$$

Proof. First,

$$\begin{aligned}||\mathcal{P}_S[f]||_{\mathcal{H}}^2 &= \langle \mathcal{P}_S[f], \mathcal{P}_S[f]\rangle_{\mathcal{H}} = \langle f - (f - \mathcal{P}_S[f]), \mathcal{P}_S[f]\rangle_{\mathcal{H}}\\ &= \langle f, \mathcal{P}_S[f]\rangle_{\mathcal{H}} - \underbrace{\langle (f - \mathcal{P}_S[f]), \mathcal{P}_S[f]\rangle_{\mathcal{H}}}_{=0} = \langle f, \mathcal{P}_S[f]\rangle_{\mathcal{H}}\,.\end{aligned}$$

Second, as the projection is in $\mathcal{H}^X$, we have $\mathcal{P}_S[f] = \sum_{i=1}^n c_i K(\boldsymbol{x}_i, \cdot)$ for yet to be determined coefficients c_i. Using the projection conditions (2.15), we obtain

$$\begin{aligned}0 &= \langle f - P_S[f], K(\boldsymbol{x}_j, \cdot)\rangle_{\mathcal{H}} = \sum_{i=1}^{n} \langle f - c_i K(\boldsymbol{x}_i, \cdot), K(\boldsymbol{x}_j, \cdot)\rangle_{\mathcal{H}}\\ &= \sum_{i=1}^{n} f(\boldsymbol{x}_j) - c_i K(\boldsymbol{x}_j, \boldsymbol{x}_i) = f(\boldsymbol{x}_j) - \sum_{i=1}^{n} c_i K(\boldsymbol{x}_j, \boldsymbol{x}_i) \qquad \forall\, j = 1 \dots n,\end{aligned}$$

which writes as the system of equations $\boldsymbol{K}\boldsymbol{c} = \boldsymbol{f}$ and gives (2.16). □

Remark 2.2.3. Using kernel translates as in (2.16) to compute the projections is also known as the "*RBF direct*" method. This is numerically infeasible in general, as the condition of the kernel matrix can get arbitrarily bad especially for close data-points. As the interpolation task itself is not unstable in function space [36], several approaches [146, 61, 60, 57, 117] have been investigated to alleviate those problems, e.g. by choosing a stable basis. In this work, we will employ the "*Newton basis*" introduced in [129] and provide details in Section 2.2.2.3.

Lemma 2.2.2 shows that the projection actually means interpolation at specific function values. This only holds true if the kernel matrix is invertible, or more general, $K(\boldsymbol{x}, \cdot), K(\boldsymbol{y}, \cdot)$ are linearly independent for any $\boldsymbol{x} \neq \boldsymbol{y} \in \Omega$. This property is dependent on the choice of kernel, however, the selection criteria of the greedy algorithms considered in this work choose new points based on the largest approximation "gain", which is not achieved when adding linear dependent dictionary elements. Hence, those algorithms will always select points whose induced kernel matrix is non-singular, even though the projection itself is unique for arbitrary points (but maybe with ambiguous coefficients due to an under-determined system $\boldsymbol{Kc} = \boldsymbol{f}$).

Orthonormal remainders

Unfortunately, due to different perspectives, there have been developing two different but closely related ways of notation in the context of greedy algorithms. The classical greedy theory [169] considers scalar products of function residuals and dictionary elements in the greedy step selection criteria. On the other hand, kernel greedy algorithms [148, 117] usually consider point-wise maxima $\max_x |f(x) - s_{f,X_m}(x)|$ in the greedy step, where s_{f,X_m} is the *interpolant* of f on the current $m - th$ point set X_m. However, using RKHS those perspectives merge together by virtue of the reproducing property. Their connection will become clear using the concept of *orthogonal remainders*, which we will introduce in the following.

Definition 2.2.4 (Orthonormal remainders)**.** Let $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subseteq \Omega$, $\boldsymbol{x} \in \Omega$ and define $\Omega_X := \{\boldsymbol{x} \in \Omega \mid K(\boldsymbol{x}, \cdot) \in \mathcal{H}^X\}$. Then we define the

$\mathcal{H}^X$-orthogonal remainder of $K(\boldsymbol{x}, \cdot)$ as

$$\tilde{\phi}_{\boldsymbol{x}} := K(\boldsymbol{x}, \cdot) - \mathcal{P}_{\mathcal{H}^X}[K(\boldsymbol{x}, \cdot)],$$

and for $\boldsymbol{x} \in \Omega \backslash \Omega_X$ the $\mathcal{H}^X$-orthonormal remainder

$$\phi_{\boldsymbol{x}} := \tilde{\phi}_{\boldsymbol{x}} \Big/ \left\|\tilde{\phi}_{\boldsymbol{x}}\right\|_{\mathcal{H}}.$$

Lemma 2.2.5 (Properties for remainders)**.** *Let $f \in \mathcal{H}$. Then*

$$\mathcal{P}_{\mathcal{H}^X \oplus \langle \phi_{\boldsymbol{x}} \rangle_{\mathcal{H}}}[f] = \mathcal{P}_{\mathcal{H}^X}[f] + \mathcal{P}_{\langle \phi_{\boldsymbol{x}} \rangle_{\mathcal{H}}}[f], \tag{2.17}$$

$$\left\langle f, \tilde{\phi}_{\boldsymbol{x}} \right\rangle_{\mathcal{H}} = \langle f - \mathcal{P}_{\mathcal{H}^X}[f], K(\boldsymbol{x}, \cdot)\rangle_{\mathcal{H}} = f(\boldsymbol{x}) - \mathcal{P}_{\mathcal{H}^X}[f](\boldsymbol{x}). \tag{2.18}$$

Furthermore, $\tilde{\phi}_{\boldsymbol{x}}$ is the Riesz-representative of the linear functional

$$\delta_{\boldsymbol{x}} - \mathcal{P}_{\mathcal{H}^X}[\cdot](\boldsymbol{x}) : \mathcal{H} \to \mathbb{R}.$$

Proof. At first, if $K(\boldsymbol{x}, \cdot) \in \mathcal{H}^X$ then condition (2.17) holds trivially. So, assume $K(\boldsymbol{x}, \cdot) \notin \mathcal{H}^X$ and $\{\varphi_1, \ldots, \varphi_l\}$ to be an orthonormal basis (ONB) of $\mathcal{H}^X$. From this and the condition $\left\langle f - \mathcal{P}_{\langle \phi_{\boldsymbol{x}} \rangle}[f], \phi_{\boldsymbol{x}} \right\rangle_{\mathcal{H}} = 0$ we get

$$\mathcal{P}_{\mathcal{H}^X}[f] = \sum_{i=1}^{l} \langle f, \varphi_i \rangle_{\mathcal{H}} \varphi_i, \qquad \mathcal{P}_{\langle \phi_{\boldsymbol{x}} \rangle_{\mathcal{H}}}[f] = \langle f, \phi_{\boldsymbol{x}} \rangle_{\mathcal{H}} \phi_{\boldsymbol{x}}.$$

As $\phi_{\boldsymbol{x}}$ is orthogonal to $\mathcal{H}^X$ by definition we know that $\{\varphi_1, \ldots, \varphi_l, \phi_{\boldsymbol{x}}\}$ is ONB of $\mathcal{H}^X \oplus \langle \phi_{\boldsymbol{x}} \rangle_{\mathcal{H}}$ and directly obtain

$$\mathcal{P}_{\mathcal{H}^X \oplus \langle \phi_{\boldsymbol{x}} \rangle_{\mathcal{H}}}[f] = \sum_{i=1}^{l} \langle f, \varphi_i \rangle_{\mathcal{H}} \varphi_i + \langle f, \phi_{\boldsymbol{x}} \rangle_{\mathcal{H}} \phi_{\boldsymbol{x}} = \mathcal{P}_{\mathcal{H}^X}[f] + \mathcal{P}_{\langle \phi_{\boldsymbol{x}} \rangle_{\mathcal{H}}}[f].$$

Next, equality (2.18) follows straightforwardly as both

$$\langle \mathcal{P}_{\mathcal{H}^X}[f], \phi_{\boldsymbol{x}} \rangle_{\mathcal{H}} = 0 = \langle f - \mathcal{P}_{\mathcal{H}^X}[f], \mathcal{P}_{\mathcal{H}^X}[K(\boldsymbol{x}, \cdot)] \rangle_{\mathcal{H}}$$

by projection properties:

$$\begin{aligned}\left\langle f, \tilde{\phi}_{\boldsymbol{x}} \right\rangle_{\mathcal{H}} &= \left\langle f - \mathcal{P}_{\mathcal{H}^X}[f], \tilde{\phi}_{\boldsymbol{x}} \right\rangle_{\mathcal{H}} \\ &= \langle f - \mathcal{P}_{\mathcal{H}^X}[f], K(\boldsymbol{x}, \cdot) - \mathcal{P}_{\mathcal{H}^X}[K(\boldsymbol{x}, \cdot)] \rangle_{\mathcal{H}} \\ &= \langle f - \mathcal{P}_{\mathcal{H}^X}[f], K(\boldsymbol{x}, \cdot) \rangle_{\mathcal{H}} = f(\boldsymbol{x}) - \mathcal{P}_{\mathcal{H}^X}[f](\boldsymbol{x}).\end{aligned}$$

Finally, the Riesz representation directly follows from (2.18). □

So, Lemma 2.2.5 allows to show the connection between both perspectives, which is established via the reproducing property of the RKHS. In equation (2.18) we see the scalar product of f with a dictionary element $K(\boldsymbol{x}, \cdot)$ in the spirit of general greedy algorithms, but also the point-wise difference $f(\boldsymbol{x}) - \mathcal{P}_{\mathcal{H}^X}[f](\boldsymbol{x})$, which is the same as $f(\boldsymbol{x}) - s_{f,X}(\boldsymbol{x})$ by Lemma 2.2.2 for non-singular kernel matrices. The expression $\langle f, \tilde{\phi}_{\boldsymbol{x}} \rangle_{\mathcal{H}}$ is equivalent to both, but isolates the f-dependency nicely by using a *modified* dictionary element, i.e. the orthogonal remainder of $K(\boldsymbol{x}, \cdot)$.

Remark 2.2.6. For any $X \subset \Omega$ and $\boldsymbol{x} \in \Omega \backslash \Omega_X$ the orthogonal remainder $\tilde{\phi}_{\boldsymbol{x}}$ actually corresponds to the direct translate $K_P^{\boldsymbol{x}}(\boldsymbol{x}, \cdot)$ of the "*Power-Kernel*" $K_P^{\boldsymbol{x}}$, see [119, 120].

The f- and f/P-Greedy algorithms

The initial idea behind the orthonormal remainder formalism was to develop a vectorial greedy algorithm that works on "modified" dictionary elements that would give the best possible improvement in the *native* norm instead of pure function value residuals. However, a closer investigation into literature revealed that the f/P-Greedy algorithm [117] produces this behaviour, though in a scalar setting and only remarked at by the authors on a side note. Hence, before introducing our vectorial formulation, we shall link both approaches and use the established nomenclature [117]. At this stage, we recall the concept of "*power functions*" (see [178, 11.2] or [117, 2.2.11]) and show the relation to orthogonal remainders.

Proposition 2.2.7 (Power function and orthogonal remainders)**.** *If the kernel matrix $\boldsymbol{K}$ is non-singular for $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \Omega$, then*

$$\left|\left|\tilde{\phi}_{\boldsymbol{x}}\right|\right|_{\mathcal{H}} = P_{K,X}(\boldsymbol{x}), \tag{2.19}$$

where $P_{K,X}(\boldsymbol{x})$ denotes the power function defined by

$$P_{K,X}(\boldsymbol{x})^2 := K(\boldsymbol{x},\boldsymbol{x}) - 2\sum_{i=1}^{n} u_i(\boldsymbol{x})K(\boldsymbol{x},\boldsymbol{x}_i) + \sum_{i,j}^{n} u_i(\boldsymbol{x})u_j(\boldsymbol{x})K(\boldsymbol{x}_i,\boldsymbol{x}_j).$$

Here $u_i, i = 1 \ldots n$ denotes the "Lagrange basis" *of $\mathcal{H}^X$, which is characterized by $u_i(\boldsymbol{x}_j) = \delta_{ij}$.*

Proof. The Lagrange basis is given by

$$u_j(\boldsymbol{x}) = \sum_{i=1}^{n} b_{ji}K(\boldsymbol{x}_i,\boldsymbol{x}),$$

where the condition $u_i(\boldsymbol{x}_j) = \delta_{ij}$ is satisfied when $\boldsymbol{K}\boldsymbol{b}_j = \boldsymbol{e}_j$, i.e. $\boldsymbol{b}_j = (\boldsymbol{K}^{-1})_j$. Here $\boldsymbol{e}_j$ denotes the i-th unit vector in $\mathbb{R}^n$ and $(\boldsymbol{K}^{-1})_j$ the j-th column of $\boldsymbol{K}^{-1}$. Next, recalling the kernel column vector shorthand (1.2, p.10) and Lemma 2.2.2 we see that

$$\begin{aligned}\mathcal{P}_{\mathcal{H}^X}[K(\boldsymbol{x},\cdot)] &= \sum_{j=1}^{n}(\boldsymbol{K}^{-1})_j^T\boldsymbol{k}(\boldsymbol{x})K(\boldsymbol{x}_j,\cdot) = \sum_{j=1}^{n}\sum_{i=1}^{n} K_{ij}^{-1}K(\boldsymbol{x},\boldsymbol{x}_i)K(\boldsymbol{x}_j,\cdot)\\ &= \sum_{i=1}^{n} K(\boldsymbol{x},\boldsymbol{x}_i)\sum_{j=1}^{n} K_{ij}^{-1}K(\boldsymbol{x}_j,\cdot) = \sum_{i=1}^{n} K(\boldsymbol{x},\boldsymbol{x}_i)u_j,\end{aligned}$$

with K_{ij}^{-1} denoting the ij-th entry of $\boldsymbol{K}^{-1}$. By definition of $\tilde{\phi}_{\boldsymbol{x}}$ this yields

$$\begin{aligned}\left|\left|\tilde{\phi}_{\boldsymbol{x}}\right|\right|_{\mathcal{H}}^2 &= \left\langle \tilde{\phi}_{\boldsymbol{x}}, \tilde{\phi}_{\boldsymbol{x}} \right\rangle_{\mathcal{H}}\\ &= \langle K(\boldsymbol{x},\cdot) - \mathcal{P}_{\mathcal{H}^X}[K(\boldsymbol{x},\cdot)], K(\boldsymbol{x},\cdot) - \mathcal{P}_{\mathcal{H}^X}[K(\boldsymbol{x},\cdot)]\rangle_{\mathcal{H}}\end{aligned}$$

$$= \left\langle K(\boldsymbol{x},\cdot) - \sum_{i=1}^{n} u_j K(\boldsymbol{x},\boldsymbol{x}_i), K(\boldsymbol{x},\cdot) - \sum_{i=1}^{n} u_j K(\boldsymbol{x},\boldsymbol{x}_i) \right\rangle_{\mathcal{H}}$$
$$= P_{K,X}(\boldsymbol{x})^2,$$

showing (2.19). □

For more background on power functions see e.g. [145, 35, 178, 117]. Even though both concepts are closely related, we will use the notion of orthogonal remainders as it will prove useful in our algorithm analysis.

With the necessary background established, we now state the scalar f- and f/P-Greedy algorithms [117, 3.1.1] using the orthogonal/-normal remainder formalism in Algorithm 2 and Algorithm 3. The equalities in (2.20) and

Algorithm 2 f-Greedy algorithm

Let $f \in \mathcal{H}$ and define $X_0 := \emptyset$, $f^0 := 0$ and for $m > 0$ the sequences

$$\boldsymbol{x}_m := \arg\max_{\boldsymbol{x}\in\Omega} \left| \left\langle f, \tilde{\phi}_{\boldsymbol{x}}^{m-1} \right\rangle_{\mathcal{H}} \right| = \arg\max_{\boldsymbol{x}\in\Omega} |f(\boldsymbol{x}) - f_{m-1}(\boldsymbol{x})|, \tag{2.20}$$
$$X_m := X_{m-1} \cup \{\boldsymbol{x}_m\},$$
$$f^m := \mathcal{P}_{\mathcal{H}^{X_m}}[f],$$

where $\tilde{\phi}_{\boldsymbol{x}}^m$ denotes the orthogonal remainder of $K(\boldsymbol{x},\cdot)$ with respect to X_m for any $m, \boldsymbol{x}$.

Algorithm 3 f/P-Greedy algorithm

Let $f \in \mathcal{H}$ and define $X_0 := \emptyset$, $f^0 := 0$ and for $m > 0$ the sequences

$$\boldsymbol{x}_m := \arg\max_{\boldsymbol{x}\in\Omega\setminus\Omega_{X_{m-1}}} \left| \left\langle f, \phi_{\boldsymbol{x}}^{m-1} \right\rangle_{\mathcal{H}} \right| = \arg\max_{\boldsymbol{x}\in\Omega\setminus\Omega_{X_{m-1}}} \frac{|f(\boldsymbol{x}) - f_{m-1}(\boldsymbol{x})|}{P_{K,X_{m-1}}(\boldsymbol{x})}, \tag{2.21}$$
$$X_m := X_{m-1} \cup \{\boldsymbol{x}_m\},$$
$$f^m := \mathcal{P}_{\mathcal{H}^{X_m}}[f],$$

where $\phi_{\boldsymbol{x}}^m$ denotes the orthonormal remainder of $K(\boldsymbol{x},\cdot)$ with respect to X_m for any $m, \boldsymbol{x}$.

(2.21) the can be easily verified using Lemma 2.2.5 and Proposition 2.2.7.

They build the bridge between the “dictionary” and “residual” perspectives, and we will discuss both algorithms and their relation in more detail for the vectorial setting in Section 2.2.2.

2.2.2 Vectorial kernel orthogonal greedy algorithm

Considering $\mathcal{H}^q$ for $q \geq 1$, it is clear that any scalar greedy approximation strategy can be straightforwardly applied to each component function f_j for some $\boldsymbol{f} \in \mathcal{H}^q$. But if one does not somehow connect the extension choices over the different component functions, the algorithms will most likely produce different subspaces $\mathcal{H}^{X_j}$ for each component f_j, leading to q disjoint kernel expansions in the worst case. This will be computationally infeasible, so the first and most obvious choice is to force all component approximations to stem from one global approximation subspace, i.e. $f_j \in \mathcal{H}^X \; \forall \; j$ for some base space $\mathcal{H}^X$. This restriction is given if we use the following vectorial projection operator on $\mathcal{H}^q$.

Definition 2.2.8 (Vectorial component-wise projection operator)**.** Let $S \subseteq \mathcal{H}$ be a linear subspace and $q \in \mathbb{N}$. Then we define the vectorial orthogonal projection operator

$$\begin{aligned} \mathcal{P}_S^q : \mathcal{H}^q &\longrightarrow S^q \\ \boldsymbol{f} &\longmapsto \left(\mathcal{P}_S[f_j]\right)_j , \quad j = 1 \dots q. \end{aligned}$$

It is easily verifiable that for this definition we have $\langle \boldsymbol{f} - \mathcal{P}_S^q[\boldsymbol{f}], \boldsymbol{g} \rangle_{\mathcal{H}^q} = 0 \; \forall \; \boldsymbol{g} \in S^q$ as

$$S^q = \langle \{ \boldsymbol{e}_i g \mid i = 1 \dots q, \; g \in S \} \rangle ,$$

with $\boldsymbol{e}_i$ denoting the i-th unit vector in $\mathbb{R}^q$.

Consequently, the scalar algorithms can be formulated straightforwardly also in the vectorial context. Note that a vectorial regression approach can be found in [159] or multioutput orthogonal least squares approximations are discussed in [26]. The vectorial version of the f-Greedy Algorithm 2 can already be found in a general Hilbert space setting in e.g. [104, §3, (3.4)] named "WSOGA 2". As our focus is the vectorial formulation of the f/P-Greedy algorithm, we shall quickly state the vectorial f-Greedy variant fitted to our RKHS setting in Algorithm 4 and call it f-VKOGA for simplicity.

Algorithm 4 f-VKOGA

Let K be a s.p.d. normalized kernel with native space $\mathcal{H}$. Further let $\boldsymbol{f} \in \mathcal{H}^q$, define $X_0 := \emptyset$, $\boldsymbol{f}^0 := 0$ and for $m > 0$ the sequences

$$\boldsymbol{x}_m := \arg\max_{\boldsymbol{x}\in\Omega} \sum_{j=1}^{q} \left\langle f_j, \tilde{\phi}^m_{\boldsymbol{x}} \right\rangle^2_{\mathcal{H}} = \arg\max_{\boldsymbol{x}\in\Omega} ||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}^m(\boldsymbol{x})||_2^2, \qquad (2.22)$$

$$X_m := X_{m-1} \cup \{\boldsymbol{x}_m\},$$

$$f_j^m := \mathcal{P}^q_{\mathcal{H}^{X_m}}[f_j], \qquad \boldsymbol{f}^m = (f_1^m, \dots, f_q^m)^T.$$

Now, an important feature of the scalar f/P-Greedy algorithm is that each extension step maximizes the $\mathcal{H}$-norm of $\mathcal{P}_{\mathcal{H}^X}[f]$ (the interpolant in [117, 3.1.4], see also [150, Thm 6.], [37]), which is equivalent to achieving the largest possible "gain" in approximation w.r.t. the native space due to the orthogonality properties $||\mathcal{P}_{\mathcal{H}^X}[f]||^2_{\mathcal{H}} = ||f||^2_{\mathcal{H}} - ||f - \mathcal{P}_{\mathcal{H}^X}[f]||^2_{\mathcal{H}}$. This aspect is not taken into account by the vectorial greedy algorithms proposed in [103, 104], which pursue a vectorial greedy search in the fashion of the standard scalar f-Greedy variant. However, it remains to verify that the transferred vectorial selection criteria of Algorithm 3 inherits this property. The concept of a *gain function* will prove useful in this context and for subsequent analysis.

Definition 2.2.9 (Gain function)**.** Let $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \subseteq \Omega$ and $\boldsymbol{f} \in \mathcal{H}^q$. Then we denote the gain function with respect to X and $\boldsymbol{f}$ by

$$G_{X,\boldsymbol{f}} : \Omega \backslash \Omega_X \longrightarrow \mathbb{R}$$

$$\boldsymbol{x} \longmapsto \sum_{j=1}^{q} \langle f_j, \phi_{\boldsymbol{x}} \rangle^2_{\mathcal{H}},$$

where $\phi_{\boldsymbol{x}}$ denotes the orthonormal remainder of $K(\boldsymbol{x},\cdot)$ with respect to X.

For our choice of vectorial RKHS, the following Lemma characterizes the $\mathcal{H}^q$-norm maximizing aspect of the f/P-Greedy algorithm using the gain function.

Lemma 2.2.10 (Gain for vectorial subspace extension). *Let $\boldsymbol{f} \in \mathcal{H}^q$, $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \subseteq \Omega$, $q \in \mathbb{N}$ and $\boldsymbol{x} \in \Omega \backslash \Omega_X$. Then*

$$\left|\left|\boldsymbol{f} - \mathcal{P}^q_{\mathcal{H}^X \oplus \langle K(\boldsymbol{x},\cdot)\rangle}[\boldsymbol{f}]\right|\right|^2_{\mathcal{H}^q} = \left|\left|\boldsymbol{f} - \mathcal{P}^q_{\mathcal{H}^X}[\boldsymbol{f}]\right|\right|^2_{\mathcal{H}^q} - G_{X,\boldsymbol{f}}(\boldsymbol{x}). \tag{2.23}$$

Proof. At first we note that in fact $\mathcal{H}^X \oplus \langle K(\boldsymbol{x},\cdot)\rangle = \mathcal{H}^X \oplus \langle \phi_{\boldsymbol{x}}\rangle$, which follows directly from the definition of $\phi_{\boldsymbol{x}}$ as $\mathcal{P}_{\mathcal{H}^X}[K(\boldsymbol{x},\cdot)] \in \mathcal{H}^X$. Moreover, for an $f \in \mathcal{H}$ we have $\mathcal{P}_{\langle \phi_{\boldsymbol{x}}\rangle}[f] = \langle f, \phi_{\boldsymbol{x}}\rangle_{\mathcal{H}} \phi_{\boldsymbol{x}}$ by (2.15, p.49). Then using Lemma 2.2.2 and 2.2.5 we deduce

$$\begin{aligned}
&\left|\left|f - \mathcal{P}_{\mathcal{H}^X \oplus \langle K(\boldsymbol{x},\cdot)\rangle}[f]\right|\right|^2_{\mathcal{H}} \\
=&\left|\left|f - \mathcal{P}_{\mathcal{H}^X \oplus \langle \phi_{\boldsymbol{x}}\rangle}[f]\right|\right|^2_{\mathcal{H}} \\
=&\left|\left|f - \mathcal{P}_{\mathcal{H}^X}[f] - \mathcal{P}_{\langle \phi_{\boldsymbol{x}}\rangle}[f]\right|\right|^2_{\mathcal{H}} \\
=&\left|\left|f - \mathcal{P}_{\mathcal{H}^X}[f]\right|\right|^2_{\mathcal{H}} - 2\left\langle f - \mathcal{P}_{\mathcal{H}^X}[f], \mathcal{P}_{\langle \phi_{\boldsymbol{x}}\rangle}[f]\right\rangle_{\mathcal{H}} + \left|\left|\mathcal{P}_{\langle \phi_{\boldsymbol{x}}\rangle}[f]\right|\right|^2_{\mathcal{H}} \\
=&\left|\left|f - \mathcal{P}_{\mathcal{H}^X}[f]\right|\right|^2_{\mathcal{H}} - 2\left\langle f, \mathcal{P}_{\langle \phi_{\boldsymbol{x}}\rangle}[f]\right\rangle_{\mathcal{H}} \\
&+ 2\underbrace{\left\langle \mathcal{P}_{\mathcal{H}^X}[f], \mathcal{P}_{\langle \phi_{\boldsymbol{x}}\rangle}[f]\right\rangle_{\mathcal{H}}}_{=0} + \left\langle f, \mathcal{P}_{\langle \phi_{\boldsymbol{x}}\rangle}[f]\right\rangle_{\mathcal{H}} \\
=&\left|\left|f - \mathcal{P}_{\mathcal{H}^X}[f]\right|\right|^2_{\mathcal{H}} - \langle f, \phi_{\boldsymbol{x}}\rangle^2_{\mathcal{H}}.
\end{aligned}$$

Using the definition of $\mathcal{P}^q$ and $G_{X,\boldsymbol{f}}$ we obtain

$$\begin{aligned}
\left|\left|\boldsymbol{f} - \mathcal{P}^q_{\mathcal{H}^X \oplus \langle K(\boldsymbol{x},\cdot)\rangle}[\boldsymbol{f}]\right|\right|^2_{\mathcal{H}^q} &= \sum_{j=1}^{q} \left|\left|f_j - \mathcal{P}_{\mathcal{H}^X \oplus \langle K(\boldsymbol{x},\cdot)\rangle}[f_j]\right|\right|^2_{\mathcal{H}} \\
&= \sum_{j=1}^{q} \left|\left|f_j - \mathcal{P}_{\mathcal{H}^X}[f_j]\right|\right|^2_{\mathcal{H}} - \langle f_j, \phi_{\boldsymbol{x}}\rangle^2_{\mathcal{H}}
\end{aligned}$$

$$= ||\boldsymbol{f} - \mathcal{P}_{\mathcal{H}^X}[\boldsymbol{f}]||^2_{\mathcal{H}^q} - G_{X,\boldsymbol{f}}(\boldsymbol{x}).$$

□

Since the orthogonal projection into a linear subspace always gives the best possible approximation in that space, a direct consequence is the following Corollary.

Corollary 2.2.11. *Let the conditions from Lemma 2.2.10 hold. Then*

$$\inf_{\boldsymbol{x}\in\Omega\backslash\Omega_X} \min_{\boldsymbol{g}\in(\mathcal{H}^X\oplus\langle K(\boldsymbol{x},\cdot)\rangle)^q} ||\boldsymbol{f}-\boldsymbol{g}||^2_{\mathcal{H}^q} = C - \sup_{\boldsymbol{x}\in\Omega\backslash\Omega_X} G_{X,\boldsymbol{f}}(\boldsymbol{x}),$$

with $C := \left|\left|\boldsymbol{f} - \mathcal{P}^q_{\mathcal{H}^X}[\boldsymbol{f}]\right|\right|^2_{\mathcal{H}^q}$.

So, in fact, any $\boldsymbol{x} \in \Omega\backslash\Omega_X$ that yields the best approximation in $\mathcal{H}^{X\cup\{\boldsymbol{x}\}}$ also maximizes $G_{X,\boldsymbol{f}}$. Seen the other way around, the largest possible local approximation gain with respect to the $\mathcal{H}^q$-norm is achieved by maximizing the gain function. Consequently, we state in Algorithm 5 the vectorial counterpart of Algorithm 3 using the gain function. A recursive applica-

Algorithm 5 f/P-VKOGA)

Let $q \in \mathbb{N}$ and $\boldsymbol{f} \in \mathcal{H}^q$, define $X_0 := \emptyset$, $\boldsymbol{f}^0 := 0$ and for $m > 0$ the sequences

$$\boldsymbol{x}_m := \arg\max_{\boldsymbol{x}\in\Omega\backslash\Omega_{X_{m-1}}} G_{X_{m-1},\boldsymbol{f}}(\boldsymbol{x}) = \arg\max_{\boldsymbol{x}\in\Omega\backslash\Omega_{X_{m-1}}} \sum_{j=1}^q \left\langle f_j, \phi^{m-1}_{\boldsymbol{x}}\right\rangle^2_{\mathcal{H}}, \tag{2.24}$$

$$X_m := X_{m-1} \cup \{\boldsymbol{x}_m\}, \tag{2.25}$$

$$\boldsymbol{f}^m := \mathcal{P}^q_{\mathcal{H}^{X_m}}[\boldsymbol{f}]. \tag{2.26}$$

tion of Lemma 2.2.10 yields the following result on the error decay and a Parseval-type identity.

Corollary 2.2.12. *Then the $\mathcal{H}^q$-approximation error is monotonously decreas-*

ing and for all $m > 0$ we have the identity

$$||\boldsymbol{f} - \boldsymbol{f}^m||^2_{\mathcal{H}^q} = ||\boldsymbol{f}||^2_{\mathcal{H}^q} - \sum_{i=1}^{m}\sum_{j=1}^{q} \left\langle f_j, \phi^{i-1}_{\boldsymbol{x}_i} \right\rangle^2_{\mathcal{H}} = ||\boldsymbol{f}||^2_{\mathcal{H}^q} - \sum_{i=1}^{m} G_{X_{i-1},\boldsymbol{f}}(\boldsymbol{x}_i).$$

2.2.2.1 Algorithm analysis

In this section we discuss some analytical properties of the VKOGA variants, perform a comparison and investigate the behaviour of limit functions $\phi_{\boldsymbol{x}}$ in the vicinity of points $\boldsymbol{x} \in \Omega_X$. Let $X \subset \Omega$ be given and denote by $\boldsymbol{x}^f, \boldsymbol{x}^{fP}$ the subspace extension choices of the Algorithms 4 and 5, respectively. Then we see that

$$\sum_{j=1}^{q} \left\langle f_j, \tilde{\phi}_{\boldsymbol{x}^f} \right\rangle^2_{\mathcal{H}} \leq \sum_{j=1}^{q} \left\langle f_j, \phi_{\boldsymbol{x}^f} \right\rangle^2_{\mathcal{H}} \leq \max_{\boldsymbol{x}\in\Omega} \sum_{j=1}^{q} \left\langle f_j, \phi_{\boldsymbol{x}} \right\rangle^2_{\mathcal{H}} = G_{X,\boldsymbol{f}}(\boldsymbol{x}^{fP}).$$

by the selection criteria definitions. This means the f/P-VKOGA algorithm will locally always make as good a choice as the f-VKOGA algorithm. Unfortunately, as the successive spaces constructed by both algorithms will in general be different, it remains an open question to us if and how to compare the performance of both variants directly at some given subspace size $m > 0$. However, with the help of Lemma 2.2.5, the f/P-VKOGA extension choice criteria (2.24) can also be written as

$$\max_{\boldsymbol{x}\in\Omega} \sum_{j=1}^{q} \left\langle f_j, \phi^{m-1}_{\boldsymbol{x}} \right\rangle^2_{\mathcal{H}} = \max_{\boldsymbol{x}\in\Omega\setminus\Omega_{X_{m-1}}} \frac{||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}^m(\boldsymbol{x})||^2_2}{||K(\boldsymbol{x},\cdot) - \mathcal{P}_{\mathcal{H}^{X_{m-1}}}[K(\boldsymbol{x},\cdot)]||^2_{\mathcal{H}}}.$$

Since the numerator equals the f-VKOGA choice (2.22, p.56), which basically considers *any* maximizing point $\boldsymbol{x}$ to be equally good for extension, the f/P-VKOGA choice scales inversely with how well the associated dictionary element or translate $K(\boldsymbol{x},\cdot)$ is already approximated by $\mathcal{H}^X$. This way, identical point-wise approximation errors closer to the points whose dictionary elements span $\mathcal{H}^X$ are considered to be worse than others. We would like to note that both algorithms can be continuously transferred over to each other by thresholding the orthogonal remainder norms at a certain value, i.e. we use the selection criteria

$$\max_{\boldsymbol{x}\in\Omega\setminus\Omega_{X_{m-1}}} \frac{||\boldsymbol{f}(\boldsymbol{x})-\boldsymbol{f}^m(\boldsymbol{x})||_2^2}{\max\{\delta, ||K(\boldsymbol{x},\cdot)-\mathcal{P}_{\mathcal{H}^{X_{m-1}}}[K(\boldsymbol{x},\cdot)]||_{\mathcal{H}}^2\}}$$

for some $\delta \in [0,1]$. Then, $\delta = 0$ and $\delta = 1$ results in the f/P- and f-VKOGA variants, respectively. This opens up a large variety of algorithm variants and the version most suitable for the current situation can be selected. Moreover, as the norm of any orthogonal remainder is independent of the considered $\boldsymbol{f} \in \mathcal{H}^q$, this can be interpreted as how well all functions involving the dictionary element $K(\boldsymbol{x},\cdot)$ in general are already approximated in $\mathcal{H}^q$. This concept is pursued directly by the P-Greedy algorithm mentioned e.g. in [117], which aims to create data-independent approximations of the function space and leads to a very uniform distribution of the selected $\boldsymbol{x}_m$.

Figure 2.3 illustrates the selection criteria using two simple scalar examples, which shows the different extension choices at $m = 6$ already given points along with the respective gain functions. The red and blue crosses mark the VKOGA and f-VKOGA selection and the dashed/solid green line the gain functions of the VKOGA/f-VKOGA algorithms, respectively. As test setting, a Gaussian with $\gamma^2 = 2.1715$ and $X = \{-7.1, -5.6, -2.1, 1.9, 5.9, 8.4\}$ have been used, whose induced RKHS serves as native space for both dictionary and test function f. The expansion coefficients of f_1 on the left, f_2 on the right hand side are

$$\boldsymbol{a}_1 = (-2.0465, 2.3066, -0.2428, 0.6805, -2.1213, -1.4411)$$

and

$$\boldsymbol{a}_2 = (1.1702, -0.2113, -0.7158, -0.5346, -1.1990, -1.1459).$$

Note that it is clear to see that the gain function of the VKOGA algorithm is

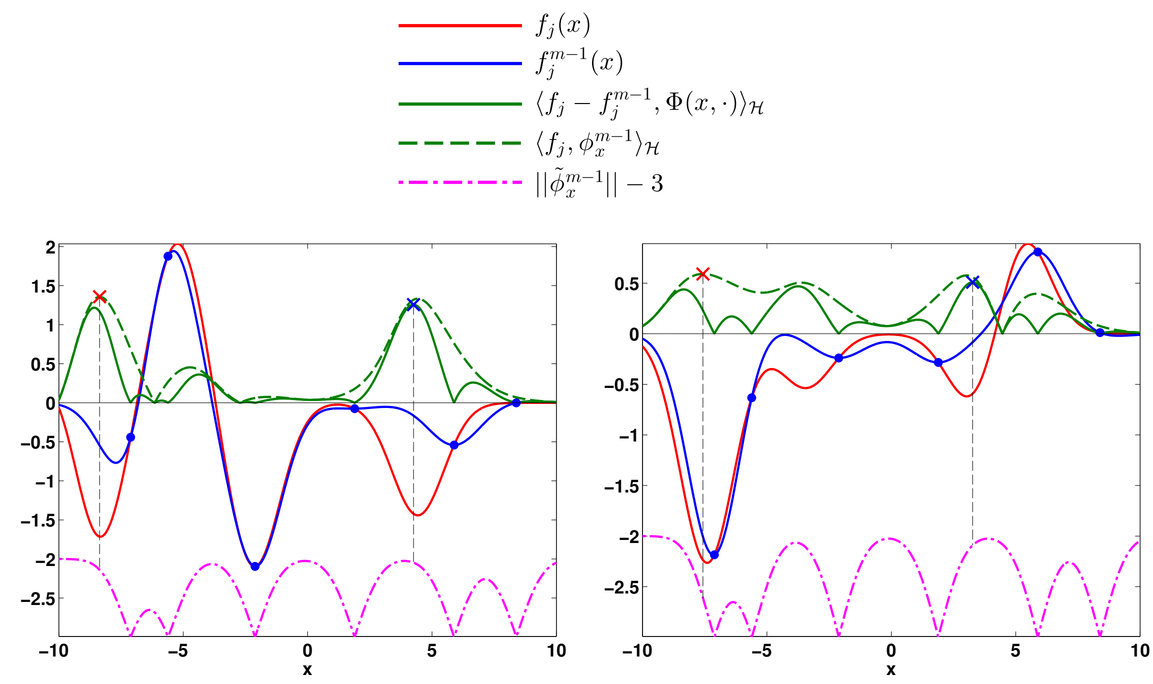


Figure 2.3: Example for functions f_1, f_2 with different selections of points using f or f/P strategy.

indeed continuous as mentioned earlier in Remark 2.2.18.

For both cases we see that the VKOGA choice selects spatially very different extension points, even though the gain of both algorithms is not very different. While in the case of f_1 (left) the extension points are selected "well away" from any existing point, the VKOGA extension selects x_7 very close to the existing center at -7.1. There it is evident that, even though the absolute approximation error is bigger elsewhere, the error weighted by the orthogonal remainder norm causes this location to be considered most worth improving. On the downside, especially because this choice involves a function-independent but RKHS-specific part, the choice of f/P VKOGA can be ill-suited if the considered function does not stem from the same RKHS as the dictionary elements. In this situation, adding more points near the same location causes the denominator to decay much faster than the nominator, as the function itself is locally not approximated as well as the "local" RKHS part. This effect can also be seen in e.g. [117, 6.1]. In our opinion, instead of considering the f/P-VKOGA variant a bad choice when the origin of the target function is unknown, we think this effect might be actively used to formulate an indicator for the foremost mentioned situation. If a considered function is "detected" not to belong to the currently chosen RKHS, one can proceed with another choice of RKHS, e.g. a different hyperparameter for the RKHS inducing kernel. We will pursue a more detailed numerical comparison of the VKOGA variants in Section 2.2.2.4.

Until now we have only considered points $\boldsymbol{x} \in \Omega \backslash \Omega_X$ for possible extension of $\mathcal{H}^X$ via the induced kernel translate $K(\boldsymbol{x}, \cdot)$. So far in literature, it remains unanswered what happens in the vicinity of points $\boldsymbol{x} \in \Omega_X$. This motivated an analysis of the behaviour of $G_{X,\boldsymbol{f}}$ in the neighborhood of Ω_X. The following Theorem yields an explicit expression for the limit functions $\phi_{\boldsymbol{x}}$ when $\boldsymbol{x} \to \boldsymbol{y} \in \Omega_X$. Under mild assumptions, it turns out that the limits in these cases can be described by orthonormal remainders of directional derivatives of $K(\boldsymbol{x}, \cdot)$.

Theorem 2.2.13 (Directional limit of orthonormal remainders)**.** *Let $K \in C^1(\Omega \times \Omega)$ (C^1 w.r.t. each argument) with existing and bounded second derivatives. Further let $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \Omega$ so that the kernel matrix $\boldsymbol{K}$ is non-singular and choose $\boldsymbol{x} \in \Omega_X$. Then $\forall\ \boldsymbol{v} \in \mathbb{R}^d$ we have*

$$\lim_{h \to 0} \phi_{\boldsymbol{x}+h\boldsymbol{v}} = \phi_{\boldsymbol{x}}^{\nabla v} \in \mathcal{H}, \tag{2.27}$$

with

$$\phi_{\boldsymbol{x}}^{\nabla v} := \begin{cases} \tilde{\phi}_{\boldsymbol{x}}^{\nabla v} \Big/ \left|\left|\tilde{\phi}_{\boldsymbol{x}}^{\nabla v}\right|\right|_{\mathcal{H}} & , \ \tilde{\phi}_{\boldsymbol{x}}^{\nabla v} \neq 0 \\ 0 & , \ \textit{else} \end{cases},$$
$$\tilde{\phi}_{\boldsymbol{x}}^{\nabla v} := \boldsymbol{v}^T \nabla_1 K(\boldsymbol{x}, \cdot) - \mathcal{P}_{\mathcal{H}^X}[\boldsymbol{v}^T \nabla_1 K(\boldsymbol{x}, \cdot)],$$

where ∇_1 denotes the gradient operator w.r.t. the first argument.

Proof. Fix $\boldsymbol{v}$. By Lemma 1.2.7 we know that $\boldsymbol{v}^T \nabla_1 K(\boldsymbol{x}, \cdot) \in \mathcal{H}$. Further, as $K(\boldsymbol{x}, \cdot) \in \mathcal{H}^X$, Lemma 2.2.2 / (2.16, p.49) with $f = K(\boldsymbol{x}, \cdot)$ gives the representation

$$K(\boldsymbol{x},\cdot) = \sum_{j=1}^{n} (\boldsymbol{K}^{-1})_j^T \boldsymbol{k}(\boldsymbol{x}) K(\boldsymbol{x}_j,\cdot). \tag{2.28}$$

Note that for $\boldsymbol{x} = \boldsymbol{x}_k, k \in \{1 \dots n\}$ we have $\boldsymbol{k}(\boldsymbol{x}_k) = \boldsymbol{K}_k$, which simplifies equation (2.28) to $(\boldsymbol{K}^{-1})_j^T \boldsymbol{k}(\boldsymbol{x}_k) = \delta_{jk}$. Now, for $h > 0$, the first order multivariate Taylor series of K at $\boldsymbol{x}$ gives

$$K(\boldsymbol{x}+h\boldsymbol{v},\cdot) = K(\boldsymbol{x},\cdot) + h\boldsymbol{v}^T\nabla_1 K(\boldsymbol{x},\cdot) + h^2 C_K(\boldsymbol{\xi},\cdot), \tag{2.29}$$

with $\boldsymbol{\xi} \in [\boldsymbol{x}, \boldsymbol{x}+h\boldsymbol{v}]$. The h-dependency of $C_K(\boldsymbol{\xi},\cdot)$ can be neglected due to the bounded second derivatives, which is why we will just write $\mathcal{O}\left(h^2\right)$ in the following. Next, with the shorthand

$$\nabla\boldsymbol{K}(\boldsymbol{x}) = \Big(\nabla_1 K(\boldsymbol{x},\boldsymbol{x}_1) \dots \nabla_1 K(\boldsymbol{x},\boldsymbol{x}_n)\Big) \in \mathbb{R}^{d\times n},$$

equation (2.29) directly gives the representation

$$\boldsymbol{k}(\boldsymbol{x}+h\boldsymbol{v}) = \boldsymbol{k}(\boldsymbol{x}) + h\nabla\boldsymbol{K}(\boldsymbol{x})^T\boldsymbol{v} + \mathcal{O}\left(h^2\right).$$

Together with (2.16, p.49) and (2.28) we see that

$$\begin{aligned}
&\mathcal{P}_{\mathcal{H}^X}[K(\boldsymbol{x}+h\boldsymbol{v},\cdot)]\\
&= \sum_{j=1}^{n} (\boldsymbol{K}^{-1})_j^T \boldsymbol{k}(\boldsymbol{x}+h\boldsymbol{v}) K(\boldsymbol{x}_j,\cdot)\\
&= \sum_{j=1}^{n} (\boldsymbol{K}^{-1})_j^T \left(\boldsymbol{k}(\boldsymbol{x}) + h\nabla\boldsymbol{K}(\boldsymbol{x})^T\boldsymbol{v}\right) K(\boldsymbol{x}_j,\cdot) + \mathcal{O}\left(h^2\right)\\
&= K(\boldsymbol{x},\cdot) + h\sum_{j=1}^{n} (\boldsymbol{K}^{-1})_j^T \nabla\boldsymbol{K}(\boldsymbol{x})^T\boldsymbol{v} K(\boldsymbol{x}_j,\cdot) + \mathcal{O}\left(h^2\right)\\
&= K(\boldsymbol{x},\cdot) + h\sum_{j=1}^{n} (\boldsymbol{K}^{-1})_j^T \begin{pmatrix} \boldsymbol{v}^T\nabla_1 K(\boldsymbol{x},\boldsymbol{x}_1)\\ \vdots \\ \boldsymbol{v}^T\nabla_1 K(\boldsymbol{x},\boldsymbol{x}_n)\end{pmatrix} K(\boldsymbol{x}_j,\cdot) + \mathcal{O}\left(h^2\right)
\end{aligned}$$

$$= K(\boldsymbol{x},\cdot) + h\mathcal{P}_{\mathcal{H}^X}[\boldsymbol{v}^T\nabla_1 K(\boldsymbol{x},\cdot)] + \mathcal{O}\left(h^2\right).$$

Using (2.29) again we obtain the representation

$$\begin{aligned}\tilde{\phi}_{\boldsymbol{x}+h\boldsymbol{v}} &= K(\boldsymbol{x}+h\boldsymbol{v},\cdot) - \mathcal{P}_{\mathcal{H}^X}[K(\boldsymbol{x}+h\boldsymbol{v},\cdot)] \\ &= h\boldsymbol{v}^T\nabla_1 K(\boldsymbol{x},\cdot) - \mathcal{P}_{\mathcal{H}^X}[\boldsymbol{v}^T\nabla_1 K(\boldsymbol{x},\cdot)] + \mathcal{O}\left(h^2\right) \\ &= h\tilde{\phi}_{\boldsymbol{x}}^{\nabla v} + \mathcal{O}\left(h^2\right).\end{aligned} \tag{2.30}$$

We conclude

$$\lim_{h\to 0}\phi_{\boldsymbol{x}+h\boldsymbol{v}} = \lim_{h\to 0}\frac{\tilde{\phi}_{\boldsymbol{x}+h\boldsymbol{v}}}{\left\|\tilde{\phi}_{\boldsymbol{x}+h\boldsymbol{v}}\right\|_{\mathcal{H}}} = \lim_{h\to 0}\frac{\tilde{\phi}_{\boldsymbol{x}}^{\nabla v} + \mathcal{O}(h)}{\left\|\tilde{\phi}_{\boldsymbol{x}}^{\nabla v}\right\|_{\mathcal{H}} + \mathcal{O}(h)} = \phi_x^{\nabla v},$$

which shows (2.27, p.62) for $\tilde{\phi}_{\boldsymbol{x}}^{\nabla v} \neq 0$. □

Now, Theorem 2.2.13 allows to draw interesting conclusions with regard to the situations where $\boldsymbol{x} \in \Omega_X$, i.e. $K(\boldsymbol{x},\cdot) \in \mathcal{H}^X$. At first we see that for any $X = \{\boldsymbol{x}_1,\dots,\boldsymbol{x}_n\}, \boldsymbol{x} \in \Omega_X, \boldsymbol{v} \in \mathbb{R}^d$ and $\boldsymbol{f} \in \mathcal{H}^q$ we have

$$\lim_{h\to 0} G_{X,\boldsymbol{f}}(\boldsymbol{x}+h\boldsymbol{v}) = \sum_{j=1}^{q}\left\langle f_j, \phi_{\boldsymbol{x}}^{\nabla v}\right\rangle_{\mathcal{H}}^2.$$

The resulting limit value depends on the direction $\boldsymbol{v}$ from which the limit is taken, which implies that $G_{X,\boldsymbol{f}}$ cannot be continuously extended to Ω in general. To illustrate the occurring discontinuities, Figure 2.4 shows the values of $\langle f, \phi_{\boldsymbol{x}}\rangle_{\mathcal{H}}$ (which corresponds to $G_{X,\boldsymbol{f}}$ in $q = 1$ dimensions without the squared scalar product) for the test settings $q = 1, \Omega = [-4,4]^2, X = \{(0,1),(-0.5,0),(2,-1),(-1,3),(-1.5,-3)\}$ and a suitable $f \in \mathcal{H}$. The discontinuities are clearly recognizable around any point $\boldsymbol{x} \in X$, which are marked by red dots. Furthermore, for each $\boldsymbol{v} \in \mathbb{R}^d$ and $\boldsymbol{f} \in \mathcal{H}^q$ equation

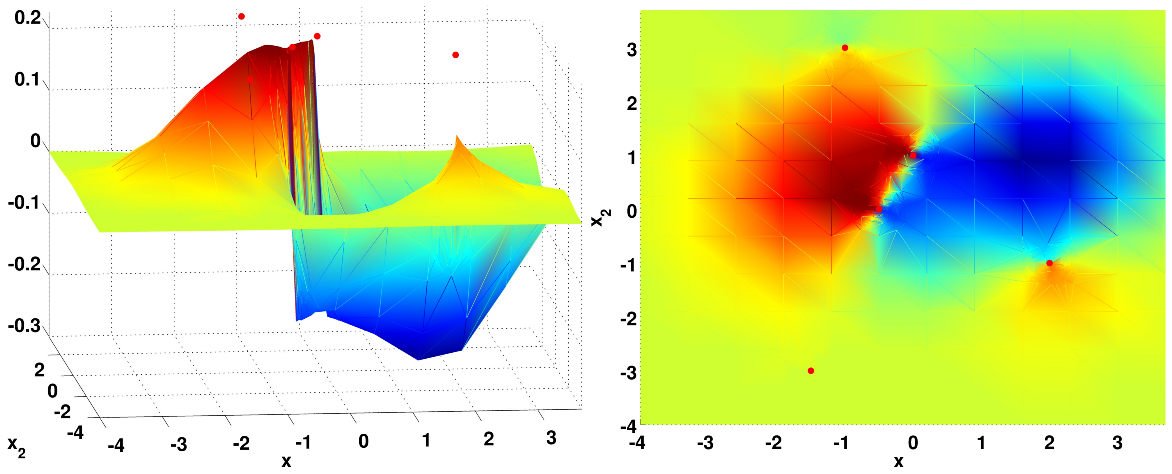


Figure 2.4: Example of $\langle f, \phi_{\boldsymbol{x}} \rangle_{\mathcal{H}}$ on $\Omega \backslash \Omega_X$. Red dots: X

(2.23, p.57) now reads as

$$\lim_{h \to 0} \left|\left| \boldsymbol{f} - \mathcal{P}^q_{\mathcal{H}^X \oplus \langle K(\boldsymbol{x}+h\boldsymbol{v},\cdot)\rangle}[\boldsymbol{f}] \right|\right|^2_{\mathcal{H}^q} = \left|\left| \boldsymbol{f} - \mathcal{P}^q_{\mathcal{H}^X}[\boldsymbol{f}] \right|\right|^2_{\mathcal{H}^q} - \sum_{j=1}^{q} \left\langle f_j, \phi^{\nabla v}_{\boldsymbol{x}} \right\rangle^2_{\mathcal{H}}.$$

Hence, the possible different $\phi^{\nabla v}_{\boldsymbol{x}}$ at any $\boldsymbol{x} \in \Omega_X$ imply that the limit of the left hand side also differs with changing $\boldsymbol{v}$. This raises the question about the projection limit as $h \to 0$, which is answered satisfactorily by the following corollary.

Corollary 2.2.14. *Let the conditions from Theorem 2.2.13 hold and let $\boldsymbol{f} \in \mathcal{H}^q$. Then we have*

$$\lim_{h \to 0} \mathcal{P}^q_{\mathcal{H}^X \oplus \langle K(\boldsymbol{x}+h\boldsymbol{v},\cdot)\rangle}[\boldsymbol{f}] = \mathcal{P}^q_{\mathcal{H}^X \oplus \langle \boldsymbol{v}^T \nabla_1 K(\boldsymbol{x},\cdot)\rangle}[\boldsymbol{f}].$$

Proof. For each component function $f_j, j = 1 \dots q$ we use the relation (2.30) to obtain

$$\begin{aligned}
\mathcal{P}_{\mathcal{H}^X \oplus \langle K(\boldsymbol{x}+h\boldsymbol{v},\cdot)\rangle}[f_j] &= \mathcal{P}_{\mathcal{H}^X}[f_j] + \mathcal{P}_{\langle \tilde{\phi}_{\boldsymbol{x}+h\boldsymbol{v}} \rangle}[f_j] \\
&= \mathcal{P}_{\mathcal{H}^X}[f_j] + \frac{\left\langle f_j, \tilde{\phi}_{\boldsymbol{x}+h\boldsymbol{v}} \right\rangle_{\mathcal{H}}}{\left|\left| \tilde{\phi}_{\boldsymbol{x}+h\boldsymbol{v}} \right|\right|^2_{\mathcal{H}}} \tilde{\phi}_{\boldsymbol{x}+h\boldsymbol{v}}
\end{aligned}$$

$$\begin{aligned}
&= \mathcal{P}_{\mathcal{H}^X}[f_j] + \frac{\left\langle f_j, \tilde{\phi}_{\boldsymbol{x}}^{\nabla v} \right\rangle_{\mathcal{H}}}{\left\| \tilde{\phi}_{\boldsymbol{x}}^{\nabla v} \right\|_{\mathcal{H}}^2} \tilde{\phi}_{\boldsymbol{x}}^{\nabla v} + \mathcal{O}\left(h^2\right) \\
&= \mathcal{P}_{\mathcal{H}^X}[f_j] + \mathcal{P}_{\langle \tilde{\phi}_{\boldsymbol{x}}^{\nabla v} \rangle}[f_j] + \mathcal{O}\left(h^2\right) \\
&= \mathcal{P}_{\mathcal{H}^X \oplus \langle \boldsymbol{v}^T \nabla_1 K(\boldsymbol{x}, \cdot) \rangle}[f_j] + \mathcal{O}\left(h^2\right).
\end{aligned}$$

Taking the limit $h \to 0$ and recalling the component-wise action of $\mathcal{P}^q$ finishes the proof. □

Remark 2.2.15. The expression $\mathcal{P}^q_{\mathcal{H}^X \oplus \langle \boldsymbol{v}^T \nabla_1 K(\boldsymbol{x}, \cdot) \rangle}[\boldsymbol{f}]$ actually corresponds to a simultaneous, component-wise directional Hermite interpolation in $\mathcal{H}^q$ since

$$\left\langle f_j, \boldsymbol{v}^T \nabla_1 K(\boldsymbol{x}, \cdot) \right\rangle_{\mathcal{H}} = \boldsymbol{v}^T \nabla_1 f_j(\boldsymbol{x}), \quad j = 1 \dots n,$$

which can be easily verified using Lemma 1.2.7. Interestingly enough, this means that the closer a considered point approaches an already included one from the direction $\boldsymbol{v}$, the "direct extension" gain is converging towards the gain that would be achieved including $\boldsymbol{v}^T \nabla_1 K(\boldsymbol{x}, \cdot)$ into $\mathcal{H}^X$, i.e. the directional derivative.

We conclude our analysis of $G_{X,\boldsymbol{f}}$ with the following Theorem.

Theorem 2.2.16 (Gain function characterization)**.** *Let $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \subseteq \Omega$ and $\boldsymbol{f} \in \mathcal{H}^q$. Then $G_{X,\boldsymbol{f}}$ is continuous on $\Omega \backslash \Omega_X$ and $\forall\ \boldsymbol{x} \in \Omega_X$ exists a neighborhood of $\boldsymbol{x}$ on which $G_{X,\boldsymbol{f}}$ is bounded.*

Proof. Let $\boldsymbol{x} \in \Omega \backslash \Omega_X$. Then for any $\boldsymbol{v} \in \mathbb{R}^d$ a similar argumentation with Taylor series as in the proof of Theorem 2.2.13 shows that

$$\lim_{h \to 0} ||\phi_{\boldsymbol{x}} - \phi_{\boldsymbol{x}+h\boldsymbol{v}}||_{\mathcal{H}} = 0,$$

from which we obtain continuity as $\lim_{\boldsymbol{y} \to \boldsymbol{x}} G_{X,\boldsymbol{f}}(\boldsymbol{y}) = G_{X,\boldsymbol{f}}(\boldsymbol{x})$. Next, for

$\boldsymbol{x} \in \Omega_X$ and an ϵ with $\overline{B_\epsilon(\boldsymbol{x})} \subset \Omega$ we see that

$$\begin{aligned}\sup_{\boldsymbol{v} \in \overline{B_\epsilon(0)}} \lim_{h \to 0} G_{X,\boldsymbol{f}}(\boldsymbol{x} + h\boldsymbol{v}) &= \sup_{\boldsymbol{v} \in \overline{B_\epsilon(0)}} \sum_{j=1}^{q} \left\langle f_j, \phi_{\boldsymbol{x}}^{\nabla v} \right\rangle_{\mathcal{H}}^2 \\ &\leq \sup_{\boldsymbol{v} \in \overline{B_\epsilon(0)}} \sum_{j=1}^{q} \|f_j\|_{\mathcal{H}}^2 \left\|\phi_{\boldsymbol{x}}^{\nabla v}\right\|_{\mathcal{H}}^2 = \|\boldsymbol{f}\|_{\mathcal{H}^q}^2 < \infty,\end{aligned}$$

which shows the boundedness. □

The most important conclusion from the above analysis for applications is, that the gain function $G_{X,\boldsymbol{f}}$ does not have any poles. This boundedness allows to actually obtain an analytic maximum in (2.24, p.58), presuming a suitable extension of $G_{X,\boldsymbol{f}}$ onto Ω. Furthermore, should it occur that the maximum of $G_{X,\boldsymbol{f}}$ is achieved at some $\boldsymbol{x} \in \Omega_X$ for a $\boldsymbol{v} \in \mathbb{R}^d$, it means that the extension of $\mathcal{H}^X$ with $\boldsymbol{v}^T \nabla_1 K(\boldsymbol{x}, \cdot)$ yields a better improvement than inclusion of *any* direct kernel translate $K(\boldsymbol{x}, \cdot)$. This initial work provides a direct connection for "hybrid" greedy algorithms that take extensions by derivatives into account and requires further investigation. However, in practical applications of this algorithm we yet consider only direct kernel translates for inclusion. Consequently, we extend $G_{X,\boldsymbol{f}}$ onto Ω by setting $G_{X,\boldsymbol{f}}(\boldsymbol{x}) := 0 \;\forall\, \boldsymbol{x} \in \Omega_X$. for the remainder of this work.

Remark 2.2.17. If in the context of Theorem 2.2.13, K is actually induced by a radial basis function ϕ via $K(\boldsymbol{x}, \boldsymbol{y}) = \phi(\|\boldsymbol{x} - \boldsymbol{y}\|)$, we directly obtain

$$\tilde{\phi}_{\boldsymbol{x}}^{\nabla v} = \phi'(\|\boldsymbol{x} - \cdot\|) \left\langle \boldsymbol{v}, \frac{\boldsymbol{x} - \cdot}{\|\boldsymbol{x} - \cdot\|} \right\rangle - \sum_{j=1}^{n} K(\boldsymbol{x}_j, \cdot) \sum_{i=1}^{n} K_{ij}^{-1} \phi'(\|\boldsymbol{x} - \boldsymbol{x}_i\|) \left\langle \boldsymbol{v}, \frac{\boldsymbol{x} - \boldsymbol{x}_i}{\|\boldsymbol{x} - \boldsymbol{x}_i\|} \right\rangle.$$

It is interesting to see that, for all directions that are free of any part towards the other centers (i.e. $\boldsymbol{v} \perp \boldsymbol{x} - \boldsymbol{x}_i$ for all i), their contribution to $\tilde{\phi}_{\boldsymbol{x}}^{\nabla v}$ vanishes completely.

Remark 2.2.18. If we assume $d = 1$ in the context of Proposition 2.2.16, then the limit of ϕ_x for $x \to \tilde{x} \in \Omega_X$ is unique and $G_{X,\boldsymbol{f}}$ is continuous on Ω $\forall\, f \in \mathcal{H}$.

2.2.2.2 Convergence analysis

In this Section we want to investigate the convergence behaviour of Algorithm 5, where we will prove a slightly improved convergence bound similar to the one established in [104] with a yet simplified proof. Note that this type of convergence rate stems from the more general theory of greedy algorithms in vectorial Hilbert/Banach spaces [107, 104].

We will need some auxiliary lemma before we can state our convergence results. The following Lemma was stated first in [168, 3.1] with proof in [40, Lemma 3.4], however, as the referred proof is for a slightly different case we state it here for completeness.

Lemma 2.2.19 (Lemma 3.1 from [168]). *Let $M > 0, t_m, a_m \geq 0$ be non-negative sequences satisfying $a_0 \leq M,\ a_{m+1} \leq a_m(1 - t_{m+1}\frac{a_m}{M})$. Then*

$$a_m \leq M \left(1 + \sum_{k=1}^{m} t_k\right)^{-1} \qquad \forall\, m \geq 0. \tag{2.31}$$

Proof. If we have $a_{m_0} = 0$ for an $m_0 \geq 0$ we have $a_m = 0\ \forall\, m \geq m_0$ and thus (2.31) holds trivially. So assume $a_m \neq 0$ for all $m \geq 0$ and we continue by induction. Then for $m = 0$ equation (2.31) is given by prerequisite. The induction step $m \to m+1$ follows via

$$\begin{aligned} a_{m+1}^{-1} &\geq a_m^{-1}\left(1 - t_{m+1}\frac{a_m}{M}\right)^{-1} \geq a_m^{-1}(1 + t_{m+1}\frac{a_m}{M}) \\ &= a_m^{-1} + \frac{t_{m+1}}{M} \geq \frac{1}{M}\left(1 + \sum_{k=1}^{m} t_k\right) + \frac{t_{m+1}}{M} \\ &= \frac{1}{M}\left(1 + \sum_{k=1}^{m+1} t_k\right), \end{aligned}$$

using the third binomial formula $(1-b)(1+b) = 1 - b^2 \leq 1$ and the prerequisites. □

The next Lemma is already known as Chebyshev's inequality [3, 3.2.7], but as it is used at crucial points within the convergence proof we shall also state it here for completeness.

Lemma 2.2.20 (Chebyshev's inequality). *Let* $a_m \in \mathbb{R}, m \in \mathbb{N}$ *be an arbitrary sequence. Then*

$$\left(\sum_{i=1}^{m} a_i\right)^2 \leq m \sum_{i=1}^{m} a_i^2 \;\forall\, m \in \mathbb{N}. \tag{2.32}$$

Proof. Case $m = 1$ holds trivially. So let (2.32) hold for $m \in \mathbb{N}$ arbitrary but fixed. By the third binomial formula or Young's inequality for products and exponent 2 we have $2ab \leq a^2 + b^2 \;\forall\, a, b \in \mathbb{R}$. Applying this m times gives

$$\begin{aligned}
\left(\sum_{i=1}^{m+1} a_i\right)^2 &= \sum_{i,j}^{m+1} a_i a_j = \sum_{i,j}^{m} a_i a_j + 2\sum_{i=1}^{m} a_i a_{m+1} + a_{m+1}^2 \\
&= \left(\sum_{i=1}^{m} a_i\right)^2 + 2\sum_{i=1}^{m} a_i a_{m+1} + a_{m+1}^2 \\
&\leq m\sum_{i=1}^{m} a_i^2 + \sum_{i=1}^{m} (a_i^2 + a_{m+1}^2) + a_{m+1}^2 \\
&= m\sum_{i=1}^{m} a_i^2 + \sum_{i=1}^{m} a_i^2 + (m+1) a_{m+1}^2 \\
&= (m+1)\left(\sum_{i=1}^{m} a_i^2 + a_{m+1}^2\right) = (m+1)\sum_{i=1}^{m+1} a_i^2.
\end{aligned}$$

□

A key aspect of the Temlyakov-type estimations is to consider a certain subclass of functions

$$\mathcal{H}_M^q := \left\{ \boldsymbol{f} \in \mathcal{H}^q \;\middle|\; f_j = \sum_{k=0}^{\infty} \alpha_k^j K(\boldsymbol{x}_k, \cdot), \sum_{k=0}^{\infty} |\alpha_k^j| \leq M, j = 1 \ldots q \right\}, \tag{2.33}$$

for $M > 0$. For more background on this methodology we refer to [40, §3].

Theorem 2.2.21 (Convergence rate of the f/P-VKOGA algorithm)**.** *Let the conditions of Algorithm 5 hold and let $M > 0$. Then for any $\boldsymbol{f} \in \mathcal{H}^q_M$, $\boldsymbol{f}^m$ converges to $\boldsymbol{f}$ no slower than*

$$||\boldsymbol{f} - \boldsymbol{f}^m||_{\mathcal{H}^q} \leq \sqrt{q}M\left(1+\frac{m}{q}\right)^{-\frac{1}{2}}, \quad m \geq 0. \tag{2.34}$$

Further, with the definition

$$c_m := \max_{\boldsymbol{x}\in\Omega} \tilde{\phi}^{m-1}_{\boldsymbol{x}}(\boldsymbol{x}) = \max_{\boldsymbol{x}\in\Omega} \left|\left|\tilde{\phi}^{m-1}_{\boldsymbol{x}}\right|\right|^2_{\mathcal{H}}, \quad m > 0,$$

we obtain the a-posteriori convergence bound

$$||\boldsymbol{f} - \boldsymbol{f}^m||_{\mathcal{H}^q} \leq \sqrt{q}M\left(1+\frac{1}{q}\sum_{k=1}^{m}\frac{1}{c_k}\right)^{-\frac{1}{2}}, \quad m \geq 0. \tag{2.35}$$

Proof. Since $c_m \leq 1 \,\forall\, m > 0$ by Lemma 2.2.5 we obtain the a-priori bound (2.34) from (2.35) by setting $c_m := 1 \,\forall\, m > 0$, which leaves us to prove (2.35). First, using Lemma 2.2.5 we see for $j = 1 \dots q$ that

$$\begin{aligned} \left|\left|f_j - f_j^{m-1}\right|\right|^2_{\mathcal{H}} &= \left\langle f_j - f_j^{m-1}, f_j - f_j^{m-1}\right\rangle_{\mathcal{H}} = \left\langle f_j, f_j - f_j^{m-1}\right\rangle_{\mathcal{H}} \\ &= \sum_{k=1}^{\infty} \alpha_k^j \left\langle K(\boldsymbol{x}_k,\cdot), f_j - f_j^{m-1}\right\rangle_{\mathcal{H}} \\ &\leq \sum_{k=1}^{\infty} |\alpha_k^j| \left|\left\langle f_j, \tilde{\phi}^{m-1}_{x_k}\right\rangle_{\mathcal{H}}\right| \\ &\leq \sum_{k=1}^{\infty} |\alpha_k^j| \max_{x\in\Omega}\left|\left\langle f_j, \tilde{\phi}^{m-1}_{\boldsymbol{x}}\right\rangle_{\mathcal{H}}\right| \\ &\leq M \max_{x\in\Omega}\left|\left\langle f_j, \tilde{\phi}^{m-1}_{\boldsymbol{x}}\right\rangle_{\mathcal{H}}\right|. \end{aligned} \tag{2.36}$$

Now, Lemma 2.2.2 gives

$$\begin{aligned} \left|\left|\tilde{\phi}_{\boldsymbol{x}}\right|\right|^2_{\mathcal{H}} &= ||K(\boldsymbol{x},\cdot) - \mathcal{P}_{\mathcal{H}^X}[K(\boldsymbol{x},\cdot)]||^2_{\mathcal{H}} \\ &= 1 - \langle K(\boldsymbol{x},\cdot), \mathcal{P}_{\mathcal{H}^X}[K(\boldsymbol{x},\cdot)]\rangle_{\mathcal{H}} \leq 1, \end{aligned}$$

and together with Lemma 2.2.20 (twice) we estimate the vectorial gain term as

$$
\begin{aligned}
G_{X_{m-1},\boldsymbol{f}}(\boldsymbol{x}_m) &= \max_{\boldsymbol{x}\in\Omega} G_{X_{m-1},\boldsymbol{f}}(\boldsymbol{x}) = \max_{\boldsymbol{x}\in\Omega\setminus\Omega_{X_{m-1}}} G_{X_{m-1},\boldsymbol{f}}(\boldsymbol{x}) \\
&= \max_{\boldsymbol{x}\in\Omega\setminus\Omega_{X_{m-1}}} \sum_{j=1}^{q} \frac{1}{\left|\left|\tilde{\phi}_{\boldsymbol{x}}^{m-1}\right|\right|_{\mathcal{H}}^2} \left\langle f_j, \tilde{\phi}_{\boldsymbol{x}}^{m-1} \right\rangle_{\mathcal{H}}^2 \\
&\geq \max_{\boldsymbol{x}\in\Omega\setminus\Omega_{X_{m-1}}} \frac{1}{c_m} \sum_{j=1}^{q} \left\langle f_j, \tilde{\phi}_{\boldsymbol{x}}^{m-1} \right\rangle_{\mathcal{H}}^2 \\
&\geq \frac{1}{c_m} \max_{\boldsymbol{x}\in\Omega\setminus\Omega_{X_{m-1}}} \max_{j=1\ldots q} \left\langle f_j, \tilde{\phi}_{\boldsymbol{x}}^{m-1} \right\rangle_{\mathcal{H}}^2 \\
&= \frac{1}{c_m} \max_{j=1\ldots q} \max_{\boldsymbol{x}\in\Omega\setminus\Omega_{X_{m-1}}} \left\langle f_j, \tilde{\phi}_{\boldsymbol{x}}^{m-1} \right\rangle_{\mathcal{H}}^2 \\
&= \frac{1}{\sqrt{q} c_m} \left(q \max_{j=1\ldots q} \max_{\boldsymbol{x}\in\Omega\setminus\Omega_{X_{m-1}}} \left\langle f_j, \tilde{\phi}_{\boldsymbol{x}}^{m-1} \right\rangle_{\mathcal{H}}^4 \right)^{\frac{1}{2}} \\
&\geq \frac{1}{\sqrt{q} c_m} \left(\sum_{j=1}^{q} \max_{\boldsymbol{x}\in\Omega\setminus\Omega_{X_{m-1}}} \left\langle f_j, \tilde{\phi}_{\boldsymbol{x}}^{m-1} \right\rangle_{\mathcal{H}}^4 \right)^{\frac{1}{2}} \\
&\geq \frac{1}{\sqrt{q} c_m} \left(\frac{1}{q} \left(\sum_{j=1}^{q} \max_{\boldsymbol{x}\in\Omega\setminus\Omega_{X_{m-1}}} \left\langle f_j, \tilde{\phi}_{\boldsymbol{x}}^{m-1} \right\rangle_{\mathcal{H}}^2 \right)^2 \right)^{\frac{1}{2}} \\
&= \frac{1}{q c_m} \sum_{j=1}^{q} \max_{\boldsymbol{x}\in\Omega\setminus\Omega_{X_{m-1}}} \left\langle f_j, \tilde{\phi}_{\boldsymbol{x}}^{m-1} \right\rangle_{\mathcal{H}}^2 .
\end{aligned}
$$

With (2.36) this means

$$
\begin{aligned}
G_{X_{m-1},\boldsymbol{f}}(\boldsymbol{x}_m) &\geq \frac{1}{q c_m} \sum_{j=1}^{q} \frac{\left|\left| f_j - f_j^{m-1} \right|\right|_{\mathcal{H}}^4}{M^2} \\
&\geq \frac{1}{q M^2 c_m} \frac{1}{q} \left(\sum_{j=1}^{q} \left|\left| f_j - f_j^{m-1} \right|\right|_{\mathcal{H}}^2 \right)^2 \\
&= \frac{1}{q^2 M^2 c_m} \left|\left| f - f^{m-1} \right|\right|_{\mathcal{H}^q}^4 .
\end{aligned}
$$

Using Lemma 2.2.10, we see that

$$\begin{aligned}||\boldsymbol{f}-\boldsymbol{f}^m||^2_{\mathcal{H}^q} &= \left|\left|\boldsymbol{f}-\boldsymbol{f}^{m-1}\right|\right|^2_{\mathcal{H}^q} - G_{X_{m-1},\boldsymbol{f}}(\boldsymbol{x}_m)\\ &\leq \left|\left|\boldsymbol{f}-\boldsymbol{f}^{m-1}\right|\right|^2_{\mathcal{H}^q} - \frac{1}{q^2M^2c_m}\left|\left|\boldsymbol{f}-\boldsymbol{f}^{m-1}\right|\right|^4_{\mathcal{H}^q}\\ &= \left|\left|\boldsymbol{f}-\boldsymbol{f}^{m-1}\right|\right|^2_{\mathcal{H}^q}\left(1-\frac{\frac{1}{qc_m}\left|\left|\boldsymbol{f}-\boldsymbol{f}^{m-1}\right|\right|^2_{\mathcal{H}^q}}{qM^2}\right).\end{aligned}$$

As $\boldsymbol{f} \in \mathcal{H}^q_M$, it is easy to see that $||f_j||_{\mathcal{H}} \leq M,\ j = 1\ldots q$. This gives

$$\left|\left|\boldsymbol{f}-\boldsymbol{f}^0\right|\right|^2_{\mathcal{H}^q} = ||\boldsymbol{f}||^2_{\mathcal{H}^q} = \sum_{j=1}^q ||f_j||^2_{\mathcal{H}} \leq \sum_{j=1}^q M^2 = qM^2.$$

Finally, applying Lemma 2.2.19 with $a_m = ||\boldsymbol{f}-\boldsymbol{f}^m||^2_{\mathcal{H}^q}$, $a_0 \leq qM^2$ and $t_m := \frac{1}{qc_m}$ gives

$$||\boldsymbol{f}-\boldsymbol{f}^m||^2_{\mathcal{H}^q} \leq qM^2\left(1+\sum_{k=1}^m \frac{1}{qc_k}\right)^{-1} \quad \forall\, m \in \mathbb{N},$$

and hence (2.35, p.70). □

Remark 2.2.22. In the context of Algorithm 4 we have monotonicity of $\mathcal{H}^q$ error decay and the same a-priori convergence rate as for Algorithm 5 can be shown to apply. Note here that the proof of convergence rates of the f-VKOGA-Algorithm has already been performed (in a more general setting) in [104], albeit using a different technique and obtaining a convergence rate which is a factor of $\sqrt{q}$ slower.

2.2.2.3 Computational aspects

As already mentioned in Remark 2.2.3, using the basis of direct kernel translates often yields ill-conditioned kernel matrices. Again, several approaches [146, 61, 60, 57] for various settings have been developed, where the "*Newton basis*" presented in in [129] proved to be a good choice in the context of greedy algorithms. The basis formulation integrates nicely into the greedy algorithms and allows computation on the fly without the need for the whole kernel matrix. Moreover, it turns out the Newton basis is very closely related to our concept of orthonormal remainders.

Definition 2.2.23 (Newton basis of $\mathcal{H}^X$)**.** Let $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \subseteq \Omega$ so that $\mathcal{H}^X$ is n-dimensional. Then the Newton basis $N_1, \dots, N_n$ of $\mathcal{H}^X$ is given by the recursion

$$N_1 := \frac{K(\boldsymbol{x}_1, \cdot)}{\sqrt{K(\boldsymbol{x}_1, \boldsymbol{x}_1)}}, \quad \tilde{N}_j = K(\boldsymbol{x}_j, \cdot) - \sum_{i=1}^{j-1} N_i(\boldsymbol{x}_i) N_i, \quad N_j = \frac{\tilde{N}_j}{\left\|\tilde{N}_j\right\|_{\mathcal{H}}},$$

for $j = 2 \dots n$ and satisfies

$$\langle N_i, N_j \rangle_{\mathcal{H}} = \delta_{ij}. \tag{2.37}$$

Condition (2.37) is easily verified by induction. Basically, the complete Newton basis is obtained by a Cholesky decomposition of the kernel matrix $\boldsymbol{K}$. As this is computed recursively and (2.37) holds, we do not have to touch any previously computed coefficients again. Moreover, we actually perform a pivoted partial Cholesky decomposition, where the next column is determined by the greedy selection criteria. Most noticeably, this allows for memory-efficient implementations as the whole of $\boldsymbol{K}$ is never needed. We refer to [129] for the complete background and continue with linking the Newton basis to our setting. Therefore, the next Lemma lists the Newton basis representations of the terms occurring in Algorithm 5, which especially shows the connection to the orthogonal remainders.

Lemma 2.2.24 (Newton basis representations)**.** *Let $N_1, \dots, N_m$ be the New-*

ton basis of $\mathcal{H}^X$ and $f \in \mathcal{H}$. Then

$$\mathcal{P}_{\mathcal{H}^X}[f] = \sum_{i=1}^{m} \langle f, N_i \rangle_{\mathcal{H}} N_i, \tag{2.38}$$

$$\mathcal{P}_{\mathcal{H}^X}[K(\boldsymbol{x}, \cdot)] = \sum_{i=1}^{m} N_i(\boldsymbol{x}) N_i, \tag{2.39}$$

$$\left|\left|\tilde{\phi}_{\boldsymbol{x}}\right|\right|_{\mathcal{H}}^2 = K(\boldsymbol{x}, \boldsymbol{x}) - \sum_{i=1}^{m} N_i^2(\boldsymbol{x}). \tag{2.40}$$

Proof. From the projection conditions (2.15, p.49) and (2.37) we immediately obtain (2.38). Equation (2.39) is a special case of (2.38) for $f = K(\boldsymbol{x}, \cdot)$.

Using (2.39) and Lemma 2.2.2 gives (2.40) via

$$\begin{aligned} \left|\left|\tilde{\phi}_{\boldsymbol{x}}\right|\right|_{\mathcal{H}}^2 &= K(\boldsymbol{x}, \boldsymbol{x}) - 2\mathcal{P}_{\mathcal{H}^X}[K(\boldsymbol{x}, \cdot)](\boldsymbol{x}) + ||\mathcal{P}_{\mathcal{H}^X}[K(\boldsymbol{x}, \cdot)]||_{\mathcal{H}}^2 \\ &= K(\boldsymbol{x}, \boldsymbol{x}) - \mathcal{P}_{\mathcal{H}^X}[K(\boldsymbol{x}, \cdot)](\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{x}) - \sum_{i=1}^{m} N_i^2(\boldsymbol{x}). \end{aligned}$$

□

Finally the following proposition states how the VKOGA can be computed efficiently using the Newton basis.

Proposition 2.2.25 (Computation of VKOGA with Newton basis). *For $m = 1$ set*

$$\begin{aligned} \boldsymbol{x}_1 &:= \arg\max_{\boldsymbol{x} \in \Omega} G_{\emptyset, \boldsymbol{f}}(\boldsymbol{x}) = \arg\max_{\boldsymbol{x} \in \Omega} \sum_{j=1}^{q} f_j(\boldsymbol{x})^2, \\ \boldsymbol{c}_1 &:= (\langle f_1, N_1 \rangle_{\mathcal{H}}, \dots, \langle f_q, N_1 \rangle_{\mathcal{H}})^T = \frac{(f_1(\boldsymbol{x}), \dots, f_q(\boldsymbol{x}))^T}{\sqrt{K(\boldsymbol{x}_1, \boldsymbol{x}_1)}} \in \mathbb{R}^q. \end{aligned}$$

Then, at the m-th iteration ($m > 1$) with given $\boldsymbol{x}_1, \dots, \boldsymbol{x}_{m-1}$, $\boldsymbol{c}_1, \dots, \boldsymbol{c}_{m-1}$

we define

$$\boldsymbol{x}_m := \arg \max_{\boldsymbol{x}\in\Omega\setminus\Omega_{X_{m-1}}} \left|\left|\boldsymbol{f}(\boldsymbol{x}) - \sum_{i=1}^{m-1} \boldsymbol{c}_i N_i(\boldsymbol{x})\right|\right|_2^2 \left(K(\boldsymbol{x},\boldsymbol{x}) - \sum_{i=1}^{m-1} N_i^2(\boldsymbol{x})\right)^{-1}, \tag{2.41}$$

$$\boldsymbol{c}_m := \begin{pmatrix} \langle f_1, N_m\rangle_{\mathcal{H}} \\ \vdots \\ \langle f_q, N_m\rangle_{\mathcal{H}} \end{pmatrix} = \frac{\boldsymbol{f}(\boldsymbol{x}_m) - \sum\limits_{i=1}^{m-1} \boldsymbol{c}_i N_i(\boldsymbol{x}_m)}{\left(K(\boldsymbol{x}_m,\boldsymbol{x}_m) - \sum\limits_{i=1}^{m-1} N_i^2(\boldsymbol{x}_m)\right)^{\frac{1}{2}}} \in \mathbb{R}^q. \tag{2.42}$$

Proof. In order to see (2.42) we note that

$$\left|\left|\tilde{N}_m\right|\right|_{\mathcal{H}}^2 = K(\boldsymbol{x}_m,\boldsymbol{x}_m) - \sum_{i=1}^{m-1} N_i^2(\boldsymbol{x}_m),$$

$$\langle f_j, N_m\rangle_{\mathcal{H}} = \frac{\left\langle f_j, \tilde{N}_m\right\rangle_{\mathcal{H}}}{\left|\left|\tilde{N}_m\right|\right|_{\mathcal{H}}} = \frac{f_j(\boldsymbol{x}_m) - \sum_{i=1}^{m-1} \langle f_j, N_i\rangle_{\mathcal{H}} N_i(\boldsymbol{x}_m)}{\left(K(\boldsymbol{x}_m,\boldsymbol{x}_m) - \sum\limits_{i=1}^{m-1} N_i^2(\boldsymbol{x}_m)\right)^{\frac{1}{2}}}.$$

With Lemma 2.2.24 we see (2.41) by

$$G_{X_{m-1},\boldsymbol{f}}(\boldsymbol{x}) = \frac{\sum\limits_{j=1}^{q} \left\langle f_j, \tilde{\phi}_{\boldsymbol{x}}^{m-1}\right\rangle_{\mathcal{H}}^2}{\left|\left|\tilde{\phi}_{\boldsymbol{x}}^{m-1}\right|\right|_{\mathcal{H}}^2} = \frac{\left|\left|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}^{m-1}(\boldsymbol{x})\right|\right|_2^2}{\left|\left|\tilde{\phi}_{\boldsymbol{x}}^{m-1}\right|\right|_{\mathcal{H}}^2} = \frac{\left|\left|\boldsymbol{f}(\boldsymbol{x}) - \sum_{i=1}^{m-1} \boldsymbol{c}_i N_i(\boldsymbol{x})\right|\right|_2^2}{K(\boldsymbol{x},\boldsymbol{x}) - \sum_{i=1}^{m-1} N_i^2(\boldsymbol{x})}.$$

□

A detailed pseudo-code implementation is provided by Algorithm 6. There, $\boldsymbol{N}.^2$ denotes a point-wise operation in the spirit of MatLab and the P_{flag} setting determines whether to use the f or f/P-VKOGA variants. This is also the reason why the residual error is computed separately (and of course different error measures can be considered as applicable). Note that the final approximant $\boldsymbol{f}^m$ will have the structure (2.38). In order to evaluate the expansion using the direct translate basis, the triangular matrix with values

Algorithm 6 : $[\boldsymbol{I}, \boldsymbol{c}, \boldsymbol{L}] = \text{VKOGA}(\boldsymbol{f}, X, P_{flag}, tol, M_{max})$

```
1: $m \leftarrow 1$    ▷ Initialize values
2: $\boldsymbol{z} \leftarrow \text{diag}(\boldsymbol{K}_X) = [K(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots K(\boldsymbol{x}_n, \boldsymbol{x}_n)] \in \mathbb{R}^n$    ▷ Diagonal of kernel matrix for (2.40)
3: $\boldsymbol{R} \leftarrow [\boldsymbol{f}(\boldsymbol{x}_1) \dots \boldsymbol{f}(\boldsymbol{x}_n)] \in \mathbb{R}^{q \times n}$ ▷ Initial residual: full function values Γ
4: $i_{max} \leftarrow \arg\max_{i=1\dots n} ||\boldsymbol{R}_i||_2^2$
5: $\boldsymbol{N} \leftarrow \boldsymbol{K}(:, i_{max})/\sqrt{z_{i_{max}}} \in \mathbb{R}^n$
6: $\boldsymbol{c}(:, m) \leftarrow \boldsymbol{R}_{i_{max}}/\sqrt{z_{i_{max}}}$
7: $\boldsymbol{p} \leftarrow \boldsymbol{N}.^2 \in \mathbb{R}^n$    ▷ Store the values of squared sums in (2.40) on $X$
8: $\boldsymbol{I} = [i_{max}]$
9: while $true$ do
10:     $\boldsymbol{R} \leftarrow \boldsymbol{R} - \boldsymbol{c}(:, m)^T \boldsymbol{N}(:, m)$    ▷ Update residual
11:     $v_{err} \leftarrow \max_{i=1\dots n} ||\boldsymbol{R}_i||_2^2$ ▷ Obtain maximum residual error and position
12:     if $v_{err} < tol \;||\; m = M_{max}$ then    ▷ Check stopping conditions
13:         break
14:     end if
15:     $m \leftarrow m + 1$
16:     if $P_{flag} = true$ then    ▷ Find new maximum error location
17:         $i_{max} \leftarrow \arg\max_{i=1\dots n} ||\boldsymbol{R}_i||_2^2 / (z_i - p_i)$    ▷ $f/P$-VKOGA variant
18:     else
19:         $i_{max} \leftarrow \arg\max_{i=1\dots n} ||\boldsymbol{R}_i||_2^2$    ▷ $f$-VKOGA variant
20:     end if
21:     $\boldsymbol{n} \leftarrow \boldsymbol{K}(:, i_{max}) - \boldsymbol{N}(:, 1{:}m)\boldsymbol{N}(i_{max}, 1{:}m)^T \in \mathbb{R}^n$ ▷ Compute new Newton basis on $X$
22:     $\boldsymbol{N}(:, m) \leftarrow \boldsymbol{n}/\sqrt{z_{i_{max}} - p_{i_{max}}}$ ▷ Normalize (via an identity formula)
23:     $\boldsymbol{c}(:, m) \leftarrow \boldsymbol{R}_{i_{max}}/\sqrt{z_{i_{max}} - p_{i_{max}}}$    ▷ Compute basis coefficients
24:     $\boldsymbol{p} \leftarrow \boldsymbol{p} + \boldsymbol{N}(:, m).^2$    ▷ Update basis squares (for norm)
25:     $\boldsymbol{I} \leftarrow [\boldsymbol{I}\; i_{max}] \in \mathbb{R}^m$
26: end while
27: $\boldsymbol{L} \leftarrow \boldsymbol{N}(\boldsymbol{I}, 1{:}m)$
```

of the computed m Newton basis functions at the selected points X_m can be used to obtain the corresponding coefficients [129].

Remark 2.2.26 (Connection to remainders). In the context of Proposition 2.2.25 we actually have $\tilde{\phi}_{x_m} = \tilde{N}_m$ and consequently $\phi_{x_m} = N_m$. This means that the orthonormal remainders in each step directly state all possible candidates of new Newton basis functions.

2.2.2.4 Experiments

Outside the scope of model reduction, we shortly want to pursue some numerical experiments to compare the VKOGA variants and discuss the a-posteriori bound. We use $d = q = 5$, the test domain $\Omega = [-5, 5]^d$ and a Gaussian kernel with $\gamma = 7.7866$, which is chosen so that $K(\boldsymbol{x}, \boldsymbol{y}) < 0.4 \; \forall \; ||\boldsymbol{x} - \boldsymbol{y}|| \geq \frac{1}{3}\text{diam}(\Omega)$, i.e. a certain locality of the kernel expansions is ensured. The test functions to approximate are of the structure

$$\boldsymbol{f}(\boldsymbol{x}) := \sum_{k=1}^{N} \boldsymbol{c}_k K(\boldsymbol{x}_k, \boldsymbol{x}) \in \mathcal{H}^q,$$

with $N = 20$ random centers within Ω and random expansion coefficients $\boldsymbol{c} \in [0, 15]^q$. Experiments showed that it does not make a considerable difference in performance if we used $\boldsymbol{f} \in \mathcal{H}^q$ (i.e. independent expansions for each dimension) or $\boldsymbol{f} \in \left(\mathcal{H}^X\right)^q$ (a common center set $X \subseteq \Omega$ for each component function), as either way the actual centers are generally not detected/chosen as centers by the greedy algorithms.

For training we use 3000 training points in Ω and we use a validation set of size 5000 from the same domain. The algorithm terminates if the expansion size exceeds $N = 300$. This is only set to ease comparability, normally one would impose maximum absolute or relative error stopping criteria over the training data. Figure 2.5 compares the maximum absolute and relative discrete L^2 errors over both training and validation data. Both variants show an exponential convergence rate which gets slower as N increases. On the training data, there is no significant difference in the performance of either variant. At least the f-VKOGA variant seems to expose a somewhat more continuous error decay and seems to overtake the f/P-VKOGA algorithm for higher N. Considering the validation set, the f/P-VKOGA algorithm yields a better approximation for smaller N values, which indicates an improved generalization potential. However, the key analytical difference is the selection criteria, which is why we plot the native norm error between full and approximated function in Figure 2.6. Here the native norm maxi-

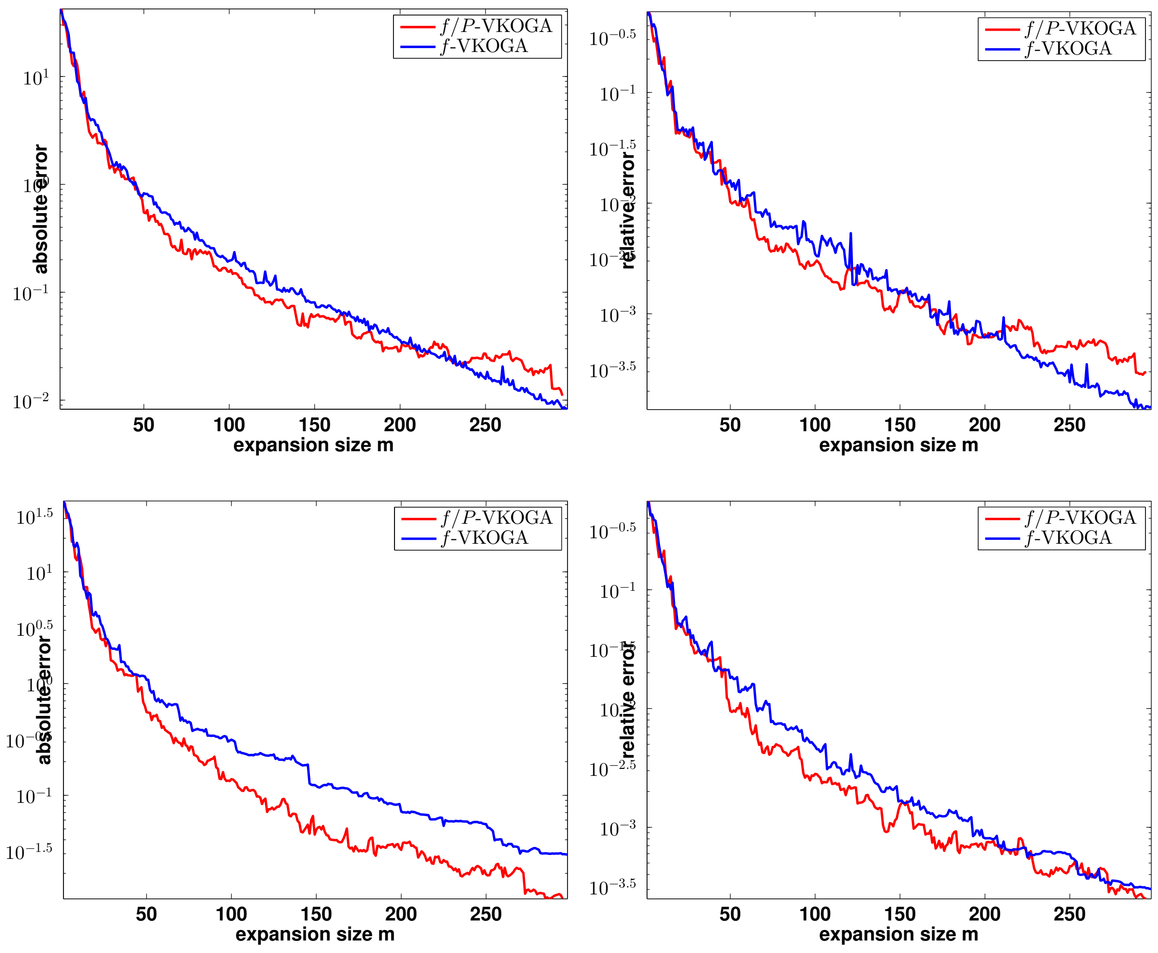


Figure 2.5: Top row: Absolute and relative errors on training data. Bottom row: Absolute and relative errors on validation data.

mizing selection criteria makes a clear difference of about an order of magnitude. We suspect that, if the true native norm $||\boldsymbol{f}||_{\mathcal{H}^q}$ is reached faster (recall $||\boldsymbol{f} - \boldsymbol{f}^m||^2_{\mathcal{H}^q} = ||\boldsymbol{f}||^2_{\mathcal{H}^q} - ||\boldsymbol{f}^m||^2_{\mathcal{H}^q}$ due to orthogonal projection), the function is approximated better *in general*, which explains the better performance on the validation set.

Figure 2.6 also contains the a-priori and f/P-VKOGA a-posteriori convergence bounds on the native norm. At first, an improvement regarding the a-posteriori bound compared to the a-priori bound is evident. Thus, if the involved function space is already approximated well during computations, an improvement of the a-priori bound is possible. However, not only are both bounds yet too conservative to be of practical use, but also requires knowledge of $\boldsymbol{f}$ in the sense of the $\mathcal{H}^q_M$ space (2.33, p.69) and norm. This is

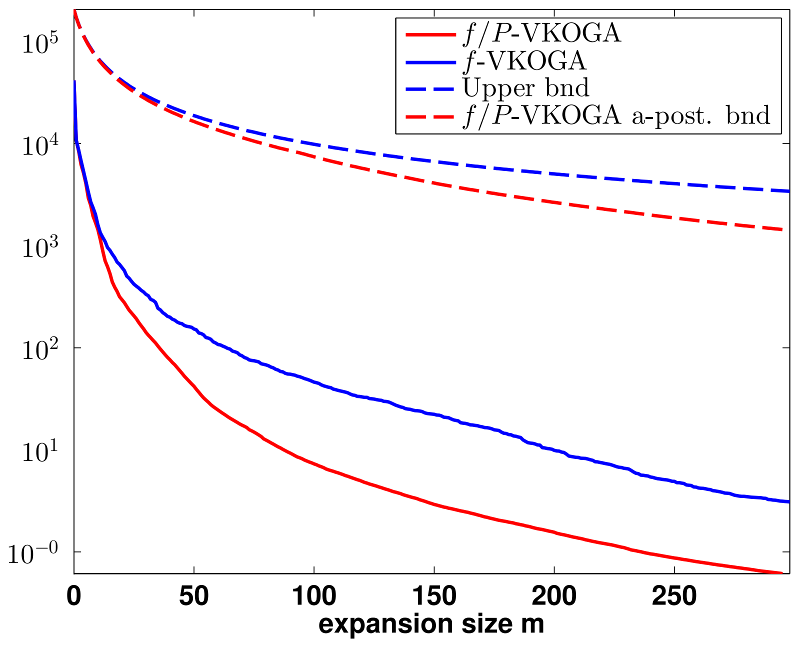


Figure 2.6: Error decay of $||\boldsymbol{f} - \boldsymbol{f}^m||_{\mathcal{H}^q}$ *for* f*- and* f/P*-VKOGA variants*

due to their theoretical background, as this order of convergence has been shown for quite general Hilbert spaces, cf. [107, 104]. To this end, the different perspectives on greedy algorithms regarding kernels have also led to different approaches regarding approximation error bounds. Other convergence results (in the scalar setting) usually involve the concept of a "*fill distance*", see [178, 117, 139] to name a few. Due to their generality, the foremost mentioned Temlyakov-style error bounds are often too conservative, while the fill distance-related bounds provide excellent convergence rates in many situations. However, the latter also suffer from the condition that a sufficiently small fill distance is hard to achieve in practice. Hence, it remains an open question if this "gap" can be closed in the future, as the practical convergence rates of the VKOGA (and kernel greedy algorithms in general) are usually orders of magnitude faster than the Temlyakov-bounds.

Remark 2.2.27. For higher dimensions the a-priori estimation assumptions can cause some problems. This is due to the following estimation in the first step of the convergence rate proof in Theorem 2.2.21:

$$\sum_{k=1}^{\infty} |\alpha_k^j| \left| \left\langle f_j, \tilde{\phi}_{\boldsymbol{x}_k}^{m-1} \right\rangle_{\mathcal{H}} \right| \leq \sum_{k=1}^{\infty} |\alpha_k^j| \max_{\boldsymbol{x} \in \Omega} \left| \left\langle f_j, \tilde{\phi}_{\boldsymbol{x}}^{m-1} \right\rangle_{\mathcal{H}} \right| .$$

This estimation holds in theory, but since Ω is replaced by a discrete training set $\hat{\Omega} \subseteq \Omega$ we might **not** have

$$\left| \left\langle f_j, \tilde{\phi}_{\boldsymbol{x}_k}^{m-1} \right\rangle_{\mathcal{H}} \right| \leq \max_{\boldsymbol{x} \in \hat{\Omega}} \left| \left\langle f_j, \tilde{\phi}_{\boldsymbol{x}}^{m-1} \right\rangle_{\mathcal{H}} \right| \; \forall \, \boldsymbol{x}_k.$$

For more details we refer to [183].

2.3 Discrete empirical interpolation

Another very successful method for MOR of nonlinear systems is the Discrete Empirical Interpolation Method (DEIM), which has been introduced in [23] and inherits its principal ideas from the Empirical Interpolation Method (EIM) [10]. The EIM has been developed in the context of parameter dependent partial differential equations (PDEs) for the case that some components cannot be decomposed in the parameter- or time-affine fashion introduced in Section 1.3.2, cf. [72]. Essentially, the EIM proposes to separate the state-space and parameter dependency of the considered components by using an affine linear combination of pre-selected functions (for fixed parameters) and suitable parameter-dependent weights. In the context of parameterized PDEs, this concept has been transferred to the discrete setting as "*Empirical Operator Interpolation*" [47, 75, 71]. Now, the DEIM is the EIM counterpart for (finite-dimensional) nonlinear dynamical systems, where the affine decomposition is pursued for the system's nonlinearity $\boldsymbol{f}$. As this method is essential for subsequent parts of this work, we shall briefly introduce it here.

The idea behind DEIM is to approximate the system's nonlinearity $\boldsymbol{f}$ by a linear combination of *DEIM basis vectors* $\mathcal{U} = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_M\} \subset \mathbb{R}^d$, where $M \leq d$ denotes the maximum *DEIM order*. The key for successful approximation is the combination of the basis functions. It is chosen in a way, that, given suitable "*magic points*" [111] $\mathcal{E} = \{\wp_1, \ldots, \wp_M\} \subseteq \{1, \ldots, d\}$, the approximation interpolates the original $\boldsymbol{f}$ on those points. Hence, for $m \leq M$ we denote by

$$\hat{\boldsymbol{f}}_m(\boldsymbol{x}) := \boldsymbol{U}_m(\boldsymbol{P}_m^T \boldsymbol{U}_m)^{-1} \boldsymbol{P}_m^T \boldsymbol{f}(\boldsymbol{x}) \tag{2.43}$$

the m-th order DEIM approximation of $\boldsymbol{f}$, where $\boldsymbol{P}_m := [\boldsymbol{e}_{\wp_1}, \ldots, \boldsymbol{e}_{\wp_m}]$, $\boldsymbol{U}_m := [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m]$ and $\boldsymbol{e}_i \in \mathbb{R}^d$ denotes the i-th unit vector in $\mathbb{R}^d$. The interpolation property on the m-DEIM points $\boldsymbol{e}_{\wp_i}$ can be easily verified by multiplying (2.43) from the left with $\boldsymbol{P}_m^T$. The assumption is that, given a suitably representable basis $\mathcal{U}$, the approximation will also reproduce the original $\boldsymbol{f}$ well on the remaining DoFs. Now, there are many choices on how to choose $\mathcal{U}$ and $\mathcal{E}$. In the original work [23], $\mathcal{U}$ is obtained from a POD of $\boldsymbol{f}$-evaluations on trajectory snapshots, or, in this context, the training values Y.

Next, Algorithm 7 describes the selection of $\mathcal{E}$ given $\mathcal{U}$. Note here that, if $\mathcal{U}$

Algorithm 7 : $\mathcal{E}$ = Magic Points($\mathcal{U}$)

```
1: $\wp_1 = \arg\max_i\{|(\boldsymbol{u}_1)_i|\}$   ▷ Initialize to maximum value index on $\boldsymbol{u}_1$
2: $\boldsymbol{U} = [\boldsymbol{u}_1],\ \boldsymbol{P} = [\boldsymbol{e}_{\wp_1}]$
3: for $l = 2 \dots M$ do
4:     $\boldsymbol{r} = \boldsymbol{u}_l - \boldsymbol{U}(\boldsymbol{P}^T\boldsymbol{U})^{-1}\boldsymbol{P}^T\boldsymbol{u}_l$   ▷ Residual of $\boldsymbol{u}_l$ for current order $l-1$
5:     $\wp_l = \arg\max_i\{|r_i|\}$   ▷ Select index with largest absolute value
6:     $\boldsymbol{U} \leftarrow [\boldsymbol{U}\ \boldsymbol{u}_l], \boldsymbol{P} \leftarrow [\boldsymbol{P}\ \boldsymbol{e}_{\wp_l}]$   ▷ Update $\boldsymbol{U}, \boldsymbol{P}$
7: end for
8: $\mathcal{E} = \{\wp_1, \dots, \wp_M\}$   ▷ Return interpolation points $\mathcal{E}$
```

is obtained by a POD, the vectors $\boldsymbol{u}_i$ are assumed to be sorted corresponding to their singular values, i.e. $\boldsymbol{u}_1$ is the first POD mode, etc.

Remark 2.3.1. Even though the computation of $\mathcal{U}$ via POD is a widely applied strategy, we would like to remark here that actually any set of linear independent vectors can serve as $\mathcal{U}$. Moreover, the set of vectors used to determine $\mathcal{E}$ can be completely independent of the DEIM basis $\mathcal{U}$ used later on. For the POD case, it is assumed that the largest values of the respective modes are most representable. This turns out to work really well in most situations, however, using orthonormalized samples of direct function/operator evaluations also proved to be a suitable choice [47].

Finally, there is one more assumption regarding the applicability of DEIM for arbitrary $\boldsymbol{f}$. In order to evaluate $\boldsymbol{f}(\boldsymbol{x})$, in general all components of $\boldsymbol{x}$ are required. This is not a problem analytically, but when it comes to model reduction, the central goal is to provide approximations that are *independent* of the full systems dimension d. Now, essentially DEIM assumes that the Jacobian pattern of $\boldsymbol{f}$ is sparse, so that in order to evaluate a component function of $\boldsymbol{f}$, only a few entries of $\boldsymbol{x}$ are necessary. This commonly is the case for dynamical systems derived from a PDE using a local spatial discretization scheme, for example. Using MatLab-style indexing, we can write this sparse dependency as

$$f_i(\boldsymbol{x}) = f_i(\boldsymbol{x}[\boldsymbol{j}_i]), \qquad i = 1 \ldots d,$$

where $\boldsymbol{j}_i = \{j_1^i, j_2^i, \ldots, j_{n_i}^i\} = \texttt{find}(\boldsymbol{J}_P[i,:])$ denotes the n_i component-indices of $\boldsymbol{x}$ required to evaluate f_i, i.e. the nonzero column indices of the i-th row of the Jacobian sparsity pattern $\boldsymbol{J}_P$ of $\boldsymbol{f}$. Hence, an m-th order DEIM approximation requires at most

$$\sum_{i=1}^{m} |\boldsymbol{j}_{\wp_i}| = \sum_{i=1}^{m} n_i$$

components of $\boldsymbol{x}$, and assuming $|\boldsymbol{j}_i| \ll d$, a considerable dimension reduction can be achieved. We will discuss this assumption in more detail in the model reduction Section 3.2.2 and will conclude this chapter on nonlinear approximation by introducing a modified DEIM variant.

Remark 2.3.2. Sometimes one global DEIM approximation of $\boldsymbol{f}$ is not sufficient, as either the desired speed (i.e. basis size) or accuracy cannot be achieved simultaneously. Hence, various methods like the parameter multi-domain "hp" empirical interpolation [51, 54] have been proposed. Further, an implicit partitioning method was introduced in [179] and a localized DEIM approach has been developed in [130]. All those methods target at increased accuracy at smaller basis size by providing multiple "local" DEIM approximations, where the locality is defined by different (geometrical or error) measures on the state or parameter domain.

2.3.1 Kernel DEIM

In the beginning of this chapter we claimed to only deal with sampling-based methods, which is, however, not entirely true for DEIM. As discussed at the end of the last section, point-wise evaluations of the nonlinearity/discrete operator are assumed to be available. Consequently, whenever one does not have access to implementation details (e.g. $\boldsymbol{f}$ is given as a black box or provided by compiled 3rd party code), the method cannot be applied.

In order to alleviate this problem and turn it into a truly sampling-based method, we propose to use a *Kernel DEIM*, which works the same way as DEIM but uses kernel approximations for the selection points or DoFs. The only requirement which is left is the possibility to obtain the Jacobian sparsity pattern $\boldsymbol{J}_P$, which is needed to determine which entries of $\boldsymbol{x}$ are effectively required to evaluate the component functions of $\boldsymbol{f}$.

The component functions $f_{\wp_i}$ are replaced by kernel approximations $\hat{f}_i : \mathbb{R}^{n_i} \to \mathbb{R}$

$$f_{\wp_i}(\boldsymbol{x}) \approx \hat{f}_i(\boldsymbol{x}) = \sum_{k=1}^{n} c_k^i K(\boldsymbol{w}_k^i, \boldsymbol{x}[\boldsymbol{j}_i]), \quad i = 1 \dots m,$$

with suitable kernel K and centers $\boldsymbol{w}_k^i = \boldsymbol{w}_{k_i}[\boldsymbol{j}_i] \in \mathbb{R}^{n_i}$. Consequently, given the Jacobian sparsity pattern $\boldsymbol{J}_P$, all selected m component functions $f_{\wp_i}$ can be replaced by kernel expansions $\hat{f}_i$, which have been trained using (the respective parts of) X and Y.

This methodology has the advantage of being able to treat $\boldsymbol{f}$ as a complete black-box regarding evaluation. Of course, the targeted learning algorithms for $f_{\wp_i}$ could be the SVR from Section 2.1, the VKOGA algorithm introduced in Section 2.2 or any other nonlinear (vectorial) approximation algorithm. Moreover, instead of learning m component functions $\hat{f}_i$, the vectorial variants can also be applied to directly learn a function $\hat{\boldsymbol{f}} : \mathbb{R}^p \to \mathbb{R}^m$ with $p = \sum_{i=1}^m n_i$. However, the advantage of component-wise learning is that the DEIM order m can be changed more easily without having to re-compute the vectorial $\hat{\boldsymbol{f}}$.

Chapter 3

Model reduction of nonlinear dynamical systems

In this chapter we shall introduce the methodology to create "*reduced order models*" (ROMs) of nonlinear parameterized dynamical systems. One well established approach to obtain ROMs is by means of *subspace projection*, where the full system is projected onto a suitable linear subspace of small dimension. The crucial assumption behind this is that the effective dynamics of the considered dynamical systems actually live on a lower dimensional manifold enclosed or approximated by such a linear subspace. This approach is applied to both "normal" and parameterized dynamical systems, where it is no surprise that, in the latter (more general) case, this subspace is naturally larger. However, as non-parameterized systems can be readily covered as special cases by all techniques applied to parameterized settings, we shall deal with parameterized systems in the following.

Additionally, in this thesis we assume that the generated subspaces suffice to cover the entire system behaviour at reasonable size for whatever chosen parameter domain $\mathcal{P}$. However, in practice, those assumptions can cause problems for more complex systems with e.g. strong variations w.r.t. the model parameters. Consequently, many approaches involving some sort of splitting of the spatial-, parameter- or time-domain into suitable regions have been devised, using local models on the respective parts. For linear parameterized systems, [106] propose to use weighted local models for suitable parameters with each an own projection subspace, which is done similarly in [5] but using an online nonlinear interpolation method for the reduced matrices. Similar schemes focusing on second order systems have been devised in e.g. [4, 126]. In the context of reduced basis (RB) methods [122, 127], both the "*hp-RB method*" [53] and the adaptive "*p-Partitioning*" approach [70] target at a suitable splitting of the parameter domain. A time-domain splitting is proposed for parameterized evolution equations in [44]. Such methods are in general also applicable for the MOR process in this thesis' context, but we refrain from their inclusion for the sake of simplicity.

Now, whilst projection techniques for linear of affine-linear systems are well understood by now [190, 7, 6, 11, 74], difficulties still arise regarding nonlinear systems. Hence, we will give the necessary background on subspace projection methods in Section 3.1 and deal with the implications for the nonlinear approximation methods (see Chapter 2) in Section 3.2 later.

3.1 Subspace projection

Recall here the linear dynamical system

$$\boldsymbol{x}'(t) = \boldsymbol{A}\boldsymbol{x}(t), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0, \qquad t \in [0, T]. \tag{3.1}$$

from Section 1.3 with state space variable $\boldsymbol{x}(t) \in \mathbb{R}^d$. Now the projection approach assumes the solution of (3.1) to essentially lie in a r-dimensional linear subspace $\mathcal{V} \subset \mathbb{R}^d$ spanned by the basis $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r\} \subset \mathbb{R}^d$. Hence, we set

$$\boldsymbol{x}(t) \approx \boldsymbol{x}^r(t) := \sum_{i=1}^{r} z_i(t)\boldsymbol{v}_i,$$

for yet unknown "*reduced basis*" coefficient functions $z_i(t), i = 1 \dots r$. For $\boldsymbol{V} = [\boldsymbol{v}_1 \, \dots \, \boldsymbol{v}_r] \in \mathbb{R}^{d\times r}$ this actually writes as $\boldsymbol{x}^r(t) = \boldsymbol{V}\boldsymbol{z}(t)$, and inserted into (3.1) one obtains

$$\boldsymbol{V}\boldsymbol{z}'(t) = \boldsymbol{A}\boldsymbol{V}\boldsymbol{z}(t), \qquad \boldsymbol{z}(0) = \boldsymbol{V}^T\boldsymbol{x}_0, \qquad t \in [0,T]. \tag{3.2}$$

However, while the initial condition $\boldsymbol{z}(0)$ can clearly be computed as the part of $\boldsymbol{x}(0)$ which lies in $\mathcal{V}$, the resulting ODE still has d equations for only r unknowns. Consequently, while $\boldsymbol{V}\boldsymbol{z}(t)$ forces the *solution* to lie in $\mathcal{V}$, we can choose another r-dimensional linear subspace $\mathcal{W} \subset \mathbb{R}^d$ with basis $\{\boldsymbol{w}_1, \dots, \boldsymbol{w}_r\}$ and restrict the *system dynamics* to $\mathcal{W}$, i.e. force the residual to be orthogonal to $\mathcal{W}$. We shall call $\mathcal{W}$ the *test space* as this concept is closely related to the commonly applied procedure to obtain weak formulations for PDEs. Similar to $\boldsymbol{z}(0)$, this is done by multiplying the system dynamics of (3.2) from the left with $\boldsymbol{W}^T$ for $\boldsymbol{W} = [\boldsymbol{w}_1, \dots, \boldsymbol{w}_r]$ to obtain the ROM or *reduced system*

$$\boldsymbol{W}^T\boldsymbol{V}\boldsymbol{z}'(t) = \boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\boldsymbol{z}(t), \qquad \boldsymbol{z}(0) = \boldsymbol{V}^T\boldsymbol{x}_0, \qquad t \in [0,T].$$

Note here that one could also use $\boldsymbol{z}(0) = \boldsymbol{W}^T\boldsymbol{x}(0)$ or $\boldsymbol{z}(0) = \boldsymbol{W}^T\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{x}(0)$ regarding initial conditions. The choice $\boldsymbol{z}(0) = \boldsymbol{V}^T\boldsymbol{x}_0$ follows a consistency argument as this choice ensures $\boldsymbol{V}\boldsymbol{z}(0) = \boldsymbol{x}_0$ if $\boldsymbol{x}_0 \in \mathcal{V}$. In applications, oftentimes $\boldsymbol{V} = \boldsymbol{W}$ is assumed, which is called the classical "*Galerkin*"-projection. If $\boldsymbol{V} \neq \boldsymbol{W}$, we speak of "*Petrov-Galerkin*" projection. However, in the latter case $\boldsymbol{V}$ and $\boldsymbol{W}$ are often assumed to be "*biorthogonal*", i.e. $\boldsymbol{W}^T\boldsymbol{V} = \boldsymbol{I}_r$, where $\boldsymbol{I}_r$ denotes the r-dimensional identity matrix.

The projection approach straightforwardly transfers to the more general case of nonlinear affine-parametric systems. As introduced in Section 1.3, (1.12a, p.19), recall

$$\boldsymbol{x}'(t) = \boldsymbol{f}(\boldsymbol{x}(t), t, \boldsymbol{\mu}) + \boldsymbol{A}(t,\boldsymbol{\mu})\boldsymbol{x}(t) + \boldsymbol{B}(t,\boldsymbol{\mu})\boldsymbol{u}(t), \tag{3.3a}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_0(\boldsymbol{\mu}), \tag{3.3b}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}(t,\boldsymbol{\mu})\boldsymbol{x}(t), \tag{3.3c}$$

stated here with output $\boldsymbol{y}$ for completeness. Similar to [74] regarding the affine components, we obtain for given biorthogonal $\boldsymbol{V}, \boldsymbol{W}$ the reduced system

$$\boldsymbol{z}'(t) = \boldsymbol{W}^T\boldsymbol{f}(\boldsymbol{V}\boldsymbol{z}(t),t,\boldsymbol{\mu}) + \sum_{i=1}^{Q_A}\theta_i^A(t,\boldsymbol{\mu})\tilde{\boldsymbol{A}}_i\boldsymbol{z}(t) + \sum_{i=1}^{Q_B}\theta_i^B(t,\boldsymbol{\mu})\tilde{\boldsymbol{B}}_i\boldsymbol{u}(t), \tag{3.4a}$$

$$\boldsymbol{z}(0) = \sum_{i=1}^{Q_0}\theta_i^0(\boldsymbol{\mu})\tilde{\boldsymbol{x}}_i^0 =: \boldsymbol{z}_0(\boldsymbol{\mu}), \tag{3.4b}$$

$$\boldsymbol{w}^r(t) = \sum_{i=1}^{Q_C}\theta_i^C(t,\boldsymbol{\mu})\tilde{\boldsymbol{C}}_i\boldsymbol{z}(t). \tag{3.4c}$$

with

$$\tilde{\boldsymbol{A}}_i := \boldsymbol{W}^T\boldsymbol{A}_i\boldsymbol{V}, \quad \tilde{\boldsymbol{B}}_i := \boldsymbol{W}^T\boldsymbol{B}_i, \quad \tilde{\boldsymbol{C}}_i := \boldsymbol{C}_i\boldsymbol{V}, \quad \tilde{\boldsymbol{x}}_i^0 := \boldsymbol{V}^T\boldsymbol{x}_i^0,$$

where i runs over the respective indices. This reduced system consists only of components, whose evaluation complexity is *independent* of d, the full state space dimension. Actually, without any further assumptions on $\boldsymbol{f}$, this is generally not true for $\boldsymbol{W}^T\boldsymbol{f}(\boldsymbol{V}\boldsymbol{z}(t))$, as it still involves d-dimensional quantities. Thus, most projection schemes involving nonlinearities enforce additional structure on $\boldsymbol{f}$, some of which we will discuss in Section 3.2 later.

Finally, various techniques to obtain reduced bases $\boldsymbol{V}, \boldsymbol{W}$ have been investigated, considering different classes of linear systems like time invariant [11], time-variant [143] or parameterized [74]. Various extensions of the balanced truncation procedure [118] to nonlinear systems are investigated in [151, 100, 31]. In the following we detail two by now well established methods for subspace computation and refer the reader to [7, 6, 190] for an overview on more available approaches. Note here that both subsequently introduced methods are mostly used in the context of Galerkin-projection, where $\boldsymbol{V} = \boldsymbol{W}$.

3.1.1 Proper orthogonal decomposition

The method known as "*Proper Orthogonal Decomposition*" (POD) aims at finding a "maximally representative" set of vectors for a given collection of vectors. This method is also known as "*Principal Component Analysis*" [86] or "*Karhunen-Loève*" transformation [92, 105] in literature and was originally introduced as Hotelling-Transformation in [81]. Extensions thereof have been considered for different specialized applications, e.g. the "*OS-POD*" [98]. The POD has since been used in a variety of applications comprising control [108], fluid dynamics [110, 141] or inverse problems [97], see also [118, 155, 170, 6, 79, 23]. We shall derive the computational background here as this method is frequently applied also in our context.

Mathematically, for $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$ and $k \leq n$, the POD computes a set of k orthonormal vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k \in \mathbb{R}^d$ that span an "optimal" subspace for X in the sense of

$$\min_{\boldsymbol{V} \in \mathbb{R}^{d \times k}} \sum_{i=1}^{n} \left|\left| \left(\boldsymbol{I}_d - \boldsymbol{V}\boldsymbol{V}^T\right) \boldsymbol{x}_i \right|\right|_2^2 \quad \text{s.t.} \quad \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}_k.$$

Lemma 3.1.1 (Optimal 1D subspace). *Let $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$ and define $\boldsymbol{X} := [\boldsymbol{x}_1 \ldots \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$. Then the solution of the minimization problem*

$$\boldsymbol{v}^* = \arg\min_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{i=1}^{n} \left|\left| \boldsymbol{x}_i - \langle \boldsymbol{v}, \boldsymbol{x}_i \rangle \boldsymbol{v} \right|\right|_2^2, \qquad \textit{s.t.} \quad ||\boldsymbol{v}|| = 1,$$

is given by $[\boldsymbol{v}^*, \lambda] = \max \sigma(\boldsymbol{X}\boldsymbol{X}^T)$, the eigenvector $\boldsymbol{v}^*$ associated with the largest eigenvalue λ of the scaled correlation matrix $\boldsymbol{X}\boldsymbol{X}^T$.

Proof. Consider

$$\begin{aligned}\sum_{i=1}^{n} ||\boldsymbol{x}_i - \langle \boldsymbol{v}, \boldsymbol{x}_i \rangle \boldsymbol{v}||_2^2 &= \sum_{i=1}^{n} \left(||\boldsymbol{x}_i||^2 - 2 \langle \boldsymbol{x}_i, \boldsymbol{v} \langle \boldsymbol{v}, \boldsymbol{x}_i \rangle \rangle + \langle \boldsymbol{v}, \boldsymbol{x}_i \rangle^2 ||\boldsymbol{v}||^2 \right) \\ &= C - \sum_{i=1}^{n} \langle \boldsymbol{v}, \boldsymbol{x}_i \rangle^2 = C - \left\langle \boldsymbol{v}, \sum_{i=1}^{n} \boldsymbol{x}_i \langle \boldsymbol{v}, \boldsymbol{x}_i \rangle \right\rangle \\ &= C - \left\langle \boldsymbol{v}, \boldsymbol{X}\boldsymbol{X}^T \boldsymbol{v} \right\rangle,\end{aligned}$$

for $||\boldsymbol{v}|| = 1$. Hence, it is equivalent to maximize $\left\langle \boldsymbol{v}, \boldsymbol{X}\boldsymbol{X}^T \boldsymbol{v} \right\rangle$ and we obtain a Lagrangian

$$\mathcal{L}(\boldsymbol{v}, \lambda) = \left\langle \boldsymbol{v}, \boldsymbol{X}\boldsymbol{X}^T \boldsymbol{v} \right\rangle - \lambda(1 - ||\boldsymbol{v}||^2).$$

The first order optimality conditions require

$$0 \overset{!}{=} \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}}(\boldsymbol{v}, \lambda) = 2(\boldsymbol{X}\boldsymbol{X}^T \boldsymbol{v} - \lambda \boldsymbol{v}),$$

hence $\boldsymbol{v}$ must be an eigenvector of $\boldsymbol{X}\boldsymbol{X}^T$. Now, for any such eigenvector we have

$$\left\langle \boldsymbol{v}, \boldsymbol{X}\boldsymbol{X}^T \boldsymbol{v} \right\rangle = \lambda ||\boldsymbol{v}||^2 = \lambda,$$

which directly yields the choice $[\boldsymbol{v}^*, \lambda] = \max \sigma(\boldsymbol{X}\boldsymbol{X}^T)$ and finishes the proof. □

Theorem 3.1.2 (POD). *Let the conditions of Lemma 3.1.1 hold. Further let* $\boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T$ *denote the eigenvalue decomposition of* $\boldsymbol{X}\boldsymbol{X}^T$*, where* $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_d) \in \mathbb{R}^{d \times d}$ *contains the sorted eigenvalues* $\lambda_1 \geq \lambda_2 \geq \ldots$ *Then for any* $1 \leq k \leq d$ *the solution to*

$$\arg \min_{\substack{\boldsymbol{V} \in \mathbb{R}^{d \times k} \\ \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}_k}} \sum_{i=1}^{n} \left|\left| \left(\boldsymbol{I}_n - \boldsymbol{V}\boldsymbol{V}^T \right) \boldsymbol{x}_i \right|\right|_2^2, \tag{3.5}$$

is given by $\boldsymbol{V}_k := \boldsymbol{V}[:, 1{:}k]$*, the first* k *columns of* $\boldsymbol{V}$*, which are also referred to as the first* k *"*POD-modes*".*

Proof. We prove by induction. For $k = 1$, application of Lemma 3.1.1 directly yields $\boldsymbol{v}_1 (= \boldsymbol{V}_1)$. Now assume to have $\boldsymbol{V}_k$ satisfying (3.5) for $k < d$. Then for any $\tilde{\boldsymbol{V}} = [\tilde{\boldsymbol{v}}_1 \dots \tilde{\boldsymbol{v}}_{k+1}] \in \mathbb{R}^{d \times (k+1)}$ we have

$$\begin{aligned}
\sum_{i=1}^{n} \left\langle \boldsymbol{x}_i, \tilde{\boldsymbol{V}}\tilde{\boldsymbol{V}}^T \boldsymbol{x}_i \right\rangle &= \sum_{i=1}^{n} \left\langle \boldsymbol{x}_i, \sum_{j=1}^{k+1} \tilde{\boldsymbol{v}}_j \left\langle \tilde{\boldsymbol{v}}_j, \boldsymbol{x}_i \right\rangle \right\rangle = \sum_{i=1}^{n} \sum_{j=1}^{k+1} \left\langle \tilde{\boldsymbol{v}}_j, \boldsymbol{x}_i \right\rangle^2 \\
&= \sum_{j=1}^{k+1} \left\langle \tilde{\boldsymbol{v}}_j, \sum_{i=1}^{n} \boldsymbol{x}_i \left\langle \boldsymbol{x}_i, \tilde{\boldsymbol{v}}_j \right\rangle \right\rangle = \sum_{j=1}^{k+1} \left\langle \tilde{\boldsymbol{v}}_j, \boldsymbol{X}\boldsymbol{X}^T \tilde{\boldsymbol{v}}_j \right\rangle \\
&= \sum_{j=1}^{k} \left\langle \tilde{\boldsymbol{v}}_j, \boldsymbol{X}\boldsymbol{X}^T \tilde{\boldsymbol{v}}_j \right\rangle + \left\langle \tilde{\boldsymbol{v}}_{k+1}, \boldsymbol{X}\boldsymbol{X}^T \tilde{\boldsymbol{v}}_{k+1} \right\rangle \\
&= \sum_{i=1}^{n} \left\langle \boldsymbol{x}_i, \tilde{\boldsymbol{V}}[:, 1{:}k](\tilde{\boldsymbol{V}}[:, 1{:}k])^T \boldsymbol{x}_i \right\rangle \\
&\quad + \left\langle \tilde{\boldsymbol{v}}_{k+1}, \boldsymbol{X}\boldsymbol{X}^T \tilde{\boldsymbol{v}}_{k+1} \right\rangle .
\end{aligned}$$

Hence we can write

$$\begin{aligned}
& \sum_{i=1}^{n} \left|\left| \left(\boldsymbol{I}_n - \tilde{\boldsymbol{V}}\tilde{\boldsymbol{V}}^T \right) \boldsymbol{x}_i \right|\right|_2^2 \\
= \;& \left(\sum_{i=1}^{n} ||\boldsymbol{x}_i||^2 - \left\langle \boldsymbol{x}_i, \tilde{\boldsymbol{V}}\tilde{\boldsymbol{V}}^T \boldsymbol{x}_i \right\rangle \right) \\
= \;& \left(\sum_{i=1}^{n} ||\boldsymbol{x}_i||^2 - \left\langle \boldsymbol{x}_i, \tilde{\boldsymbol{V}}[:, 1{:}k](\tilde{\boldsymbol{V}}[:, 1{:}k])^T \boldsymbol{x}_i \right\rangle - \left\langle \tilde{\boldsymbol{v}}_{k+1}, \boldsymbol{X}\boldsymbol{X}^T \tilde{\boldsymbol{v}}_{k+1} \right\rangle \right) \\
= \;& \left(\left|\left| \left(\boldsymbol{I}_n - \tilde{\boldsymbol{V}}[:, 1{:}k](\tilde{\boldsymbol{V}}[:, 1{:}k])^T \right) \boldsymbol{x}_i \right|\right|_2^2 - \left\langle \tilde{\boldsymbol{v}}_{k+1}, \boldsymbol{X}\boldsymbol{X}^T \tilde{\boldsymbol{v}}_{k+1} \right\rangle \right) .
\end{aligned}$$

Using the induction assumption this yields

$$\begin{aligned}
& \min_{\substack{\tilde{\boldsymbol{V}} \in \mathbb{R}^{d \times k+1} \\ \tilde{\boldsymbol{V}}^T \tilde{\boldsymbol{V}} = \boldsymbol{I}_{k+1}}} \sum_{i=1}^{n} \left|\left| \left(\boldsymbol{I}_n - \tilde{\boldsymbol{V}}\tilde{\boldsymbol{V}}^T \right) \boldsymbol{x}_i \right|\right|_2^2 \\
= \;& \min_{\substack{\tilde{\boldsymbol{V}} \in \mathbb{R}^{d \times k+1} \\ \tilde{\boldsymbol{V}}^T \tilde{\boldsymbol{V}} = \boldsymbol{I}_{k+1}}} \left(\left|\left| \left(\boldsymbol{I}_n - \tilde{\boldsymbol{V}}[:, 1{:}k](\tilde{\boldsymbol{V}}[:, 1{:}k])^T \right) \boldsymbol{x}_i \right|\right|_2^2 - \left\langle \tilde{\boldsymbol{v}}_{k+1}, \boldsymbol{X}\boldsymbol{X}^T \tilde{\boldsymbol{v}}_{k+1} \right\rangle \right)
\end{aligned}$$

$$= \min_{\substack{\hat{\boldsymbol{V}} \in \mathbb{R}^{d\times k}\\ \hat{\boldsymbol{V}}^T\hat{\boldsymbol{V}}=\boldsymbol{I}_k}} \left|\left|\left(\boldsymbol{I}_n - \hat{\boldsymbol{V}}\hat{\boldsymbol{V}}^T\right)\boldsymbol{x}_i\right|\right|_2^2 - \max_{\substack{\tilde{\boldsymbol{v}}_{k+1}\in\mathbb{R}^d\\ ||\tilde{\boldsymbol{v}}_{k+1}||=1\\ \hat{\boldsymbol{V}}^T\tilde{\boldsymbol{v}}_{k+1}=\boldsymbol{0}}} \left\langle \tilde{\boldsymbol{v}}_{k+1}, \boldsymbol{X}\boldsymbol{X}^T\tilde{\boldsymbol{v}}_{k+1}\right\rangle$$

$$= \left|\left|\left(\boldsymbol{I}_n - \boldsymbol{V}_k\boldsymbol{V}_k^T\right)\boldsymbol{x}_i\right|\right|_2^2 - \max_{\substack{\tilde{\boldsymbol{v}}_{k+1}\in\mathbb{R}^d\\ ||\tilde{\boldsymbol{v}}_{k+1}||=1\\ \boldsymbol{V}_k^T\tilde{\boldsymbol{v}}_{k+1}=\boldsymbol{0}}} \left\langle \tilde{\boldsymbol{v}}_{k+1}, \boldsymbol{X}\boldsymbol{X}^T\tilde{\boldsymbol{v}}_{k+1}\right\rangle,$$

With the definition $\boldsymbol{X}_k := \boldsymbol{X} - \boldsymbol{V}_k\boldsymbol{V}_k^T\boldsymbol{X}$ it is straightforward to see that $\boldsymbol{X}_k\boldsymbol{X}_k^T = \boldsymbol{X}\boldsymbol{X}^T - \boldsymbol{V}_k\boldsymbol{\Lambda}_k\boldsymbol{V}_k^T$. Moreover, with $\boldsymbol{\Lambda}_k := \boldsymbol{\Lambda}[1{:}k, 1{:}k]$ we see that

$$\boldsymbol{X}_k\boldsymbol{X}_k^T\boldsymbol{V}_k = \boldsymbol{X}\boldsymbol{X}^T\boldsymbol{V}_k - \boldsymbol{V}_k\boldsymbol{\Lambda}_k\boldsymbol{V}_k^T\boldsymbol{V}_k = \boldsymbol{V}_k\boldsymbol{\Lambda}_k - \boldsymbol{V}_k\boldsymbol{\Lambda}_k = \boldsymbol{0},$$

i.e. the eigenvectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_k$ are in the kernel of $\boldsymbol{X}_k\boldsymbol{X}_k^T$. Together this shows

$$\begin{aligned}
\max_{\substack{\tilde{\boldsymbol{v}}_{k+1}\in\mathbb{R}^d\\ ||\tilde{\boldsymbol{v}}_{k+1}||=1\\ \boldsymbol{V}_k^T\tilde{\boldsymbol{v}}_{k+1}=\boldsymbol{0}}} \left\langle \tilde{\boldsymbol{v}}_{k+1}, \boldsymbol{X}\boldsymbol{X}^T\tilde{\boldsymbol{v}}_{k+1}\right\rangle &= \max_{\substack{\tilde{\boldsymbol{v}}_{k+1}\in\mathbb{R}^d\\ ||\tilde{\boldsymbol{v}}_{k+1}||=1\\ \boldsymbol{V}_k^T\tilde{\boldsymbol{v}}_{k+1}=\boldsymbol{0}}} \left\langle \tilde{\boldsymbol{v}}_{k+1}, (\boldsymbol{X}_k\boldsymbol{X}_k^T + \boldsymbol{V}_k\boldsymbol{\Lambda}_k\boldsymbol{V}_k^T)\tilde{\boldsymbol{v}}_{k+1}\right\rangle \\
&= \max_{\substack{\tilde{\boldsymbol{v}}_{k+1}\in\mathbb{R}^d\\ ||\tilde{\boldsymbol{v}}_{k+1}||=1\\ \boldsymbol{V}_k^T\tilde{\boldsymbol{v}}_{k+1}=\boldsymbol{0}}} \left\langle \tilde{\boldsymbol{v}}_{k+1}, \boldsymbol{X}_k\boldsymbol{X}_k^T\tilde{\boldsymbol{v}}_{k+1}\right\rangle \\
&= \max_{\substack{\tilde{\boldsymbol{v}}_{k+1}\in\mathbb{R}^d\\ ||\tilde{\boldsymbol{v}}_{k+1}||=1}} \left\langle \tilde{\boldsymbol{v}}_{k+1}, \boldsymbol{X}_k\boldsymbol{X}_k^T\tilde{\boldsymbol{v}}_{k+1}\right\rangle. \qquad (3.6)
\end{aligned}$$

As in Lemma 3.1.1, (3.6) is maximized for $[\tilde{\boldsymbol{v}}, \tilde{\lambda}] = \max\sigma\left(\boldsymbol{X}_k\boldsymbol{X}_k^T\right)$. Now, again using $\boldsymbol{V}_k^T\tilde{\boldsymbol{v}} = 0$ we see that

$$\tilde{\lambda}\tilde{\boldsymbol{v}} = \boldsymbol{X}_k(\boldsymbol{X}_k)^T\tilde{\boldsymbol{v}} = (\boldsymbol{X}\boldsymbol{X}^T - \boldsymbol{V}_k\boldsymbol{\Lambda}_k\boldsymbol{V}_k^T)\tilde{\boldsymbol{v}} = \boldsymbol{X}\boldsymbol{X}^T\tilde{\boldsymbol{v}}, \qquad (3.7)$$

so $(\tilde{\boldsymbol{v}}, \tilde{\lambda})$ is eigenpair of $\boldsymbol{X}\boldsymbol{X}^T$, too. Since (3.7) holds for any eigenpair of $\boldsymbol{X}_k\boldsymbol{X}_k^T$, we must have $\tilde{\lambda} = \lambda_{k+1}$ by maximal choice and hence $\boldsymbol{v}_{k+1} := \tilde{\boldsymbol{v}}$ and $\boldsymbol{V}_{k+1} := [\boldsymbol{V}_k\ \boldsymbol{v}_{k+1}]$. $\square$

As can be seen in Theorem 3.1.2, in order to compute the L^2-optimal approximating subspace for a set X, an eigenvalue problem for $\boldsymbol{X}\boldsymbol{X}^T$ needs to be solved, at least up to the first largest k eigenvalues. However, if $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ with $d \gg n$ (i.e. if there are less training vectors than sample dimension), we deal with a large matrix $\boldsymbol{X}\boldsymbol{X}^T \in \mathbb{R}^{d\times d}$ that has at most n nonzero eigenvalues, i.e. we have a maximum of n POD-modes. This is computationally infeasible, and the following corollary provides a more efficient way of computation in these situations.

Corollary 3.1.3. *Let $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ and $\boldsymbol{X} = [\boldsymbol{x}_1 \dots \boldsymbol{x}_n] \in \mathbb{R}^{d\times n}$ with $d > n$. Further let*

$$\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^T$$

be the eigen-decomposition of $\boldsymbol{X}^T\boldsymbol{X} \in \mathbb{R}^{n\times n}$ such that the diagonal of $\boldsymbol{\Lambda}$ contains the eigenvalues sorted in descending order. Then for

$$\boldsymbol{V} := \boldsymbol{X}\boldsymbol{W}\boldsymbol{\Lambda}^{-\frac{1}{2}} \in \mathbb{R}^{d\times n},$$

the k-th order POD of X is given by $\boldsymbol{V}_k = \boldsymbol{V}[:, 1{:}k]$ for any $k \leq n$.

Proof. At first we notice that $\boldsymbol{X}^T\boldsymbol{X}$ is positive semi-definite, which grants that $\boldsymbol{\Lambda}^{-\frac{1}{2}}$ is well-defined (leave zero for zero eigenvalues, e.g. use pseudo-inverse $(\boldsymbol{\Lambda}^+)^{\frac{1}{2}}$). Then we see that

$$\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T = \boldsymbol{X}\boldsymbol{W}\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{W}^T\boldsymbol{X}^T = \boldsymbol{X}\boldsymbol{X}^T,$$

which yields an "economic" eigen-decomposition of $\boldsymbol{X}\boldsymbol{X}^T$ as

$$\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Lambda},$$

but yet $\boldsymbol{V} \in \mathbb{R}^{d\times n}$. Hence, application of Theorem 3.1.2 finishes the proof. □

Consequently, depending on the situation it is more efficient to compute the eigenvalues of either $\boldsymbol{X}\boldsymbol{X}^T$ or $\boldsymbol{X}^T\boldsymbol{X}$, as there are at most $\min\{d, n\}$ nonzero eigenvalues anyways. We would like to conclude this section with introducing the "*singular value decomposition*" (SVD) for arbitrary matrices and show the relation to POD.

Definition 3.1.4 ("Economic" singular value decomposition)**.** For any matrix $\boldsymbol{X} \in \mathbb{R}^{d\times n}$, we denote by

$$\boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{W}^T$$

a SVD of $\boldsymbol{X}$, where for $k = \min\{d, n\}$ we have $\boldsymbol{V} \in \mathbb{R}^{d\times k}, \boldsymbol{W} \in \mathbb{R}^{k\times n}$ with pairwise orthonormal columns and a diagonal matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{k\times k}$ containing the "*singular values*" $\sigma_1 \geq \ldots \geq \sigma_k \geq 0$ on the diagonal.

Corollary 3.1.5 (Link of POD to SVD)**.** *Let* $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ *be arbitrary and denote by*

$$\boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{V}_d\boldsymbol{\Lambda}_d\boldsymbol{V}_d^T, \qquad \boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{W}_n\boldsymbol{\Lambda}_n\boldsymbol{W}_n^T,$$

the eigen-decompositions of the respective matrices. Further let

$$\boldsymbol{\Sigma}_d = \boldsymbol{\Lambda}_d^{\frac{1}{2}}, \quad \boldsymbol{W}_d = \boldsymbol{X}^T\boldsymbol{V}_d\boldsymbol{\Sigma}_d^{-1}, \quad \boldsymbol{\Sigma}_n = \boldsymbol{\Lambda}_n^{\frac{1}{2}}, \quad \boldsymbol{V}_n = \boldsymbol{X}\boldsymbol{W}_n\boldsymbol{\Sigma}_n^{-1}.$$

Then we have

$$\boldsymbol{V}_d\boldsymbol{\Sigma}_d\boldsymbol{W}_d^T = \boldsymbol{X} = \boldsymbol{V}_n\boldsymbol{\Sigma}_n\boldsymbol{W}_n^T, \tag{3.8}$$

and for $k = \min\{d, n\}$ *we have* $\boldsymbol{\Sigma}_d[1{:}k, 1{:}k] = \boldsymbol{\Sigma}_n[1{:}k, 1{:}k]$.

Proof. As before, $\boldsymbol{X}\boldsymbol{X}^T$ and $\boldsymbol{X}^T\boldsymbol{X}$ are positive semi-definite and hence $\boldsymbol{\Sigma}_d, \boldsymbol{\Sigma}_n$ are well-defined. Direct use of the definitions yields (3.8) via

$$\boldsymbol{V}_d\boldsymbol{\Sigma}_d\boldsymbol{W}_d^T = \boldsymbol{V}_d\boldsymbol{\Sigma}_d\boldsymbol{\Sigma}_d^{-T}\boldsymbol{V}_d^T\boldsymbol{X} = \boldsymbol{X} = \boldsymbol{X}\boldsymbol{W}_n\boldsymbol{\Sigma}_n^{-1}\boldsymbol{\Sigma}_n\boldsymbol{W}_n^T = \boldsymbol{V}_n\boldsymbol{\Sigma}_n\boldsymbol{W}_n^T.$$

Furthermore, we see that e.g.

$$\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{V}_n = \boldsymbol{X}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{W}_n\boldsymbol{\Sigma}_n^{-1} = \boldsymbol{X}\boldsymbol{W}_n\boldsymbol{\Lambda}_n\boldsymbol{\Sigma}_n^{-1} = \boldsymbol{X}\boldsymbol{W}_n\boldsymbol{\Sigma}_n^{-1}\boldsymbol{\Lambda}_n = \boldsymbol{V}_n\boldsymbol{\Lambda}_n.$$

Thus, we must have $\boldsymbol{V}_n \subset \boldsymbol{V}_d$, i.e. all column vectors of $\boldsymbol{V}_n$ are eigenvectors of $\boldsymbol{X}\boldsymbol{X}^T$. By symmetry, the first k eigenvalues of $\boldsymbol{\Lambda}_d$ and $\boldsymbol{\Lambda}_n$ match and so do the singular values. Conclusively, the economic SVD (as in Definition 3.1.4) of $\boldsymbol{X}$ is given by the set of matrices with $k \times k$ dimensional singular value matrix. $\square$

Note here that, in order to compute an SVD, one can also apply QR-style algorithms instead of solving a suitable eigenvalue problem. This is sometimes the numerically more stable method, as the singular values are the square roots of the respective eigenvalues of the correlation matrices or *Gramians*. Consequently, we will synonymously identify the first k POD-modes of $\boldsymbol{X}$ with the first k left singular vectors of its SVD. Algorithm 8 describes the procedure to compute the first k POD-modes for a matrix $\boldsymbol{X}$, where EIG($\boldsymbol{A}, k$) returns the eigenpairs corresponding to the largest k eigenvalues. We refer

Algorithm 8 $\boldsymbol{V}$ = POD($\boldsymbol{X}, k$)

```
1: $[d, n] \leftarrow$dims($\boldsymbol{X}$)
2: if $d \leq n$ then
3:     $[\boldsymbol{V}, \Lambda] \leftarrow$EIG($\boldsymbol{X}\boldsymbol{X}^T, k$)
4: else
5:     $[\boldsymbol{W}, \Lambda] \leftarrow$EIG($\boldsymbol{X}^T\boldsymbol{X}, k$)
6:     $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}\Lambda^{-\frac{1}{2}}$
7: end if
8: return $\boldsymbol{V}$
```

the reader to [86, 98] and references therein for more details on POD/PCA and SVD.

3.1.2 POD-Greedy

The "*POD-Greedy*" algorithm was first introduced in [72] and is an approach to compute a reduced basis tailored to the context of parameterized dynamical systems or evolution equations. Basically, it computes the reduced basis from a set of trajectories (for different parameters) whilst compressing the time-information of each trajectory via POD. Assume to be given a set of *training parameters* $\Xi \subset \mathcal{P} \subseteq \mathbb{R}^p$ and corresponding trajectories of the affine-parametric system (3.3, p.87), which we denote by $\boldsymbol{X}_{\boldsymbol{\mu}} = [\boldsymbol{x}_{\boldsymbol{\mu}}(t_0) \dots \boldsymbol{x}_{\boldsymbol{\mu}}(t_{N-1})] \in \mathbb{R}^{d \times N}$ for a given $\boldsymbol{\mu} \in \Xi$ and N discrete time-steps $0 = t_0 \leq \dots \leq t_{N-1} = T$. Algorithm 9 describes the POD-Greedy procedure, using POD as in Algorithm 8. Here $tol > 0$ describes the approximation error tolerance and $r \in \mathbb{N}$ the maximal subspace size. Note that, of course, different choices for both initial space $\boldsymbol{V}_1$ and error measure $||\cdot||_1$ can be applied. Commonly, the first POD mode of the concatenated initial values is chosen as initial space, or passing an empty $\boldsymbol{V}_1$ will result in selection of the trajectory with the largest error measure by itself, similar to starting the DEIM algorithm 7 in Section 2.3 with the largest entry of $|\boldsymbol{u}_1|$. Recently, the POD-Greedy algorithm has been proven to yield a quasi-optimal convergence rate in [69].

Algorithm 9 $\boldsymbol{V}$ = POD-Greedy($\Xi, \boldsymbol{V}_1, tol, r$)

```
1:  $\boldsymbol{V} = \boldsymbol{V}_1$
2:  $k = 2$
3:  **while** true **do**
4:      /* Error $err$ using parameter $\boldsymbol{\mu}_k$ */
5:      $[err, \boldsymbol{\mu}_k] = \arg\max_{\boldsymbol{\mu} \in \Xi} \left|\left|\boldsymbol{X}_{\boldsymbol{\mu}} - \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{X}_{\boldsymbol{\mu}}\right|\right|_F$
6:      **if** $k \geq r \vee err \leq tol$ **then**
7:          **return** $\boldsymbol{V}$
8:      **end if**
9:      $\boldsymbol{v}_k = \text{POD}(\boldsymbol{X}_{\boldsymbol{\mu}_k} - \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{X}_{\boldsymbol{\mu}_k}, 1)$
10:     $\boldsymbol{V} \leftarrow [\boldsymbol{V}\ \boldsymbol{v}_k]$
11:     $k \leftarrow k + 1$
12: **end while**
```

Remark 3.1.6. The maximum error computation of line 4 in Algorithm 9 is very expensive, as full trajectories $\boldsymbol{X}_{\boldsymbol{\mu}}$ have to be computed for all $\boldsymbol{\mu} \in \Xi$. If an a-posteriori error estimator is available, a modified variant of the POD-Greedy algorithm can be applied. Therein, line 4 is replaced by

$$[e_k, \boldsymbol{\mu}_k] = \arg\max_{\boldsymbol{\mu} \in \Xi} \Delta(\boldsymbol{V}, \boldsymbol{\mu}),$$

where $\Delta(\boldsymbol{V}, \boldsymbol{\mu})$ stands for a less costly subspace- and parameter dependent error estimator. This way, no extra full simulations but for the selected error-maximizing parameters have to be computed and larger training parameter sets Ξ can be employed. The POD-DEIM a-posteriori error estimators presented in Section 5.2 are promising candidates for application in this context.

3.2 Projection of nonlinear $\boldsymbol{f}$

As already pointed out at the end of Section 3.1, the straightforward subspace projection of general nonlinear components of dynamical systems still involves high-dimensional quantities, whose reduction was aimed at in the first place. In detail, given biorthogonal $\boldsymbol{V}, \boldsymbol{W}$ (e.g. obtained by POD (Section 3.1.1) or POD-Greedy (Section 3.1.2)), the projected nonlinear term of (3.3, p.87) writes as $\boldsymbol{W}^T \boldsymbol{f}(\boldsymbol{V}\boldsymbol{z}(t)) \in \mathbb{R}^r$, but yet we have

$$\boldsymbol{f}(\boldsymbol{V}\boldsymbol{z}(t)) \in \mathbb{R}^d \qquad \text{and} \qquad \boldsymbol{V}\boldsymbol{z}(t) \in \mathbb{R}^d.$$

Consequently, today's methods dealing with projection based MOR of nonlinear dynamical systems try to provide means that allow for a *d-independent* evaluation complexity of the projected nonlinear term. This can be done in various ways, however, most frequently either a certain structure of $\boldsymbol{f}$ is assumed or suitable approximations to $\boldsymbol{f}$ are proposed. Of course, the first approach is of its own interest, as, for example, bilinear quadratic nonlinearities have recently been shown to be efficiently projectable while covering a large class of systems in [12]. While providing approximations to arbitrary $\boldsymbol{f}$ seems to be the more general approach, this approach faces the trade-off between loss of accuracy and improved speed by efficient projection. One well known method is the trajectory piece-wise linear (TPWL) approach [135, 136], which basically creates an affine-linear approximation of $\boldsymbol{f}$ by a weighted sum of Jacobians at suitable locations. This method is extended by moment matching techniques in [15] or to a piece-wise polynomial scheme in [45]. An "approximate reduction" method is introduced in [166] and model reduction for weakly nonlinear systems by bilinearization has been discussed in [132], for example.

In this work we shall investigate two different approaches that also use approximations of $\boldsymbol{f}$. Motivated by [131], we investigate the conditions for which kernel expansions yield efficient projected terms in Section 3.2.1. The projection behavior for the DEIM approximation is detailed in Section 3.2.2 along the lines of [23]. Note that the technique of using a DEIM approximation for the nonlinear part in principle corresponds to an approach involving an empirical operator interpolation in the PDE setting [47, 75], cf. Section 2.3.

3.2.1 Projection of kernel expansions

In this section we shall assume that the system's nonlinearity $\boldsymbol{f}$ is approximated by a product kernel expansion

$$\hat{\boldsymbol{f}}(\boldsymbol{x}, t, \boldsymbol{\mu}) = \sum_{i=1}^{n} \boldsymbol{c}_i K(\boldsymbol{x}_i, \boldsymbol{x}) K_t(t_i, t) K_{\mathcal{P}}(\boldsymbol{\mu}_i, \boldsymbol{\mu}), \tag{3.9}$$

with coefficient vectors $\boldsymbol{c}_i \in \mathbb{R}^d$ and center triples $(\boldsymbol{x}_i, t_i, \boldsymbol{\mu}_i), i = 1 \ldots k$. Here, $K, K_t, K_{\mathcal{P}}$ denote the state, time and parameter kernels that can be given independently. If the considered system is time or parameter independent, the corresponding kernels are omitted in (3.9). For example, the nonlinear kernel approximation techniques introduced in Sections 2.1 and 2.2 can be readily applied in order to obtain such expansions.

Clearly, the projection of (3.9) into $\mathcal{V}, \mathcal{W}$ directly applies to the coefficient vectors $\boldsymbol{c}_i$ via $\tilde{\boldsymbol{c}}_i := \boldsymbol{W}^T \boldsymbol{c}_i, i = 1 \ldots k$. This results in an overall reduced expansion

$$\tilde{\hat{\boldsymbol{f}}}(\boldsymbol{z}, t, \boldsymbol{\mu}) := \boldsymbol{W}^T \hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(t), t, \boldsymbol{\mu}) = \sum_{j=1}^{k} \tilde{\boldsymbol{c}}_i K(\boldsymbol{V}\boldsymbol{z}(t), \boldsymbol{x}_i) K_t(t_i, t) K_{\mathcal{P}}(\boldsymbol{\mu}_i, \boldsymbol{\mu}).$$

Unfortunately, this is still expensive to simulate given that K is still evaluated at $\boldsymbol{V}\boldsymbol{z}(t) \in \mathbb{R}^d$. In order to avoid input arguments of high dimension d, we consider two special classes of kernels, where we will omit the time-dependency of $\boldsymbol{x}(t)$ or $\boldsymbol{z}(t)$ for ease of reading.

Inner product kernels

As also mentioned in [131], the first type of kernels that allow efficient argument evaluations are the inner product kernels as in Definition 1.1.3. Then we have

$$K(\boldsymbol{V}\boldsymbol{z}, \boldsymbol{x}_i) = \phi(\langle \boldsymbol{V}\boldsymbol{z}, \boldsymbol{x}_i \rangle_G) = \phi(\langle \boldsymbol{z}, \boldsymbol{z}_i \rangle) =: \tilde{K}(\boldsymbol{z}, \boldsymbol{z}_i),$$

for $\boldsymbol{z}_i := \boldsymbol{V}^T \boldsymbol{G} \boldsymbol{x}_i \in \mathbb{R}^r,\ i = 1 \dots k$. This way, it is sufficient to project the state space centers $\boldsymbol{x}_i$ into $\mathcal{V}$ and any evaluations of K can be computed *loss-less* and efficiently (e.g. during the reduced simulation) via $\tilde{K}$. Some examples for those kernels are the linear and polynomial kernels, see Section 1.1.1.

Translation- and rotation-invariant kernels

In extension to [131] we introduce a further class of kernels that allow for efficient *loss-less* argument evaluations, namely the translation- and rotation-invariant kernels as given in Definition 1.1.4. We impose the additional requirement $\boldsymbol{x}_i \in \mathcal{V}$, i.e. $\boldsymbol{x}_i = \boldsymbol{V}\boldsymbol{z}_i$ for some $\boldsymbol{z}_i \in \mathbb{R}^r,\ i = 1 \dots k$. Then we obtain

$$K(\boldsymbol{V}\boldsymbol{z}, \boldsymbol{x}_i) = \phi(||\boldsymbol{V}\boldsymbol{z} - \boldsymbol{V}\boldsymbol{z}_i||_G) = \phi(||\boldsymbol{z} - \boldsymbol{z}_i||_{\boldsymbol{V}^T\boldsymbol{G}\boldsymbol{V}}) =: \tilde{K}(\boldsymbol{z}, \boldsymbol{z}_i),$$

with $\boldsymbol{V}^T\boldsymbol{G}\boldsymbol{V}$ being a small $\mathbb{R}^{r\times r}$ matrix inducing a new norm on $\mathbb{R}^r$, which equals the standard Euclidean norm if $\boldsymbol{G} = \boldsymbol{I}_d$.

Note that the assumption $\boldsymbol{x}_i \in \mathcal{V}$ is of a technical nature. We either extend $\mathcal{V}$ by the span of the $\boldsymbol{x}_i$, or, if the kernel expansion is created with knowledge of $\mathcal{V}$, one can choose $\boldsymbol{x}_i \in \mathcal{V}$ in the first place.

Finally, with the "reduced" kernel $\tilde{K}$ from either variant, we obtain

$$\tilde{\hat{\boldsymbol{f}}}(\boldsymbol{z}) = \sum_{j=1}^{k} \tilde{\boldsymbol{c}}_i \tilde{K}(\boldsymbol{z}(t), \boldsymbol{z}_i) K_t(t_i, t) K_{\mathcal{P}}(\boldsymbol{\mu}_i, \boldsymbol{\mu}), \tag{3.10}$$

whose evaluation complexity does not involve d. Moreover, given $\boldsymbol{x}_i \in \mathcal{V}$ for the latter case, the projection does not infer any extra loss in accuracy as shown above. Possible applications of the reduction technique involve electric circuits [131] or biochemical systems [34], or more generally, any systems where a good approximation of the nonlinearity with few kernel components can be found. We will give a nonlinear circuit example in Section 3.4.1.

3.2.2 DEIM approximation projection

An efficient subspace projection can also be pursued for the DEIM approximation scheme introduced in Section 2.3. Recall for $m \leq d$ the m-th order DEIM approximation

$$\hat{\boldsymbol{f}}_m(\boldsymbol{x}) := \boldsymbol{U}_m(\boldsymbol{P}_m^T \boldsymbol{U}_m)^{-1} \boldsymbol{P}_m^T \boldsymbol{f}(\boldsymbol{x}) \tag{3.11}$$

of $\boldsymbol{f}$ as in (2.43, p.81) with $\boldsymbol{U}_m := [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m]$ and $\boldsymbol{P}_m = [\boldsymbol{e}_{\wp_1}, \ldots, \boldsymbol{e}_{\wp_m}]$. Now, an efficient projection is possible due to the affine structure of (3.11), as projection with $\boldsymbol{V}, \boldsymbol{W}$ yields

$$\tilde{\hat{\boldsymbol{f}}}_m(\boldsymbol{z}) := \tilde{\boldsymbol{U}}_m \boldsymbol{P}_m^T \boldsymbol{f}(\boldsymbol{V}\boldsymbol{z}) = \boldsymbol{W}^T \boldsymbol{U}_m(\boldsymbol{P}_m^T \boldsymbol{U}_m)^{-1} \boldsymbol{P}_m^T \boldsymbol{f}(\boldsymbol{V}\boldsymbol{z}), \tag{3.12}$$

with reduced $\tilde{\boldsymbol{U}}_m := \boldsymbol{W}^T \boldsymbol{U}_m(\boldsymbol{P}_m^T \boldsymbol{U}_m)^{-1} \in \mathbb{R}^{r \times m}$ and $\tilde{\hat{\boldsymbol{f}}}_m : \mathbb{R}^r \to \mathbb{R}^r$. Similar to the kernel approximation case, we still need to deal with $\boldsymbol{f}(\boldsymbol{V}\boldsymbol{z})$. In the DEIM case, we now make use of the additional sparsity assumption on the Jacobian of $\boldsymbol{f}$ as described in Section 2.3. With that we obtain

$$\boldsymbol{P}_m^T \boldsymbol{f}(\boldsymbol{V}\boldsymbol{z}) = \begin{pmatrix} f_{\wp_1}(\boldsymbol{V}\boldsymbol{z}) \\ \vdots \\ f_{\wp_m}(\boldsymbol{V}\boldsymbol{z}) \end{pmatrix} = \begin{pmatrix} f_{\wp_1}(\boldsymbol{V}[\boldsymbol{j}_{\wp_1}, :]\boldsymbol{z}) \\ \vdots \\ f_{\wp_m}(\boldsymbol{V}[\boldsymbol{j}_{\wp_m}, :]\boldsymbol{z}) \end{pmatrix} \in \mathbb{R}^m,$$

where $\boldsymbol{j}_i = \{j_1^i, j_2^i, \ldots, j_{n_i}^i\} = \texttt{find}(\boldsymbol{J}_P[i, :])$ denotes the n_i component-indices of $\boldsymbol{x}$ required to evaluate f_i. Now, as $\boldsymbol{V}[\boldsymbol{j}_{\wp_i}, :] \in \mathbb{R}^{n_i \times r}$, it reconstructs just the full state space quantities needed to evaluate the $\wp_i$-th component function of $\boldsymbol{f}$. Again, essentially

$$\sum_{i=1}^{m} |\boldsymbol{j}_{\wp_i}| = \sum_{i=1}^{m} n_i \ll d$$

is required to achieve a considerable dimension reduction, which imposes an upper limit on m depending on the sparsity pattern of $\boldsymbol{f}$.

3.3 Offline/online decomposition

A key concept to obtain efficient reduced order models is the "*offline-online decomposition*", which is well-known from the RB setting [127] and has been successfully applied to obtain fast reduced models under various conditions, see e.g. [74, 10, 47, 173, 83]. Basically, every computation step involving high-dimensional operations of minimum complexity $\mathcal{O}(d)$ are done in the *offline stage*, whilst the simulation of the reduced system is done in the *online stage*, only involving operations whose complexity is dependent on the reduced dimension $r \ll d$ and other "small" quantities like the DEIM order m, for example. As shortly motivated in the introductory part of this work, the need for such a decomposition is evident considering high-level applications like optimization, control, Monte-Carlo simulations or inverse modeling. They all involve multiple simulations of the considered model, resulting in an extensive overall simulation time unless a fast online simulation is possible.

In detail, for the dynamical systems and approximations introduced so far, the offline/online phases are given as follows. Note here, in the following Chapter 5, we shall discuss different error estimation techniques, which will add additional steps to both stages.

3.3.1 Offline stage

Training data

For any system type introduced in Section 1.3, the first step is to compute training data. For parameterized systems, we first define the set of training parameters $\Xi \subset \mathcal{P}$ for the parameter domain $\mathcal{P} \subseteq \mathbb{R}^p$. Unless more sophisticated methods are applied, a set of trajectories has to be computed, which involves potentially many expensive simulations of the full system, yielding training data X and Y.

Projection spaces

Next, using e.g. a global POD or the POD-Greedy method introduced in the previous sections, the projection matrices $\boldsymbol{V}, \boldsymbol{W} \in \mathbb{R}^{d \times r}$ are computed, whose columns span the reduced basis $\mathcal{V}$ or test space $\mathcal{W}$, respectively.

Nonlinear approximation

If the system's nonlinearity already allows for efficient projection, then this step can be omitted. Otherwise, the approximation of $\boldsymbol{f}$ is computed by e.g. using the component-wise SVR from Section 2.1, the VKOGA algorithm from Section 2.2.2, the DEIM from Section 2.3 or any other suitable approximation technique. For kernel-based approximations, as shortly discussed in Section 3.2.1, a-priori knowledge of $\boldsymbol{V}$ can be helpful in obtaining efficient approximations, which is why this step is listed after computing $\boldsymbol{V}, \boldsymbol{W}$.

System projection

As introduced in [74], the operations during the offline phase for the affine-parametric components the system (1.12a, p.19) consists of computing

$$\begin{aligned}
\tilde{\boldsymbol{A}}_i &= \boldsymbol{W}^T \boldsymbol{A}_i \boldsymbol{V}, \quad i = 1 \dots Q_A, & \tilde{\boldsymbol{B}}_i &= \boldsymbol{W}^T \boldsymbol{B}_i, \quad i = 1 \dots Q_B,\\
\tilde{\boldsymbol{C}}_i &= \boldsymbol{C}_i \boldsymbol{V}, \quad i = 1 \dots Q_C, & \tilde{\boldsymbol{x}}_i^0 &= \boldsymbol{V}^T \boldsymbol{x}_i^0, \quad i = 1 \dots Q_0.
\end{aligned}$$

Now depending on the applied approximation, if used, we compute

- **Kernel-based approximation with inner product kernels**
  Compute the reduced centers $\boldsymbol{z}_i = \boldsymbol{V}^T \boldsymbol{G} \boldsymbol{x}_i, i = 1 \dots k$.

- **Kernel-based approximation with RBF kernels**
  Compute the new RBF-norm inducing matrix $\boldsymbol{V}^T \boldsymbol{G} \boldsymbol{V} \in \mathbb{R}^{r \times r}$. If $\boldsymbol{x}_i \in \mathcal{V}$ has not been ensured during nonlinear approximation, augment $\boldsymbol{V}$ by the centers $\boldsymbol{x}_i$.

- **DEIM approximation**
  Assuming to have $\boldsymbol{U}_M, \boldsymbol{P}_M, M \leq d$ available from nonlinear approximation, choose $m \leq M$. Then compute $\tilde{\boldsymbol{U}}_m = \boldsymbol{W}^T \boldsymbol{U}_m (\boldsymbol{P}_m^T \boldsymbol{U}_m)^{-1} \in \mathbb{R}^{r \times m}$ and $\boldsymbol{V}[\boldsymbol{j}_{\wp_i}, :] \in \mathbb{R}^{n_i \times r}, i = 1 \dots m$. For every new $m \leq M$, those steps need to be repeated.

3.3.2 Online stage

At the online stage, the reduced system

$$\begin{aligned}
\boldsymbol{z}'(t) &= \tilde{\hat{\boldsymbol{f}}}(\boldsymbol{z}(t), t, \boldsymbol{\mu}) + \sum_{i=1}^{Q_A} \theta_i^A(t, \boldsymbol{\mu}) \tilde{\boldsymbol{A}}_i \boldsymbol{z}(t) + \sum_{i=1}^{Q_B} \theta_i^B(t, \boldsymbol{\mu}) \tilde{\boldsymbol{B}}_i \boldsymbol{u}(t),\\
\boldsymbol{z}(0) &= \sum_{i=1}^{Q_0} \theta_i^0(\boldsymbol{\mu}) \tilde{\boldsymbol{x}}_i^0 =: \boldsymbol{z}_0(\boldsymbol{\mu}), \qquad \boldsymbol{w}^r(t) = \sum_{i=1}^{Q_C} \theta_i^C(t, \boldsymbol{\mu}) \tilde{\boldsymbol{C}}_i \boldsymbol{z}(t).
\end{aligned}$$

as in (3.4, p.88) is solved for $t \in [0,T]$, where $\tilde{\hat{\boldsymbol{f}}}$ is either given by (3.10, p.100) or (3.12, p.100) (for an $m \leq M$).

3.4 Applications

In this section we shall introduce some example systems which will be subsequently used during this work.

3.4.1 Nonlinear circuit

The first model we consider is the nonlinear resistor–capacitor circuit (RC circuit or RC ladder) introduced in [28] which has also been mentioned in the key paper [131]. This model has been frequently used to investigate reduction techniques for bilinear or weakly nonlinear systems [136, 31, 9].

The RC circuit consists of nonlinear resistors and unit capacitors, with intermediate voltage measurement nodes denoted by $\boldsymbol{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$. An independent current source with input signal $u(t)$ is connected at node one. The current/voltage dependence for each resistor follows the nonlinear relation

$$g(x) = e^{40x} + x - 1.$$

Considering Kirchhoff's current law, we obtain the change of voltage as

$$\begin{aligned}\boldsymbol{x}' &= \begin{pmatrix} -g(x_1) - g(x_1 - x_2) \\ g(x_1 - x_2) - g(x_2 - x_3) \\ \vdots \\ g(x_{d-1} - x_d) \end{pmatrix} \\ &= \begin{pmatrix} -2x_1 + x_2 - e^{40x_1} + 2 - e^{40(x_1 - x_2)} \\ x_1 - 2x_2 + x_3 + e^{40(x_1 - x_2)} - e^{40(x_2 - x_3)} \\ \vdots \\ x_{d-1} - 2x_d + e^{40(x_{d-1} - x_d) - 1} \end{pmatrix}\end{aligned}$$

$$= \boldsymbol{Ax} + \begin{pmatrix} e^{40x_1} + 2 - e^{40(x_1-x_2)} \\ e^{40(x_1-x_2)} - e^{40(x_2-x_3)} \\ \vdots \\ e^{40(x_{d-1}-x_d)} - 1 \end{pmatrix} =: \boldsymbol{Ax} + \boldsymbol{f}(\boldsymbol{x}),$$

where $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ is the discrete Laplacian and $\boldsymbol{f}$ the nonlinear part. Further, extending [9], we consider the input signals

$$u_1(t) = e^{-t}, \qquad u_2(t) = \frac{1}{2}\cos\left(\frac{2\pi t}{10}\right),$$
$$u_3(t) = \begin{cases} 0 & t \leq 0.3 \\ 1 & t > 0.3 \end{cases}, \qquad u_4(t) = \frac{\cos(\pi t) + 1}{t+1},$$

which connects to the circuit via $\boldsymbol{B} = [1\ 0\ \dots\ 0]^T \in \mathbb{R}^d$. The output signal $w(t)$ is the voltage between node one and ground, mapped via $\boldsymbol{C} = \boldsymbol{B}^T$. Altogether, we obtain a nonlinear dynamical system

$$\boldsymbol{x}'(t) = \boldsymbol{Ax}(t) + \boldsymbol{f}(\boldsymbol{x}(t)) + \boldsymbol{B}u_i(t), \quad w(t) = \boldsymbol{Cx}(t), \quad \boldsymbol{x}(0) = \boldsymbol{0}, \quad (3.13)$$

where we set $T = 3$ and $\Delta t = 0.0025$. For the full model dimension we set $d = 10000$ and use a semi-implicit Euler scheme

$$(\boldsymbol{I}_d - \Delta t\boldsymbol{A})\boldsymbol{x}(t_{i+1}) = \boldsymbol{x}(t_i) + \Delta t\big(\boldsymbol{f}(\boldsymbol{x}(t_i)) + \boldsymbol{Bu}(t_i)\big), \quad (3.14)$$

on n_t equidistant time-steps $t_i := (i-1)\Delta t,\ \Delta t = \frac{T}{n_t-1}, i = 1\dots n_t$.

This results in a training set $X = X_{u_1} \cup X_{u_2} \cup X_{u_3} \cup X_{u_4}$ composed of four trajectories for different inputs u_i. For subspace computation, we perform a global POD on X with target subspace size $r = 40$. For nonlinear approximation, we use the VKOGA Algorithm 5/6 with $\boldsymbol{f}, X$ and the f-Greedy selection criteria. The maximum expansion size is $n_{max} = 300$ along with a maximum relative error of 10^{-5} and maximum absolute residual error of 10^{-5} on X. As kernel we use the Gaussian 1.3 with 18 linearly spaced γ values ranging in $[0.05, 0.33]$. Figure 3.1 shows the POD singular value decay on the left and the maximum absolute approximation errors on X for subsequent iteration steps of the VKOGA computing $\hat{\boldsymbol{f}}$ on the right. The different plots on the right hand side correspond to different γ values used as kernel hyperparameter.

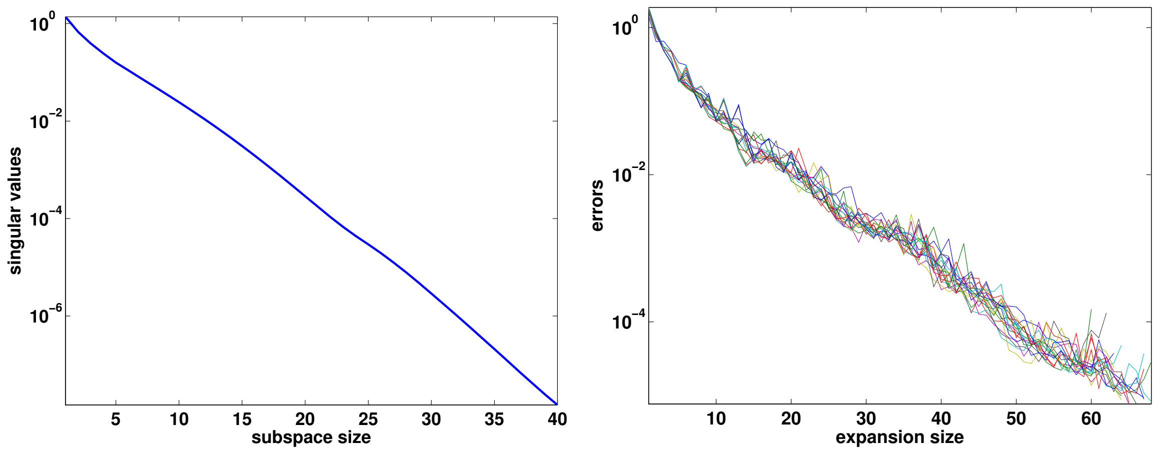


Figure 3.1: Absolute (left) and relative (right) approximation errors on X for $\hat{\boldsymbol{f}}$

Most remarkably, the singular values decrease exponentially and hence the approximation quality of $\mathcal{V}$ increases at the same exponential rate. Also the approximation error for each selection of γ values has exponential error decay, where the algorithm gets unstable at around $n = 60$ for larger γ values. From all 18 hyperparameters, the 5th with $\gamma = 0.116596$ was chosen, yielding a 10000-dimensional vectorial kernel expansion $\hat{\boldsymbol{f}}$ with $n = 65$ centers. For that, Figure 3.2 shows the absolute (left) and relative (right) approximation errors on X for $\hat{\boldsymbol{f}}$.

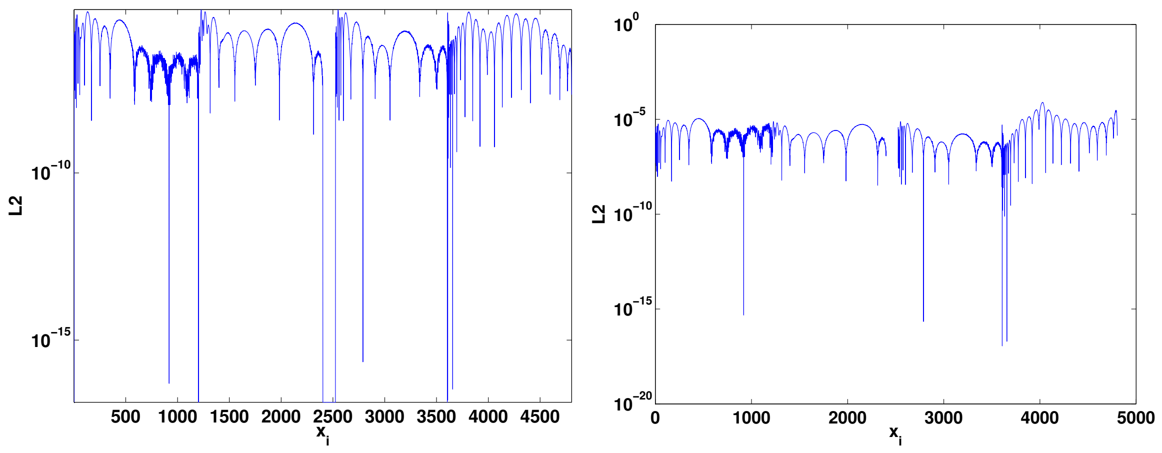


Figure 3.2: Absolute (left) and relative (right) approximation errors on X for $\hat{\boldsymbol{f}}$

Now, applying a Galerkin projection to the full system 3.13 using the first 40 POD modes and substituting $\hat{\boldsymbol{f}}$ we obtain the reduced system, where Figure 3.3 shows the simulation results for $d = 100$ and different inputs u_i.

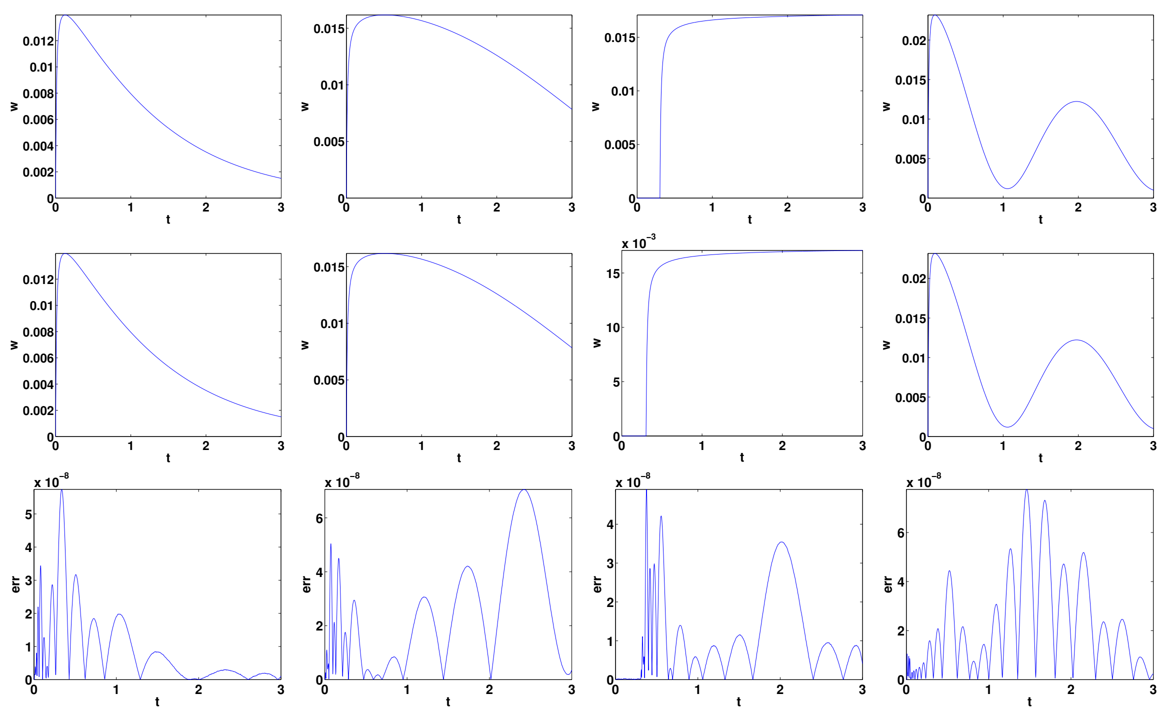


Figure 3.3: RC ladder dynamics for four different inputs u_i per column. Full (top), reduced dynamics (middle) and errors (bottom)

Shown are the full simulations on the top row and the reduced simulations on the middle row, respectively. The bottom row contains the discrete L^2-errors of the output over time, which are for $u_1 : 4.75837 \times 10^{-7}$, $u_2 : 1.09481 \times 10^{-6}$, $u_3 : 5.35993 \times 10^{-7}$, $u_4 : 1.02407 \times 10^{-6}$, which is considered very competitive considering a subspace size of $r = 40$ and a kernel expansion with $n = 65$ centers.

Furthermore, we want to investigate the approximation quality for previously unknown inputs and define

$$u_5(t) = \frac{\cos(0.5\pi t) + 1}{t + 1}, \quad u_6(t) = \begin{cases} 0 & t \leq 1.3 \\ 2 & t > 1.3 \end{cases}, \quad u_7(t) = \frac{\cos(3\pi t) + 1}{2t + 1}.$$

These functions state three new inputs with previously unknown characteristics: u_5 has a slower oscillation, u_6 has a stronger discontinuity at a different time-step and u_7 oscillates and decays faster. The results are shown in Figure 3.4.

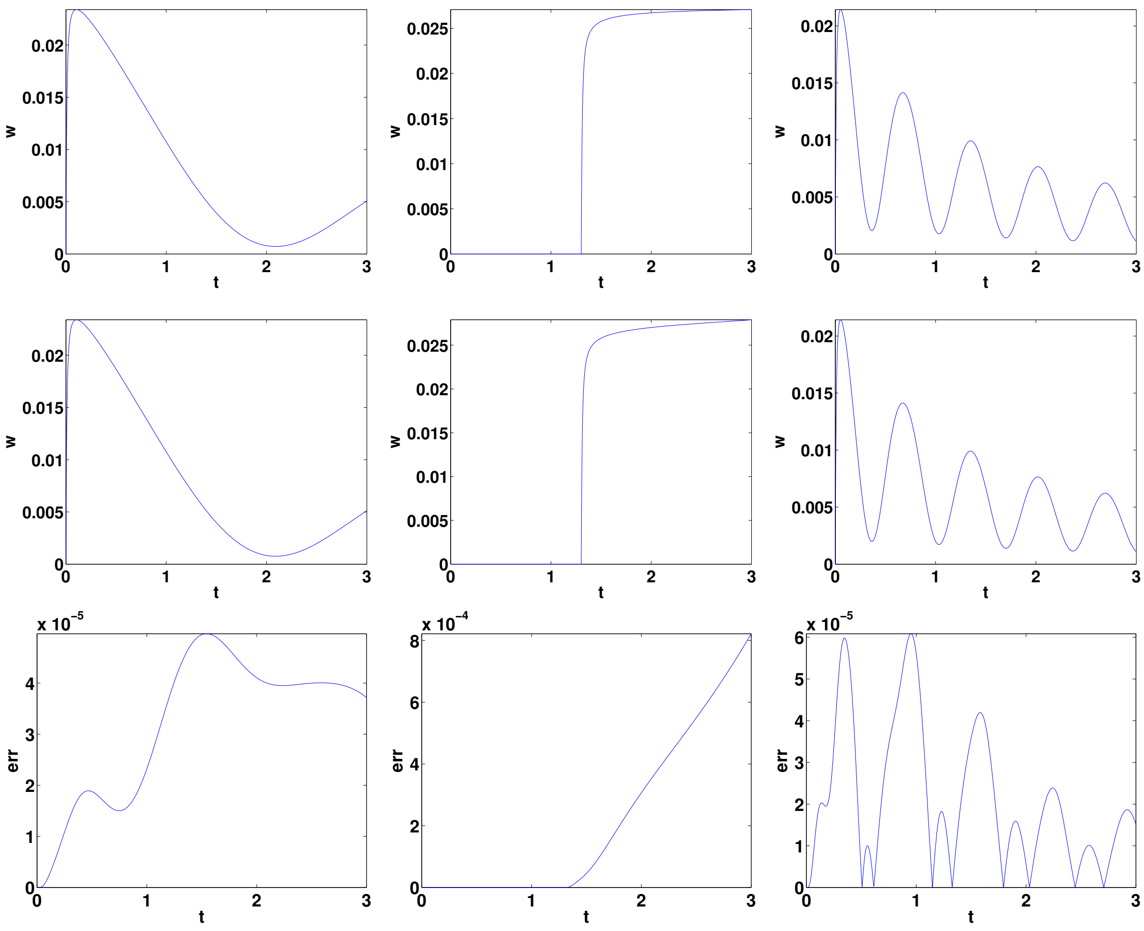


Figure 3.4: RC ladder dynamics for u_5, u_6, u_7. Full (top row), reduced (middle row) and absolute errors (bottom row)

For these new inputs, the discrete L^2 errors are for $u_5 : 0.00120874, u_6 : 0.0118018$ and $u_7 : 0.000930239$. As to expect, this is larger than the simulation errors for the training inputs, but still around one percent for u_6 and orders of magnitude less for the others. Consequently, this demonstrates the capabilities of the proposed reduction scheme also for previously unknown inputs.

3.4.2 Cell apoptosis

This model has been originally introduced for 1D in [34] and has hence been refined in cooperation with M. Daub. to its present state, using a 2D-domain $\Omega = [0,1] \times [0,1.5]$. The model is basically a reaction-diffusion system developed in order to study the dynamical behavior of protein concentrations of a cell apoptosis model in space and time. Due to the large variety of possible (parameter) choices, a comprehensive analysis cannot be carried out within this work and we refer to [34] and references therein for full model details and background.

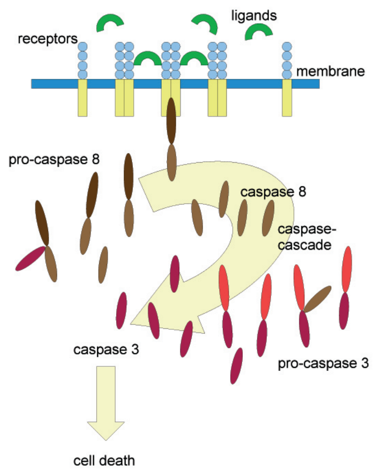


Figure 3.5: The "caspase cascade" triggering the controlled cell death [34]

The involved proteins build a network called "caspase cascade", which is depicted in Figure 3.5. It consists of four different reactants x_i, y_i, x_a, y_a called procaspase-8, procaspase-3, caspase-8 and caspase-3, respectively. Their interaction is modeled by the system

$$\begin{aligned}
\frac{\partial x_a}{\partial t} &= k_{c1} x_i y_a - k_{d1} x_a + D_1 \Delta x_a, \\
\frac{\partial y_a}{\partial t} &= k_{c2} y_i x_a^2 - k_{d2} y_a + D_2 \Delta y_a, \\
\frac{\partial x_i}{\partial t} &= -k_{c1} x_i y_a - k_{d3} x_i + k_{p1} + D_3 \Delta x_i, \\
\frac{\partial y_i}{\partial t} &= -k_{c2} y_i x_a^2 - k_{d4} y_i + k_{p2} + D_4 \Delta y_i,
\end{aligned}$$

where k_* and D_i are suitable scaled constants controlling creation, interaction and diffusion of the involved quantities that are detailed in [34]. Furthermore, the procaspase-8 gets activated by receptors located in the cellular membrane, which naturally leads to geometrically parameter-dependent boundary conditions. Consequently, we introduce for $\mu_1 \in [0,1]$ a boundary part

$$\Gamma_{\mu_1} := \{\boldsymbol{x} \in \partial\Omega \mid |x_1 - 0.5| \leq 0.5\mu_1 \vee |x_2 - 0.75| \leq 0.75\mu_1\} \subseteq \partial\Omega,$$

on which we impose the Neumann conditions

$$\left.\frac{\partial x_a}{\partial n}\right|_{\Gamma_{\mu_1}} = \mu_2 x_i, \qquad \left.\frac{\partial x_i}{\partial n}\right|_{\Gamma_1} = -\mu_2 x_i,$$

where $\mu_2 \in [10^{-5}, 10^{-2}]$ describes the reaction rate of the activation of procaspase-8. On all other parts of $\partial\Omega$ we enforce homogeneous Neumann conditions and have $x_a = y_a = 10^{-2},\ x_i = y_i = 0.01$ as initial conditions.

Next, discretization on a 100×150 grid leads to a discrete system

$$\boldsymbol{x}'(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{\mu}),$$

with $d = 60000$-dimensional state space $\boldsymbol{x}(t) = (\boldsymbol{x}_a^T, \boldsymbol{y}_a^T, \boldsymbol{x}_i^T, \boldsymbol{y}_i^T)^T$, discrete Laplacian $\boldsymbol{A}$ and nonlinear reaction operator $\boldsymbol{f}$. We simulate up to $T = 500s$ and again discretize in time using a semi-implicit Euler scheme (3.14, p.105) with $\Delta t = 5s$. For model reduction we apply the DEIM procedure 7 on $\boldsymbol{f}$ with $M = 200$ and choose 200 random parameters $\Xi \subset \mathcal{P} = [0,1] \times [10^{-5}, 10^{-2}]$ to obtain training trajectories $X := \{X_{\boldsymbol{\mu}} \mid \boldsymbol{\mu} \in \Xi\}$. The projection subspace $\boldsymbol{V}$ was generated by the POD-Greedy algorithm 9 on X augmented with Y (evaluations of $\boldsymbol{f}$ on X) and $\boldsymbol{AX}$, which will be of interest again in Section 5.4. Running up to an error tolerance of 10^{-6} the resulting subspace is of dimension 282, a reduction by a factor of ≈ 212. In order to compute the matrix DEIM and similarity transformations, we used 20200 uniformly chosen samples of X and set $M_J = 200, k_{max} = 50$. Figure 3.6 shows the simulation results on the middle slice $[0.5] \times [0, 1.5] \subset \Omega$ over time for $\boldsymbol{\mu} = (0.777, 0.00132)^T \notin \Xi$ for the different protein concentrations.

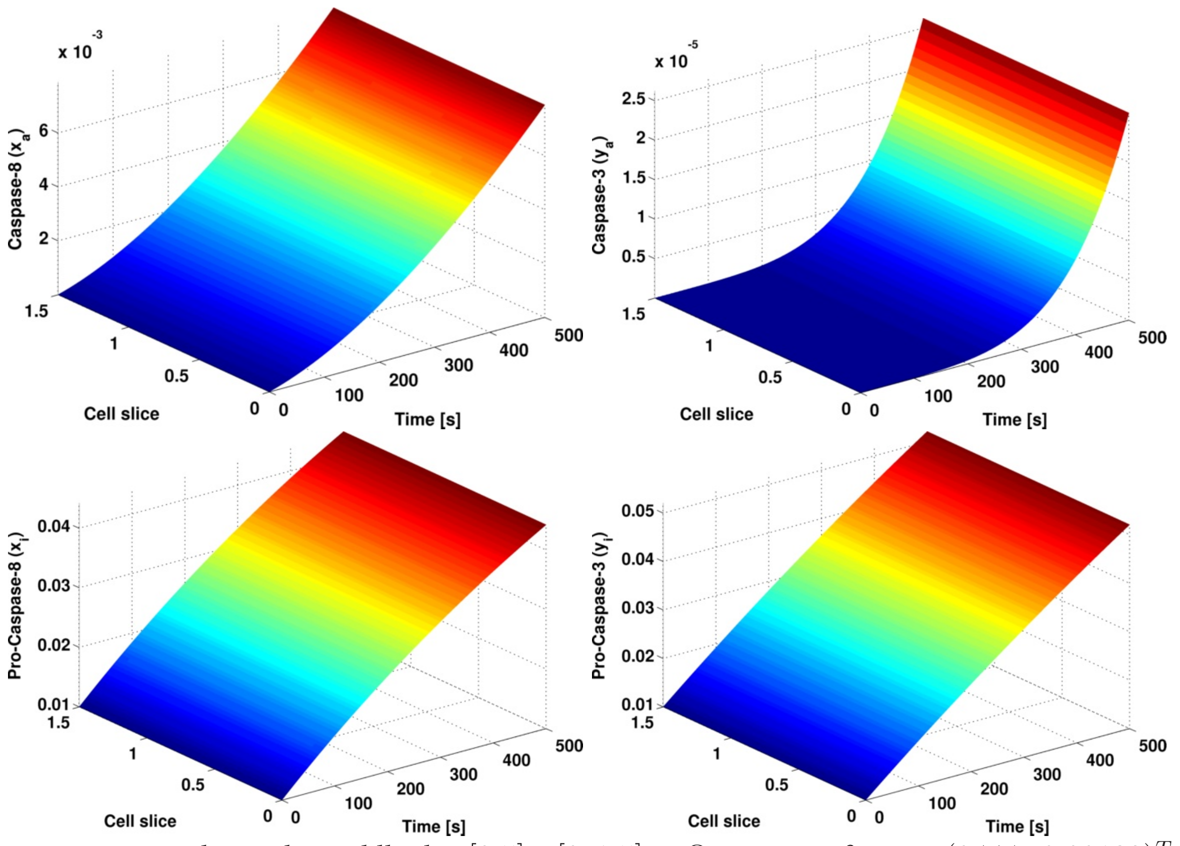
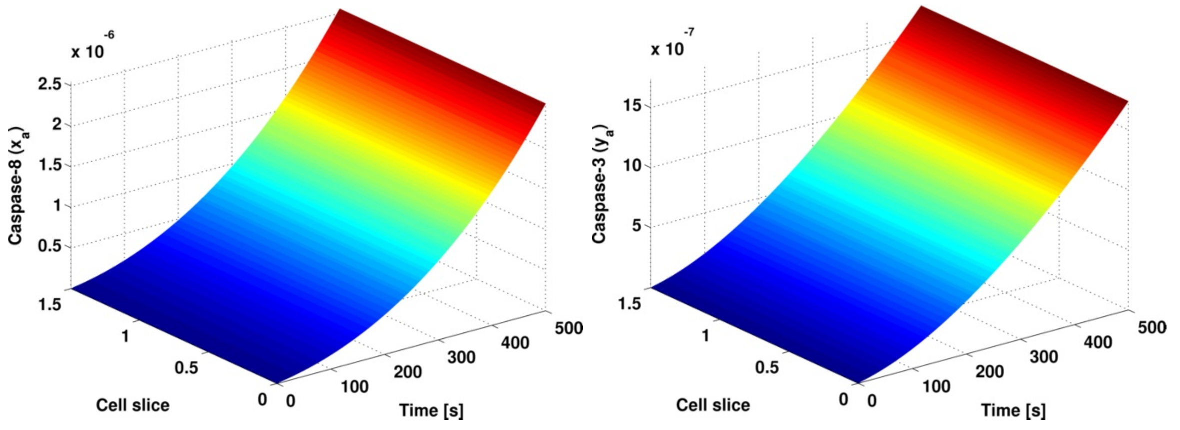


Figure 3.6: Results on the middle slice $[0.5] \times [0, 1.5] \subset \Omega$ *over time for* $\boldsymbol{\mu} = (0.777, 0.00132)^T$

Next, Figure 3.7 shows the the absolute errors of full vs. reduced solution, where a DEIM order of $m = 107$ has been used.



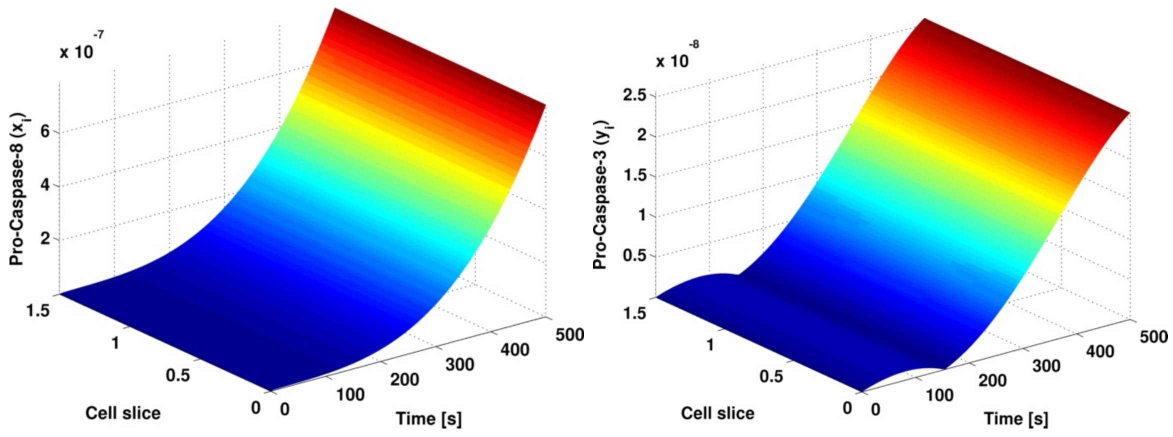


Figure 3.7: Absolute errors of full vs. reduced solution, DEIM order $m = 107$

The resulting relative error on the caspase-3/8 concentrations is about 1%, while the relative error for the pro-caspase concentrations is roughly at the order of 10^{-5}. The simulation time for the full model was $\approx 11s$, while the reduced model simulation (excluding error estimation) takes $0.3864s$ (averaged over 10 runs). This is a speedup factor of roughly 28, which is acceptable keeping in mind the threshold introduced by the natural overhead regarding the simulation procedure.

3.4.3 1D viscous Burgers equation

In this section, we consider a parameterized 1D unsteady viscous Burgers equation over the unit interval $\Omega := [0, 1]$ and time $t \in [0, T]$ with $T = 1$. This model has essentially been used in [23] as an application example for the DEIM method and has been refined in own current work [186] to the version used here. The solution $x(\xi, t)$ is given by the partial differential equation

$$\frac{\partial x}{\partial t}(\xi, t) = \mu \frac{\partial^2 x}{\partial \xi^2}(\xi, t) - \frac{\partial}{\partial \xi}\left(\frac{x(\xi, t)^2}{2}\right) + \langle \boldsymbol{b}(\xi), \boldsymbol{u}(t) \rangle, \tag{3.15}$$

with diffusion coefficient $\mu \in \mathcal{P} := [0.01, 0.06]$ and homogeneous initial and Dirichlet boundary conditions.
Further there are external forces $\boldsymbol{u}(t)$ at locations $\boldsymbol{b}(\xi) = (b_1(\xi), b_2(\xi))^T$

given by

$$u_1(t) = \sin(2\pi t), \qquad b_1(\xi) = \begin{cases} 4e^{-(\frac{\xi-0.2}{0.03})^2} & \xi \in [0.1, 0.3], \\ 0 & \text{else}, \end{cases}$$

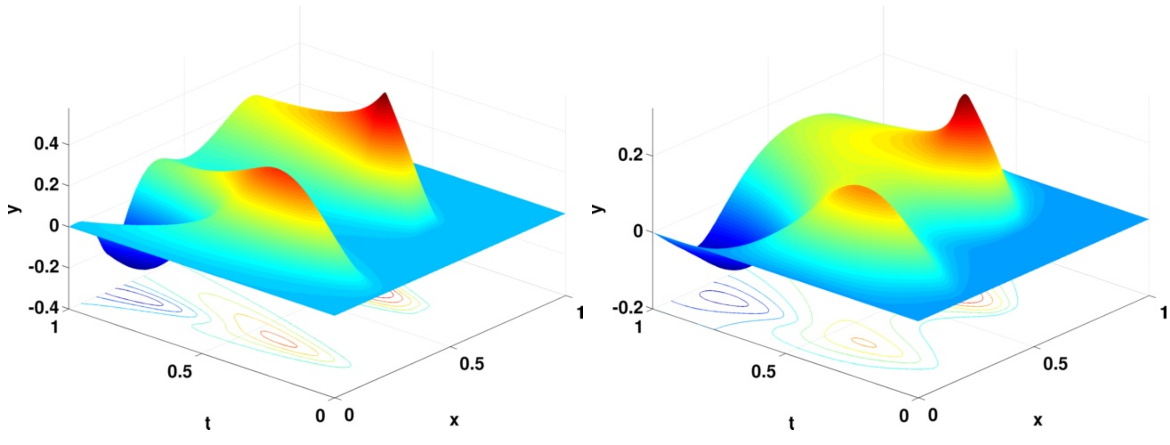
$$u_2(t) = \begin{cases} 1 & t \in [0.2, 0.4], \\ 0 & \text{else}, \end{cases} \qquad b_2(\xi) = \begin{cases} 4 & \xi \in [0.6, 0.7], \\ 0 & \text{else}. \end{cases}$$

The component $b_1(\xi)u_1(t)$ realizes an oscillating excitation centered at $\xi = 0.2$ in the shape of a Gaussian curve, and $b_2(\xi)u_2(t)$ gives a discontinuous signal over a limited time in the interval $[0.6, 0.7]$. Next, spatial discretization of the model via finite differences yields a system of $d = 500$ ODEs

$$\boldsymbol{x}'(t) = \mu \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{f}(\boldsymbol{x}(t)) + \boldsymbol{B}\boldsymbol{u}(t), \tag{3.16}$$

where $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ is the discrete Laplacian and $\boldsymbol{B} \in \mathbb{R}^{d\times 2}$. Further we have $\boldsymbol{f}(\boldsymbol{x}) = -\boldsymbol{x}.{*}\boldsymbol{A}_\xi\boldsymbol{x}$ with first-order central finite difference operator $\boldsymbol{A}_\xi \in \mathbb{R}^{d\times d}$. Here '$.*$' denotes element-wise multiplication. We choose $\boldsymbol{G} = \boldsymbol{I}_d$ (L^2-norm in state space) and the time integration of (3.16) is performed via a semi-implicit Euler scheme similar to (3.14, p.105) with $n_t = 100$ time-steps.

Figure 3.8 shows solutions of the system (3.15) for minimal (left) and maximal (middle) μ value in $\mathcal{P}$. The rightmost plot is the difference of both previous solutions and illustrates the behavioral change of the model over $\mathcal{P}$.



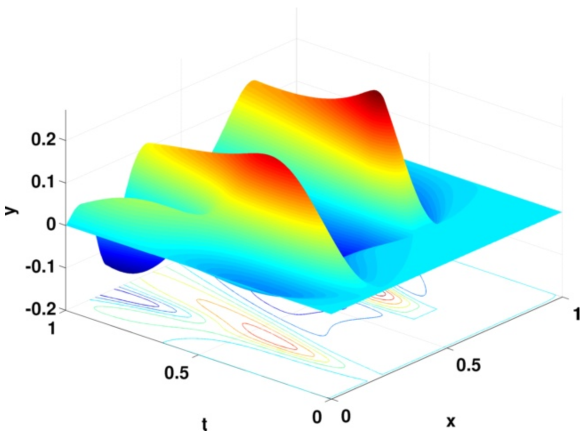


Figure 3.8: Simulation results for minimal (left), maximal (right) parameter values in $\mathcal{P}$ and their difference (bottom)

In order to compute the reduced basis, we choose a discrete parameter set $\Xi \subseteq \mathcal{P}$ of 100 log-equidistant values to generate the training trajectories. The state space projection matrices $\boldsymbol{V} = \boldsymbol{W}$ are obtained via the POD-Greedy Algorithm 9 with maximum subspace size 100. However in this experiment, we computed the true maximum L^2-state space error over all training trajectories $X_\mu := \{\boldsymbol{x}(t_1;\mu), \ldots, \boldsymbol{x}(t_{n_t};\mu)\}$ in each POD-Greedy extension step, where each trajectory data was augmented by $\boldsymbol{f}$-evaluations $Y_\mu := \{\boldsymbol{f}(\boldsymbol{x}(t_1;\mu)), \ldots, \boldsymbol{f}(\boldsymbol{x}(t_{n_t};\mu))\}, \mu \in \Xi$. We further used the span of the $\boldsymbol{B}$ matrix columns as initial space, yielding a total maximum error

$$\max_{\mu\in\Xi} \max_{\boldsymbol{x}\in X_\mu \cup Y_\mu} \|\boldsymbol{x} - \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{x}\| \leq 1.5099 \times 10^{-6}.$$

Inclusion of Y_μ and the $\boldsymbol{B}$-span in $\boldsymbol{V}$ is not necessary for good reduced trajectories at this stage, but will become more important in the context of error estimation, see Section 5.4.1. The DEIM approximation basis $\boldsymbol{U}_M$ for $\hat{\boldsymbol{f}}$ is obtained by performing a POD (See Algorithm 8) with maximum order $M = 200$ on $Y := \bigcup_{\mu\in\Xi} Y_\mu$. This gives a maximum relative error

$$\max_{\boldsymbol{x}\in\boldsymbol{X}} \left|\left|\boldsymbol{f}(\boldsymbol{x}) - \hat{\boldsymbol{f}}_M(\boldsymbol{x})\right|\right| \Big/ ||\boldsymbol{f}(\boldsymbol{x})|| \approx 3.15 \times 10^{-11}.$$

Figure 3.9 shows full, reduced simulation and the absolute error for $\mu = 0.04 \notin \Xi$ and DEIM approximation order $m = 12$.

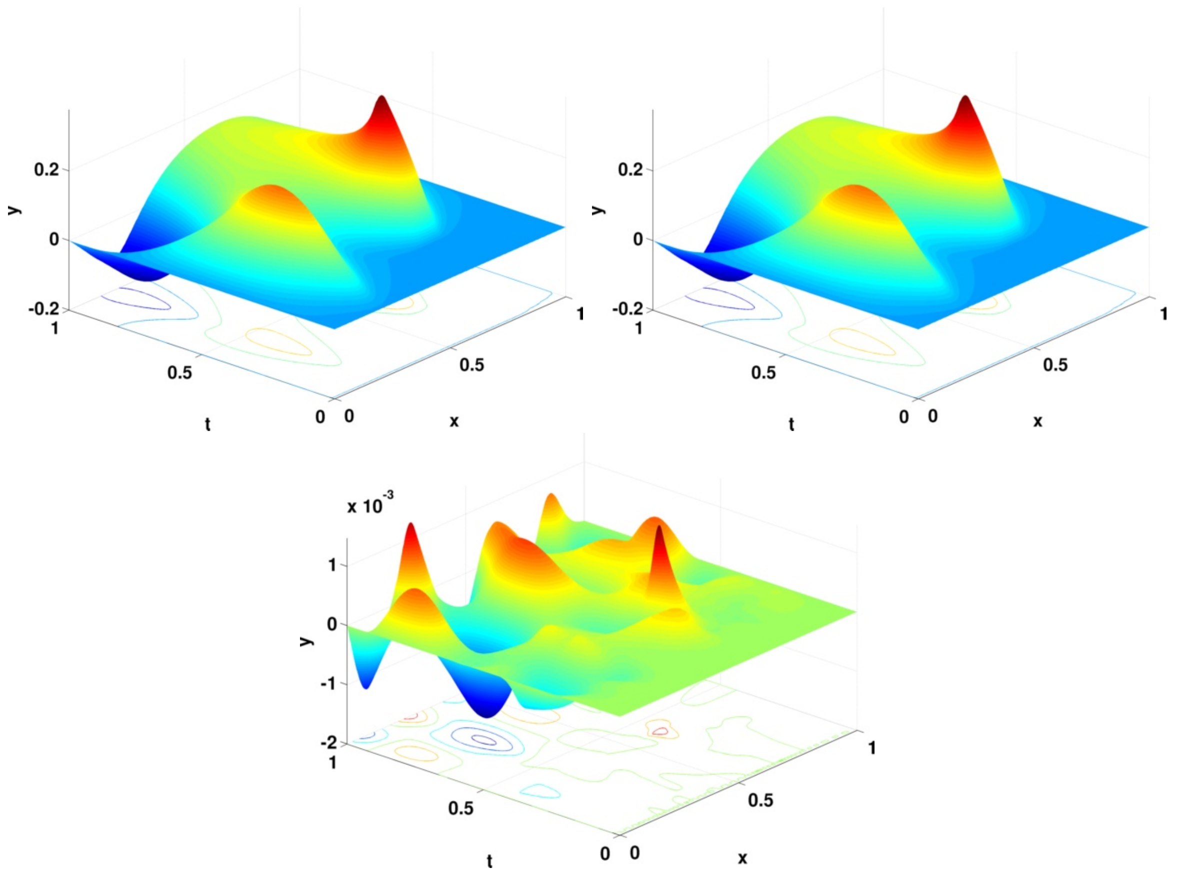


Figure 3.9: Full (left), reduced simulation (right) and the absolute error (bottom) for $\mu = 0.04$

For this setting, the reduced model already yields visually indistinguishable results with a maximum relative error of 0.0062.

Chapter 4

Model reduction of multiscale models

In this chapter we shall introduce a different concept of model reduction using kernel methods. In the previous chapters, we considered dynamical systems which were reduced by a combination of subspace projection and the nonlinear approximation techniques introduced in Chapter 2. However, outside the scope of nonlinear dynamical systems, some of the nonlinear approximation techniques can be applied to provide surrogate models in a completely different way. In detail, we shall consider multiscale models, where multiple phenomena and their interaction over different scales in space and/or time have to be considered. Frequently, during simulations, the involved intermediate- or small-scale models are used in a many-query context, where the larger-scale models require many evaluations of the latter ones for different configurations. In mechanics, a classic example for this situation is also known as ""*FEM*^{2}"" approach, where two models for different scales exist, each discretized by the finite element (FE) method. However, a common feature shared by most of those multiscale models is that the overall simulation time is unfeasibly high due to many and possibly slow evaluations of the involved microscale models.

At first, e.g. for the FEM^2 case, all the reduction techniques introduced in

Chapter 3 (including [6, 122, 127, 106, 4, 126, 5, 53, 74, 70, 44, 47]) can in principle be used to provide a fast surrogate microscale model. Related, the RB element method [112, 142, 113] has been developed to enhance reusability of reduced models for different geometric configurations. A different approach consists of the heterogeneous multiscale methods [48, 49, 2], which use different spatial resolutions to speedup full scale simulations, and the "Reduced Model Multiscale Method" [189] provides surrogate models by means of POD (See Section 3.1.1 or [176, 86]).

However, a second approach is to replace the microscale model by analytical functions, which is closely related to the "Response Surface Methodology" (RSM) [121, 16]. As in Chapter 2, the basic principle is to pre-select the functional form of the approximant and compute suitable coefficients, which is also the pursued approach in this chapter. Additionally, we will focus on the particular situation where the effective interface between the involved models is of moderate dimension, i.e. the models are connected via small number of DoFs. We would like to remark that this concept can also be readily applied to more involved models that vary over more than two scales, e.g. the construction of a surrogate of an intermediate-scale model can be facilitated by first creating a surrogate for a small-scale model used therein. For simplicity, we assume to have two models on a macro/micro scale, denoted by $\mathcal{C}/\mathcal{D}$ for $\mathcal{C}$oarse and $\mathcal{D}$etailed, respectively. The communication between $\mathcal{C}$ and $\mathcal{D}$ is given via an input space $\mathcal{P}_\mathcal{D} \subseteq \mathbb{R}^n$ (parameters) for $\mathcal{D}$ and an output space $\mathcal{P}_\mathcal{C} \subseteq \mathbb{R}^m$ (response) of $\mathcal{D}$, which is also used as input for $\mathcal{C}$. Figure 4.1 illustrates the considered model interaction, where the multiple instances of $\mathcal{D}$ show that, in general, many evaluations for $\boldsymbol{x}_i \in \mathcal{P}_\mathcal{D}$ are required as multiple $\boldsymbol{y}_i \in \mathcal{P}_\mathcal{C}$ are necessary for one simulation (step) of $\mathcal{C}$. Now, from

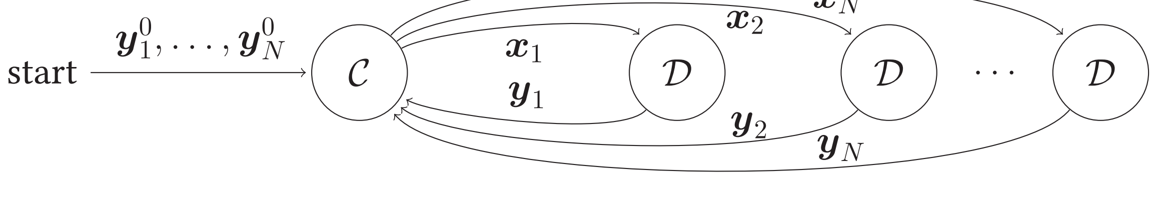


Figure 4.1: Communication diagram for macro and microscale model

the perspective of the macroscale model $\mathcal{C}$, we can consider a simulation of the small-scale model $\mathcal{D}$ as an evaluation of the (nonlinear) function

$$\begin{aligned} \boldsymbol{f} : \mathcal{P}_{\mathcal{D}} &\longrightarrow \mathcal{P}_{\mathcal{C}} \\ \boldsymbol{x} &\longmapsto \boldsymbol{f}(\boldsymbol{x}), \end{aligned}$$

where $\boldsymbol{f}(\boldsymbol{x})$ denotes a complete simulation of $\mathcal{D}$ using the input $\boldsymbol{x}$.

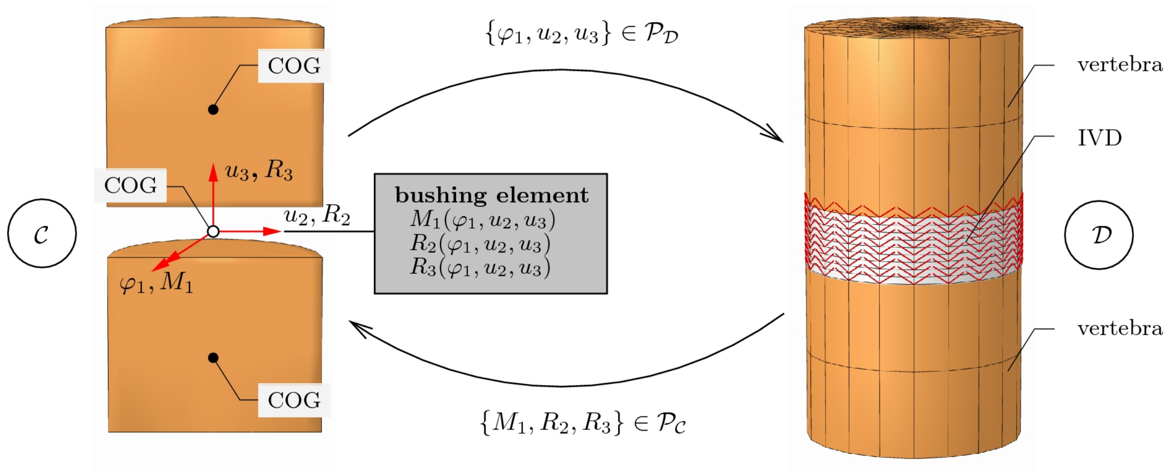
Any surrogate model now basically aims at providing an approximate $\hat{\boldsymbol{f}} \approx \boldsymbol{f}$, maintaining the response characteristics of $\mathcal{P}_{\mathcal{D}}$ as good as possible at ideally dramatically reduced evaluation cost. It is evident that a vast range of techniques can be used to provide such approximations, e.g. the methods introduced in Chapter 2 (including [25, 165, 152, 178, 14, 149, 29, 180, 123]). Another approach is the "Kriging"-interpolation [124] developed in the context of geographical information systems and recent approaches employ a mix of global sampling with local variational methods for reduced order modeling [1]. The RSM is also applied to approximate the configuration space in the context of the "Global Modal Parameterization" method [18]. For example, the PCE [180, 123] is used in the RSM context in e.g. [91] and current cooperative work is pursued in the context of skeletal muscle models [137, 138].

Here, we will investigate the possible contributions of kernel methods to the RSM. In detail, we shall apply the SVR algorithm from Section 2.1 and VKOGA algorithms from Section 2.2. We also demonstrate their successful application to two different real application scenarios. The first concerns a human spine multibody system with detailed intervertebral disk (IVD) responses in Section 4.1. Next, we will investigate a two-phase flow model describing saturation overshoot phenomena in porous media in Section 4.2.

4.1 Human spine multibody system

We will first introduce the considered model and establish the link to the multiscale setting with $\mathcal{C}/\mathcal{D}$. One current trend in computational biomechanics is to perform simulations using an overall human model. Following this, there are models available that are either based on a computationally cheap multi-body system (MBS), cf. [38, 39, 55, 56, 64, 154] among others, or that follow a continuum-mechanical modelling approach using finite elements as spatial discretization scheme [125, 84, 174, 82, 50, 158]. Combinations thereof exist, where scale-bridging and homogenisation techniques need to be applied to transfer continuum quantities like stresses and strains of the "micro" FE model $\mathcal{D}$ to integral forces $\mathcal{P}_{\mathcal{C}}$ and displacements $\mathcal{P}_{\mathcal{D}}$ used in the "macro" MBS $\mathcal{C}$ and vice versa, see Figure 4.1. In the context of the spine, the scale bridging has been applied using co-simulations [55, 56] or by performing pre-computations with an approximation of the mechanical response $\mathcal{P}_{\mathcal{C}}$ using surrogate models [91], which is the RSM approach investigated here using kernel methods. Following [91, 90], suitable surrogate models for a simplified setting of the IVD are based on a full cubic polynomial function, which is a constitutive assumption and thus only valid if the mechanical response $\mathcal{P}_{\mathcal{C}}$ follows the characteristics of a cubic polynomial. The same holds for the general kernel based approximation methods, however, choosing polynomial kernels readily allows to reproduce polynomial spaces, which makes this approach more general.

From a mechanical point of view, it is especially challenging to model the spine. It consists of 24 vertebrae each having an IVD in between as well as ligaments and muscles attached. In particular, during the definition of the spine in a MBS, the 3D structure of the IVD needs to be geometrically reduced to a single point, thereby accounting for its integral mechanical behaviour $\mathcal{P}_{\mathcal{C}}$ using a so-called bushing element, see Figure 4.3. Due to the inhomogeneous fibre-reinforced microstructure of the IVD, this reduction is an even more challenging task, as it leads to highly non-linear and coupled characteristics of the integral response $\mathcal{P}_{\mathcal{C}}$. To account for these require-



$\{\varphi_1, u_2, u_3\} \in \mathcal{P}_\mathcal{D}$
COG
COG
COG
u_3, R_3
u_2, R_2
φ_1, M_1
bushing element
$M_1(\varphi_1, u_2, u_3)$
$R_2(\varphi_1, u_2, u_3)$
$R_3(\varphi_1, u_2, u_3)$
$\mathcal{C}$
$\mathcal{D}$
vertebra
IVD
vertebra
$\{M_1, R_2, R_3\} \in \mathcal{P}_\mathcal{C}$

Figure 4.2: "Macroscopic" MBS and "Microscopic" FE model of a simplified motion segment of the human spine restricted to in-plane movements including two rigid cylindrical vertebrae and an idealised IVD as a three-degree-of-freedom bushing element. Image: [91]

ments, a homogenisation scheme is applied to transfer kinematical as well as mechanical reaction quantities between the MBS and the FE model. In a first step, the DoF $\mathcal{P}_\mathcal{D} = \{\varphi_i, u_i\}$ of the MBS, i. e., the integral rotations φ_i as well as the displacements u_i for $i = 1, 2, 3$, are converted to boundary conditions in the FE model according to the schematic drawing of Figure 4.3a). Herein, the discrete rotations φ_i of the bushing element, located at the center of gravity (COG) of the IVD, are parameterized with respect to the IVD's top surface $\mathrm{d}a$. This in turn yields a conversion into nodal displacements $\bar{\mathbf{u}}_S$, i. e., Dirichlet boundary conditions of the discretized FE model. Following

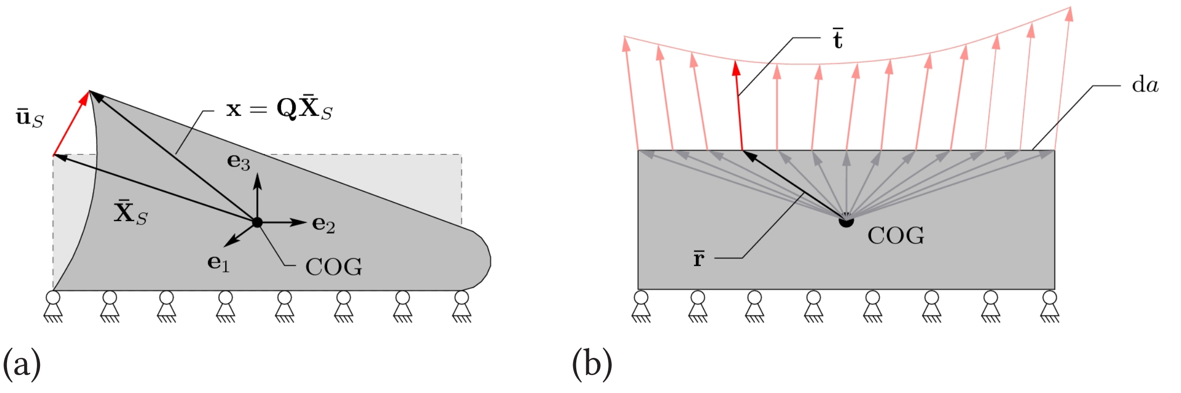


Figure 4.3: Schematic drawing of a sagittally cut IVD with a) the parameterization of the Dirichlet boundary conditions for the FE model and b) the resulting surface traction vector $\bar{\mathbf{t}}$ as well as the corresponding lever arm $\bar{\mathbf{r}}$ to the COG, which is needed for the integral computation of the moment over the top surface $\mathrm{d}a$. Image: [91]

[91], the kinematics of the simplified cylindrical IVD is reduced to motion

in the sagittal plane, cf. the MBS in Figure 4.2. Thus, the needed parameterization of the discrete angle φ_1 as well as the discrete displacements u_2 and u_3 of the MBS is given by

$$\begin{aligned}\bar{\mathbf{u}}_S = &\underbrace{\left[u_2 + (\cos\varphi_1 - 1)\,\bar{X}_2 - \sin\varphi_1\,\bar{X}_3 \right]}_{\bar{u}_{S2}} \mathbf{e}_2 \\ &+ \underbrace{\left[u_3 + \sin\varphi_1\,\bar{X}_2 + (\cos\varphi_1 - 1)\,\bar{X}_3 \right]}_{\bar{u}_{S3}} \mathbf{e}_3 \,.\end{aligned}$$

After a deformation state of the MBS is applied to the IVD, its mechanical deformation behaviour, i.e. its reaction stress, is computed using the FE model of [89, 88]. Herein, the integral behaviour $\mathcal{P}_\mathcal{C}$ of the IVD is captured by applying a homogenisation procedure to the inhomogeneously distributed surface traction vector $\bar{\mathbf{t}}$, which is the reaction stress of the IVD due to the applied deformation $\bar{\mathbf{u}}_S$, cf. Figure 4.3b). Following [91], the discrete mechanical reaction of the IVD in terms of a force $\mathbf{R} = R_i\,\mathbf{e}_i$ and a moment $\mathbf{M} = M_i\,\mathbf{e}_i$ is computed via an integration of the surface traction vector $\bar{\mathbf{t}}$, viz.:

$$\mathbf{R} = \int\limits_{\partial\Omega} \bar{\mathbf{t}}\,\mathrm{d}a \quad \text{and} \quad \mathbf{M} = \int\limits_{\partial\Omega} \bar{\mathbf{r}} \times \bar{\mathbf{t}}\,\mathrm{d}a\,. \tag{4.1}$$

Finally, the relationship between the DOF of the MBS as input quantities $\mathcal{P}_\mathcal{D}$ and the integral mechanical response $\mathcal{P}_\mathcal{C}$ resulting from the homogenised FE model (4.1) as output quantity writes as

$$\begin{aligned}\boldsymbol{f} : \mathcal{P}_\mathcal{D} &\longrightarrow \mathcal{P}_\mathcal{C} \\ (\varphi_1, u_2, u_3) &\longmapsto (M_1, R_2, R_3).\end{aligned}$$

4.1.1 Experiments

The training data for the pursued experiments has been obtained running $n = 2473$ simulations of the IVD with different parameters, sampled starting from zero and running towards the corners of the parameter domain (due to quasi-static simulations). As comparison we use the results from the Polynomial Chaos Expansion (PCE) approximation [180, 123], which has already been applied in this context in [91]. We ran the VKOGA Algorithm 6 with both the f and f/P selection criteria for different kernels and hyperparameters. For the Gaussian we set a γ range of $[15, 70]$, of which we chose 150 linearly spaced values. The Wendland kernel has been used with 90 configurations in total, choosing 30 linearly spaced hyperparameters $\gamma \in [15, 70]$, each in combination with smoothness's $k \in \{1, 2, 3\}$. Naturally for this setting, the dimension $d = 3$ was chosen for the Wendland kernel. The maximum expansion size was fixed to $M = 600$ for both, along with a maximum absolute and relative error of 10^{-5} on the training data as stopping criteria, where the relative error is computed via

$$e_{rel} = \max_{x \in X} \left|\left| f(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x}) \right|\right|_2 \Big/ ||f(\boldsymbol{x})||_2 \,.$$

In the case of the ϵ-SVR, we fixed $\epsilon = 0.013375$ and preliminary experiments showed that the resulting approximation errors were quite sensitive to the combination of hyperparameters γ and regularization factors λ. Consequently, we decided to use an extensive set of configurations and chose 25 linearly spaced samples for both λ and ϵ from $[10^{-6}, 10^{-3}]$ and $[10^{-3}, 0.1]$, respectively. Additionally, we chose to take 25 linearly spaced γ values from $[10, 35]$ as Gaussian hyperparameters, yielding a total of 15625 different configurations. For each, a maximum iteration count of 60000 and $\delta = 0.0001$ were fixed as stopping conditions.

Table 4.1 shows details for the approximation results, including run-times and various error measures for the best configurations, respectively. Among all configurations and all algorithms, the best configuration corresponds to the minimum maximal absolute L^∞ error (L^2 point-wise) over X. The extensive computation time for the ϵ-SVR is due to the large amount of different configurations. In average, one ϵ-SVR solve took $108s$ (which would come up to e.g $3h$ for 100 configurations), so the SVR is yet more expensive to compute than the VKOGA variants. Note, however, in [162] a C/C++

Kernel type:	Wendland kernel K_w		Gauss kernel K_g		
Algorithm:	f-VKOGA	f/P-VKOGA	f-VKOGA	f/P-VKOGA	SVR
Configurations	90	90	150	150	15625
Time	168s	70.1s	9.1s	1.53s	468h
Best configuration	γ=37.7586, k=1, d=3	γ=30.1724, k=1, d=3	$\gamma = 16.4765$	γ=15	γ=10, λ=0.00075
Expansion size K	600	586	120	1	876
Max. L^2 error	4.80567	33.4541	79.3389	37511.3	1224.26
Max. relative L^2 error	5.42474	14.0741	16.0223	41930	479.266

Table 4.1: Comparison of approximation results for different kernels and modes.

implementation is presented that is shown to outperform LibSVM in some cases.

Moreover, the VKOGA algorithms using the Gaussian kernel stopped early, indicating a wrong choice of kernel space. If the correct space (i.e. kernel hyperparameter) is chosen, it can be expected that errors in the vicinity of already included points are comparably small. If not, subsequent iterations cause many closely neighboured points to be included in order to “fix” the local errors. This effect is even more severe using the f/P-VKOGA variant as the weighting by the orthogonal remainder norm $\|\tilde{\phi}_{\boldsymbol{x}}\|_{\mathcal{H}}$ penalizes such errors even more (see Section 2.2.2.1). Numerically, this causes the following: While the expression $\sqrt{z_{i_{max}} - p_{i_{max}}}$ in Algorithm 6 (page 76) is analytically always positive (see (2.40, p.74)), it gets close to zero very fast for closely neighbored centers. Now, as the value $p_{i_{max}}$ is successively accumulated over the iterations, values slightly greater than one may occur, leading to complex roots and hence an emergency stop of the algorithm. Again, this effect is especially inconvenient using the f/P-Greedy variant, since all the residuals have to be weighted before selecting the next center, cf. lines 22-23 of Algorithm 6. Of course, it is possible to use $\sqrt{\max\{eps, z_{i_{max}} - p_{i_{max}}\}}$, but experiments showed that no real improvements are made once the first values $p_{i_{max}} > 1$ appear. Now, due to the hierarchical structure of the VKOGA algorithm, is is easy to select the number of points that yielded the best error at a possibly intermediate algorithm step. This also explains the small number of centers, as 120 points or just one sample have been found to yield the smallest overall error until the numerical issues occurred. Even though the described effect is not as severe for the f/P-VKOGA variant using the Wendland kernel, yet both f/P-variants using either Gaussian or Wendland kernels are outperformed by their f-VKOGA counterparts regarding the maximum errors. Hence, we will narrow down the further discussion to those variants and shall also include the PCE approximation from [91] for comparison.

Now, Figure 4.4 and Table 4.2 provide more insight to the approximation quality of the selected algorithms. Figure 4.4 which shows the absolute and relative approximation errors on X (L^2 in $\mathcal{P}_\mathcal{D}$) in the upper and lower row, respectively. Table 4.2 additionally states the mean and median errors for the respective methods, corresponding to the red and green lines in the plots of Figure 4.4. Most importantly, we sorted the data points in Figure

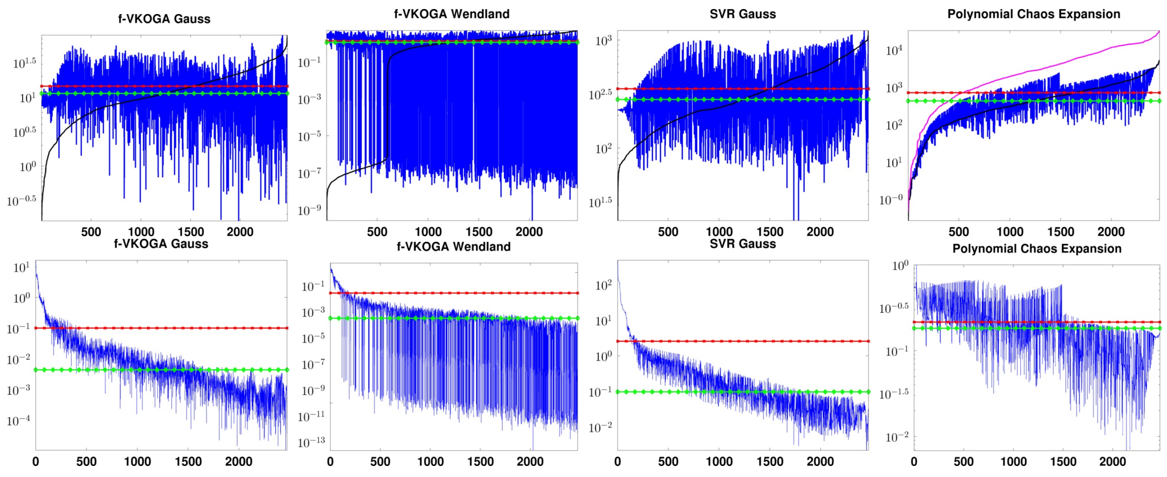


Figure 4.4: Plots of absolute and relative approximation errors (top/bottom rows) on X for different approximation methods, where the training points of both sets are sorted ascending by norm of Y entries (added in magenta in top right plot). The red/green lines are the mean/median and the black lines in the upper row denote the respective absolute errors sorted ascending by value.

4.4 by the norm of the corresponding Y-values in ascending order, which are also added in magenta in the top right image (absolute PCE errors). The solid black lines in the upper row images show the same absolute errors but sorted by value. It becomes clear that all large relative errors occur at close-to zero function values, while the relative errors are decreasing for higher Y

values for each method. Hence, the maximum relative errors are less suitable as quality measure than mean or median errors in this case. Even more, considering the median error of 0.09671 for the ϵ-SVR, it does not only outperform the PCE approximation w.r.t. the same measure, but shows that the result is not as bad as suggested by the mean relative error. Interestingly, the PCE is very precise at the smallest values of Y, where on the other side the SVR solution yields a somewhat constantly higher error level also for close-to-zero training data. With respect to the maximum absolute error,

	Absolute errors			Relative errors		
	max	mean	median	max	mean	median
f-VKOGA, Gaussian	79.3389	14.4025	11.2859	16.0223	0.10017	0.00436
f-VKOGA, Wendland	4.80567	1.33274	1.10283	5.42474	0.02711	0.00031
ϵ-SVR, Gaussian	1224.26	353.849	281.309	479.266	2.55756	0.09671
PCE	5240.14	705.049	424.474	1.00000	0.21536	0.18245

Table 4.2: Approximation error overview for selected algorithms and configurations

all approximation methods outperform the PCE approximation, where the VKOGA results are yet about two orders of magnitude better than the SVR result. While the relative errors of the SVR are worse than the PCE, both VKOGA approximations are also better in this case. It clearly shows the superiority of the f-VKOGA-approximation using Wendland kernels, which only has a mean relative error of 2% on X compared to 21% of the PCE and matches at least three digits on half of X.

Analytically, the VKOGA solutions should have zero error at each of the 600 support vectors. This is visually confirmed for the Wendland kernel case, where the black line of sorted absolute errors is around 10^{-7} for the first 600 points. This effect is less visible for the Gaussian approximation, which indicates that the numerical stability becomes an issue during evaluation of the expansion. This is confirmed considering the resulting expansion coefficients in the direct translate basis: While the Wendland kernel expansion has an average coefficient vector norm $\frac{1}{K}\sum_{i=1}^{K}||\boldsymbol{c}_i|| = 3.83 \times 10^6$ with entry values (over all coefficient vectors) in the range $[-1.97 \times 10^8, 1.97 \times 10^8]$, the same values are 1.03×10^{15} and $[-7.07 \times 10^{15}, 5.46 \times 10^{15}]$ for the Gaussian version. As the interpolation conditions do not apply for the SVR algorithm, the numerical condition is even better considering values of 4.77×10^5 as mean coefficient vector norm and entry values ranging in $[-2.13 \times 10^7, 2.13 \times 10^7] = [-C, C]$ with $C := \frac{1}{2\lambda n}$, cf. Proposition 2.1.2.

Next, Figure 4.5 shows different response surfaces for the f-VKOGA approximation using Wendland kernels. Shown are the responses for $(u_2, \varphi_1) \to$

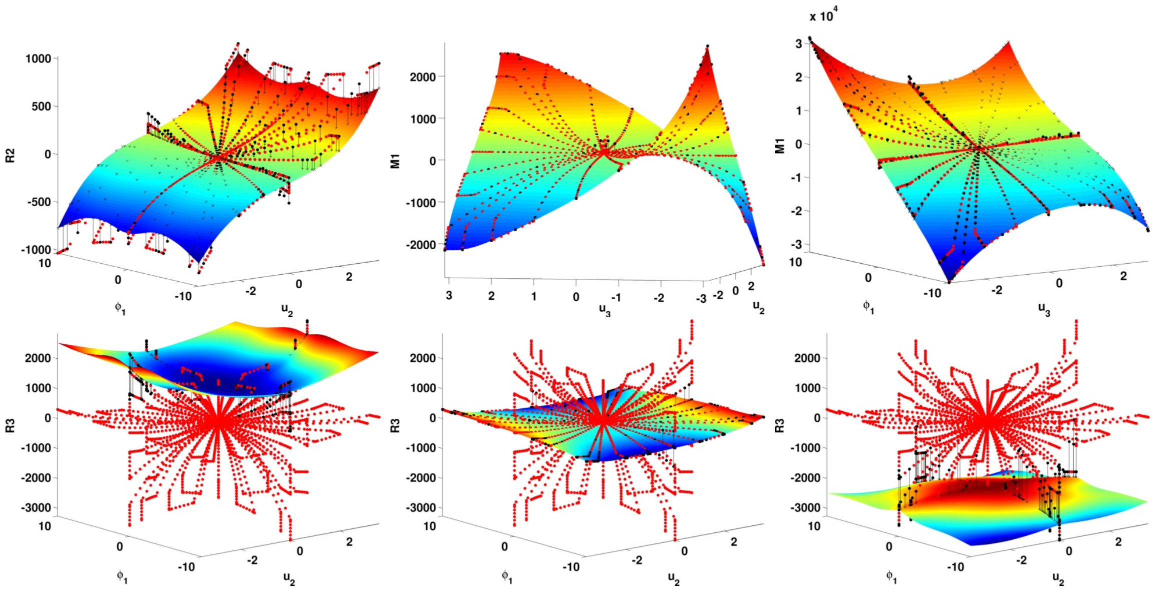


Figure 4.5: Plots of approximation using Wendland kernels and best configuration $\gamma : 37.7586, k : 1, d : 3$. *Top:* $(u_2, \varphi_1) \to R_2$, $(u_2, u_3) \to M_1$ *and* $(u_3, \varphi_1) \to M_1$ *for each a fixed third/leftover input dimension value. Bottom:* $(u_2, \varphi_1) \to R_3$ *for different* u_3.

R_2, $(u_2, u_3) \to M_1$ and $(u_3, \varphi_1) \to M_1$ for each a fixed third/leftover input dimension value. The lower row shows the approximations for $(u_2, \varphi_1) \to R_3$, with the third argument u_3 chosen over three increasing values from the training data range of u_3. The red dots are the projected locations of Y, and the black dots show the projections of function values on the selected centers along with a black line drawn to the actual approximated value on the respective center. Note here that only in the lower row we plotted all training function values of the selected component. The other plots show only a selection of (center-) values, whose corresponding data points are in the vicinity of the currently selected third dimension values. The top row illustrates the range of different nonlinear response characteristics, which are all successfully covered by the considered approximation. The clusters of black markers around the response surface in each image of the lower row show that the actual process to approximate is continuous in the sense that data points with similar u_3 values lead to local areas of function values. Note that this does not imply “locality” of the kernel expansion; taking a closer look into the expansion details reveals quite the opposite. The bounding box of X has a diameter of approximately 22, while the kernel width of $\gamma = 37.7586$ for the best solution is almost double that size, cf. Table 4.1, which clearly shows that each center of the expansion contributes to every function value.

Finally, we shall look into the speedup gained using the surrogate models. So far, there have been no directly coupled full-scale simulations of the MBS yet, and the application of the PCE in [91] made the MBS simulation possible in the first place. Consequently, the proven applicability of the PCE together with the improved performance of the kernel methods on the training data shows that a significant improvement of the simulation quality of the MBS can be achieved. Nevertheless, we shortly derive the expected speedup using kernel surrogate models. Evaluating the three dimensional kernel expansion with 600 centers averages to $4 \times 10^{-4}s$ (in MatLab). The present full IVD model implementation does not have a constant run-time due to quasi-static simulations, each starting from zero input. As the different target configurations X take different overall computation times, we consider the average of about 70 minutes taking 24 quasi-time steps, so about three minutes per single step. As “ideal” case (in terms of run-time), we assume each a discrete time-step in the macromodel requires only one quasi-time step in the IVD model, which already yields a speedup factor of 18×10^5. Considering realistic simulations of 24 disks in a full MBS system over 30 seconds with time step of $\Delta t = 0.001s$, we obtain $24 * 30 * 1000 = 720000$ queries to $\boldsymbol{f}$. In the above setting this would result in $600h$ evaluation time of the full IVD model responses compared to $4.8min$ for the surrogate model evaluations.

4.2 Saturation overshoots in porous media

The second model we consider in our experiments stems from the work [94, 93], which is an extension of [95] to the multidimensional case. Therein, a two-phase flow problem in porous media on a two-dimensional domain is investigated, more specifically, the infiltration of a wetting fluid into a homogeneous porous medium which is initially filled by a non-wetting fluid, where both fluids are assumed immiscible. The model attempts to simulate overshooting saturation waves, which have been experimentally observed as peaks close to the head of the infiltrating fluid front [43]. The solution strategy is to apply the heterogeneous multiscale method involving a macro- and microscale model. The macroscale model behavior is governed by a hyperbolic conservation law and, neglecting gravitational effects, can be described by the two-phase fractional flow formulation

$$\begin{aligned} S_t + \operatorname{div}(\boldsymbol{v} f(S)) &= 0, \\ \boldsymbol{v} &= -\boldsymbol{K}\lambda(S)\nabla p, \\ \operatorname{div} \boldsymbol{v} &= 0, \end{aligned} \tag{4.2}$$

with suitable boundary and initial conditions. The unknowns are wetting saturation $S(\boldsymbol{x}, t) \in [0, 1]$, velocity field $\boldsymbol{v}(\boldsymbol{x}, t) \in \mathbb{R}^2$ and pressure $p(\boldsymbol{x}, t) \in \mathbb{R}$. Further we have the flux function f and total mobility λ given by

$$\begin{aligned} f(S) &= \frac{\lambda^w(S)}{\lambda(S)}, & \lambda(S) &= \lambda^w(S) + \lambda^{nw}(S), \\ \lambda^w(S) &= S^2, & \lambda^{nw}(S) &= (1-S)^2, \end{aligned}$$

where $\lambda^{nw/w} : [0, 1] \to \mathbb{R}$ describe the single fluid mobilities for wetting and non-wetting phases, respectively. Other quantities like the permeability matrix $\boldsymbol{K} \in \mathbb{R}^{2\times 2}$ are given suitably.

This model allows to produce the observed overshoot effects as the solutions may contain nonclassical shock waves [94, 102]. Unfortunately, besides lack of a unique solution, the equations (4.2) alone do not produce the observed overshoots as microscale information is missing. Hence, a regularized model is considered that takes a dynamic rate-dependent capillary pressure into account, see [94, 93] and references therein. However, the resulting computational efforts would be unfeasibly large if the resulting model was solved globally. Instead, a further simplification consists in solving the regularized model only in one dimension and in normal direction at the wave front, as the overshoot effects/saturation values are of interest only at those locations in order to update the local velocity field. Consequently, this results in a "microscale" detailed, but one-dimensional problem given as

$$\partial_t s + \partial_x \left(f(s)v \right) = -\epsilon \partial_x \left(f(s)\lambda^{nw}(s)\partial_x p_c(s) \right) + \tau\epsilon^2 \partial_x \left(f(s)\lambda^{nw}(s)\partial_x \partial_t s \right), \tag{4.3}$$

with unknown scalar saturation $s(x,t) : \mathbb{R} \times [0,T] \to \mathbb{R}$ and regularization parameter $\epsilon > 0$. Further parameters are the rate parameter $\tau > 0$, which determines the height of the overshoot, the static capillary pressure p_c and the velocity $v \in \mathbb{R}$. Note here that the velocity does not have any effect on the overshoot in which we are interested (no gravity is considered in the model), and thus a value of $v = 1$ is assumed.

The initial conditions are set to $s(x \le 0, 0) = s_l$, $s(x > 0, 0) = s_r$, consisting of the current saturations of the macroscale problem at the phase boundary, where s_l/s_r stem from the wetting and non-wetting side of the wave front. Recall that $s_r = 0$ due to the assumption of having the porous medium initially filled with non-wetting fluid. Finally, to obtain the overshooting "plateau" value, the regularized model is simulated until the overshooting area has developed sufficiently and $\overline{s}$ can be extracted. Then, $\overline{s}$ is the missing microscale information which is fed back into the macroscale model using a specific flux function, cf. [94, §6.1] for details.

Consequently, we identify (4.2)/(4.3) with $\mathcal{C}/\mathcal{D}$, respectively. Setting $\mathcal{P}_{\mathcal{C}} =$

$\mathbb{R}^+$ and $\mathcal{P}_\mathcal{D} = \mathbb{R}^+ \times \mathbb{R}^+$, the interaction function reads as

$$\begin{aligned} f : \mathcal{P}_\mathcal{D} &\longrightarrow \mathcal{P}_\mathcal{C} \\ (s_l, \tau) &\longmapsto \overline{s}. \end{aligned}$$

4.2.1 Experiments

For the experiments the values ϵ and $s_r = 0$ have been fixed for simplicity. To this end, it is clear that the proposed technique can also deal with situations where the corresponding values are considered a possibly changing quantity. Next, we generated training data $Y = ((s_{l_1}, \tau_1), \dots, (s_{l_N}, \tau_N))$ for $N = 5000$ settings with corresponding solutions of the regularized problem $X = (\overline{s}_1, \dots, \overline{s}_N)$. Since the s_l values are linearly spaced within $[0, 1]$ but the τ values range in $[0, 10^7]$, we additionally performed a logarithmic transformation and re-scaling of the τ values to $[0, 1]$. The total computation time for Y, X was $404h$.

For the ϵ-SVR (Algorithm 1 in 2D SMO variant) we chose a Gaussian kernel with 25 linearly spaced γ values in $[0.0833, 0.2498]$, which corresponds to a decay of the Gaussian below machine precision for radial distances between $[0.5, 1.5]$. We chose 25 linearly spaced λ values from $[0.005, 1]$ along with a constant $\epsilon = 0.001$. As stopping criteria, we fixed a maximum iteration count of 50000 and chose $\delta = 0.0001$. The VKOGA (Algorithm 6) has been run using the Wendland kernel for 90 different kernel configurations. We used $d = 1$ and chose $k \in \{1, 2, 3\}$ combined with 30 linearly spaced dilation hyperparameters $\gamma \in [2, 3]$. With the Gaussian, we ran the algorithm for 100 different linearly spaced $\gamma \in [0.05, 0.16]$. Both the f and f/P-Greedy variants have been considered and the termination criterion was a maximum of 1000 centers or a maximum point-wise absolute error less than 0.001 on X.

At first we will focus on the approximation results and consider the embedding into the original problem later. Table 4.3 summarizes the algorithm results and Figures 4.6 and 4.7 show the training data and approximations for the best configurations, respectively. The relative error is computed via

Kernel type:	Wendland kernel K_w		Gauss kernel K_g		
Algorithm:	f-VKOGA	f/P-VKOGA	f-VKOGA	f/P-VKOGA	SVR
Time	$494s$	$838s$	$867s$	$44.3s$	$7.4h$
Best configuration	γ=2.9655, k=2, d=1	γ=2.0689, k=1, d=1	γ=0.0523	γ=0.0853	γ=0.0832, λ=0.0879
Expansion size	709	928	417	15	1362
Max. L^2 error	0.0972	0.000616	0.288	0.636	0.208
Max. relative L^2 error	0.0434	0.0583	2.37	1.41	0.389

Table 4.3: Comparison of approximation results for different kernels and algorithms

$\max_{x \in X} \left|\left| f(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x}) \right|\right|_2 \Big/ ||f(\boldsymbol{x})||_2$. At first, the computation time for any approximation is negligible compared to the time needed to obtain the training data in the first place. Both variants using the Wendland kernels find a configuration which satisfies the absolute error bounds before the maximum number of centers is reached. All configurations for the Gaussians do not reach the desired absolute error. The f-Greedy stops at the limit of 1000 centers and the f/P-Greedy due to numerical instability during Newton basis computation. However, for both cases the absolute errors at each extension step are available, and due to the incremental nature of the Newton basis one can simply take the first centers up to the position of the effective minimal absolute error, compare the experiments from Section 4.1.1. This results in seemingly "incomplete" approximations with Gaussians having only 417 and 15 centers, however, this is the best one can do for the given "bad" configuration set and error measure. Figure 4.6 shows the training data on the left and the VKOGA f-Greedy approximation using Wendland kernels along with a top view, where the expansion centers are marked by black dots. The overall function is very well approximated and the centers are mainly concentrated around the discontinuity and edges. Some peaks of oscillation can be seen in the top view around locations without sufficient training data density. Next, Figure 4.7 shows further selected results from Table 4.3. Regarding the VKOGA results using Wendland kernels, it is interesting to see that the selected centers for the f-Greedy variant in Figure 4.6 are more

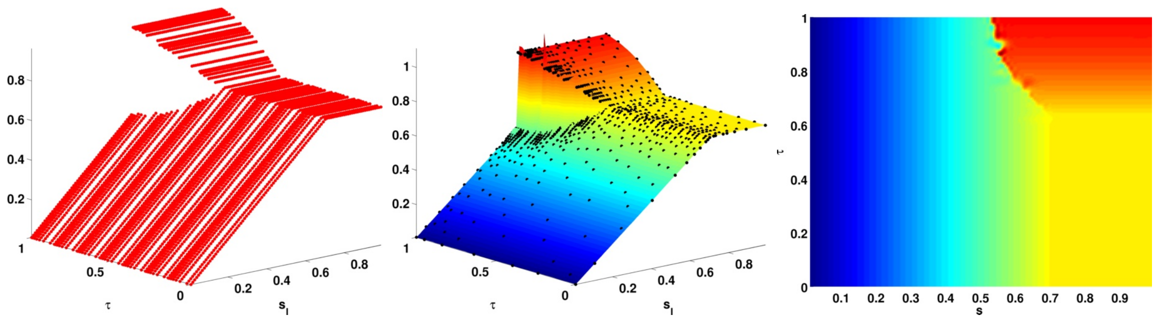


Figure 4.6: Left: Training data, center: Wendland f-Greedy approximation with centers, right: $X - Y$ view

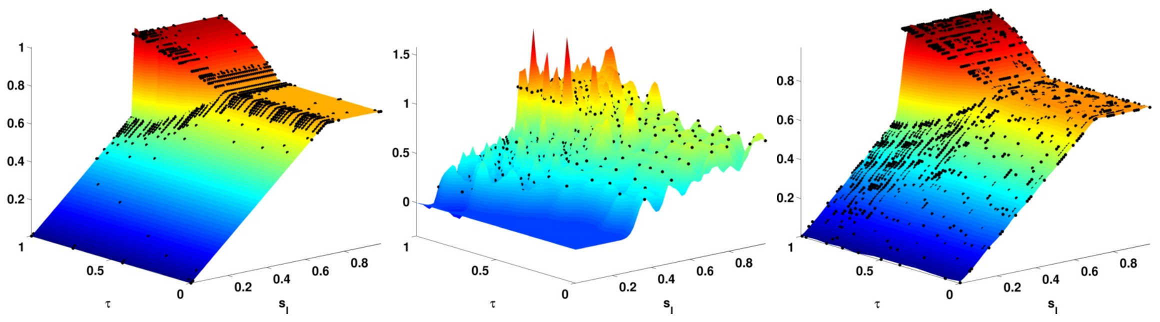


Figure 4.7: Best results for different algorithms. Left to right: f/P Greedy with Wendland kernels, f-Greedy with Gaussians, ϵ-SVR with Gaussian.

equally distributed than the ones selected by the f/P-Greedy counterpart in Figure 4.7. In both cases, they are more dense around the discontinuity and the "edge" at $s_l \approx 0.7$, but in the case of f/P-Greedy the centers are almost exclusively located there. This is due to the subspace-weighted point selection criteria (2.24, p.58), which penalizes errors closer to already included data sites. In a way, an accumulation of centers indicates that the function on the respective area is not well-contained in the considered RKHS; for more details see Section 2.2.2.1 or [183]. This effect is even more obvious considering the center plot of Figure 4.7: This f-Greedy approximation is an example, where the typical "initial guess" of using Gaussians as kernels is suboptimal. A possible explanation for the poor performance is surely motivated by the following arguments. The "true" function to approximate is not even continuous, but the functions from the Gaussian-induced RKHS are C^∞. Thus, the C^∞ approximation is numerically infeasible to compute and comes with strong oscillations around the discontinuity. On the other side, the Wendland kernels (and especially best configurations) use kernels

with smoothness's $k = 2$ or $k = 1$, which is much closer to the true nature of the function to approximate.
Furthermore, the SVR approximation needs more centers than the other approaches and a considerably higher overall computation time, but the small peaks occurring for the interpolation-based approaches are not observed here. On the downside, as clearly visible in the rightmost plot, the discontinuity is resolved less accurate. Thus, the final choice of algorithm will depend on the desired qualities of the approximation and their impact on the overall simulation.

Now we will detail the macromodel simulation errors for approximations from Table 4.3, where we shall restrict ourselves to the f-Greedy variants using both kernel types and the SVM result. Figure 4.8 shows the macromodel simulation results using those micromodel approximations. While visually

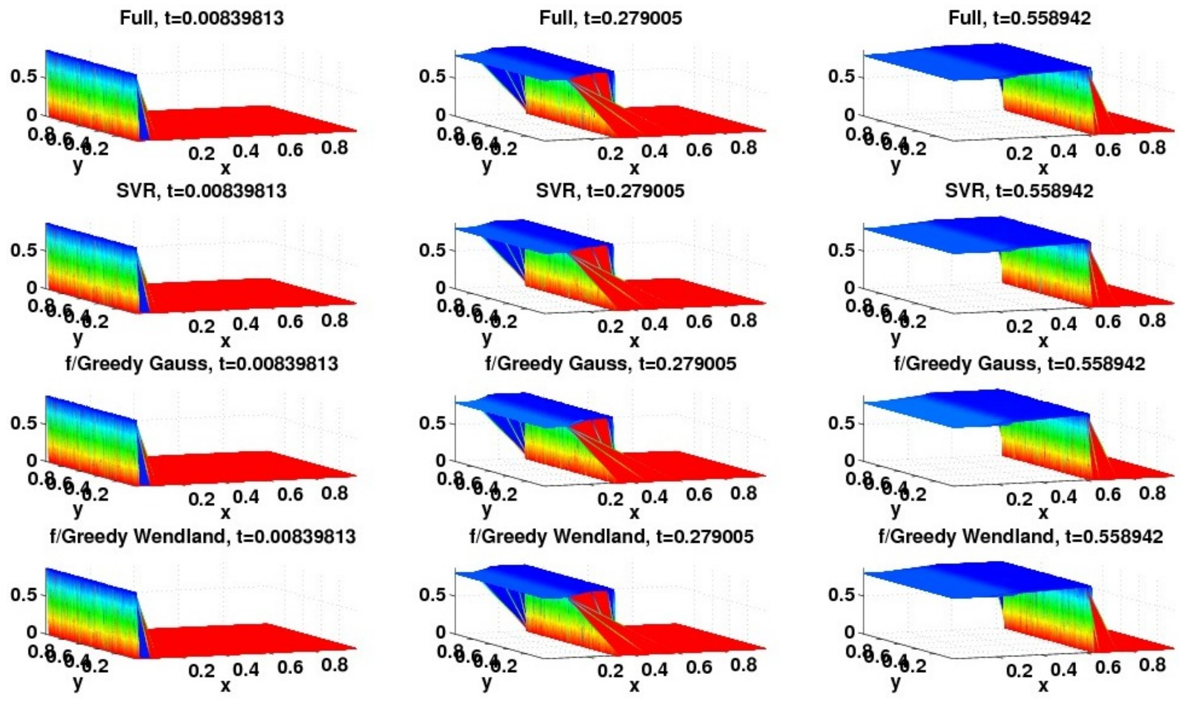


Figure 4.8: 3D view for simulations using full microscale model, SVR and f-Greedy approximations using Gaussian/Wendland kernels.

all reasonable, they are of yet different quality. As the plots are hard to distinguish, Table 4.4 shows the respective errors for discrete L^2, L^1-measures in space and max/mean over simulation time. While the results using the SVR or Gaussian f-Greedy kernel expansions are comparable, both are out-

	SVR		f-Greedy (Gauss)		f-Greedy (Wendland)	
	absolute	relative	absolute	relative	absolute	relative
Max L^2	10.9913	0.09726	15.1986	0.12804	2.22058	0.03928
Mean L^2	6.75654	0.08212	8.19281	0.09686	1.14767	0.01509
Max L^1	205.170	0.04194	585.554	0.05852	35.4055	0.01393
Mean L^1	111.437	0.01519	267.679	0.03193	19.4393	0.00298

Table 4.4: Macroscale model errors for different microscale surrogates.

performed by the results using the Wendland-kernel f-Greedy approximation. With only a maximum relative L^1-error of about 1% over the entire state and time domain a remarkably good reduced macroscale simulation is achieved. Regarding speedup, the evaluation time for a full-scale single microscale model (4.3, p.130) is $0.1s$, and $1.2 \cdot 10^{-4}s$ for the SVR approximation using 1384 centers. The full macroscale problem (4.2, p.129) takes 20h 40min to simulate, but only 24min using the SVR surrogate model. Hence, even for the "most expensive" kernel approximation with 1384 centers this makes up a speedup factor of ≈ 51, which is of course higher for approximations with less centers.

In order to investigate the reduction error behaviour over time, Figure 4.9 shows the simulation errors using L^2 in state space. Both the f-Greedy and SVR approximation using Gaussians yield a continuously increasing simulation error compared to the full solution. This effect does not occur with the f-Greedy approximation using Wendland kernels, which additionally gives a worst-case absolute error of about an order of magnitude smaller than the others. Also the maximum absolute L^2 error stays below 4% with an average of about 1%, cf. Table 4.4. Finally, Figure 4.10 displays the logarithmic absolute errors using the usual approximations at $t = 0.6$ for the subsection $[0.2, 0.65]$ of the time domain. Here, the f-Greedy and SVR approximation using Gaussians both have an absolute error of magnitude one at the wavefront, where the f-Greedy approximation is about an order of magnitude better. In this case the f-Greedy Gaussian approximation has a higher, but more even mean error than both other solutions. Even though the SVM and

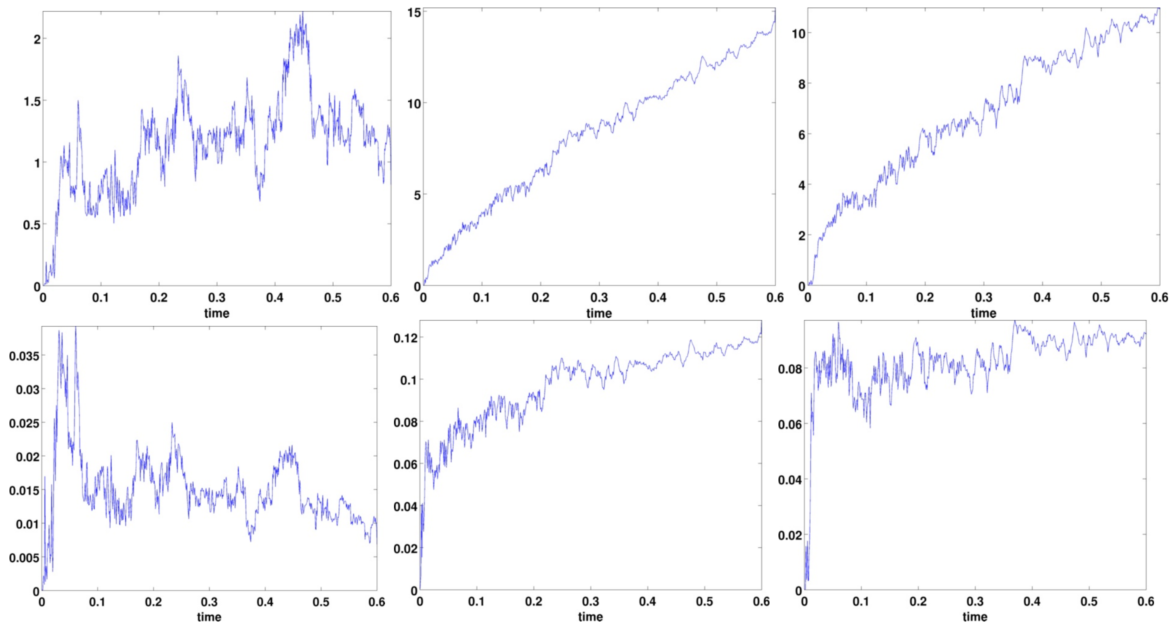


Figure 4.9: Macromodel absolute (top) and relative (bottom) errors for f-Greedy with Wendland (left) or Gaussian (middle) kernel and SVR (right) approximations.

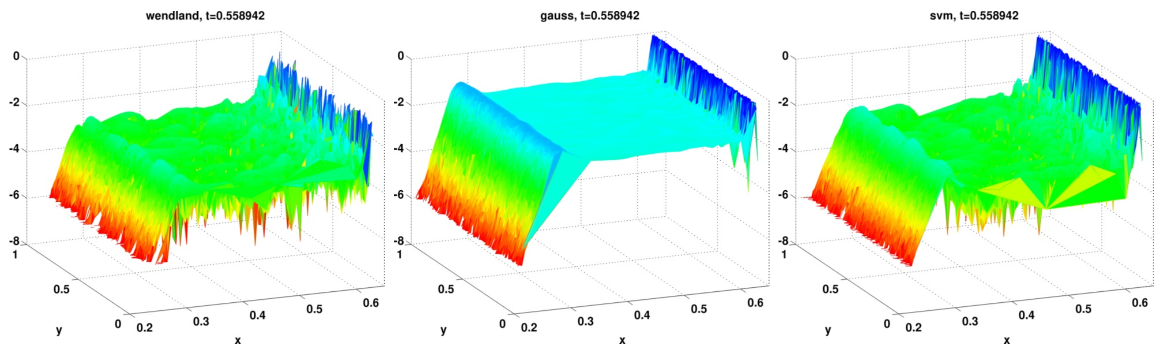


Figure 4.10: Macromodel logarithmic absolute state space errors at $t = 0.6$ for f-Greedy with Wendland (left), Gaussian (middle) kernel and SVR (right) approximations.

Wendland-kernel solutions look similar, the main difference is the more correct position of the second (slower) wavefront, which has advanced visibly further in the SVM case and causes the larger error.

Chapter 5

A-posteriori error estimation

One of the most important prerequisites of successful model reduction approaches is to provide error estimates for the reduction process. Without that, there is little use for reduced models in most practical situations, as there is no way to tell if the results obtained from a reduced simulation are anywhere close to the results using the full model. Ideally, those estimates also have to be *fast and rigorous*. Their efficient computation is essential, as the whole point of reduced models is increased simulation speed. Clearly, error estimation procedures that last about as long as the time needed for full simulations are of little help. The rigorosity (i.e. a provable upper bound) is not a complete necessity in all situations, e.g. when applied in the POD-Greedy procedure in Section 3.1.2, but is of course essential for predictive applications or control scenarios.

However, as computing the true reduction error equals the cost of a full model simulation, obtaining sharp and fast error estimators is a challenging task. So far, results on error estimation for reduced order models of general nonlinear dynamical systems are limited, primarily because of lack of known structure (e.g. bi-linearity) within the nonlinear parts. A-priori error analysis of POD reduction schemes in different applications are considered e.g. in

[79, 176] and a-priori error bounds for certain nonlinear systems are derived in [134]. The TPWL approach also introduces a-posteriori error estimators [136, 135] assuming negative monotonicity of the involved nonlinear function. Error estimates using dual-weighted-residual methods can be found in [116] and [80], where the latter employs a "*small sample statistical method*" to estimate the state space error norm. Related recent work on a-posteriori error estimation has been done for nonlinear parameterized evolution equations [47] or linear non-affine time-varying PDEs [65]. A related scheme for error estimators of linear but parameterized dynamical systems can be found in [74, 87].

In this chapter, we shall derive a-posteriori error estimation procedures for dynamical systems using both kernel methods and DEIM as means of nonlinear approximation, where intermediate results have been published [182, 181, 186] in similar form. The estimators can be fully decomposed in the offline/online fashion and hence fit into the paradigms introduced in Section 3.3. Key structural elements to all derived estimators are an estimation of the error system, local Lipschitz constant estimations and application of the comparison lemma [76]. The estimators bound the error in the state space for finite time intervals instead of giving frequency domain error bounds.

Now, in order to keep the notational complexity at a minimum, we shall introduce the core concepts for "simple" nonlinear systems of the form

$$\boldsymbol{x}'(t) = \boldsymbol{f}(\boldsymbol{x}(t)), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0. \tag{5.1}$$

We will deal with the more general case of affine-parametric systems with inputs like (1.12a, p.19) later. To this end, we assume to have given biorthogonal projection matrices $\boldsymbol{V}, \boldsymbol{W}$ and an approximation $\hat{\boldsymbol{f}}$ to $\boldsymbol{f}$. Recalling the reconstructed state space vector $\boldsymbol{x}^r(t) = \boldsymbol{V}\boldsymbol{z}(t)$, we denote the "*state space error*" by $\boldsymbol{e}(t) := \boldsymbol{x}(t) - \boldsymbol{x}^r(t)$, which is also given as the solution of the error system

$$\boldsymbol{e}'(t) = \boldsymbol{f}(\boldsymbol{x}(t)) - \boldsymbol{V}\tilde{\hat{\boldsymbol{f}}}(\boldsymbol{z}(t)) = \boldsymbol{f}(\boldsymbol{x}(t)) - \boldsymbol{V}\boldsymbol{W}^T\hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(t)), \tag{5.2a}$$
$$\boldsymbol{e}(0) = \boldsymbol{x}_0 - \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{x}_0. \tag{5.2b}$$

Now, we want to find a state space error estimator Δ that satisfies

$$||\boldsymbol{e}(t)||_G \leq \Delta(t), \qquad t \in [0, T].$$

While there might be other ways to do this, estimating the norm of the right hand side of (5.2a) is the first basic step we consider in order to obtain a-posteriori error estimators. In detail, we will aim to obtain an estimation of (5.2a) like

$$||\boldsymbol{e}'(t)||_G \leq \alpha(t) + \beta(t)\,||\boldsymbol{e}(t)||_G\,, \tag{5.3}$$

for suitable $\alpha(t), \beta(t)$, which ideally can be computed fast in the online-phase and should be independent of any complete $\boldsymbol{f}$ evaluations or $\boldsymbol{x}(t)$. This kind of differential inequalities can be easily turned into an explicit bound using the "*comparison lemma*" [76, p.32], which is also applied frequently to bound solutions or errors, but *a-priori*. We state it here in a tailored version for completeness.

Lemma 5.0.1 (Comparison Lemma). *Let $T > 0$, $\Delta, \alpha, \beta : [0, T] \to \mathbb{R}$ be integrable functions, Δ differentiable and assume*

$$\Delta'(t) \leq \beta(t)\Delta(t) + \alpha(t) \quad \forall\, t \in [0, T]. \tag{5.4}$$

Then

$$\Delta(t) \leq \int_0^t \alpha(s) e^{\int_s^t \beta(\tau)d\tau} ds + e^{\int_0^t \beta(\tau)d\tau}\Delta(0) \quad \forall\, t \in [0, T]. \tag{5.5}$$

Furthermore, (5.5) *is an equality if and only if* (5.4) *is an equality.*

Proof. Define $v(t) := \int_0^t \beta(\tau)d\tau$. Then (5.5) follows from

$$\begin{aligned}
\Delta(t) &= e^{v(t)}e^{-v(t)}\Delta(t) - e^{v(t)}e^{-v(0)}\Delta(0) + e^{v(t)}e^{-v(0)}\Delta(0)\\
&= e^{v(t)}\int\limits_0^t \left(e^{-v(s)}\Delta(s)\right)' ds + e^{v(t)}\Delta(0)\\
&= e^{v(t)}\int\limits_0^t e^{-v(s)}\big(\Delta'(s) - \beta(s)\Delta(s)\big)ds + e^{v(t)}\Delta(0)\\
&\leq e^{v(t)}\int\limits_0^t e^{-v(s)}\alpha(s)ds + e^{v(t)}\Delta(0)\\
&= \int\limits_0^t \alpha(s)e^{\int\limits_s^t \beta(\tau)d\tau}ds + e^{\int\limits_0^t \beta(\tau)d\tau}\Delta(0).
\end{aligned}$$

This derivation also directly shows the equality of (5.5) if (5.4) is an equality. On the other hand, if (5.5) is an equality, then

$$\begin{aligned}
0 &= \Delta(t) - \int\limits_0^t \alpha(s)e^{\int\limits_s^t \beta(\tau)d\tau}ds + e^{\int\limits_0^t \beta(\tau)d\tau}\Delta(0)\\
&= e^{v(t)}\int\limits_0^t e^{-v(s)}\big(\Delta'(s) - \beta(s)\Delta(s) - \alpha(s)\big)ds \quad \forall\, t \in [0, T],
\end{aligned}$$

which shows equality in (5.4) as the exponential is strictly positive. □

From this it is clear that we can immediately obtain explicit estimators Δ once an estimation of the type (5.3) is available. In doing that, we face a number of choices, given that we have to "get" from $\boldsymbol{f}(\boldsymbol{x}(t))$ to $\boldsymbol{V}\boldsymbol{W}^T\boldsymbol{f}(\boldsymbol{V}\boldsymbol{z}(t))$ somehow. The obvious choice is to insert zero elements into (5.2a) that can be estimated separately (e.g. using the triangular inequality) later on. As there are several possibilities with different advantages, the complete list of all six possibilities is given by

$$\begin{aligned}
\boldsymbol{e}'(t) &= \boldsymbol{f}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{x}(t)) + \hat{\boldsymbol{f}}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(t)) \\
&\quad + \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right) \hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(t)), \qquad (5.6a)\\
\boldsymbol{e}'(t) &= \boldsymbol{f}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{x}(t)) + \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right) \hat{\boldsymbol{f}}(\boldsymbol{x}(t)) \\
&\quad + \boldsymbol{V}\boldsymbol{W}^T \hat{\boldsymbol{f}}(\boldsymbol{x}(t)) - \boldsymbol{V}\boldsymbol{W}^T \hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(t)), \\
\boldsymbol{e}'(t) &= \boldsymbol{f}(\boldsymbol{x}(t)) - \boldsymbol{f}(\boldsymbol{V}\boldsymbol{z}(t)) + \boldsymbol{f}(\boldsymbol{V}\boldsymbol{z}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(t)) \\
&\quad + \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right) \hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(t)), \qquad (5.6b)\\
\boldsymbol{e}'(t) &= \boldsymbol{f}(\boldsymbol{x}(t)) - \boldsymbol{f}(\boldsymbol{V}\boldsymbol{z}(t)) + \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right) \boldsymbol{f}(\boldsymbol{V}\boldsymbol{z}(t)) \\
&\quad + \boldsymbol{V}\boldsymbol{W}^T \boldsymbol{f}(\boldsymbol{V}\boldsymbol{z}(t)) - \boldsymbol{V}\boldsymbol{W}^T \hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(t)), \\
\boldsymbol{e}'(t) &= \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right) \boldsymbol{f}(\boldsymbol{x}(t)) + \boldsymbol{V}\boldsymbol{W}^T \left(\boldsymbol{f}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{x}(t))\right) \\
&\quad + \boldsymbol{V}\boldsymbol{W}^T \left(\hat{\boldsymbol{f}}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(t))\right), \\
\boldsymbol{e}'(t) &= \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right) \boldsymbol{f}(\boldsymbol{x}(t)) + \boldsymbol{V}\boldsymbol{W}^T \left(\boldsymbol{f}(\boldsymbol{x}(t)) - \boldsymbol{f}(\boldsymbol{V}\boldsymbol{z}(t))\right) \\
&\quad + \boldsymbol{V}\boldsymbol{W}^T \left(\boldsymbol{f}(\boldsymbol{V}\boldsymbol{z}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(t))\right).
\end{aligned}$$

Basically, at some stage, all versions need to deal with the approximation error $\boldsymbol{f} - \hat{\boldsymbol{f}}$, the projection residual $\left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)$ and some Lipschitz-type estimation for the argument change $\boldsymbol{x} \to \boldsymbol{V}\boldsymbol{z}$. In our work, we will investigate the possibilities (5.6a) and (5.6b) closer, as the other variations only estimate the projection error "earlier" on the bottom line. While (5.6a) requires an a-priori bound on $\boldsymbol{f} - \hat{\boldsymbol{f}}$ (it is evaluated at the unknown full state space $\boldsymbol{x}(t)$), it allows to perform Lipschitz-type estimations using $\hat{\boldsymbol{f}}$, which can presumably be provided easier than e.g. a Lipschitz constant of $\boldsymbol{f}$. This requirement of a Lipschitz-type estimation involving $\boldsymbol{f}$ is the drawback of variant (5.6b), however, the approximation error $\boldsymbol{f} - \hat{\boldsymbol{f}}$ can be evaluated using the well-known reduced state $\boldsymbol{z}(t)$. While the order of introduction is arbitrary, we shall stick with the chronological order in which the results have been obtained during the last years. Consequently, we will discuss error estimators of the type (5.6a) for reduced systems involving kernels in Section 5.1 and Section 5.1.3. Estimators for POD-DEIM reduced nonlinear systems starting from equation (5.6b) will be discussed in Section 5.2. In the following, we will denote by

$$E_A(\boldsymbol{x}) := \boldsymbol{f}(\boldsymbol{x}) - \hat{\boldsymbol{f}}(\boldsymbol{x}) \tag{5.7}$$

the approximation error of the nonlinearity $\boldsymbol{f}$ and define the initial error as

$$E_0 := \left|\left|\boldsymbol{x}_0 - \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{x}_0\right|\right|_G.$$

5.1 Error estimation for systems with kernel expansions

In this section we shall assume $\hat{\boldsymbol{f}}$ to be given as a kernel expansion

$$\hat{\boldsymbol{f}}(\boldsymbol{x}) = \sum_{i=1}^{n} \boldsymbol{c}_i K(\boldsymbol{x}_i, \boldsymbol{x}), \tag{5.8}$$

similar to (1.7, p.14) or (3.9, p.98). Starting from the error system representation (5.6a), Theorem 5.1.1 states a first a-posteriori error estimator using a global Lipschitz condition.

Theorem 5.1.1 (Global Lipschitz constant error Estimator (GLE)). *Let K be Lipschitz with constant $L_K \in \mathbb{R}$. Then, given the nonlinear dynamical system (5.2, p.139), the state space error is rigorously bounded via*

$$||\boldsymbol{e}(t)||_G \leq \Delta_{GLE}(t) \;\; \forall\, t \in [0, T],$$

with

$$\Delta_{GLE}(t) := \int_0^t \alpha(s) e^{\int_s^t L_{\hat{\boldsymbol{f}}} dr} ds + e^{\int_0^t L_{\hat{\boldsymbol{f}}} ds} E_0 = \int_0^t \alpha(s) e^{L_{\hat{\boldsymbol{f}}}(t-s)} ds + e^{L_{\hat{\boldsymbol{f}}} t} E_0,$$
$$\alpha(t) := ||E_A(\boldsymbol{x}(t))||_G + \left|\left|\left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right) \hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(t))\right|\right|_G,$$

$$L_{\hat{\boldsymbol{f}}} := L_K \sum_{i=1}^{n} ||\boldsymbol{c}_i||_G .$$

Proof. Using the kernel expansion structure it is easy to see that

$$\left|\left|\hat{\boldsymbol{f}}(\boldsymbol{x}_1) - \hat{\boldsymbol{f}}(\boldsymbol{x}_2)\right|\right|_G \leq L_{\hat{\boldsymbol{f}}} \, ||\boldsymbol{x}_1 - \boldsymbol{x}_2||_G ,$$

for $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^d$. With this we obtain

$$\begin{aligned} ||\boldsymbol{e}'(t)|| &= \Big|\Big|\boldsymbol{f}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{x}(t)) + \hat{\boldsymbol{f}}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(t)) \\ &\qquad + \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(t))\Big|\Big|_G \\ &\leq ||E_A(\boldsymbol{x}(t))||_G + \left|\left|\hat{\boldsymbol{f}}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(t))\right|\right|_G \\ &\qquad + \left|\left|\left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(s))\right|\right|_G \\ &\leq \alpha(t) + L_{\hat{\boldsymbol{f}}} \, ||\boldsymbol{e}(t)||_G , \end{aligned}$$

and application of Lemma 5.0.1 for $\Delta(t) = ||\boldsymbol{e}(t)||_G$ and $\beta(t) \equiv L_{\hat{\boldsymbol{f}}}$ finishes the proof. $\square$

Now, while the second summand of the α term is readily computable, we yet need to find an $\boldsymbol{x}(t)$-independent expression for E_A.

If $\hat{\boldsymbol{f}}$ was obtained using the VKOGA variants from Section 2.2.2, the a-priori or a-posteriori error bounds from Section 2.2.2.2 could be applied. However, as already seen in the experiments in Section 2.2.2.4, those bounds are very conservative and are not suitable to derive sharp error estimates.

As already shortly mentioned at the end of Section 2.2.2.4, an alternative is the fill-distance based approach introduced in [178, §11], whenever $\boldsymbol{f}$ is obtained by orthogonal projection, i.e. interpolation. Those estimates are considerably sharp and have high convergence rates, but suffer from high continuity requirements regarding the kernels and a small fill distance. While smooth kernels can be appropriate in many situations, the necessary good coverage of the considered domain is practically infeasible, especially keeping in mind that one central goal of the nonlinear kernel approximation schemes introduced in Chapter 2 targets a *sparse* function representation.

Moreover, in the case of SVMs, various results on learning rates have been established [163, 188, 164], of which the "*oracle inequality*"-type results [161, 160] are probably closest to our setting. But besides being valid for only certain types of kernels, the inherent statistical perspective causes those rates to include noise assumptions on the training data, which are not necessary in our deterministic setting.

To this end, while the above approach is an analytical result in its own interest, it remains an open question if a-priori error bounds that either are not overly conservative or do not depend on a tight coverage of the considered domain can be devised for practical purposes. Even so, due to the modularity of the estimator in Theorem 5.1.1, any a-priori estimation technique can be applied and substituted for E_A. Additionally, whatever estimation for E_A is used, the overall estimator still involves a *global* Lipschitz constant L_K of the kernel, which contributes to the estimation result in an exponential way. As this might even be the more crucial part in the overall estimation process, we will focus on ways to improve that part of the estimation process in the following.

5.1.1 Local Lipschitz constants

Closer investigation of the error estimator from Theorem 5.1.1 shows that if the kernel Lipschitz constant L_K is large, the estimator will grow fast and the estimation will be overly conservative. But yet there are improvement possibilities as those error estimators do not utilize all information given during the estimation process. So far, only

$$\left|\left|\hat{\boldsymbol{f}}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{x}^r(t))\right|\right|_G \leq L_{\hat{\boldsymbol{f}}} \left|\left|\boldsymbol{x}(t) - \boldsymbol{x}^r(t)\right|\right|_G$$

is used. But as $\boldsymbol{x}^r(t)$ is known during the reduced simulation, a variant like

$$\left|\left|\hat{\boldsymbol{f}}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{x}^r(t))\right|\right|_G \leq L_{\hat{\boldsymbol{f}}}(\boldsymbol{x}^r(t))\,||\boldsymbol{x}(t) - \boldsymbol{x}^r(t)||_G$$

should be possible, where L utilizes information from the reduced system. As those estimations use local information we call them "*local Lipschitz constant estimations*". Moreover, assuming that

$$||\boldsymbol{e}(t)||_G = ||\boldsymbol{x}(t) - \boldsymbol{x}^r(t)||_G < \Theta(t)$$

for some $\Theta > 0$ (maybe a coarse upper bound, see Section 5.1.2 for details) could further improve the local Lipschitz constant estimation, as it would mean

$$\boldsymbol{x}(t) \in \overline{B_\Theta\left(\boldsymbol{x}^r(t)\right)},$$

which allows for a even more localized Lipschitz constant estimation

$$\left|\left|\hat{\boldsymbol{f}}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{x}^r(t))\right|\right|_G \leq L_{\hat{\boldsymbol{f}},\Theta}(\boldsymbol{x}^r(t))\,||\boldsymbol{x}(t) - \boldsymbol{x}^r(t)||_G\,.$$

Therefore, we propose an approach using local secant gradients involving an a-priori error bound $||\boldsymbol{e}(t)||_G \leq \Theta$, which enables efficient computation of those secant gradients utilizing the kernel expansion center locations. The derived theory and results for a-posteriori error estimation of kernel-based dynamical systems have been published in [182] in similar form.

The key to our local Lipschitz constant computations is to use a subclass of the RBF kernels introduced in Section 3.2.1, (1.1.4, p.10), induced by a certain class of scalar functions ϕ:

Definition 5.1.2 (Bell functions). A function $\phi : \mathbb{R}_+ \to \mathbb{R}_+$ is called a "*bell function*", if

$$i) \quad \phi \in C^2(\mathbb{R}_+),$$

$$ii) \quad ||\phi||_{L^\infty} \leq B,\ B > 0, \tag{5.9a}$$

$$iii) \quad \phi'(0) \leq 0, \tag{5.9b}$$

$$iv) \qquad \exists\, r_0 > 0 :\ \phi''(r)(r - r_0) > 0\, \forall\, r \neq r_0. \tag{5.9c}$$

Condition $iv)$ means that r_0 denotes a unique turning point where ϕ is strictly concave/convex on $[0, r_0]/[r_0, \infty[$, respectively. We will denote the set of all bell functions with $\mathcal{B}$ and assume $\phi \in \mathcal{B}$ for the remainder of this section. For example, the Gaussian kernel is induced by a bell function $\phi(r) = e^{-(r/\beta)^2}$. Then we have $r_0 = \beta/\sqrt{2}$ and $L_K = |\phi'(r_0)|$ in the context of Theorem 5.1.1.

We will state results regarding bell functions first and connect these to the multidimensional case later.

Lemma 5.1.3 (Basic properties of bell functions). *Let $\phi \in \mathcal{B}$. Then we have*

$$\min_{x \in \mathbb{R}_+} \phi'(x) = \phi'(x_0) \tag{5.10a}$$

$$\phi'(x) \overset{x\to\infty}{\longrightarrow} 0 \tag{5.10b}$$

$$\phi'(x) < 0\, \forall\, x \in \mathbb{R}^+, \tag{5.10c}$$

where (5.10c) *means especially that ϕ is strictly monotonously decreasing and takes its maximum value in $\phi(0)$.*

Proof. Property (5.10a) follows directly from (5.9b) and (5.9c). Especially we have $\phi'(x_0) < 0$. Now we show (5.10b) by contradiction. Assume

$$\exists\, \epsilon > 0, z_0 > 0\, \forall\, z \geq z_0 : |\phi'(z)| \geq \epsilon.$$

Then by the mean value theorem we see that

$$\forall\, n \in \mathbb{N}\, \exists\, \xi_n > z_0 : \frac{\phi(z_0 + n) - \phi(z_0)}{n} = \phi'(\xi_n).$$

But this means

$$|\phi(z_0 + n) - \phi(z_0)| = n|\phi'(\xi_n)| \geq \epsilon n \overset{n\to\infty}{\longrightarrow} \infty,$$

which contradicts with the boundedness (5.9a) of ϕ. Finally, we can see statement (5.10c) as follows: Condition (5.9b) and (5.9c) guarantee

$$\phi'(x) < 0 \,\forall\, x \in \,]0, x_0].$$

Condition (5.9c) also says for $x > x_0$ that ϕ' is strictly monotonously increasing; but this already means

$$\phi'(x) < 0 \,\forall\, x > x_0,$$

since if we had a x' with $\phi'(x') = 0$ this would imply $\phi'(x) > 0 \,\forall\, x > x'$ in contradiction with condition (5.10b). □

The notion of secant gradients will prove useful for our further investigations.

Definition 5.1.4 (Secant gradient function). For any $\phi \in \mathcal{B}, s \in \mathbb{R}_+$, we define the *secant gradient function* $S_s : \mathbb{R}_+ \longrightarrow \mathbb{R}$ as

$$S_s(r) := \begin{cases} \frac{\phi(r)-\phi(s)}{r-s}, & r \neq s, \\ \phi'(s), & r = s. \end{cases} \tag{5.11}$$

Its derivative is given as

$$S_s'(r) := \begin{cases} \frac{\phi'(r) - \frac{\phi(r)-\phi(s)}{r-s}}{r-s} = \frac{\phi'(r)-S_s(r)}{r-s} & r \neq s, \\ \phi''(r) & r = s. \end{cases}$$

Well-definedness and continuous differentiability of S_s can easily be verified by elementary analysis. The following lemma characterizes the minimum secant gradient position.

Lemma 5.1.5 (Minimum secant gradient). *Let $\phi \in \mathcal{B}, s \in \mathbb{R}_+$. Then*

$$r_s := \arg\min_{r \in \mathbb{R}_+} S_s(r) \tag{5.12}$$

exists, is unique and exactly one of the following cases holds:

$$s < r_0 \leq r_s, \tag{5.13a}$$

$$r_s \leq r_0 < s, \tag{5.13b}$$

$$s = r_0 = r_s. \tag{5.13c}$$

Proof. It is clear that $\phi'(r_0) \leq S_s(r) \leq 0 \,\forall\, r \in \mathbb{R}_+$. Using the boundedness of ϕ we see that

$$|S_s(r)| \leq \left|\frac{\phi(r) - \phi(s)}{r - s}\right| \leq \frac{B + \phi(s)}{|r - s|} \overset{r\to\infty}{\longrightarrow} 0.$$

As S_s is continuous and $S_s(r_0) < 0 \,\forall\, s$ we see that $r_s < \infty$ and hence obtain existence of a minimizer. Next we show the inequalities (5.13) and note that the necessary condition for a minimizer is

$$0 = S_s'(r_s) = \frac{\phi'(r_s) - S_s(r_s)}{r_s - s} \Leftrightarrow \phi'(r_s) = S_s(r_s). \tag{5.14}$$

Recall that every bell function has a unique $r_0 > 0$ as given by property (5.9c, p.146). Then we differentiate the cases

- $s < r_0$
  We continue with proof by contradiction and therefore assume we had $r_s < r_0$. Now choose a $r \in \mathbb{R}_+$ with $\max(r_s, s) < r < r_0$. Then we have $S_s(r) = \frac{\phi(r)-\phi(s)}{r-s} = \phi'(\xi)$ for some $\xi \in]s, r[$ by the mean value theorem. Now as ϕ is strictly concave on $[0, r_0]$ by the bell function condition (5.9c, p.146) and $\phi'(r) \leq 0$ by (5.9b, p.145) we obtain $S_s(r_s) = \phi'(r_s) > \phi'(\xi) = S_s(r)$, which contradicts with the minimizing property of r_s as obviously r_s is not a minimizer of S_s.

- $r_0 < s$
  The case $r_0 < s$ is obtained by an analogous argument using the strict convexity of ϕ on $[r_0, \infty[$ instead of the concavity as in the former case.

- $r_0 = s$
  This case is clear when considering the limit process for both other cases.

At last, we prove the uniqueness of the maximizers. Assume there are two maximizing $r_1, r_2 \in \mathbb{R}_+$ with $r_1 \neq r_2$ Without loss of generality say $r_1 < r_2$. Consider the case $s < r_0$. Then we know by inequality (5.13a) that we must have $s < r_0 \leq r_1 < r_2$. Now, ϕ is strictly convex on $[r_0, \infty[$ and in combination with the necessary condition (5.14) we get

$$S_s(r_1) = \phi'(r_1) > \phi'(r_2) = S_s(r_2),$$

which contradicts the maximizing assumptions on r_1, r_2. The case $r_0 < s$ follows by the same arguments using (5.13b) and concavity. Finally, $r_0 = s$ directly implies $r_1 = r_2$ by equation (5.13c), which shows well-definedness of (5.12, p.147) and concludes the proof. $\square$

Figure 5.1 shows two examples for minimum secant gradients and the position of r_s with each $s < r_0$ (left) and $s > r_0$ (right), using the Gaussian inducing bell function $\phi(r) = \exp(-r^2/\beta^2)$ with $\beta = 2$.

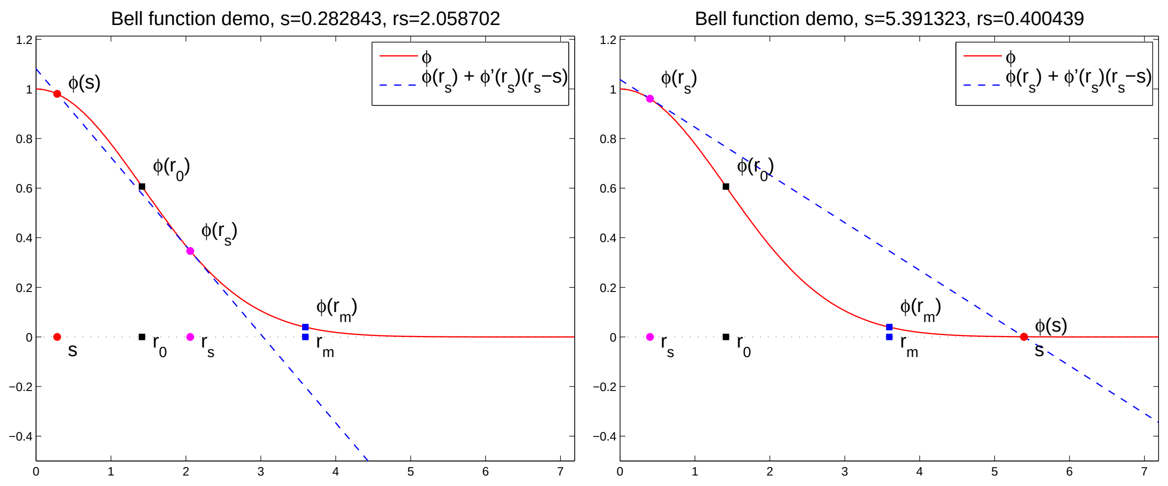


Figure 5.1: Gaussian ϕ with $\beta = 2$, $r_0 = \sqrt{2}$ and $r_m = \frac{\sqrt{2}}{1-e^{-1/2}} \approx 3.5942$

The following Lemma establishes a helpful result in order to include the aforementioned a-priori bound Θ.

Lemma 5.1.6 (Monotony of secant gradient derivatives for bell functions). *Let $s \in \mathbb{R}_+ \backslash \{r_0\}$ and S_s be given as in (5.11, p.147). Then*

$$S_s'(r)(r - r_s) > 0 \,\forall\, r \in \mathbb{R}_+ \backslash \{r_s\}.$$

Proof. Choose $s \in \mathbb{R}_+$ and recall $S_s'(r) = \frac{\phi'(r) - S_s(r)}{r-s}$ from Definition 5.1.4. First consider the case $s < r_0$ and differentiate by locations of r:

- $r < s < r_0 \leq r_s$
  We have $r - r_s < 0,\ r - s < 0$ and $S_s(r) > \phi'(r)$ by concavity.

- $r = s < r_0 \leq r_s$
  We have
  $$\begin{aligned} S_s'(s) &= \lim_{t \to s} S_s'(t) = \lim_{t \to s} \frac{\phi'(t) - S_s(t)}{t - s} \\ &= \lim_{t \to s} \frac{\phi''(t) - S_s'(t)}{1} = \phi''(s) - \lim_{t \to s} S_s'(t). \end{aligned}$$
  This means $S_s'(s) = \frac{\phi''(s)}{2} < 0$ and as $r < r_s$ we have $S_s'(r)(r - r_s) > 0$.

- $s < r \leq r_0 \leq r_s$
  We have $r - r_s < 0,\ r - s > 0$ and $S_s(r) < \phi'(r)$ by concavity.

- $s < r_0 \leq r < r_s$
  Since $\phi'(r_0) < S_s(r_0)$ and $\phi'(r_s) = S_s(r_s)$ we must have $\phi'(r) < S_s(r) \,\forall\, r \in [r_0, r_s[$. Otherwise, since $\phi'(\cdot) - S_s(\cdot)$ is continuous, there would exist an $r' \in [r_0, r_s[$ with
  $$S_s(r') = \phi'(r') < \phi'(r_s) = S_s(r_s),$$
  as $r' < r_s$ and ϕ is strictly convex on $[r_0, \infty[$. But this is a contradiction to the minimal choice of r_s, and together with $r - r_s < 0,\ r - s > 0$ we obtain $S_s'(r)(r - r_s) > 0$.

- $s < r_0 \leq r_s < r$

At r_s we have $S_s'(r_s) = 0$ and thus

$$S_s''(r_s) = \frac{\phi''(r_s) - 2S_s'(r_s)}{r_s - s} = \frac{\phi''(r_s)}{r_s - s} > 0,$$

as ϕ is strictly convex for $r > r_0$. So there exists an $\epsilon > 0$ with $S_s'(r_s + \epsilon) > 0$. In fact for any point $r > r_0$ with $S_s'(r) = 0$ we have $S_s''(r) > 0$, meaning all of them are local minima. But as S_s' is continuous there cannot be two consecutive local minima without a local maximum. Hence, $S_s'(r)$ cannot equal zero for $r > r_s$ and since $S_s'(r_s + \epsilon) > 0$ we must have $S_s'(r) > 0\ \forall\, r > r_s$.

Similar to the previous proofs, the case $r_0 < s$ is obtained using an analogous argumentation with concavity and convexity interchanged. □

Now, further assume to restrict admissible r values to $r \in \overline{B_\Theta(s)} \cap \mathbb{R}_+$ for a $\Theta > 0$, where $B_\Theta(s)$ denotes an open ball of radius Θ around s and $\overline{B_\Theta(s)}$ its closure. With this condition we can state a result regarding local Lipschitz constant computations.

Proposition 5.1.7 (Local Lipschitz estimation using secant gradients). *Let $\Theta > 0, s \in \mathbb{R}_+, \phi \in \mathcal{B}$ and $\Gamma := \overline{B_\Theta(s)} \cap \mathbb{R}_+$. Then*

$$|\phi(r) - \phi(s)| \leq |S_s(r_{\Theta,s})||r - s|, \qquad \forall\, r \in \Gamma,$$

with

$$r_{\Theta,s} = \begin{cases} s + \operatorname{sign}(r_s - s)\,\Theta, & r_s \notin \Gamma, \\ r_s, & r_s \in \Gamma. \end{cases} \tag{5.15}$$

Proof. Assume $s \neq r_0$. Then Lemma 5.1.6 implies that S_s is monotonically decreasing on $[0, r_s]$ and increasing on $[r_s, \infty[$. For $r_s \notin \Gamma$, as S_s is negative, $|S_s|$ takes its maximum value at the border of Γ that is closest to r_s. Since $\arg\min_{r\in\mathbb{R}_+} S_s(r) = \arg\max_{r\in\mathbb{R}_+} |S_s(r)|$, case $r_s \in \Gamma$ follows by definition. The case $s = r_0$ implies $r_s = r_0$ and hence case two, as of course $r_0 \in \Gamma =$

$\overline{B_\Theta(r_0)}$ and $\Theta > 0$. We finally conclude $|\phi(x) - \phi(y)| \leq ||S_s||_{L^\infty(\Gamma)} |r - s| = |S_s(r_{\Theta,s})||r - s|$. $\square$

Next, we generalize these results to the multidimensional kernel setting.

Corollary 5.1.8 (Local Lipschitz constants for bell function kernels). *Let the conditions of Proposition 5.1.7 hold and choose $\boldsymbol{y}, \boldsymbol{z} \in \Gamma$. Further assume to have given an RBF Kernel K induced by ϕ and set $s = ||\boldsymbol{y} - \boldsymbol{z}||_G$. Then we have*

$$|K(\boldsymbol{x}, \boldsymbol{z}) - K(\boldsymbol{y}, \boldsymbol{z})| \leq |S_s(r_{\Theta,s})| \, ||\boldsymbol{x} - \boldsymbol{y}||_G \quad \forall\, \boldsymbol{x} \in \overline{B_\Theta(y)}.$$

Proof. Fix $\boldsymbol{y}, \boldsymbol{z} \in \Gamma,\ \Theta > 0$, let $R := \left\{\boldsymbol{x} \in \Gamma \;\middle|\; ||\boldsymbol{x} - \boldsymbol{z}||_G \in \overline{B_\Theta(s)}\right\}$. Then using Proposition 5.1.7 we estimate

$$\begin{aligned} |K(\boldsymbol{x}, \boldsymbol{z}) - K(\boldsymbol{y}, \boldsymbol{z})| &= |\phi(||\boldsymbol{x} - \boldsymbol{z}||_G) - \phi(||\boldsymbol{y} - \boldsymbol{z}||_G)| \\ &\leq |S_s(r_{\Theta,s})| \left| ||\boldsymbol{x} - \boldsymbol{z}||_G - ||\boldsymbol{y} - \boldsymbol{z}||_G \right| \\ &\leq |S_s(r_{\Theta,s})| \, ||\boldsymbol{x} - \boldsymbol{y}||_G \ \forall\, \boldsymbol{x} \in R. \end{aligned}$$

The fact that $\overline{B_\Theta(y)} \subset R$ finishes the proof. $\square$

Finally, those results can be used to state an improved version of the error estimator derived in Theorem 5.1.1. As the above results hold true at any location for any bound Θ, we will assume the a-priori error bound to be time-dependent, i.e.

$$||\boldsymbol{e}(t)||_G \leq \Theta(t)\ \forall\, t \in [0, T]. \tag{5.16}$$

If no further knowledge is available, $\Theta(t) \equiv \infty$ is a valid option, in whose case we obtain $r_{\Theta,s} \equiv r_s$ in the context if Proposition 5.1.7. We also introduce the notation

$$d_i(t) := ||\boldsymbol{x}^r(t) - \boldsymbol{x}_i||_G\,, \quad i = 1 \dots N, \tag{5.17}$$

for the distance of the reduced state $\boldsymbol{x}^r(t)$ to each expansion center $\boldsymbol{x}_i$ during the reduced simulation. In fact, the coarse error bound means nothing but

$\boldsymbol{x}(t) \in \overline{B_{\Theta(t)}(\boldsymbol{x}^r(t))}\ \forall\, t \in [0,T]$. Consequently, for any $t \in [0,T], i \in \{1 \dots N\}$ we identify $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^d$ and $\Theta > 0$ from Corollary 5.1.8 with $\boldsymbol{x}(t), \boldsymbol{x}^r(t)$ of the full/reduced solution, the kernel expansion centers $\boldsymbol{x}_i$ and $\Theta(t)$, respectively. This allows to obtain local Lipschitz constant estimations at each time t using the current reduced simulation's state $\boldsymbol{x}^r(t)$ as "center" of locality:

Theorem 5.1.9 (**L**ocal **S**ecant gradient **L**ipschitz error **E**stimator (LSLE))**.** *Let the error system be given as in (5.2, p.139), where the kernel K of the expansion (5.8, p.142) is induced by a bell function ϕ. Further, let $\Theta(t)$ be an a-priori error bound. Then the state space error is bounded via*

$$||e(t)||_G \leq \Delta^{\Theta}_{LSLE}(t)\ \forall\, t \in [0,T],$$

with

$$\begin{aligned}
\Delta^{\Theta}_{LSLE}(t) &:= \int_0^t \alpha(s) e^{\int_s^t \beta(r)dr} ds + e^{\int_0^t \beta(s)ds} E_0,\\
\alpha(t) &:= ||E_A(\boldsymbol{x}(t))||_G + \left|\left| \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right) \hat{\boldsymbol{f}}(\boldsymbol{V}\boldsymbol{z}(t)) \right|\right|_G,\\
\beta(t) &:= \sum_{i=1}^n ||\boldsymbol{c}_i||_G \left| S_{d_i(t)}(r_{\Theta(t),d_i(t)}) \right|.
\end{aligned}$$

Proof. The α term is derived exactly as in the proof of Theorem 5.1.1. Next, using the kernel expansion and Corollary 5.1.8 yields

$$\begin{aligned}
\left|\left| \hat{\boldsymbol{f}}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{x}^r(t)) \right|\right|_G &\leq \sum_{i=1}^n ||\boldsymbol{c}_i||_G \left(K(\boldsymbol{x}(t), \boldsymbol{x}_i) - K(\boldsymbol{x}^r(t), \boldsymbol{x}_i)\right)\\
&\leq \sum_{i=1}^n ||\boldsymbol{c}_i||_G \left| S_{d_i(t)}(r_{\Theta(t),d_i(t)}) \right| ||\boldsymbol{e}(t)||_G\\
&= \beta(t) ||\boldsymbol{e}(t)||_G\,.
\end{aligned}$$

Application of Lemma 5.0.1 for the ODE $\Delta'(t) = \beta(t)\Delta(t) + \alpha(t),\ \Delta(0) = E_0$ yields the result. $\square$

5.1.2 Iterative a-posteriori error estimators

We still need to justify the assumption (5.16) because it requires existence of an a-priori bound in order to obtain an a-posteriori error estimator. The core idea is to use the a-priori bound in an iterative scheme starting with $\Theta(t) \equiv \infty$, which is trivially satisfied. Then the resulting a-posteriori error estimate is reused as a-priori bound for another iteration of the error estimation procedure. A nice feature of this procedure is that it allows to balance the error bound's sharpness against the computational costs. The next theorem shows that actual progress can be made under reasonable assumptions and the iterative scheme reproduces the first estimate in the worst case.

Theorem 5.1.10 (Convergence of error estimator iterations). *Let the conditions of Theorem 5.1.9 hold and define*

$$\Theta_0(t) \equiv \infty, \qquad \Theta_m(t) := \Delta_{LSLE}^{\Theta_{m-1}}(t), \quad m \geq 1.$$

Recall the distances $d_i(t)$ from (5.17, p.152), r_s from Lemma 5.1.5,(5.12, p.147) and for $m \in \mathbb{N}$ let

$$\Gamma_m := \left\{ t \in [0,T] \;\middle|\; \exists\, i \in \{1 \dots N\} : \, r_{d_i(t)} \notin \overline{B_{\Theta_m(t)}\left(d_i(t)\right)} \right\}. \tag{5.18}$$

If $\Gamma_1 \neq \emptyset$, then

$$\begin{aligned} &\Gamma_{m+1} \neq \emptyset \;\; \forall\, m \in \mathbb{N}, \\ &\Delta_{LSLE}^{\Theta_{m+1}}(t) \begin{cases} < \Delta_{LSLE}^{\Theta_m}(t), & t > \inf \Gamma_{m+1} \\ = \Delta_{LSLE}^{\Theta_m}(t), & t \leq \inf \Gamma_{m+1} \end{cases} \quad \forall\, m \in \mathbb{N}. \end{aligned} \tag{5.19}$$

Else if $\Gamma_1 = \emptyset$, then

$$\Delta_{LSLE}^{\Theta_m}(t) = \Theta_1(t) \;\forall\, m \geq 1, \quad t \in [0,T].$$

Further, the sequence of iterates converges uniformly to a continuous lower *error bound*

$$\Delta^{\infty}_{LSLE}(t) := \lim_{m\to\infty} \Delta^{\Theta_m}_{LSLE}(t), \quad t \in [0,T].$$

Proof. At first we see that Γ_m is open (in fact a union of open intervals) since $d_i(t), \Theta_m(t)$ and $r_{d_i(t)}$ are continuous in t. As shorthand for any fixed $t \in [0,T]$ let

$$\beta_m(t) := \sum_{i=1}^{n} ||\boldsymbol{c}_i||_G \, |S_{d_i(t)}(r_{\Theta_m(t),d_i(t)})|, \qquad\qquad s_i := d_i(t),$$

for any $m \in \mathbb{N}$. We prove (5.19) by induction and begin with $m = 0$. We have $\Gamma_1 \neq \emptyset$ by assumption. So, for $t \in \Gamma_1$ condition (5.18) implies that

$$M := \left\{ i \in \{1 \dots N\} \;\middle|\; r_{s_i} \notin \overline{B_{\Theta_1(t)}\left(s_i\right)} \right\}$$

is nonempty. Further, we know that

$$|S_{s_i}\big(s_i + \operatorname{sign} r_{s_i} - s_i\Theta_1(t)\big)| < |S_{s_i}(r_{s_i})|, \quad \forall\, i \in M,$$

as Lemma 5.1.6 holds since $s_i \neq r_0 \;\forall\, i \in M$. With the help of Proposition 5.1.7 we deduce

$$\begin{aligned}
\beta_1(t) &= \sum_{i\in M} ||\boldsymbol{c}_i||_G \left|S_{s_i}(r_{\Theta_1(t),s_i})\right| + \sum_{i\notin M} ||\boldsymbol{c}_i||_G \left|S_{s_i}(r_{\Theta_1(t),s_i})\right| \\
&= \sum_{i\in M} ||\boldsymbol{c}_i||_G \left|S_{s_i}\big(s_i + \operatorname{sign} r_{s_i} - s_i\Theta_1(t)\big)\right| + \sum_{i\notin M} ||\boldsymbol{c}_i||_G \left|S_{s_i}(r_{s_i})\right| \\
&< \sum_{i\in M} ||\boldsymbol{c}_i||_G \left|S_{s_i}(r_{s_i})\right| + \sum_{i\notin M} ||\boldsymbol{c}_i||_G \left|S_{s_i}(r_{s_i})\right| = \beta_0(t) \quad \forall\, t \in \Gamma_1.
\end{aligned} \tag{5.20}$$

For $\Gamma_1^c := [0,T]\backslash\Gamma_1$ we have $r_{\Theta_1(t),s_i} = r_{s_i} \;\forall\, i \in \{1 \dots N\}$ for any $t \in \Gamma_1^c$ by (5.15, p.151) and thus $\beta_{1|\Gamma_1^c} \equiv \beta_{0|\Gamma_1^c}$. Now as $\alpha(s) \geq 0$, for $t > \inf \Gamma_1$ we know with (5.20) that

$$\int\limits_{\Gamma_1\cap[0,t]} \alpha(s)e^{\int_0^s \beta_1(r)dr}\, ds < \int\limits_{\Gamma_1\cap[0,t]} \alpha(s)e^{\int_0^s \beta_0(r)dr}\, ds,$$

which proves $\Delta^{\Theta_1}_{LSLE}(t) < \Delta^{\Theta_0}_{LSLE}(t)$. Equality is given for $t \leq \inf\Gamma_1$ as $\beta_{1|\Gamma_1^c} \equiv \beta_{0|\Gamma_1^c}$ and $[0, \inf\Gamma_1] \subseteq \Gamma_1^c$, which establishes the induction base.

For the induction step let $\Gamma_m \neq \emptyset$ and

$$\Delta^{\Theta_m}_{LSLE}(t) < \Delta^{\Theta_{m-1}}_{LSLE}(t) \quad \forall\, t > \inf\Gamma_m,$$

which means $\Theta_{m+1}(t) < \Theta_m(t)\ \forall\ t > \inf\Gamma_m$ by definition. This means $\Gamma_{m+1} \neq \emptyset$ as $\Gamma_m \subseteq \Gamma_{m+1}$. So, for any $t \in \Gamma_{m+1}$ we have

$$r_{\Theta_{m+1}(t),s_i} = s_i - \Theta_{m+1}(t) > s_i - \Theta_m(t) = r_{\Theta_m(t),s_i},$$

which gives $\beta_{m+1}(t) < \beta_m(t)$ analogous to (5.20). Splitting the integral as above using Γ_{m+1} directly gives (5.19, p.154), concluding the induction.

If $\Gamma_1 = \emptyset$ we have $\beta_m \equiv \beta_1\ \forall\, m \geq 1$ and thus $\Delta^{\Theta_m}_{LSLE} \equiv \Theta_1\ \forall\, n \geq 1$.

Next, for each $t \in [0,T]$ we see from (5.19, p.154) that $\left\{\Delta^{\Theta_m}_{LSLE}(t)\right\}_{m\in\mathbb{N}}$ is monotonically decreasing and bounded by zero. Thus, we have a point-wise convergence of $\Delta^{\Theta_m}_{LSLE}$ to the limit function $\Delta^{\infty}_{LSLE}(t) := \lim_{n\to\infty} \Delta^{\Theta_m}_{LSLE}(t)$.

From the induction we know that for $\forall\, m,k \in \mathbb{N}, k \geq m$ we have $\beta_k(t) \leq \beta_m(t)\ \forall\, t \in [0,T]$. This implies

$$\left|\Delta^{\Theta_k}_{LSLE}(t) - \Delta^{\Theta_m}_{LSLE}(t)\right| \leq \left|\Delta^{\Theta_k}_{LSLE}(T) - \Delta^{\Theta_m}_{LSLE}(T)\right| \quad \forall\, t \in [0,T].$$

Taking the supremum on the left and considering $k \to \infty$ we obtain

$$\left|\left|\Delta^{\infty}_{LSLE} - \Delta^{\Theta_m}_{LSLE}\right|\right|_{L^\infty(0,T)} \leq \left|\Delta^{\infty}_{LSLE}(T) - \Delta^{\Theta_m}_{LSLE}(T)\right| \overset{m\to\infty}{\longrightarrow} 0,$$

which finally shows uniform convergence and continuity of $\Delta^{\infty}_{LSLE}(T)$. $\square$

The crucial assumption here is that the first estimation Θ_1 is small enough, so that for some time t and expansion center $\boldsymbol{x}_i$ the minimum secant gradient point $r_{d_i(t)}$ is not contained in $\overline{B_{\Theta_1(t)}\left(d_i(t)\right)}$, i.e. $\Gamma_1 \neq \emptyset$ holds to enable (5.19, p.154). This is reasonable in the context of kernel-based model reduction, since the expansion centers are naturally scattered over the high dimensional state space.

5.1.3 Parameterized systems with inputs

In this section, we shall investigate extensions of the derived error estimators for more complex dynamical systems. The class of dynamical systems we consider is given by the class of affine-parametric systems as introduced in (1.12, p.19), motivated by [74]:

$$\boldsymbol{x}'(t) = \boldsymbol{f}(\boldsymbol{x}(t), t, \boldsymbol{\mu}) + \boldsymbol{B}(t, \boldsymbol{\mu})\boldsymbol{u}(t), \tag{5.21a}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_0(\boldsymbol{\mu}), \tag{5.21b}$$

$$\boldsymbol{w}(t) = \boldsymbol{C}(t, \boldsymbol{\mu})\boldsymbol{x}(t). \tag{5.21c}$$

Note here that we omit any linear part $\boldsymbol{A}$ in this section due to historical reasons, however, it can be readily included in any of the derived error estimators of Section 5.1 by the means introduced in Section 5.2. Recall here the parameter domain $\mathcal{P} \subseteq \mathbb{R}^p$ and parameters $\boldsymbol{\mu} \in \mathcal{P}$, where we will suppress the $\boldsymbol{\mu}$-dependencies where clear from context and otherwise indicate it by an extra semicolon-separated $\boldsymbol{\mu}$ argument for simplicity of notation. Further we redefine the initial error as

$$E_0(\boldsymbol{\mu}) := \left|\left|\left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\boldsymbol{x}_0(\boldsymbol{\mu})\right|\right|_G .$$

As we have a potentially time- and parameter-dependent nonlinearity, we assume the kernel approximation to be given as

$$\hat{\boldsymbol{f}}(\boldsymbol{x}, t, \boldsymbol{\mu}) = \sum_{i=1}^{n} \boldsymbol{c}_i K(\boldsymbol{x}_i, \boldsymbol{x}) K_t(t_i, t) K_{\mathcal{P}}(\boldsymbol{\mu}_i, \boldsymbol{\mu}), \tag{5.22}$$

alike (3.9, p.98) in Section 3.2.1. Following the lines of Theorem 5.1.1, we can state a global Lipschitz constant estimator also for this system.

Theorem 5.1.11 (Global Lipschitz Estimator). *Let the dynamical system* (5.21) *be given. Further, let the state space kernel K be L_K-Lipschitz continuous with respect to the first variable. Then $\forall\, \boldsymbol{\mu} \in \mathcal{P}$ the state space error is bounded via*

$$||\boldsymbol{e}(t; \boldsymbol{\mu})||_G \leq \Delta_{GLE}(t, \boldsymbol{\mu}) \;\; \forall\, t \in [0, T],$$

with

$$\Delta_{GLE}(t, \boldsymbol{\mu}) := \int_0^t \alpha(s, \boldsymbol{\mu}) e^{\int_s^t \beta(r, \boldsymbol{\mu}) dr} ds + e^{\int_0^t \beta(s, \boldsymbol{\mu}) ds} E_0(\boldsymbol{\mu})$$

and

$$\begin{aligned} \alpha(t, \boldsymbol{\mu}) &:= ||E_A(\boldsymbol{x}(t); \boldsymbol{\mu})||_G \\ &\qquad + \left|\left| \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right) \left(\hat{\boldsymbol{f}}(\boldsymbol{x}^r(t), t, \boldsymbol{\mu}) + \boldsymbol{B}(t, \boldsymbol{\mu})\boldsymbol{u}(t)\right) \right|\right|_G, \\ \beta(t, \boldsymbol{\mu}) &:= L_K \sum_{i=1}^{n} ||\boldsymbol{c}_i||_G \, |K_t(t, t_i) K_{\mathcal{P}}(\boldsymbol{\mu}, \boldsymbol{\mu}_i)|. \end{aligned}$$

Proof. Using the kernel expansion structure we directly obtain

$$\begin{aligned} &\left|\left| \hat{\boldsymbol{f}}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{x}^r(t)) \right|\right|_G \\ &\leq \sum_{i=1}^{n} ||\boldsymbol{c}_i||_G \left| K(\boldsymbol{x}(t), \boldsymbol{x}_i) - K(\boldsymbol{x}^r(t), \boldsymbol{x}_i) \right| |K_t(t, t_i) K_{\mathcal{P}}(\boldsymbol{\mu}, \boldsymbol{\mu}_i)| \\ &\leq L \sum_{i=1}^{n} ||\boldsymbol{c}_i||_G \, |K_t(t, t_i) K_{\mathcal{P}}(\boldsymbol{\mu}, \boldsymbol{\mu}_i)| \, ||\boldsymbol{e}(t)||_G \end{aligned}$$

$$= \beta(t, \boldsymbol{\mu}) \left|\left|\boldsymbol{e}(t;\boldsymbol{\mu})\right|\right|_G .$$

The existence of inputs causes an additional term $\left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\boldsymbol{B}(t,\boldsymbol{\mu})\boldsymbol{u}(t)$ occurring in the error system (5.6a, p.141). Now, performing the same steps as in the proof of Theorem 5.1.1 shows

$$\begin{aligned}
\left|\left|\boldsymbol{e}'(t;\boldsymbol{\mu})\right|\right|_G &= \Big|\Big|\boldsymbol{f}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{x}(t)) + \hat{\boldsymbol{f}}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{x}^r(t)) \\
&\qquad + \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\hat{\boldsymbol{f}}(\boldsymbol{x}^r(t) + \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\boldsymbol{B}(t,\boldsymbol{\mu})\boldsymbol{u}(t)\Big|\Big|_G \\
&\leq \left|\left|E_A(\boldsymbol{x}(t);\boldsymbol{\mu})\right|\right|_G + \left|\left|\hat{\boldsymbol{f}}(\boldsymbol{x}(t)) - \hat{\boldsymbol{f}}(\boldsymbol{x}^r(t))\right|\right|_G \\
&\quad + \left|\left|\left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\left(\hat{\boldsymbol{f}}(\boldsymbol{x}^r(t) + \boldsymbol{B}(t,\boldsymbol{\mu})\boldsymbol{u}(t)\right)\right|\right|_G \\
&\leq \beta(t,\boldsymbol{\mu}) \left|\left|\boldsymbol{e}(t;\boldsymbol{\mu})\right|\right|_G + \alpha(t,\boldsymbol{\mu}).
\end{aligned}$$

Finally, application of the comparison lemma for the ODE

$$\Delta'(t) = \beta(t,\boldsymbol{\mu})\Delta(t) + \alpha(t,\boldsymbol{\mu}), \qquad\qquad \Delta(0) = E_0(\boldsymbol{\mu}),$$

yields the error estimator as analytical solution. □

Of course, the improved local Lipschitz constant estimation can be applied for this type of systems. We state all involved computational steps here for completeness.

Theorem 5.1.12 (**L**ocal **S**ecant gradient **L**ipschitz error **E**stimator (LSLE)). *Let the dynamical system (5.21, p.157) be given, and let the state space kernel K of $\hat{\boldsymbol{f}}$ be induced by a bell function ϕ. Further, let $\Theta(t,\boldsymbol{\mu})$ be an a-priori error bound. Then the state space error is bounded via*

$$\left|\left|\boldsymbol{e}(t)\right|\right|_G \leq \Delta^{\Theta}_{LSLE}(t) \quad \forall\, t \in [0,T],$$

with

$$\Delta^{\Theta}_{LSLE}(t) := \int\limits_0^t \alpha(s) e^{\int_s^t \beta(r)dr} ds + e^{\int_0^t \beta(s)ds} E_0,$$

$$\begin{aligned}\alpha(t,\boldsymbol{\mu}) &:= ||E_A(\boldsymbol{x}(t);\boldsymbol{\mu})||_G \\ &\quad + \left|\left|\left(\boldsymbol{I}-\boldsymbol{V}\boldsymbol{W}^T\right)\left(\hat{\boldsymbol{f}}(\boldsymbol{x}^r(t),t,\boldsymbol{\mu})+\boldsymbol{B}(t,\boldsymbol{\mu})\boldsymbol{u}(t)\right)\right|\right|_G, \\ \beta(t) &:= \sum_{i=1}^n L_\Theta(d_i(t))\,||\boldsymbol{c}_i||_G\,|K_t(t,t_i)K_{\mathcal{P}}(\boldsymbol{\mu},\boldsymbol{\mu}_i)|.\end{aligned}$$

Here we have $L_\Theta(d_i(t)) := \left|S_{d_i(t)}(r_{\Theta,i})\right|$, where we define for

$$\Gamma(t,\boldsymbol{\mu}) = \overline{B_{\Theta(t,\boldsymbol{\mu})}\left(d_i(t)\right)} \cap \mathbb{R}_+$$

the quantities

$$\begin{aligned} r_{\Theta,i} &:= \begin{cases} d_i(t)+\operatorname{sign}\left(r_i-d_i(t)\right)\Theta(t,\boldsymbol{\mu}), & r_i \notin \Gamma(t,\boldsymbol{\mu}), \\ r_i, & r_i \in \Gamma(t,\boldsymbol{\mu}), \end{cases} \\ r_i &:= \arg\min_{r\in\mathbb{R}_+} S_{d_i(t)}(r) = \arg\min_{r\in\mathbb{R}_+} \frac{\phi(d_i(t))-\phi(r)}{d_i(t)-r}, \end{aligned}$$

similar to Proposition 5.1.7.

Proof. The derivation follows the proof of Theorem 5.1.11. The α term is obtained exactly the same way, and using Corollary 5.1.8 we see that

$$\begin{aligned} &\left|\left|\hat{\boldsymbol{f}}(\boldsymbol{x}(t))-\hat{\boldsymbol{f}}(\boldsymbol{x}^r(t))\right|\right|_G \\ &\leq \sum_{i=1}^n ||\boldsymbol{c}_i||_G\left|K(\boldsymbol{x}(t),\boldsymbol{x}_i)-K(\boldsymbol{x}^r(t),\boldsymbol{x}_i)\right||K_t(t,t_i)K_{\mathcal{P}}(\boldsymbol{\mu},\boldsymbol{\mu}_i)| \\ &\leq \sum_{i=1}^n \left|S_{d_i(t)}(r_{\Theta,i})\right|\,||\boldsymbol{e}(t;\boldsymbol{\mu})||_G\,||\boldsymbol{c}_i||_G\,|K_t(t,t_i)K_{\mathcal{P}}(\boldsymbol{\mu},\boldsymbol{\mu}_i)| \\ &= \beta(t,\boldsymbol{\mu})\,||\boldsymbol{e}(t;\boldsymbol{\mu})||_G\,. \end{aligned}$$

As usual, application of Lemma 5.0.1 finishes the proof. □

Of course, the iterative scheme presented in Section 5.1.2 can be applied here, too. We see in the $\alpha(t,\boldsymbol{\mu})$ term that if any input $\boldsymbol{u}(t)$ happens to maintain the full system in the subspace $\mathcal{V}$, then the resulting zero approximation error will be verified *a-posteriori* by both the GLE and LSLE error estimators.

5.1.3.1 Output error estimation

Once a bound $\Delta(t, \boldsymbol{\mu})$ for the state error is available the output error

$$\begin{aligned} \boldsymbol{e}_w(t;\boldsymbol{\mu}) &:= \boldsymbol{w}(t) - \tilde{\boldsymbol{C}}(t,\boldsymbol{\mu})\boldsymbol{z}(t) \\ &= \boldsymbol{C}(t,\boldsymbol{\mu})\boldsymbol{x}(t) - \boldsymbol{C}(t,\boldsymbol{\mu})\boldsymbol{V}\boldsymbol{z}(t) = \boldsymbol{C}(t,\boldsymbol{\mu})\boldsymbol{e}(t;\boldsymbol{\mu}) \end{aligned}$$

can be bounded by

$$||\boldsymbol{e}_w(t;\boldsymbol{\mu})|| \leq C_o(t,\boldsymbol{\mu})\,||\boldsymbol{e}(t;\boldsymbol{\mu})||_G \leq C_o(t,\boldsymbol{\mu})\Delta(t,\boldsymbol{\mu}), \quad t \in [0,T],$$

using the G-induced matrix norm

$$C_o(t,\boldsymbol{\mu}) := ||\boldsymbol{C}(t,\mu)||_G = \sup_{\boldsymbol{x}\in\mathbb{R}^d} \frac{||\boldsymbol{C}(t,\boldsymbol{\mu})\boldsymbol{x}||_G}{||\boldsymbol{x}||_G}.$$

Note that this can further be estimated using the affine structure as

$$\begin{aligned} \sup_{\boldsymbol{x}\in\mathbb{R}^d} \frac{||\boldsymbol{C}(t,\boldsymbol{\mu})\boldsymbol{x}||_G}{||\boldsymbol{x}||_G} &\leq \sup_{\boldsymbol{x}\in\mathbb{R}^d} \frac{\sum_{i=1}^{Q_C} \left|\theta_i^C(t,\boldsymbol{\mu})\right| ||\boldsymbol{C}_i\boldsymbol{x}||_G}{||\boldsymbol{x}||_G} \\ &\leq \sum_{i=1}^{Q_C} \left|\theta_i^C(t,\boldsymbol{\mu})\right| ||\boldsymbol{C}_i||_G \,. \end{aligned}$$

5.1.4 Computational aspects

Before we will present some experiments regarding the derived error estimators, we will discuss certain computational aspects. In detail, the error estimators can be decomposed in an offline/online fashion as shown in Section 5.1.4.1 and the maximum secant gradient locations from Proposition 5.1.7 can be efficiently computed using a penalized Newton iteration presented in Section 5.1.4.2.

5.1.4.1 Offline/online decomposition

Additionally to the steps regarding the offline/online decomposition of the dynamical system described in Section 3.3, we will detail how to decompose the derived error estimators of the previous sections in the same fashion similar to [181, 74]. Since the LSLE variant for parameterized systems of Theorem 5.1.12 includes all previously considered quantities, we shall detail the decomposition for that case.

From the proofs of the error estimator theorems we see that the comparison Lemma 5.0.1 plays a key role to obtain an explicit expression for the error estimate. On the other hand, it also shows that any estimator can be obtained by solving a small auxiliary ODE, which can in fact be computed "on the fly" with the reduced simulation by adding an extra dimension to the reduced system (3.4, p.88). As defined in Section 1.3.2, (1.12, p.19), we assume $\boldsymbol{B}$ and $\boldsymbol{x}_0$ to be given as

$$\boldsymbol{B}(t,\boldsymbol{\mu}) = \sum_{i=1}^{Q_B} \theta_i^B(t,\boldsymbol{\mu})\boldsymbol{B}_i, \qquad \boldsymbol{x}_0(\boldsymbol{\mu}) = \sum_{i=1}^{Q_0} \theta_i^0(\boldsymbol{\mu})\boldsymbol{x}_i^0.$$

Now, essentially the offline stage prepares the efficient computation of the $\alpha(t,\boldsymbol{\mu})$ term at the online stage.

Offline stage Let $\boldsymbol{M} = [\boldsymbol{c}_1 \dots \boldsymbol{c}_N] \in \mathbb{R}^{d\times n}$ and recall the norm inducing matrix $\boldsymbol{G}$ from the introduction of Chapter 1. Further we see that

$$\begin{aligned} d_i(t) &= ||\boldsymbol{x}^r(t) - \boldsymbol{x}_i||_G \\ &= \sqrt{||\boldsymbol{V}\boldsymbol{z}(t)||_G^2 - 2\boldsymbol{z}(t)^T\boldsymbol{V}^T\boldsymbol{G}\boldsymbol{x}_i + ||\boldsymbol{x}_i||_G^2} \\ &= \sqrt{||\boldsymbol{z}(t)||_G^2 - 2\boldsymbol{z}(t)^T\tilde{\boldsymbol{z}}_i + ||\boldsymbol{x}_i||_G^2}, \end{aligned}$$

where $\tilde{\boldsymbol{z}}_i := \boldsymbol{V}^T\boldsymbol{G}\boldsymbol{x}_i$ and $||\boldsymbol{x}_i||_G^2, i = 1\dots n$ can be precomputed. This together with $||\boldsymbol{c}_i||_G,\ i = 1\dots n$ constitutes the offline-computations for the $\beta(t,\boldsymbol{\mu})$ term and

$$\begin{aligned}
\tilde{\boldsymbol{x}}_i^0 &:= \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right) x_i^0, && i = 1\dots Q_0,\\
\tilde{\boldsymbol{x}}_{ij}^0 &:= (\tilde{\boldsymbol{x}}_i^0)^T \boldsymbol{G}\tilde{\boldsymbol{x}}_j^0, && i,j = 1\dots Q_0,\\
\boldsymbol{M}_1 &:= \tilde{\boldsymbol{M}}^T \boldsymbol{G}\tilde{\boldsymbol{M}} \in \mathbb{R}^{n\times n} && \tilde{\boldsymbol{M}} = \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\boldsymbol{M},\\
\tilde{\boldsymbol{B}}_i &:= \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right) B_i, && i = 1\dots Q_B,\\
\boldsymbol{M}_{2,i} &:= \tilde{\boldsymbol{M}}^T \boldsymbol{G}\tilde{\boldsymbol{B}}_i \in \mathbb{R}^{n\times l}, && i = 1\dots Q_B,\\
\boldsymbol{M}_{3,ij} &:= \tilde{\boldsymbol{B}}_i^T \boldsymbol{G}\tilde{\boldsymbol{B}}_j \in \mathbb{R}^{l\times l}, && i,j = 1\dots Q_B,
\end{aligned}$$

comprises the computations for intermediate small matrices for the α term.

Online stage The online part consists of computing the ODE

$$\Delta'(t) = \alpha(t,\boldsymbol{\mu}) + \beta(t,\boldsymbol{\mu})\Delta(t), \qquad \Delta(0) = \sqrt{\sum_{i,j}^{Q_0} \theta_i^0(\boldsymbol{\mu})\theta_j^0(\boldsymbol{\mu})\tilde{\boldsymbol{x}}_{ij}^0},$$

with

$$\begin{aligned}
\boldsymbol{k}(t) &:= \left(K^r(\boldsymbol{z}(t),\boldsymbol{z}_i)K_t(t,t_i)K_{\mathcal{P}}(\boldsymbol{\mu},\boldsymbol{\mu}_i)\right)_{i=1}^n \in \mathbb{R}^n,\\
\boldsymbol{M}_2(t,\boldsymbol{\mu}) &:= \sum_{i=1}^{Q_B} \theta_i^B(t,\boldsymbol{\mu})\boldsymbol{M}_{2,i},\\
\boldsymbol{M}_3(t,\boldsymbol{\mu}) &:= \sum_{i,j=1}^{Q_B} \theta_i^B(t,\boldsymbol{\mu})\theta_j^B(t,\boldsymbol{\mu})\boldsymbol{M}_{3,ij},\\
\alpha(t,\boldsymbol{\mu}) &= \Big(\boldsymbol{k}(t)^T\boldsymbol{M}_1\boldsymbol{k}(t) + 2\boldsymbol{k}(t)^T\boldsymbol{M}_2(t,\boldsymbol{\mu})\boldsymbol{u}(t)\\
&\qquad + \boldsymbol{u}(t)^T\boldsymbol{M}_3(t,\boldsymbol{\mu})\boldsymbol{u}(t)\Big)^{\frac{1}{2}}.
\end{aligned}$$

The α-decomposition follows directly from the definitions. Notice that if there are no inputs $\boldsymbol{u}(t)$ for a given system only the constant matrix $\boldsymbol{M}_1$ has to be computed as $\boldsymbol{M}_2$ and $\boldsymbol{M}_3$ do not appear. However, all the offline matrices $\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{M}_3$ are small matrices only depending on Q_B, n and l.

5.1.4.2 Penalized Newton iteration

When using the improved local Lipschitz constant estimators from Theorem 5.1.9 and/or Theorem 5.1.12, another important aspect for fast online evaluation is efficient computation of the $\beta(t,\boldsymbol{\mu})$ term. In the setting of a reduced simulation this term reads as

$$\begin{aligned}
\beta(t,\boldsymbol{\mu}) &= \sum_{i=1}^{n} ||\boldsymbol{c}_i||_G \left|S_{d_i(t)}(r_{\Theta(t,\boldsymbol{\mu}),d_i(t)})\right|, \\
r_{\Theta(t,\boldsymbol{\mu}),d_i(t)} &= \begin{cases} d_i(t) + \operatorname{sign}\left(d_i(t) - r_{d_i(t)}\right)\Theta(t,\boldsymbol{\mu}), & r_{d_i(t)} \notin \overline{B_{\Theta(t,\boldsymbol{\mu})}\left(d_i(t)\right)} \\ r_{d_i(t)}, & r_{d_i(t)} \in \overline{B_{\Theta(t,\boldsymbol{\mu})}\left(d_i(t)\right)} \end{cases}, \\
r_{d_i(t)} &= \arg\min_{r\in\mathbb{R}_+} \frac{\phi(r)-\phi(d_i(t))}{r-d_i(t)}.
\end{aligned}$$

So, in theory, n small minimization problems need to be solved at any time $t \in [0,T]$ during integration, whereas in practice, Lemma 5.1.5 allows to reduce the amount of problems to be solved.

For example, say $d_i(t) < r_0$ for a $t \in [0,T],\ i \in \{1\dots N\}$. Then condition (5.13a, p.148) yields $r_0 \leq r_{d_i(t)}$. So, if $d_i(t) + \Theta(t,\boldsymbol{\mu}) < r_0$ there is no need to compute $r_{d_i(t)}$ as $r_{\Theta(t,\boldsymbol{\mu}),d_i(t)} = d_i(t) + \Theta(t,\boldsymbol{\mu})$ anyways. However, in Proposition 5.1.15 we propose using a penalized Newton iteration scheme for the cases an efficient computation of the minimizer is required. As the reduced state variable $\boldsymbol{z}(t)$ changes continuously, so does $r_{d_i(t)}$, which keeps the number of Newton iterations for subsequent (next time-step) computations small when using the old values as starting point for the new iterations.

We first state some preparatory lemma before we stating our Newton iteration scheme in Proposition 5.1.15.

Lemma 5.1.13. *Let ϕ be a bell function. Then the mapping*

$$\begin{aligned}
\eta(s) : \mathbb{R}_+ &\longrightarrow \mathbb{R}_+ \\
s &\longmapsto r_s = \arg\min_{r\in\mathbb{R}_+} S_s(r)
\end{aligned}$$

is monotonically decreasing.

Proof. Note here that η is well-defined by Lemma 5.1.5. Choose some $s_1, s_2 \in \mathbb{R}_+$ with $s_1 \neq s_2$. Without loss of generality assume $s_1 < s_2$. Let $r_i := \eta(s_i)$, $i \in \{1, 2\}$. At first consider the case $s_1 < s_2 \leq r_0$. Define $l(r) := \phi(s_1) + \phi'(r_1)(r - s_1)$. Then we see that $l(s_2) \leq \phi(s_2)$, as

$$\frac{\phi(s_2) - \phi(s_1)}{s_2 - s_1} \geq \frac{\phi(r_1) - \phi(s_1)}{r_1 - s_1} = \phi'(r_1) = \frac{l(s_2) - \phi(s_1)}{s_2 - s_1}.$$

Using $\phi'(r_1) = \frac{\phi(r_1)-\phi(s_1)}{r_1-s_1}$, l can also be written as $l(r) = \phi(r_1) + \phi'(r_1)(r - r_1)$. So we conclude

$$\begin{aligned}\phi'(r_2) = \frac{\phi(r_2) - \phi(s_2)}{r_2 - s_2} &\leq \frac{\phi(r_1) - \phi(s_2)}{r_1 - s_2} \\ &\leq \frac{\phi(r_1) - l(s_2)}{r_1 - s_2} = \frac{\phi(r_1) - \phi(s_1)}{r_1 - s_1} = \phi'(r_1),\end{aligned}$$

and since $\phi(x)$ is convex for $x > r_0$ we must have $r_1 \geq r_2$. The proof for the case $r_0 \leq s_1 < s_2$ is analogous using the concavity on $[0, r_0]$. □

Lemma 5.1.14 (Bounds of r_s). *Let ϕ be a bell function. Then we have*

$$0 \leq r_s < r_m \; \forall \, s \in \mathbb{R}^+ \qquad \textit{for} \qquad r_m := \frac{\phi(0) r_0}{\phi(0) - \phi(r_0)}.$$

Proof. Fix an $s \in \mathbb{R}^+$. We already know that $0 \leq r_s$ by definition. At first we see that

$$r_m = r_0 \frac{\phi(0)}{\phi(0) - \phi(r_0)} > r_0 \frac{\phi(0)}{\phi(0)} = r_0.$$

So, if $s > r_0$ we have $r_s \leq r_0 < r_m$ by Lemma 5.1.5, (5.13b, p.148). Obviously, for $s = r_0$ we have $r_s = r_0 < r_m$. This leaves us with the case $s < r_0$, i.e. $r_0 \leq r_s$ by (5.13a, p.148). Let $l_s(r) := \phi(r_0) + S_s(r_0)(r - r_0)$ be the line through $\phi(r_0)$ with gradient $S_s(r_0)$. Since $S_s(r_0) < 0$ and $l_s(r_0) = \phi(r_0) > 0$ we know that

$$\exists! \; \eta_s \in \mathbb{R}_+ : \; l_s(\eta_s) = 0.$$

Also, as $0 > S_s(r_0) > \phi'(r_0)$ we have

$$\exists\, \delta > 0 : \; \phi(r_0 + \delta) < l_s(r_0 + \delta).$$

But as $\phi(\eta_s) > 0$, $l_s(\eta_s) = 0$ and $\phi - l_s$ is continuous we know that

$$\exists\, t \in \,]r_0, \eta_s[\, : \; \phi(t) = l_s(t).$$

Further we have

$$\begin{aligned} S_{r_0}(t) &= \frac{\phi(t) - \phi(r_0)}{t - r_0} = \frac{l_s(t) - \phi(r_0)}{t - r_0} \\ &= \frac{\phi(r_0) + S_s(r_0)(t - r_0) - \phi(r_0)}{t - r_0} = S_s(r_0). \end{aligned}$$

Since ϕ is convex on $[r_0, \infty[$, we also know that

$$\exists!\, \xi \in]r_0, t[\, : \; \phi'(\xi) = S_{r_0}(t).$$

With the minimizing property of r_s we see that

$$\phi'(\xi) = S_{r_0}(t) = S_s(r_0) > S_s(r_s) = \phi'(r_s),$$

but this means $r_s < \xi$ by the strict convexity of ϕ on $[r_0, \infty[$. Finally, as $s \in \mathbb{R}_+$ was arbitrary, we obtain $r_s < \xi < t < \eta_s \quad \forall\, s \in \mathbb{R}_+$. Now Lemma 5.1.13 yields

$$r_s \leq r_{s=0} = \eta(0) < \eta_0 \quad \forall\, s \in \mathbb{R}_+,$$

and solving $l_0(\eta_0) = 0$ gives the desired bound $r_m = \eta_0$. $\square$

Proposition 5.1.15 (Penalized Newton iteration). *Let ϕ be a bell function. Define the unpenalized objective function as $n(r) := \phi'(r) - S_s(r)$ and let*

$$r_m := \frac{\phi(0) r_0}{\phi(0) - \phi(r_0)}, \qquad c(\nu) := \frac{n'(\nu)^2}{4n(\nu)}, \qquad d(\nu) := 2\frac{n(\nu)}{n'(\nu)} - \nu.$$

Then $r_s = \arg\min\limits_{r \in \mathbb{R}_+} \frac{\phi(r) - \phi(s)}{r - s}$ can be computed as root of the penalized objective *function n_p given by*

$$s \leq r_0: \qquad n_p(r) := \begin{cases} c(r_0)(r + d(r_0))^2, & r, \leq r_0, \\ n(r), & r_0 < r < r_m, \\ c(r_m)(r + d(r_m))^2, & r \geq r_m, \end{cases}$$

$$r_0 < s: \qquad n_p(r) := \begin{cases} c(0)(r + d(0))^2, & r \leq 0, \\ n(r), & 0 < r < r_0, \\ c(r_0)(r + d(r_0))^2, & r \geq r_0. \end{cases}$$

Proof. At first consider the penalty polynomials and define $p_\nu(x) = c(\nu)(x + d(\nu))^2$. Then it is easy to validate that $p_\nu(\nu) = n(\nu), p'_\nu(\nu) = n'(\nu)$. Those polynomials will be used to extend n at appropriate points. Next, existence and uniqueness of r_s have already been shown in Lemma 5.1.5. At first consider the case $s < r_0$. As the necessary condition for a minimizer $r \geq r_0$ we have $0 = S'_s(r) = \frac{\phi'(r) - S_s(r)}{r - s}$, and since $s < r_s$ by Lemma 5.1.5, $0 = n(r) = \phi'(r) - S_s(r)$ is also sufficient for $r \geq r_0$. Since $n(s) = \phi'(s) - S_s(s) = 0$, we replace $n(r)$ for $r < r_0$ by $p_{r_0}(r)$. Even though $S'_s(r) > 0 \;\forall\; r > r_s$ by Lemma 5.1.6 and hence $n(r) \neq 0 \;\forall\; r > r_s$ we see that $\lim_{r \to \infty} n(r) = 0$. To avoid the Newton iteration being drawn to ∞ we replace $n(r)$ by p_{r_m} for $r \geq r_m$ as we know that $r_s < r_m \;\forall\; s \in \mathbb{R}_+$ from Lemma 5.1.14. For $s > r_0$ we have $0 \leq r_s$ and so we replace $n(r)$ by p_0 for $r \leq 0$ and $n(r)$ by p_{r_0} for $r_0 \leq r$ since again $n(s) = 0$. Of course, $s = r_0$ means $r_s = r_0$ and no iterations have to be performed. $\square$

5.1.5 Numerical experiments

In this part we present numerical experiments for the error estimators developed in the last sections. As discussed in the beginning of Section 5.1, there is still need for practical a-priori or a-posteriori estimates for the nonlinearity approximation error E_A (5.7, p.142). However, as the main focus of the last part was on efficient estimation of local Lipschitz constants, we shall employ synthetic dynamical systems using kernel expansions as nonlinearity. This allows to pursue experiments for systems that do not have an approximation part $\hat{\boldsymbol{f}}$ and hence E_A.

Before we state the experimental setup, we introduce two more error estimator modifications. First, the convergence result from Theorem 5.1.10 motivates a heuristic variant of our local estimators. This originates from the fact that the limit function Δ^{∞}_{LSLE} reproduces itself when used as a-priori bound within the iterations. Numerically (up to the integration error) this can be exploited using the error estimate from the previous time step as bound at the current time step. We will refer to this time-discrete method by “LSLE TD”. Experiments indicate that this variant indeed seems to bound the iterated LSLE estimators from below. Second, when setting

$$\begin{aligned} & \beta(t) = \frac{||\boldsymbol{f}(\boldsymbol{x}(t)) - \boldsymbol{f}(\boldsymbol{x}^r(t))||_G}{||\boldsymbol{x}(t) - \boldsymbol{x}^r(t)||_G} \\ \text{or} \qquad & \beta(t,\boldsymbol{\mu}) = \frac{||\boldsymbol{f}(\boldsymbol{x}(t),t,\boldsymbol{\mu}) - \boldsymbol{f}(\boldsymbol{x}^r(t),t,\boldsymbol{\mu})||_G}{||\boldsymbol{x}(t) - \boldsymbol{x}^r(t)||_G}, \end{aligned}$$

depending on the considered case, we obtain the smallest possible estimation for this estimator structure, since this is the best local Lipschitz constant at any $t \in [0, T]$. As this version requires $\boldsymbol{x}(t)$ it is not considered a practical error estimator but rather a comparative “Lower Bound”.

Our first test environment aims at a system without parameters but with additional input. Let $d = 240000$ in order to represent a large-scale system, $T = 20$, $\boldsymbol{G} = \boldsymbol{I}_d$. The kernel expansion (5.8, p.142) uses $N = 20$ and $(\boldsymbol{x}_i)_j := \frac{50}{N-1}(i-1),\ i = 1\dots N,\ j = 1\dots d$. The kernel used is a Gaussian $K(\boldsymbol{x},\boldsymbol{y}) = \exp(-\left\|\boldsymbol{x}-\boldsymbol{y}\right\|^2/\gamma^2)$ with $\gamma = 224$, which is chosen to have

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j) < 10^{-5}\ \forall\ |i-j| \geq 2.$$

This way a certain locality of the kernel expansion is ensured. Finally, the expansion coefficient vectors are given via

$$(\boldsymbol{c}_i)_j = \exp(-(\boldsymbol{x}_i)_j/15),\ i = 1 \dots N,\ j = 1 \dots d.$$

Further we set $x_0 = (1 \dots 1)^T \in \mathbb{R}^d$ and $B = (1 \dots 1)^T \in R^{d \times 1}$. To emphasize the input-dependent behavior of our estimators, we choose $l = 1$ and two system inputs

$$u_1(t) = \frac{1}{25} \sin\left(\frac{t}{3}\right), \qquad u_2(t) = \frac{1}{2} e^{-(12-t)^2},$$

which represent one oscillating and one localized input. We use an explicit Euler scheme with time-step $\Delta t = 0.05$ as solver.

In order to investigate the error estimator behavior, we decided to have means of controlling the projection subspace quality. Since the system setup is homogeneous in each component, we would obtain zero approximation error for any input using $\hat{\boldsymbol{V}} = \hat{\boldsymbol{W}} = (1, \dots, 1)^T/\sqrt{d}$ as projection matrices. Then, for a given degree θ we set the rotated subspace projection matrices to $\boldsymbol{V} := \boldsymbol{R}\hat{\boldsymbol{V}},\ W := \boldsymbol{R}\hat{\boldsymbol{W}}$ for an orthogonal block matrix $\boldsymbol{R} := \operatorname{diag}\left(\hat{\boldsymbol{R}}, \boldsymbol{I}_{d-200}\right)$ with

$$\hat{\boldsymbol{R}} := \operatorname{diag}(\boldsymbol{R}_2, \dots, \boldsymbol{R}_2) \in \mathbb{R}^{200 \times 200}, \qquad \boldsymbol{R}_2 = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}.$$

This way, θ continuously controls the quality of the projection subspace. The results for the different error estimators can be seen in Figure 5.2. One immediately notices the large exponential growth rates of both the GLE and LSLE variant (without iterations). As we do not pose any assumptions on the stability of the considered systems, those rates are a necessary drawback of rigorous error estimators. However, compared to the GLE, the LSLE has a significantly lower exponential increase rate, which again is reduced drastically to a useful level when using estimator iterations or the LSLE TD variant.

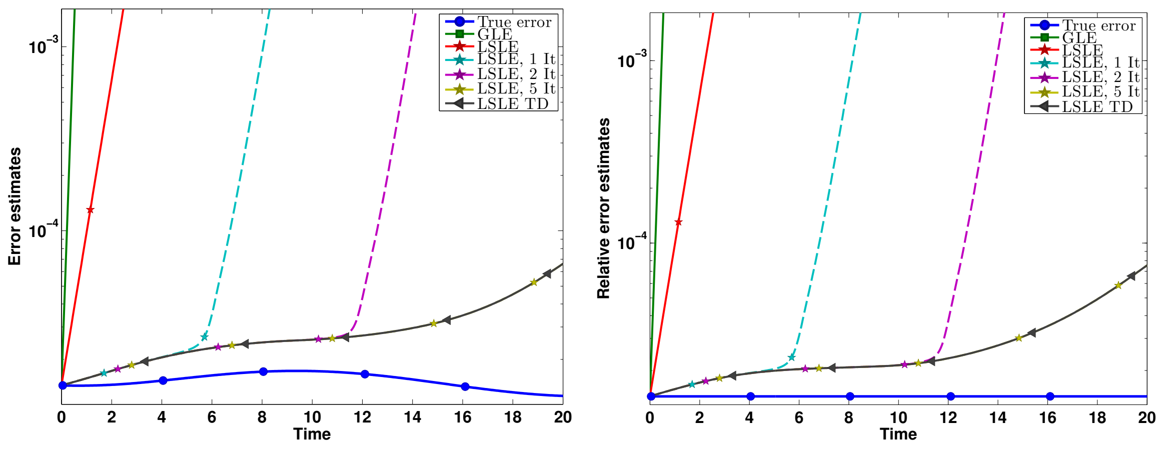


Figure 5.2: Absolute (left) and relative (right) error estimates using $\theta = .0005$ and input u_1

Note here that the main results are the iterable LSLE estimator and the LSLE TD variant; the other estimators are included for comparison purposes.

The rigorosity of all estimators is verified as all are upper bounds of the true error. We also see that the LSLE TD variant is a tight lower bound for the iterated LSLE estimator (LSLE, 5 It and LSLE TD are indistinguishable) which suggests that the LSLE TD variant is the numerical pendant of the iteration limit Δ^{∞}_{LSLE}. Due to the very good performance in both computational cost and estimation sharpness this is the best estimator to use in practice. Furthermore, the LSLE TD variant grows with the same rate as the "Lower Bound" estimation up to about $t = 16$, which shows the effectiveness of the local Lipschitz constants together with the a-priori bounds. We also see that iterations of the LSLE estimator improve the error bound, but e.g. the first and second LSLE iterations fall back to their specific increase rate after a certain time. This occurs when the a-priori bound is too big to have a positive effect regarding the choice (5.15, p.151) from Proposition 5.1.7.

The left image in Figure 5.3 shows the computation time plotted against the error estimate for each estimator variant. In one extreme the full error computation takes about 20s (top left) but is of course the sharpest. The green square in the lower right corner denotes the GLE estimator, which is cheap (by computation-time) but too large to be usable. Now, comparing identical star-symbols, we see that estimator iterations also come with higher computational costs while improving. Hence, the iteration number can be used as a balancing parameter between online run-time and error bound accuracy. The right hand image of Figure 3 shows the output using u_2 and $\theta = 0.05$. We see that it is nicely bounded by the LSLE-TD estimator; note that the reduced and full models' outputs are indistinguishable in this plot. Identical conclusions can be drawn by using either input for Figures 5.2 and 5.3.

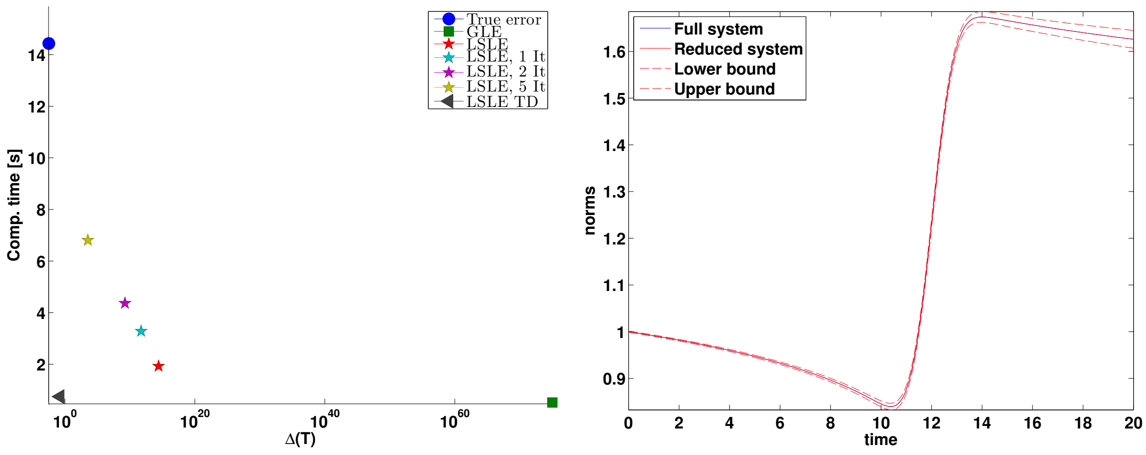


Figure 5.3: Left: Computation times and output for u_3 and $\theta = .05$. Right: The output error bounded by the LSLE TD variant

Table 5.1 shows the estimated errors at $T = 20$ for different θ values and both inputs. Albeit the true error is of the same magnitude for both inputs

Model	$\|\|\boldsymbol{e}(T)\|\|_G$	GLE	LSLE	LSLE 2 It	LSLE 5 It	LSLE TD	Low. b.
$\theta = 0.0005, u_1$	$1.3E{-}5$	$1.1E{+}73$	$3.2E{+}11$	$1.4E{+}2$	$6.6E{-}5$	$6.6E{-}5$	$2.2E{-}5$
$\theta = 0.005, u_1$	$1.3E{-}4$	$1.1E{+}74$	$3.2E{+}12$	$1.2E{+}5$	$6.7E{-}4$	$6.6E{-}4$	$2.2E{-}4$
$\theta = 0.05, u_1$	$1.3E{-}3$	$1.1E{+}75$	$3.2E{+}13$	$1.2E{+}8$	$2.3E{+}0$	$6.7E{-}3$	$2.2E{-}3$
$\theta = 0.5, u_1$	$1.3E{-}2$	$1.1E{+}76$	$3.1E{+}14$	$1.0E{+}11$	$9.2E{+}5$	$8.0E{-}2$	$2.2E{-}2$
$\theta = 0.0005, u_2$	$2.3E{-}5$	$1.1E{+}73$	$8.5E{+}11$	$2.3E{+}3$	$1.8E{-}4$	$1.8E{-}4$	$3.0E{-}5$
$\theta = 0.005, u_2$	$2.3E{-}4$	$1.1E{+}74$	$8.5E{+}12$	$1.0E{+}6$	$2.7E{-}3$	$1.8E{-}3$	$3.0E{-}4$
$\theta = 0.05, u_2$	$2.3E{-}3$	$1.1E{+}75$	$8.5E{+}13$	$5.6E{+}8$	$3.6E{+}2$	$1.9E{-}2$	$3.0E{-}3$
$\theta = 0.5, u_2$	$2.3E{-}2$	$1.1E{+}76$	$8.4E{+}14$	$3.4E{+}11$	$3.5E{+}7$	$3.6E{+}0$	$3.1E{-}2$

Table 5.1: Errors of estimation runs for different θ values and inputs u_1, u_2 at $T = 20$

and any θ, the estimators perform differently for u_1 and u_2. This table emphasizes the huge improvements over many orders of magnitude from GLE to LSLE and again to the iterated LSLE versions.

Next, we will pursue experiments for affine-parameterized synthetic systems. The test setting is the same as before but with some new quantities. Let $\mathbf{1} := (1 \dots 1)^T \in \mathbb{R}^d$ and choose $\mathcal{P} = [0,1] \times [0,10] \times [-1,1]$ as parameter domain. Now we assume a kernel expansion like (5.22, p.158) or (3.9, p.98), albeit without direct time-dependency ($K_t \equiv 1$). The expansion (5.22, p.158) uses $N = 20$ and centers

$$\boldsymbol{x}_i := \frac{50(i-1)}{N-1}\mathbf{1}, \qquad \boldsymbol{\mu}_i := \frac{10(i-1)}{N-1}(0,1,0)^T, \qquad i = 1 \dots N.$$

Further, $K_{\mathcal{P}}$ is a Gaussian with $\gamma_{\mathcal{P}} = 5.3733$. Note that $K_{\mathcal{P}}$ only uses the second entry of μ, while $\mu_{\{1,3\}}$ are ignored. Finally, the expansion coefficient vectors are given via $\boldsymbol{c}_i = \exp(-\boldsymbol{x}_i/15) \in \mathbb{R}^d$, $i = 1 \dots N$ and we define $x_0(\boldsymbol{\mu}) = \mu_3\mathbf{1}$ as initial value. So parameter μ_2 is an *"expansion parameter"* influencing the system's inner dynamics and μ_3 sets the *"initial value"*; μ_1 will be discussed later. As in [182, 181], we use $\theta = 0.05$ to control $\boldsymbol{V}$ and hence the subspace quality, and we average the output using $\boldsymbol{C}(t,\mu) = \sqrt{d}\mathbf{1}$.

In the next figures, we compare the different estimated output errors $||\boldsymbol{e}_w(t)||$ using $\mu_2 = 5, \mu_3 = -0.2$. Figure 5.4 shows the absolute and relative output

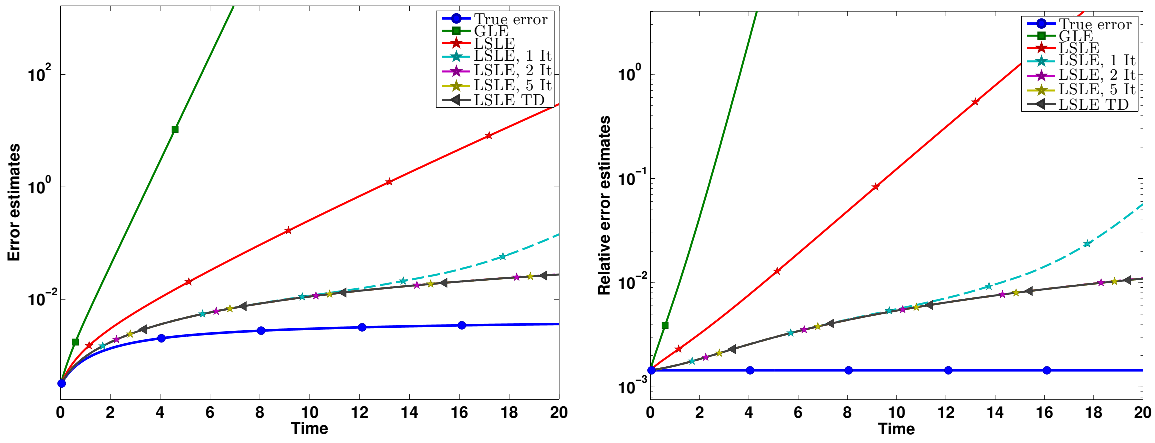


Figure 5.4: Left: Absolute L^2 state space errors of estimators. Right: Relative errors for estimators, using $\mu_2 = 5, \mu_3 = -0.2$

errors over time. The improvement from the GLE over the LSLE variant is due to the local Lipschitz constant estimations and the LSLE iterations further improve the estimation by several orders of magnitude. The fact that the LSLE is indistinguishable from the discrete LSLE TD variant after five iterations shows a fast convergence of the iteration scheme and strongly supports the applicability of the heuristic LSLE TD variant as substitute for $\Delta^{\Theta_\infty}_{LSLE}(t,\mu)$. The left hand plot in Figure 5.5 shows the computation times against the estimated output errors at $T = 20$. All estimators are $10 - 20$ times faster than computing the full error. The GLE is the cheapest but coarsest estimator, and the LSLE TD is slightly slower but yields the best estimation results. Moreover, the right hand plot of Figure 5.5 contains a parameter

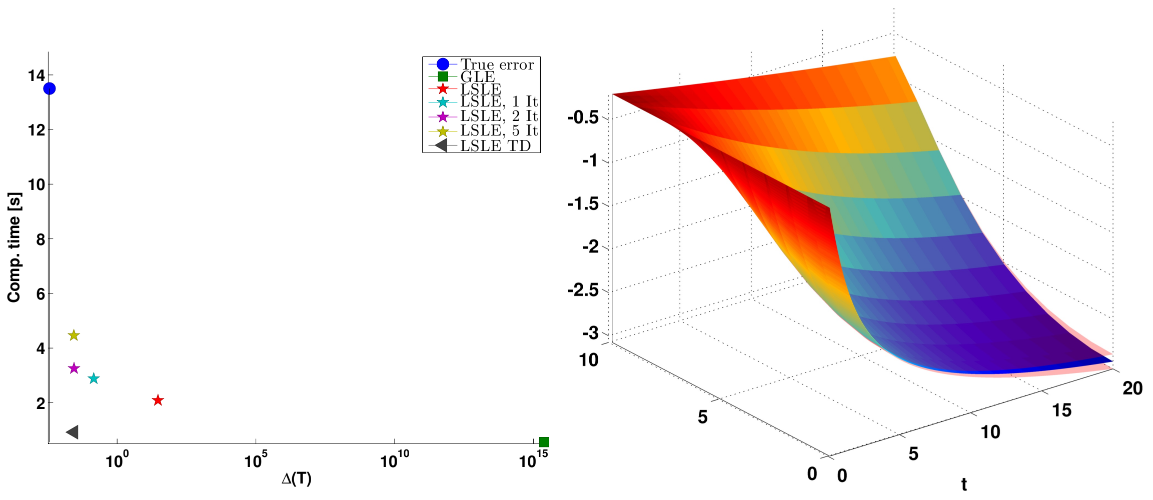


Figure 5.5: Left: Computation times for estimator variants. Right: Parameter sweep for $\mu_2 \in [0,10], \mu_1 = 0, \mu_3 = -0.2$

sweep for μ_2 ranging over $[0,10]$. Shown are the simulation outputs up to $T = 20$ along with the error bounds of the LSLE TD in transparent light-red. The error bounds stay sharp over the whole parameter range $[0,10]$, even though the system's dynamics change considerably for different μ_2.

Table 5.2 shows the output errors at $T = 20$ along with the computation times and the overestimation factors. We also observe that too many iterations of the LSLE estimation do not necessarily yield a relevant improvement. The LSLE TD estimator only overestimates by a factor of 7.6.

In order to show the influence of external input we choose an affine-parametric input matrix

$$\boldsymbol{B}(t,\boldsymbol{\mu}) = \mu_1 \begin{pmatrix} \mathbf{1}\ \mathbf{0} \end{pmatrix} + (1-\mu_1) \begin{pmatrix} \mathbf{0}\ \mathbf{1} \end{pmatrix} \in \mathbb{R}^{d\times 2},$$

with inputs

$$\boldsymbol{u}_1(t) = \begin{pmatrix} \frac{2}{5}\sin(\frac{t}{3}) \\ e^{-(12-t)^2} \end{pmatrix}, \qquad \boldsymbol{u}_2(t) = \begin{pmatrix} \frac{1}{2}\sin(\frac{t}{2}) \\ 4e^{-7(12-t)^2} - \frac{1}{2}e^{-(5-t)^2} \end{pmatrix}.$$

Both represent each an oscillating and a local stimulation of different kind. For both settings Figure 5.6 show that the LSLE TD estimator gives good es-

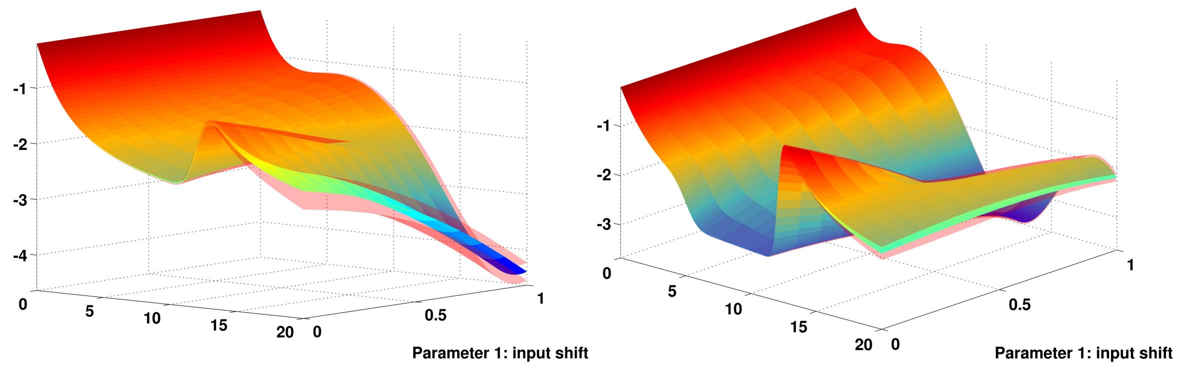


Figure 5.6: Parameter sweep for input shift $\mu_1 \in [0,1]$ *and* u_1 *(left) or* u_2 *(right)*

timations over the parameter range. Finally, Figure 5.7 displays a 2D sweep for μ_1, μ_2 and the output and error bounds by LSLE TD at $T = 20$. One can

Name	$\Delta(20)$	Time	Overestimation
True error	$3.650e-03$	21.43s	$1.000e+00$
GLE	$3.682e+15$	0.62s	$1.009e+18$
LSLE	$3.251e+01$	2.05s	$8.907e+03$
LSLE, 1 It	$1.568e-01$	2.79s	$4.295e+01$
LSLE, 2 It	$2.839e-02$	3.11s	$7.779e+00$
LSLE, 5 It	$2.801e-02$	4.13s	$7.674e+00$
LSLE TD	$2.801e-02$	0.90s	$7.674e+00$
Lower bound	$3.652e-03$	44.02s	$1.001e+00$

Table 5.2: Estimator statistics at $T = 20$

see that the estimation sharpness is more sensitive to changes of the inner dynamics compared to different inputs, which is explainable by the strong influence of μ_2 to the critical $\beta(t, \boldsymbol{\mu})$ term.

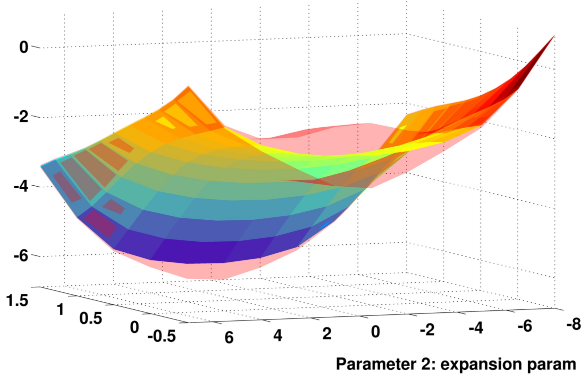


Figure 5.7: Parameter sweep for input shift *and* expansion parameter, $\mu_1 \in [0, 1], \mu_2 \in [-8, 7]$ *and using* u_1

5.2 Error estimation for DEIM reduced systems

In this section, we consider a-posteriori error estimators derived from the error system expansion type (5.6b, p.141), where the results have intermediately been published in [186] in similar form. As already mentioned in the introducing part of this chapter, this representation has the advantage of being able to evaluate E_A at the reconstructed reduced state space variable, but requires some kind of Lipschitz constant estimation for the system's nonlinearity $\boldsymbol{f}$. To this end, providing approximations to $\boldsymbol{f}$ via DEIM (see Sections 2.3 and 3.2.2) allows to efficiently compute E_A. This, and a new technique involving a "*Matrix-DEIM*" (MDEIM) and partial similarity transformations provide Lipschitz-constants for any nonlinearity $\boldsymbol{f}$ that allows application of the standard DEIM. In more detail, the key ideas to our error estimation procedure is to use local logarithmic Lipschitz constant [156] approximations and an estimation of the DEIM approximation error using a higher order DEIM approximation, where the latter extends the ideas from [66, 171] and is done similarly in [47] for the PDE case. For the POD-DEIM approach an a-priori error estimate in terms of neglected singular values (for both projection and DEIM basis) has been recently developed in [24].

However, while theoretically satisfying, this a-priori estimate has little practical value with respect to assessing the accuracy of a reduced order solution during a simulation.

As before, our approach involves a preliminary offline phase to construct the components of the estimator and an efficient online calculation as the reduced order model simulation proceeds. Similar to the procedure in Section 5.1, we will introduce the main concepts for the "simple" system (5.1, p.138) first and extend the results to more general parameterized systems like (5.21, p.157) later. At first we establish some required concepts and useful intermediate results.

5.2.1 Logarithmic Lipschitz constants

A crucial part for the error estimation process is the concept of logarithmic Lipschitz constants for functions. They have been introduced in [33] for the linear case and a more general theory has become available since. See [156] for an elegant overview.

Definition 5.2.1 (Logarithmic Lipschitz constants)**.** For a function $\boldsymbol{f} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ we define the logarithmic Lipschitz constant with respect to $\boldsymbol{G}$ by

$$L_G[f] := \lim_{h \rightarrow 0^+} \frac{1}{h} \left(\sup_{\boldsymbol{x} \neq \boldsymbol{y} \in \mathbb{R}^d} \frac{||\boldsymbol{x} - \boldsymbol{y} + h(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y}))||_G}{||\boldsymbol{x} - \boldsymbol{y}||_G} - 1 \right). \tag{5.23}$$

Next we give an equivalent representation that is more suitable for applications.

Lemma 5.2.2 (Equivalent representations for logarithmic Lipschitz constants)**.** *If $\boldsymbol{f}$ is Lipschitz-continuous, the logarithmic Lipschitz constant of $\boldsymbol{f}$ is given by*

$$L_G[f] = \sup_{\boldsymbol{x} \neq \boldsymbol{y} \in \mathbb{R}^d} \frac{\langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y}) \rangle_G}{||\boldsymbol{x} - \boldsymbol{y}||_G^2}.$$

Proof. Put $\Delta \boldsymbol{f} := \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y})$ and note $\|\Delta \boldsymbol{f}\|_G \leq L \, ||\boldsymbol{x} - \boldsymbol{y}||_G$ for some L due to the Lipschitz continuity of $\boldsymbol{f}$. It is straightforward to show that

$$\begin{aligned}&\frac{1}{h}\left(\frac{||\boldsymbol{x}-\boldsymbol{y}+h\Delta\boldsymbol{f}||_G}{||\boldsymbol{x}-\boldsymbol{y}||_G}-1\right)\\ =&\frac{2\langle\boldsymbol{x}-\boldsymbol{y},\Delta\boldsymbol{f}\rangle_G+h\,||\Delta\boldsymbol{f}||_G^2}{||\boldsymbol{x}-\boldsymbol{y}||_G\left(||\boldsymbol{x}-\boldsymbol{y}+h\Delta\boldsymbol{f}||_G+||\boldsymbol{x}-\boldsymbol{y}||_G\right)}.\end{aligned}$$

Now, using

$$||\boldsymbol{x}-\boldsymbol{y}||_G\,(1-hL)\leq||\boldsymbol{x}-\boldsymbol{y}+h\Delta\boldsymbol{f}||_G\leq||\boldsymbol{x}-\boldsymbol{y}||_G\,(1+hL)$$

gives

$$\begin{aligned}\frac{2\langle\boldsymbol{x}-\boldsymbol{y},\Delta\boldsymbol{f}\rangle_G}{||\boldsymbol{x}-\boldsymbol{y}||_G^2\,(2+hL)}&\leq\frac{1}{h}\left(\frac{||\boldsymbol{x}-\boldsymbol{y}+h\Delta\boldsymbol{f}||_G}{||\boldsymbol{x}-\boldsymbol{y}||_G}-1\right)\\&\leq\frac{2\langle\boldsymbol{x}-\boldsymbol{y},\Delta\boldsymbol{f}\rangle_G}{||\boldsymbol{x}-\boldsymbol{y}||_G^2\,(2-hL)}+hL^2.\end{aligned}$$

Thus,

$$\begin{aligned}\frac{2}{2+hL}\sup_{\boldsymbol{x}\neq\boldsymbol{y}\in\mathbb{R}^d}\frac{\langle\boldsymbol{x}-\boldsymbol{y},\Delta\boldsymbol{f}\rangle_G}{||\boldsymbol{x}-\boldsymbol{y}||_G^2}&\leq\sup_{\boldsymbol{x}\neq\boldsymbol{y}\in\mathbb{R}^d}\frac{1}{h}\left(\frac{||\boldsymbol{x}-\boldsymbol{y}+h\Delta\boldsymbol{f}||_G}{||\boldsymbol{x}-\boldsymbol{y}||_G}-1\right)\\&\leq\frac{2}{2-hL}\sup_{\boldsymbol{x}\neq\boldsymbol{y}\in\mathbb{R}^d}\frac{\langle\boldsymbol{x}-\boldsymbol{y},\Delta\boldsymbol{f}\rangle_G}{||\boldsymbol{x}-\boldsymbol{y}||_G^2}+hL^2,\end{aligned}$$

and finally, taking limits as $h \to 0^+$ across the inequalities will establish the result. □

Corollary 5.2.3 (Logarithmic Lipschitz constants for linear functions)**.** *The logarithmic Lipschitz constant of a linear function* $\boldsymbol{f}(\boldsymbol{x}) \equiv \boldsymbol{A}\boldsymbol{x}$ *with* $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ *is given by*

$$L_G[\boldsymbol{A}] := \lim_{h\to 0^+}\frac{||\boldsymbol{I}+h\boldsymbol{A}||_G-1}{h}, \tag{5.24}$$

where $||\cdot||_G$ *is the* $\boldsymbol{G}$*-induced matrix norm. Furthermore,* (5.24) *is equivalent* *to both*

$$L_G[\boldsymbol{A}] = \sup_{\boldsymbol{x}\in\mathbb{R}^d\setminus\{0\}} \frac{\langle \boldsymbol{x}, \boldsymbol{A}\boldsymbol{x}\rangle_G}{\langle \boldsymbol{x}, \boldsymbol{x}\rangle_G}, \tag{5.25}$$

$$L_G[\boldsymbol{A}] = \max\Big\{\sigma\Big(\frac{1}{2}(\tilde{\boldsymbol{A}} + \tilde{\boldsymbol{A}}^T)\Big)\Big\}, \tag{5.26}$$

where $\tilde{\boldsymbol{A}} := \boldsymbol{C}^T\boldsymbol{A}\boldsymbol{C}^{-T}$ *and* $\boldsymbol{G} = \boldsymbol{C}\boldsymbol{C}^T$ *denotes the Cholesky factorization of* $\boldsymbol{G}$.

Proof. We obtain (5.24) directly from (5.23, p.176) using linearity and the matrix norm definition. As the map $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ is Lipschitz, application of Lemma 5.2.2 directly yields (5.25). Let $\boldsymbol{x} \in \mathbb{R}^d\setminus\{0\}$ and $\boldsymbol{y} := \boldsymbol{C}^T\boldsymbol{x}$. Then

$$\begin{aligned}\frac{\langle \boldsymbol{x}, \boldsymbol{A}\boldsymbol{x}\rangle_G}{\langle \boldsymbol{x}, \boldsymbol{x}\rangle_G} = \frac{\boldsymbol{x}^T\boldsymbol{G}\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{G}\boldsymbol{x}} &= \frac{\boldsymbol{y}^T\boldsymbol{C}^T\boldsymbol{A}\boldsymbol{C}^{-T}\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{y}}\\ &= \frac{1}{2}\frac{\boldsymbol{y}^T\tilde{\boldsymbol{A}}\boldsymbol{y} + \boldsymbol{y}^T\tilde{\boldsymbol{A}}^T\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{y}} = \frac{1}{2}\frac{\boldsymbol{y}^T(\tilde{\boldsymbol{A}} + \tilde{\boldsymbol{A}}^T)\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{y}},\end{aligned}$$

which shows equality of (5.25) and (5.26) via

$$\sup_{\boldsymbol{x}\in\mathbb{R}^d\setminus\{0\}} \frac{\langle \boldsymbol{x}, \boldsymbol{A}\boldsymbol{x}\rangle_G}{\langle \boldsymbol{x}, \boldsymbol{x}\rangle_G} = \sup_{\boldsymbol{y}\in\mathbb{R}^d\setminus\{0\}} \frac{1}{2}\frac{\boldsymbol{y}^T(\tilde{\boldsymbol{A}} + \tilde{\boldsymbol{A}}^T)\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{y}} = \max\Big\{\sigma\Big(\frac{1}{2}(\tilde{\boldsymbol{A}} + \tilde{\boldsymbol{A}}^T)\Big)\Big\}.$$

$\square$

Remark 5.2.4. In our expression for the logarithmic Lipschitz constant of a linear function given above, we abused notation a bit by entering $\boldsymbol{A}$ rather than $\boldsymbol{f}$ as the argument in order to keep with the notation introduced by Dahlquist [33]. The notion of logarithmic Lipschitz constants is a natural generalization of the logarithmic norm [156].

As demonstrated in Section 5.1 and [182, 181] for reduced kernel-based systems, some local information given by the reduced state space coordinates can be useful. In this context, the notion of "*local logarithmic Lipschitz constants*" will be an important aspect.

Definition 5.2.5 (Local logarithmic Lipschitz constants)**.** For a function $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^d$ we define the local logarithmic Lipschitz constant at $\boldsymbol{x} \in \mathbb{R}^d$ with respect to $\boldsymbol{G}$ by

$$L_G[f](\boldsymbol{x}) := \sup_{\boldsymbol{y}\in\mathbb{R}^d} \frac{\langle \boldsymbol{x}-\boldsymbol{y}, \boldsymbol{f}(\boldsymbol{x})-\boldsymbol{f}(\boldsymbol{y})\rangle_G}{||\boldsymbol{x}-\boldsymbol{y}||_G^2}.$$

Remark 5.2.6. Without further assumptions, the (local) logarithmic Lipschitz constant may be unbounded. However, in this work we assume the system nonlinearities $\boldsymbol{f}$ to be Lipschitz-continuous, which not only ensures existing solutions for the considered systems (in finite time) but also grants boundedness of both global and local constants. This is easily verified using Lemma 5.2.2 and the Cauchy-Schwarz inequality.

5.2.2 Error estimation of DEIM approximation

In this section, we shall establish the formalism needed in order to compute the DEIM approximation error. Similar to [47, 67, 66, 171], the key idea is to assume to have a maximum DEIM order M at which the approximation is essentially exact (meaning it is accurate to within working precision). Of course, this is satisfied in the worst case only for $M = d$, but our following experiments show that in practice far smaller values can be used in order to obtain very accurate estimates of the DEIM approximation error. Another possible approach could be based upon the recent results on a-posteriori EIM approximation errors in [52], which work without the exactness assumption but use Taylor expansions around suitable points instead.

The following lemma facilitates an efficient computation by providing a useful decomposition of the DEIM projection matrix relative to an intermediate lower order DEIM approximation.

Lemma 5.2.7 (DEIM matrix decomposition)**.** *Let* $d, m, m' \in \mathbb{N}, m+m' \leq d$ *and matrices* $\boldsymbol{U}_m, \boldsymbol{P}_m \in \mathbb{R}^{d\times m}, \boldsymbol{U}_{m'}, \boldsymbol{P}_{m'} \in \mathbb{R}^{d\times m'}$ *be given. Assume that the matrices* $\boldsymbol{U} := [\boldsymbol{U}_m\, \boldsymbol{U}_{m'}],\ \boldsymbol{P} := [\boldsymbol{P}_m\, \boldsymbol{P}_{m'}]$ *each have linearly independent* *columns and that the matrices $\boldsymbol{P}^T\boldsymbol{U}$ and $\boldsymbol{P}_m^T\boldsymbol{U}_m$ are non-singular. Define the oblique projectors*

$$\Pi := \boldsymbol{U}(\boldsymbol{P}^T\boldsymbol{U})^{-1}\boldsymbol{P}^T, \tag{5.27a}$$

$$\Pi_m := \boldsymbol{U}_m(\boldsymbol{P}_m^T\boldsymbol{U})_m^{-1}\boldsymbol{P}_m^T, \tag{5.27b}$$

$$\Pi_{m'} := (\boldsymbol{I} - \Pi_m)\boldsymbol{U}_{m'}(\boldsymbol{P}_{m'}^T(\boldsymbol{I} - \Pi_m)\boldsymbol{U}_{m'})^{-1}\boldsymbol{P}_{m'}^T. \tag{5.27c}$$

Then we have

$$\Pi = \Pi_m + \Pi_{m'}(\boldsymbol{I} - \Pi_m).$$

Proof. First note that Π may be written as $\Pi = \boldsymbol{U}_m\boldsymbol{X} + \boldsymbol{U}_{m'}\boldsymbol{Y}$ with

$$\begin{pmatrix} \boldsymbol{P}_m^T\boldsymbol{U}_m & \boldsymbol{P}_m\boldsymbol{U}_{m'} \\ \boldsymbol{P}_{m'}^T\boldsymbol{U}_m & \boldsymbol{P}_{m'}^T\boldsymbol{U}_{m'} \end{pmatrix} \begin{pmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{pmatrix} = \begin{pmatrix} \boldsymbol{P}_m^T \\ \boldsymbol{P}_{m'}^T \end{pmatrix}. \tag{5.28}$$

One step of block Gaussian elimination applied to (5.28) is achieved by multiplying both sides with the non-singular block matrix

$$\begin{pmatrix} \boldsymbol{I} & 0 \\ -\boldsymbol{P}_{m'}^T\boldsymbol{U}_m(\boldsymbol{P}_m^T\boldsymbol{U}_m)^{-1} & \boldsymbol{I} \end{pmatrix}.$$

This provides the transformed equation

$$\begin{pmatrix} \boldsymbol{P}_m^T\boldsymbol{U}_m & \boldsymbol{P}_m\boldsymbol{U}_{m'} \\ \boldsymbol{0} & \boldsymbol{P}_{m'}^T(\boldsymbol{I} - \Pi_m)\boldsymbol{U}_{m'} \end{pmatrix} \begin{pmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{pmatrix} = \begin{pmatrix} \boldsymbol{P}_m^T \\ \boldsymbol{P}_{m'}^T(\boldsymbol{I} - \Pi_m) \end{pmatrix}. \tag{5.29}$$

The non-singularity of $\boldsymbol{P}^T\boldsymbol{U}$ implies the non-singularity of $\boldsymbol{P}_{m'}^T(\boldsymbol{I}-\Pi_m)\boldsymbol{U}_{m'}$ to give the expression

$$\boldsymbol{U}_{m'}\boldsymbol{Y} = \boldsymbol{U}_{m'}(\boldsymbol{P}_{m'}^T(\boldsymbol{I} - \Pi_m)\boldsymbol{U}_{m'})^{-1}\boldsymbol{P}_{m'}^T(\boldsymbol{I} - \Pi_m).$$

Now (5.29) implies $\boldsymbol{X} = \Pi_m - \Pi_m \boldsymbol{U}_{m'} \boldsymbol{Y}$ and it follows that

$$\Pi = \boldsymbol{U}_m \boldsymbol{X} + \boldsymbol{U}_{m'} \boldsymbol{Y} = \Pi_m + (\boldsymbol{I} - \Pi_m)\boldsymbol{U}_{m'}\boldsymbol{Y} = \Pi_m + \Pi_{m'}(\boldsymbol{I} - \Pi_m)$$

as claimed. □

Note that, for improved readability, we write $\boldsymbol{U}_{m'}$ instead of introducing new variables $\boldsymbol{U}'_{m'}$ etc. An application of Lemma 5.2.7 with $m' := M - m$ directly leads to the following theorem, which shows how to efficiently compute the DEIM approximation error using only DEIM matrices of sizes m and m' instead of $M (= m + m')$.

Theorem 5.2.8 (DEIM error representation). *Let $\mathcal{U} := \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_M\}$ be a DEIM basis with corresponding set of interpolation points $\mathcal{E} = \{\wp_1, \ldots, \wp_M\}$. Further assume that the M-th order DEIM approximation of $\boldsymbol{f}$ is exact, i.e. $\hat{\boldsymbol{f}}_M \equiv \boldsymbol{f}$ on Ω. For $m \leq M - 1, m' = M - m$ set*

$$\begin{aligned} \boldsymbol{P}_m &:= [\boldsymbol{e}_{\wp_1}, \ldots, \boldsymbol{e}_{\wp_m}], & \boldsymbol{P}_{m'} &:= [\boldsymbol{e}_{\wp_{m+1}}, \ldots, \boldsymbol{e}_{\wp_{m+m'}}], \\ \boldsymbol{U}_m &:= [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m], & \boldsymbol{U}_{m'} &:= [\boldsymbol{u}_{m+1}, \ldots, \boldsymbol{u}_{m+m'}]. \end{aligned}$$

Then the approximation error for the m-th order DEIM approximation of $\boldsymbol{f}$ is given by

$$\boldsymbol{f}(\boldsymbol{y}) - \hat{\boldsymbol{f}}_m(\boldsymbol{y}) = \big(\Pi - \Pi_m\big)\boldsymbol{f}(\boldsymbol{y}) = \Pi_{m'}\big(\boldsymbol{I} - \Pi_m\big)\boldsymbol{f}(\boldsymbol{y}), \tag{5.30}$$

with matrices $\Pi, \Pi_m, \Pi_{m'}$ as defined in (5.27).

This form of the error has a revealing interpretation. It expresses the error as an oblique projection of the DEIM approximation error $(\boldsymbol{I} - \Pi_m)\boldsymbol{f}(\boldsymbol{y})$ whose norm is within a factor of $\|\Pi_m\| = \|(\boldsymbol{P}_m^T \boldsymbol{U}_m)^{-1}\|$ of the optimal POD error when the basis $\boldsymbol{U}_m$ is orthogonal [23].

As the DEIM approximation error estimation of Theorem 5.2.8 is only a part of the overall error estimation process, we first investigate the estimation quality for different m, m' choices separately. We consider the 1D viscous

Burgers equation introduced in Section 3.4.3 as example system; we refer to that part for the applied settings and definitions. As test data we use the snapshot training data $X = \bigcup_{\mu\in\Xi} X_\mu$ and compare against the true error given via Y. For this, Figures 5.8 and 5.9 display true and estimated DEIM approximation errors, using L^2 in state space and L^∞ over Y. The left plot

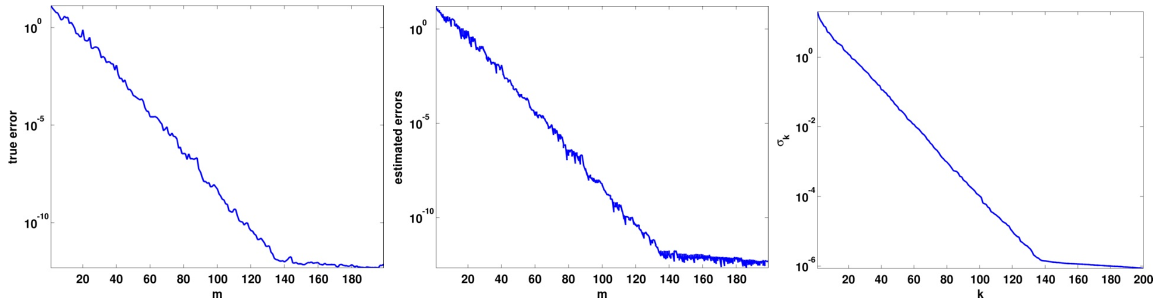
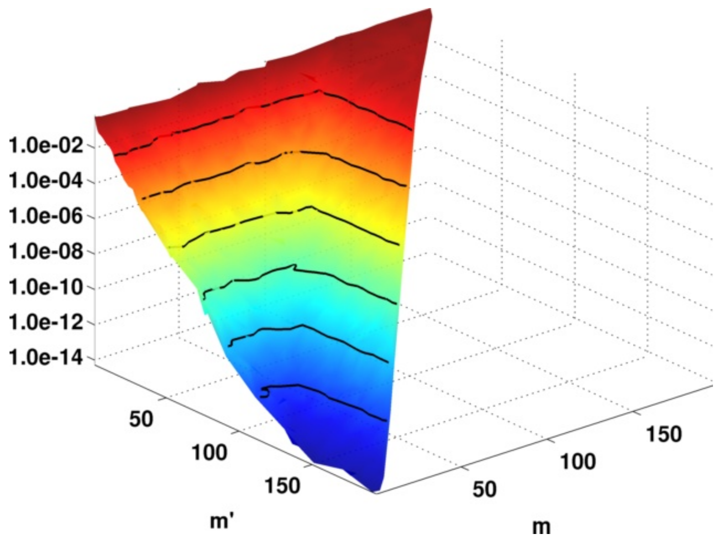


Figure 5.8: True (left) and estimated (middle) absolute DEIM approximation errors. Right: Singular value decay of POD on Y

of Figure 5.8 shows the true error decay for increasing m on X. The middle plot shows the estimated error via (5.30) over the same DEIM orders for all remaining possible m' values (i.e. $m < m' \leq M$), plotted in an overlay. This shows that the estimation is very closely following the true DEIM error, independently of the current m' choice. In fact, the visible deviations are all essentially caused by the first $1 \leq m' \leq 4$ values. Finally, the rightmost plot of Figure 5.8 shows the singular values of the POD on Y to obtain the DEIM basis $\mathcal{U}$. The change of decay rate indicates that the main dynamics of $\boldsymbol{f}$ are captured at around 140 modes, which matches with the true error decay observable in the leftmost plot. In order to give more insight, the left plot of Figure 5.9 displays the mean relative error

$$\text{mean}_{\boldsymbol{x}\in X} \|\boldsymbol{f}(\boldsymbol{x}) - \hat{\boldsymbol{f}}_m(\boldsymbol{x}) - \Pi_{m'}(\boldsymbol{I} - \Pi_m)\boldsymbol{f}(\boldsymbol{y})\| / \|\boldsymbol{f}(\boldsymbol{x}) - \hat{\boldsymbol{f}}_m(\boldsymbol{x})\|$$

between the true and estimated DEIM approximation errors on X. The contours are located at the levels 10^{-2} to 10^{-12}. The "bends" of the contours in the image are where $m + m' \approx 130$, meaning that no real estimation improvement can be achieved with higher m' values. This is in accordance with the stagnating decay of the singular values as shown in the right plot of Figure 5.8, as no essential new information is covered using larger m' values.



	0.1		0.01	
m	max	∅	max	∅
1	14	4	25	14
2	13	5	24	13
7	15	5	26	14
14	15	6	23	15
21	16	8	23	14
28	11	4	19	11
35	13	4	24	12
42	14	5	22	12
49	15	6	23	11
56	16	4	21	12
63	14	4	23	11
70	15	4	21	11
77	14	5	23	13
84	11	5	21	10
91	14	5	23	13
98	13	5	24	13
102	14	4	22	14

Figure 5.9: Plot: Mean relative error between true and estimated DEIM approximation error over training set X. Table: Minimum required m' values for given m and relative errors.

Thus, further comments on the figure focus on the area where $m+m' \leq 130$. Most importantly, the estimation accuracy for increasing m' values at fixed m is improving exponentially. The other way around, it is interesting to see that the contours are basically straight lines on each level, which means that for fixed m' the same error estimation precision is achieved for all m values.

To provide some values of the plot on the left hand side in Figure 5.9, the table on the right shows how large m' must be chosen in order to obtain 10% or 1% maximum or mean relative error on X, respectively, for different DEIM orders m. The mean values on the 0.01 relative error column are corresponding to points on the contour of the left plot for $1 \leq m \leq 102$, e.g. the contour is located around $m' = 14$ which is sufficient (in average) to ensure a relative error estimation error of less than one percent. In summary, this illustrates that a very good estimation of the actual DEIM error is possible for our example experiment.

5.2.3 Rigorous a-posteriori error estimation

With these preliminaries, we can present a rigorous a-posteriori error estimator for DEIM reduced systems. Note here that, as the system variant (5.6b, p.141) is used, the nonlinear approximation error E_A is evaluated at $\boldsymbol{x}^r(t)$ and hence does not need to be separated from the other terms as in the previous kernel-related estimates from e.g. Theorem 5.1.9.

Theorem 5.2.9 (A-posteriori error estimation for DEIM reduced systems). *Let the system (5.1, p.138) be given and the conditions from Theorem 5.2.8 hold. If $\boldsymbol{f}$ is Lipschitz-continuous, then the state space error is rigorously bounded via*

$$||\boldsymbol{e}(t)||_G \leq \Delta_{EI}(t) \;\; \forall\, t \in [0,T],$$

with

$$\Delta_{EI}(t) := \int_0^t \alpha(s) e^{\int_s^t \beta(\tau)d\tau} ds + e^{\int_0^t \beta(\tau)d\tau} E_0, \tag{5.31a}$$

$$\alpha(t) := \left|\left|\left(\Pi_{m'}(\boldsymbol{I} - \Pi_m) + \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\Pi_m\right) \boldsymbol{f}(\boldsymbol{y}^r(t))\right|\right|_G, \tag{5.31b}$$

$$\beta(t) := L_G[\boldsymbol{f}](\boldsymbol{x}^r(t)), \tag{5.31c}$$

and the matrix definitions from Lemma 5.2.7, (5.27, p.180).

Proof. Note that for $m' = M - m$, Theorem 5.2.8 implies

$$\begin{aligned} &\boldsymbol{f}(\boldsymbol{x}^r(t)) - \boldsymbol{V}\boldsymbol{W}^T \hat{\boldsymbol{f}}_m(\boldsymbol{x}^r(t)) \\ &= \boldsymbol{f}(\boldsymbol{x}^r(t)) - \hat{\boldsymbol{f}}_m(\boldsymbol{x}^r(t)) + \hat{\boldsymbol{f}}_m(\boldsymbol{x}^r(t)) - \boldsymbol{V}\boldsymbol{W}^T \hat{\boldsymbol{f}}_m(\boldsymbol{x}^r(t)) \\ &= \Pi_{m'}(\boldsymbol{I} - \Pi_m)\boldsymbol{f}(\boldsymbol{y}^r(t)) + \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\Pi_m \boldsymbol{f}(\boldsymbol{y}^r(t)) \end{aligned} \tag{5.32}$$

Now, from (5.6b, p.141) and (5.32) we obtain

$$\begin{aligned} \langle \boldsymbol{e}(t), \boldsymbol{e}'(t)\rangle_G = {}& \langle \boldsymbol{e}(t), \boldsymbol{f}(\boldsymbol{x}(t)) - \boldsymbol{f}(\boldsymbol{x}^r(t))\rangle_G \\ &+ \left\langle \boldsymbol{e}(t), \boldsymbol{f}(\boldsymbol{x}^r(t)) - \boldsymbol{V}\boldsymbol{W}^T \hat{\boldsymbol{f}}_m(\boldsymbol{x}^r(t))\right\rangle_G \end{aligned}$$

$$
\begin{aligned}
&\leq L_G[\boldsymbol{f}](\boldsymbol{x}^r(t))\,||\boldsymbol{e}(t)||_G^2 + \langle \boldsymbol{e}(t), \Pi_{m'}(\boldsymbol{I} - \Pi_m)\boldsymbol{f}(\boldsymbol{y}^r(t))\rangle_G \\
&\quad + \left\langle \boldsymbol{e}(t), \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\Pi_m \boldsymbol{f}(\boldsymbol{y}^r(t))\right\rangle_G \\
&\leq L_G[\boldsymbol{f}](\boldsymbol{x}^r(t))\,||\boldsymbol{e}(t)||_G^2 + ||\boldsymbol{e}(t)||_G\,\alpha(t).
\end{aligned}
$$

This finally gives

$$
\frac{\partial}{\partial t}\,||\boldsymbol{e}(t)||_G = \frac{\langle \boldsymbol{e}(t), \boldsymbol{e}'(t)\rangle_G}{||\boldsymbol{e}(t)||_G} \leq L_G[\boldsymbol{f}](\boldsymbol{x}^r(t))\,||\boldsymbol{e}(t)||_G + \alpha(t),
$$

and, with $\Delta(t) := ||\boldsymbol{e}(t)||_G$, $\Delta(0) = E_0$, application of the comparison Lemma 5.0.1 yields the desired results. □

Note that zero error is achieved if the DEIM approximation is exact, i.e. $m = M$ in our context, and the trajectory is completely contained in the subspace $\boldsymbol{V}$. In this case we will have zero initial error and $\alpha(t) \equiv 0$ which means the estimator correctly predicts zero error.

Remark 5.2.10. The error estimator of Theorem 5.2.9 assumes exactness of the DEIM approximation of order M. As an alternative, if an a-priori bound ϵ on the DEIM approximation error in Ω is available, the above estimation procedure would straightforwardly lead to a rigorous a-posteriori error estimator without the exactness assumption. However, Section 5.4.1.1 demonstrates the practicability of the above approach even for substantially smaller m' values than $M - m$.

5.2.4 Efficient error estimation

The error estimator introduced in Theorem 5.2.9 makes explicit use of the local logarithmic Lipschitz constant $L_G[\boldsymbol{f}](\boldsymbol{x}^r(t))$, which is (as well as its global counterpart $L_G[\boldsymbol{f}]$) not readily available in most practical situations. Therefore, let $\boldsymbol{J} : \Omega \to \mathbb{R}^{d\times d}$ denote the Jacobian $\boldsymbol{J}(\boldsymbol{x})$ of $\boldsymbol{f}$ at $\boldsymbol{x} \in \Omega$. With the Taylor expansion of $\boldsymbol{f}$ around $\boldsymbol{x}^r(t)$ we obtain

$$\begin{aligned}\frac{\langle \boldsymbol{e}(t), \boldsymbol{f}(\boldsymbol{x}(t)) - \boldsymbol{f}(\boldsymbol{x}^r(t))\rangle_G}{||\boldsymbol{e}(t)||_G^2} &= \frac{\left\langle \boldsymbol{e}(t), \boldsymbol{J}(\boldsymbol{x}^r(t))\boldsymbol{e}(t) + \mathcal{O}\left(||\boldsymbol{e}(t)||_G^2\right)\right\rangle_G}{||\boldsymbol{e}(t)||_G^2} \\ &= \frac{\langle \boldsymbol{e}(t), \boldsymbol{J}(\boldsymbol{x}^r(t))\boldsymbol{e}(t)\rangle_G}{||\boldsymbol{e}(t)||_G^2} + \mathcal{O}\left(||\boldsymbol{e}(t)||_G\right),\end{aligned} \tag{5.33}$$

for any $t \in [0, T]$. With Definition 5.2.5 and (5.25, p.178) this directly gives a first order approximation of the local logarithmic Lipschitz constant

$$L_G[\boldsymbol{f}](\boldsymbol{x}^r(t)) = L_G[\boldsymbol{J}(\boldsymbol{x}^r(t))] + \mathcal{O}\left(||\boldsymbol{e}(t)||_G\right). \tag{5.34}$$

Using the approximation (5.34) avoids the need to obtain $L_G[\boldsymbol{f}](\boldsymbol{x}^r(t))$, but it comes with additional cost: The computation of the Jacobian logarithmic norm is expensive as it involves solving an eigenvalue problem of high dimension d due to the representation (5.26, p.178).

We will address this issue in the following discussion, where we propose to apply a suitable partial similarity transformation to the Jacobians which has been designed to preserve the largest eigenvalues of their symmetric parts. The key ingredient is to perform a POD of their corresponding eigenvectors, which in turn allows us to bound the resulting eigenvalue approximation error in terms of the remaining eigenvalues of their covariance matrix. We explain this idea for general symmetric matrices in the following theorem, and refer to [97, 176] for details on POD and related error estimates. Recall here the MatLab style notation introduced in Chapter 1.

Theorem 5.2.11 (Approximation of eigenvalues for a family of symmetric matrices). *Let a continuous family of symmetric matrices $\boldsymbol{H}(t) \in \mathbb{R}^{d\times d}$ over $t \in [a, b]$ be given and let $[\lambda(t), \boldsymbol{q}(t)] := \lambda_{max}(\boldsymbol{H}(t))$ denote the largest eigenvalue $\lambda(t)$ with corresponding normalized eigenvector $\boldsymbol{q}(t)$ of $\boldsymbol{H}(t)$. Let*

$\sup_{t\in[a,b]} ||\boldsymbol{H}(t)|| \leq C_H$ *hold. Further, define*

$$\boldsymbol{R} = \int_a^b \boldsymbol{q}(t)\boldsymbol{q}(t)^T dt,$$

and let $\boldsymbol{Q\Sigma}^2\boldsymbol{Q}^T = \boldsymbol{R}$ *be the eigen-decomposition of* $\boldsymbol{R}$ *with*

$$\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}, \quad \boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_d), \quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d > 0.$$

For $k \leq d$ *let* $\boldsymbol{Q}_k := \boldsymbol{Q}[:, 1{:}k]$ *and* $\lambda_k(t) := \lambda_{max}(\boldsymbol{Q}_k^T\boldsymbol{H}(t)\boldsymbol{Q}_k)$. *Then*

$$\int\limits_a^b |\lambda(t) - \lambda_k(t)|dt \leq 4C_H \sum_{j>k} \sigma_j^2.$$

Proof. First, define the vector valued function $\boldsymbol{w}(t) := \boldsymbol{\Sigma}^{-1}\boldsymbol{Q}^T\boldsymbol{q}(t), \quad t \in [a,b]$. Note that

$$\int_a^b \boldsymbol{w}(t)\boldsymbol{w}(t)^T dt = \int_a^b \boldsymbol{\Sigma}^{-1}\boldsymbol{Q}^T\boldsymbol{q}(t)\boldsymbol{q}^T(t)\boldsymbol{Q\Sigma}^{-1}dt = \boldsymbol{\Sigma}^{-1}\boldsymbol{Q}^T\boldsymbol{RQ\Sigma}^{-1} = \boldsymbol{I}.$$

Thus, $\int_a^b w_i(t)w_j(t)dt = \delta_{ij}$ (the Kronecker delta), where $w_i(t)$ is the i-th component of $\boldsymbol{w}(t)$. Now, partition

$$\boldsymbol{Q} = [\boldsymbol{Q}_k, \tilde{\boldsymbol{Q}}_k], \quad \boldsymbol{\Sigma} = \operatorname{diag}\left(\boldsymbol{\Sigma}_k, \tilde{\boldsymbol{\Sigma}}_k\right), \quad \boldsymbol{w}(t) = [\boldsymbol{w}_k^T(t), \tilde{\boldsymbol{w}}_k^T(t)]^T,$$

with $\tilde{\boldsymbol{Q}}_k := \boldsymbol{Q}(:, (k{+}1){:}d)$, $\boldsymbol{\Sigma}_k, \tilde{\boldsymbol{\Sigma}}_k$ denoting the appropriate diagonal blocks of $\boldsymbol{\Sigma}$ and $\boldsymbol{w}_k(t), \tilde{\boldsymbol{w}}_k(t)$ denoting the corresponding sub-vectors of $\boldsymbol{w}(t)$. Put

$$\boldsymbol{q}(t) = \boldsymbol{q}_k(t) + \tilde{\boldsymbol{q}}_k(t), \quad \text{with} \quad \boldsymbol{q}_k(t) := \boldsymbol{Q}_k\boldsymbol{\Sigma}_k\boldsymbol{w}_k(t), \quad \tilde{\boldsymbol{q}}_k(t) := \tilde{\boldsymbol{Q}}_k\tilde{\boldsymbol{\Sigma}}_k\tilde{\boldsymbol{w}}_k(t),$$

and note that $\boldsymbol{q}_k^T(t)\tilde{\boldsymbol{q}}_k(t) = 0$ for all $t \in [a,b]$. We observe that

$$\int_a^b \tilde{\boldsymbol{q}}_k^T(t)\tilde{\boldsymbol{q}}_k(t)dt = \int_a^b \tilde{\boldsymbol{w}}_k^T(t)\tilde{\boldsymbol{\Sigma}}_k^2\tilde{\boldsymbol{w}}_k(t)dt = \sum_{j>k}\sigma_j^2 \int_a^b w_j^2(t)dt = \sum_{j>k}\sigma_j^2. \tag{5.35}$$

In the following we omit the argument t and set $\boldsymbol{q} = \boldsymbol{q}(t)$, $\boldsymbol{q}_k = \boldsymbol{q}_k(t)$, $\tilde{\boldsymbol{q}}_k = \tilde{\boldsymbol{q}}_k(t)$, etc. Then

$$\begin{aligned} \boldsymbol{q}_k^T \boldsymbol{H} \boldsymbol{q}_k &= (\boldsymbol{q} - \tilde{\boldsymbol{q}}_k)^T \boldsymbol{H} (\boldsymbol{q} - \tilde{\boldsymbol{q}}_k) = \lambda - 2\boldsymbol{q}^T \boldsymbol{H} \tilde{\boldsymbol{q}}_k + \tilde{\boldsymbol{q}}_k^T \boldsymbol{H} \tilde{\boldsymbol{q}}_k \\ &= \lambda - 2\lambda \boldsymbol{q}^T \tilde{\boldsymbol{q}}_k + \tilde{\boldsymbol{q}}_k^T \boldsymbol{H} \tilde{\boldsymbol{q}}_k = \lambda - 2\lambda \tilde{\boldsymbol{q}}_k^T \tilde{\boldsymbol{q}}_k + \tilde{\boldsymbol{q}}_k^T \boldsymbol{H} \tilde{\boldsymbol{q}}_k. \end{aligned} \tag{5.36}$$

Moreover, since $\mu \leq ||\boldsymbol{H}|| \leq C_H$ for any eigenvalue μ of $\boldsymbol{H}$, the definitions of $\lambda = \lambda(t)$ and $\lambda_k = \lambda_k(t)$ imply

$$\lambda = \sup_{\boldsymbol{v} \neq 0} \frac{\boldsymbol{v}^T \boldsymbol{H} \boldsymbol{v}}{\boldsymbol{v}^T \boldsymbol{v}} \geq \sup_{\boldsymbol{Q}_k \boldsymbol{v}_k \neq 0} \frac{\boldsymbol{v}_k^T \boldsymbol{Q}_k^T \boldsymbol{H} \boldsymbol{Q}_k \boldsymbol{v}_k}{\boldsymbol{v}_k^T \boldsymbol{v}_k} = \lambda_k \geq \frac{\boldsymbol{q}_k^T \boldsymbol{H} \boldsymbol{q}_k}{\boldsymbol{q}_k^T \boldsymbol{q}_k}. \tag{5.37}$$

Combining (5.36) and (5.37) provides with $||\boldsymbol{q}_k||^2 + ||\tilde{\boldsymbol{q}}_k||^2 = ||\boldsymbol{q}||^2 = 1$ that

$$\begin{aligned} 0 \leq \lambda - \lambda_k &\leq \lambda - \frac{\boldsymbol{q}_k^T \boldsymbol{H} \boldsymbol{q}_k}{\boldsymbol{q}_k^T \boldsymbol{q}_k} \\ &= \lambda - \frac{\lambda - 2\lambda \tilde{\boldsymbol{q}}_k^T \tilde{\boldsymbol{q}}_k + \tilde{\boldsymbol{q}}_k^T \boldsymbol{H} \tilde{\boldsymbol{q}}_k}{1 - \tilde{\boldsymbol{q}}_k^T \tilde{\boldsymbol{q}}_k} = \frac{\lambda \tilde{\boldsymbol{q}}_k^T \tilde{\boldsymbol{q}}_k - \tilde{\boldsymbol{q}}_k^T \boldsymbol{H} \tilde{\boldsymbol{q}}_k}{1 - ||\tilde{\boldsymbol{q}}_k||^2} \\ &= \left(\lambda - \frac{\tilde{\boldsymbol{q}}_k^T \boldsymbol{H} \tilde{\boldsymbol{q}}_k}{\tilde{\boldsymbol{q}}_k^T \tilde{\boldsymbol{q}}_k} \right) \frac{||\tilde{\boldsymbol{q}}_k||^2}{1 - ||\tilde{\boldsymbol{q}}_k||^2} \leq 2 ||\boldsymbol{H}|| \frac{||\tilde{\boldsymbol{q}}_k||^2}{1 - ||\tilde{\boldsymbol{q}}_k||^2}, \end{aligned}$$

which is equivalent to

$$0 \leq (\lambda - \lambda_k)(1 - ||\tilde{\boldsymbol{q}}_k||^2) \leq 2 ||\boldsymbol{H}|| \, ||\tilde{\boldsymbol{q}}_k||^2 .$$

Hence,

$$0 \leq \lambda - \lambda_k \leq (2 ||\boldsymbol{H}|| + \lambda - \lambda_k) ||\tilde{\boldsymbol{q}}_k||^2 \leq 4 ||\boldsymbol{H}|| \, ||\tilde{\boldsymbol{q}}_k||^2 \leq 4C_H ||\tilde{\boldsymbol{q}}_k||^2 ,$$

and therefore, using (5.35),

$$\int_a^b |\lambda(t) - \lambda_k(t)| dt \leq 4C_H \int_a^b ||\tilde{\boldsymbol{q}}_k(t)||^2 \, dt = 4C_H \sum_{j>k} \sigma_j^2$$

as claimed. □

Remark 5.2.12. If $\boldsymbol{R}$ is rank deficient, the above argument is still valid simply by replacing $\boldsymbol{\Sigma}^{-1}$ with the pseudo-inverse $\boldsymbol{\Sigma}^{+}$.

This result can be applied directly in the context of approximating the logarithmic norms of the local Jacobians.

Proposition 5.2.13 (Jacobian partial similarity transform). *Let $\boldsymbol{C}\boldsymbol{C}^T$ denote the Cholesky decomposition of the weighting matrix $\boldsymbol{G}$, and consider the family of symmetric matrices*

$$\boldsymbol{H}(t) := \frac{1}{2}\left(\boldsymbol{C}^T\boldsymbol{J}(\boldsymbol{x}^r(t))\boldsymbol{C}^{-T} + \left(\boldsymbol{C}^T\boldsymbol{J}(\boldsymbol{x}^r(t))\boldsymbol{C}^{-T}\right)^T\right).$$

Then, we have the corresponding values $C_H > 0$, $\{\sigma_i\}_{i=1\dots d}$ and $\boldsymbol{Q} \in \mathbb{R}^{d\times d}$ from Theorem 5.2.11, that allow us to write

$$\begin{aligned}\lambda(t) &= L_G[\boldsymbol{J}(\boldsymbol{x}^r(t))],\\ \lambda_k(t) &= L_{I_k}\left[\boldsymbol{Q}_k^T\boldsymbol{C}^T\boldsymbol{J}(\boldsymbol{x}^r(t))\boldsymbol{C}^{-T}\boldsymbol{Q}_k\right].\end{aligned} \tag{5.38}$$

Now we directly obtain the estimate

$$\int_0^T \left|L_G[\boldsymbol{J}(\boldsymbol{x}^r(t))] - L_{I_k}\left[\boldsymbol{Q}_k^T\boldsymbol{C}^T\boldsymbol{J}(\boldsymbol{x}^r(t))\boldsymbol{C}^{-T}\boldsymbol{Q}_k\right]\right| dt \leq C_H \sum_{j>k}\sigma_j^2. \tag{5.39}$$

Note here that in practice the matrix $\boldsymbol{R}$ in Theorem 5.2.11 is not available. Instead, $\boldsymbol{Q}$ is obtained as the set of left singular vectors of the singular value decomposition (SVD) of a *snapshot matrix*

$$\boldsymbol{S}_n = \sqrt{\frac{b-a}{n}}\left[\boldsymbol{q}(t_1)\ \dots\ \boldsymbol{q}(t_n)\right].$$

Assuming equally spaced points $t_j \in [a,b]$ with $t_1 = a, t_n = b$, we have

$$\lim_{n\to\infty} \boldsymbol{S}_n \boldsymbol{S}_n^T = \int_a^b \boldsymbol{q}(t)\boldsymbol{q}(t)^T dt = \boldsymbol{R}.$$

Thus, for a sufficiently large n, the sum of the neglected squared singular values of $\boldsymbol{S}_n$ will be arbitrarily close to the corresponding neglected eigenvalues of $\boldsymbol{R}$ and can safely be used in the estimate. The detailed argument is similar to one given in [97]. However, for a given set of training data $X = \{\boldsymbol{x}_i \mid i = 1 \dots n\} \subseteq \Omega$, Algorithm 10 describes how to obtain $\boldsymbol{Q}$ in practice. Notationally, the method SVD performs the singular value decomposition $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{W}^T$ for a matrix $\boldsymbol{A}$.

Algorithm 10 : $\boldsymbol{Q}$ = getTransformationMatrix(X)

```
1: $\boldsymbol{S}_n \leftarrow []$
2: for all $\boldsymbol{x}_i \in X$ do
3:     $\boldsymbol{M} \leftarrow \boldsymbol{C}^T \boldsymbol{J}(\boldsymbol{x}_i)\boldsymbol{C}^{-T}$
4:     $[\lambda_i, \boldsymbol{q}_i] \leftarrow \lambda_{max}\left(\frac{1}{2}(\boldsymbol{M} + \boldsymbol{M}^T)\right)$
5:     $\boldsymbol{S}_n \leftarrow [\boldsymbol{S}_n\ \boldsymbol{q}_i]$
6: end for
7: $[\boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{W}^T] \leftarrow SVD\left(\sqrt{\frac{b-a}{n}}\boldsymbol{S}_n\right)$
8: return $\boldsymbol{Q}$
```

Before we can derive an efficient error estimator variant with the transformation introduced above, there is one problem left to deal with. The reduced matrix from the right hand side of (5.38) is of small size $k \times k$, however, its computation involves the transformed Jacobian $\boldsymbol{C}^T \boldsymbol{J}(\boldsymbol{x}^r(t))\boldsymbol{C}^{-T} \in \mathbb{R}^{d\times d}$ which makes its computation infeasible during reduced simulations. Thus, we propose to apply a Matrix-DEIM approximation, which not only reduces evaluation costs for the Jacobian but also allows an efficient offline/online decomposition of (5.38), which we will discuss in Section 5.3. A similar idea named "Multi-Component EIM" has been formulated and successfully applied in [171, §4.3.2]. Consequently, for any $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ we define the transformation

$$\begin{aligned} \Upsilon : \mathbb{R}^{d\times d} &\to \mathbb{R}^{d^2} \\ \boldsymbol{A} &\mapsto \Upsilon[\boldsymbol{A}] := \left(A_1^T, A_2^T, \dots, A_d^T\right)^T, \end{aligned}$$

which maps the matrix entries of $\boldsymbol{A}$ column-wise into a vector (equivalent to the MatLab operation $\boldsymbol{A}(:)$, also known as *vec*-operation).

Proposition 5.2.14 (Matrix DEIM). *Choose $M_J \leq d$ and let $\hat{\boldsymbol{U}}_{M_J}, \hat{\boldsymbol{P}}_{M_J} \in \mathbb{R}^{d^2} \times M_J$ denote the corresponding matrices for the DEIM basis and interpolation points obtained by application of the known DEIM approximation procedure from Section 2.3/[23] to the vector-valued function $\Upsilon[\boldsymbol{J}(\boldsymbol{x})]$. Then, for $m_J \leq M_J$, the m_J-th order MDEIM approximation of $\boldsymbol{J}$ is given via*

$$\tilde{\boldsymbol{J}}_{m_J}(\boldsymbol{x}) := \Upsilon^{-1}\left[\hat{\boldsymbol{U}}_{m_J}(\hat{\boldsymbol{P}}_{m_J}\hat{\boldsymbol{U}}_{m_J})^{-1}\hat{\boldsymbol{P}}_{m_J}^T \Upsilon[\boldsymbol{J}(\boldsymbol{x})]\right],$$

where $\hat{\boldsymbol{U}}_{m_J} := \hat{\boldsymbol{U}}_{M_J}(:, 1{:}m_J)$ and $\hat{\boldsymbol{P}}_{m_J} := \hat{\boldsymbol{P}}_{M_J}(:, 1{:}m_J)$.

To this end, no extra assumptions on $\boldsymbol{f}$ need to be made that are not already required by the standard DEIM, as the point-wise evaluations of the Jacobian entries can always be approximated via finite differences and point-wise evaluation of the underlying $\boldsymbol{f}$. Of course, direct computation of Jacobian entries is preferred if possible. Moreover, the evaluation technique for reduced variables $\boldsymbol{V}\boldsymbol{z}(t)$ carries over directly as proposed in the original work [23, §3.5]. We would also like to mention that any matrix approximation technique using linear combinations of basis matrices could be applied in this context, see [21] for an approach using a POD-basis with least squares weights and structure-preserving constraints.

Next we investigate the approximation quality for the logarithmic norm of the similarity transformed, MDEIM Jacobian $L_{I_k}\left[\boldsymbol{Q}_k^T \boldsymbol{C}^T \tilde{\boldsymbol{J}}_{m_J}(\boldsymbol{y}^r(t))\boldsymbol{C}^{-T}\boldsymbol{Q}_k\right]$ against those of the true Jacobians $L_G[\boldsymbol{J}(\boldsymbol{y}^r(t))]$. Similar to Section 5.2.2, this is done outside the scope of the error estimation process, where we shall use the offline data from the 1D viscous Burgers model of Section 3.4.3

again. Consequently, Figure 5.10 shows the maximum (left) and mean (right)

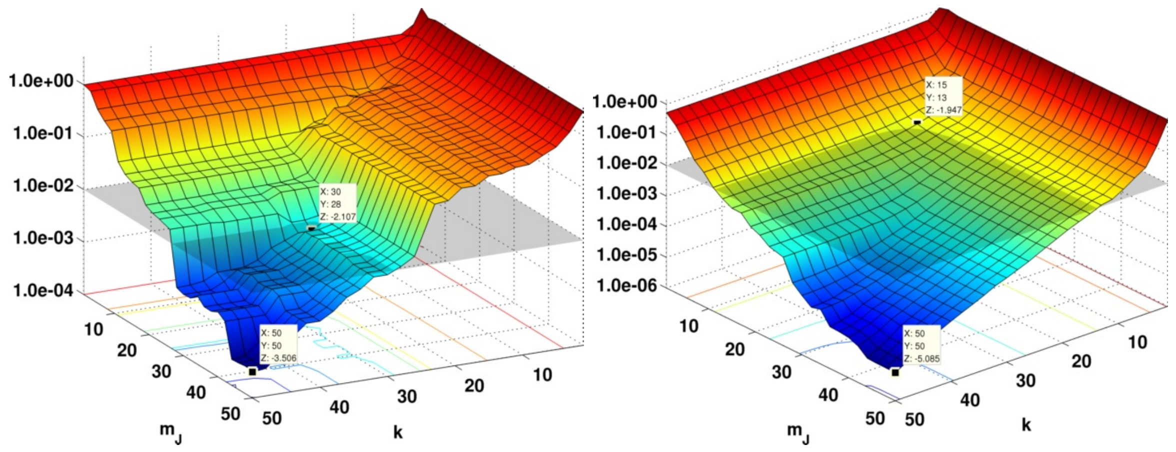


Figure 5.10: Maximum (left) and mean (right) approximation error of the logarithmic norm over X

logarithmic norm approximation error over Y for different m_J and k values. While values of $m_J = 13, k = 15$ are sufficient on average to have less than 1% relative error, in the worst case $m_J = 28, k = 30$ are needed to ensure the same tolerance over Y. But even for maximal $m_J = k = 50$, the relative errors are already only 3.162×10^{-4} (worst case) and 8.222×10^{-6} (average). These results strongly indicate that both performing the similarity transformation and MDEIM approximation of Propositions 5.2.13 and 5.2.14, respectively, are suitable for a low-cost but accurate approximation of Jacobian logarithmic norms.

Now, using the above results from Propositions 5.2.13 and 5.2.14, we derive an efficient approximation of the estimator introduced in Theorem 5.2.9, where the computational complexity for obtaining $\beta(t)$ (see (5.31c, p.184)) is independent of the high dimension d.

Theorem 5.2.15 (Estimator using local Jacobian logarithmic norms)**.** *Let the conditions from Theorem 5.2.8, Propositions 5.2.13 and 5.2.14 hold and replace (5.31c, p.184) by*

$$\beta(t) := L_{I_k}\left[\boldsymbol{Q}_k^T \boldsymbol{C}^T \tilde{\boldsymbol{J}}_{m_J}(\boldsymbol{x}^r(t)) \boldsymbol{C}^{-T} \boldsymbol{Q}_k\right]. \tag{5.40}$$

Then the estimator (5.31a, p.184) can be approximated arbitrarily close with increasing m_J and k and is exactly reproduced for $m_J = d^2, k = d$.

Proof. We have $\boldsymbol{J} \equiv \tilde{\boldsymbol{J}}_{m_J}$ for $m_J = d^2$ as each entry of the Jacobian is interpolated. Together with (5.39, p.189) for $k = d$ we obtain equality of (5.31c, p.184) and (5.40). □

Remark 5.2.16. One other obvious alternative in order to obtain an approximation of the Jacobian logarithmic norm is to use the reduced projected Jacobian $\boldsymbol{W}^T\boldsymbol{J}(\boldsymbol{x}^r(t))\boldsymbol{V}$ and thus only solve an eigenvalue problem of size $r \ll d$. When also applying the MDEIM approximation of Proposition 5.2.14, the only real difference lies in the transformation with $\boldsymbol{V}, \boldsymbol{W}$ instead of $\boldsymbol{Q}_k$, with the advantage of not having to compute $\boldsymbol{Q}$. However, as the projection subspace spanned by $\boldsymbol{V}$ is not designed to preserve any eigenvalues and the results from Proposition 5.2.13 are "optimal" in a sense, this approach is expected to have a lower approximation quality of the logarithmic norms.

Addressing this problem by incorporating suitable information into $\boldsymbol{V}, \boldsymbol{W}$ is certainly an interesting question for future work, but does not essentially avoid the necessity of a training sample set of eigenvectors during subspace computation.

5.2.5 General affine-parameterized setting

In this section we shall transfer the results to the case of more complex dynamical systems. In detail, we shall assume the structure introduced in Section 1.3, (1.12, p.19), where all previous definitions remain valid unless explicitly indicated otherwise. Let $\boldsymbol{f} : \mathcal{D} \to \Omega$ be Lipschitz-continuous w.r.t. the first argument. Moreover, if any of the components $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{x}_0$ are not given in the time- and parameter-affine form, it can be readily obtained by application of the MDEIM 5.2.14/DEIM 2.43 methods as approximation.

Now, applying the (Petrov-)Galerkin projection with $\boldsymbol{V}, \boldsymbol{W}$ and substituting the DEIM approximation of $\boldsymbol{f}(\boldsymbol{x}, t, \boldsymbol{\mu})$ to the system (1.12, p.19) yields the reduced system as in (3.4, p.88). For this case, the error system reads as

$$\begin{aligned} \boldsymbol{e}'(t) &= \boldsymbol{A}(t,\boldsymbol{\mu})\boldsymbol{x}(t) - \boldsymbol{V}\boldsymbol{W}^T\boldsymbol{A}(t,\boldsymbol{\mu})\boldsymbol{V}\boldsymbol{z}(t) + \boldsymbol{f}(\boldsymbol{x}(t),t,\boldsymbol{\mu}) \\ &\quad - \boldsymbol{V}\boldsymbol{W}^T\hat{\boldsymbol{f}}_m(\boldsymbol{x}^r(t),t,\boldsymbol{\mu}) + \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\boldsymbol{B}(t,\boldsymbol{\mu})\boldsymbol{u}(t), \\ \boldsymbol{e}(0) &= \boldsymbol{x}_0(\boldsymbol{\mu}) - \boldsymbol{V}\boldsymbol{W}^T\boldsymbol{x}_0(\boldsymbol{\mu}), \end{aligned} \tag{5.41}$$

where we omit the implicit dependence of $\boldsymbol{e}(t)$ on the current parameter $\boldsymbol{\mu} \in \mathcal{P}$ for improved readability. Recall $E_0(\boldsymbol{\mu}) = ||\left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\boldsymbol{x}_0(\boldsymbol{\mu})||_G$ as the initial error. Next we state the a-posteriori error estimation result from Theorem 5.2.9 in the context of the generalized system 1.12.

Theorem 5.2.17 (A-posteriori error estimation for parameterized DEIM reduced systems)**.** *Let the conditions from Theorem 5.2.8 hold for $\boldsymbol{f}(\boldsymbol{x}(t),t,\boldsymbol{\mu})$. Then $\forall\, \boldsymbol{\mu} \in \mathcal{P}$ and all inputs $\boldsymbol{u}(t)$ the state space error is rigorously bounded via*

$$||\boldsymbol{e}(t;\boldsymbol{\mu})||_G \leq \Delta_{EI}(t,\boldsymbol{\mu}) \;\; \forall\, t \in [0,T],$$

with

$$\begin{aligned} \Delta_{EI}(t,\boldsymbol{\mu}) &:= \int_0^t \alpha(s,\boldsymbol{\mu}) e^{\int_s^t \beta(r,\boldsymbol{\mu})dr} ds + e^{\int_0^t \beta(r,\boldsymbol{\mu})dr} E_0(\boldsymbol{\mu}), \\ \alpha(t,\boldsymbol{\mu}) &:= \Big|\Big|\left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\boldsymbol{A}(t,\boldsymbol{\mu})\boldsymbol{V}\boldsymbol{z}(t) + \Pi_{m'}(\boldsymbol{I} - \Pi_m)\boldsymbol{f}(\boldsymbol{x}^r(t),t,\boldsymbol{\mu}) \\ &\quad + \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\left(\Pi_m \boldsymbol{f}(\boldsymbol{x}^r(t),t,\boldsymbol{\mu}) + \boldsymbol{B}(t,\boldsymbol{\mu})\boldsymbol{u}(t)\right)\Big|\Big|_G, \end{aligned} \tag{5.42a}$$

$$\beta(t,\boldsymbol{\mu}) := \sum_{i=1}^{Q_A} |\theta_i^A(t,\mu)| L_G[\boldsymbol{A}_i] + L_G\left[\boldsymbol{f}(\cdot,t,\boldsymbol{\mu})\right](\boldsymbol{x}^r(t)), \tag{5.42b}$$

and the matrix definitions from Lemma 5.2.7, (5.27, p.180).

Proof. The proof is along the lines of the proof of Theorem 5.2.9. We introduce the shorthands

$$\tilde{\boldsymbol{A}}(t,\boldsymbol{\mu}) := \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\boldsymbol{A}(t,\boldsymbol{\mu})\boldsymbol{V}, \;\; \tilde{\boldsymbol{B}}(t,\boldsymbol{\mu}) := \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\boldsymbol{B}(t,\boldsymbol{\mu}),$$

easily verify

$$\boldsymbol{A}(t,\boldsymbol{\mu})\boldsymbol{x}(t) - \boldsymbol{V}\boldsymbol{W}^T\boldsymbol{A}(t,\boldsymbol{\mu})\boldsymbol{V}\boldsymbol{z}(t) = \boldsymbol{A}(t,\boldsymbol{\mu})\boldsymbol{e}(t) + \tilde{\boldsymbol{A}}(t,\boldsymbol{\mu})\boldsymbol{z}(t), \quad (5.43)$$

and see with (1.11, p.18) that

$$L_G[\boldsymbol{A}(t,\boldsymbol{\mu})] = \sup_{\boldsymbol{x}\in\mathbb{R}^d\setminus\{0\}} \sum_{i=1}^{Q_A} \theta_i^A(t,\boldsymbol{\mu}) \frac{\langle \boldsymbol{x}, \boldsymbol{A}_i\boldsymbol{x}\rangle_G}{||\boldsymbol{x}||_G^2} \leq \sum_{i=1}^{Q_A} |\theta_i^A(t,\boldsymbol{\mu})| L_G[\boldsymbol{A}_i]. \quad (5.44)$$

Now, from (5.41) we obtain using (5.43),(5.44) and (5.32, p.184) that

$$\begin{aligned}
&\langle \boldsymbol{e}(t), \boldsymbol{e}'(t)\rangle_G \\
&= \langle \boldsymbol{e}(t), \boldsymbol{A}(t,\boldsymbol{\mu})\boldsymbol{e}(t)\rangle_G + \langle \boldsymbol{e}(t), \boldsymbol{f}(\boldsymbol{x}(t),t,\boldsymbol{\mu}) - \boldsymbol{f}(\boldsymbol{x}^r(t),t,\boldsymbol{\mu})\rangle_G \\
&\quad + \left\langle \boldsymbol{e}(t), \tilde{\boldsymbol{A}}(t,\boldsymbol{\mu})\boldsymbol{z}(t) + \tilde{\boldsymbol{B}}(t,\boldsymbol{\mu})\boldsymbol{u}(t)\right\rangle_G \\
&\quad + \left\langle \boldsymbol{e}(t), \boldsymbol{f}(\boldsymbol{x}^r(t),t,\boldsymbol{\mu}) - \boldsymbol{V}\boldsymbol{W}^T\hat{\boldsymbol{f}}_m(\boldsymbol{x}^r(t),t,\boldsymbol{\mu})\right\rangle_G \\
&\leq L_G[\boldsymbol{A}(t,\boldsymbol{\mu})]\, ||\boldsymbol{e}(t)||_G^2 + L_G\big[\boldsymbol{f}(\cdot,t,\boldsymbol{\mu})\big](\boldsymbol{x}^r(t))\, ||\boldsymbol{e}(t)||_G^2 \\
&\quad + \left\langle \boldsymbol{e}(t), \tilde{\boldsymbol{A}}(t,\boldsymbol{\mu})\boldsymbol{z}(t) + \tilde{\boldsymbol{B}}(t,\boldsymbol{\mu})\boldsymbol{u}(t)\right\rangle_G \\
&\quad + \big\langle \boldsymbol{e}(t), \big(\Pi_{m'}(\boldsymbol{I} - \Pi_m) + \big(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\big)\Pi_m\big)\, \boldsymbol{f}(\boldsymbol{x}^r(t),t,\boldsymbol{\mu})\big\rangle_G \\
&\leq \beta(t,\boldsymbol{\mu})\, ||\boldsymbol{e}(t)||_G^2 + ||\boldsymbol{e}(t)||_G\, \alpha(t,\boldsymbol{\mu}).
\end{aligned}$$

Application of the comparison Lemma 5.0.1 as before completes the proof.

□

Following the same arguments of Theorem 5.2.15, efficient approximations of the general case estimator introduced in Theorem 5.2.17 are also possible using Propositions 5.2.14 and 5.2.13, i.e. by substituting

$$\beta(t) = L_{I_k}\big[\boldsymbol{Q}_k^T\boldsymbol{C}^T\tilde{\boldsymbol{J}}_{m_J}(\boldsymbol{x}^r(t),t,\boldsymbol{\mu})\boldsymbol{C}^{-T}\boldsymbol{Q}_k\big].$$

Note that output-error estimations as introduced in Section 5.1.3.1 are applicable here, too.

5.3 Offline/online decomposition

The computations for the error estimator from Theorem 5.2.17 and the local logarithmic Lipschitz constant approximation of Theorem 5.2.15 allow a decomposition into an offline/online fashion as already applied in Sections 5.1.4.1 and 3.3. Basically, using the scalar product representation for the squared $\alpha(t)$ term (5.42a) (in the spirit of $||\boldsymbol{x}||_G^2 = \boldsymbol{x}^T\boldsymbol{G}\boldsymbol{x}$) yields a sum of various components, of which all large scale quantities can be precomputed. We state the decomposition here for the context of Theorem 5.2.17, as it also covers the case for Theorem 5.2.9. From the offline stage as depicted in Section 3.3, we are already given a subspace projection matrix $\boldsymbol{V}$ and the DEIM approximation (2.43, p.81) with corresponding maximal order M, basis $\mathcal{U}$ and interpolation points $\mathcal{E}$ as in Section 2.3. Then the following stages can be identified in order to compute the proposed error estimators, where we assume the indices i, j to run over the appropriate ranges given by Q_A, Q_B and Q_0.

Offline stage I This has to be done only once for each system setting.

1. Compute offline terms for the purely $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{x}_0$-related components as

$$\begin{aligned} \tilde{\boldsymbol{x}}_i^0 &:= \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\boldsymbol{x}_i^0, & \tilde{\boldsymbol{x}}_{ij}^0 &:= (\tilde{\boldsymbol{x}}_i^0)^T\boldsymbol{G}\tilde{\boldsymbol{x}}_j^0, \\ \tilde{\boldsymbol{A}}_i &:= \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\boldsymbol{A}_i\boldsymbol{V}, & \tilde{\boldsymbol{B}}_i &:= \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\tilde{\boldsymbol{B}}_i, \\ \boldsymbol{M}_{6,ij} &:= \tilde{\boldsymbol{A}}_i^T\boldsymbol{G}\tilde{\boldsymbol{A}}_j, & \boldsymbol{M}_{9,ij} &:= \tilde{\boldsymbol{A}}_i^T\boldsymbol{G}\tilde{\boldsymbol{B}}_j, \\ \boldsymbol{M}_{12,ij} &:= \tilde{\boldsymbol{B}}_i^T\boldsymbol{G}\tilde{\boldsymbol{B}}_j. \end{aligned}$$

2. Choose a $M_J \leq d$ as maximum order and compute the MDEIM as in Proposition 5.2.14, yielding the matrices $\hat{\boldsymbol{U}}_{M_J}, \hat{\boldsymbol{P}}_{M_J} \in \mathbb{R}^{d^2 \times M_J}$.

3. Compute $\boldsymbol{Q} \in \mathbb{R}^{d\times d}$ as in Proposition 5.2.13 or Algorithm 10.

Offline stage II For every (new) choice of $m \leq M, m' \leq M - m, m_J \leq M_J, k \leq d$ perform the steps

1. Compute central quantities for $\alpha(t)$ term (5.42a, p.194). Here, we prefer the equivalent expression

$$\begin{aligned}
&\left(\Pi_{m'}(\boldsymbol{I} - \Pi_m) + \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\Pi_m\right) \boldsymbol{f}(\boldsymbol{x}^r(t), t, \boldsymbol{\mu}) \\
=\ &\left(\left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right) - \Pi_{m'}\right) \Pi_m \boldsymbol{f}(\boldsymbol{y}^r(t), t, \boldsymbol{\mu}) + \Pi_{m'} \boldsymbol{f}(\boldsymbol{y}^r(t), t, \boldsymbol{\mu}) \\
=\ &\boldsymbol{M}_1 \boldsymbol{P}_m^T \boldsymbol{f}(\boldsymbol{y}^r(t), t, \boldsymbol{\mu}) - \boldsymbol{M}_2 \boldsymbol{P}_{m'}^T \boldsymbol{f}(\boldsymbol{y}^r(t), t, \boldsymbol{\mu})
\end{aligned}$$

for the DEIM approximation error estimate, where $\boldsymbol{M}_1, \boldsymbol{M}_2$ are defined as

$$\begin{aligned}
\boldsymbol{M}_1 &:= \left(\left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right) - \Pi_{m'}\right) \boldsymbol{U}_m \left(\boldsymbol{P}_m^T \boldsymbol{U}_m\right)^{-1}, \\
\boldsymbol{M}_2 &:= (\boldsymbol{I} - \Pi_m)\boldsymbol{U}_{m'} \left(\boldsymbol{P}_m^T (\boldsymbol{I} - \Pi_m)\boldsymbol{U}_{m'}\right)^{-1}.
\end{aligned}$$

Basically, this formulation allows to nicely separate pre-computable matrices $\boldsymbol{M}_1, \boldsymbol{M}_2$ from the evaluation of $\boldsymbol{f}$ at both sets of interpolation.

2. Compute mixed offline quantities for the $\alpha(t)$ term

$$\begin{aligned}
\boldsymbol{M}_3 &:= \boldsymbol{M}_1^T \boldsymbol{G} \boldsymbol{M}_1, & \boldsymbol{M}_4 &:= \boldsymbol{M}_1^T \boldsymbol{G} \boldsymbol{M}_2, & \boldsymbol{M}_5 &:= \boldsymbol{M}_2^T \boldsymbol{G} \boldsymbol{M}_2. \\
\boldsymbol{M}_{7,i} &:= \boldsymbol{M}_1^T \boldsymbol{G} \tilde{\boldsymbol{A}}_i, & \boldsymbol{M}_{8,i} &:= \boldsymbol{M}_2^T \boldsymbol{G} \tilde{\boldsymbol{A}}_i, & i &= 1 \dots Q_A, \\
\boldsymbol{M}_{10,j} &:= \boldsymbol{M}_1^T \boldsymbol{G} \tilde{\boldsymbol{B}}_j, & \boldsymbol{M}_{11,j} &:= \boldsymbol{M}_2^T \boldsymbol{G} \tilde{\boldsymbol{B}}_j, & j &= 1 \dots Q_B.
\end{aligned}$$

3. Compute offline vectors for the Jacobian MDEIM via

$$\begin{aligned}
\hat{\boldsymbol{U}} &:= \hat{\boldsymbol{U}}_{m_J} (\hat{\boldsymbol{P}}_{m_J}^T \hat{\boldsymbol{U}}_{m_J})^{-1}, \\
\hat{\boldsymbol{U}}_{m_J} &:= \hat{\boldsymbol{U}}_{M_J}[:, 1{:}m_J], & \hat{\boldsymbol{P}}_{m_J} &:= \hat{\boldsymbol{P}}_{M_J}[:, 1{:}m_J].
\end{aligned}$$

4. Select partial similarity transform matrix of size k as $\boldsymbol{Q}_k := \boldsymbol{Q}[:, 1{:}k]$

and compute

$$\tilde{\boldsymbol{U}}[:,j] := \Upsilon_k[\boldsymbol{Q}_k^T \boldsymbol{C}^T \Upsilon^{-1}[\hat{\boldsymbol{U}}[:,j]]\boldsymbol{C}^{-T}\boldsymbol{Q}_k] \in \mathbb{R}^{k^2}, \quad j = 1\dots m_J,$$

where Υ_k denotes the same transformation as Υ but for $k \times k$ matrices.

Online stage In the online stage we can compute the error estimator by solving

$$\Delta'(t) = \beta(t)\Delta(t) + \alpha(t), \qquad \Delta(0) = E_0(\boldsymbol{\mu}), \qquad t \in [0,T],$$

with

$$E_0(\boldsymbol{\mu}) = \left|\left|\left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\boldsymbol{x}_0(\boldsymbol{\mu})\right|\right|_G = \left(\sum_{i,j}^{Q_0} \theta_i^0(\mu)\theta_j^0(\mu)\tilde{\boldsymbol{x}}_{ij}^0\right)^{\frac{1}{2}},$$

which may be computed along with the reduced order trajectory by adjoining a single scalar equation to the reduced system introduced in Section 3.3.2. The $\alpha(t)$ term (5.42a, p.194) is given by

$$\begin{aligned}\alpha(t,\boldsymbol{\mu}) = \Big(&\boldsymbol{v}_1^T\boldsymbol{M}_3\boldsymbol{v}_1 - 2\boldsymbol{v}_1^T\boldsymbol{M}_4\boldsymbol{v}_2 + \boldsymbol{v}_2^T\boldsymbol{M}_5\boldsymbol{v}_2 + \boldsymbol{z}(t)^T\boldsymbol{M}_6(t,\boldsymbol{\mu})\boldsymbol{z}(t)\\ &+ 2\boldsymbol{v}_1\boldsymbol{M}_7(t,\boldsymbol{\mu})\boldsymbol{z}(t) - 2\boldsymbol{v}_2\boldsymbol{M}_8(t,\boldsymbol{\mu})\boldsymbol{z}(t) + 2\boldsymbol{z}(t)^T\boldsymbol{M}_9(t,\boldsymbol{\mu})\boldsymbol{u}(t)\\ &+ 2\boldsymbol{v}_1\boldsymbol{M}_{10}(t,\boldsymbol{\mu})\boldsymbol{u}(t) - 2\boldsymbol{v}_2\boldsymbol{M}_{11}(t,\boldsymbol{\mu})\boldsymbol{u}(t)\\ &+ \boldsymbol{u}(t)^T\boldsymbol{M}_{12}(t,\boldsymbol{\mu})\boldsymbol{u}(t)\Big)^{\frac{1}{2}},\end{aligned}$$

with the low-dimensional quantities

$$\boldsymbol{v}_1 := \boldsymbol{P}_m^T\boldsymbol{f}(\boldsymbol{y}^r(t),t,\boldsymbol{\mu}) \in \mathbb{R}^m, \qquad \boldsymbol{v}_2 := \boldsymbol{P}_{m'}^T\boldsymbol{f}(\boldsymbol{y}^r(t),t,\boldsymbol{\mu}) \in \mathbb{R}^{m'},$$

$$\boldsymbol{M}_6(t,\boldsymbol{\mu}) := \sum_{i,j=1}^{Q_A} \theta_i^A(t,\boldsymbol{\mu})\theta_j^A(t,\boldsymbol{\mu})\boldsymbol{M}_{6,ij},$$

$$\begin{aligned}
\boldsymbol{M}_{\{7,8\}}(t,\boldsymbol{\mu}) &:= \sum_{i=1}^{Q_A} \theta_i^A(t,\boldsymbol{\mu})\boldsymbol{M}_{\{7,8\},i},\\
\boldsymbol{M}_9(t,\boldsymbol{\mu}) &:= \sum_{i=1}^{Q_A}\sum_{j=1}^{Q_B} \theta_i^A(t,\boldsymbol{\mu})\theta_j^B(t,\boldsymbol{\mu})\boldsymbol{M}_{9,ij},\\
\boldsymbol{M}_{\{10,11\}}(t,\boldsymbol{\mu}) &:= \sum_{i=1}^{Q_B} \theta_i^B(t,\boldsymbol{\mu})\boldsymbol{M}_{\{10,11\},i},\\
\boldsymbol{M}_{12}(t,\boldsymbol{\mu}) &:= \sum_{i,j=1}^{Q_B} \theta_i^B(t,\boldsymbol{\mu})\theta_j^B(t,\boldsymbol{\mu})\boldsymbol{M}_{12,ij}.
\end{aligned}$$

Depending on the setting, $\beta(t)$ can be obtained either via (5.31c, p.184) or via (5.40, p.192).

The computational complexity of the offline stages is $\mathcal{O}\left(d^2\right)$ and $\mathcal{O}\left(d\right)$, respectively, whereas the online stage involves operations of complexity $\mathcal{O}\left(\max\{k^3, m^2, m'^2, m_J^2\}\right)$. For large systems (spatially discretized PDE's, for example), the sparsity of the Jacobian can be used in the offline stage I, so that instead of d^2 rows only the number of rows with nonzero entries have to be stored. We emphasize that the MDEIM approximation of the Jacobian not only gives a fast approximation, but is also crucial in order to apply the similarity transformation at the offline phase, which would in general not be possible if full Jacobians were used.

5.4 Numerical experiments

In this Section we demonstrate the effectiveness of this a posteriori error estimation procedure with two computational examples. However, before we detail the results we would like to remark on an issue arising due to the offline/online decomposition of the $\alpha(t)$ term (5.42a, p.194) in the error estimators. Therein, each component is projected *separately* to build the reduced quantities $\boldsymbol{M}_{xx}$ (cf. Section 5.3, offline phases I+II). It is evident that the projection subspace $\mathcal{V}$ should *simultaneously* approximate the $\boldsymbol{f}$-values and column spans of $\boldsymbol{A}, \boldsymbol{B}$ (if applicable) to allow for a good estimation in each component. This of course is not to be expected when the trajectory data used to compute $\mathcal{V}$ stems from the *sum* of all right hand side components of the considered system. Preliminary experiments confirmed that, if the respective values had not been included in the subspace computation process, the error estimates would not go below a certain order of magnitude. Hence, we included the respective $\boldsymbol{f}(\boldsymbol{x})$ and/or $\boldsymbol{A}\boldsymbol{x}$-evaluations in the subspace computation process of the numerical examples to follow. This, of course, leads to higher reduced dimensions in the worst case, and investigations on whether those extended subspaces could be used *only* within the projected error estimator terms yet need to be done.

The first example is the 1D viscous Burger's equation introduced in Section 3.4.3 and the second involves the 2D reaction-diffusion model for cell apoptosis introduced in Section 3.4.2. With these examples, the validity of the assumption of near exactness for modest values of m' is illustrated (see Section 5.2.2). We also show the effectiveness of the local logarithmic Lipschitz constant estimates (see Section 5.2.4) and we compare our estimate based on inexact quantities to one that uses exact quantities.

5.4.1 1D viscous Burgers equation

We recall from Section 3.4.3 and Figure 3.9 that for $\mu = 0.04$ a value of $m = 12$ was sufficient to yield a relative error of less than one percent. In order to focus our discussion on the influences of the various choices m', m_J and k, we shall fix those values for the remainder of the experiments. However, we emphasize that the error estimators are applicable for any choice of $m \leq M,\ \mu \in \mathcal{P}$ and input $\boldsymbol{u}(t)$.

Next, we recall the estimator structure from Theorems 5.2.15 and 5.2.17. There, different choices of m' influence the α terms (5.31b, p.184), (5.42a, p.194), and m_J (Jacobian MDEIM approximation order) and k (partial similarity transformation size) affect the β terms (5.40, p.192), (5.42b, p.194), respectively.

In order to pointedly investigate the various choices and their influences on the estimation results, we discuss the effects for different m' values in Section 5.4.1.1 and focus on the local logarithmic Lipschitz constant estimation in Sections 5.4.1.2 and 5.4.1.3.

5.4.1.1 DEIM approximation error estimation analysis

The experiments pursued in Section 5.2.2 show that the DEIM approximation error estimation is indeed very useful in practical applications, however, thus so far giving insights only to the m, m'-estimation technique by itself. Consequently, we consider its application in the context of the error estimator derived in Theorem 5.2.17 in the following. In order to be able to focus on this aspect only, we shall use a modified variant of the estimator in Theorem 5.2.17, which avoids using MDEIM approximated Jacobians and partial similarity transformations but still uses directly computable local Lipschitz values. Therefore, during the next experiment we shall use the actual values

$$\beta(t) := \frac{\langle \boldsymbol{y}(t) - \boldsymbol{y}^r(t), \boldsymbol{f}(\boldsymbol{y}(t)) - \boldsymbol{f}(\boldsymbol{y}^r(t))\rangle_G}{||\boldsymbol{y}(t) - \boldsymbol{y}^r(t)||_G^2}, \tag{5.45}$$

instead of $L_{I_k}\left[\boldsymbol{Q}_k^T \boldsymbol{C}^T \tilde{\boldsymbol{J}}_{m_J}(\boldsymbol{y}^r(t)) \boldsymbol{C}^{-T} \boldsymbol{Q}_k\right]$ as in Theorem 5.2.15.

Moreover, we introduce a reference estimate by additionally using the true DEIM approximation error

$$\begin{array}{rl} & \boldsymbol{f}(\boldsymbol{y}^r(t)) - \boldsymbol{V}\boldsymbol{W}^T \hat{\boldsymbol{f}}_m(\boldsymbol{y}^r(t)) \\ \text{instead of} & \left(\Pi_{m'}(\boldsymbol{I} - \Pi_m) + \left(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{W}^T\right)\Pi_m\right)\boldsymbol{f}(\boldsymbol{y}^r(t)) \end{array}$$

within the $\alpha(t, \boldsymbol{\mu})$ term (5.42a, p.194). Those two changes make this reference estimator as sharp as this estimator structure allows. Of course both above modifications are not applicable in practice as they require the full system's trajectory, and are introduced here only for demonstration purposes.

To this end, Figure 5.11 shows the resulting error estimations for a selection of m' approximation error orders in a full view (left) and a zoom (right). The

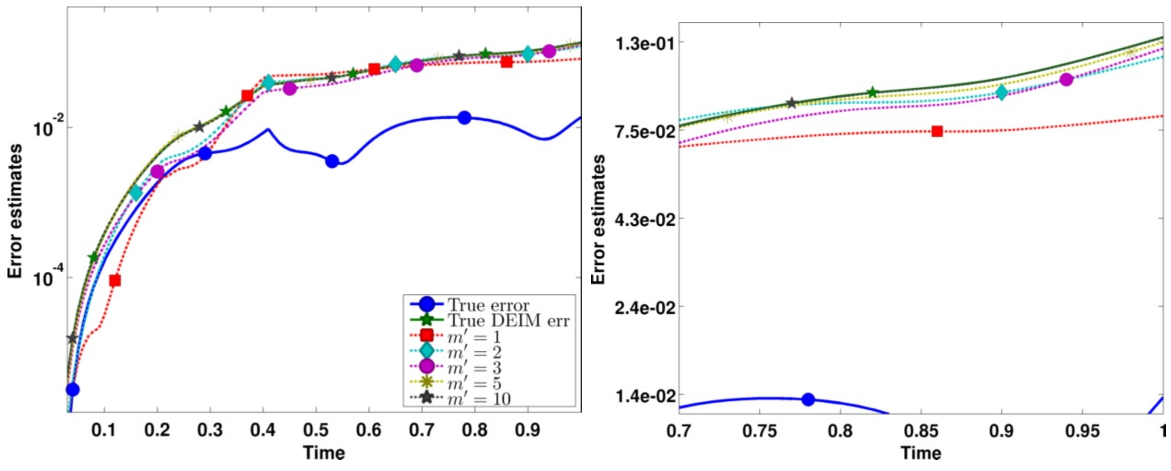


Figure 5.11: Absolute errors over time for different DEIM approximation error orders m'.

solid blue line is the true reduction error, and the green star-marked line is the reference estimate using the true DEIM approximation error. For the reduced simulation we have $||\boldsymbol{e}(T)||_G = 0.0137$ (final time T) and the rigorous reference estimate yields $\Delta(T) = 0.135$, which means an effectivity of 9.89 for the current $\mu = 0.04$. Most importantly, this illustrates that the overall estimator structure is suitable for error estimation. Now, regarding the m' influence, at first we notice that except for $m' = 1$ the reduction error estimates using the m'-order DEIM error estimate are very close to the reference estimate, where $m' = 1$ also is the only setting which is not an upper bound as the true error is underestimated at $t \in [0.05, 0.2]$. On the other side, for $m' = 10$ the results are already indistinguishable from the reference estimate even in the zoomed detail. As this is already quite satisfactory, we will now fix $m' = 10$ for all subsequent estimations.

5.4.1.2 Local logarithmic Lipschitz constant estimations using Jacobians

As our next step, we illustrate the occurring loss in efficiency when using the Jacobian-type approximation (5.34, p.186) instead of the true local logarithmic Lipschitz constant (5.45). Figure 5.12 displays the resulting estimates in full (left) and zoom (right). Here, the green star-marked line is the same reference estimate from the previous setting (5.45) and the orange star-marked line uses the estimation via the logarithmic norm of the local Jacobian (5.34, p.186). As an intermediate variant, we additionally plot a blue square-marked estimate which uses

$$\beta(t) = \langle \boldsymbol{y}(t) - \boldsymbol{y}^r(t), J(\boldsymbol{y}^r(t))(\boldsymbol{y}(t) - \boldsymbol{y}^r(t))\rangle_G \,/\, ||\boldsymbol{y}(t) - \boldsymbol{y}^r(t)||_G^2\,,$$

i.e. the true local logarithmic Lipschitz constant of the local Jacobian. This

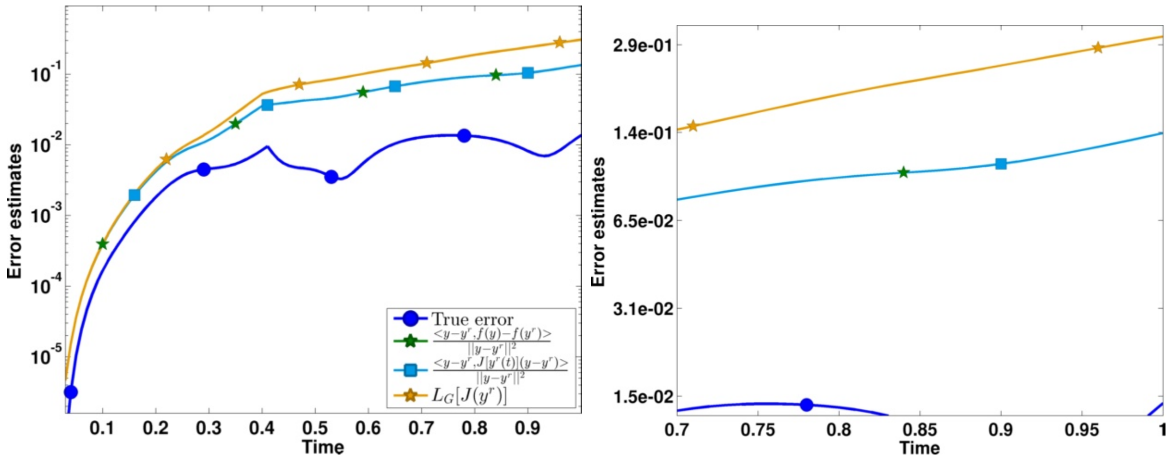


Figure 5.12: Absolute errors over time with full local Jacobian logarithmic norms

shows two things: Firstly, the estimation results for either using $\boldsymbol{f}(\boldsymbol{y}(t)) - \boldsymbol{f}(\boldsymbol{y}^r(t))$ or $J(\boldsymbol{y}^r(t))(\boldsymbol{y}(t)) - \boldsymbol{y}^r(t))$ inside the $\beta(t)$ computation are visibly indistinguishable and match on the first three digits ($\Delta(T) = 0.1354$), which makes the first order approximation already very useful even when the true error is about 1%. Secondly, using $L_G[J(\boldsymbol{y}^r(t))]$ yields a very similar estimate with $\Delta(T) = 0.308$, which is roughly double as large as the sharp reference with an effectivity of ≈ 22. This is of course a loss in efficiency, but is computed using values that are readily available during offline computations. Finally we note that the computation time for the first reference estimate was $0.43s$, but the estimate involving $L_G[J(\boldsymbol{y}^r(t))]$ took $28.27s$ to compute.

5.4.1.3 Jacobian logarithmic norm approximation

Finally, we will look into the estimation quality for different configurations of the practically applicable error estimator variant of Theorem 5.2.17, which involves the previously investigated approximations of the $\beta(t)$ term. Recall that the results of Section 5.4.1.1 led us to fix $m' = 10$ for the following experiments as this value turned out to provide a good replacement estimation for the true DEIM approximation error. Figure 5.13 shows the absolute errors over time for different matrix DEIM orders m_J and partial similarity transformation sizes k in full (left) and zoom (right). Both star-marked es-

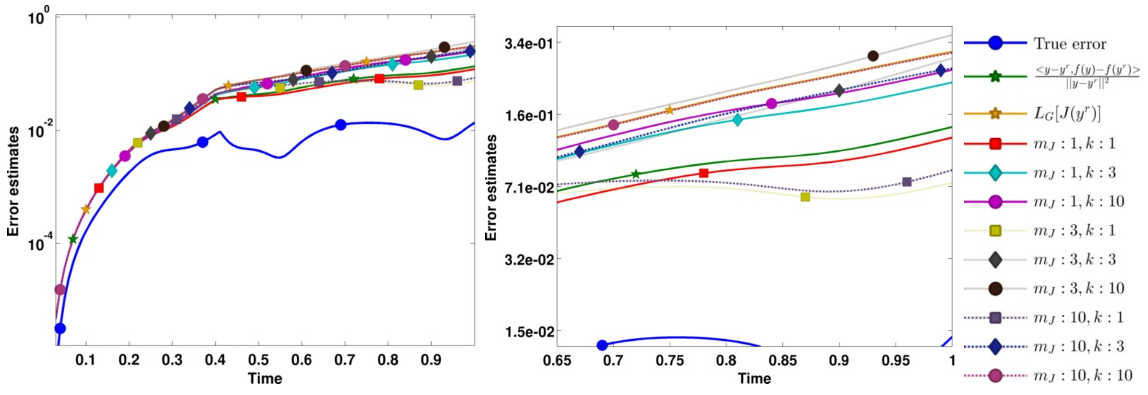


Figure 5.13: Absolute errors over time for different matrix DEIM orders m_J and partial similarity transformation sizes k.

timations denote the already known reference estimates. In all other plots, the line style is identical for different m_J values and the marker style identical for different k values. One can clearly see that for both increased m_J and k the estimations get closer to the orange reference as expected. For $m_J = k = 10$ we already obtain $\Delta(T) = 0.3045$, which matches the orange reference up to two digits but only needs a computation time of $0.48s$ (speedup of 58.8). On a closer look, one can see that estimates with the same k are relatively similar and approach the orange reference from below with increasing k. This is due to the fact that for any $\boldsymbol{J} \in \mathbb{R}^{d \times d}$

$$L_{I_k}\left[\boldsymbol{Q}_k^T \boldsymbol{J} \boldsymbol{Q}_k\right] = \max_{\boldsymbol{z} \in \mathbb{R}^k} \frac{\left\langle \boldsymbol{z}, \boldsymbol{Q}_k^T \boldsymbol{J} \boldsymbol{Q}_k \boldsymbol{z} \right\rangle}{\left\|\boldsymbol{z}\right\|^2}$$

$$= \max_{\boldsymbol{y} \in \operatorname{range}(\boldsymbol{Q}_k)} \frac{\langle \boldsymbol{y}, \boldsymbol{J}\boldsymbol{y} \rangle}{\left\|\boldsymbol{y}\right\|^2} \leq \max_{\boldsymbol{y} \in \mathbb{R}^d} \frac{\langle \boldsymbol{y}, \boldsymbol{J}\boldsymbol{y} \rangle}{\left\|\boldsymbol{y}\right\|^2} = L_{I_d}[\boldsymbol{J}],$$

i.e. the maximum eigenvalues of the similarity transformed Jacobians are bounded by the maximum eigenvalues of the full Jacobians as $\operatorname{range}(\boldsymbol{Q}_k) \subseteq \mathbb{R}^d$.

5.4.2 Cell apoptosis: 2D reaction-diffusion

As no real speedup is to be expected for most 1D cases, we shall consider a more complex 2D problem, namely the cell apoptosis model introduced in Section 3.4.2. We refer to that section for the experiment context. Figure 5.14 shows error estimation results for different configurations of the error estimator from Theorem 5.2.17. As in Section 5.4.1, we included reference

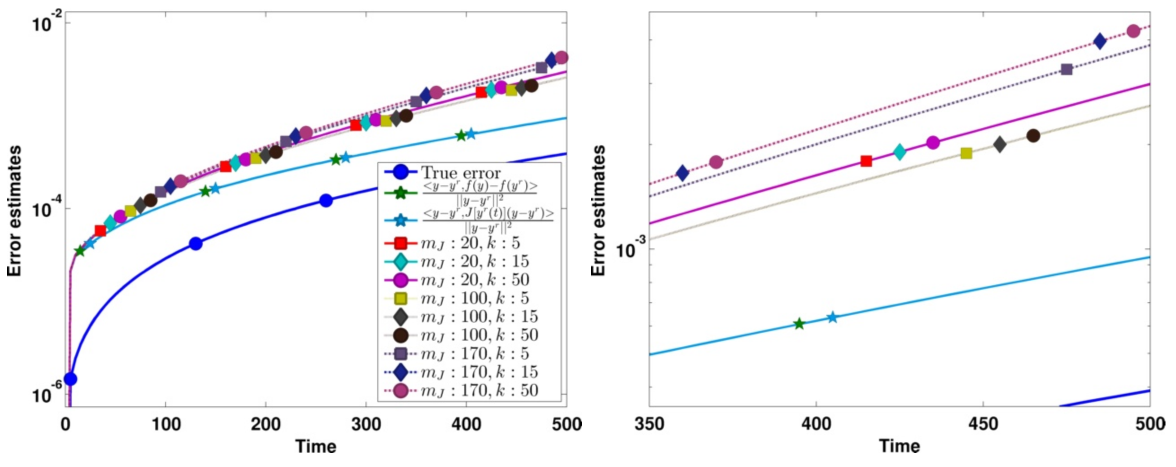


Figure 5.14: Absolute errors over time for different k, m_J combinations

estimates using the true local logarithmic Lipschitz constants of both $\boldsymbol{f}$ and the Jacobian $\boldsymbol{J}$ and use identical line- and marker-types for the same m_J/k values as before. We decided to use the true DEIM approximation error for the references and fixed $m' = 10$ for all other estimates. Analogous to the previous experiment, both reference estimates are visually indistinguishable and confirm that the approximation (5.33, p.186) via Jacobians is applicable. As the estimation results for fixed m_J are lying close to each other, different m_J values seem to have more impact on the estimation results than different k values in this case. However, the connection between increasing m_J and the estimation results is not monotonous as all estimates for $m_J = 100$ are below the ones using $m_J = 20$.

Figure 5.15 shows the relative errors for the same configurations on the left and the image on the right illustrates the computation times required for the different configurations plotted against the estimated error at $T = 500$. Note here that the times for the reference estimates also include the time needed to compute the full solution required for the local constants. Finally,

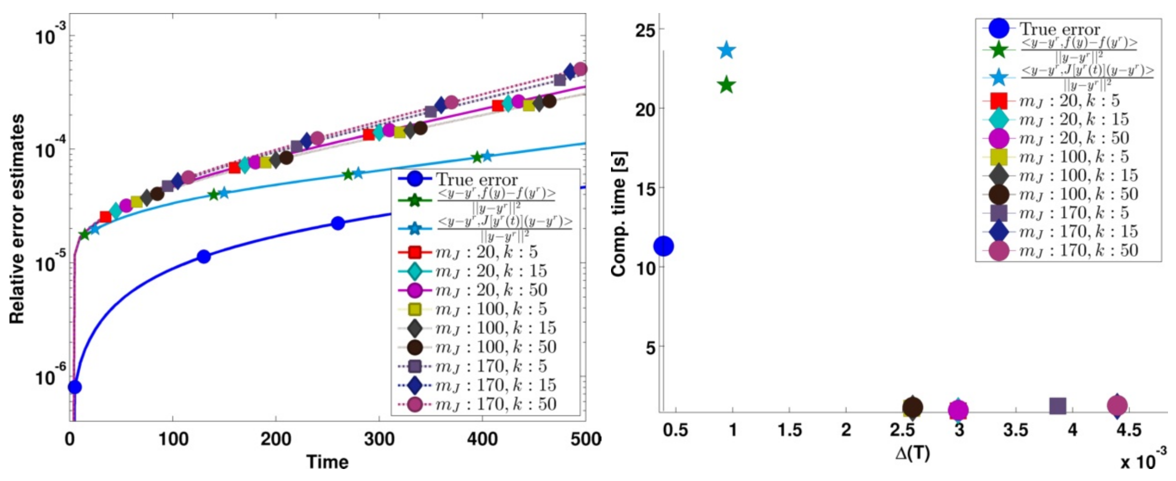


Figure 5.15: Relative errors (left) and computation times against estimates at $T = 500$ (right) for different estimator configurations.

Table 5.3 shows an overview of the used estimator configurations regarding computation times and effectivities. Remarkable at this point is that the effectivities of all estimator configurations are in the order of ten at $T = 500$. As expected, the computation times are increasing along with higher m_J and k values, but compared to the full solution the speedup including the error estimator is still about a factor of ten. This higher cost of about twice the pure reduced simulation time is the price we need to pay for the a-posteriori error estimation. Of course, seen as an absolute value, this still is subject to further improvement. However, given the achieved effectivities the estimation results are already very competitive.

Version	$\Delta(T = 500)$	Time	Effectivity
True error	3.908×10^{-4}	11.31s	1.000
$\frac{<y-y^r, f(y)-f(y^r)>}{\|\|y-y^r\|\|^2}$	9.476×10^{-4}	21.44s	2.425
$\frac{<y-y^r, J[y^r(t)](y-y^r)>}{\|\|y-y^r\|\|^2}$	9.476×10^{-4}	23.63s	2.425
$m_J : 20, k : 5$	2.989×10^{-3}	0.92s	7.649
$m_J : 20, k : 15$	2.989×10^{-3}	0.94s	7.649
$m_J : 20, k : 50$	2.989×10^{-3}	0.97s	7.649
$m_J : 100, k : 5$	2.580×10^{-3}	1.08s	6.602
$m_J : 100, k : 15$	2.588×10^{-3}	1.09s	6.622
$m_J : 100, k : 50$	2.588×10^{-3}	1.15s	6.622
$m_J : 170, k : 5$	3.869×10^{-3}	1.22s	9.900
$m_J : 170, k : 15$	4.392×10^{-3}	1.20s	1.124×10^{1}
$m_J : 170, k : 50$	4.392×10^{-3}	1.27s	1.124×10^{1}

Table 5.3: Overview for all used estimator configurations.

Conclusion

Summary

In this thesis, the fields of nonlinear approximation and model reduction were merged and we provided extensive examples supporting the usability of the presented methodology. The main goals of this thesis have been an investigation of applicability of kernel methods in the context of model order reduction and a-posteriori error estimation for nonlinear dynamical systems.

In the field of nonlinear approximation, vectorial kernel-based approximation techniques, the DEIM and a possible combination thereof was discussed. In particular, a new efficient SMO formulation of the ϵ-SVR without offset was introduced. A vectorial greedy algorithm with an improved theoretical convergence rate was developed, where the introduction of the concept of orthogonal remainders proved to build a bridge between different mathematical perspectives. Further analytical analysis of the proposed VKOGA scheme showed that a directional Hermite-type interpolation is obtained in the limit process of including two points in closing vicinity. Those findings along with supporting experiments have been published in [183]. Moreover, the "*Kernel-DEIM*" has been introduced in order to provide a black-box capable DEIM variant.

With respect to MOR, the investigated technique for nonlinear dynamical systems builds on suitable Galerkin-type subspace projection along with applications of the aforementioned nonlinear approximation techniques to obtain efficiently projectable approximations of nonlinear parts. For the kernel-based approximations, we extended the scope of kernels for which an efficient projection could be devised. Common to all the introduced schemes for parameter-dependent problems is the possibility for an offline/online decomposition to enable efficient reduced simulations, facilitated in part by the use of affine-parametric system components.

Furthermore, we put an emphasis on error estimation for the proposed projection-based MOR methods and introduced efficient a-posteriori error estimators for both kernel-based and DEIM-reduced systems. These were also published in [182, 181] and [186] respectively. Key features for the estimators are the local Lipschitz constant estimations and an application of the comparison lemma. In more detail, the kernel-based estimators further benefited from an iterative scheme, which provides a balance of estimation sharpness against computational costs. The derived error estimators are readily capable of handling nonlinear parameter- and/or time-dependent systems with inputs and outputs. Here we provided illustrating experiments for a fully parameterized case. The DEIM estimators utilized higher order DEIM basis vectors and points to efficiently estimate the local DEIM approximation error. Furthermore, a matrix DEIM of the nonlinearity Jacobians was employed along with partial similarity transformations to obtain a local first-order approximation of logarithmic Lipschitz constants.

In conclusion, some of the proposed nonlinear approximation methods were shown to be applicable for model reduction of multiscale models. By providing cheap surrogate kernel approximations, the run-time for macroscale problems with frequently solved microscale-problems can be reduced dramatically. We illustrated this effect using two real-world applications from fluid dynamics in porous media and a multibody spine model using detailed intervertebral disk models. The corresponding theory and experimental results have been published in [185].

Discussion & future work

This thesis covers aspects from quite independent research fields, which unsurprisingly leaves us with too many yet unanswered questions. However, there are some follow-up topics and issues that are promising candidates for further investigation.

Nonlinear approximation

In the field of nonlinear approximation with kernels, the considered kernel expansion structure is of the form

$$\hat{\boldsymbol{f}} = \sum_{i=1}^{n} \boldsymbol{c}_i K(\boldsymbol{x}, \cdot) \quad \text{or} \quad \hat{\boldsymbol{f}}(\boldsymbol{x}, t, \boldsymbol{\mu}) = \sum_{i=1}^{n} \boldsymbol{c}_i K(\boldsymbol{x}_i, \boldsymbol{x}) K_t(t_i, t) K_{\mathcal{P}}(\boldsymbol{\mu}_i, \boldsymbol{\mu}).$$

In the first case, this structure involves one single type of kernel with fixed hyperparameters (if available). Similar to the ideas of sparse grids [20], expansions allowing for different types of kernels or varying hyperparameters like multilevel kernel expansions (e.g. [68, 187, 63]) could yield further improvements regarding approximation quality and sparseness. Also, we did not address the important aspect of kernel hyperparameter selection, at least not more as to the extent of choosing equidistant values in some predefined range. This topic is getting continued attention for various settings, cf. [22, 85, 59, 99] for example, and is certainly an open question to be considered in the context of model reduction applications. For the time- and parameter-dependent second case, all the above applies for each the state, time or parameter domain. Additionally, it remains an open question if criteria for suitable/ideal algebraic combination of the kernels can be devised, which will certainly depend on the application.

For the ϵ-SVR case, a truly simultaneous vectorial formulation in SMO-fashion is yet missing or at least not known to the author. Similar to the VKOGA case, it is to expect that the effectively selected centers will in general differ for each component function. This requires evaluation of the kernel on all of X in the worst case, rendering the approximation less practicable due to higher evaluation costs. Consequently, an ϵ-insensitive loss function with vectorial input like

$$L : \Omega \times \Omega \to \mathbb{R}_+, \qquad (\boldsymbol{x}, \boldsymbol{y}) \mapsto ||\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x})||_\epsilon,$$

and ϵ-insensitive norm is a possible starting point for derivation of vectorial gain functions that select the center with the best approximation gain for all components simultaneously.

Regarding the VKOGA, the limit behaviour of resembling a Hermite-type interpolation for subsequent addition of identical points could be used for "hybrid" greedy algorithms that take extensions by derivatives into account. Certainly, this introduces new challenges to algorithm and/or convergence analysis due to extension of the considered dictionaries. But the fact that in some situations the inclusion of a directional derivative $\phi_{\boldsymbol{x}}^{\nabla v}$ can yield the *largest* possible gain (in native $\mathcal{H}$-norm) motivates further investigations of this topic, cf. Section 2.2.2.1.

Furthermore, the experiments considering the f/P-VKOGA suggest that this variant might be a suboptimal choice if the origin or native space $\mathcal{H}$ of the target function is unknown. In our opinion, this effect might be utilized to our advantage by using it as an indicator to assess the correct choice of RKHS for the present type of function, cf. Section 2.2.2.1. This could be done by an early-stopping criterion based on the vicinity of newly added data points to existing ones, for example.

Subspace projection based MOR approaches

As shortly remarked on in Section 3.1.2, substitution of a-posteriori error estimators in the POD-Greedy algorithm is a practical and proven approach to enable fast and adaptive subspace computation for parameterized systems. The practically applicable variant of the POD-DEIM a-posteriori error estimator presented in Theorem 5.2.17 of Section 5.2.5 is, in general, applicable for this context, too. While the principal POD-Greedy algorithm can be applied straightforwardly, some difficulties might arise running the reduced simulations in order to evaluate the error estimator. Therein, it is implied that any nonlinear approximation is available for projection into the current subspace, but for the case of VKOGA approximations we have the restriction $\boldsymbol{x}_i \in \mathcal{V}$, for example. This means the expansion centers have to come from the current reduced space $\mathcal{V}_m$ at step m. Now, this is not a problem if the approximation is built using the initial space, i.e. $\boldsymbol{x}_i \in < \boldsymbol{u}_1 >$, as the condition $\boldsymbol{x}_i \in \mathcal{V}_m$ will remain true due to the hierarchical construction $\mathcal{V}_1 \subset \mathcal{V}_2 \subset \ldots \subset \mathcal{V}_m$. However, it is to expect that the maximum possible accuracy for any approximation will be achieved by allowing the centers to stem from the current subspace $\mathcal{V}_m$, meaning it would have to be re-computed at each intermediate POD-Greedy step. To this end, an implementation of the POD-Greedy scheme including the error estimator, derivation of suitable schemes to address the above issue and an investigation regarding the overall applicability has yet to be done.

In addition to the briefly mentioned splitting techniques for the spatial-, parameter- or time-domain [53, 44, 70], truly nonlinear projection approaches could also be of interest for dimensionality reduction. Considerable approaches comprise of Kernel PCA [152], local-linear embedding [140, 46, 144] or diffusion maps [30], see [101] for an overview. The incorporation of those alternatives to Galerkin-type linear subspace projection schemes into e.g. the a-posteriori error estimators derived in this work is surely of interest for its own.

MOR with kernels and error estimation

For a-posteriori error estimation, effective and sharp local Lipschitz constant estimation has proven to be of paramount importance. While the secant-gradient based strategy introduced in Section 5.1.1 along with the iterative scheme presented in Section 5.1.2 proved to yield useful results for academic examples, cf. Section 5.1.5, there is still room for improvement regarding the kernel expansion coefficient norms in the expression

$$\beta(t) := \sum_{i=1}^{n} ||\boldsymbol{c}_i||_G \left|S_{d_i(t)}(r_{\Theta(t),d_i(t)})\right|,$$

see Theorem 5.1.9. So far, simply their norm is used, which can of course lead to larger Lipschitz constants from that side. For direct RBF methods, for example, the coefficients of direct translates tend to get large with opposite signs if two interpolation points are in close vicinity to each other. Hence, a suitable weighting of those coefficients with regards to e.g. the error direction could be a possible starting point for further improvements.

MOR with POD-DEIM and error estimation

As briefly outlined in Section 2.3.1, all kernel-based approximation methods are purely sampling-based, in the sense of not needing the approximation target for anything else but training data generation. Now, this is not entirely true for the DEIM-method, as it yet requires evaluations of selected components of the approximated function. In the case of cheap component evaluations this might not be a problem, however, if the evaluation of even a few components still turns out to be expensive or access to the underlying code is not possible, this is a serious restriction to this method. Future work should involve investigation of the possibilities of the Kernel-DEIM to alleviate problems with this scenario.

As remarked on at the end of Section 2.3, similar to the projection case we silently assumed that one global DEIM approximation is sufficient for nonlinear approximation of the considered functions. While the "local" DEIM approaches [51, 54, 130, 179] are certainly a way out of this, an inclusion into the a-posteriori error estimators considered in this work is the next step for applicability to more complex models.

Finally, if some suitable estimate of the *error direction* was available, the local logarithmic Lipschitz constant estimations could be further improved, see Sections 5.2 and 5.2.1. Recall that, in this process, the logarithmic Lipschitz constant of the local Jacobian (or at least its matrix-DEIM approximation) is used to estimate

$$\frac{\langle \boldsymbol{e}(t), \boldsymbol{J}\boldsymbol{e}(t)\rangle}{||\boldsymbol{e}(t)||^2} \leq \max\Big\{\sigma\Big(\frac{1}{2}(\boldsymbol{J}+\boldsymbol{J}^T)\Big)\Big\} = L[\boldsymbol{J}].$$

For simplicity, we omit the time-dependency, assume $\boldsymbol{J}$ as the Jacobian of $\boldsymbol{f}$ at $\boldsymbol{x}^r(t)$ and set $\boldsymbol{G} = \boldsymbol{I}_d$. Now, assume we have an eigen-decomposition $\boldsymbol{J} + \boldsymbol{J}^T = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T$ with $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d)$ and $\boldsymbol{\Sigma}$ containing the sorted eigenvalues $\lambda_1 \geq \ldots \geq \lambda_d$ on its diagonal. Then, in the above estimation we could write

$$\begin{aligned}
\langle \boldsymbol{e}, \boldsymbol{J}\boldsymbol{e}\rangle &= \frac{1}{2}\left\langle \boldsymbol{e}, (\boldsymbol{J}+\boldsymbol{J}^T)\boldsymbol{e}\right\rangle = \frac{1}{2}\left\langle \boldsymbol{e}, \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{e}\right\rangle \\
&= \frac{\lambda_1}{2}\left\langle \boldsymbol{e}, \boldsymbol{u}_1\boldsymbol{u}_1^T\boldsymbol{e}\right\rangle + \frac{1}{2}\left\langle \boldsymbol{e}, \sum_{i=2}^{d} \lambda_i \boldsymbol{u}_i\boldsymbol{u}_i^T\boldsymbol{e}\right\rangle \\
&= \frac{\lambda_1}{2}\left\langle \boldsymbol{e}, \boldsymbol{u}_1\right\rangle^2 + \frac{1}{2}\left\langle \boldsymbol{e}, \left(\boldsymbol{J}+\boldsymbol{J}^T - \lambda_1\boldsymbol{u}_1\boldsymbol{u}_1^T\right)\boldsymbol{e}\right\rangle \\
&= \frac{\lambda_1}{2}\left\langle \boldsymbol{e}, \boldsymbol{u}_1\right\rangle^2 + \frac{1}{2}\left\langle \boldsymbol{e}, \left(\boldsymbol{J}_1+\boldsymbol{J}_1^T\right)\boldsymbol{e}\right\rangle.
\end{aligned}$$

using $\boldsymbol{J}_1 = \boldsymbol{J} - \lambda_1\boldsymbol{u}_1\boldsymbol{u}_1^T$. This directly gives

$$\frac{\langle \boldsymbol{e}, \boldsymbol{J}\boldsymbol{e}\rangle}{||\boldsymbol{e}||^2} \leq \frac{\lambda_1}{2}\frac{\langle \boldsymbol{e}, \boldsymbol{u}_1\rangle^2}{||\boldsymbol{e}||^2} + L[\boldsymbol{J}_1] = \frac{\lambda_1}{2}\langle \vec{\boldsymbol{e}}, \boldsymbol{u}_1\rangle^2 + L[\boldsymbol{J}_1],$$

where $\vec{\boldsymbol{e}}$ indicates the normalized error vector $\boldsymbol{e}$. It is immediately clear that this extraction process could be driven further for the first m eigenvalues in the sense of

$$\frac{\langle \boldsymbol{e}, \boldsymbol{J}\boldsymbol{e}\rangle}{||\boldsymbol{e}||^2} \leq \sum_{i=1}^{m} \frac{\lambda_m}{2}\langle \vec{\boldsymbol{e}}, \boldsymbol{u}_m\rangle^2 + L[\boldsymbol{J}_m]$$

with the obvious definitions, but recall that the eigenvalue decomposition is required for the full system's Jacobians $\boldsymbol{J} \in \mathbb{R}^{d\times d}$. Thus, computing the two largest eigenvalues and appropriate subspaces will already be expensive enough in practice. Now, conceptionally speaking, this method could prove effective for situations where the largest (or first few) eigenvalues of the symmetric part of the Jacobian are considerably large compared to the remaining ones. Given an error direction, those eigenvalues are weighted using the part of $\vec{\boldsymbol{e}}$ in the direction of the eigenvectors, which is ≤ 1 and reproduces $L[\boldsymbol{J}]$ in the worst case. Moreover, using the partial similarity transformation scheme as introduced in Proposition 5.2.13 and Algorithm 10, we can efficiently decompose $\langle \vec{\boldsymbol{e}}, \boldsymbol{u}_1 \rangle^2$ in an offline/online fashion if we have an expression of $\vec{\boldsymbol{e}}$ like $\vec{\boldsymbol{e}} = \boldsymbol{M}\boldsymbol{g}$. Then, for $m \leq k$ and $\boldsymbol{Q}_k$ denoting the k-th order partial similarity transformation matrix, we can write

$$\sum_{i=1}^{m} \langle \vec{\boldsymbol{e}}, \boldsymbol{u}_m \rangle^2 = \sum_{i=1}^{m} (\boldsymbol{g}^T \boldsymbol{M}^T \boldsymbol{Q}_k \boldsymbol{q}_m)^2 = \sum_{i=1}^{m} (\boldsymbol{g}^T \tilde{\boldsymbol{Q}}_k \boldsymbol{q}_m)^2,$$

where $\tilde{\boldsymbol{Q}}_k := \boldsymbol{M}^T \boldsymbol{Q}_k$ and $\boldsymbol{q}_m$ are the m-largest eigenvalues of the $k \times k$ transformed MDEIM-Jacobian. Of course, it remains to be clarified to what extend a reasonably small k can cover more than one eigenvector subspace for all occurring Jacobians, and it is unclear if such a representation $\vec{\boldsymbol{e}} = \boldsymbol{M}\boldsymbol{g}$ can be devised. To that extent, we know that $\boldsymbol{e} \perp \mathcal{V}$, where we recall $\mathcal{V}$ is the projection subspace spanned by the columns of $\boldsymbol{V} \in \mathbb{R}^{d \times r}$. So, if $\mathcal{V}$ is e.g. given by a POD, which creates a hierarchical basis w.r.t. to representability, the next POD modes $\boldsymbol{v}_{r+1}, \ldots, \boldsymbol{v}_{r+k}$ are likely to capture the most part of the error, making $\boldsymbol{M} = [\boldsymbol{v}_{r+1} \ldots \boldsymbol{v}_{r+k}] \in \mathbb{R}^{d \times k}$ a possible candidate along with some "higher order" reduced state space estimate $\boldsymbol{g} \in \mathbb{R}^k$. Moreover, if we knew $\boldsymbol{u}_m \in \mathcal{V}$ for all m, this would give $\langle \vec{\boldsymbol{e}}, \boldsymbol{u}_m \rangle^2 = 0$ due to $\boldsymbol{e} \perp \mathcal{V}$. Thus, for example, including the space spanned by the eigenvectors of the largest eigenvalues will remove the contribution of $\langle \vec{\boldsymbol{e}}, \boldsymbol{u}_1 \rangle^2$ completely without even knowing $\vec{\boldsymbol{e}}$. Conclusively, while being far from complete, this outline stretches first ideas on how the error estimation procedure could be improved.

Software

All the experimental results presented in this works have been obtained using the MatLab framework *KerMor* [184], which has been developed and grown along with the pursued research.

KerMor makes extensive use of MatLab's object-oriented programming features and hence provides a highly extendable framework for implementation and model order reduction of dynamical systems. In it's current state, nonlinear affine parameter- and time-dependent systems with inputs and outputs as introduced in Section 1.3.2 are readily implementable. For solution of the ODEs, some directly implemented solvers as well as all MatLab built-in solvers can be employed and e.g. Jacobian information is generally provided and can be specified explicitly. The Petrov-Galerkin procedure is generically implemented and allows for subspace projection of all possible combinations of system components. Moreover, all the presented nonlinear approximation schemes (ϵ-SVR, VKOGA, DEIM) including some more are available as substitutions for the nonlinear components and seamlessly integrate into the subspace projection process. Suitable interfaces have to be implemented in order to use e.g. DEIM approximations and the involved error estimators. Furthermore, all the presented error estimators can be optionally attached to any fitting model and are automatically evaluated during reduced simulations if enabled.
The offline/online methodology is inherently implemented for all steps of the model reduction process, including parameter sampling, training data generation, subspace computation, approximation and error estimation. Regarding large matrices, memory issues are avoided by using (optional) file-

system based data structures, which yet allow to be handled as simple matrices.
KerMor is available upon request to the author. The documentation as well as downloads of release and current versions can be found at `http://www.morepas.org/software/kermor`.

Moreover, I personally would like to draw your attention to the small tool `mtoc++`, which has been developed in joint work with M. Drohmann (Sandia Labs). It provides a filter for MatLab source files to be read by doxygen (`http://www.doxygen.org`) in order to generate extensive documentation and class hierarchies from code and in-code comments. `mtoc++` is available on the MatLab FileExchange `http://www.mathworks.com/matlabcentral/fileexchange/33826` and documentation and downloads can also be found at the MoRePas Website `http://www.morepas.org/software/mtocpp`.

Acronyms & Symbols

Acronyms

DEIM	Discrete empirical interpolation method, page 23
DoF	Degrees of Freedom, page 16
FE	Finite element, page 117
IVD	Intervertebral Disk, page 119
MBS	Multi-body system, page 120
MDEIM	Matrix DEIM, page 175
MOR	Model order reduction, page 6
ODE	Ordinary differential equation, page 16
PDE	Partial differential equation, page 80
QP	Quadratic Program, page 31
RBF	Radial basis function, page 10
RB	Reduced basis, page 86
RKHS	Reproducing kernel Hilbert space, page 13
ROM	Reduced order model, page 85
s.p.d.	symmetric, positive definite (kernel), page 13
SMO	Sequential minimal optimization, page 31
SVM	Support vector machine, page 23

SVR	Support vector regression, page 23
TPWL	Trajectory piece-wise linear (MOR approach), page 97
VKOGA	Vectorial Kernel Orthogonal Greedy Algorithm, page 48

Symbols

$\boldsymbol{A}$	Dynamical system linear part matrix, page 17
$\boldsymbol{\alpha}^{\pm}$	SVR Lagrange multipliers or coefficients, page 29
B	Bell function upper bound, page 145
b	Bound for ϵ-SVR clipping operation, page 41
$\mathcal{B}$	Set of all Bell functions, page 146
$\boldsymbol{B}$	Dynamical system input mapping matrix, page 17
$\boldsymbol{C}$	Dynamical system output conversion matrix, page 17
$\boldsymbol{c}_i$	Vectorial kernel expansion coefficients, page 98
d	State space dimension $d \in \mathbb{N}$, page 7
$\mathcal{D}$	$\boldsymbol{f}$-domain for time- and parameter dependent case, page 19
Δ	Error estimator, page 139
$\delta_{\boldsymbol{x}}$	Point evaluation functional in $\mathcal{H}^*$, page 13
$d_i(t)$	Distance of reconstructed state space variable to expansion centers, $d_i(t) = \|\|\boldsymbol{x}^r(t) - \boldsymbol{x}_i\|\|_G$, page 152
$\boldsymbol{D}$	Dynamical system input to output conversion matrix, page 17
E_A	Approximation error $\boldsymbol{f} - \hat{\boldsymbol{f}}$, page 141
E_0	Initial error, page 142
$\boldsymbol{f}$	General nonlinear function, page 17
$\hat{\boldsymbol{f}}$	Approximant to $\boldsymbol{f}$ (either kernel-based or DEIM), page 98
Γ	Subdomain of $\mathbb{R}_+$ in the context of local Lipschitz constant estimations, page 151

γ	RBF dilation hyperparameter or kernel width, page 11
$G_{X,\boldsymbol{f}}$	Gain function for vectorial greedy algorithms, page 56
$\boldsymbol{G}$	Scalar product matrix $\boldsymbol{G} \in GL_d(\mathbb{R})$, page 7
$\mathcal{H}$	Reproducing kernel Hilbert space / Native space, page 15
$\mathcal{H}^q$	Vectorial RKHS, page 15
$\boldsymbol{I}_d$	d-dimensional identity matrix, page 8
$\boldsymbol{J}_P$	$\boldsymbol{f}$-Jacobian sparsity pattern, page 82
K	Kernel $K : \Omega \times \Omega \to \mathbb{R}$, page 8
$\boldsymbol{K}, \boldsymbol{K}_X$	Kernel matrix (for sets $X \subset \Omega$), page 10
$\boldsymbol{k}, \boldsymbol{k}_X$	Kernel vector (for sets $X \subset \Omega$), page 10
l	Dynamical system input space dimension $l \in \mathbb{N}$, page 17
$\mathcal{L}$	Lagrangian of primary objective in ϵ-SVR formulation, page 29
N	Newton basis $N_1, \ldots$ for linear subspaces of RKHS $\mathcal{H}$, page 73
Ω	State space domain $\Omega \subset \mathbb{R}^d$, page 7
p	Parameter domain dimension $p \in \mathbb{N}$, page 18
$\mathcal{P}$	Parameter domain $\mathcal{P} \subseteq \mathbb{R}^p$, page 18
ϕ	Bell function, page 145
$\phi_{\boldsymbol{x}}$	$\mathcal{H}^X$-orthonormal remainder of $K(\boldsymbol{x}, \cdot)$, page 51
$\tilde{\phi}_{\boldsymbol{x}}$	$\mathcal{H}^X$-orthogonal remainder of $K(\boldsymbol{x}, \cdot)$, page 51
q	Dimension $q \in \mathbb{N}$ of vectorial RKHS $\mathcal{H}^q$, page 15
r	Reduced state space dimension $r \ll d$, page 86
S	Secant gradient function for bell functions $\phi \in \mathcal{B}$, page 147
σ	Spectrum operator for matrix eigenvalues and -vectors, page 8
T	Dynamical system simulation end time $T > 0$, page 16

t	Time variable for dynamical systems, page 16
θ	Scalar coefficient functions $[0,T]\times\mathcal{P}\rightarrow\mathbb{R}$, page 18
$\Theta,\Theta(t)$	A-priori error bound, page 145
Υ	"Vectorize"-operation, page 191
$\boldsymbol{u}(t)$	Dynamical system input function $\boldsymbol{u}:[0,T]\rightarrow\in\mathbb{R}^l$, page 17
W	Dual objective function of ϵ-SVR, page 29
$\boldsymbol{w}(t)$	Dynamical system output $\boldsymbol{w}:[0,T]\rightarrow\in\mathbb{R}^k$, page 17
X	Training inputs $X=\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n\}\subseteq\Omega$, page 21
Ξ	Training parameter set $\Xi\subset\mathcal{P}\subseteq\mathbb{R}^p$, page 96
ξ	Spatial variable for PDE applications, page 112
ξ_i^+,ξ_i^-	SVR slack variables, page 28
$\boldsymbol{X}$	Training sample matrix $\boldsymbol{X}=[\boldsymbol{x}_1\ldots]$, page 89
$\boldsymbol{x}(t)$	Full state space vector, page 16
$\boldsymbol{x}_0$	Dynamical system initial state, page 16
Y	Training outputs $\boldsymbol{y}_1=\boldsymbol{f}(\boldsymbol{x}_1)\ldots$, page 21
$\boldsymbol{y}$	Nonlinear function evaluations $\boldsymbol{y}=\boldsymbol{f}(\boldsymbol{x})$, page 21

Bibliography

[1] H.S. Abdel-Khalik. On Nonlinear Reduced Order Modeling. In *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011)*, Rio de Janeiro, RJ, Brazil, May 2011. American Nuclear Society (ANS).

[2] A. Abdulle, E. Weinan, B. Engquist, and E. Vanden-Eijnden. The heterogeneous multiscale method. *Acta Numerica*, 21:1–87, 5 2012.

[3] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions*, volume 10 of *Applied Mathematics Series*. National Bureau of Standards, 1972.

[4] D. Amsallem, J. Cortial, K. Carlberg, and C. Farhat. A method for interpolating on manifolds structural dynamics reduced-order models. *Internat. J. Numer. Methods Engrg.*, 80(9):1241–1258, 2009.

[5] D. Amsallem and C. Farhat. An online method for interpolating linear parametric reduced-order models. *SIAM J. Sci. Comput.*, 33, No.5:2169–2198, 2011.

[6] A.C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. SIAM Publications, Philadelphia, PA, 2005.

[7] A.C. Antoulas. An overview of approximation methods for large-scale dynamical systems. *Annu. Rev. Contr.*, 29:181–190, 2005.

[8] N. Aronszajn. Theory of reproducing kernels. *Trans. Am. Math. Soc.*, 68(3):337–404, 1950.

[9] Z. Bai and D. Skoogh. A projection method for model reduction of bilinear dynamical systems. *Linear Algebra Appl.*, 415(2-3):406 – 425, 2006. Special Issue on Order Reduction of Large-Scale Systems.

[10] M. Barrault, Y. Maday, N.C. Nguyen, and A.T. Patera. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *C. R. Math. Acad. Sci. Paris Series I*, 339:667–672, 2004.

[11] P. Benner. Numerical linear algebra for model reduction in control and simulation. *GAMM-Mitt.*, 29(2):275–296, 2006.

[12] P. Benner and T. Breiten. Krylov-subspace based model reduction of nonlinear circuit models using bilinear and quadratic-linear approximations. In *Progress in Industrial Mathematics at ECMI 2010*, Mathematics in Industry, pages 153–159. Springer Berlin Heidelberg, 2012.

[13] C. Berg, J. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups*. Springer, 1984.

[14] C. M. Bishop. *Pattern Recognition and Machine Learning*, volume 2. Springer, August 2006. ISBN 978-0-387-31073-2.

[15] B. Bond and L. Daniel. Parameterized model order reduction of nonlinear dynamical systems. In *Proc. of ICCAD 2005*, pages 487 – 494, nov. 2005.

[16] G. Box and N. Draper. *Response Surfaces, Mixtures, and Ridge Analyses*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2 edition, 2007.

[17] M. Bozzini, M. Rossini, and R. Schaback. Generalized whittle-matern and polyharmonic kernels. *Adv. Comput. Math.*, pages 1–13, August 2012.

[18] O. Brüls, P. Duysinx, and J.-C. Golinval. The global modal parameterization for nonlinear model-order reduction in flexible multibody dynamics. *Internat. J. Numer. Methods Engrg.*, 69(5):948–977, 2007.

[19] M. D. Buhmann. *Radial Basis Functions - Theory and Implementations*, volume 12 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, August 2003.

[20] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numer.*, 13:147–269, 4 2004.

[21] K. Carlberg, R. Tuminaro, and P. Boggsz. Effcient structure-preserving model reduction for nonlinear mechanical systems with application to structural dynamics. Preprint, Sandia National Laboratories, Livermore, CA 94551, USA, 2012.

[22] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Mach. Learn.*, 46(1-3):131–159, March 2002.

[23] S. Chaturantabut and D. Sorensen. Nonlinear Model Reduction via Discrete Empirical Interpolation. *SIAM J. Sci. Comput.*, 32(5):2737–2764, 2010.

[24] S. Chaturantabut and D. Sorensen. A State Space Error Estimate for POD-DEIM Nonlinear Model Reduction. *SIAM J. Numer. Anal.*, 50(1):46–63, 2012.

[25] S. Chen, C.F.N. Cowan, and P.M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Trans. Neural Networks*, 2(2):302 –309, mar 1991.

[26] S. Chen, P.M. Grant, and C.F.N. Cowan. Orthogonal least-squares algorithm for training multioutput radial basis function networks. *IEE Proc. F*, 139(6):378 –384, dec 1992.

[27] S. Chen and J. Wigger. Fast orthogonal least squares algorithm for efficient subset model selection. *IEEE Trans. Signal Processing*, 43(7):1713 –1715, jul 1995.

[28] Y. Chen and J. White. A quadratic method for nonlinear model order reduction. In *Macromodeling*, pages 477–480, 2000.

[29] H. Cohen. *Numerical Approximation Methods*, volume 1. Springer-Verlag New York Inc., 2011.

[30] R. R. Coifman and S. Lafon. Diffusion maps. *Appl. Comput. Harmon. Anal.*, 21(1):5 – 30, 2006.

[31] M. Condon and R. Ivanov. Empirical balanced truncation of nonlinear systems. *J. Nonlinear Sci.*, 14:405–414, 2004. 10.1007/s00332-004-0617-5.

[32] N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines.* Cambridge University Press, 2000. ISBN: 0 521 78019 5.

[33] G. Dahlquist. *Stability and error bounds in the numerical integration of ordinary differential equations.* Royal Institute of Technology, 130, Stockholm, Sweden, 1959.

[34] M. Daub, S. Waldherr, F. Allgöwer, P. Scheurich, and G. Schneider. Death wins against life in a spatially extended apoptosis model. *Biosystems*, 108(1-3):45–51, April 2012.

[35] G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Constr. Approx.*, 13:57–98, 1997.

[36] S. De Marchi and R. Schaback. Stability of kernel-based interpolation. *Advances in Computational Mathematics*, 32(2):155–161, 2010.

[37] S. De Marchi, R. Schaback, and H. Wendland. Near-optimal data-independent point locations for radial basis function interpolation. *Adv. Comput. Math.*, 23:317–330, 2005.

[38] M. de Zee, L. Hansen, T. B. Andersen, C. Wong, J. Rasmussen, and E. B. Simonsen. On the development of a detailed rigid-body spine model. In M. Doblare, M. Cerrolaza, and H. Rodrigues, editors, *Proceedings of International Congress on Computational Bioengineering*, Zaragosa, 2003.

[39] M. de Zee, L. Hansen, C. Wong, J. Rasmussen, and E. B. Simonsen. A generic detailed rigid-body lumbar spine model. *J. Biomech.*, 40:1219–1227, 2007.

[40] R. DeVore and V. Temlyakov. Some remarks on greedy algorithms. *Adv. Comput. Math.*, 5:173–187, 1996.

[41] R. A. DeVore. Nonlinear approximation. *Acta Numer.*, 7:51–150, 1998.

[42] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition (Stochastic Modelling and Applied Probability).* Springer, corrected edition, April 1996.

[43] D. A. DiCarlo. Experimental measurements of saturation overshoot on infiltration. *Water Resour. Res.*, 40(4), Apr 2004.

[44] M. Dihlmann, M. Drohmann, and B. Haasdonk. Model reduction of parametrized evolution problems using the reduced basis method with adaptive time-partitioning. In *Proc. of ADMOS 2011*, 2011.

[45] N. Dong and J. Roychowdhury. Piecewise polynomial nonlinear model reduction. In *Proc. of DAC 2003*, pages 484–489, June 2003.

[46] D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. Natl. Acad. Sci. USA*, 100(10):5591–5596, 2003.

[47] M. Drohmann, B. Haasdonk, and M. Ohlberger. Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation. *SIAM J. Sci. Comput.*, 34(2):A937–A969, 2012.

[48] W. E and B. Engquist. The heterognous multiscale methods. *Commun. Math. Sci.*, 1(1):87–132, 2003.

[49] W. E, B. Engquist, X. Li, W. Ren, and E. Vanden-Eijnden. The heterogeneous multiscale method: A review. Technical report, Princeton University, 2004.

[50] R. Eberlein, G. A. Holzapfel, and M. Fröhlich. Multi-segment FEA of the human lumbar spine including the heterogeneity of the anulus fibrosus. *Comput. Mech.*, 34:147–165, 2004.

[51] J. L. Eftang. *Reduced basis methods for parametrized partial differential equations.* Dissertation, Norwegian University of Science and Technology, Trondheim, May 2011.

[52] J. L. Eftang, M. A. Grepl, and A. T. Patera. A posteriori error bounds for the empirical interpolation method. *C. R. Math.*, 348(9-10):575 – 579, 2010.

[53] J. L. Eftang, D. J. Knezevic, and A. T. Patera. An hp certified reduced basis method for parametrized parabolic partial differential equations. *Math. Comp. Model Dyn.*, 17(4):395 – 422, 2011.

[54] J. L. Eftang and B. Stamm. Parameter multi-domain ”hp” empirical interpolation. *Internat. J. Numer. Methods Engrg.*, 90(4):412–428, 2012.

[55] V. Esat and M. Acar. Viscoelastic finite element analysis of the cervical intervertebral discs in conjunction with a multi-body dynamic model of the human head and neck. *Proc. Inst. Mech. Eng. H J. Eng. Med.*, 223:249–262, 2009.

[56] V. Esat, D. W. Lopik, and M. Acar. Combined multi-body dynamic and fe models of human head and neck. In M. D. Gilchrist, editor, *IUTAM Symposium on Impact Biomechanics: From Fundamental Insights to Applications*, pages 91–100. Springer-Verlag, Netherlands, 2005.

[57] G. Fasshauer and M. McCourt. Stable evaluation of Gaussian Radial Basis Function Interpolants. *SIAM J. Sci. Comput.*, 34(2):A737–A762, 2012.

[58] G. E. Fasshauer. Positive definite kernels: Past, present and future. In M. Buhmann, S. De Marchi, and Plonka-Hoch, editors, *Kernel Functions and Meshless Methods*, volume 4 of *Dolomites Research Notes on Approximation*, pages 21–63, 2011. Special Issue.

[59] M. Fauvel. Kernel matrix approximation for learning the kernel hyperparameters. In *Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International*, pages 5418–5421, 2012.

[60] B. Fornberg, E. Larsson, and N. Flyer. Stable computations with gaussian radial basis functions. *SIAM J. Sci. Comput.*, 33(2):869–892, April 2011.

[61] B. Fornberg and C. Piret. A stable algorithm for flat radial basis functions on a sphere. *SIAM J. Sci. Comput.*, 30(1):60–80, Oct 2007.

[62] M. G. Genton. Classes of kernels for machine learning: a statistics perspective. *J. Mach. Learn. Res.*, 2:299–312, Mar 2002.

[63] E. Georgoulis, J. Levesley, and F. Subhan. Multilevel sparse kernel-based interpolation. *SIAM J. Sci. Comput.*, 35(2):A815–A831, 2013.

[64] M. Günther and H. Ruder. Synthesis of two-dimensional human walking: a test of the λ-model. *Biol. Cybernet.*, 8:89–106, 2003.

[65] M. Grepl. Certified reduced basis methods for nonaffine linear time-varying partial differential equations. *Math. Model. Meth. Appl. Sci.*, 22, 2012.

[66] M. Grepl, Y. Maday, N. Nguyen, and A.T. Patera. Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *Math. Model. Numer. Anal.*, 41(3):575–605, May 2007.

[67] M. A. Grepl. *Reduced-Basis Approximation and A Posteriori Error Estimation for Parabolic Partial Differential Equations.* Dissertation, Massachusetts Institute of Technology, June 2005.

[68] V. Guigue, A. Rakotomamonjy, and S. Canu. Kernel basis pursuit. In *Machine Learning: ECML 2005*, volume 3720 of *Lecture Notes in Computer Science*, pages 146–157. Springer Berlin / Heidelberg, 2005.

[69] B. Haasdonk. Convergence rates of the POD-Greedy method. *Math. Model. Numer. Anal.*, 47:859–873, 2013.

[70] B. Haasdonk, M. Dihlmann, and M. Ohlberger. A training set and multiple basis generation approach for parametrized model reduction based on adaptive grids in parameter space. *Math. Comp. Model Dyn.*, 17:423–442, 2012.

[71] B. Haasdonk and M. Ohlberger. Reduced basis method for explicit finite volume approximations of nonlinear conservation laws. In *Proc. 12th International Conference on Hyperbolic Problems: Theory, Numerics, Application*, 2008.

[72] B. Haasdonk and M. Ohlberger. Reduced basis method for finite volume approximations of parametrized linear evolution equations. *Math. Model. Numer. Anal.*, 42(2):277–302, 2008.

[73] B. Haasdonk and M. Ohlberger. Efficient reduced models for parametrized dynamical systems by offline/online decomposition. In *Proc. MATHMOD 2009, 6th Vienna International Conference on Mathematical Modelling*, 2009.

[74] B. Haasdonk and M. Ohlberger. Efficient reduced models and a posteriori error estimation for parametrized dynamical systems by offline/online decomposition. *Math. Comp. Model Dyn.*, 17(2):145 – 161, 2011.

[75] B. Haasdonk, M. Ohlberger, and G. Rozza. A reduced basis method for evolution schemes with parameter-dependent explicit operators. *Electron. Trans. Numer. Anal.*, 32:145–161, 2008.

[76] J.K. Hale. *Ordinary differential equations*, volume XXI of *Pure and Applied Mathematics.* Wiley-Interscience, 1969.

[77] Joao P. Hespanha. *Linear Systems Theory.* Princeton University Press, Princeton, New Jersey, Sept 2009.

[78] D. Hilbert. Grundzüge einer allgemeinen Theorie der linearen Integralgleichungen. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, pages 595–618, 1910.

[79] M. Hinze and S. Volkwein. Proper orthogonal decomposition surrogate models for nonlinear dynamical systems: Error estimates and suboptimal control in dimension reduction of large-scale systems. In *Dimension Reduction of Large-Scale Systems, Lecture Notes in Computational and Applied Mathematics*, volume 45, pages 261–306. Springer Berlin Heidelberg, 2005.

[80] C. Homescu, L. R. Petzold, and R. Serban. Error estimation for reduced-order models of dynamical systems. *SIAM J. Numer. Anal.*, 43:2005, 2005.

[81] H. Hotelling. Analysis of a complex of statistical variables into principal components. *J. Educ. Psychol.*, 24(6):417–441, Sep. 1933.

[82] P. Hunter, P. V. Coveney, B. de Bono, V. Diaz, J. Fenner, A. F. Frangi, P. Harris, R. Hose, P. Kohl, P Lawford, K. McCormack, M. Mendes, S. Omholt, A. Quarteroni, J. Skar, J. Tegner, S. Randall Thomas, I. Tollis, I. Tsamardinos, J. H. G. M. van Beek, and M. Viceconti. A vision and strategy for the virtual physiological human in 2010 and beyond. *Phil. Trans. R. Soc. A*, 368:2595–2614, 2010.

[83] D.B.P. Huynh, G. Rozza, S. Sen, and A.T. Patera. A successive constraint linear optimization method for lower bounds of parametric coercivity and inf-sup stability constants. *C. R. Math. Acad. Sci. Paris Series I*, 345:473–478, 2007.

[84] M. Iwamoto, Y. Nakahira, A. Tamura, H. Kimpara, I. Watanabe, and K. Miki. Development of advanced human models in THUMS. In *6th European LS-DYNA Users' Conference*, pages 47–56. 2007.

[85] T. Jebara. Multi-task feature and kernel selection for svms. In *Proc. of ICML 2004*, 2004.

[86] I. T. Jolliffe. *Principal Component Analysis.* Springer-Verlag, 2002.

[87] N. Jung, A.T. Patera, B. Haasdonk, and B. Lohmann. Model order reduction and error estimation with an application to the parameter-dependent eddy current equation. *Math. Comp. Model. Dyn.*, 2010. , in print, online: 09 June 2011.

[88] N. Karajan. An extended biphasic description of the inhomogeneous and anisotropic intervertebral disc. Dissertation, Bericht Nr. II-19 aus dem Institut für Mechanik (Bauwesen), Universität Stuttgart, 2009.

[89] N. Karajan. Multiphasic intervertebral disc mechanics: Theory and application. *Arch. Comput. Method Eng.*, 19:261–339, 2012.

[90] N. Karajan, W. Ehlers, S. Oladyshkin, and D. Otto. Application of the polynomial chaos expansion to approximate the homogenised behaviour of the intervertebral disc. *Biomech. Model. Mechanobiol.*, Submitted, 2013.

[91] N. Karajan, O. Röhrle, W. Ehlers, and S. Schmitt. Linking continuous and discrete intervertebral disc models through homogenisation. *Biomech. Model. Mechanobiol.*, 12:453–466, 2013.

[92] K. Karhunen. Über lineare Methoden in der Wahrscheinlichkeitsrechnung. *Ann. Acad. Sri. Fennicae, ser. A l . Math. Phys.*, 37, 1946.

[93] F. Kissling. *Analysis and Numerics for Nonclassical Wave Fronts in Porous Media.* Dissertation, Universität Stuttgart, 2013. Verlag Dr. Hut, München, ISBN 978-3-8439-0996-9.

[94] F. Kissling and C. Rhode. The Computation of Nonclassical Shock Waves in Porous Media with a Heterogeneous Multiscale Method: The Multidimensional Case. Preprint, Stuttgart Research Centre for Simulation Technology, Nov 2012. Submitted to SIAM Multiscale Modeling and Simulation.

[95] F. Kissling and C. Rohde. The computation of nonclassical shock waves with a heterogeneous multiscale method. *Netw. Heterog. Media*, 5(3):661–674, Sep 2010. Workshop on New Trends in Model Coupling, Theory, Numerics and Applications, Paris, FRANCE, SEP 02-04, 2009.

[96] S. Konyagin and V. Temlyakov. Greedy approximation with regard to bases and general minimal systems. *Serdica Mathematical Journal*, 28(4):305–328, 2002. ISSN: 1310-6600.

[97] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM J. Numer. Anal.*, 40(2):pp. 492–515, 2003.

[98] K. Kunisch and V. Volkwein. Proper orthogonal decomposition for optimality systems. *Math. Model. Numer. Anal.*, 42:1–23, 2008.

[99] H. Laanaya, F. Abdallah, H. Snoussi, and C. Richard. Learning general Gaussian kernel hyperparameters of SVMs using optimization on symmetric positive-definite matrices manifold. *Pattern Recogn. Lett.*, 32(13):1511–1515, Oct 2011.

[100] S. Lall, J.E. Marsden, and S. Glavaski. A subspace approach to balanced truncation for model reduction of nonlinear control systems. *Internat. J. Robust Nonlinear Control*, 12(5):519–535, 2002.

[101] J. A. Lee and M. Verleysen. Nonlinear dimensionality reduction. In M.I. Jordan, R. Nowak, and B. Schölkopf, editors, *Information Science and Statistics.* Springer, 2007.

[102] P. G. LeFloch. Fundamental concepts and examples. In *Hyperbolic Systems of Conservation Laws*, Lectures in Mathematics. ETH Zürich, pages 1–26. Birkhäuser Basel, 2002.

[103] D. Leviatan and V. Temlyakov. Simultaneous approximation by greedy algorithms. *Adv. Comput. Math.*, 25:73–90, 2006.

[104] D. Leviatan and V.N. Temlyakov. Simultaneous greedy approximation in Banach spaces. *J. Complexity*, 21(3):275 – 293, 2005.

[105] M. M. Loeve. *Probability Theory.* Van Nostrand, Princeton, NJ, 1955.

[106] B. Lohmann and R. Eid. Efficient order reduction of parametric and nonlinear models by superposition of locally reduced models. In *Methoden und Anwendungen der Regelungstechnik. Erlangen-Münchener Workshops 2007 und 2008*, pages 27–36. Shaker Verlag, Aachen, 2009.

[107] A. Lutoborski and V. N. Temlyakov. Vector greedy algorithms. *J. Complexity*, 19(4):458 – 473, 2003.

[108] H. V. Ly and H. T. Tran. Modeling and control of physical processes using proper orthogonal decomposition. *Math. Comput. Model.*, 33(1-3):223–236, Jan 2001.

[109] L. Ma and Z. Wu. Kernel based approximation in Sobolev spaces with radial basis functions. *Appl. Math. Comput.*, 215(6):2229–2237, 2009.

[110] X. Ma and G. Karniadakis. A low-dimensional model for simulating three-dimensional cylinder flow. *J. Fluid Mech.*, 458:181–190, May 2002.

[111] Y. Maday, N.C. Nguyen, A.T. Patera, and G.S.H. Pau. A general multipurpose interpolation procedure: the magic points. *Commun. Pure Appl. Anal.*, 8(1):383–404, Jan 2009.

[112] Y. Maday and E. Ronquist. A reduced-basis element method. *C. R. Math.*, 335(2):195 – 200, 2002.

[113] Y. Maday and E. Ronquist. The reduced basis element method: Application to a thermal fin problem. *SIAM J. Sci. Comput.*, 26(1):240–258, 2006.

[114] S.G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Processing*, 41(12):3397 –3415, dec 1993.

[115] B. Matern. *Spatial Variation.* Springer, New York, 1960.

[116] M. Meyer and H.G. Matthies. Efficient model reduction in non-linear dynamics using the Karhunen-Loève expansion and dual-weighted-residual methods. *Comput. Mech.*, 31:179–191, 2003.

[117] S. Müller. *Complexity and Stability of Kernel-based Reconstructions.* Dissertation, Georg-August-Universität Göttingen, Institut für Numerische und Angewandte Mathematik, Lotzestrasse 16-18, D-37083 Göttingen, Jan 2009. Göttinger Online Klassifikation: EDDF 050.

[118] B.C. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Trans. Automat. Control*, AC–26(1):17–32, 1981.

[119] M. Mouattamid. *Theory of Power Kernels.* Dissertation, Georg-August-Universität Göttingen, 2005.

[120] M. Mouattamid and R. Schaback. Recursive kernels. *Anal. Theory Appl.*, 25(4):301–316, 2009.

[121] R. H. Myers, D. C. Montgomery, and C. M. Anderson-Cook. *Response Surface Methodology - Process and Product Optimization Using Designed Experiments.* Wiley Series in Probability and Statistics. Wiley, 3 edition, 2009.

[122] N.C. Nguyen, K. Veroy, and A.T. Patera. Certified real-time solution of parametrized partial differential equations. In S. Yip, editor, *Handbook of Materials Modeling*, pages 1523–1558. Springer, 2005.

[123] S. Oladyshkin and W. Nowak. Data-driven uncertainty quantification using the arbitrary polynomial chaos expansion. *Reliab. Eng. Syst. Safety*, 106(0):179 – 190, 2012.

[124] M.A. Oliver and R. Webster. Kriging: a method of interpolation for geographical information systems. *Int. J. Geogr. Inf. Sci.*, 4(3):313–332, 1990.

[125] F. Oshita, K. Omori, Y. Nakahira, and K. Miki. Development of a finite-element model of the human body. In *7th International LS-DYNA Users Conference*, pages 37–48. 2002.

[126] H. Panzer, J. Mohring, R. Eid, and B. Lohmann. Parametric model order reduction by matrix interpolation. *at - Automatisierungstechnik*, 58(8):475–484, 2010.

[127] A.T. Patera and G. Rozza. *Reduced Basis Approximation and a Posteriori Error Estimation for Parametrized Partial Differential Equations.* MIT, 2007. Version 1.0, Copyright MIT 2006-2007, to appear in (tentative rubric) MIT Pappalardo Graduate Monographs in Mechanical Engineering.

[128] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 40 – 44, Nov 1993.

[129] M. Pazouki and R. Schaback. Bases for kernel-based spaces. *J. Comp. Appl. Math.*, 236(4):575 – 588, 2011. International Workshop on Multivariate Approximation and Interpolation with Applications (MAIA 2010).

[130] B. Peherstorfer, D. Butnaru, K. Willcox, and H. Bungartz. Localized Discrete Empirical Interpolation Method. Technical Report TR-13-1, Aerospace Computational Design Lab, Dept. of Aeronautics & Astronautics, MIT, June 2013.

[131] J. Phillips, J. Afonso, A. Oliveira, and L.M. Silveira. Analog macromodeling using kernel methods. In *Proc. of ICCAD 2003*, pages 446–453, November 2003.

[132] J.R. Phillips. Projection frameworks for model reduction of weakly nonlinear systems. In *Proc. of DAC 2000*, pages 184–189, 2000.

[133] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in kernel methods*, pages 185–208, Cambridge, MA, USA, 1999. MIT Press.

[134] T. Reis and M. Heinkenschloss. Model reduction with a-priori error bounds for a class of nonlinear electrical circuits. In *Proc. of CDC/CCC 2009*, pages 5376–5383, Dec. 2009.

[135] M. Rewienski and J. White. A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. *IEEE Trans. Computer-Aided Design*, 22(2):155 – 170, feb. 2003.

[136] M.J. Rewienski. *A Trajectory Piecewise-Linear Approach to Model Order Reduction of Nonlinear Dynamical Systems.* Dissertation, Massachusetts Institute of Technology, Cambridge, USA, 2003.

[137] O. Röhrle, J. B. Davidson, and A.J. Pullan. A physiologically based, multi-scale model of skeletal muscle structure and function. *Front. Physiol.*, 3:358, 2012.

[138] O. Röhrle, M. Sprenger, E. Ramasamy, and T. Heidlauf. Multi-scale skeletal muscle modelling: From cellular level to a multi-segment skeletal muscle model of the upper limb. In E. Kuhl and G. Holzapfel, editors, *Computer Models in Biomechanics: From Nano to Macro*, pages 103–116, Springer Netherlands, 2012.

[139] C. Rieger, R. Schaback, and B. Zwicknagl. Sampling and stability. In *Mathematical Methods for Curves and Surfaces*, volume 5862 of *Lecture Notes in Computer Science*, pages 347–369. Springer Berlin Heidelberg, 2010.

[140] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[141] C.W. Rowley. Model reduction for fluids, using balanced proper orthogonal decomposition. *Int. J. Bifurcat. Chaos*, 15(3):997–1013, 2005.

[142] G. Rozza. *Shape design by optimal flow control and reduced basis techniques: Applications to bypass configurations in haemodynamics.* Dissertation, École Polytechnique Fédérale de Lausanne, November 2005.

[143] H. Sandberg and A. Rantzer. Balanced truncation of linear time-varying systems. *IEEE Trans. Automat. Contr.*, 49(2):217–229, Feb 2004.

[144] L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *J. Mach. Learn. Res.*, 4:119–155, December 2003.

[145] R. Schaback. Reconstruction of multivariate functions from scattered data. Manuscript, University of Göttingen, 1997. `http://num.math.uni-goettingen.de/schaback/teaching/rbfbook.pdf`.

[146] R. Schaback. Limit problems for interpolation by analytic radial basis functions. *J. Comp. Appl. Math.*, 212(2):127 – 149, 2008.

[147] R. Schaback and S. De Marchi. Nonstandard kernels and their applications. In L. P. Bos, S. De Marchi, M. R. Zaglia, and M. Vianello, editors, *Dolomites Research Notes on Approximation*, volume 2, 2009.

[148] R. Schaback and H. Wendland. Adaptive greedy techniques for approximate solution of large RBF systems. *Numer. Algorithms*, 24:239–254, 2000.

[149] R. Schaback and H. Wendland. Kernel techniques: From machine learning to meshless methods. *Acta Numer.*, 15:543–639, May 2006.

[150] R. Schaback and J. Werner. Linearly constrained reconstruction of functions by kernels with applications to machine learning. *Adv. Comput. Math.*, 25:237–258, 2006.

[151] J.M.A. Scherpen. Balancing for nonlinear systems. *Syst. Contr. Lett.*, 21(2):143–153, 1993.

[152] B. Schölkopf and A. J. Smola. *Learning with Kernels.* Adaptive Computation and Machine Learning. The MIT Press, 2002.

[153] E. Schmidt. Zur Theorie der linearen und nichtlinearen Integralgleichungen. *Math. Ann.*, 63:433–476, 1907.

[154] S. Schmitt. *Über die Anwendung und Modifikation des Hill'schen Muskelmodells in der Biomechanik.* Dissertation, Eberhard Karls Universität Tübingen, 2006.

[155] B. Schölkopf, A. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5):1299–1319, July 1998.

[156] G. Söderlind. The logarithmic norm. history and modern theory. *BIT Numerical Mathematics*, 46:631–652, 2006. 10.1007/s10543-006-0069-9.

[157] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004. ISBN: 0521813972.

[158] A. Shirazi-Adl. Analysis of large compression loads on lumbar spine in flexion and torsion using a novel wrapping element. *J. Biomech.*, 39:267–275, 2006.

[159] K. Slavakis, P. Bouboulis, and S. Theodoridis. Adaptive multiregression in Reproducing Kernel Hilbert Spaces: The Multiaccess MIMO Channel Case. *IEEE Trans. Neural Netw. Learn. Syst.*, 23(2):260 –276, feb. 2012.

[160] I. Steinwart and A. Christman. *Support Vector Machines.* Science + Business Media. Springer, 2008.

[161] I. Steinwart, D. Hush, and C. Scovel. An oracle inequality for clipped regularized risk minimizers. In J. Platt B. Schölkopf and T. Hoffman, editors, *Advances in Neural Information Processing Systems 20*, pages pp. 1321–1328, Cambridge, MA, 2007. MIT Press.

[162] I. Steinwart, D. Hush, and C. Scovel. Training SVMs Without Offset. *J. Mach. Learn. Res.*, 12:141–202, February 2011.

[163] I. Steinwart and C. Scovel. Fast rates for support vector machines. In P. Auer and R. Meir, editors, *Learning Theory*, volume 3559 of *Lecture Notes in Computer Science*, pages 279–294. Springer Berlin Heidelberg, 2005.

[164] I. Steinwart and C. Scovel. Fast rates for support vector machines using gaussian kernels. *Ann. Statist.*, 25(2):575–607, 2007.

[165] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*, volume 12 of *Texts in Applied Mathematics.* Springer, 3 edition, 2002.

[166] P. Tabuada, A. D. Ames, A. Julius, and G. J. Pappas. Approximate reduction of dynamic systems. *Syst. Contr. Lett.*, 57(7):538–545, 2008.

[167] V. N. Temlyakov. The best m-term approximation and greedy algorithms. *Adv. Comput. Math.*, 8:249–265, 1998.

[168] V. N. Temlyakov. Weak greedy algorithms. *Adv. Comput. Math.*, 12:213–227, 2000.

[169] V. N. Temlyakov. Greedy approximation. *Acta Numer.*, 17:235–409, 2008.

[170] J. B. Tenenbaum, V. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[171] T. Tonn. *Reduced-Basis Method (RBM) for Non-Affine Elliptic Parametrized PDEs.* Dissertation, Universität Ulm, 2011.

[172] J. A. Tropp, A. C. Gilbert, and M. J. Strauss. Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit. *Signal Processing*, 86(3):572 – 588, 2006.

[173] K. Veroy and A.T. Patera. Certified real-time solution of the parametrized steady incompressible Navier-Stokes equations: Rigorous reduced-basis a posteriori error bounds. *Int. J. Numer. Meth. Fluids*, 47:773–788, 2005.

[174] M. Viceconti, G. Clapworthy, and S. Van Sint Jan. The virtual physiological human – A European initiative for in silico human modelling. *J. Physiol. Sci.*, 58:441–446, 2008.

[175] P. Vincent and Y. Bengio. Kernel matching pursuit. *Mach. Learn.*, 48:165–187, 2002.

[176] S. Volkwein. Proper Orthogonal Decomposition: Theory and Reduced-Order Modelling. Lecture notes, Universität Konstanz, 2012. `http://www.math.uni-konstanz.de/numerik/personen/volkwein/teaching/POD-Book.pdf`.

[177] G. Wahba. *Spline Models for Observational Data.* CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM Society for Industrial and Applied Mathematics, 1990.

[178] H. Wendland. *Scattered data approximation*, volume 17 of *Cambridge Monographs on Applied and Computational Mathematics.* Cambridge University Press, 2005.

[179] B. Wieland. *Reduced Basis Methods for Partial Differential Equations with Stochastic Influences.* Dissertation, Universität Ulm, April 2013.

[180] N. Wiener. The homogeneous chaos. *Amer. J. Math.*, 60(4):897–936, Oct 1938.

[181] D. Wirtz and B. Haasdonk. A-posteriori error estimation for parameterized kernel-based systems. In *Proc. MATHMOD 2012 - 7th Vienna International Conference on Mathematical Modelling*, 2012.

[182] D. Wirtz and B. Haasdonk. Efficient a-posteriori error estimation for nonlinear kernel-based reduced systems. *Syst. Contr. Lett.*, 61(1):203 – 211, 2012.

[183] D. Wirtz and B. Haasdonk. An improved vectorial kernel orthogonal greedy algorithm. *Dolomites Research Notes on Approximation*, 6:83–100, 2013.

[184] D. Wirtz, B. Haasdonk, and T. Strecker. KerMor: Kernel-based Model Order Reduction of Nonlinear Dynamical Systems. Software, University of Stuttgart, IANS, 2013.

[185] D. Wirtz, N. Karajan, and B. Haasdonk. Model order reduction of multiscale models using kernel methods. Preprint, University of Stuttgart, SRC SimTech, June 2013. Submitted.

[186] D. Wirtz, D.C. Sorensen, and B. Haasdonk. A-posteriori error estimation for DEIM reduced nonlinear dynamical systems. *SIAM J. Sci. Comput.*, Oct 2013. In print.

[187] Q. Wu, Y. Ying, and D.-X. Zhou. Multi-kernel regularized classifiers. *J. Complexity*, 23(1):108 – 134, 2007.

[188] Q. Wu and D.-X. Zhou. Analysis of support vector machine classification. *J. Comput. Anal. Appl.*, 8(2):99–119, 2006.

[189] J. Yvonnet and Q. C. He. The reduced model multiscale method (R3M) for the non-linear homogenization of hyperelastic media at finite strains. *J. Comput. Phys.*, 223(1):341–368, April 2007.

[190] K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control.* Prentice Hall PTR, Upper Saddle River, New Jersey 07458, 1996.

Deutsche Zusammenfassung

Inzwischen spielt die Modellierung und Simulation natürlicher oder technischer Prozesse eine wichtige Rolle in der Forschung und Industrie. Die nach wie vor anwachsenden Hardwarekapazitäten erlauben dabei immer detailliertere, höher aufgelöste Simulationen in bisher undenkbaren Größenordnungen. Jedoch sind viele realistische Modelle *parametrisiert* durch z.B. variablen Vorgaben in Form- oder Material. In vielen Anwendungen wie der Optimierung oder Kontrolle müssen daher sehr viele verschiedene Lösungen eines Problems für verschiedene Parameter berechnet werden, welches die Gesamtaufgabe zeitlich sehr kostenintensiv machen kann. Um dem entgegenzuwirken, wurde das Gebiet der (mathematischen) Modellordnungsreduktion (MOR) entwickelt, welches zum Ziel hat, das Verhalten eines gegebenen Modells günstig (im Bezug auf Simulationszeit) bei gleichzeitig hinreichender Genauigkeit zu approximieren.

In dieser Arbeit werden MOR Techniken für nichtlineare dynamische Systeme und Multiskalenmodelle untersucht. Dynamische Systeme beschreiben dabei (in diesem Zusammenhang) ganz allgemein die Änderung eines Prozesses über die Zeit, und Multiskalenmodelle beschreiben interagierende Prozesse, die jedoch auf verschiedenen Skalen im Raum oder der Zeit stattfinden. Kernelemente der Arbeit sind die Untersuchung von geeigneten Methoden zur Approximation der in den Modellen auftretenden nichtlinearen Terme und die Bereitstellung von Fehlerschätzern für einige Klassen dynamischer Systeme. Dabei werden Teilbereiche des maschinellen Lernens und der Modellreduktion zusammengeführt.

Konkret werden nach einer Einführung der wichtigsten Grundlagen in Kapitel 1 verschiedene Methoden zur Approximation nichtlinearer Funktionen in Kapitel 2 behandelt. Dabei werden auf Kernmethoden basierende Ansätze wie Support Vektor Maschinen (SVM, siehe [162, 152]) und vektorielle Greedy-Algorithmen [169, 117, 183] untersucht, sowie die bekannte Discrete Empirical Interpolation Method (DEIM) [23].

Anschließend werden verschiedene Verfahren zur MOR von dynamischen Systemen in Kapitel 3 diskutiert, wobei ein spezielles Augenmerk auf die Einbindung der in Kapitel 2 eingeführten Approximationsmethoden gelegt wird. Essentiell für eine effiziente MOR ist die günstige Berechnung von reduzierten nichtlinearen Termen, welche beim Einsatz von Kernmethoden (sowie speziellen Kernen, [131]) und der DEIM möglich ist. Mehrere, später

wiederholt vorkommende numerische Beispiele runden das Kapitel ab. Das Kapitel 4 beschreibt anschließend eine Vorgehensweise zur MOR von Multiskalenproblemen. Dabei wird angenommen, dass die involvierten Mikro- und Makromodelle über eine niedrigdimensionale Schnittstelle Informationen austauschen [121, 16]. Das Ziel ist nun, einige der in Kapitel 2 vorgestellten Approximationsmethoden auf diese Schnittstelle anzuwenden und somit einfache Funktionen als Ersatz für die Mikromodelle zu verwenden. Die Anwendbarkeit dieses Verfahrens mit den vorgestellten Methoden wird danach an mehreren realistischen Problemstellungen aus den Gebieten der Wirbelsäulensimulation [91] und Strömungslehre [94] aufgezeigt, siehe auch [185].

Zuletzt werden in Kapitel 5 die im Rahmen der diskutierten Reduktiontechniken entwickelten Fehlerschätzer eingeführt. Hauptbestandteile der Fehlerschätzer sind dabei die Anwendung des *comparison lemma* [76] zur Gewinnung einer geschlossenen Form und eine effziente Abschätzung lokaler Lipschitzkonstanten, um dem in diesem Zusammenhang häufigen exponentiellen Zuwachs der Fehlerschätzer entgegenzuwirken. Für kernbasierte dynamische Systeme wird in Abschnitt 5.1 ein iterativ verbesserbarer Fehlerschätzer entwickelt, der mit Hilfe von Sekantensteigungen lokale Lipschitzkonstanten berechnet [182] und auch für parametrisierte Systeme anwendbar ist [181]. Für DEIM reduzierte dynamische Systeme wird in Abschnitt 5.2 ein verwandter Fehlerschätzer entwickelt. Dieser benutzt eine auf der Auswertung höherer DEIM-Moden basierende Schätzung des DEIM-Fehlers. Darüber hinaus wird eine effiziente Näherung der lokalen logarithmischen Norm [156] über die Lösung eines kleinen Eigenwertproblems bestimmt. Die dazu verwendete Matrix stellt eine passend ähnlichkeitstransformierte und approximierte Version der lokalen Jacobimatrix der Nichtlinearität dar [186].

Die Arbeit schließt mit einer kurzen Zusammenfassung und einer Diskussion noch offener Fragen und Erweiterungsmöglichkeiten.