Delft University of Technology

Master Thesis

Nonintrusive reduced order modelling using locally adaptive sparse grids and local basis interpolation

by

Denis Bougrimov

A thesis presented for the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on April 20th, 2022 at 14:45.

Student number: 4602269
Project duration: September 1, 2021 - April 1, 2022
Thesis comittee:

Dr. Zoltán Perkó	TU Delft, supervisor
Dr. ir. Danny Lathouwers	TU Delft
Prof. dr. ir. Arnold Heemink	TU Delft

TU Delft

Contents

Abstract

Sensitivity analysis and uncertainty quantification of nuclear reactors requires many expensive high-fidelity simulations. To approximate the dynamics of such a time and parameter dependent system efficiently and effectively, reduced order modelling (ROM) is used. In previous research, a ROM was constructed which used a combination of proper orthogonal decomposition (POD) and a locally adaptive sampling strategy based on sparse grids. To improve the representation of the physics in local parts of the parameter domain, in this thesis, the previous ROM is altered to use multiple local bases instead of one fixed global basis covering the entire parameter domain.

The spatially dependent local bases and local time dependent coefficients are interpolated separately in the parameter domain using the method of interpolation on a tangent space to the Grassmann manifold (ITSGM). The separate interpolators of the local bases and local coefficients are coupled in a space-time coupled approach. The algorithm based on sparse grids is modified so that it can adaptively draw new tangent planes in the parameter domain in a hierarchical manner. This results in the domain being split up into into smaller overlapping subdomains, each having their own tangent plane, number of basis vectors, and interpolators that use radial basis functions (RBF). The algorithm was tested on the Burgers Equation and the Molenkamp test which have an analytical solution. Then, the algorithm was tested on a numerically solved 2D neutron diffusion problem. First the parametric dependence of the modes and coefficients was analyzed, after which the performance of the algorithm was evaluated on these models via different experiments.

The results of the experiments on the 1D Burgers equation and the 2D neutron diffusion problem showed that the algorithm can adaptively and hierarchically draw new tangent planes and can accurately interpolate to new unknown solutions in the subdomains in both a 1D and 2D parameter setting. Higher order non-linear modes and coefficients are harder to interpolate than lower order modes and coefficients. The performance of the interpolator also depends on a combination of the chosen RBF and the size of the subdomains. The results from the Molenkamp test showed that in the smooth setting the new algorithm achieved a higher error with a higher number of evaluations, while in the steep setting the new algorithm performed on par with the previous algorithm, only if the interpolation accuracy threshold is set lower. The results from the neutron diffusion problem indicate that the first principal angle can act as an indicator of which parts of the parameter domain contain more relatable physics than other parts of the domain.

The dependence of RBF interpolator on the subdomain size is caused by how the RBF values scale for further distanced points from the point of interpolation interest. Also more higher order modes than necessary can be included in the local basis interpolation method without worsening the interpolation accuracy of lower order modes, given that no numerical noise is present in the local bases or coefficients. Interpolating the time-dependent behaviour can be easier in the space-time coupled approach than in the approach based on the global basis. However, a lower interpolation accuracy threshold should be chosen as full time evolutions are being interpolated instead of single state vectors. The current interpolation method scales worse than the previous algorithm, as the corner points of the parameter domain will always have to be sampled when using the RBF interpolator compared to local linear basis functions. Additionally, far more data is needed to represent this ROM compared to the ROM based on the global basis.

This research presented a method that generalizes reduced order modelling of time and parameter dependent problems on sparse grids, by using multiple local bases instead of a fixed global basis to represent the underlying physics of a model. The novel local basis interpolation scheme was competitive with the global basis approach on test problems, showing that manifold methods such as ITSGM have great potential to be utilized in reduced order models. However, more research on matrix interpolation methods is needed to improve the overall performance, scalability and efficiency of the algorithm.

1. Introduction

Large-scale complex systems, like nuclear reactors, can have multi-physics interactions on different length and time scales, which can depend on many different input parameters. To be able to capture all the relevant dynamics of such a nuclear reactor, high fidelity models are evaluated. However, as these models are expensive to evaluate, multiple repeated evaluations of them are not possible, which is needed in applications like uncertainty and sensitivity analysis. To approximate the underlying dynamics of the system efficiently and effectively, reduced order models are used (ROM). To build a ROM, a limited amount of high-fidelity model evaluations is required for various combinations of input parameters. These evaluations are then used to build the ROM in the offline phase after which new unknown solutions over the full range of input parameters can be rapidly and cost effectively approximated in the online phase.

There are many different ROM methods that have their advantages and disadvantages depending on the model that it is built from. For instance, applications of proper orthogonal decomposition (POD) are used to study unsteady flows and model turbulence, while alterations of rational interpolation and balanced truncation methods are used in a wide range of topics such as circuit theory, to analyze and predict signal propagation and interference in electric circuits or structural mechanics, and to study vibration suppression in large structures or behavior of micro-electromechanical systems. The main problem in building a ROM for a large-scale complex non-linear system like a nuclear reactor, is the curse of dimensionality, which is the exponential increase of computational resources and high-fidelity calculations as the parameter space dimension gets larger. Therefore, according to a survey that reviewed most of the existing ROM methods [1], for non-linear systems, the ROM method of POD is most suitable. The main use of POD in physics is to decompose a set of precalculated high fidelity snapshots, i.e. an ensemble of solution fields, into a set of basis functions and coefficients [2]. In essence, each basis function carries a fraction of the energy of the ensemble and the ROM is constructed by finding the smallest possible set of basis functions which can be used to reconstruct each snapshot of the ensemble with a high enough accuracy. An example is a set of state vectors like flux ϕ or temperature $\mathbf{T}$ that depend on different system parameters and time, which can be decomposed into spatially dependent basis functions and respective parameter and time dependent coefficients. The advantage of using POD is that it can be used in a data-driven way where no knowledge of the governing equations of the system is needed. Thus, it operates in a non-intrusive way where the model is seen as a black box and the POD model only operates on the inputs and outputs. This is advantageous in for instance nuclear reactor simulations, in which most of the time no analytical solutions are available in the neutronics and fluid dynamics calculations.

For instance, in [3], a reduced basis method was used to model control rod movement for a nuclear reactor core. A multi-group neutron diffusion equation was used to calculate the neutron kinetics which is parameterised by the height of the control rod. From a full-order model of dimension 133,810, a reduced model was built which reached a computational speedup of a factor of 30,000, while the neutron flux distribution could be reproduced with a relative accuracy of 10^{-4}. In [4], a ROM based on POD was built in combination with a locally adaptive sampling strategy based on sparse grids for a model of a molten salt fast reactor. The main idea of these sparse grids is that the high-fidelity model is only evaluated at points in the input parameter space that improve the overall ROM by a high enough value and represent most of the behaviour in the system. An example of how such a ROM is build up and used can be found in [5]. Only between 462 and 4495 points were evaluated in a parameter domain with dimension 27, and new neutron multiplication factors and flux solutions could be approximated within an accuracy of $5 \cdot 10^{-5}$ and 1% respectively. In [6], the method of POD was combined with artificial neural networks. A reduced basis was extracted from a collection of high-fidelity solutions via POD, after which multi-layer perceptrons were used to accurately approximate the coefficients of the reduced order model. The POD+ANN method was then tested on the nonlinear Poisson equation and driven

cavity viscous flows, described by steady incompressible Navier–Stokes equations. A local basis approximation approach was developed recently in [7], where POD is combined with a manifold interpolation approach where the local POD basis matrices themselves are interpolated in the parameter domain. This method was first tested on a shear-frame structure, and then on a more complex 3D numerical case study of an earthquake-excited wind turbine tower. More background information on ROMs can be found in papers [8–16].

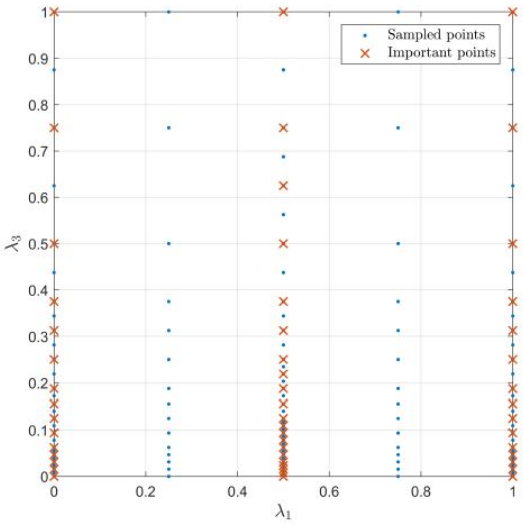


Figure 1: Example of a sparse grid in which the model is only evaluated at specific parameter values over the parameter domain. The important points are the ones that in the end are added to the ROM.

Every input parameter can have a different effect on the overall state of a system. For instance, for some parameters, transients can occur because of a change in one of these parameters in the system, after which the reactor again goes to a new stable or unstable state. Some sets of parameters in the overall parameter domain will have a similar effect and some groups of parameters will have a different effect. A simple example of this could be a fission cross-section value, where a higher fission cross-section might result in the reactor becoming critical while a lower fission-cross section might leave the reactor at a sub critical state. The physics in these critical or sub critical domains might be completely different from each other, meaning that the basis vectors representing the system could also change between these regions. In previous research [4], a global basis that represents the basis vectors spanning the full parameter domain was used. The first problem with this global basis is that the global basis coefficients are more non-linear than local coefficients. The second problem is that some modes are only needed locally. The main cause of these problems is that the physics is being represented with more information, in this case global basis vectors, than is necessary in some parts of the parameter domain, where only local basis vector might be needed. Moreover, using local bases instead of a fixed global basis is a more generalized approach, in which not only the coefficients of the basis vectors are interpolated, but the actual local bases that represent the physics are interpolated as well. Therefore, in this research, local bases from multiple sets of local snapshot sets are constructed. A method to interpolate the local bases, is based on interpolation on a tangent space to the Grassmann manifold (ITSGM). In general, the main goal of this research is to combine a local basis interpolation method that can be used in a time and parameter dependent setting, with the locally adaptive sparse grid sampling scheme and test the performance of this algorithm.

The report is structured in the following way: Chapter 2 comprises all background information and needed theory. Then, in Chapter 3, different models on which the algorithm is tested, are described in detail with their respective experiments. Afterwards, the results are shown in Chapter 4, which is split up in a section where the modes of the local bases are analyzed and another section in which the performance of the algorithm is examined in context of the experiments. In Chapter 5 some of the results are discussed that are unexpected and require attention. In Chapter 6 a conclusion is made which will be the end of the thesis. The references and an appendix with extra plots are provided at the end of the report.

2. Theory

In the following chapter, the essential background theory is explained in detail. In Section 2.1, proper orthogonal decomposition and the locally adaptive sampling strategy based on sparse grids will be described. In Section 2.2, basic theory about manifolds is given that includes important details about the Grassmann and Stiefel manifold. In Section 2.3, the method of interpolation of local bases on a tangent plane to the Grassmann manifold will be explained, which also includes a space-time coupled approach. Afterwards, the full algorithm is described that combines the local basis interpolation method with the locally adaptive sparse grids in Section 2.4.

2.1 Reduced order modelling

2.1.1 Proper orthogonal decomposition

Let $f(\mathbf{x})$ be a function, depending on a vector $\mathbf{x} \in \Omega \subset \mathbb{R}^{N_x}$, that is going to be approximated. $f(\mathbf{x})$ can then be written as a linear combination of N_m basis functions $\varphi^i(\mathbf{x})$ with coefficients c_i

$$f(\mathbf{x}) \cong \sum_{i=1}^{N_m} c_i \varphi^i(\mathbf{x}). \tag{1}$$

The following minimization problem results in the best approximation of $f(\mathbf{x})$:

$$\min_{\varphi^i} \left\| f(\mathbf{x}) - \sum_{i=1}^{N_m} c_i \varphi^i(\mathbf{x}) \right\|_{L_2}, \tag{2}$$

where

$$\|f(\mathbf{x})\|_{L_2} = \sqrt{\int_\Omega |f(\mathbf{x})|^2 dx} \tag{3}$$

is the L2 norm. The basis functions of $f(\mathbf{x})$ are required to form an orthonormal set

$$\int_\Omega \varphi^{k1}(\mathbf{x}) \varphi^{k2}(\mathbf{x}) dx = \begin{cases} 1 & k_1 = k_2 \\ 0 & k_1 \neq k_k \end{cases}, \tag{4}$$

which leads to a formula for the coefficients c_i:

$$c_i = \int_\Omega f(\mathbf{x}) \varphi^i(\mathbf{x}) dx. \tag{5}$$

In Equation (5), each coefficient c_i only depends on its own basis function $\varphi^i(x)$, which can be explained by the orthogonality of all the basis functions. Each set of basis functions must abide Equation (2), which means that each set of basis functions results in a minimum error in the approximation of $f(\mathbf{x})$. There are different ways to construct such a POD basis. As in this work the data will not be represented by functions $f(\mathbf{x})$, but by matrices $S(\mathbf{x})$, the Single Value Decomposition-approach (SVD) is used to find the solution to the minimisation problem, using the method from [17].

The singular value decomposition is a matrix decomposition method that can be used to decompose any arbitrary matrix $\mathbf{S} \in \mathbb{R}^{m \times n}$ into a product of matrices $\mathbf{U} \in \mathbb{R}^{m \times n}$, $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ given by Equation (6).

$$\mathbf{S} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T \tag{6}$$

The elements on the diagonal of $\mathbf{\Sigma}$ are called the singular values σ_i and are ordered in a decreasing fashion along the diagonal in the following way: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$ with rank $n = \min(m, n)$.

The columns of matrix $\mathbf{U}$ and $\mathbf{V}$ in Equation (6) represent the left and right singular vectors. $\mathbf{U}$ is often called the basis, and the product of $\mathbf{\Sigma}$ and $\mathbf{V}$ is called the matrix of expansion coefficients. It is important to note that the matrices $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices. For a singular value σ of matrix $\mathbf{S}$, there exist vectors $\mathbf{u}$ with length m and vector $\mathbf{v}$ with length n both with unit-length, such that

$$\mathbf{S}\mathbf{v} = \sigma\mathbf{u} \text{ and } \mathbf{S}^T\mathbf{u} = \sigma\mathbf{v}. \tag{7}$$

An approximation of matrix $\mathbf{S}$ can be made by keeping only the first left N_m singular values. To make sure the dimensions in the SVD match the matrices are truncated resulting in $\mathbf{U}_{tr} \in \mathbb{R}^{m \times N_m}$, $\mathbf{\Sigma}_{tr} \in \mathbb{R}^{N_m \times N_m}$ and $\mathbf{V}_{tr} \in \mathbb{R}^{n \times N_m}$. The approximation of $\mathbf{S}$ is then given by:

$$\mathbf{S} \approx \mathbf{S}_{tr} = \mathbf{U}_{tr}\mathbf{\Sigma}_{tr}\mathbf{V}_{tr}^T. \tag{8}$$

In the case where POD is applied on a vector function, Equation (1) becomes

$$\mathbf{y}(\mathbf{x}) \approx \sum_{i=1}^{N_m} c_i(\mathbf{x})\mathbf{u}_i, \tag{9}$$

where the outputs $\mathbf{y}(\mathbf{x})$ are approximated with basis vectors $\mathbf{u}_i$ and the amplitudes c_i. Placing p number of output vectors $\mathbf{y}(\mathbf{x})$ of length d in a matrix, where each vector defines the state for a different parameter combination, leads to the following:

$$\mathbf{M} = [\mathbf{y}(\mathbf{x}_1), \mathbf{y}(\mathbf{x}_2), \ldots, \mathbf{y}(\mathbf{x}_p)] \in \mathbb{R}^{d \times p}. \tag{10}$$

In this case the minimization problem of Equation (2) becomes:

$$\min_{\mathbf{u}_i} E = \min_{\mathbf{u}_i} \left[\sum_{j=1}^{p} \left\| \mathbf{y}(\mathbf{x}_j) - \sum_{i=1}^{N_m} c_i(\mathbf{x}_j)\mathbf{u}_i \right\|_{L_2} \right], \tag{11}$$

where E represents the error. To produce an approximation of $\mathbf{M}$, only the first N_m left singular vectors are taken as a set of basis vectors for the reduced basis. Then, by using the orthogonality of the basis functions from Equation (4), the amplitude $c_i(\mathbf{x}_j)$ can be calculated by projecting each output vector/solution onto the reduced basis:

$$c_i(\mathbf{x}_j) = \langle \mathbf{u}_i, \mathbf{y}(\mathbf{x}_j) \rangle, \tag{12}$$

where $\langle \cdot, \cdot \rangle$ is a dot product operation between two vectors. The number of POD modes or basis vectors N_m in the reduced basis is directly tied to the dimensionality of the ROM: using more vectors is computationally more expensive but is more accurate. To determine the number of modes N_m that are needed for a certain approximation error, the following truncation error can be calculated:

$$e_{\text{tr}} = \sqrt{\frac{\sum_{k=N_m+1}^{n} \sigma_k^2}{\sum_{k=1}^{n} \sigma_k^2}} < \gamma_{tr}, \ \forall N_m \in [1, \ldots n]. \tag{13}$$

It is important to note that the truncation error is defined for the approximation of the whole matrix $\mathbf{M}$, not of the individual vectors $\mathbf{y}(\mathbf{x}_j)$. So it could be that the approximation error of an individual vector is larger or smaller than the given truncation error. In previous research [4], an SVD was done on a single snapshot matrix $\mathbf{M}$ representing the states over the whole parameter domain, including time as a parameter as well. $\mathbf{M}$ consists of all local snapshot matrices $\mathbf{S}_i$, which contain the snapshot vectors for a certain input parameter vector $\boldsymbol{\lambda}_i$ with a certain time discretization $\mathbf{t}$. $\mathbf{S}_i$ is defined in the following way:

$$\mathbf{S}_i = [\mathbf{y}(\boldsymbol{\lambda}_i, t_1), \ldots, \mathbf{y}(\boldsymbol{\lambda}_i, t_{N_t})] \in \mathbb{R}^{N_x \times N_t}, \quad i = 1, \ldots, N_\lambda, \tag{14}$$

after which $\mathbf{M}$ is formed by concatenation of these snapshot matrices $\mathbf{S}_i$:

$$\mathbf{M} = [\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_{N_\lambda}] \in \mathbb{R}^{N_x \times (N_\lambda \times N_t)}. \tag{15}$$

Each snapshot vector $\mathbf{y}(\boldsymbol{\lambda}_i, t_\tau) \in \mathbb{R}^{N_x}$ represents the state of a physical field for a certain input parameter vector $\boldsymbol{\lambda}_i$ at a certain time point and has length N_x which is the number of degrees of freedom in the system given by its spatial discretization $\mathbf{x}$. Applying the SVD from Equation (6) on the matrix $\mathbf{M}$ results in the following:

$$\mathbf{M} = \mathbf{U}_{global}\boldsymbol{\Sigma}(\mathbf{V}(\boldsymbol{\lambda}))^T, \tag{16}$$

where $\mathbf{U}_{global} \in \mathbb{R}^{N_x \times (N_\lambda \times N_t)}$ is a global basis with basis vectors that represent the physics in the entire parameter domain and $\mathbf{V}(\boldsymbol{\lambda})$ is defined by:

$$\mathbf{V}(\boldsymbol{\lambda}) = [\mathbf{v}(\lambda_1, t_1), \ldots, \mathbf{v}(\lambda_1, t_{N_t}), \ldots, \mathbf{v}(\lambda_{N_\lambda}, t_1), \ldots \mathbf{v}(\lambda_{N_\lambda}, t_{N_t})] \in \mathbb{R}^{(N_\lambda \times N_t) \times (N_\lambda \times N_t)}, \tag{17}$$

where the vectors $\mathbf{v} \in \mathbb{R}^{N_\lambda \times N_t}$ contain the time and parameter dependent coefficients of the all the basis vectors in $\mathbf{U}_{global}$. The truncation error in Equation (13) that is used to calculate the number of modes needed for a certain maximum approximation accuracy, is valid only for the approximation of the full snapshot matrix and not the single state vectors in the snapshot matrix. The issue with this comes into play when dealing with systems in which the physics can change very abruptly in a small part of the parameter domain. As that small part of the parameter domain will only account for a small fraction of the state vectors in the full snapshot matrix, the basis vectors, or physics that these state vectors add to the global basis $\mathbf{U}_{global}(\mathbf{x})$ will be minimal compared to the basis vectors that represent the rest of the parameter domain. When truncating the global basis based on the truncation error, only the N_m left singular vectors are taken that represent most of the physics in the full snapshot matrix, leading to the potential removal of basis vectors that represent that important small part of the parameter domain. Another problem is that the dependence of the coefficients of the modes in the global basis can be much more complex than those of a local basis, as local solutions for certain parameter and time combinations are being represented with more information, in this case basis vectors, than is necessary in some parts of the parameter/time domain. Furthermore, using local bases rather than a fixed global basis is a more generalized technique, since it interpolates not only the coefficients of the basis vectors, but also the actual local bases that describe the physics.

Therefore, in this research, local bases from multiple sets of local snapshot matrices $\mathbf{S}_i$ are constructed, where each set of snapshots represents the time dependent physical field at only one parameter combination. Doing an SVD on these snapshot matrices leads to:

$$\mathbf{S}(\boldsymbol{\lambda}_i) = \mathbf{U}(\boldsymbol{\lambda}_i)\boldsymbol{\Sigma}(\boldsymbol{\lambda}_i)(\mathbf{V}(\boldsymbol{\lambda}_i))^T \tag{18}$$

where $\mathbf{U}(\boldsymbol{\lambda}_i)$ are the local parameter dependent basis matrices, with their respective expansion coefficient in the product of $\boldsymbol{\Sigma}(\boldsymbol{\lambda}_i)$ and $\mathbf{V}(\boldsymbol{\lambda}_i)$. In previous research, only the expansion coefficients in Equation (12) were interpolated, but in this research 3 different interpolators are built for the $\mathbf{U}(\boldsymbol{\lambda}_i)$, $\boldsymbol{\Sigma}(\boldsymbol{\lambda}_i)$ and $\mathbf{V}(\boldsymbol{\lambda}_i)$ matrices.

2.1.2 Locally adaptive sampling on sparse grids

In physical models that are characterized by many different input parameters, the accuracy and efficiency of the POD method will be highly dependent on the parameter sampling scheme. Moreover, it is required that the sampling scheme results in a ROM which can accurately represent the dynamics of the full-order model with as few high-fidelity simulations as possible. Sampling the points randomly in the parameter space is not the best strategy, as there is a chance that some physical behaviour might be missed. Sampling the whole domain with small intervals through each dimension is also not an option, as this is too computationally expensive due to the curse of dimensionality. The best solution would be to use sparse grids, which are built by following a hierarchical tree in which the distance between nodes becomes smaller and smaller as the depth of the tree increases. The values in the tree are scaled by mapping the parameter space to a space with a range of [0; 1]. In Figure 2, the layout of this tree is shown, where the tree starts at the first node at level $i = 1$ with value 0.5 and grows out as i increases. In a 1D parameter domain,

each node at a level i has one father point at level $i-1$ and two children at level $i+1$ except for $i=2$ where the nodes onlv have one child.

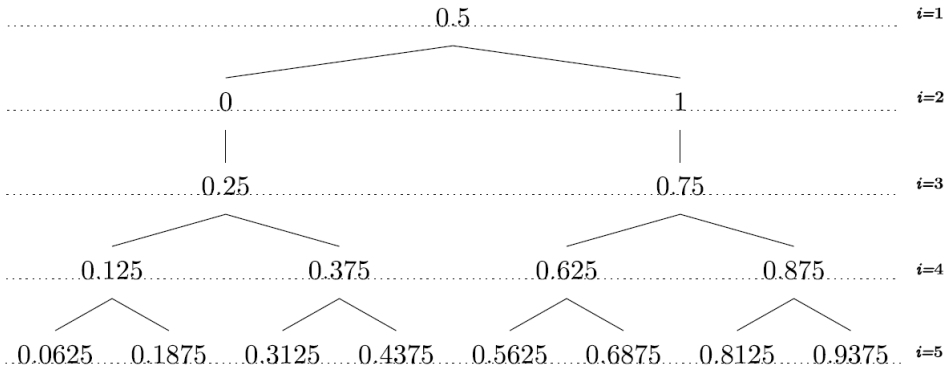


Figure 2: The tree structure that the sparse grid is based on. Each level i contains a set of equidistant nodes [4].

The equations that represent the sampling scheme are given by:

$$m^i = \begin{cases} 1 & \text{if } i = 1, \\ 2^{i-1} + 1 & \text{if } i > 1, \end{cases} \tag{19}$$

$$\lambda_j^i = \begin{cases} 0.5 & \text{for } j = 1 \quad \text{if } m^i = 1, \\ \frac{j-1}{m^i-1} & \text{for } j = 1, 2, \ldots, m^i \quad \text{if } m^i > 1, \end{cases} \tag{20}$$

where m_i represents the number of nodes at level i, j represents the index of a point in a level and λ_j^i represents the scaled parameter value between 0 and 1 of node j at level i. A generalisation can also be made for the multidimensional case, where each parameter point is multidimensional. In this case index i represents the level in the tree for a single dimension, index l represents the level of a point in multiple dimensions. In multiple dimensions every node is surrounded by δ (number of dimensions) backward points or fathers. These fathers have all but one parameter the same as their child which is connected to a value higher up in the tree. Every point also has 2δ forward points or children, again with all but one parameter the same which is connected to a value one step lower down the tree. Nodes at level $l = 2$ are an exception, as these have fewer forward points. It is also important to note that in multiple dimensions a node can be reached from several backward points. An example of such a point is $[0; 0.25]$, which can be reached from $[0; 0]$ or $[0.5; 0.25]$. In Figure 3 the evolution of the sparse grid is shown for a 2D parameter domain:

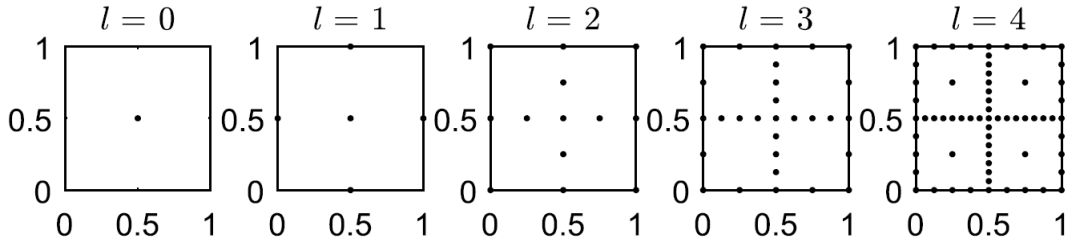


Figure 3: Sparse grid points in a 2D setting for different levels l [4].

An example of how locally adaptive sampling on a sparse grid works in 1D is as follows: First the parameter space is mapped to a space with a range of [0; 1]. The algorithm starts by sampling one point in the middle of the domain at [0.5], which is level $i = 1$. Then it extrapolates the solution to the 'children' of 0.5 which are points 0 and 1 in level $i = 2$ and samples the true high-fidelity solutions at these points. The approximation error of the extrapolation at these points is then checked and if the error is larger than a certain threshold, the next children of these points are sampled. For this example, it is assumed that the extrapolation error at 0 and 1 is higher than

a certain threshold. This results in the children of points 0 and 1 being sampled, which are 0.25 and 0.75 at $i = 3$. If the interpolation at a point, for instance 0.25, is deemed accurate enough, the children 0.125 and 0.375, will not be simulated in the next iteration to save up time, while the children of 0.75, so 0.625 and 0.875, are sampled. In short, points at subsequent levels are checked by interpolating the results from the important points of previous generations.

The adaptivity comes into play when the errors are checked according to a threshold error. If the error of a certain point is too high, that point is deemed important, and its forward points (children) are generated. To sample the children, a forward operator is used that acts on a set of points, in this case $\mathcal{S} = \{\boldsymbol{\lambda}_s | s = 1, \ldots, N_s\}$, and returns all forward points for the points in $\mathcal{S}$ as follows:

$$F(\mathcal{S}) = \{(\lambda_{f,1}, \ldots, \lambda_{f,\delta}) \mid \exists i, s : b(\lambda_{f,i}) = \lambda_{s,i} \wedge \lambda_{f,j} = \lambda_{s,j}\ \forall j \neq i, s \in [1, \ldots, N_s], j, i \in [1, \ldots \delta]\}, \tag{21}$$

where $b(\lambda_f)$ is a function that returns the father of a child node λ_f from the tree. A backward point can also be defined for λ which is a point with a parent node along one of the dimensions of λ. From this, a backwards operator B can be defined that operates on the set $\mathcal{S}$ and returns the set of all backward points and can be defined as:

$$B(\mathcal{S}) = \{(\lambda_{b,1}, \ldots, \lambda_{b,\delta}) \mid \exists i, s : b(\lambda_{s,i}) = \lambda_{b,i} \wedge \lambda_{b,j} = \lambda_{s,j}\ \forall j \neq i, s \in [1, \ldots, N_s], j, i \in [1, \ldots \delta]\}. \tag{22}$$

By applying the backward operator successively, a set of ancestor points $\mathcal{S}$ for all points λ in $\mathcal{S}$ can be defined via:

$$\Gamma(\mathcal{S}) = \bigcup_{s=1}^{L} (B)^s (\mathcal{S}), \tag{23}$$

where $(B)^L (\mathcal{S}) = (0.5, \ldots, 0.5)$. After iteration $k-1$, there is a set of important points $\mathcal{Z}^{k-1} = \{\boldsymbol{\lambda}_\zeta | \zeta = 1, \ldots, N_z\}$ and an unimportant/inactive set $\mathcal{I}^{k-1}$. The points in the test set $\mathcal{T}^k = \{\boldsymbol{\lambda}_r | r = 1, \ldots, N_r\}$ for the next iteration are selected via the forward operator F applied on the old set of important points $\mathcal{Z}^{k-1}$ which is defined as:

$$\mathcal{T}^k = F(\mathcal{Z}^{k-1}). \tag{24}$$

The data at the important points $\mathcal{Z}^{k-1}$ are then interpolated to approximate the solutions at the testing points $\mathcal{T}^k$, leading to N_r interpolated snapshot matrices $\mathbf{S}_{int,r}$ containing interpolated snapshots $[\mathbf{y}_{int}(\boldsymbol{\lambda}_r, t_1), \ldots, \mathbf{y}_{int}(\boldsymbol{\lambda}_r, t_{N_t})]$. Afterwards, at each parameter combination in $\mathcal{T}^k$ the interpolation accuracy is checked by calculating the following error over time:

$$\epsilon_{r,\tau} = \frac{\|(\mathbf{y}_{true}(\boldsymbol{\lambda}_r, t_\tau) - \mathbf{y}_{int}(\boldsymbol{\lambda}_r, t_\tau))\|_{L_2}}{(\|\mathbf{y}_{true}(\boldsymbol{\lambda}_r, t_\tau)\|_{L_2} + \eta)}, \quad \tau = 1, \ldots, N_t,\ r = 1, \ldots, N_r, \tag{25}$$

where $\mathbf{y}_{true}(\boldsymbol{\lambda}_\mathbf{r}, t_\tau)$ are the true snapshot vectors for a certain time and parameter combination. η is an offset that is added for state vectors that have a near zero magnitude. Afterwards, the maximum of the error is taken in time for each parameter combination in $\mathcal{T}^k$, which is defined by:

$$\epsilon_r = \max_{\epsilon_{r,\tau}} \{\epsilon_{r,\tau}, \tau = 1, \ldots, N_t\},\ r = 1, \ldots, N_r. \tag{26}$$

If the error ϵ_r at a given point in $\mathcal{T}^k$ is higher than a chosen error threshold, then the point is not estimated accurately enough by the interpolant from the last iteration. This means that this point needs to be included into the set of important points $\mathcal{Z}^k$ at iteration k. In short, $\mathcal{Z}^k$ is given by:

$$\mathcal{Z}^k = \{\boldsymbol{\lambda}_r \in \mathcal{T}^k | \epsilon_r > \gamma_{int}\}, \tag{27}$$

where γ_{int} is the threshold for the maximum error between the true and approximated solution at the testing point. For a multivariate sparse grid, there is also a greediness parameter μ that can be set, which is the minimum percentage of fathers of a child, on which the interpolation error should be higher than γ_{int} for the child to be sampled. For example, in a 2D setting each child point after level l=2, has 2 fathers. If μ is set to 50% then only at one of the fathers the error should be

higher than γ_{int}, meaning that only one of the fathers should be included as an important point in $\mathcal{Z}^k$ for the child to be sampled in the new test set $\mathcal{T}^{k+1}$. Maximally greedy results in less points being sampled.

2.2 Manifold theory

2.2.1 Basic concepts of differential geometry

The simplest way to think about an abstract concept as a manifold is to look at our own planet Earth as a sphere, which is a three-dimensional geometric object. Yet, it is possible to make two-dimensional representations of earth, which are the flat maps that constitute an atlas and are used to navigate on a local patch on earth. To understand the meaning of flatness the following example is used. The sum of the angles of a triangle on a sphere is not equal to 180°, as a sphere is not a flat space. However, locally, a patch on the sphere can be approximated as a flat space on which the sum of the angles is almost equal to 180°. In mathematical terms, the local flat patch is a Euclidean space with no curvature, while the whole sphere is not a Euclidean space as it is curved. To make a full two-dimensional image of the surface of earth, multiple maps must be essentially glued together and at the edges of the maps there must be a transition from one map to the other.

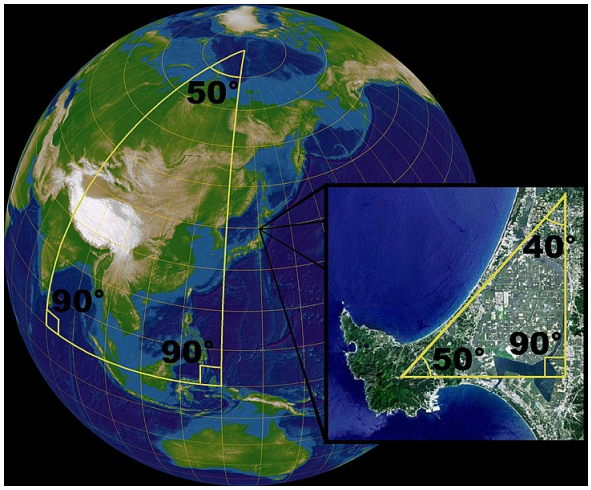


Figure 4: Visualization of how the angles inside a triangle change in a Euclidean and non-Euclidean space [18].

In mathematical terms [19], a manifold is a topological space built up of local regions, with each local region being homeomorphic to a subset of Euclidean space of dimension $\mathbb{R}^n$. A local region contains a subset of all points that constitute the manifold. Homeomorphic means that points from the manifold can be mapped to a subset of Euclidean space via so called charts, which are invertible maps (invertible continuous functions) between a local region of the manifold and the Euclidean space. So, a chart is a combination of a local patch on the manifold and its respective map, and the collection of charts that cover the manifold constitute an atlas. For a sphere, a chart allows for a coordinate transformation of a coordinate on the sphere to a coordinate on a local Euclidean space that the chart represents.

In Figure 5, the local patches P_α and P_β have their respective maps Φ_α and Φ_β, leading to charts (P_α, Φ_α) and (P_β, Φ_β). Note that P_α and P_β share a region on the manifold. If there are two charts in an atlas that map two partly overlapping regions to Euclidean space, and there is a smooth invertible function $\Phi_\beta \circ \Phi_\alpha^{-1}$ that can represent the coordinates of that overlapping region in both Euclidean spaces, then mathematically speaking, the charts are said to be locally compatible with each other. The symbol $\circ$ denotes a function composition, for example $(g \circ f)(x) = g(f(x))$. If all the charts are locally compatible with each other, one can define directions and differentiable functions on the manifold, which results in a differentiable or smooth manifold [20, 21]. These

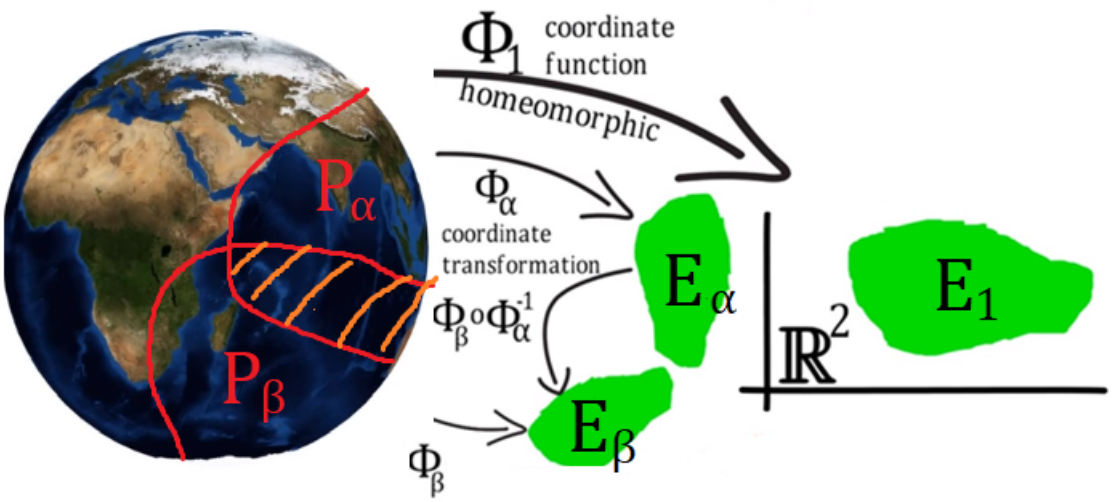


Figure 5: Coordinate functions can be used to do a coordinate transformation of points on the manifold to the Euclidean space and vice versa. Overlapping local regions on the manifold require that the individual charts are compatible with each other.

manifolds are generalizations of curves and surfaces to arbitrary dimensions and are locally similar enough to a vector space which allows one to do calculus. To each point in a differentiable manifold a tangent plane can be assigned, which is also an n-dimensional 'flat' Euclidean vector space that consists of tangent vectors of the curves that run through that point on the manifold. The useful thing about this tangent space is that due to the Euclidian nature there is no curvature to be dealt with, which makes interpolating in this space much easier.

A special class of differential manifolds is a Riemannian manifold, in which on every point there is a tangent space that is equipped with an inner product, which varies smoothly from point to point. This inner product allows one to define the length, area, curvature, and divergence of vector-fields. An important problem in differential geometry and data processing on manifolds is to determine the shortest path between two points on the manifold. The length of a curve in Euclidean space defined by $c : [a, b] \to \mathbb{R}^n$ is $L(c) = \int_a^b ||\dot{c}(t)||dt$, where $\dot{c}$ is the velocity on the curve and t can be associated with 'time'. However, in the manifold setting, the inner product for tangent vectors is needed that is consistent with the manifold structure. Let $\mathcal{M}$ be a manifold with a point p on the curve $c(t) \in \mathcal{M}$ and $\mathcal{T}_p\mathcal{M}$ a tangent plane at point p with a tangent vector v. This situation is presented in Figure 6.

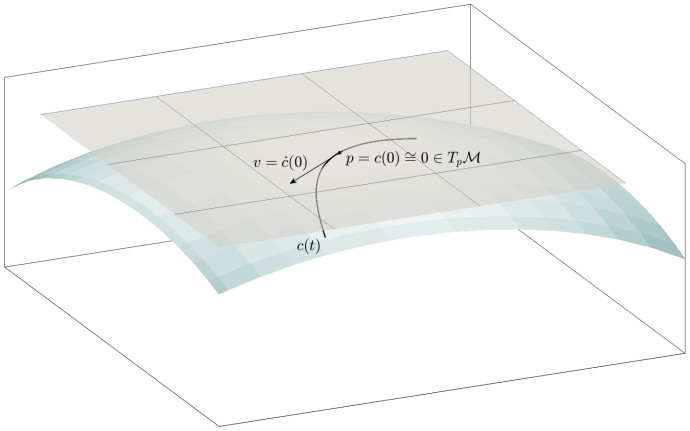


Figure 6: Visualization of a manifold, represented by the curved surface with the tangent space $\mathcal{T}_p\mathcal{M}$ attached at point p. The tangent vector $v = \dot{c}(0) \in T_p\mathcal{M}$ is the velocity vector of a curve $c : t \mapsto c(t) \in \mathcal{M}$ [22].

The length of a tangent vector $v \in \mathcal{T}_p\mathcal{M}$ on a point $p \in \mathcal{M}$ is then defined by $||v||_p := \sqrt{\langle v, v \rangle_p}$, where $\langle v, v \rangle_p$ defines a dot product between vectors in the tangent space at point p which is a vector space. The length of a curve on a manifold then becomes [22]:

$$L(c) = \int_a^b ||v||_p dt = \int_a^b ||\dot{c}(t)||_{c(t)} dt = \int_a^b \sqrt{\langle \dot{c}(t), \dot{c}(t) \rangle_{c(t)}} dt. \tag{28}$$

At every point $p \in \mathcal{M}$ the Euclidean space $\mathbb{R}^n$ can be decomposed into an orthogonal direct sum:

$$\mathbb{R}^n = T_p\mathcal{M} \oplus T_p\mathcal{M}^\perp, \tag{29}$$

where $T_p\mathcal{M}^\perp$ is the orthogonal complement of $T_p\mathcal{M}$. The orthogonal complement of the subspace $T_p\mathcal{M}$ of a vector space in $\mathbb{R}^n$, equipped with a bilinear form B, is the set $T_p\mathcal{M}^\perp$ of all vectors in $\mathbb{R}^n$ that are orthogonal to every vector in $T_p\mathcal{M}$. The dot product is an example of a bilinear form which is a function that is linear in both arguments of the dot product and the orthogonality is defined with respect to the inner product on $\mathbb{R}^n$.

For a curve on a manifold that is embedded in Euclidian space, the orthogonal projection of the directional derivative along the curve onto the tangent space of the manifold is called the covariant derivative. In simple terms, it is the part of the directional derivative that someone living on the manifold can see. In mathematical terms, let $\frac{D}{dt}(t)$ denote the covariant derivative in the direction of t and let $\Pi_p : \mathbb{R}^n \to T_p\mathcal{M}$ denote the orthogonal projection onto the tangent space at p. The covariant derivative of a vector field $v(t) \in T_{c(t)}\mathcal{M}$ along a curve $c(t)$ is then the tangent component of $\dot{v}(t)$, which can be written as $\frac{Dv}{dt}(t) = \Pi_{c(t)}(\dot{v}(t))$. In terms of the curve $c(t)$ this would be:

$$\frac{D\dot{c}}{dt}(t) = \Pi_{c(t)}(\ddot{c}(t)). \tag{30}$$

On a manifold $\mathcal{M}$, a curve $c : [a, b] \to \mathcal{M}$ is called a geodesic, if the curve is a uniquely defined path with the shortest distance between two points on the manifold and the covariant derivative of its velocity vector field vanishes, i.e. the geodesic equation:

$$\frac{D\dot{c}}{dt}(t) = 0 \; \forall t \in [a, b], \tag{31}$$

holds. So, it can be stated that the geodesics on Riemannian manifolds are the constant-speed curves with acceleration vectors orthogonal to the corresponding tangent spaces, i.e., $\ddot{c}(t) \in T_{c(t)}\mathcal{M}^\perp$ and the metric of these geodesics is induced by the Euclidean inner product. An example of a geodesic is a great circle on a unit sphere. When considered as curves in $\mathbb{R}^3$, the velocity vector along the great circle is always perpendicular to the acceleration vector pointing to the center of the sphere. When viewed as entities on a 2D surface S^2, the curves do not experience any acceleration, as they are just straight lines on a surface.

For a given starting point $c(0) = p \in \mathcal{M}$ and a starting velocity $\dot{c}(0) = v \in \mathcal{T}_p\mathcal{M}$, the geodesic equation in Equation (31) can be translated to an initial value problem of second order with guaranteed existence and uniqueness of a solution. Using this, a tangent vector $v \in \mathcal{T}_p\mathcal{M}$ can be mapped to the endpoint of a geodesic that starts from $p \in \mathcal{M}$ with velocity v. This mapping is done back and forth via so-called exponential and logarithmic maps [22]. In simple terms, the exponential map takes a tangent vector to the manifold at a certain point and runs along the geodesic starting at that point for a unit time. Naturally, the distance traveled over the manifold, depends on the velocity. The exponential map is formalized as follows:

$$\mathrm{Exp}_p^{\mathcal{M}} : T_p\mathcal{M} \supset B_\varepsilon(0) \to \mathcal{M}, \quad v \mapsto q := \mathrm{Exp}_p^{\mathcal{M}}(v) := c_{p,v}(1). \tag{32}$$

Here, $t \mapsto c_{p,v}(t)$ is the geodesic that starts from p with velocity v and $B_\varepsilon(0) \subset T_p\mathcal{M}$ is the open ball with radius ε with center 0 in the tangent space. An open ball of radius ε is a collection of points with a distance less than ε from a fixed point in Euclidean space. To clarify, the Riemannian exponential map provides a local parameterisation of a small region around a location $p \in \mathcal{M}$ in terms of coordinates of the flat vector space $T_p\mathcal{M}$, which is referred to as representing the manifold

in normal coordinates. Normal coordinates are defined in a way such that the Riemannian distance between p and $q = \mathrm{Exp}_p^{\mathcal{M}}(v)$ is the same as the length of the tangent vector $\|v\|_p$ as measured in the metric on $T_p\mathcal{M}$. This is only valid given that v is contained in a neighborhood of $0 \in T_p\mathcal{M}$, where the exponential map is a diffeomorphism between $B_\varepsilon(0)$ and an open domain on $\mathcal{D}_p \subset \mathcal{M}$ around the point p. A diffeomorphism is defined as a smooth, differentiable, invertible map between manifolds. This means that the exponential map has a smooth invertible map that is only valid locally, because the tangent plane only linearises the manifold in a small area around the point of tangency. The inverse map is called the Riemannian logarithm and is defined as:

$$\log_p^{\mathcal{M}} : \mathcal{M} \supset \mathcal{D}_p \to B_\varepsilon(0) \subset T_p\mathcal{M}, \quad q \mapsto v := \left(\mathrm{Exp}_p^{\mathcal{M}}\right)^{-1}(q), \tag{33}$$

where v satisfies $c_{p,v}(1) = q$. So, the logarithmic map, sends a point q on the manifold to a tangent vector defined on the tangent space at a different point p on the manifold. The workings of the exponential and the logarithmic map are shown in Figure 7. The Riemannian exponential depends on the Riemannian metric that defines the length of the geodesics. This Riemannian metric is defined by inner product on the tangent plane. Different manifolds can have different metrics, which leads to different geodesics and thus to different exponential and logarithm maps.

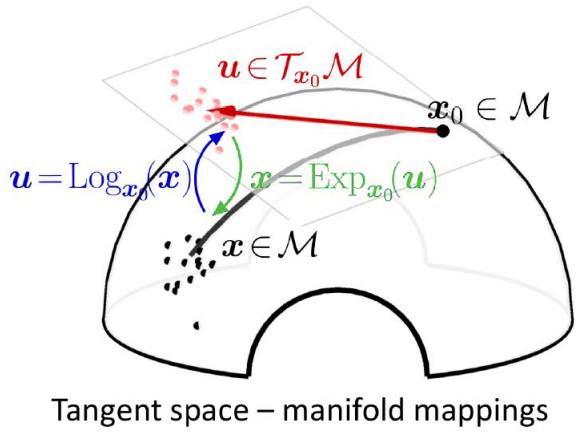


Figure 7: Visualization of the exponential and logarithmic maps. The exponential map sends tangent vectors at point x_0 to the end points x of the geodesic curves and the logarithmic map does the opposite [23].

2.2.2 The Grassmann and compact Stiefel manifold

The Grassmann manifold $\mathcal{G}(k,p)$ is the manifold in which each point represents a k-dimensional linear subspace in $\mathbb{R}^p$ and each element of $\mathcal{G}(k,p)$ is non-uniquely spanned by the columns of a full-rank orthonormal $p \times k$ matrix which constitute the Stiefel manifold $\mathcal{S}t(k,p)$ [24, 25]. The following example can be used to make this clear: $\mathcal{G}(1,p)$ is the set of all lines through the origin, while $\mathcal{S}t(1,p)$ is the set of all possible basis vectors, which can also be seen as a sphere. $\mathcal{G}(2,p)$ is the set of all planes while $\mathcal{S}t(2,p)$ is the set of all 2-frames in $\mathbb{R}^p$. A k-frame is a set of k orthonormal linearly independent vectors in a vector space. An example of this is shown in Figure 8. It is possible to define the exponential map on the entire tangent space if and only if the manifold is complete, which means that starting at any point p, a "straight" line can be drawn indefinitely along any direction. Therefore, in this research, the compact Stiefel manifold $\mathcal{S}t^c(k,p)$ is used, which is complete.

In the Grassmann manifold given by $\mathcal{G}(k,p)$, each point on the manifold is a subspace $\mathbf{m}$ and is represented by a $p \times k$ orthonormal matrix $\mathbf{Y}$. This is not a unique representation as any matrix $\mathbf{YP}$ in the set of orthogonal transformations of $\mathbf{Y}$, where $\mathbf{P} \in \mathcal{O}(k)$ are orthogonal square matrices of degree k, can represent the same subspace. This is given by:

$$\{\mathbf{YP},\ \mathbf{P} \in \mathcal{O}(k)\},\ \mathcal{O}(k) := \{\mathbf{P} \in \mathbb{R}^{k\times k},\ \mathbf{P}^T\mathbf{P} = I_k\}. \tag{34}$$

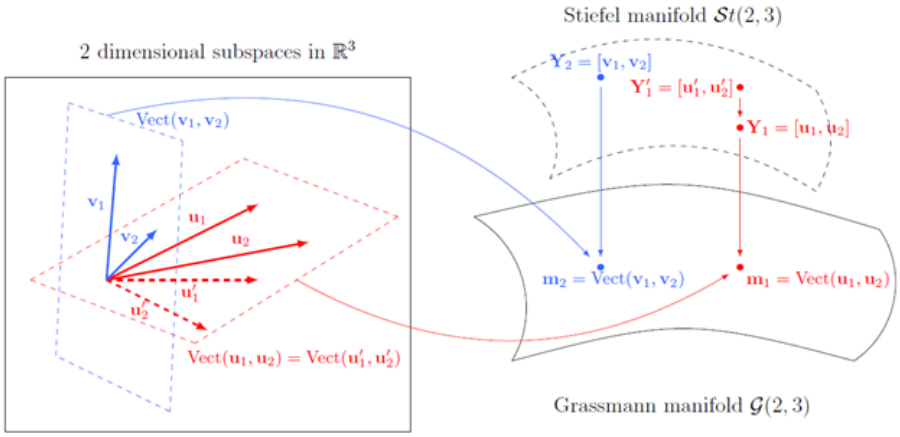


Figure 8: Points on the Grassmann and Stiefel manifold. The linearly independent basis vectors in $\mathbb{R}^3$ spanning the red and blue 2D planes of $\mathcal{G}(2,3)$ correspond to points in $\mathcal{S}t(2,3)$ [26].

An example of an orthogonal transformation is the rotation of the unit vectors $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ in the Cartesian plane. From this, a map π can be established for which it holds that for each subspace $\mathbf{m}$, there is a basis $\mathbf{Y}$. As π is a function that maps a basis $\mathbf{Y}$ to every subspace $\mathbf{m}$, π is a surjective function, which is given by:

$$\pi : \mathbf{Y} \in St^c(k,p) \mapsto \pi(\mathbf{Y}) = \mathbf{m} := \{\mathbf{YP},\ \mathbf{P} \in \mathcal{O}(k)\} \in \mathcal{G}(k,p). \tag{35}$$

The set of all matrices that represent the same point $\mathbf{m}$ is called the fiber of π at $\mathbf{m}$ and is given by:

$$\pi^{-1}(\mathbf{m}) = \{\mathbf{YP}, \quad \mathbf{P} \in \mathrm{O}(k)\}. \tag{36}$$

A visualization of a fiber on the Grassmann manifold is given in Figure 9.

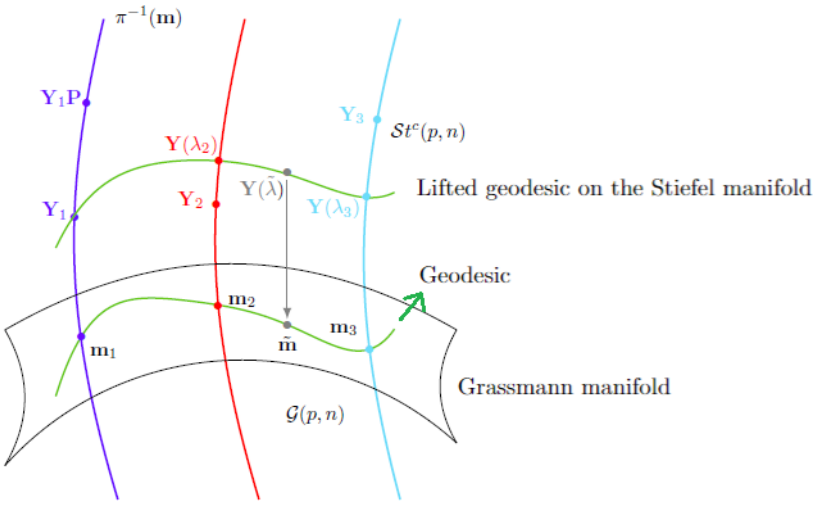


Figure 9: Visualization of fibers on the Grassmann and Stiefel manifold [25]. The lines defined by $\pi^{-1}(\mathbf{m})$ represent the fibers. The geodesic in the Grassmann manifold is lifted into the Stiefel manifold along the fibers.

The collection of all fibers at different points $\mathbf{m}$ on the manifold is called the fiber bundle, which is defined by $\mathcal{E}$. On each point on a fiber in the Stiefel manifold a tangent plane can be drawn. The collection of all these tangent planes is called a tangent bundle which is vector space that can be decomposed into an orthogonal direct sum as in Equation (29). The tangent vectors in this bundle

can be subdivided into a set of vectors that are tangent to the fibers and a set of vectors that are orthogonal to the tangent vectors to the fibers. The former set is called the vertical bundle and the latter is called the horizontal bundle, which are built up of vertical spaces and horizontal spaces respectively at the points on the fibers.

In mathematical terms, if $\pi : \mathcal{E} \to \mathcal{M}$ defines a fiber bundle over a smooth manifold $\mathcal{M}$ and on the fiber $\mathcal{E}_p$, going through point $p \in \mathcal{M}$, there is a point $e \in \mathcal{E}_p$ with $\pi(e) = p$, then the vertical space $\mathcal{V}_e$ at e is defined as a vector subspace of the tangent space to the fiber $\mathcal{E}_p$ at point $p = \pi(e)$. This vertical space contains tangent vectors to the fiber, which is given by $\mathcal{V}_e \subset T_e(\mathcal{E}_p)$. A horizontal space $\mathcal{H}_e$ is then defined by choosing a subspace of $\mathcal{T}_e(\mathcal{E}_p)$, such that $\mathcal{T}_e(\mathcal{E}_p)$ is the direct sum of $\mathcal{V}_e$ and $\mathcal{H}_e$ defined by:

$$\mathcal{T}_e(\mathcal{E}_p) = \mathcal{V}_e \oplus \mathcal{H}_e \in \mathbb{R}^n, \tag{37}$$

in which every vector in $\mathcal{V}_e$ is orthogonal to every vector in $\mathcal{H}_e$. The horizontal and vertical space are visualised in Figure 10.

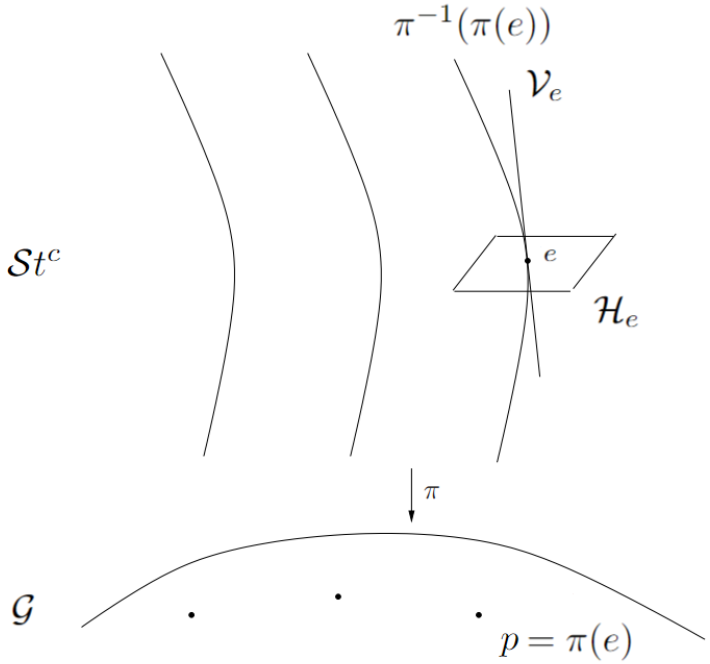


Figure 10: Visualization of the horizontal and vertical space in context of the Grassmann and Stiefel manifolds [27]. The horizontal space is the orthogonal complement to the vertical space, which is the tangent space along the fiber.

Let $c(t)$ be a curve in $\mathcal{M}$ that runs through the point $p = c(0)$. A lift of $c(t)$ through e is a curve $\tilde{c}(t)$ in $\mathcal{E}_p$ such that

$$\tilde{c}(0) = e, \text{ and } \pi(\tilde{c}(t)) = c(t). \tag{38}$$

So, a horizontal lift is defined as a lift in which every tangent velocity vector of the curve $\tilde{c}(t)$ lies in the horizontal space of the fiber $\mathcal{E}_p$:

$$\dot{\tilde{c}}(t) \in \mathcal{H}_{\tilde{c}(t)}. \tag{39}$$

The horizontal lift is path-dependent. Namely, when two smooth curves in $\mathcal{M}$ that coincide at $c_1(0) = c_2(0) = p$ and also intersect at another point $q \in \mathcal{M}$, are lifted horizontally to $\mathcal{E}$ through the same point on the fiber $e \in \pi^{-1}(p)$, then it holds that they will pass through different points

of $\pi^{-1}(q)$.

For the Grassmann manifold, the horizontal space is defined by:

$$\text{Hor}_Y := \{\mathbf{Z} \in \mathbb{R}^{p,k},\ \mathbf{Z}^T\mathbf{Y} = 0\}, \tag{40}$$

in which the matrices $\mathbf{Z}$ are called the horizontal lifts. To link the horizontal lift to the velocity vectors in the tangent space, we define the following properties of the horizontal space:

1. The tangent space at a point $\mathbf{m}$ given by $\mathcal{T}_{\mathbf{m}}\mathcal{G}(k,p)$ is isomorphic to any horizontal space $\text{Hor}_{\mathbf{Y}}$ with $\mathbf{Y}$ such that $\pi(\mathbf{Y}) = \mathbf{m}$. Isomorphic means that the topological spaces are topologically equivalent, meaning that they can be morphed into each other. The isomorphism is given by:

$$\mathrm{d}\pi_{\mathbf{Y}|\text{Hor}_{\mathbf{Y}}} : \text{Hor}_{\mathbf{Y}} \mapsto \mathcal{T}_{\mathbf{m}}\mathcal{G}(k,p), \tag{41}$$

   where $\mathrm{d}\pi_{\mathbf{Y}|\text{Hor}_{\mathbf{Y}}}$ is the directional derivative of the map π at $\mathbf{Y}$ in a certain direction in the horizontal space at $\mathbf{Y}$. This direction is related to the velocity along a lifted curve $\tilde{c}(t)$ given by $\frac{\mathrm{d}}{\mathrm{d}t}\big|_{t=0}(\pi(\tilde{c}(t)))$.

2. For any velocity vector $v \in \mathcal{T}_{\mathbf{m}}\mathcal{G}(k,p)$, there is a unique matrix $\mathbf{Z} \in \text{Hor}_{\mathbf{Y}}$, which is called the horizontal lift. This horizontal lift is linked to the velocity vector by the following mapping:

$$\mathrm{d}\pi_{\mathbf{Y}} \cdot \mathbf{Z} = v. \tag{42}$$

3. For any other orthogonal transformation $\mathbf{P} \in \mathcal{O}(k)$, $\mathbf{ZP}$ is another horizontal lift of v. This horizontal lift belongs to the vector space $\text{Hor}_{\mathbf{YP}}$ and is given by:

$$\mathrm{d}\pi_{\mathbf{YP}} \cdot \mathbf{ZP} = v. \tag{43}$$

To elaborate on these properties, in Figure 9, there is a direct connection between the geodesic in the Grassmann and Stiefel manifold via the fiber bundle. In Figure 10, this connection is established by the fact that the horizontal space is isomorphic to the tangent space of the Grassmann manifold. In essence, the horizontal lift can be regarded as the measure of how a curve on the Grassmann manifold is lifted into the horizontal space of a point e on a fiber that goes through a reference point on the Grassmann manifold. The tangents along this curve all lie in the horizontal space at e, meaning that the curve does not move in the directions defined by the vertical space at e. In essence, the lift of the geodesic on the Grassmann manifold to the horizontal space of a point on a fiber, leads to a unique curve of orthonormal matrices $\mathbf{Y}$, of which the velocity vector v is represented by $\mathbf{Z}$. By setting up logarithmic maps and exponential maps, transformations can be made between the $\mathbf{Z}$ and $\mathbf{Y}$ matrices.

First, a definition of a distance (Riemannian metric) on the Stiefel manifold is given by:

$$\langle v_1, v_2\rangle := \langle \mathbf{Z}_1, \mathbf{Z}_2\rangle_{\mathbf{Y}}, \tag{44}$$

in which the inner product is defined by:

$$\langle \mathbf{Z}_1, \mathbf{Z}_2\rangle := tr(\mathbf{Z}_1^T\mathbf{Z}_2^T),\ \mathbf{Z}_1, \mathbf{Z}_2 \in \mathbb{R}^{p,k}. \tag{45}$$

where Z_1 and Z_2 are both horizontal lifts of respectively velocity vectors v_1 and v_2. This inner product is essential for the exponential and logarithmic map as now a measure of distance is available to define unique geodesics. The geodesic equation that represents a unique minimal geodesic between points $\mathbf{m}_1$ and $\mathbf{m}_2$ on the Grassmann manifold is given by [24, 28]:

$$\ddot{\mathbf{Y}} + \mathbf{Y}\left(\dot{\mathbf{Y}}^T\dot{\mathbf{Y}}\right) = 0. \tag{46}$$

Next, through the geodesic equation, a logarithmic map can be established. Let $\mathbf{m}_0$ be a point on the Grassmann manifold where a tangent plane is drawn and $\mathbf{m}_1$ be another point close to $\mathbf{m}_0$.

These points have their respective bases $\mathbf{Y}_0 \in \mathbb{R}^{p\times k}$ and $\mathbf{Y}_1 \in \mathbb{R}^{p\times k}$. The logarithmic map can then be used to calculate the horizontal lift $\mathbf{Z}_1$ of a geodesic going from $\mathbf{m}_0$ to $\mathbf{m}_1$, which is defined by [26]:

$$\mathbf{Y}_1 \left(\mathbf{Y_0}^T \mathbf{Y}_1\right)^{-1} - \mathbf{Y_0} = \mathbf{\Phi}_1 \mathbf{\Omega}_1 \mathbf{\Psi}_1^T \quad \text{(SVD)}, \tag{47}$$

$$\mathbf{Z}_1 = \mathbf{\Phi}_1 \tan^{-1}(\mathbf{\Omega}_1)\, \mathbf{\Psi}_1^T. \tag{48}$$

where $\mathbf{\Phi}_1 \in \mathbb{R}^{p\times k}$ and $\mathbf{\Psi}_1 \in \mathbb{R}^{k\times k}$ are respectively the matrices with left and right singular vectors of the horizontal lifts $\mathbf{Z}_1$. The values in the diagonal matrix $\tan^{-1}(\mathbf{\Omega}_1) \in \mathbb{R}^{k\times k}$ are referred to as the Jordan's principal angles, which are a set of minimized angles between $\mathbf{Y}_0$ and $\mathbf{Y}_1$ that are invariant under any orthogonal transformation of $\mathbf{Y}_0$ and $\mathbf{Y}_1$. The exponential map $Exp_{\mathbf{m_0}}(v)$ can then be used to map the horizontal lift back to $\mathbf{Y}_{Exp,1}$, which is defined by:

$$\mathbf{Z}_1 = \mathbf{\Phi}_1\, \mathbf{\Theta}_1\, \mathbf{\Psi}_1^T \quad \text{(SVD)}, \tag{49}$$

$$\mathbf{Y}_{Exp,1} = [\mathbf{Y}_0 \mathbf{\Psi}_1 \cos(\mathbf{\Theta}_1) + \mathbf{\Phi}_1 \sin(\mathbf{\Theta}_1)]\mathbf{\Psi}_1^T. \tag{50}$$

where $\mathbf{\Theta}_1 = \tan^{-1}(\mathbf{\Omega}_1)$ and the principal angles are given by $\theta_1 \geq \ldots \geq \theta_k \geq 0$. It can be checked that the exponential map satisfies the geodesic equation. It should be noted that in general $\mathbf{Y}_{Exp,1} \neq \mathbf{Y}_1$. The reason for this is that the basis matrices representing a certain subspace are not unique, so there could be any other orthogonal transformation of $\mathbf{Y}$ that comes out of the Exponential map. Another important point to note is that the radius of the open ball in which the exponential map is a diffeomorphism, resulting in a valid inverse logarithmic map, is connected to the principal angles [24, 25]. This is because the principal angle is a measure for the length of the geodesic [29, 30]. The following holds for different values of the first principal angle:

(a) If the largest principal angle $\theta_1 < \pi/2$, then the geodesic is unique minimizing.

(b) If the largest principal angle $\theta_1 = \pi/2$, then the geodesic is non-unique minimizing.

(c) If the largest principal angle $\theta_1 > \pi/2$, then the geodesic is not minimizing.

2.3 Local basis interpolation

2.3.1 Interpolation on a tangent space to the Grassmann manifold (ITSGM)

The information in Section 2.2 can now be used to obtain an interpolation curve between points $\mathbf{m}_1, \ldots, \mathbf{m}_{N_\lambda}$ on the Grassmann manifold $\mathcal{G}(k,p)$, where each point $\mathbf{m}_i$ corresponds to a parameter combination $\boldsymbol{\lambda}_i$. The ITSGM algorithm is outlined as follows:

- First, a reference point $\mathbf{m}_0 \in \{\mathbf{m}_1, \ldots, \mathbf{m}_{N_\lambda}\}$ is chosen on which the tangent plane is drawn.
- Then, the logarithmic map $\text{Log}_{\mathbf{m}_0}$ is used to linearize around $\mathbf{m}_0$, in order to define velocity vectors $v_i := \text{Log}_{\mathbf{m}_0}(\mathbf{m}_i)$ on the tangent plane $\mathcal{T}_{\mathbf{m}_0}\mathcal{G}$, by calculating the horizontal lifts $\mathbf{Z}_i$ using the Logarithmic map in Equations (47) and (48). This leads to a set of parameter combinations $\boldsymbol{\lambda}$ and corresponding $\mathbf{Z}$ matrix pairs given by $\{\boldsymbol{\lambda}_i, \mathbf{Z}_i\}_{i=1}^{N_\lambda}$, where $\mathbf{Z}_i \in \mathbb{R}^{p\times k}$.
- An interpolated curve $\boldsymbol{\lambda} \mapsto v(\boldsymbol{\lambda})$ is set up between vectors v_i, by interpolating horizontal lifts $\mathbf{Z}_i$ using for instance Lagrangian polynomials or radial basis functions (RBF) resulting in:

$$v(\boldsymbol{\lambda}_i) = v_i, \quad \forall i = 1, \ldots, N. \tag{51}$$

- Using the exponential map $\text{Exp}_{\mathbf{m}_0}$, an interpolated curve between the curves running from $\mathbf{m}_0$ to $\mathbf{m}_i$ on $\mathcal{G}(p,k)$ can be converted back using:

$$\boldsymbol{\lambda} \mapsto \mathbf{m}(\boldsymbol{\lambda}) := \text{Exp}_{\mathbf{m}_0}(v(\boldsymbol{\lambda})), \tag{52}$$

  so that

$$\mathbf{m}(\boldsymbol{\lambda}_i) = \mathbf{m}_i, \quad \forall i = 1, \ldots, N. \tag{53}$$

Another but similar way to think about this problem is to define curves on the compact Stiefel manifold $\mathcal{S}t^c(k,p)$ instead of the ones defined on the Grassmann manifold $\mathcal{G}(k,p)$. The starting point is a set of orthonormal matrices $\mathbf{Y}_1, \ldots, \mathbf{Y}_{N_{\boldsymbol{\lambda}}} \in \mathbb{R}^{p\times k}$ in the compact Stiefel manifold $\mathcal{S}t^c(k,p)$, corresponding to parameter values $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_N$. Once a reference parameter value $\boldsymbol{\lambda}_0$ has been chosen, a curve is obtained:

$$\boldsymbol{\lambda} \mapsto \mathbf{Y}(\boldsymbol{\lambda}) \tag{54}$$

$$\mathbf{Y}(\boldsymbol{\lambda}_i) \neq \mathbf{Y}_i \tag{55}$$

As the exponential map gives back an orthogonally transformed version of the original orthogonal matrix, it holds that $\mathbf{Y}(\boldsymbol{\lambda}_i) \neq \mathbf{Y}_i$. So, such a curve will not be an interpolation curve between the matrices $\mathbf{Y}_1, \ldots, \mathbf{Y}_{N_{\boldsymbol{\lambda}}}$. For the interpolation method to work it is important that the first principal angle between $\mathbf{Y}_0$ and $\mathbf{Y}_1$ is lower than $\pi/2$, else the geodesic equation will not have a unique solution and the logarithmic map will not be valid. In short, a reference parameter combination is chosen on which to draw a tangent plane. Then the horizontal lifts $\mathbf{Z}_i$ are calculated using the Logarithmic map in Equations (47) and (48). The horizontal lifts are interpolated to a new parameter combination λ and the resulting horizontal lift $\mathbf{Z}(\lambda)$ is mapped back using the exponential map in Equations (49) and (50), resulting in an interpolated orthonormal matrix $\mathbf{Y}(\lambda)$.

2.3.2 Space-time coupled local basis interpolation

There are many different tangent space interpolation methods that have been developed in research for both only parameter and parameter-time dependent problems. For non-time dependent problems, one of the earliest research projects in which tangent space interpolation was applied, is a paper by D. Amsallem and C. Farhat [31], in which the ROMs (truncated basis matrices) of a high-fidelity aeroelastic computational model of an F-16 Block 40 aircraft were interpolated to new free-stream Mach numbers. In a paper by D.G. Giovanis and M.D. Shields [32], research was done in uncertainty quantification, in which Delaunay triangulation was used to refine the probability space into simplex elements. For every simplex, the high-dimensional solutions corresponding to its

vertices (sample points) are projected onto the Grassmann manifold and an approximation of the solution (ROM) within each element was obtained by interpolation on the Grassmann manifold. This method was applied to study the probability of shear band formation in a bulk metallic glass.

For time dependent problems, in a paper by S. Pawar [33], the interpolation of the basis matrices and the POD coefficients were separated, where the basis matrices were interpolated on the Grassmann manifold, while the POD coefficients were predicted using a long short term memory neural network that was trained in the offline stage. This model was tested on the one-dimensional Burgers equation and the two-dimensional vorticity transport equation. Combined space time interpolation was also applied in a paper by Y. Lu in [34], where both the bases $\mathbf{U}$ and coefficients $\mathbf{V}$ were interpolated on the Grassmann manifold and $\mathbf{\Sigma}$ matrices were interpolated using Lagrange polynomials. The interpolation method was combined with a sampling scheme that subdivided the parameter domain hierarchically into cuboids of which the vertices were then used to interpolate to the center of the cuboid at which the error was checked. More background information and methods on Grassmann interpolation schemes can be found in papers [35–39]. The interpolation method that is used in this research is based on [26] and uses a space-time coupling approach, which is explained in the next part.

Let $\mathbf{S}_i \in \mathbb{R}^{N_x \times N_t}$ be the snapshot matrices, representing a space and time-dependent physical field given by $[\mathbf{y}(\boldsymbol{\lambda}_i, t_1), \ldots, \mathbf{y}(\boldsymbol{\lambda}_i, t_{N_t})]$, for a particular parameter combination $\boldsymbol{\lambda}_i \in \mathbb{R}^{\delta}$, with $i = 1, \ldots, N_\lambda$. N_x is the number of spatially discretized points and N_t the number of points in time. The goal is now to:

1. Apply an SVD on all $\mathbf{S}_i$ given by Equation (6), resulting in orthogonal matrices with left singular vectors $\mathbf{U}_i \in \mathbb{R}^{N_x \times N_t}$, matrices with singular values $\mathbf{\Sigma}_i \in \mathbb{R}^{N_t \times N_t}$, and orthogonal matrices with right singular vectors $\mathbf{V}_i \in \mathbb{R}^{N_t \times N_t}$. The matrices are then truncated, where only the N_m left column vectors are kept in the matrices, leading to $\mathbf{U}_{i,tr}(\boldsymbol{\lambda}_i) \in \mathbb{R}^{N_x \times N_m}$, $\mathbf{\Sigma}_{i,tr} \in \mathbb{R}^{N_m \times N_m}$ and $\mathbf{V}_{i,tr} \in \mathbb{R}^{N_t \times N_m}$ and finally $\mathbf{S}_{i,tr}$.

2. The spatial and temporal bases $\mathbf{U}$ and $\mathbf{V}$ are only unique up to sign within the SVD, meaning that multiplying both respective basis vectors in $\mathbf{U}$ and $\mathbf{V}$ of the same order with a minus sign does not lead to change in the matrix product defined by the SVD. So, a unique matrix representation for orthonormal matrices $\tilde{\mathbf{U}}_{i,tr} \in \mathcal{S}t^c(N_m, N_x)$ and $\tilde{\mathbf{V}}_{i,tr} \in \mathcal{S}t^c(N_m, N_t)$ must be found. For clarity, the unique truncated matrices will just be denoted as $\mathbf{U}_i$ and $\mathbf{V}_i$.

3. Choose a parameter combination $\boldsymbol{\lambda}_0 \in \{\boldsymbol{\lambda}_i, \ldots, \boldsymbol{\lambda}_{N\lambda}\}$ on which a tangent plane is drawn. Use the ITSGM Algorithm to get interpolated curves:

$$\boldsymbol{\lambda} \mapsto \mathbf{U}(\boldsymbol{\lambda}), \quad \boldsymbol{\lambda} \mapsto \mathbf{V}(\boldsymbol{\lambda}) \tag{56}$$

   with $\mathbf{U}(\boldsymbol{\lambda}_i) \neq \mathbf{U}_i$ and $\mathbf{V}(\boldsymbol{\lambda}_i) \neq \mathbf{V}_i$. The $\mathbf{Z}$ matrices of $\mathbf{U}$ and $\mathbf{V}$ are respectively $\mathbf{Z}^U$ and $\mathbf{Z}^V$. It is important to note that the number of modes is fixed for a certain tangent plane, as the dimension of the manifolds are linked to the number of modes.

4. Find an interpolation curve $\boldsymbol{\lambda} \mapsto \mathbf{S}(\boldsymbol{\lambda})$ between matrices $\mathbf{S}_{1,tr}, \ldots, \mathbf{S}_{\mathrm{N}_\lambda,tr}$, using curves obtained by Equation (56). The interpolated snapshot matrix will be denoted by $\mathbf{S}_{int}(\boldsymbol{\lambda})$ with physical field vectors given by $[\mathbf{y}_{int}(\boldsymbol{\lambda}_i, t_1), \ldots, \mathbf{y}_{int}(\boldsymbol{\lambda}_i, t_{N_t})]$.

Point 2 can be solved by making an intrinsic choice on the orientation of the vectors in the spatial and temporal bases. First, any couple of vectors $(\mathbf{u}, \mathbf{v})$ is defined modulo ± 1, and $\mathbf{v}$ is obtained in a unique way from $\mathbf{u}$. A choice of orientation is then made by taking the first column vector $\mathbf{y}$ in $\mathbf{S} = [\mathbf{y}_1, \ldots, \mathbf{y}_{N_t}]$ for which holds that the scalar product $\langle \mathbf{y}, \mathbf{u} \rangle$ is non zero and the sign that is imposed is the sign of that same scalar product. This so called oriented SVD is summarized as follows:

1. Compute an SVD of $\mathbf{S}$, truncate and obtain spatial unit vectors $\mathbf{u}_1, \ldots, \mathbf{u}_{N_m}$ and temporal unit vectors $\mathbf{v}_1, \ldots, \mathbf{v}_{N_m}$.

2. Consider the column vectors $\mathbf{y}_1, \ldots, \mathbf{y}_{N_t}$ of $\mathbf{S}$.

3. For $i = 1, \ldots, N_m$ define
$$\varepsilon_i := \frac{\langle \mathbf{y}(\mathbf{u}_i), \mathbf{u}_i \rangle}{|\langle \mathbf{y}(\mathbf{u}_i), \mathbf{u}_i \rangle|}, \tag{57}$$
where $\mathbf{y}(\mathbf{u}_i)$ is the first column vector of $\mathbf{S}$ such that $\langle \mathbf{u}_i, \mathbf{y} \rangle \neq 0$.

4. For $i = 1, \ldots, N_m$, make sign replacement
$$\mathbf{u}_i \leftarrow \varepsilon_i \mathbf{u}_i, \quad \mathbf{v}_i \leftarrow \varepsilon_i \mathbf{v}_i.$$

Finding an interpolation curve for the full solution $\boldsymbol{\lambda} \mapsto \mathbf{S}(\boldsymbol{\lambda})$, is also still a problem. Recall that the SVD is a product of the $\mathbf{U}$, $\boldsymbol{\Sigma}$ and $\mathbf{V}$ matrices. As the interpolation of $\mathbf{U}$ and $\mathbf{V}$ are done separately and it holds that $\mathbf{U}(\boldsymbol{\lambda}_i) \neq \mathbf{U}_i$ and $\mathbf{V}(\boldsymbol{\lambda}_i) \neq \mathbf{V}_i$, it is not guaranteed that the product $\mathbf{U}(\boldsymbol{\lambda}_i)\boldsymbol{\Sigma}_i\mathbf{V}(\boldsymbol{\lambda}_i)$ will give back the original matrix $\mathbf{S}_{i,r}$. Therefore, an extra step is being done where the $\boldsymbol{\Sigma}$ matrices are recalculated by projecting the $\mathbf{S}_{i,r}$ matrices on $\mathbf{U}(\boldsymbol{\lambda}_i)$ and $\mathbf{V}(\boldsymbol{\lambda}_i)$ that come out of the exponential maps, which gives leads to the following coupling matrix:

$$\mathbf{C}_i = \mathbf{U}(\boldsymbol{\lambda}_i)^T \mathbf{S}_{i,tr} \mathbf{V}(\boldsymbol{\lambda}_i), \tag{58}$$

which leads to the following interpolation curve:

$$\boldsymbol{\lambda} \mapsto \mathbf{C}(\boldsymbol{\lambda}) \in \mathbb{R}^{N_m \times N_m}. \tag{59}$$

The final interpolation curve for $\mathbf{S}$ is then defined by:

$$\lambda \mapsto \mathbf{S}(\boldsymbol{\lambda}) := \mathbf{U}(\boldsymbol{\lambda})\mathbf{C}(\boldsymbol{\lambda})\mathbf{V}(\boldsymbol{\lambda})^T \tag{60}$$

2.3.3 Radial basis function (RBF) interpolation

To clarify, 3 separate interpolators are built for the matrices $\mathbf{Z}^{\mathbf{U}}$, $\mathbf{C}$ and $\mathbf{Z}^{\mathbf{V}}$. It is important to note that in the space-time coupled approach, all the values in the $\mathbf{Z}$ and $\mathbf{C}$ matrices will have to be interpolated. There are two ways to go about this, either entry-by-entry interpolation or full matrix interpolation via a linear combination of weights. The problem with entry by entry interpolation is that first of all, for systems that have a very large spatial discretization, which means that N_x is large, it is very impractical and expensive to interpolate the $\mathbf{Z}^U$ matrices that have the same dimensions as the $\mathbf{U} \in \mathbb{R}^{N_x \times N_m}$. Secondly, it is theoretically possible with entry-by-entry interpolation that the interpolant leaves the tangent space, even if all sample points are contained in it. This can be checked by calculating the tangency constraint of the horizontal space $\mathbf{Z}^T\mathbf{Y} = 0$. Therefore, in this research RBF interpolation was applied to interpolate full matrices.

There are multiple ways to interpolate full matrices instead of single coefficients. In [7] and [40], multivariate Lagrange polynomials and inverse distance weighting were used as interpolation methods. However a more accurate approach to interpolation would be to use radial basis functions (RBF) [41]: Given N_λ sample parameter combinations defined by $\boldsymbol{\lambda}_i$, with corresponding matrices $\mathbf{Q}_i$, where $i = 1, \ldots, N_\lambda$. The RBF interpolant at a new parameter combination $\boldsymbol{\lambda}^*$ then reads:

$$\hat{\mathbf{Q}}(\boldsymbol{\lambda}^*) = (\mathbf{Q}_1, \ldots, \mathbf{Q}_{N_\lambda}) \mathbf{D}^{-1} \mathbf{d}(\boldsymbol{\lambda}^*). \tag{61}$$

Here, $\mathbf{D} \in \mathbb{R}^{N_\lambda \times N_\lambda}$ is the quadratic radial distance matrix with entries $D_{i,j} = \text{rbf}(\|\boldsymbol{\lambda}_i - \boldsymbol{\lambda}_j\|), i, j = 1, \ldots, r$, and $\mathbf{d}(\boldsymbol{\lambda}^*)$ is a radial distance vector $\mathbf{d}(\boldsymbol{\lambda}^*) = (\text{rbf}(\|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}^*\|), \ldots, \text{rbf}(\|\boldsymbol{\lambda}_{N_\lambda} - \boldsymbol{\lambda}^*\|))^T \in \mathbb{R}^r$. The radial basis function is denoted by rbf: $\mathbb{R}_{\geq 0} \to \mathbb{R}$ and common choices of RBFs are:

	linear	cubic	multiquadric	Gaussian	thin plate spline
$\text{rbf}(x)$	x	x^3	$\sqrt{1+x^2}$	$\exp(-x^2)$	$x^2 \log(x)$

The RBF weights vector is then given by $\boldsymbol{\omega}(\boldsymbol{\lambda}^*) := \mathbf{D}^{-1}\mathbf{d}(\boldsymbol{\lambda}^*)$, which can then be used in the RBF interpolant as coefficients in a linear combination with the sample matrices. This linear combination is given by:

$$\hat{\mathbf{Q}}\left(\boldsymbol{\lambda}^*\right) = \sum_{i=1}^{N_\lambda} \omega_i\left(\boldsymbol{\lambda}^*\right) \mathbf{Q}(\boldsymbol{\lambda}_i). \tag{62}$$

In this research the multiquadric RBF was used primarily. All information is now available to build a locally adaptive algorithm based on sparse grids that uses the local basis interpolation method.

2.4 Local basis interpolation on sparse grids

There are 2 important issues of the local interpolation method that must be resolved. The first issue is the combination of a fixed number of modes that can be used per tangent plane and a changing number of modes needed for a certain approximation accuracy over the parameter domain. The problem is that if only one tangent plane were to be used over the full parameter domain, the number of modes used at the tangent plane will be too low or too high in a certain part of the domain. If the number is too low, then there are not enough modes to approximate the solution with a high enough accuracy, while if the number is too high, higher-order modes are included in the ROM that represent noise. Another issue is that for the interpolation method to work properly, the principal angle must stay below $\pi/2$, meaning that the matrices in $\mathbf{U}_i$ (and $\mathbf{V}_i$) should be related to each other to a certain degree. A simple example of this is a laminar flow versus a turbulent flow, where the modes can change drastically from one flow domain to another. Moreover, the number of modes to represent these flow domains will also be completely different. A turbulent flow is much more complex than a laminar flow, meaning that more modes need to be used to represent this turbulent flow. To resolve this issue, the algorithm should be able to adaptively draw new tangent planes. When an interpolation is done to a new test point, only the closest available tangent plane must be used for the interpolation. This leads to a so-called Voronoi diagram where the domain is split up in subdomains where all the points in a subdomain have the same closest point:

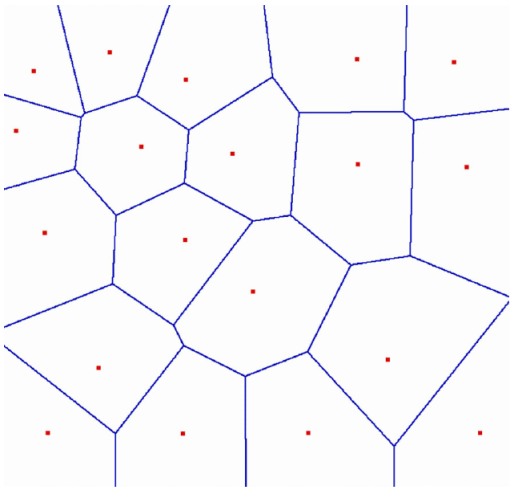


Figure 11: Example of a Voronoi diagram in which the domain is split up in subdomains where all the points in a subdomain have the same closest point [42].

In essence, the algorithm should be able to split up the domain in subdomains, depending on how the tangent planes are placed, and use the tangent plane assigned to a certain subdomain when using ITSGM within that domain. New tangent planes are added if the number of modes is not compatible with the previous closest tangent plane or if the first principal angle between the basis at the tangent plane and a point in the respective subdomain is too high. In the next part a rundown of the algorithm will be given as an example to get a better understanding of how the algorithm works.

The algorithm starts of with iteration $k = 1$ by sampling the first point $[0.5; 0.5]$ in the middle of the domain and draws the first tangent plane on that point. This first point is then added to the set of points with tangent planes $\mathcal{TP} = \{\boldsymbol{\lambda}_\xi | \xi = 1, \ldots, N_{tp}\}$, where N_{tp} is the number of tangent planes. Then this point is added to the set of all points $\mathcal{X}^1$ and to the set of important points $\mathcal{Z}^1 = \{\boldsymbol{\lambda}_\zeta | \zeta = 1, \ldots, N_z\}$, where N_z is the number of important points. The error that is used to determine how many modes are needed for a certain approximation accuracy is based on the approximation of the single physical field vectors, instead of the full time evolution of the physical field. So, the number of modes that is chosen for a tangent plane will be based on the following error criteria:

$$\epsilon_{\xi,\tau}^{N^*_{m,\xi}} = \frac{\|\mathbf{y}_{true}(\boldsymbol{\lambda}_\xi, t_\tau) - \mathbf{y}^{N^*_{m,\xi}}(\boldsymbol{\lambda}_\xi, t_\tau)\|_{L_2}}{(\|\mathbf{y}(\boldsymbol{\lambda}_\xi, t_\tau)\|_{L_2} + \eta)}, \quad \tau = 1, \ldots, N_t, \tag{63}$$

where $\mathbf{y}^{N^*_{m,\xi}}(\boldsymbol{\lambda}_\xi, t_\tau)$ are the approximated physical field vectors in matrices $\mathbf{S}_i^{N^*_{m,\xi}}$ with $N^*_{m,\xi}$ modes at a parameter combination $\boldsymbol{\lambda}_\xi$ in the set $\mathcal{TP}$. η is an offset that deals with state vectors that have near zero magnitude. The number of possible modes $\mathbf{N}^*_{m,\xi}$ for the tangent plane at $\boldsymbol{\lambda}_\xi$ is then given by:

$$\mathbf{N}^*_{m,\xi} = \{N^*_{m,\xi} | \epsilon_{\xi,\tau}^{N^*_{m,\xi}} < \epsilon_{tol,max} \ \forall \tau \ \wedge \max(\epsilon_{\xi,\tau}^{N^*_{m,\xi}}) > \epsilon_{tol,min}\}, \tag{64}$$

where the errors for all time steps should be below the maximum error tolerance $\epsilon_{tol,max}$ and the maximum error over all time steps should be above the minimum error tolerance $\epsilon_{tol,min}$. The reason for the minimum error tolerance is to not include modes that represent numerical noise. Then the number of modes for tangent plane $\xi = 1$ is given by $N_{m,1} = \mathbf{N}^*_{m,1}(\lceil \frac{\text{end}}{2} \rceil)$, where end specifies the last index of the array $\mathbf{N}^*_{m,1}$. Then in iteration $k = 2$, the forward points are sampled based on point $[0.5; 0.5]$, which are added to $\mathcal{X}^2$ and the new points are checked as a test set $\mathcal{T}^2$. A visualization of the beginning of the sparse grids is given below:

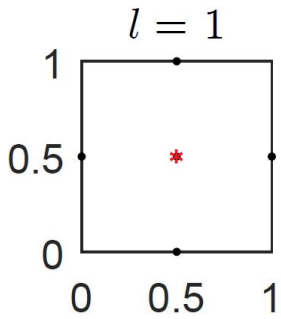


Figure 12: Second iteration of the sparse grid with a tangent plane (star) in the middle at $[0.5; 0.5]$ and points of the test set $\mathcal{T}^2$ given in black.

The next step is determining what the closest tangent plane is of the points in the testing set $\mathcal{T}^2$. The closest tangent plane is simply calculated based on Euclidean distance in the parameter domain. In this case, there is only one tangent plane $[0.5; 0.5]$, which results in a set of important points $\mathcal{C}_{test,1} = \{\boldsymbol{\lambda}_{\psi,1} | \psi = 1, \ldots, N_{test,1}\} \subset \mathcal{T}^2$ where $N_{test,1}$ is the amount of test points with the same closest tangent plane with index $\xi = 1$. As the RBF interpolator does not work with a single sample point, the testing set is immediately included into the important points $\mathcal{Z}^2$ and results in a set of training points with the same closest tangent plane $\mathcal{C}_{train,1} = \{\boldsymbol{\lambda}_{\chi,1} | \chi = 1, \ldots, N_{train,1}\} \subset \mathcal{Z}^2$, where $N_{test,1}$ is the amount of training points with the same closest tangent plane with index $\xi = 1$. This results in the a distribution of points in Figure 13, where the color of the points matches its respective tangent plane.

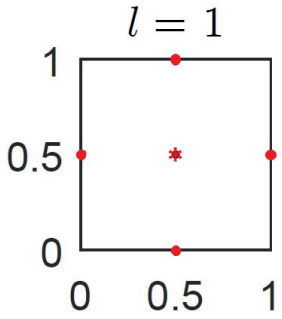


Figure 13: Second iteration of the sparse grid with the first tangent plane domain $\mathcal{C}_{train,1}$ defined in red.

The next step is determining for the points in $\mathcal{C}_{train,1}$ whether the number of modes $N_{m,1}$, used at their closest tangent plane, leads to an accurate approximation of their own time evolutions. This is done by again checking the same error in Equation (63). If the number of modes is not compatible for a set of points $\mathcal{R} = \{\boldsymbol{\lambda}_\rho | \rho = 1, \ldots, N_r\} \subset \mathcal{C}_{train,1}$, then new tangent planes must be added. Afterwards, it is also checked if the first principal angles between the subspaces at the tangent plane and the points in $\mathcal{C}_{train,1}$ are lower than a tolerance $\theta_{1,tol}$. If the first principal angle is not compatible for a set of points $\mathcal{R} \subset \mathcal{C}_{train,1}$, then again new tangent planes must be added in a hierarchical way. To check the first principal angles, the matrices $\mathbf{Z}^U_{\zeta,\xi}(\boldsymbol{\lambda}_\zeta)$, $\mathbf{Z}^V_{\zeta,\xi}(\boldsymbol{\lambda}_\zeta)$ and $\mathbf{C}_{\zeta,\xi}(\boldsymbol{\lambda}_\zeta)$ are calculated for all important points with index ζ and tangent planes, where in this case there is only one tangent plane with index $\xi = 1$. The steps that are taken when new tangent planes are added are given in Algorithm 1.

For this example, it is assumed that no new tangent planes were added in this iteration. Then in iteration $k = 3$, first the forward points are calculated based on the important points $\mathcal{Z}^2$ resulting in the testing set $\mathcal{T}^3$, and their closest tangent plane is determined, which leads to the situation in Figure 14.

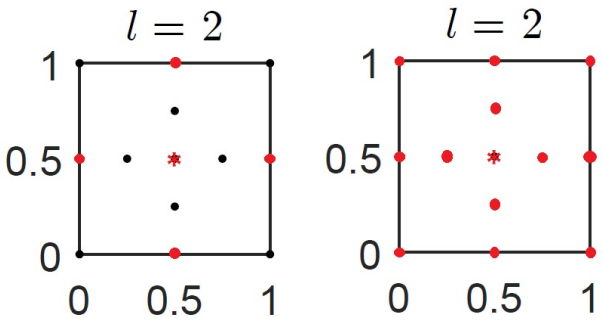


Figure 14: Third iteration of the sparse grid where the closest tangent plane is determined for the new points in the test set $\mathcal{T}^3$, which are colored in black on the left.

Then the closest tangent planes of the test points are calculated. As there is only one tangent plane there is only one set of points $\mathcal{C}_{test,1}$ with one closest tangent plane. The matrices $\mathbf{Z}^U_{\zeta,1}(\boldsymbol{\lambda}_\zeta)$, $\mathbf{Z}^V_{\zeta,1}(\boldsymbol{\lambda}_\zeta)$ and $\mathbf{C}_{\zeta,1}(\boldsymbol{\lambda}_\zeta)$ of the points in $\mathcal{C}_{train,1}$ are then interpolated to the points in $\mathcal{C}_{test,1} \subset \mathcal{T}^3$ and the time evolutions $\mathbf{S}_{int}(\boldsymbol{\lambda}_r)$ with state vectors $\mathbf{y}_{int}(\boldsymbol{\lambda}_r, t_\tau)$ are reconstructed. The errors are then calculated according to Equations (25), (26) and (27). Now, as an example, it is assumed that for some of the newly added important points, there is an incompatibility with the closest tangent plane. As a result, new tangent planes are added according to Algorithm 1 which leads to a distribution of tangent planes with a subsequent division of the parameter domain, which is shown in Figure 15.

Algorithm 1: Addition of new tangent planes.

Input: All sampled points $\mathcal{X}$, important points $\mathcal{Z}^k$, unimportant points $\mathcal{I}^{k-1}$, points $\boldsymbol{\lambda}_\rho \in \mathcal{R}$ that are incompatible in case of: (A) incompatible modes or (B) high first principal angle θ_1, the current set of tangent planes $\mathcal{TP}$, error tolerances $\epsilon_{tol,min}$ and $\epsilon_{tol,max}$.

Output: the new set of tangent planes $\mathcal{TP}$, the new set of important points $\mathcal{Z}$

1 **Initialize:** $\mathcal{TP}_{new} = \{\varnothing\}$, $\mathcal{R} = \{\varnothing\}$

2 **while** $\mathcal{R} \neq \{\varnothing\}$ **do**

3 **for** $\boldsymbol{\lambda}_\rho \in \mathcal{R}$ **do**

4 Calculate the ancestors of $\boldsymbol{\lambda}_\rho$ from $\mathcal{X}$ using Equation (23).; Include ancestors with highest ancestry without a tangent plane to $\mathcal{TP}_{new} = \{\boldsymbol{\lambda}_{\xi_{new}} | \xi_{new} = 1, \dots, N_{tp,new}\}$.

5 For the new points with coordinates $\boldsymbol{\lambda}_{\xi_{new}} \in \mathcal{TP}_{new}$, calculate the required number of modes $N_{m,\xi_{new}}$ with Equations (63) and (64).

6 **end**

7 $\mathcal{R} = \{\varnothing\}$

8 $\mathcal{TP} = \mathcal{TP} \cup \mathcal{TP}_{new}$

9 Find unimportant points with new tangent plane: $\mathcal{Z}^{new} = \{\boldsymbol{\lambda}_{\xi_{new}} \in \mathcal{I}^{k-1}\}$.

10 $\mathcal{Z}^k = \mathcal{Z}^k \cup \mathcal{Z}^{new}$, $\mathcal{I}^{k-1} = \mathcal{I}^{k-1} / \mathcal{Z}^{new}$

11 Calculate new sets of points with the same closest tangent plane $\mathcal{C}_{train,\xi} = \{\boldsymbol{\lambda}_{\chi,\xi} | \chi = 1, \dots, N_{train,\xi}\}, \forall \xi$.

12 In case (B): Calculate the $\mathbf{Z}^U_{\zeta,\xi}(\boldsymbol{\lambda}_\zeta)$, $\mathbf{Z}^V_{\zeta,\xi}(\boldsymbol{\lambda}_\zeta)$ and $\mathbf{C}_{\zeta,\xi}(\boldsymbol{\lambda}_\zeta)$ matrices.

13 Calculate new possible incompatible points $\mathcal{R}$ that are in $\mathcal{Z}^k$.

14 Case (A): Evaluate the following error:

$$\epsilon_{\chi,\xi,\tau} = \frac{\|\mathbf{y}_{true}(\boldsymbol{\lambda}_{\chi,\xi}, t_\tau) - \mathbf{y}^{N_{m,\xi}}(\boldsymbol{\lambda}_{\chi,\xi}, t_\tau)\|_{L_2}}{(\|\mathbf{y}_{true}(\boldsymbol{\lambda}_{\chi,\xi}, t_\tau)\|_{L_2} + \eta)}, \quad \tau = 1, \dots, N_t, \ \forall \boldsymbol{\lambda}_{\chi,\xi} \in \mathcal{C}_{train,\xi}, \forall \xi \tag{65}$$

where $N_{m,\xi}$ is the number of modes that a point is approximated with, which is equal to the number of modes of the closest tangent plane. Then $\mathcal{R}$ is given by:

$$\mathcal{R} = \{\boldsymbol{\lambda}_{\chi,\xi} \in \mathcal{Z}^k / \mathcal{TP} \mid \exists \tau (\epsilon^{N_{m,\xi}}_{\chi,\xi,\tau} > \epsilon_{tol,max}) \ \wedge \max(\epsilon^{N_{m,\xi}}_{\chi,\xi,\tau}) < \epsilon_{tol,min}\}. \tag{66}$$

Case (B): $\mathcal{R}$ is given by:

$$\mathcal{R} = \{\boldsymbol{\lambda}_{\chi,\xi} \in \mathcal{Z} / \mathcal{TP} \mid \theta_{1,\chi,\xi} > \theta_{1,tol}\}. \tag{67}$$

where θ_1 is calculated using the Logarithmic map in Equation (48) in line 13.

15 **end**

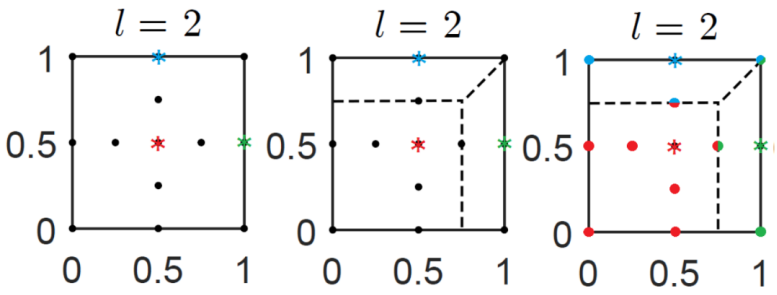


Figure 15: Third iteration of the sparse grid in which 2 new tangent planes are drawn, resulting in 3 different subdomains with their respective tangent planes given by the colour red, blue and green. The black dashed lines represent the boundaries of the subdomains similar to the Voronoi diagram.

This leads to sets $\mathcal{C}_{train,1}$, $\mathcal{C}_{train,2}$ and $\mathcal{C}_{train,3}$ with new matrices $\mathbf{Z}^U_{\zeta,\xi}(\boldsymbol{\lambda}_\zeta)$, $\mathbf{Z}^V_{\zeta,\xi}(\boldsymbol{\lambda}_\zeta)$ and $\mathbf{C}_{\zeta,\xi}(\boldsymbol{\lambda}_\zeta)$.

It can be observed that there are points that have 2 closest tangent planes at the same time. If there is a point $\boldsymbol{\lambda}_{\xi_{new}}$ in Algorithm 1 with multiple closest tangent planes, then the compatibility is checked separately for each tangent plane. In a case where a test point is on the edge of a parameter subdomain, the interpolation on that point will be carried out using the tangent plane with the highest number of modes. In iteration $k = 4$, the sparse grids look as follows in Figure 16.

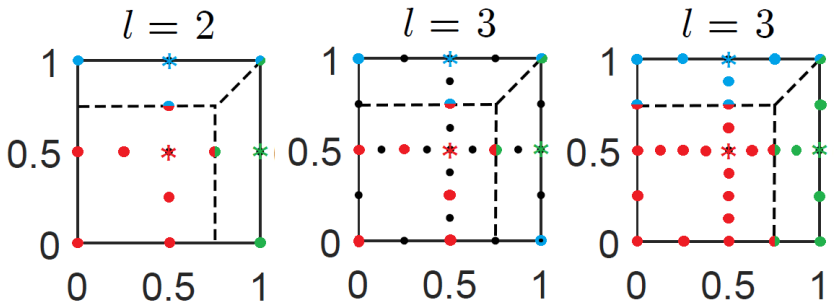


Figure 16: Fourth iteration of the sparse grid where the closest tangent plane is determined for the new points in test set $\mathcal{T}^4$, which are colored in black in the middle. Points on the boundaries of the subdomains have multiple colors.

Specifically for the RBF interpolator, it is very important to consider how the training samples are distributed. When interpolating to a new testing point, for instance $[0.5; 0.875]$ in the blue subdomain, the algorithm takes all the important points in $\mathcal{Z}^3$ that are in and on the boundary of the blue subdomain as training points that are used in Equation (62). This is specifically done to reduce the amount of SVD calculations that have to be done with the logarithmic mapping. However, this definition of the training points leads to a problem for points such as $[0; 0.75]$. As this point is on the boundary of the subdomain and is not 'surrounded' by training points, due to the nature of the RBF interpolator, the interpolation accuracy at this point will be suboptimal. Therefore, important points of neighbouring tangent planes are also included as training points for the target subdomain. This essentially leads to overlapping subdomains. As the refining of tangent planes follows the sparse grid hierarchy there is a straightforward way to calculate the neighbouring tangent planes. This is done as laid out in Algorithm 2.

The overlap can also be chosen larger by choosing the coordinates of the second closest neighbour or even third closest neighbour. The training points for a subdomain are then picked by finding all important points that are within the subdomain ranges, which leads to the square or cuboid subdomains for the training points, while the subdomains for the test points (to where an interpolation is done), are still defined following the Voronoi diagram. This will result in all testing points being 'surrounded' by training points leading to optimal interpolation performance. After applying Algorithm 2 the sets $\mathcal{C}_{train,1}$, $\mathcal{C}_{train,2}$ and $\mathcal{C}_{train,3}$ are updated and the new matrices $\mathbf{Z}^U_{\zeta,\xi}(\boldsymbol{\lambda}_\zeta)$, $\mathbf{Z}^V_{\zeta,\xi}(\boldsymbol{\lambda}_\zeta)$ and $\mathbf{C}_{\zeta,\xi}(\boldsymbol{\lambda}_\zeta)$ are calculated, which results in the distributions of training points in Figure 17.

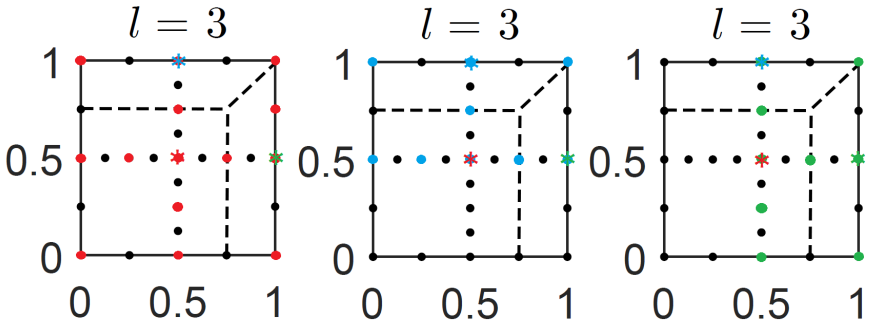


Figure 17: Fourth iteration of the sparse grid where the training points are shown for each tangent plane as a results of the overlap between subdomains.

Algorithm 2: Overlap of subdomains

Input: The set of tangent planes $\mathcal{TP}$, all important points $\mathcal{Z}^k$, index ν representing the ν^{th} neighbouring tangent plane closest to another tangent plane.

Output: The parameter subdomain ranges for each dimension δ for each tangent plane in $\mathcal{TP}$ denoted by $[\boldsymbol{\lambda}_{left}, \boldsymbol{\lambda}_{right}]_\xi$, where $\boldsymbol{\lambda}_{left}$ and $\boldsymbol{\lambda}_{right}$ are a δ-dimensional vector with each dimension containing the coordinate of the respectively ν^{th} left and right closest neighbour in that dimension.

```
for λ_ξ ∈ TP do
    for o = 1 : δ do
        Find all tangent planes with all coordinates the same expect for dimension o.
        Find ν^th left and right closest neighbour tangent planes with coordinates
         λ_left,neighbour and λ_right,neighbour.
        if λ_left = {∅} then
            λ_left,ξ,o = 0
        else
            λ_left,ξ,o = λ_left,neighbour,o
        end
        if λ_right = {∅} then
            λ_left,ξ,o = 1
        else
            λ_right,ξ,o = λ_right,neighbour,o
        end
    end
end
```

For the points in $\mathcal{T}^4$, the closest tangent planes are determined which results in sets $\mathcal{C}_{test,1}$, $\mathcal{C}_{test,2}$ and $\mathcal{C}_{test,3}$. The matrices $\mathbf{Z}^U_{\zeta,\xi}(\boldsymbol{\lambda}_\zeta)$, $\mathbf{Z}^V_{\zeta,\xi}(\boldsymbol{\lambda}_\zeta)$ and $\mathbf{C}_{\zeta,\xi}(\boldsymbol{\lambda}_\zeta)$ in sets $\mathcal{C}_{train,1}$, $\mathcal{C}_{train,2}$ and $\mathcal{C}_{train,3}$ are then interpolated to the points in $\mathcal{C}_{test,1}$, $\mathcal{C}_{test,2}$ and $\mathcal{C}_{test,3}$ respectively, which leads to the distribution in Figure 18.

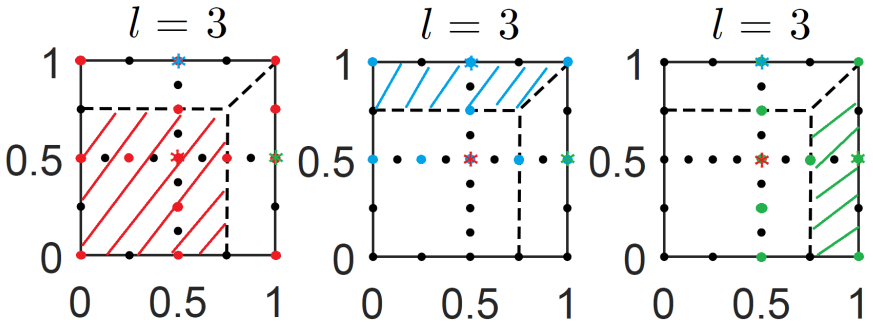


Figure 18: Fourth iteration of the sparse grid where the training and test points are shown for each subdomain.

For instance, an interpolation of the training data at the red points is done to all the black points in the red subdomain and the interpolation error is checked at these test points according to Equations (25), (26) and (27). This is also done similarly for the rest of the subdomains. In the subsequent iterations of the algorithm all preceding steps are repeated until all the errors at the new test points are below the error tolerance γ_{int}. To summarize and finalize Chapter 2, the full algorithm is described in Algorithm 3.

Algorithm 3: The Locally adaptive sampling algorithm based on local basis interpolation.

Input: Interpolation accuracy threshold γ_{int}, minimum and maximum error tolerance for the number of modes $\epsilon_{tol,min}$ and $\epsilon_{tol,max}$, the number of neighbours ν, the first principal angle tolerance $\theta_{1,tol}$, the greediness parameter μ.

Output: The grid points $\mathcal{X}^k$, the important points $\mathcal{Z}^k$ denoted with index ζ, the points with tangent planes $\mathcal{TP}^k$ denoted with index ξ, number of modes at each tangent plane $N_{m,\xi}$, the horizontal lifts $\mathbf{Z}^U_{\zeta,\xi}$, $\mathbf{Z}^V_{\zeta,\xi}$ and the coupling matrices $\mathbf{C}_{\zeta,\xi}$ of the important points,

1 **Initialize:** $k = 1$, $\boldsymbol{\lambda}_{nom} = (0.5, \ldots, 0.5)$, $\mathcal{X}^1 = \mathcal{Z}^1 = \{\boldsymbol{\lambda}_{nom}\}$, oriented SVD: $\mathbf{U}(\boldsymbol{\lambda}_{nom})$, $\boldsymbol{\Sigma}(\boldsymbol{\lambda}_{nom})$, $\mathbf{V}(\boldsymbol{\lambda}_{nom})$.

2 Calculate number of modes $N_{m,1}$ needed to approximate $\mathbf{S}(\boldsymbol{\lambda}_{nom})$ using Equations (63) and (64) and add to $\mathcal{TP}$.

3 Calculate $\mathbf{Z}^U_{1,1} \in \mathbb{R}^{N_x \times N_{m,1}}$, $\mathbf{Z}^V_{1,1} \in \mathbb{R}^{N_t \times N_{m,1}}$ and $\mathbf{C}_{1,1} \in \mathbb{R}^{N_{m,1} \times N_{m,1}}$ using the Logarithmic map from Equations (47) and (48).

4 $k = 2$, sample forward points and form test set $\mathcal{T}^2 = F(\mathcal{Z}^1)$.

5 Calculate $\mathbf{S}(\boldsymbol{\lambda}_r)$ and apply oriented SVDs: $\mathbf{U}(\boldsymbol{\lambda}_r)$, $\boldsymbol{\Sigma}(\boldsymbol{\lambda}_r)$ and $\mathbf{V}(\boldsymbol{\lambda}_r)$, $\forall \boldsymbol{\lambda}_r \in \mathcal{T}^2$.

6 $\mathcal{Z}^2 = \mathcal{Z}^1 \cup \mathcal{T}^2$.

7 Check compatibility of modes for $\mathcal{Z}^2$ and draw new tangent planes using Algorithm 1A if incompatibility.

8 Calculate point sets with same closest tangent plane from $\mathcal{Z}^2$, leading to sets $\mathcal{C}_{train,\xi}$.

9 Check θ_1 for $\mathcal{Z}^2$ and draw new tangent planes using Algorithm 1B if incompatibility.

10 Calculate point sets with same closest tangent plane from $\mathcal{Z}^2$, leading to sets $\mathcal{C}_{\xi,train}$.

11 Calculate the overlap in $\mathcal{TP}$ using Algorithm 2 and update $\mathcal{C}_{train,\xi}$.

12 For all tangent planes ξ and important points ζ, calculate $\mathbf{Z}^U_{\zeta,\xi}$, $\mathbf{Z}^V_{\zeta,\xi}$ and $\mathbf{C}_{\zeta,\xi}$.

13 $\epsilon_1 = \inf$.

14 **while** any $\epsilon_r > \gamma_{int}$ **do**

15 $k = k + 1$.

16 $\mathcal{T}^k = F(\mathcal{Z}^{k-1})$.

17 Calculate snapshot matrices $\mathbf{S}(\boldsymbol{\lambda}_r)$ $\forall \boldsymbol{\lambda}_r \in \mathcal{T}^k$.

18 Determine sets of points with closest tangent plane ξ, resulting $\mathcal{C}_{\xi,test}$.

19 For all tangent planes ξ interpolate $\mathbf{Z}^U_{\zeta,\xi}$, $\mathbf{Z}^V_{\zeta,\xi}$ and $\mathbf{C}_{\zeta,\xi}$ to points $\mathcal{C}_{\xi,test}$ using points in $\mathcal{C}_{\xi,train}$ with the RBF interpolator in Equation (62).

20 Reconstruct the interpolated snapshot matrices $\mathbf{S}_{int}(\boldsymbol{\lambda}_r)$.

21 Determine the interpolation errors ϵ_r using Equations (25), (26) and (27).

22 Test points with too high error: $\mathcal{K}^k = \{\boldsymbol{\lambda}_r \in \mathcal{T}^k \mid \epsilon_r > \gamma_{int}\}$.

23 $\mathcal{Z}^k = \mathcal{Z}^{k-1} \cap \mathcal{K}^k$.

24 Add missing ancestors of points in $\mathcal{K}^k$ with partial ancestry in greedy case.

25 Repeat line 7-12, with the new set $\mathcal{Z}^k$.

26 **end**

27

3. Experimental Method

The algorithm based on local bases is tested on time-dependent problems with analytical and numerical solutions. The first model is a 1D nonlinear advection–diffusion problem with 1 input parameter. The second problem is a 2D advection-reaction problem with 5 different input parameters, of which results from the algorithm based on the global basis are available. The third problem is a 2D neutron diffusion problem that is solved numerically, which has 2 input parameters. In the next parts the ranges for the input parameter are given for each problem and more details about the experiments are provided.

3.1 The 1D Burgers equation

The one-dimensional Burgers equation is an example of a nonlinear advection–diffusion problem. It is obtained as a result of combining nonlinear wave motion with linear diffusion. The problem is dependent on the Reynolds number Re and can be written as [33]:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \frac{1}{\mathrm{Re}}\frac{\partial^2 u}{\partial x^2} + \Pi, \quad x \in [0,1], \quad t \in [0,1.5] \tag{68}$$

of which the exact solution, when the perturbation term $\Pi = 0$, is given by:

$$u(x,t) = \frac{\frac{x}{t+1}}{1 + \sqrt{\frac{t+1}{t_0}}\exp\left(\mathrm{Re}\,\frac{x^2}{4t+4}\right)}, \qquad t_0 = \exp(\mathrm{Re}\,/8). \tag{69}$$

For higher values of $\frac{1}{Re}$ the wave-breaking is suppressed, and shock discontinuities are smoothed out resulting in a smooth solution. However, in the inviscid limit, where the diffusion term $\frac{1}{Re}$ becomes low, the smooth viscous solution converges to a discontinuous shock wave. The data snapshots are generated using this analytical solution and the algorithm is tested for a Reynolds number range of $Re = [20, 1000]$. The analytical solution for different Reynolds numbers can be seen in Figure 19.

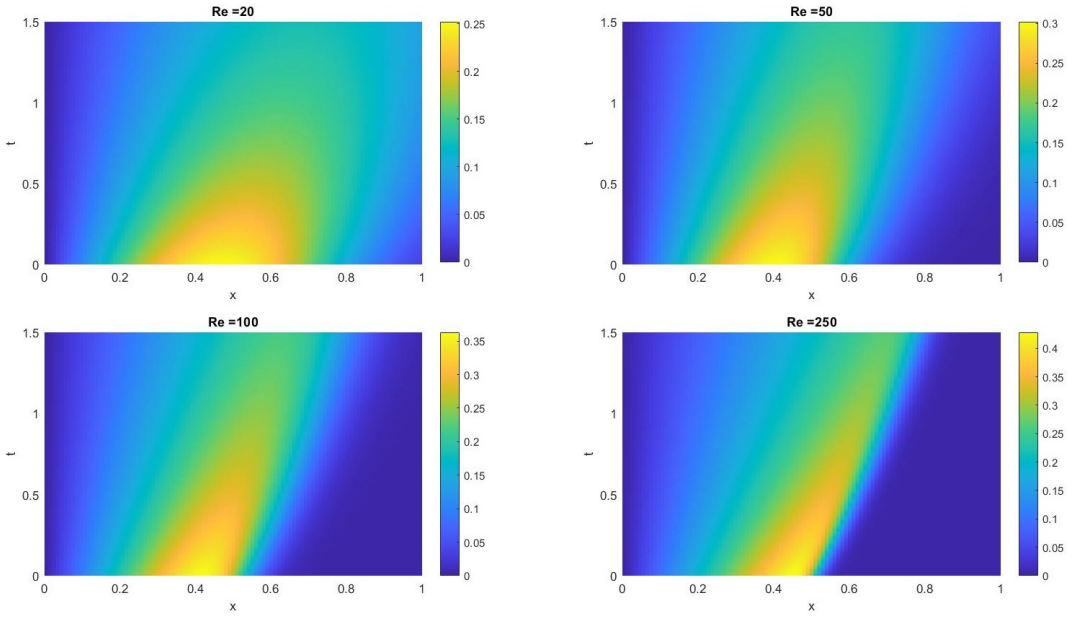


Figure 19: Time evolution of the analytical solution of the Burgers equation in Equation (69) for different Reynolds numbers.

For higher Reynolds numbers the wave front becomes more discontinuous. It will be interesting to see what the effect of this discontinuity is on the number of modes that is needed to represent the analytical solution as this is important for the placement of tangent planes in the algorithm.

The spatial discretization that was used for the Burgers problem was 100 points in the range $x \in [0, 1]$ and the temporal discretization contained 100 time points in the range $t \in [0, 1.5]$. The interpolation accuracy γ_{int} is set to $1 \cdot 10^{-2}$ and the maximum error tolerance for the number of modes is set to $\epsilon_{tol,max} = 9.9 \cdot 10^{-3}$. This is done to prevent cases where the approximation accuracy for a certain mode is very close to γ_{int} which might lead to the model not converging when sampling. To see the effect of the approximation tolerance range $[\epsilon_{tol,min}, \epsilon_{tol,max}]$ on the number of tangent planes that are drawn, 3 experiments are carried out where $\epsilon_{tol,min}$ is set to $1 \cdot 10^{-3}$, $5 \cdot 10^{-4}$ and $1 \cdot 10^{-4}$. This is repeated 6 times in which the number of neighbours that are used to calculate the subdomains for the training points is set to 1, 2 or 3, and the RBF function is set to multiquadric or cubic. The values that are used in these experiments are tabulated in Table 1.

Table 1: The values that are of importance for the algorithm, for the different experiments. MQ stands for the multiquadric RBF. The same experiments are repeated for the Cubic RBF where CU indicates the cubic RBF.

Experiment	RBF	γ_{int}	$\epsilon_{tol,max}$	$\epsilon_{tol,min}$	# neighbours
MQ.A.1	multiquadric	$1 \cdot 10^{-2}$	$9.9 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	1
MQ.B.1	multiquadric	$1 \cdot 10^{-2}$	$9.9 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	1
MQ.C.1	multiquadric	$1 \cdot 10^{-2}$	$9.9 \cdot 10^{-3}$	$1 \cdot 10^{-4}$	1
MQ.A.2	multiquadric	$1 \cdot 10^{-2}$	$9.9 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	2
MQ.B.2	multiquadric	$1 \cdot 10^{-2}$	$9.9 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	2
MQ.C.2	multiquadric	$1 \cdot 10^{-2}$	$9.9 \cdot 10^{-3}$	$1 \cdot 10^{-4}$	2
MQ.A.3	multiquadric	$1 \cdot 10^{-2}$	$9.9 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	3
MQ.B.3	multiquadric	$1 \cdot 10^{-2}$	$9.9 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	3
MQ.C.3	multiquadric	$1 \cdot 10^{-2}$	$9.9 \cdot 10^{-3}$	$1 \cdot 10^{-4}$	3

These same experiments are carried out with the cubic RBF and the experiments are named CU.A.1 for example. To check the performance of the resulting ROM, 1000 new Reynolds numbers are sampled using LHS (Latin hypercubic sampling), that constitute the set of check points $\mathcal{CH} = \{\boldsymbol{\lambda}_\kappa | \kappa = 1, \ldots, N_{check}\}$. For these new Reynolds numbers, the true analytical solution is calculated over time t using equation (68), which leads to 1000 different time evolutions defined by snapshot matrices $\mathbf{S}_{true,\kappa}$. This leads to 10^5 snapshots $\mathbf{y}_{true}(\boldsymbol{\lambda}_\kappa, t_\tau)$ in total. Afterwards, an interpolation is done to the new Reynolds numbers in the check set using the sampled important points from the algorithm and calculates $\mathbf{U}(\boldsymbol{\lambda}_\kappa)$, $\mathbf{C}(\boldsymbol{\lambda}_\kappa)$ and $\mathbf{V}(\boldsymbol{\lambda}_\kappa)$ and reconstructs the interpolated time evolutions $\mathbf{S}(\boldsymbol{\lambda}_\kappa)$ containing interpolated snapshots $\mathbf{y}_{int}(\boldsymbol{\lambda}_\kappa, t_\tau)$. Then the interpolated solutions $\mathbf{y}_{int}(\boldsymbol{\lambda}_\kappa, t_\tau)$ are compared with the true solutions $\mathbf{y}_{true}(\boldsymbol{\lambda}_\kappa, t_\tau)$ by calculating the L_2 error over all time and parameter combinations.

$$\epsilon_{\kappa,\tau} = \frac{\|(\mathbf{y}_{true}(\boldsymbol{\lambda}_\kappa, t_\tau) - \mathbf{y}_{int}(\boldsymbol{\lambda}_\kappa, t_\tau))\|_{L_2}}{(\|\mathbf{y}_{true}(\boldsymbol{\lambda}_\kappa, t_\tau)\|_{L_2} + \eta)}, \quad \tau = 1, \ldots, N_t, \ \kappa = 1, \ldots, N_{check}, \tag{70}$$

after which the maximum L_2 error is taken over all time and parameter combinations given by:

$$\epsilon_{L_2,max} = \max_{\epsilon_{\kappa,\tau}} \{\epsilon_{r,\tau}, \tau = 1, \ldots, N_t, \ \kappa = 1, \ldots, N_{check}\}. \tag{71}$$

The individual interpolation accuracy of $\mathbf{U}$, $\mathbf{C}$ and $\mathbf{V}$ will also be analyzed. This is done by taking the L_2 error between the interpolated columns of $\mathbf{U}(\boldsymbol{\lambda}_\kappa)$, $\mathbf{C}(\boldsymbol{\lambda}_\kappa)$ and $\mathbf{V}(\boldsymbol{\lambda}_\kappa)$ and the true columns in $\mathbf{U}_{true,\kappa}$, $\mathbf{C}_{true,\kappa}$ and $\mathbf{V}_{true,\kappa}$. These errors are given by:

$$\epsilon_{L_2,\mathbf{u}} = \frac{\|(\mathbf{u}_{true,\kappa,i} - \mathbf{u}_{int,i}(\boldsymbol{\lambda}_\kappa)\|_{L_2}}{\|\mathbf{u}_{true,\kappa,i}\|_{L_2}}, \qquad i = 1,\ldots,N_m,\ \kappa = 1,\ldots,N_{check}, \tag{72}$$

$$\epsilon_{L_2,\mathbf{c},\mathbf{i}} = \frac{\|(\mathbf{c}_{true,\kappa,i} - \mathbf{c}_{int,i}(\boldsymbol{\lambda}_\kappa))\|_{L_2}}{\|\mathbf{c}_{true,\kappa,i}\|_{L_2}}, \qquad i = 1,\ldots,N_m,\ \kappa = 1,\ldots,N_{check}, \tag{73}$$

$$\epsilon_{L_2,\mathbf{v},\mathbf{i}} = \frac{\|(\mathbf{v}_{true,\kappa,i} - \mathbf{v}_{int,i}(\boldsymbol{\lambda}_\kappa))\|_{L_2}}{\|\mathbf{v}_{true,\kappa,i}\|_{L_2}}, \qquad i = 1,\ldots,N_m,\ \kappa = 1,\ldots,N_{check}, \tag{74}$$

where the vectors $\mathbf{u}_{true,\kappa,i}$, $\mathbf{c}_{true,\kappa,i}$ and $\mathbf{v}_{true,\kappa,i}$ are the true columns of $\mathbf{U}_{true,\kappa}$, $\mathbf{C}_{true,\kappa}$ and $\mathbf{V}_{true,\kappa}$, and $\mathbf{u}_{int,i}(\boldsymbol{\lambda}_\kappa)$, $\mathbf{c}_{int,i}(\boldsymbol{\lambda}_\kappa)$ and $\mathbf{v}_{int,i}(\boldsymbol{\lambda}_\kappa)$ are the interpolated columns in $\mathbf{U}(\boldsymbol{\lambda}_\kappa)$, $\mathbf{C}(\boldsymbol{\lambda}_\kappa)$ and $\mathbf{V}(\boldsymbol{\lambda}_\kappa)$. $\epsilon_{L_2,max}$, the number of evaluations and the number of drawn tangent planes are included in the results.

3.2 The Molenkamp test with a 5-dimensional parameter domain

The Molenkamp test represents a two-dimensional advection problem, where the exact solution can be seen as a Gaussian cloud of material moving in a circular path without changing its shape. In this research, a slight modification is made to the function by adding a reaction term, resulting in the amplitude of the solution to decay over time. The dimensionless equation representing the Molenkamp problem is given by [4]:

$$\frac{\partial q(x,y,t)}{\partial t} + u\frac{\partial q(x,y,t)}{\partial x} + v\frac{\partial q(x,y,t)}{\partial y} + \lambda_3 q(x,y,t) = 0, \quad (x,y) \in [-1,1], \tag{75}$$

with initial condition given by:

$$q(x,y,0) = \lambda_1 0.01^{\lambda_2 h(x,y,0)^2}, \quad h(x,y,0) = \sqrt{\left(x - \lambda_4 + \frac{1}{2}\right)^2 + (y-\lambda_5)^2}. \tag{76}$$

u and v describe a solid body rotation where $u = -2$ and $v = 2$. The exact solution to this boundary value problem is given by:

$$q(x,y,t) = \lambda_1 0.01^{\lambda_2 h(x,y,t)^2} e^{-\lambda_3 t}, \tag{77}$$

$$h(x,y,t) = \sqrt{\left(x - \lambda_4 + \frac{1}{2}\cos 2\pi t\right)^2 + \left(y - \lambda_5 + \frac{1}{2}\sin 2\pi t\right)^2}. \tag{78}$$

The problem has a 5-dimensional parameter space in which the input parameters are denoted by λ_i for $i = 1,\ldots,5$. The parameter λ_1 is a linear coefficient that represents the scaling of the cloud. Increasing λ_2 results in the solution having a steeper gradient (spike-like). λ_3 is the decay constant of the cloud and lastly, λ_4 and λ_5 control the initial coordinates of the center of the cloud with respect to the center of the Cartesian plane. The exact solution for different time steps is shown in the Figure 20. It shows a Gaussian cloud initially centered at $x = -0.5$ and $y = 0$, which moves counter-clockwise in a circle completing a full rotation at $t = 1$. As the cloud also decays to reach a near-zero magnitude after a full rotation. For a higher λ_2 value the Gaussian cloud becomes steeper and that the cloud amplitude decays much faster for a larger decay constant λ_3.

The ROM based on the global basis was also tested on the Molenkamp problem. By testing the ROM based on local bases on this problem with similar input parameters and comparing the results, it can be determined how well the new algorithm performs relative to the old algorithm. Therefore, most input parameters to the algorithm are kept the same to have a fair comparison. The analytical solution is evaluated on a Cartesian uniform 100×100 grid, which translates to 10000 degrees of freedom. The amount of time points is set to 11 in a range of $t \in [0,1]$. The interpolation accuracy γ_{int} is set to $\gamma_{int} = 1 \cdot 10^{-3}$ and the maximum error tolerance for the number

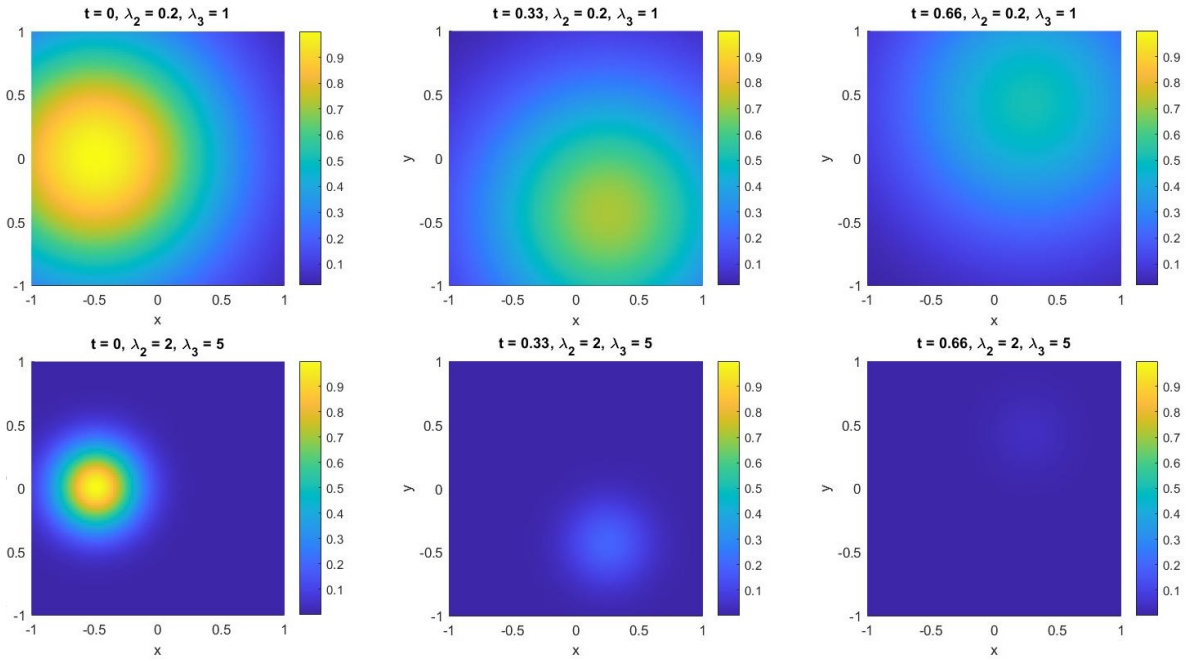


Figure 20: Time evolution of the analytical solution of the Molenkamp problem in Equation (78) for different steepness values λ_2 and different decay constants λ_3.

of modes is set to $\epsilon_{tol,max} = 9.9 \cdot 10^{-4}$. The algorithm is set to sample maximally greedy, where μ in this case is equal to 1. In this experiment, $\epsilon_{tol,min}$ is neglected and instead the first mode that results in an error below $\epsilon_{tol,max}$ is taken. The reason for this is that the amount of time points that are used in these experiments results in the approximation error to change between $1 \cdot 10^{-3}$ and $1 \cdot 10^{-16}$ with only one mode, which effectively leads to only one tangent plane being drawn over the full parameter domain. This allows for an interesting experiment to see what the true performance is of this interpolation method without any of the added issues of drawing extra tangent planes. The algorithm will be tested on two different domains for the parameter λ_2, which are the smooth and steep domain. The parameter ranges for these domains in the Molenkamp problem are shown in Table 2.

Table 2: The range for the input parameters in the Molenkamp test.

Parameter	Lower bound	Upper bound
λ_1	1	20
λ_2	0.1 (2 for steep setting)	0.2 (4 for steep setting)
λ_3	1	5
λ_4	-0.1	0.1
λ_5	-0.1	0.1

Just like in the experiment with the algorithm based on the global basis, 1000 different parameter combinations are sampled using LHS. 1000 check points are not enough to cover the whole parametric dependence of the Molenkamp problem in the full parameter domain and the error variance can be large depending on which random check points were chosen. As the Molenkamp problem is a 2D problem, an absolute relative percentage difference will be used instead of the L_2 error to show the error of the individual entries in the interpolated state vectors $\mathbf{y}_{int}(\boldsymbol{\lambda}_\kappa, t_\tau)$ and the matrices $\mathbf{U}(\boldsymbol{\lambda}_\kappa)$ and $\mathbf{C}(\boldsymbol{\lambda}_\kappa)$, which is defined by:

$$\epsilon_{\%,\mathbf{y}} = \frac{|\mathbf{y}_{true}(\boldsymbol{\lambda}_\kappa, t_\tau) - \mathbf{y}_{int}(\boldsymbol{\lambda}_\kappa, t_\tau)|}{\max\left(\mathbf{y}_{true}(\boldsymbol{\lambda}_\kappa, t_\tau)\right)}, \tag{79}$$

$$\epsilon_{\%,\mathbf{U}} = \frac{|\mathbf{U}_{true,\kappa} - \mathbf{U}(\boldsymbol{\lambda}_\kappa)|}{\max\left(\mathbf{U}_{true,\kappa}\right)}, \tag{80}$$

$$\epsilon_{\%,\mathbf{C}} = \frac{|\mathbf{C}_{true,\kappa} - \mathbf{C}(\boldsymbol{\lambda}_\kappa)|}{\max\left(\mathbf{C}_{true,\kappa}\right)}. \tag{81}$$

$\epsilon_{L_2,max}$, the number of evaluations, but also the ratio of important points over the total amount of sampled points and the percentage of snapshots at which the interpolation error was higher than γ_{int}, are included in the results.

3.3 A 2D neutron diffusion problem with a fixed source

For the following problem a 1-group time dependent neutron diffusion equation is solved in a 2-dimensional geometry, which is given by [43]:

$$\frac{1}{v}\frac{\partial \Phi\left(\vec{r},t\right)}{\partial t} = \vec{\nabla}\left(D\left(\vec{r}\right)\vec{\nabla}\Phi\left(\vec{r},t\right)\right) - \Sigma_a\left(\vec{r}\right)\Phi\left(\vec{r},t\right) + \nu\Sigma_f\left(\vec{r}\right)\Phi\left(\vec{r},t\right) + S\left(\vec{r},t\right), \tag{82}$$

where $\Phi\left(\vec{r},t\right)$ denotes the flux, $D\left(\vec{r}\right)$ is the diffusion coefficient, Σ_a and Σ_f are the absorption and fission cross sections and $S\left(\vec{r},t\right)$ is an additional external source. ν denotes the amount of neutrons released per fission event. This continuous PDE problem is converted to a discrete problem by applying linear constraints which are determined by a finite set of basis functions. The resulting mesh is built using continuous triangular elements. The geometry and mesh are shown in Figure 21. The geometry consists of 3 different materials (M1, M2, M3) that represent an arbitrary fuel, absorber, and moderator, where the materials have a respectively high fission, capture, and scatter cross section. Domain F3 contains the fuel M1, domain F1 contains the moderator M2 and domains F4 and F2 contain absorber material M3. A high initial flux in domain F4 is chosen which will then diffuse throughout the full domain where it eventually approaches the fuel in F3. The boundary conditions are Dirichlet for boundary E2 and Neumann for E3-E7. The main parameters that are perturbed in this problem, are the source term S in domain F3, the fission cross section of the fuel material $\sigma_{f,M1}$ and the capture cross section of the absorber $\sigma_{c,M3}$. A built-in PDE solver in Matlab [44] is then used to discretise and solve the problem using Newton's method. The goal is to simulate 2 different operating domains that contain different modes.

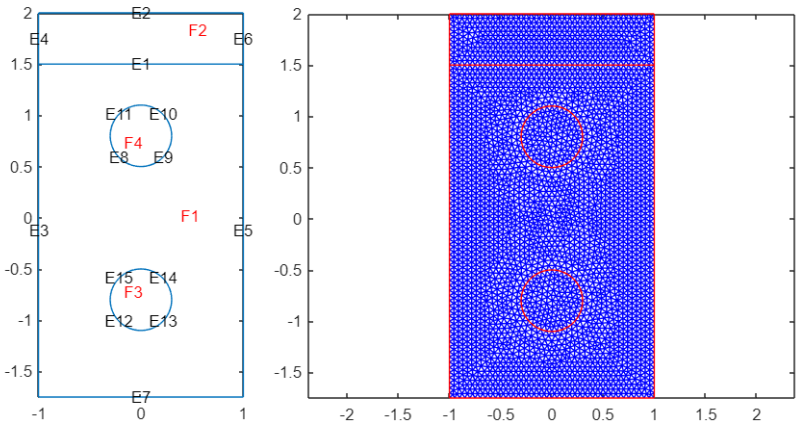


Figure 21: On the left, the 2D geometry is shown that is used for the 1-group neutron diffusion experiment. On the right, the resulting mesh is shown that is used for the numerical solver. The numbers F1-F4 denote the material domains, with their own specified parameters such as cross sections terms. E2-E13 denote the edges of the different domains.

It is expected that the added source in region F3, will lead to a very different looking solution. In Figure 22 the flux solution is plotted for different time points, where the upper row represents the

flux with no source and minimal σ_f and the lower row represents the flux with added source and maximal σ_f.

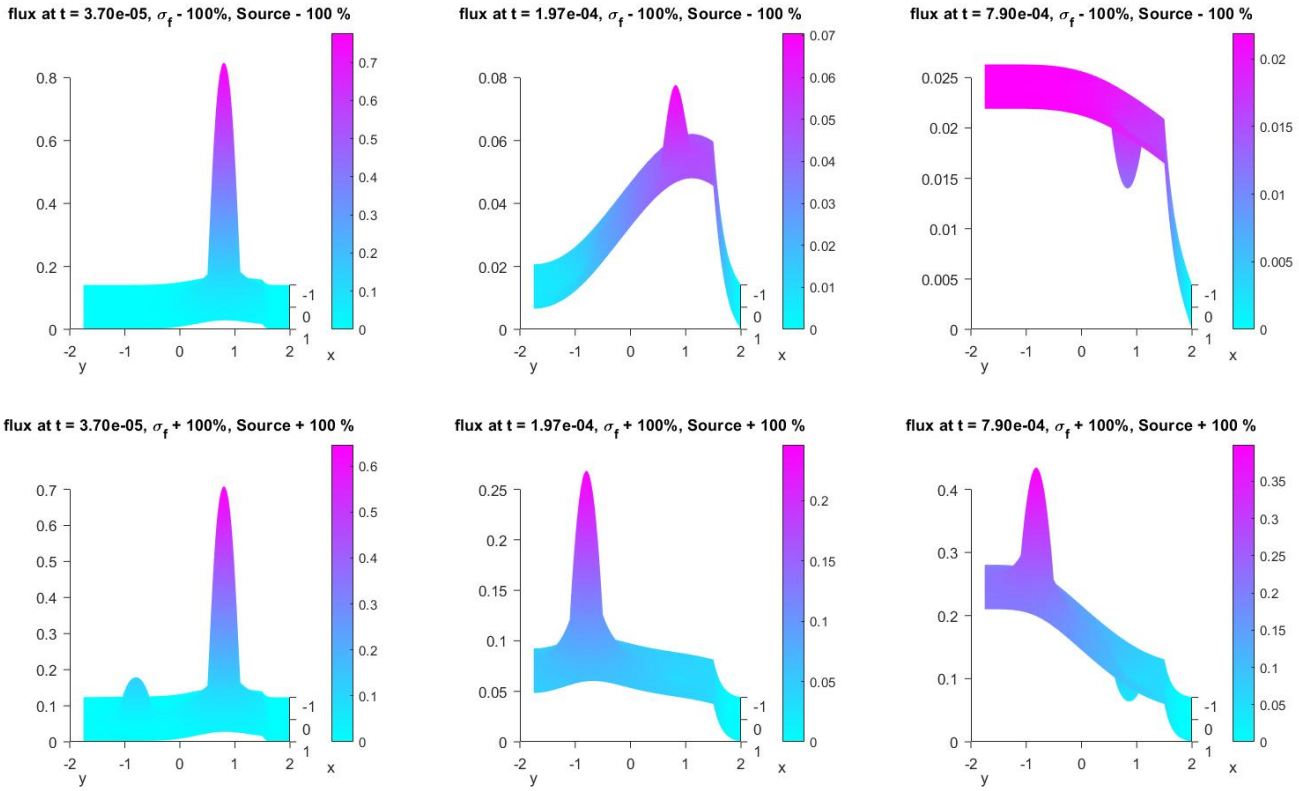


Figure 22: Time evolution of the numerical solution of the flux in the neutron diffusion problem for different source term values and cross sections σ_f.

It can be observed that the time evolution of the flux is different between the situation with and without the source term. In the upper middle and right plot, the flux slowly diffuses from the absorber to the fuel, while in the lower middle and right plot, the flux at the fuel completely overrules the flux at the absorber. The middle graphs clearly show that if the source term is added, the flux at negative y-values increases faster because of the added source term than the diffusion of the initial flux at the absorber to the fuel rod.

The maximum size for the triangular elements of the mesh was set 0.05, which led to 13423 degrees of freedom in the flux. The amount of time points was set to 221, where 6 time points were chosen for the range $\mathbf{t}_1 \in [0.5, 1] \cdot 10^{-5}$, 145 time points were chosen in the range $\mathbf{t}_2 \in [1.1 : 30] \cdot 10^{-5}$ and 70 points were used in the range $\mathbf{t}_3 \in [3.1 : 10] \cdot 10^{-4}$, leading to $\mathbf{t} = [\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3]$. The reason for $\mathbf{t}_1$ is to make sure the numerical solver is stable in the early time steps. In range $\mathbf{t}_2$ most of the changes in time were happening, while fewer dynamics happened in $\mathbf{t}_3$, which is why $\mathbf{t}_2$ has more time points. The interpolation tolerance γ_{int} is set to $5 \cdot 10^{-3}$ and the minimum and maximum error tolerance, $\epsilon_{tol,min}$ and $\epsilon_{tol,max}$, are set to $1 \cdot 10^{-3}$ and $4.95 \cdot 10^{-3}$. The algorithm is set to sample minimally greedy, where μ in this case is equal to 0. The number of neighbours is set to 1 and the chosen RBF is the multiquadric function. 4 experiments are done where 2 parameters, the source and the fission or capture cross sections, are perturbed with different perturbation percentages. Additionally, different principal angle tolerances $\theta_{1,tol}$ are used which allows to see which parts of the parameter domain are more related to each other than other domains. The experiments are listed in Table 3.

Table 3: The input parameters that are perturbed in the neutron diffusion problem. S stands for the source term and σ stands for either the fission or the capture cross sections of respective materials M1 or M3. $\theta_{1,tol}$ represents the tolerance for the first principal angle when drawing tangent planes.

Experiment	Chosen parameters	Perturbation %	$\theta_{1,tol}$
S.$\sigma_{f,M1}$.A	S, $\sigma_{f,M1}$	[-100,0], [-100,100]	$\pi/2$
S.$\sigma_{f,M1}$.B	S, $\sigma_{f,M1}$	[-100,100], [-100,100]	$\pi/2$
S.$\sigma_{f,M1}$.C	S, $\sigma_{f,M1}$	[-100,100], [-100,100]	$2\pi/5$
S.$\sigma_{c,M3}$	S, $\sigma_{c,M3}$	[-100,100], [-50,400]	$\pi/2$

For this 2D parameter domain, 100 check points are sampled that represent different perturbation percentages for the source term and the cross sections σ_f or σ_c. The main parameters that are tracked are the same as the Molenkamp problem, but in this case also the number of tangent planes is tracked. In Chapter 4, first an analysis is made of the modes for each of the three problems, after which the actual performance of the algorithm is examined.

4. Results

4.1 Dependencies of the modes and coefficients on the parameters of the models.

In this section an analysis will be made of the modes and coefficients in connection with the input parameters for each individual problem. As a local basis and coefficient interpolation is done, it is important to now the dependence of the modes on the parameters in the problems. Multiple plots will be given in which the modes and coefficients are shown and are afterwards discussed.

4.1.1 Influence of the Reynolds number in the Burgers equation.

In Figure 23, modes 1, 5, 10, 15 and 20 of the analytical solution for the Burgers equation in Equation (69) are plotted for different Reynolds numbers.

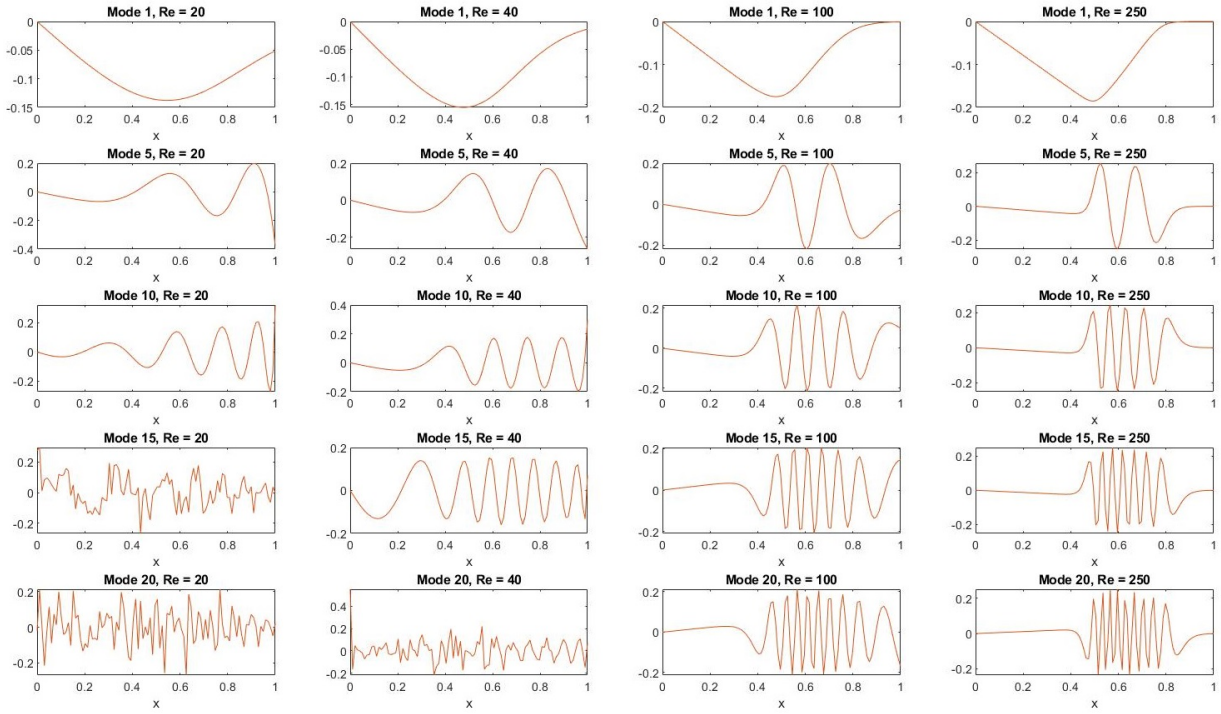


Figure 23: Different modes that represent the physics of the Burgers equation at different Reynolds numbers.

First, as the Reynolds number increases, the frequency of the modes increases and the oscillatory behaviour becomes sharper, which is expected. Namely, as the Reynolds number increases, the wavefront becomes sharper, meaning that higher frequency modes are needed to approximate the sharp part of the solution. At mode 15 and 20 of $Re = 20$ and mode 20 of $Re = 40$ noisy behaviour occurs. The reason for this is that for lower Reynolds numbers, the solution is smooth and simple meaning that not many modes are needed to represent the analytical solution. In this case, the higher order modes do not represent the physics of the model, but instead represent the numerical noise coming from the SVD calculation. As the number of modes, needed for a certain approximation accuracy, increases as a function of the Reynolds number, an analysis is made of the number of tangent planes that will have to be drawn over the Reynolds number domain. The noisy behaviour is also expected to occur in the coefficients which are plotted in Figure 24.

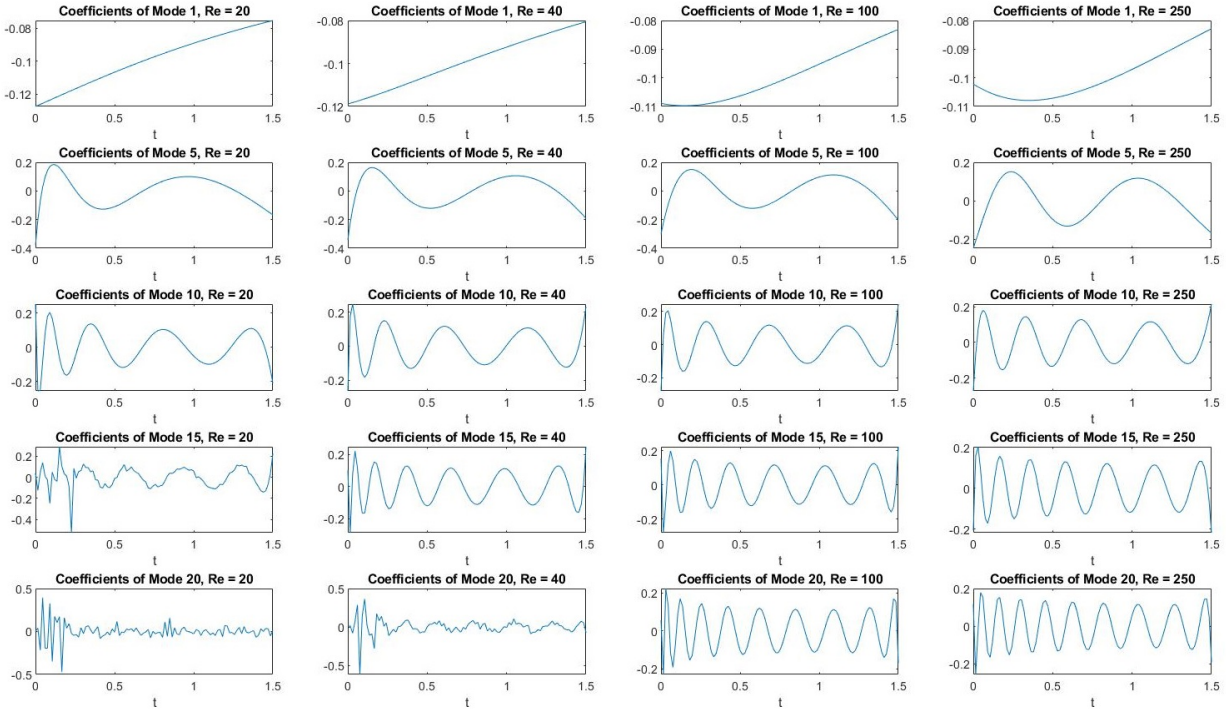


Figure 24: The coefficients $\mathbf{v}_i(t)$ that represent the time-dependent behaviour of different modes i for solutions of the Burgers equation at different Reynolds numbers.

Similarly to the modes, the coefficients become more oscillatory as the mode order increases. Noise occurs starting from the same modes 15 and 20 respectively at Reynolds numbers of 20 and 40. The wave-like pattern of the coefficient is tied to the fact that the solution to the Burgers equation is a positive moving wavefront. This means that the oscillatory modes and the coefficients are counterbalancing each other to produce a positive solution over the full domain. As the frequency of the modes is higher than their respective coefficients it will be interesting to see if for this problem the interpolation of the modes is more difficult than the coefficients.

4.1.2 Effect of the steepness and decay constant in the Molenkamp test.

In Figure 25, modes 1 to 4 are plotted for the Molekamp solution in Equation (78) in the smooth setting with a steepness value λ_2 of 0.1 and 0.2 and an exponential decay constant λ_3 of 1 and 5.

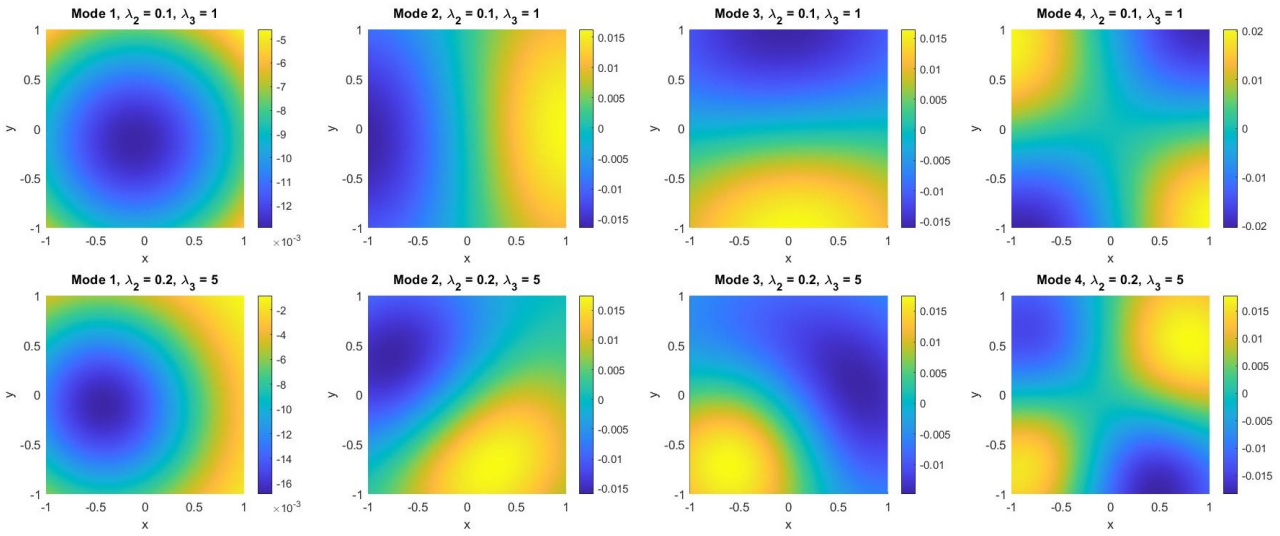


Figure 25: Different modes that represent the physics of the analytical solution for the Molenkamp test in the smooth setting with respective low and high steepness and decay constants.

The first mode becomes smaller and is shifted to the left. The reason for this is that as the steepness value increases, the gas cloud covers less area of the Cartesian plane. To represent this gas cloud, the modes must become more localized as well. This can be seen more clearly for mode 2 and 3, where instead of the mode covering the almost half of the plane, the mode covers only a corner. In Figure A1 in the Appendix, mode 5 until 8 are plotted. The coefficients for modes 1-4 are also plotted in Figure 26.

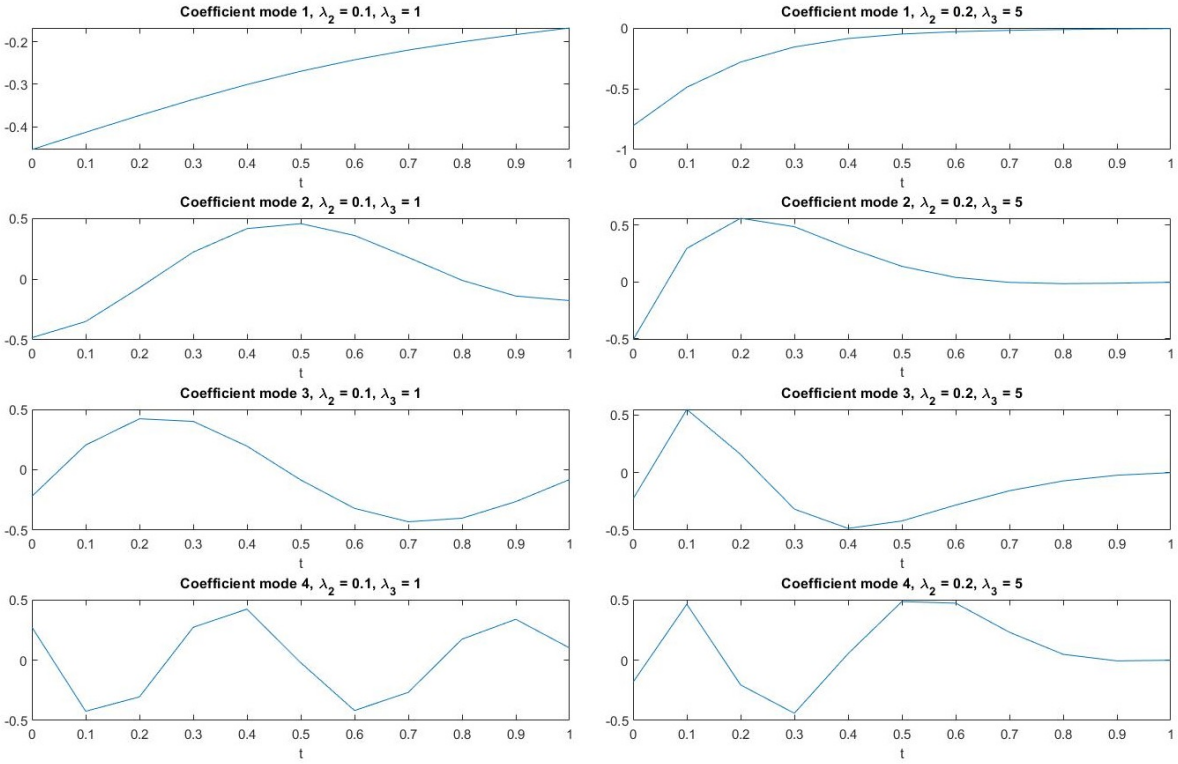


Figure 26: The coefficients that represent the amplitude of the modes in time for the Molenkamp test in the smooth setting with respective low and high steepness and decay constants.

For higher order modes the coefficients become less smooth and start to become oscillatory. This behaviour is similar to the Burgers equation, where in this case the modes and the coefficients counterbalance each other to form the positive rotating Gaussian cloud. For a high decay constant, the coefficients decrease to zero, which is expected, as for a higher decay constant the full solution should go to zero faster. The coefficients of modes 5-8 are given in Figure A2 in the Appendix.

In Figure 27, the modes 1 to 4 are plotted for the Molenkamp solution in the steep setting with a steepness value λ_2 of 2 and 4 and a exponential decay constant λ_3 of 1 and 5. As was already stated for the smooth setting, the modes become even more localized as the steepness value increases. Already at modes 2, 3 and 4 the rotational character of the Molenkamp problem can be noticed. The frequency of the rotating wave pattern also increases with the steepness parameter λ_2, which can be seen even more clearly in modes 5-8 that are plotted in the Appendix in Figure A3. In the problem with the Burgers equation, similar behaviour was seen where a more non-linear discontinuous shockwave led to more non-linear modes being needed to represent the solution.

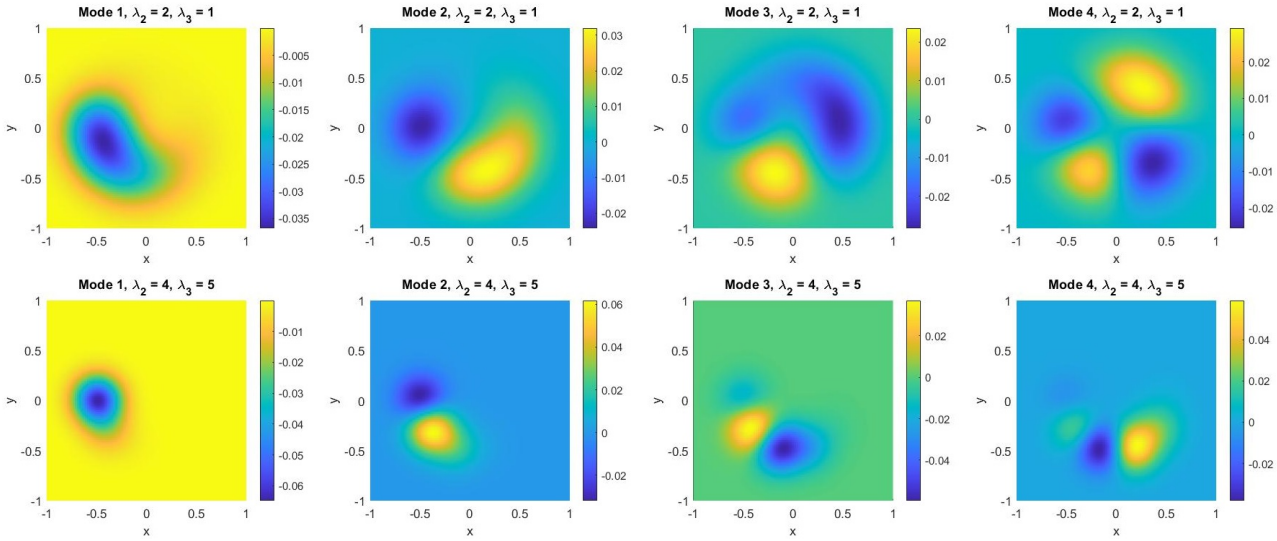


Figure 27: Different modes that represent the physics of the analytical solution for the Molenkamp test in the steep setting with respective low and high steepness and decay constants.

The coefficients of these modes are plotted in Figure 28 and Figure A4 in the Appendix.

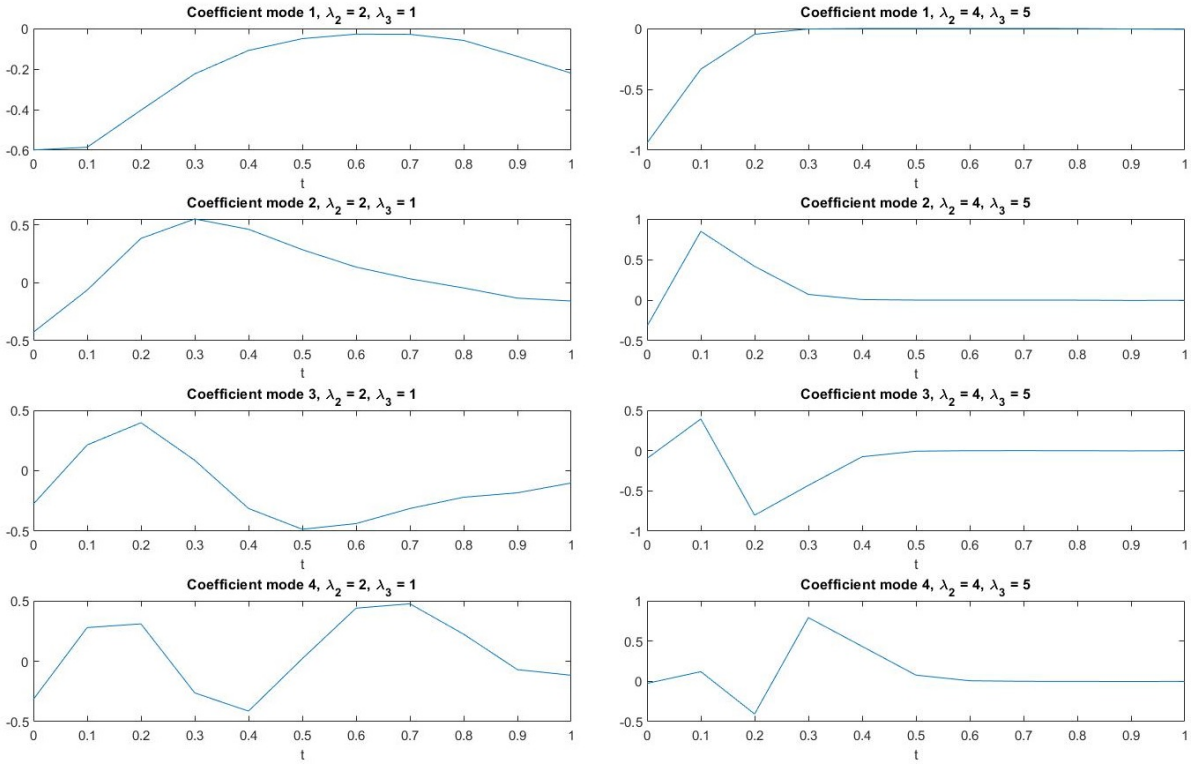


Figure 28: The coefficients that represent the amplitude of the modes in time for the Molenkamp test in the steep setting with respective low and high steepness and decay constants.

Again, for a high decay constant the coefficients decrease to zero. The coefficients look very similar to the coefficients in the smooth setting, which is expected. As there is a clear distinction in the dependence of the modes and coefficients on the different parameters, it will be interesting to see how many unique points the algorithm will sample for each parameter λ_i and how the interpolator behaves as a function of these parameters.

4.1.3 Dependence on the source and fission cross section in the neutron diffusion problem.

In Figure 29, modes 1, 2, 3 and 4 are plotted of the flux that is calculated via the 1 group time-dependent neutron diffusion code. The modes 4-8 are plotted in the appendix in Figure A5. These modes are plotted for the situation with and without source where the fuel had respectively the minimum and maximum number of fission cross section within the domain.

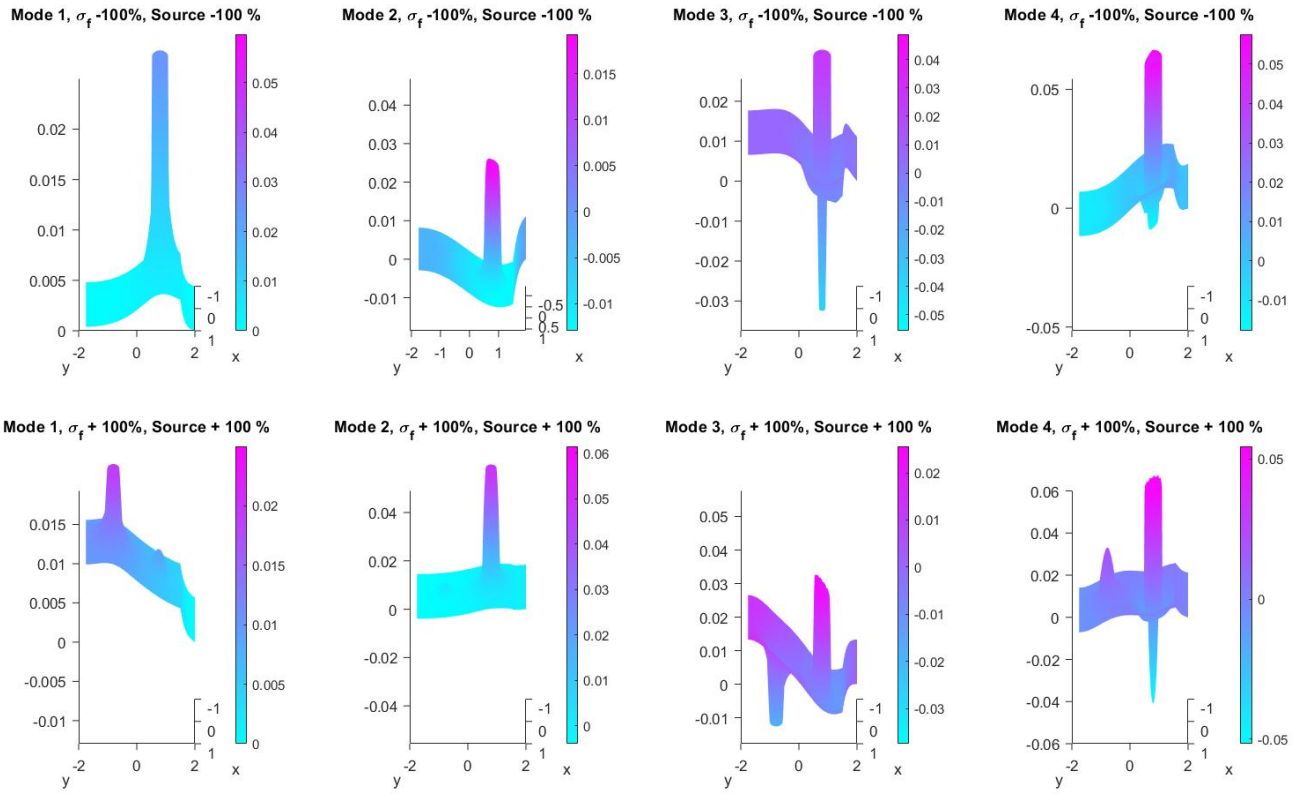


Figure 29: Different modes that represent the physics of the numerical solution for the 2D neutron diffusion problem in cases with and without a source term at the fuel rod with respective minimum and maximum σ_f.

The main difference of all 4 modes between the upper and lower row is the presence of the fuel rod, whether it be a peak or a trough in the graph. This can also be deduced from the solution in Figure 22, where the middle graphs clearly show that if the source term is added, the flux for negative y-values increases faster because of the added source than the diffusion of the initial flux at the absorber to the fuel rod. This explains why already the first mode looks very different, as the source flux peak will be present throughout many of the state vectors in the time evolution with the source term compared to the time evolution without the source term. As the modes between the situation with and without source term are very different, it will be interesting to see how the principal angle behaves over the domain, as the principal angle could be an indicator that shows how relatable the physics is in certain regions of the domain. One detail that must be noted is the number of noise that occurs in the higher order modes in Figure A5. The coefficients of these mode are given in Figure 30.

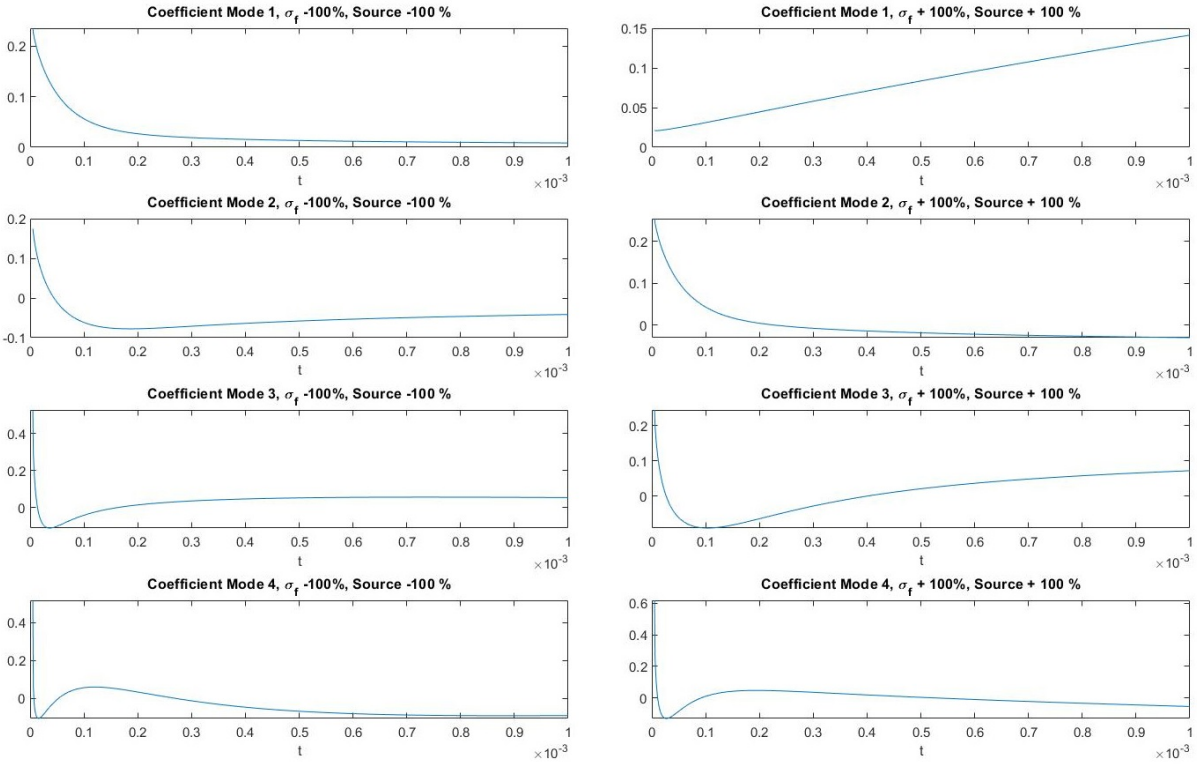


Figure 30: The coefficients that represent the amplitude of the modes 1-4 in time for the neutron diffusion problem.

The most important detail to notice from the coefficients is that over time, most of the coefficients go to zero while the first mode of the situation with the source term does not go to zero. This happens because without any fission cross section and source term, the system is not able to produce any flux on its own. This leads to a very subcritical system in which the system slowly goes to zero flux. For the situation with the source term, the system can produce some flux which means that after a certain time the system reaches steady state where only the first principal mode will represent the system as there is no change in time anymore.

4.2 Algorithm performance analysis

In this section the main results of all experiments mentioned in Chapter 3 will be tabulated and discussed. For experiments with the highest interpolation error $\epsilon_{L_2,max}$, the interpolation accuracy will be analyzed more in-depth, in which for instance also the interpolation accuracy of $\mathbf{U}(\boldsymbol{\lambda}_{check})$, $\mathbf{C}(\boldsymbol{\lambda}_{check})$ and $\mathbf{V}(\boldsymbol{\lambda}_{check})$ will be checked. Moreover, in certain occasions, extra plots will be given that are used to show key details about the interpolation method.

4.2.1 Results of the experiments on the Burgers equation

The errors for the experiments done on the Burgers equation using the multiquadric RBF are tabulated in Table 4. The $\epsilon_{L2,max}$ is given, with the respective Reynolds number at which that value was obtained. Then, the number of evaluations or number of unique sampled parameter combinations and the number of tangent planes is provided.

Table 4: Results of the experiments for the Burgers equation. The multiquadric function is the chosen RBF, $Re_{max,error}$ indicates the Reynolds number at which $\epsilon_{L2,max}$ was obtained.

Experiment	$\epsilon_{L2,max}$ [%]	$Re_{max,error}$	# evaluations	# tangent planes
MQ.A.1	0.97	43	35	12
MQ.B.1	0.97	43	35	11
MQ.C.1	0.95	43	29	8
MQ.A.2	0.92	326	35	12
MQ.B.2	0.91	111	35	11
MQ.C.2	1.44	179	33	8
MQ.A.3	2.11	233	33	12
MQ.B.3	2.11	233	33	11
MQ.C.3	3.48	571	31	8

The $\epsilon_{L_2,max}$ over the 10^5 snapshots is lower than $\gamma_{int} = 1{\cdot}10^{-2}$ for the upper half of the experiments in Table 4, while the lower half is at maximum 3.5 times as high. The difference in the number of evaluations between the different experiments is small with only a large difference in evaluations between experiment MQ.B.1 and MQ.C.1. However, it is difficult to say whether this is solely due to the number of tangent planes or not. The number of tangent planes drops from 12 to 8 over experiments labeled A, B and C. This is expected as the error tolerance range for the chosen number of modes at a tangent plane increases over these experiments, which results in a larger range of different numbers of modes that are compatible with the approximation error requirement over the parameter domain. This can be seen when looking at the resulting sampled sparse grids for experiment MQ.A.1 and MQ.C.1, which are shown in Figures 31 and 32.

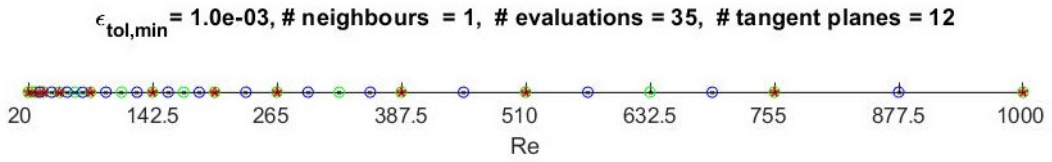


Figure 31: The resulting sampled grid of experiment MQ.A.1 for the Burgers equation.

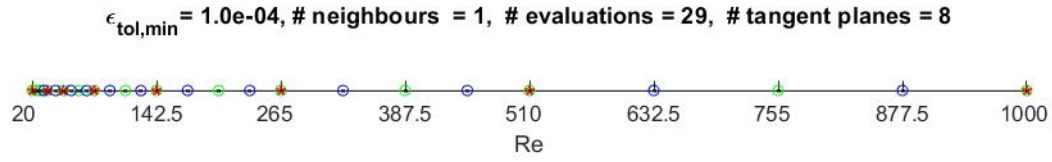


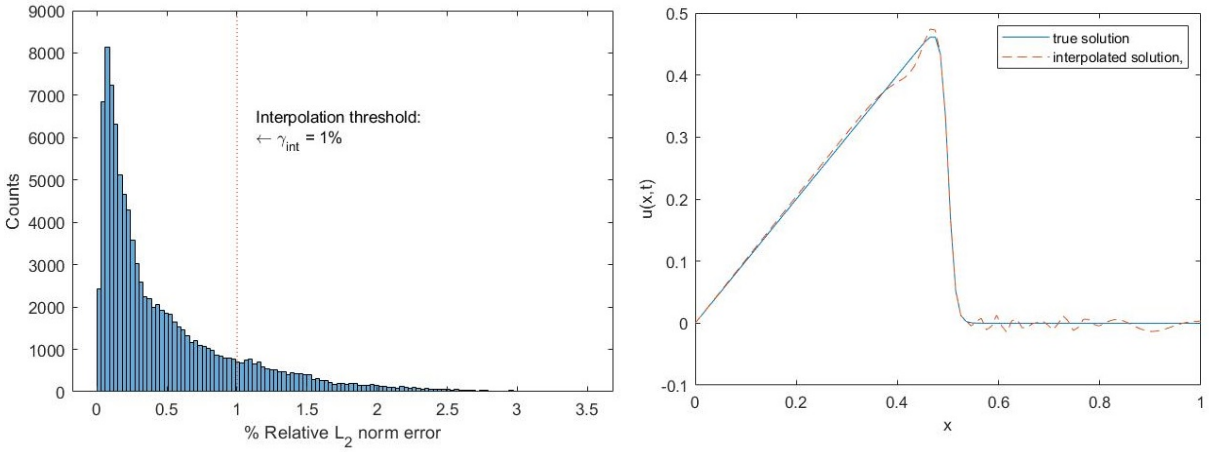
Figure 32: The resulting sampled grid of experiment MQ.C.1 for the Burgers equation.

The green points are the important points that are included in the ROM, the blue points are the inactive points where the interpolation error was low enough and the red points indicate a tangent plane. The first thing to note is the workings of the hierarchical sparse grid, where each unimportant point is surrounded by an important green point that is included in the full ROM and that the sampling around an unimportant blue point stops as the interpolation error was low enough there. The density of sampled points in general is higher at lower Reynolds numbers. This is expected as in Figure 19 there is a larger transition in dynamics in the lower Reynolds range 20-100 than in the higher Reynolds number range 100-250, which is more difficult to interpolate. The density of tangent planes is also higher for lower Reynolds numbers than for higher Reynolds numbers. These tangent planes were solely drawn because of the difference in modes needed to stay within the error tolerance range. This can again be deduced from the analytical solutions of the Burgers equation which becomes a sharp shock wave very fast in the lower Reynolds number range, after which the sharpness does not increase very much anymore. The $\epsilon_{L_2,max}$ increases when the error tolerance range increase and when more neighbours are used to calculate the overlapping training subdomains. The former can be deduced from the error increase from experiment MQ.B.2 to MQ.C.2 and from MQ.B.3 to MQ.C.3. The latter is mostly visible in the results of experiment MQ.C.1, MQ.C.2 and MQ.C.3, where the error equals 0.95%, 1.44% and 3.48%. What is notable is that for experiments where 1 neighbour was used, the value at which the highest error was achieved, $Re_{max,error} = 43$, typically lies in a much smaller training subdomain compared to the higher $Re_{max,error}$ values in experiments where more than 1 neighbour was used. These findings indicate that it is preferred to use smaller training subdomains when using the multiquadric RBF. The results from the experiments using the cubic RBF are provided in Table 5.

Table 5: Results of the experiments for the Burgers equation. The cubic equation is the chosen RBF.

Experiment	$\epsilon_{L2,max}$ [%]	$Re_{max,error}$	# evaluations	# tangent planes
CU.A.1	2.16	65	31	12
CU.B.1	2.16	65	33	11
CU.C.1	2.18	112	23	8
CU.A.2	1.16	24	33	12
CU.B.2	0.91	112	33	11
CU.C.2	0.71	387	27	8
CU.A.3	0.92	326	33	12
CU.B.3	0.91	112	33	11
CU.C.3	0.75	24	27	8

$\epsilon_{L_2,max}$ is within an acceptable range, which means that the algorithm can successfully provide a well performing ROM as well, using the cubic RBF. The lowest and highest achieved errors are 0.71% and 2.18%, with 27 and 23 evaluations in experiment CU.C.2 and CU.C.1 respectively. The difference in the number of evaluations compared to the results of the multiquadric equation is very small. The dependence of $\epsilon_{L_2,max}$ on the training subdomain size is opposite compared to the situation with the multiquadric RBF, as $\epsilon_{L_2,max}$ decreases as both the number of neighbours increases and as the number of tangent planes decreases, resulting in larger subdomains. The former can be seen in general by observing that the error of experiments A, B and C with 1 neighbour are almost twice as high as the experiments with 2 and 3 neighbours. The latter can be seen most clearly in experiments CU.C.1, CU.C.2 and CU.C.3 where the error decreases from 2.18% to 0.71% and 0.75%. These findings indicate that larger subdomains are preferred when using the cubic RBF. In general, the difference in error and evaluations is quite small, so it can be stated that both RBFs are proper candidates to be used in the algorithm, depending on the amount and distribution of the tangent planes in the parameter domain. For the experiment MQ.C.3, which had the largest error, a more in depth analysis is made. Figure 33 shows a histogram of the errors over all interpolated state vectors in experiment MQ.C.3 and the plot of the solution with the maximum error are plotted below.



(a) Histogram of the errors over all interpolated state vectors. (b) The true and interpolated solution at $Re_{max,error}$.

Figure 33: Overview of the results from experiment MQ.C.3.

For a substantial number of state vectors, the interpolation error is above the interpolation threshold of 1%, but the average is well below γ_{int}. When looking at the interpolated solution it can be see that the interpolated solution has some wave like noise with a small amplitude, which could point to a relatively high error in the higher order modes or the coefficients of these modes. To know how the error evolves over time Figure 34 shows a 3D histogram where the counts are given for a certain error to occur in period of time.

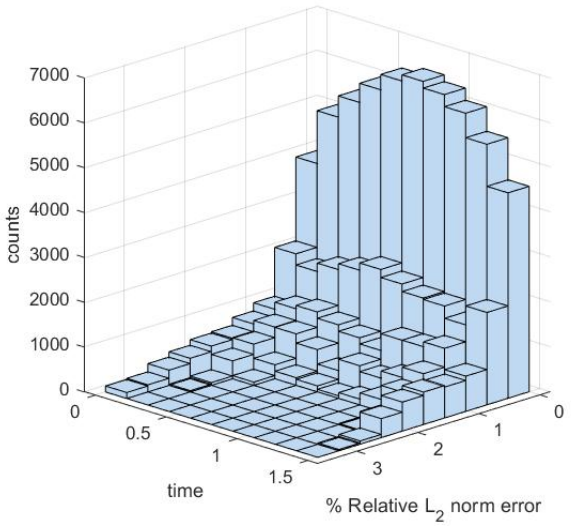


Figure 34: 3D histogram that shows the counts of a certain error to occur in a period of time.

The highest errors occur at the beginning of the time evolution. This high error can be explained by looking at how the high order modes are interpolated as these are necessary to approximate the sharp wavefront at the start of the time evolution. Therefore, in Figure 35, the relative maximum L2 errors are plotted for the individual columns in the interpolated **U**, **C** and **V** matrices.

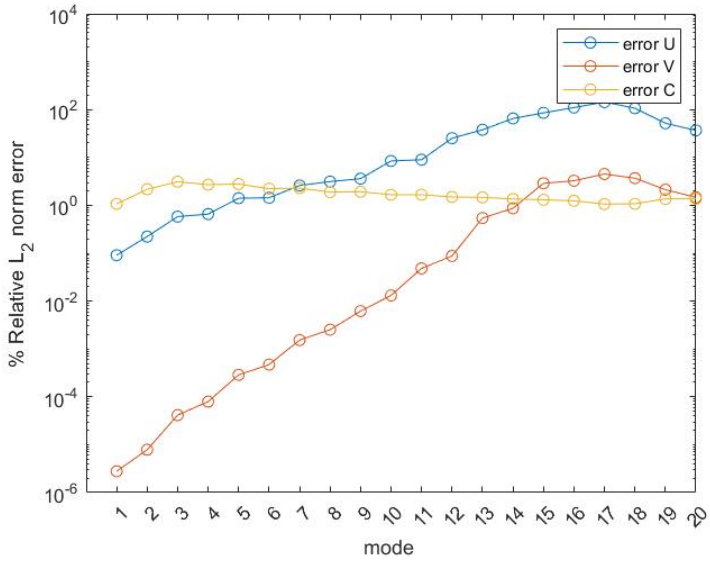


Figure 35: The relative L2 errors are plotted for the individual columns in the interpolated **U**, **C** and **V** matrices of the solution with the highest error $\epsilon_{L_2,max}$. mode on the x-axis indicates the index of the column in these matrices.

According to expectations, for both the modes and the coefficients, it is more difficult to interpolate the higher order non-linear modes and coefficients than the low-order ones. The interpolation of the modes is more difficult than the interpolation of the coefficients, which was expected from the mode analysis of the Burgers equation. It is also interesting to note that for the Burgers equation the interpolation error of the coupling coefficients does not increase with the order of the modes. This could indicate that each of the coupling coefficients in the coupling matrix **C** might have a different dependence regardless of the order of the column or mode that is looked at. The interpolated modes 11 until 14 and the respective coefficients are plotted in Figure 36.

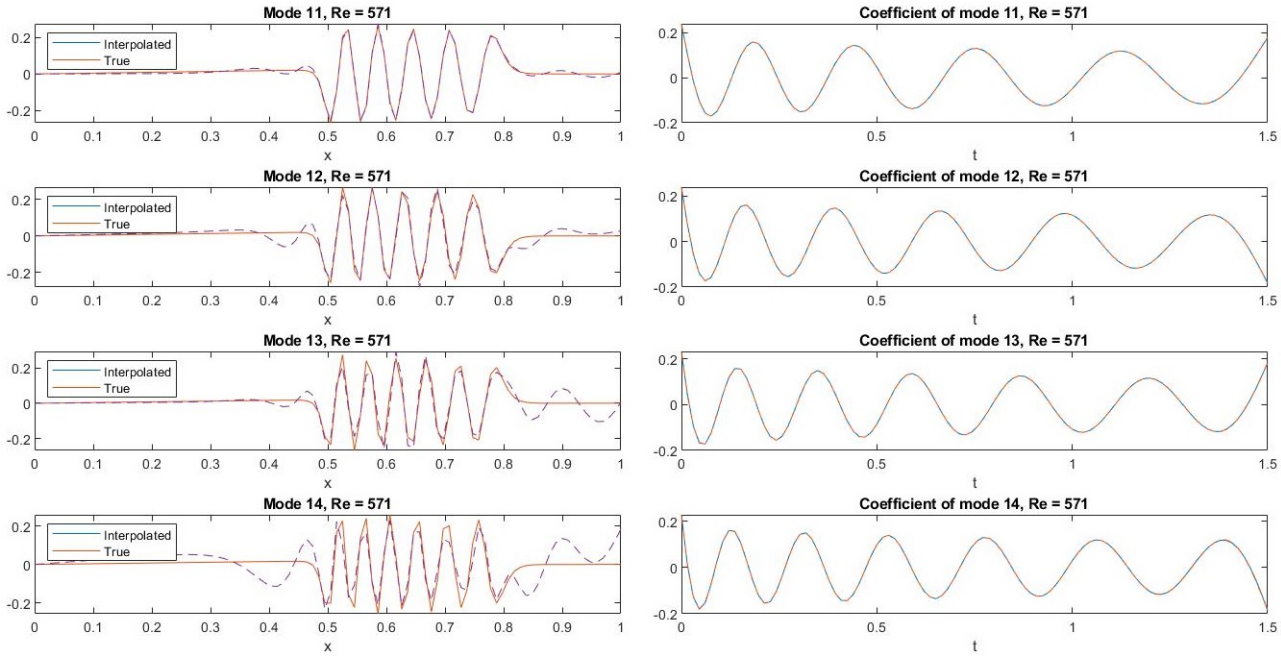
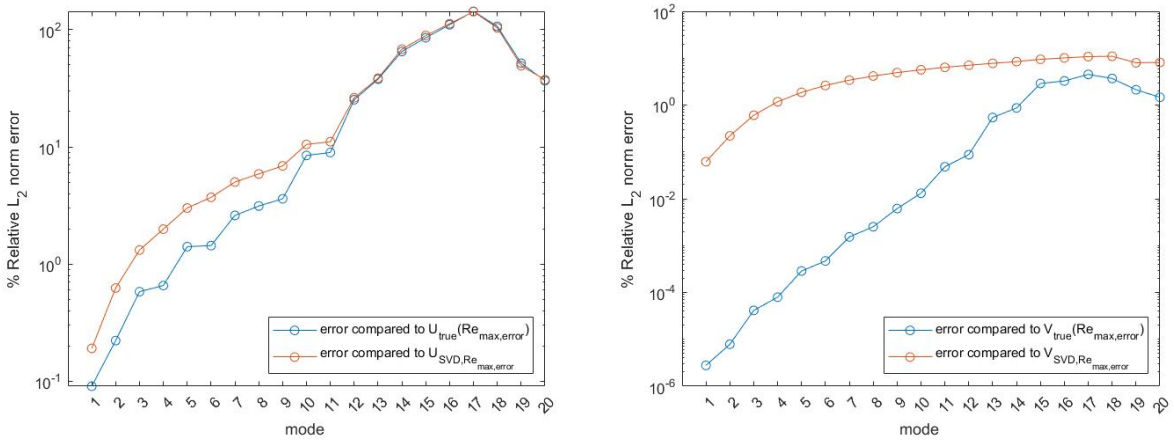


Figure 36: Plots of the true and interpolated modes 1 to 14 with the respective coefficients of the interpolated of the interpolated solution with the highest error $\epsilon_{L_2,max}$.

As the order of the mode increases, the modes and coefficients become more nonlinear, and it

becomes harder to accurately interpolate them. From Equation (54) and (55), it was explained that the resulting orthonormal matrix that comes out of the exponential map is an orthogonally transformed version of the same matrix. This can be substantiated by comparing the interpolated columns of $\mathbf{U}(Re_{max,error})$ and $\mathbf{V}(Re_{max,error})$ of the solution with highest error to the columns of the true matrices $\mathbf{U}_{SVD,Re_{max,error}}$ and $\mathbf{V}_{SVD,Re_{max,error}}$ that come out of the SVD and the true columns that come out of the exponential map $\mathbf{U}_{true}(Re_{max,error})$ and $\mathbf{V}_{true}(Re_{max,error})$. These errors are plotted in Figure 37.



(a) Interpolation error of $\mathbf{U}(Re_{max,error})$.

(b) Interpolation error of $\mathbf{V}(Re_{max,error})$.

Figure 37: Comparison of the error between the interpolated modes and coefficients and the true modes and coefficients from the SVD and the exponential map.

The errors are not identical, as $\mathbf{U}_{true}(Re_{max,error})$ and $\mathbf{V}_{true}(Re_{max,error})$ are orthogonally transformed compared to $\mathbf{U}_{SVD,Re_{max,error}}$ and $\mathbf{V}_{SVD,Re_{max,error}}$. The error difference becomes smaller and smaller for higher order modes and coefficients, which is unexpected. As it is not of great importance for the overall performance of the algorithm, and as this effect is just a byproduct of the exponential map, this will not be brought up in the discussion.

To check whether the distance of a tangent plane to a point of interpolation interest influences the interpolation performance, 2 extra experiments were done where the minimum of the error tolerance range $\epsilon_{tol,min}$ is neglected. In this case, the lowest number of modes was chosen that results in an approximation error lower than $\epsilon_{tol,max}$. The cubic RBF is used, $[\epsilon_{tol,min}, \epsilon_{tol,max}]$ is set to $[0, 9.99 \cdot 10^{-3}]$, $\theta_{1,tol}$ is set to $\frac{\pi}{2}$ and γ_{int} is set to $1 \cdot 10^{-2}$. In this case, the first modes is taken that results in an approximation error lower than $\epsilon_{tol,max}$. The resulting grid is shown in Figure 38.

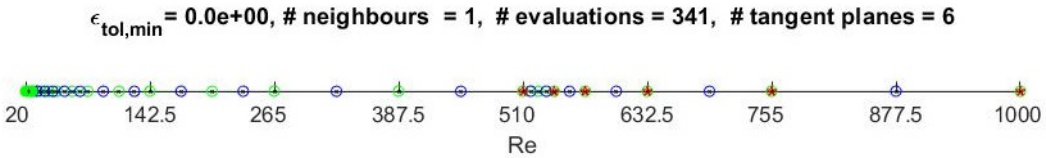


Figure 38: Resulting sampled grid of an experiment of the Burgers equation where $\epsilon_{tol,min}$ is neglected and $\epsilon_{tol,max}$ equals $9.9 \cdot 10^{-3}$.

The algorithm was not able to converge, as it kept sampling points at the left edge of the domain. As expected, there are no tangent planes on the left half of the domain, meaning that only the tangent plane at $Re = 510$ is used to interpolate in the left part of the domain. However, what is interesting to see is that despite having such a far placed tangent plane relative to the left edge of the domain, the interpolation method can still accurately interpolate to a Reynolds number of around 30. Comparing the inactive points in Figure 31 to the inactive points in Figure 38,

it can be seen that from Reynolds number 30 and onward, the same points are labeled inactive. This indicates that including more higher order modes, which are not numerical noise, does not deteriorate the interpolation. The fact that no tangent plane was drawn on the left side also indicates that the first principal angles were below $\frac{\pi}{2}$. In the second experiment $\epsilon_{tol,max}$ is reduced to $9.9 \cdot 10^{-4}$. This is done to make sure that the number of modes chosen at the tangent plane at $Re = 510$ is within numerical noise range at the left part of the domain. The resulting sparse grid is shown in Figure 39.

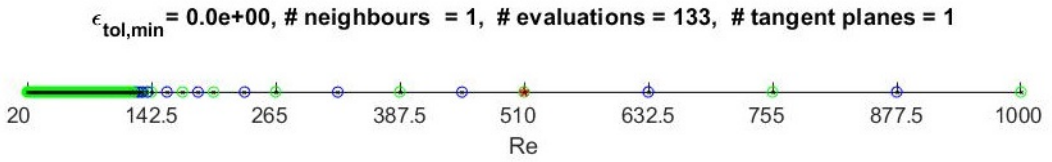


Figure 39: Resulting sampled grid of an experiment of the Burgers equation where $\epsilon_{tol,min}$ is neglected and $\epsilon_{tol,max}$ equals $9.9 \cdot 10^{-4}$.

The algorithm also did not converge in this case, as it kept sampling points on the left side of the domain. Only one tangent plane was drawn in this case, meaning that the number of modes needed to stay within the approximation range $[\epsilon_{tol,min}, \epsilon_{tol,max}]$ does not change on the right side of the domain. At this tangent plane 21 modes were needed, which is well within numerical noise range according to Figure 23. The number of points being marked important span a much larger part of the Reynolds number domain compared to Figure 38. This indicates that numerical noise does deteriorate the performance of the ITSGM method. In Figure 40, plots are given of the horizontal lifts $\mathbf{Z}^U$ at $Re = 20$ from the 2 experiments.

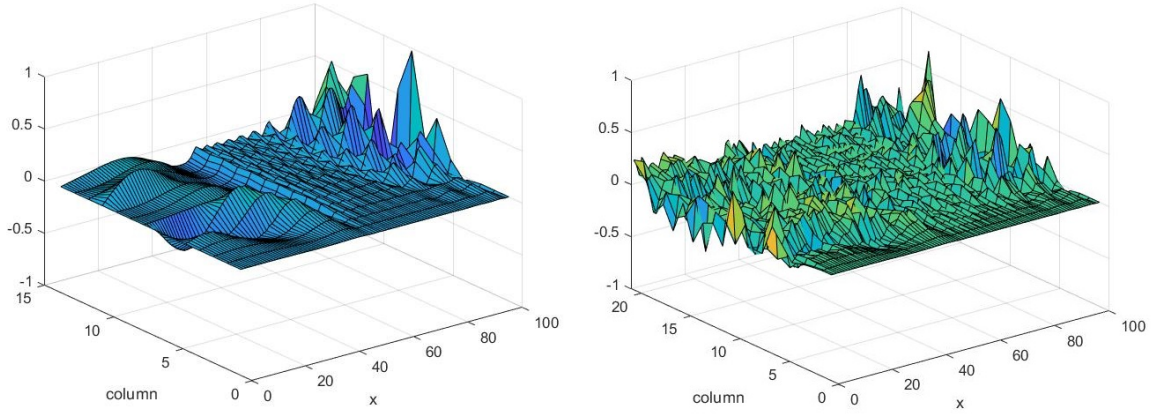


Figure 40: Plots of the values in the matrices representing the horizontal lifts $\mathbf{Z}^U$ at $Re = 20$ for a tangent plane at $Re = 510$. $\epsilon_{tol,max}$ is set to $9.99 \cdot 10^{-3}$ and $9.99 \cdot 10^{-4}$ for left and right plot respectively.

.

There is substantially more noise in the horizontal lift that was calculated from the tangent plane with 21 modes compared to the one with 14 modes. The noise is also present throughout the whole matrix, instead of only up until column 14 which is equal to the number of modes at which noise occurs at $Re = 20$. It seems that the noise in the modes carries over to the horizontal lifts, which causes the interpolation of $\mathbf{Z}^U$ at the left side of the domain to fail. It is also expected that this occurred in the horizontal lift $\mathbf{Z}^V$. In Figure 41 the horizontal lift $\mathbf{Z}^U$ at $Re = 23.82$ for a tangent plane at $Re = 20$ is shown from experiment MQ.A.1.

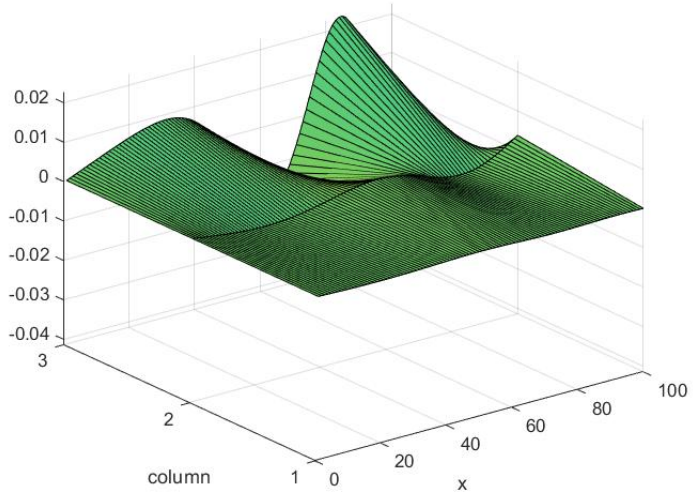


Figure 41: Plot of the values in the matrix representing the horizontal lift $\mathbf{Z}^U$ at $Re = 23.82$ for a tangent plane at $Re = 20$.

There is no noise in any of the columns in the horizontal lift. In Figure 42 the result of an experiment is shown where the principal angle tolerance was set to 0.4999π, which led to the grid in Figure 42.

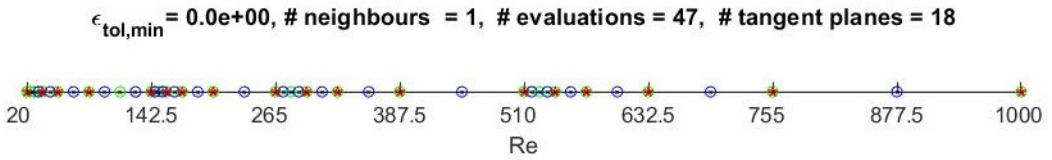


Figure 42: Resulting sampled grid of an experiment of the Burgers equation where $\epsilon_{tol,min}$ is neglected and $\theta_{1,tol}$ is set to 0.4999π.

In this case the algorithm did converge. The first noticeable thing is that far more tangent planes were drawn than any of the experiment in Table 4. This is mainly caused by the fact that that the principal angle at $Re = 20$ gets extremely close (1.5707π) to the value of $\frac{\pi}{2}$ for a very small difference in Reynolds number. Then, because of the tangent plane at $Re = 20$, the rest of the tangent planes were then drawn due to a difference in compatible number of modes. This is also substantiated by the fact that there are small regions where there are multiple tangent planes to the right of the higher hierarchical tangent planes. This happens because, to the right of the higher hierarchical tangent planes, the number of modes will never be enough as the Reynolds number keeps increasing, meaning that more modes are needed to stay below $\epsilon_{tol,max}$. This is a direct result of only taking the lowest number of modes that results in an error below $\epsilon_{tol,max}$, instead of more modes.

4.2.2 Results of the experiments on the Molenkamp test

The errors for the experiments done on the Molenkamp problem are tabulated in Table 6. The interpolation accuracy tolerance and the maximum L_2 error over the interpolated state vectors $\epsilon_{L2,max}$ are provided. As it is a problem with a 5D input parameter space, more attention is put on the number of evaluations, the ratio of important points and the number of interpolated state vectors where the interpolation accuracy is higher than the tolerance γ_{int}.

Table 6: Results of the experiment done on the Molenkamp test.

Experiment	γ_{int}	$\epsilon_{L2,max}$ [%]	# evaluations	% important points	% state vectors with error $> \gamma_{int}$
Smooth	$1 \cdot 10^{-3}$	2.19	2957	81.1	10.6
Steep.A	$1 \cdot 10^{-3}$	2.77	3833	83.4	9.95
Steep.B	$5 \cdot 10^{-4}$	0.56	5086	85.3	9.3

When comparing the results to the ones that were achieved with the algorithm based on the global basis in [4], it can be noted that for an interpolation accuracy tolerance γ_{int} of $1 \cdot 10^{-3}$, the maximum relative L_2 error is much higher. More specifically, for the smooth setting, the algorithm with the global basis only sampled 1379 different parameter combinations and achieved a maximum relative L_2 error of 0.17%, while the algorithm based on the local basis achieved an error of 2.19% with 2957 evaluations, which is more than twice as many. For the results of the smooth experiment, the percentage of important points is the lowest, while the percentage of state vectors where the interpolation error was higher than the tolerance γ_{int}, is the highest. Moreover, for the experiment Steep.A the number of 3833 evaluations is much lower than the number of 5086 evaluations in the algorithm based on the global basis. However, the error achieved with the algorithm based on the global basis was 0.33% which is much lower than the error of 2.77% achieved by the algorithm based on the local bases. In experiment Steep.B where a lower interpolation accuracy tolerance γ_{int} was used, roughly the same number of 5086 evaluations was achieved as the 5093 evaluations with the algorithm based on the global basis. It is hard to say for 1000 check points in a 5D parameter domain, whether these error results are an accurate representation of the full interpolation error over the full parameter domain. Moreover, in [4] no values were given regarding the percentage of important points and percentages of interpolated state vectors with an error higher than γ_{int}. Nevertheless, in this setting it can be said that the difference in interpolation error of 0.59 and 0.33%, for these 1000 points is small enough to say that these algorithms can perform similarly well in the steep setting of the Molenkamp problem. 85.3% of the parameter combinations were regarded as important, which is the highest out of all the experiments and that the percentage of interpolated state vectors with an error lower than γ_{int} is the highest, which is expected as the interpolation accuracy threshold is stricter. Moreover, the number of state vectors with an interpolation error higher than the threshold γ_{int} is also the lowest. The interpolated solution with the highest L_2 error in the smooth setting was at $t = 0.8$ and is shown in Figure 43.

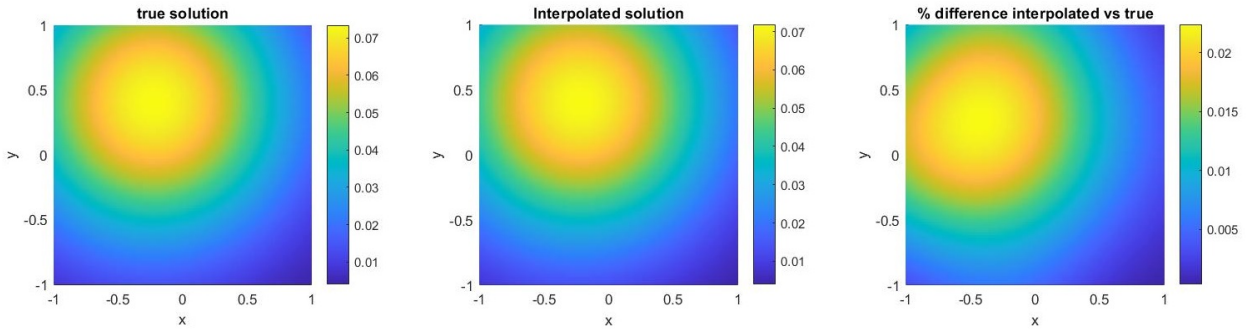


Figure 43: The true and interpolated solution of the check point with the highest interpolation error for the smooth Molenkamp problem are plotted on the left and middle respectively, The relative percentage difference for each coordinate is also plotted on the right.

On the right of Figure 43, the relative percentage error from Equation (79) is used for the entries of the interpolated state vector with the highest error. The overall difference between the true and interpolated solution is at maximum 0.025%. It should be noted that this error is lower than the L_2 error as it is calculated on a single coordinate compared to the L_2 error which is calculated over the full solution. From this it can be stated that overall, the interpolation is done correctly. The error is shaped differently compared to both solutions. To see what causes this, the interpolation error of the individual matrices $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{C}$ of the interpolated solution with the highest error in the smooth setting is given in Figure 44. The interpolation error of the columns in the coupling

matrices is higher compared to the error of the interpolated modes or the columns representing the coefficients in **V**. The error does not increase as a function of the mode order. The reason for this is that when looking at Figure 25 and A1, the modes are quite similar in terms on smoothness, as these modes have a similar function of representing the rotational movement of the gas cloud. It is slightly harder to interpolate the modes compared to the coefficients in Figure 44. From this it can be deduced that the modes and coefficients in **V** were interpolated successfully in the smooth setting.

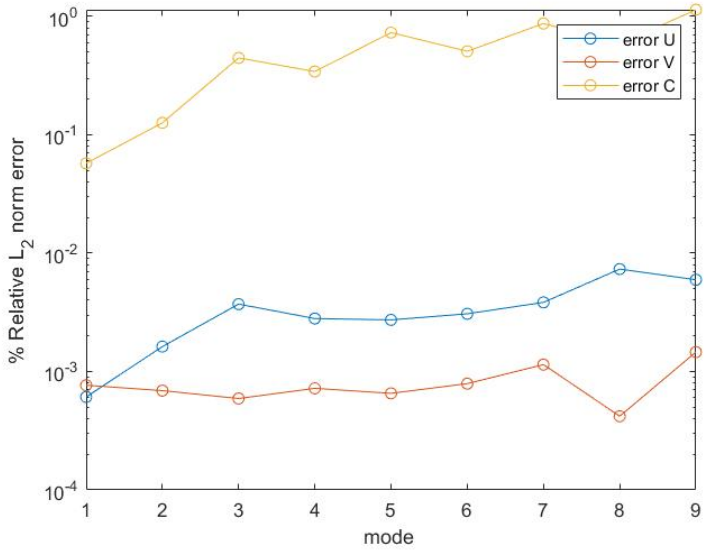


Figure 44: The relative L2 errors are plotted for the individual columns in the interpolated **U**, **C** and **V** matrices of the solution with the highest error $\epsilon_{L_2,max}$. mode on the x-axis indicates the index of the column in these matrices.

For confirmation that the modes and coefficients are interpolated successfully, the true and interpolated modes 1-4 and modes 5-8 are plotted with respective percentage differences in Figures 45, 46 and A7 in the Appendix respectively.

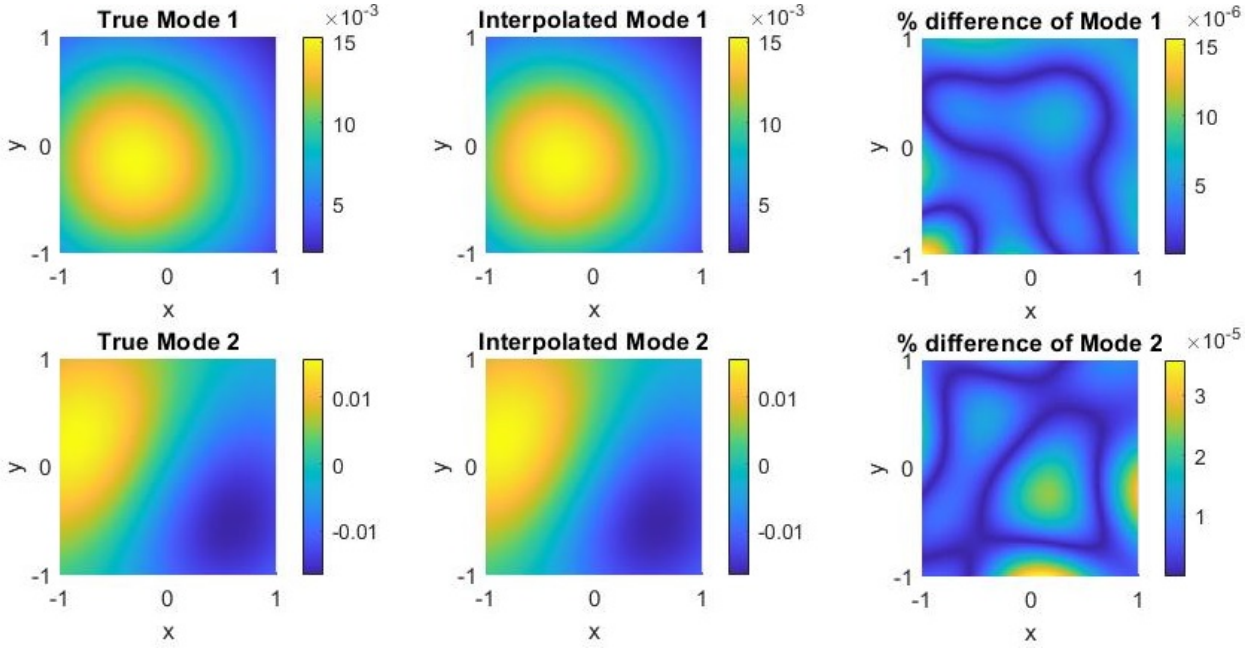


Figure 45: Plots of the true and interpolated modes 1 and 2 of the interpolated solution with the highest error $\epsilon_{L_2,max}$. The percentage difference between the two is plotted on the right.

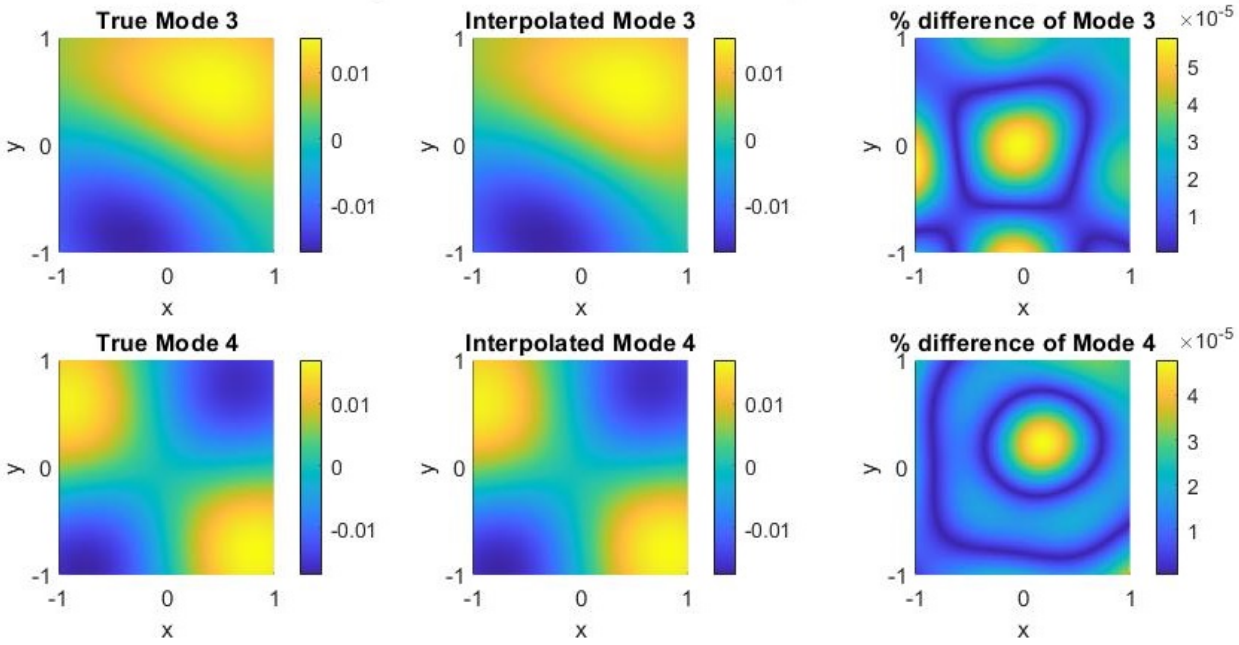


Figure 46: Plots of the true and interpolated modes 3 and 4 of of the interpolated solution with the highest error $\epsilon_{L_2,max}$. The percentage difference between the two is plotted on the right.

The percentage differences for all modes are much smaller compared to the error of the solution itself. The same high interpolation accuracy can be seen in the interpolation of the coefficients of these modes in Figure 47 and A8. From this it can be stated that the modes and coefficients are interpolated successfully in the smooth setting of the Molenkamp problem.

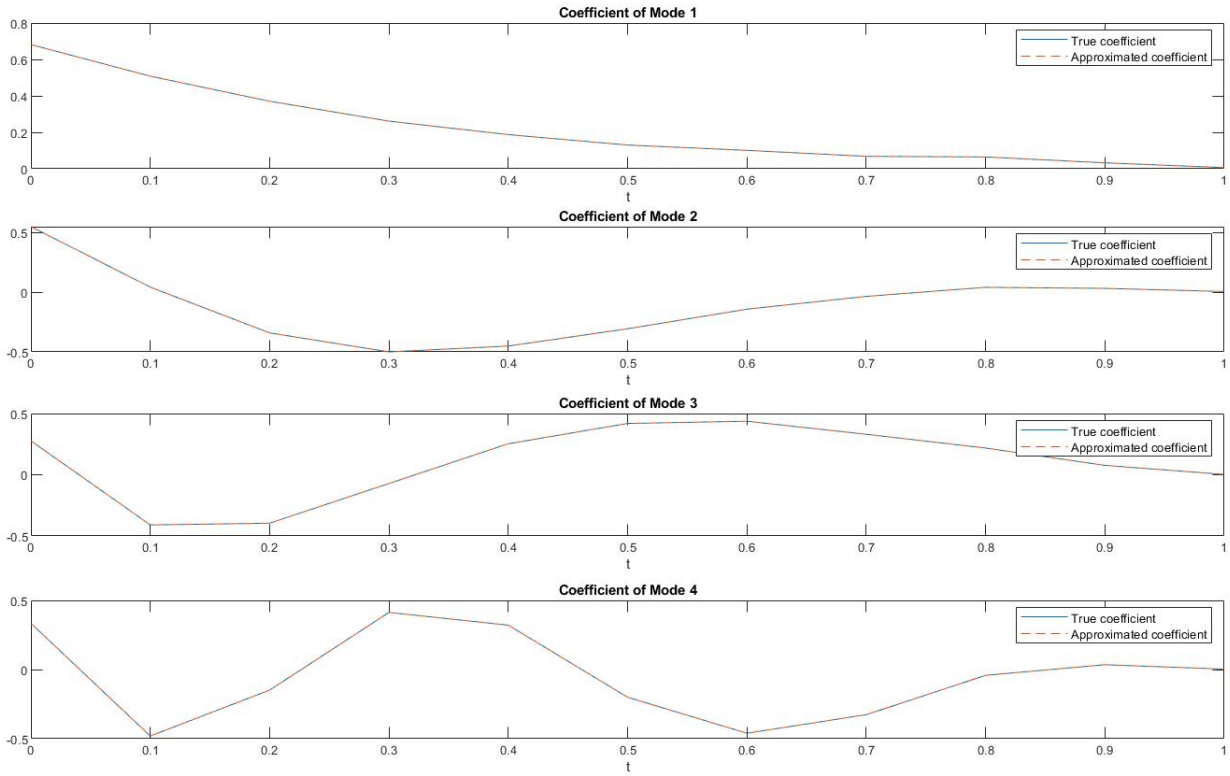


Figure 47: Plots of the true and interpolated coefficients of modes 1 to 4 of the interpolated solution with the highest error $\epsilon_{L_2,max}$.

An analysis must be made of the relatively large error of the coupling matrix coefficients. First, as the coupling matrix is approximately similar to the singular value matrices, in which all the off-diagonal elements are zero, it could be argued that the errors at the diagonal elements overrule the error at the off-diagonal elements in the L_2 error calculation. Therefore, a plot is made of the L_2 error of the columns in the interpolated coupling matrix from Figure 44 and of the columns in the same interpolated coupling matrix where the diagonal elements are set equal to the diagonals elements in the true coupling matrix. This plot is shown in Figure 48.

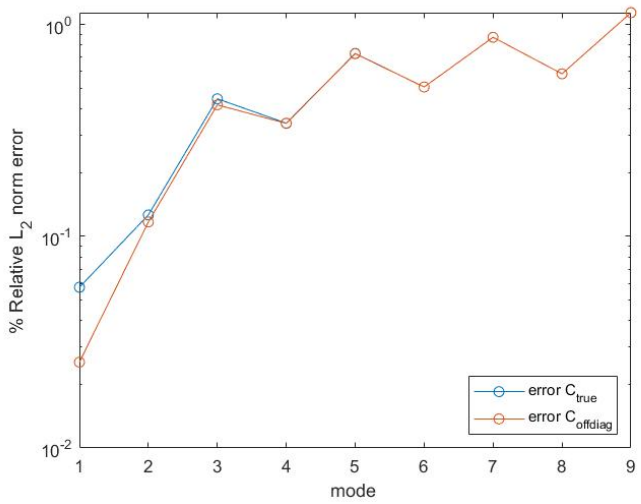


Figure 48: The L_2 error of the columns in the interpolated coupling matrix from Figure 44 and the L_2 error of the columns in the same interpolated coupling matrix, where the diagonal elements are set equal to the true coupling matrix.

There is a big difference in the L_2 error of the very first column in the coupling matrix. Moreover, this difference decreases with the order of the mode. This is expected as the singular values are ordered in decreasing values along the diagonal of $\mathbf{\Sigma}$, so it expected that this holds for the coupling coefficients in general as well. Therefore, this leads to a relatively larger error for the lower order modes than higher order modes. This indicates that, except for the first mode, the off-diagonal elements are the issue. It is also interesting to see how the interpolation error of the coupling coefficients would translate to the error of the amplitudes of the modes, if the coupling matrix were to be multiplied with the matrix $\mathbf{V}$. These amplitudes are basically the POD coefficients. This is analyzed by looking at the percentage difference of the coupling coefficients and the resulting POD coefficients when multiplying with the interpolated $\mathbf{V}$ matrix. This error is plotted in Figure 49.

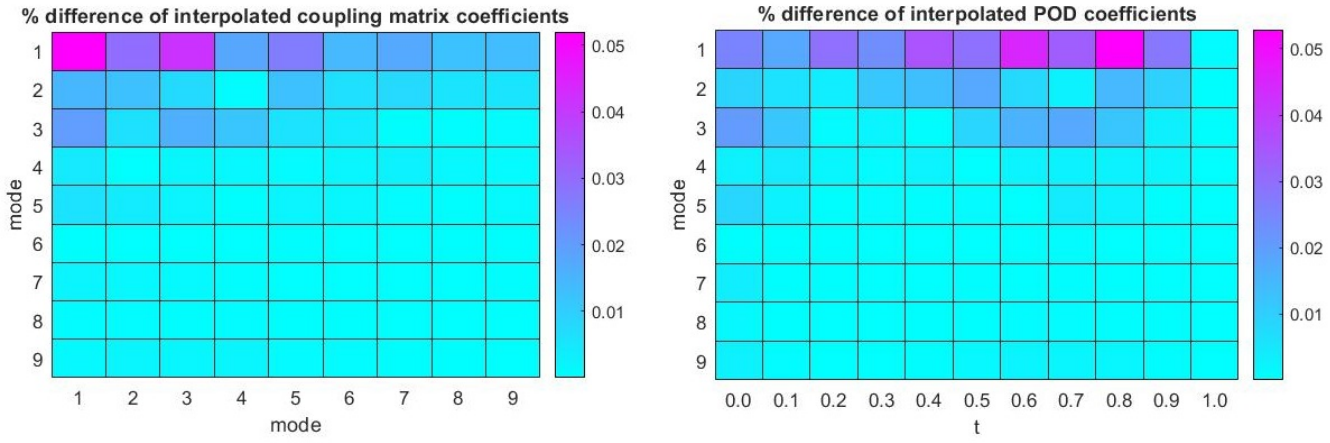


Figure 49: The percentage difference of the coupling coefficients and the resulting POD coefficients when multiplying with the interpolated $\mathbf{V}$ matrix, of the interpolated solution with the highest error in the smooth setting.

At $t = 0.8$, the error of the POD coefficients of mode 1 is relatively high, which could substantiate why the shape of the percentage error in Figure 43 is the same cloud that is deformed to the left lower side. Next, the plots of experiment Steep.A are given. The interpolated solution with the highest error in the steep setting is at $t = 0.8$ and is given in Figure 50.

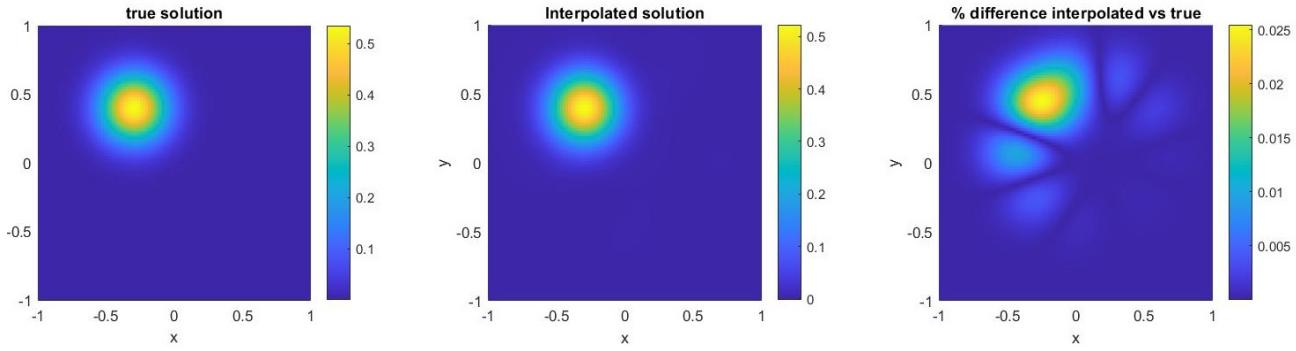


Figure 50: The true and interpolated solution of the check point with the highest interpolation error the steep Molenkamp problem are plotted on the left and middle respectively, The relative percentage difference for each coordinate is also plotted on the right.

The overall shape of the interpolated solution is correct, and the percentage difference is quite small. Also the rotating wavelike pattern of the modes appears in the error. To understand where this comes from, the error of the interpolation of $\mathbf{U}$, $\mathbf{C}$ and $\mathbf{V}$ is analyzed, which is plotted in Figure 51.

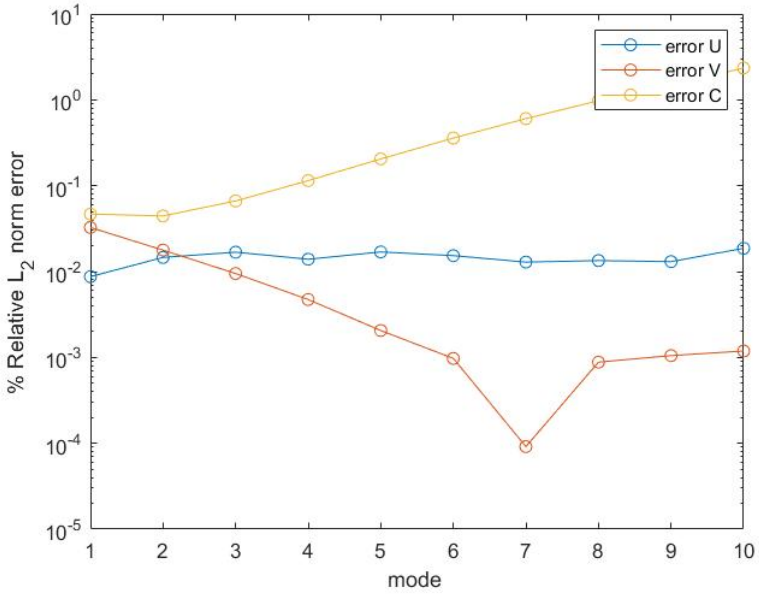


Figure 51: The relative L_2 errors are plotted for the individual columns in the interpolated $\mathbf{U}$, $\mathbf{C}$ and $\mathbf{V}$ matrices of the solution with the highest error $\epsilon_{L_2,max}$. mode on the x-axis indicates the index of the column in these matrices.

The error of the interpolation of the modes does not vary too much and the error of the coefficients is lower than the error of the modes, which is very similar to the smooth case in Figure 44. The error of the coupling coefficients again becomes larger for higher order modes. From this, it can be deduced that the modes and coefficients in $\mathbf{V}$ were also interpolated successfully in the steep setting. For confirmation that the modes are interpolated accurately, the higher order modes 5-8 are plotted in Figure 52.

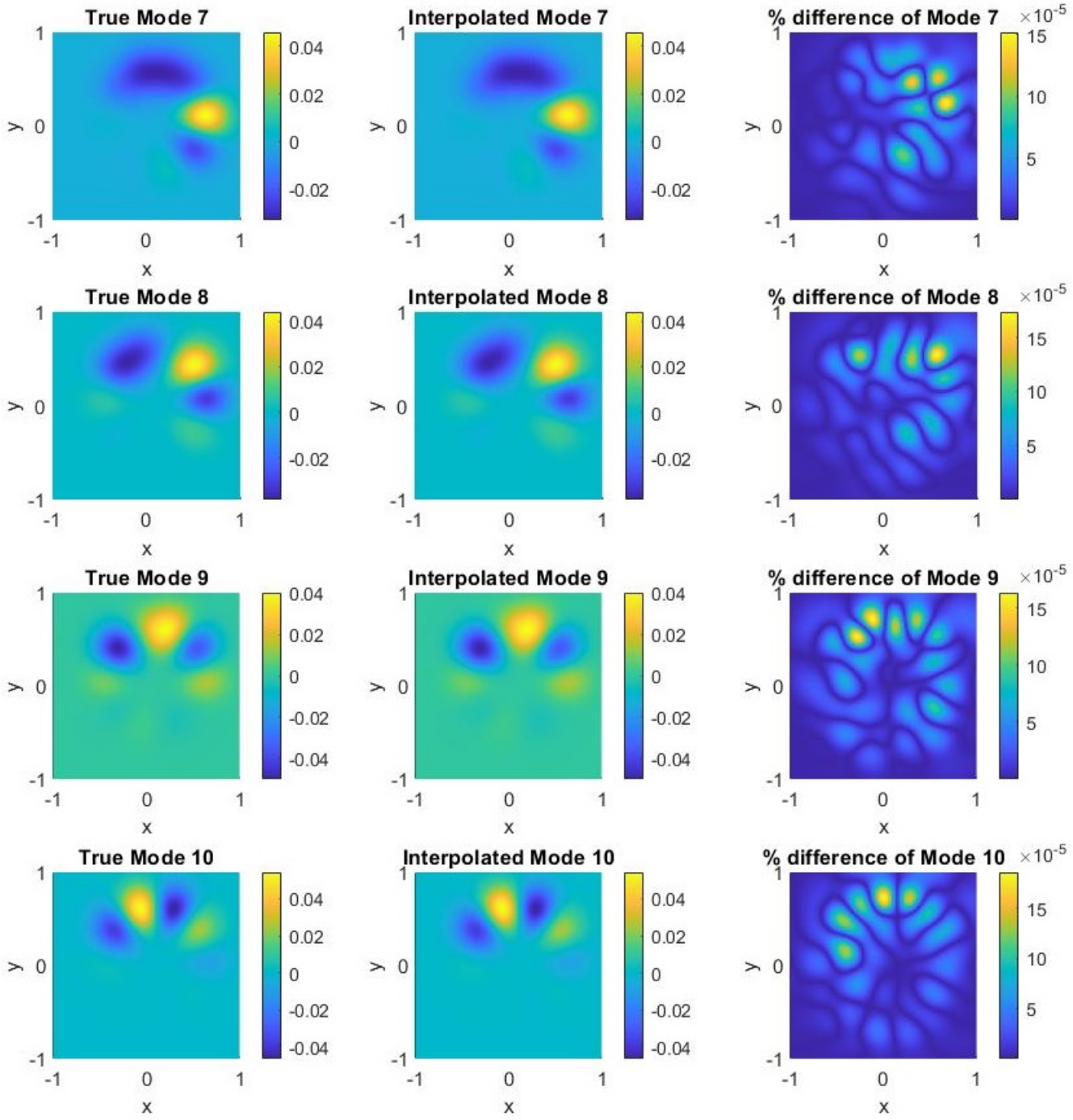


Figure 52: Plots of the true and interpolated modes 7 to 10 of of the interpolated solution with the highest error $\epsilon_{L_2,max}$. The percentage difference between the two is plotted on the right.

There is a rotational wavelike pattern in the modes, which was expected from the mode analysis, and the percentage error is again very small. The spherically rotating shape in the error of the solution in Figure 50 can again be substantiated by looking at the percentage difference of the coupling coefficients and the resulting POD coefficients when multiplying with the interpolated **V** matrix. These are plotted in Figure 53.

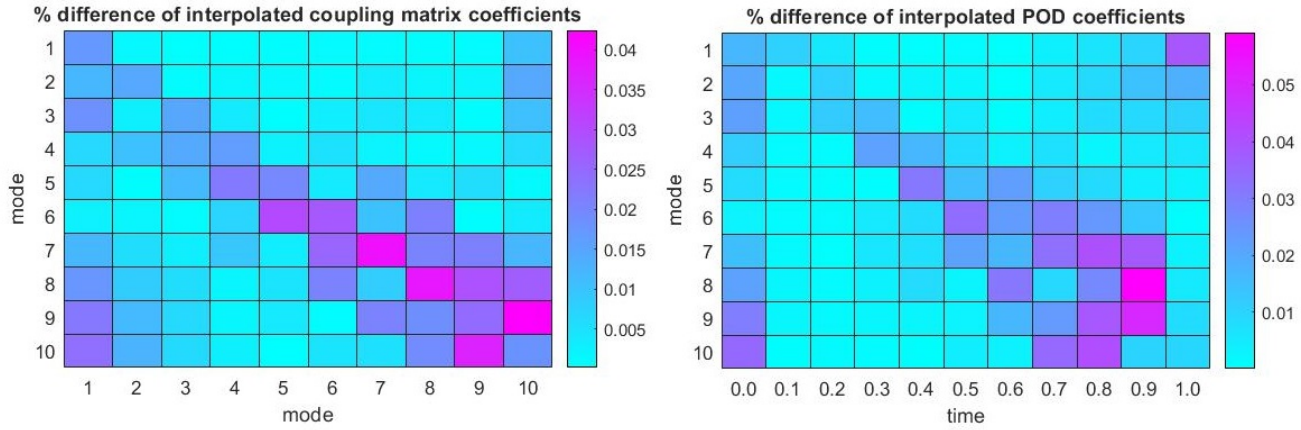


Figure 53: The percentage difference of the coupling coefficients and the resulting POD coefficients when multiplying with the interpolated $\mathbf{V}$ matrix of the interpolated solution with the highest error in the Steep.A experiment.

The error of the POD coefficients of modes 6 to 10 is high at $t = 0.8$, which is the time at which the error of the physical solution was highest. From this, it can be deduced that the reason for the shape of the percentage error in Figure 50 where a rotating wave can be seen in the left upper corner, is the inaccurate amplitudes of modes 6 to 10 in Figure 45, which represent the rotating motion in the left upper corner of the domain. The last result to discuss is how many unique parameter values were sampled in each separate parameter dimension. The unique amounts for each parameter for all the experiments are tabulated in Table 7.

Table 7: Number of unique values per parameter that were sampled in various experiments on the Molenkamp problem.

Parameter	Number of unique values in Smooth setting	Number of unique values in Steep.A setting	Number of unique values in Steep.B setting
λ_1	17	17	19
λ_2	13	15	17
λ_3	19	19	23
λ_4	13	17	17
λ_5	13	17	17

The unique values achieved with the algorithm based on local bases are very different compared to the ones achieved with the global basis in the time adaptive mode in [4], for both the smooth and steep settings. As in the smooth setting much more evaluations were done with the algorithm based on local bases, it speaks for itself that the number of unique values is much higher in general and that is hard to compare the 2 algorithms. This also includes experiment Steep.A, in which much less evaluations were done, but a higher error was achieved in the local bases case. However, the results of experiment Steep.B can be used for comparison as in this case both achieved an almost similar error with a very small difference in the total number of evaluation. More specifically, for the linear parameter λ_1 and the steepness parameter λ_2, the algorithm based on the global basis only sampled 3 and 13 unique values, while the algorithm based on the local bases sampled 19 and 17 unique values in experiment Steep.B. However, the number of points sampled for the decay constant λ_3 of 19 and 23 is much lower than the 33 unique values that were needed in the global case. Lastly, the number of unique values for the initial rotation parameters λ_4 and λ_5, are identical between the two algorithms. The reason for this large difference is clarified in the discussion.

4.2.3 Results of the experiments on the 2D neutron diffusion problem

The errors for the experiments done on the neutron diffusion problem are tabulated in Table 8.

Table 8: Results of the experiment done on the Neutron diffusion problem.

Experiment	$\epsilon_{L_2,max}$ [%]	# evaluations	% important points	# tangent planes	% state vectors with error > γ_{int}
S.σ_f.A	1.47	41	51.2	1	11.3
S.σ_f.B	4.66	97	40.2	1	19.0
S.σ_f.C	2.27	71	43.7	3	25.1
S.σ_c	2.72	169	44.4	8	11.3

For an interpolation threshold γ_{int} of $5 \cdot 10^{-3}$ the highest and lowest achieved maximum L_2 errors over all experiments is 4.66 % and 1.47%. For a set interpolation tolerance γ_{int} of $5 \cdot 10^{-3}$, the maximum errors achieved in experiment S.σ_f.A, S.σ_f.C and S.σ_c are acceptable. This is because the error of the flux estimation is small enough to determine how the flux generally behaves for a new parameter and striving for a higher accuracy would not be worth the extra numerical simulations that would have to be made. This acceptance range is entirely problem dependent, where in different problems a much higher accuracy would be required. The percentage of interpolated state vectors with an error larger than γ_{int} in S.σ_f.C is 25.1%, which is quite high. There is a difference in achieved error between experiment S.σ_f.B and S.σ_f.C, which only differ in the set tolerance for the first principal angle when drawing new tangent planes. As a result of this tolerance difference, a higher number of tangent planes were drawn, thereby producing smaller subdomains in which the multiquadric RBF interpolator performs better. To elaborate, the error reduced from 4.66% to 2.27%, while simultaneously reducing the number of evaluations from 97 to 71. Judging from the error in experiment S.σ_f.C and S.σ_c, the hierarchical placement of the tangent planes and the overlapping of the subdomains was a success and that the interpolation within these subdomains led to an acceptable result. This can be substantiated by looking at the resulting sparse grids of experiments S.σ_f.B and S.σ_f.C and the error distribution of both experiments which are plotted below.

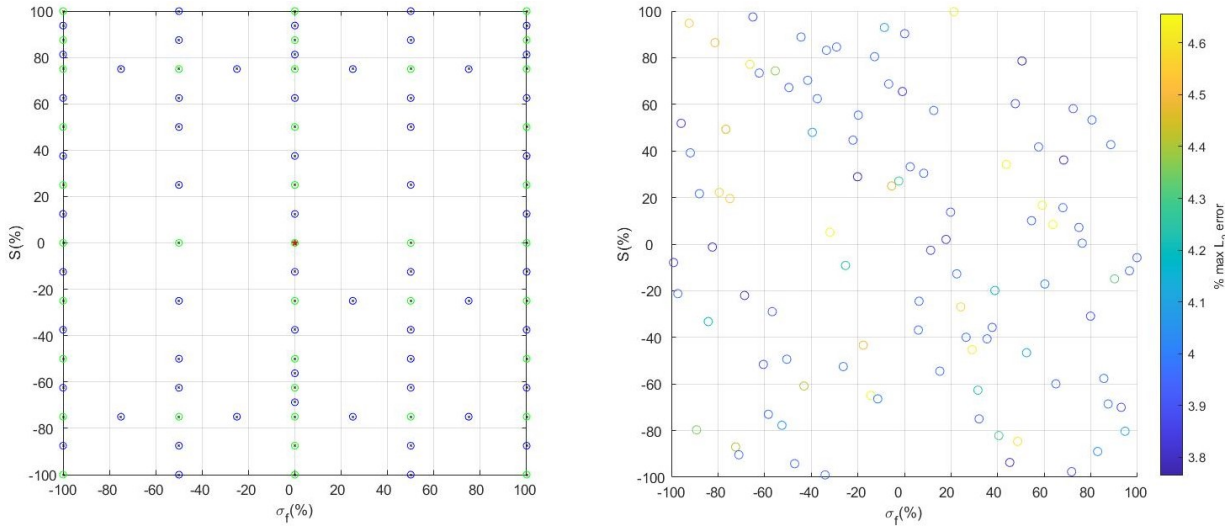


Figure 54: The resulting sparse grid of experiment S.σ_f.B on the left, with the error distribution on the right. The green points are the important points that are included in the ROM, the blue points are the inactive points where the interpolation error was low enough and the red points indicate a tangent plane.

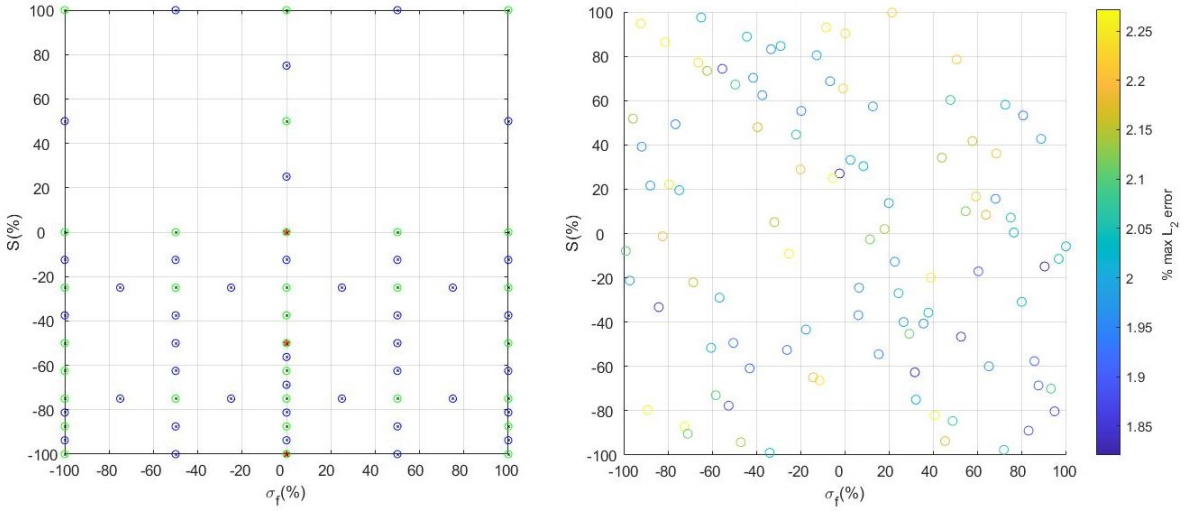


Figure 55: The resulting sparse grid of experiment S.σ_f.C on the left, with the error distribution on the right. The green points are the important points that are included in the ROM, the blue points are the inactive points where the interpolation error was low enough and the red points indicate a tangent plane.

2 extra tangent planes were placed in the lower half of the domain. This is expected as the overall solution changes the most when changing from zero source to positive source in Figure 22. As a result of the early added tangent planes in the lower half of the parameter domain, the resulting errors in the upper half were lower due to a smaller upper subdomain. This led to less points having to be sampled in the upper half of the domain than in the lower half in Figure 55 compared to Figure 54. The error distribution is mostly uniform, where there is no clearly visible difference in the error that is achieved in the different subdomains in Figure 55. From this, it can also be stated that the interpolation in the different subdomain is done successfully. The solution with the highest errors for experiment S.σ_f.B is plotted in Figure 56.

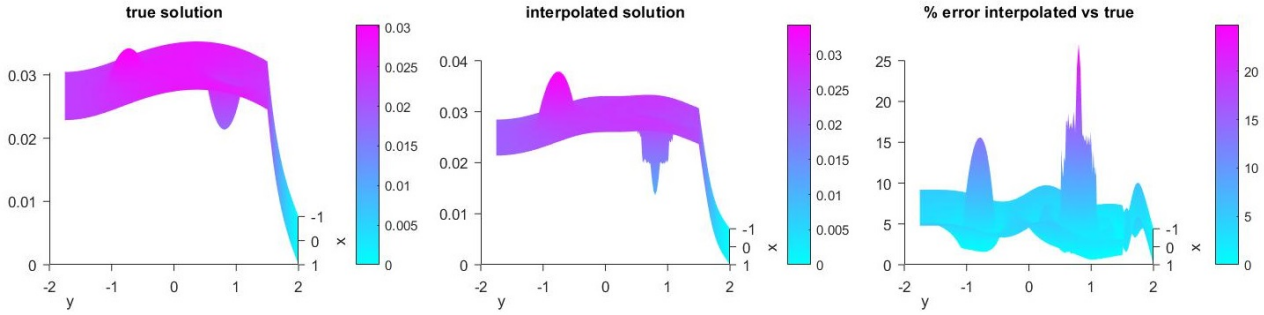


Figure 56: The true and interpolated solutions of the flux with the highest interpolation error from experiment $S.\sigma_f.B$ are plotted on the left and middle respectively. The relative percentage difference for each coordinate is also plotted on the right.

The overall shape of the interpolated solution is correct. However, the error at the fuel and the absorber is high, and the flux minimum at the absorber is much sharper than the true solution. The high error at the fuel and absorber is expected as the flux changes more abruptly in these regions compared to the flux in the moderator. There also seems to be some noise in the interpolated solution, which could be caused by the overall noise in the modes themselves. The reason as to why the error and noise occur, will be made clear later when the interpolated modes and coefficients are analyzed. The solution with the highest errors for experiment S.σ_f.C is plotted in Figure 56.

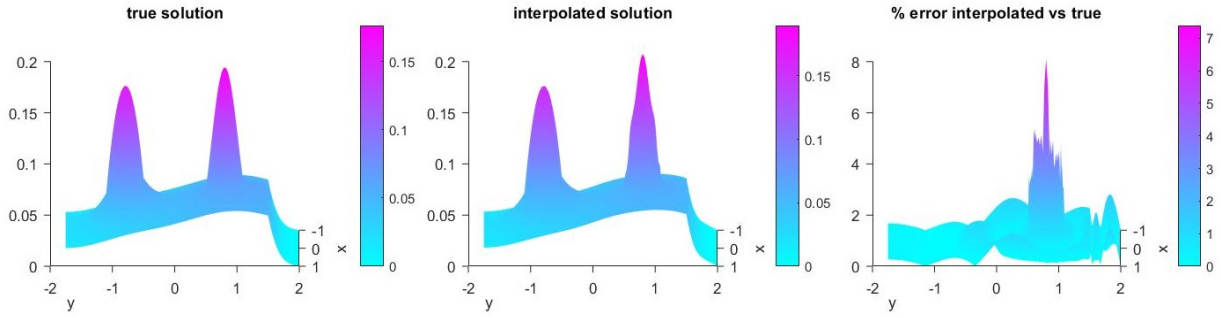


Figure 57: The true and interpolated solutions of the flux with the highest interpolation error from experiment $S.\sigma_f.C$ are plotted on the left and middle respectively. The relative percentage difference for each coordinate is also plotted on the right.

The overall shape of the interpolated solution is again correct, and the interpolation of the final solution was more accurate than in experiment S.σ_f.B. However, a small part of the flux peak at the absorber is sharper than the true solution. There is no noise in the interpolated solution compared to the interpolated solution of experiment S.σ_f.B. As the percentage of state vectors with an error higher than γ_{int} is quite high, a histogram is plotted showing the counts of errors in different parts of the time evolution in Figure 58. For both problems the highest errors occur mostly in the range of $t = [2 \cdot 10^{-4}, 3 \cdot 10^{-4}]$. This is expected as most of the change in the solution happens at the early stages of the time evolution where the flux at the fuel increases because of the source term and the initial flux at the absorber diffuses away. The dynamics in this time range will change the most as a function of the source and σ_f compared to a different part of the time evolution.

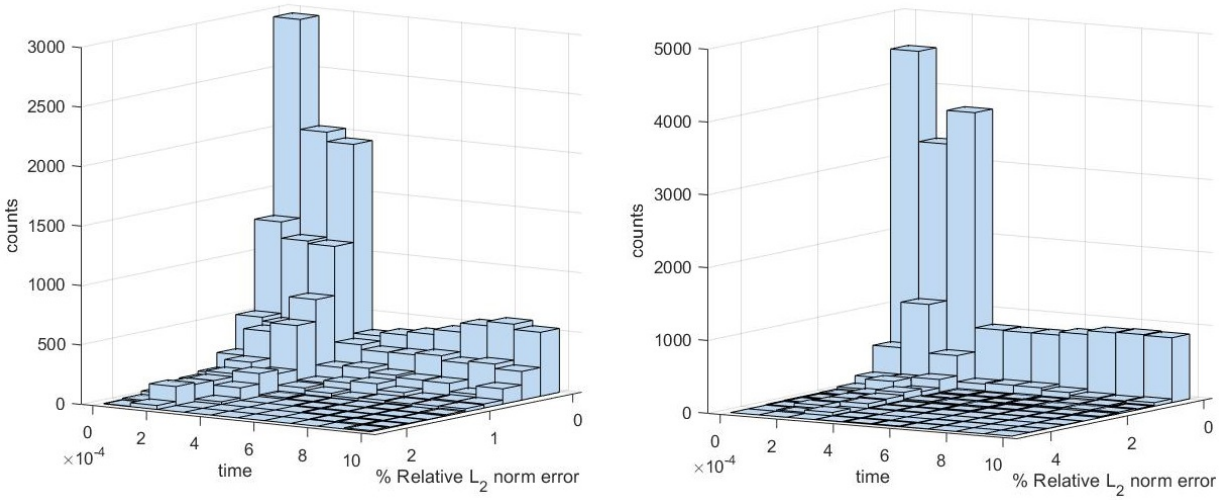


Figure 58: 3D histogram that shows the counts of a certain error to occur in a period of time.

Next, in Figure 59 the error of the interpolation of **U**, **C** and **V** is given. The achieved errors in experiment S.σ_f.B are higher than in experiment S.σ_f. The interpolation error of the modes is higher than the error of the coupling coefficients or the coefficients in **V**. This is expected as the time-dependent behaviour of the modes in Figure 30 is not very complex compared to the overall shape of the modes themselves. For the 3 different interpolated matrices, it is harder to interpolate the higher order modes and columns than the lower order ones. This could be attributed to the fact that the higher order modes are noisier compared to the lower order modes, especially in this numerically solved problem.

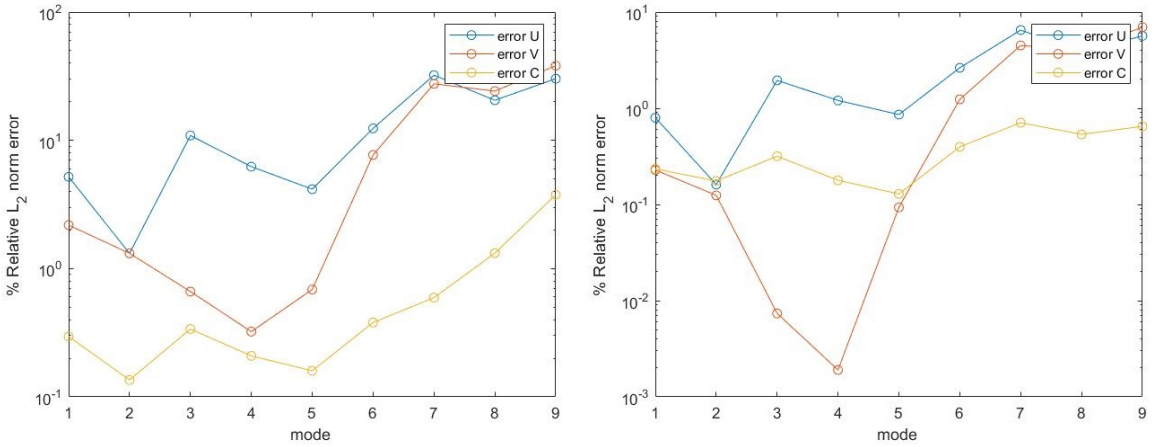


Figure 59: The relative L2 errors are plotted for the individual columns in the interpolated **U**, **C** and **V** matrices of the solution with the highest error $\epsilon_{L_2,max}$. The results of experiment S.σ_f.B and S.σ_f.C are shown on the left and right respectively. mode on the x-axis indicates the index of the column in these matrices.

The true and interpolated modes 1-3 of the solution with the highest L_2 error in experiment S.σ_f.B are plotted in Figure 60.

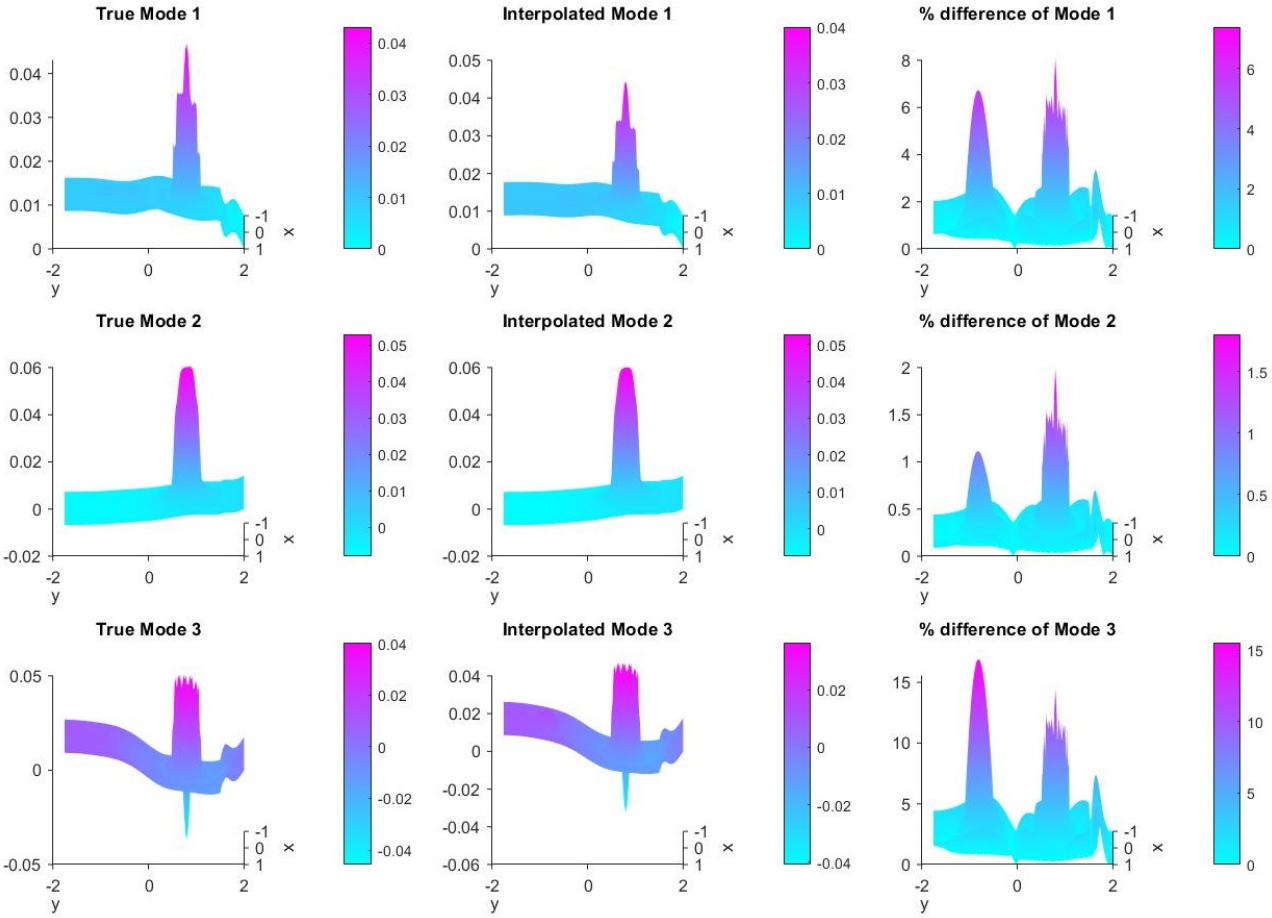


Figure 60: Plots of the true and interpolated modes 1 to 3 of the interpolated solution with the highest error $\epsilon_{L_2,max}$. The percentage difference is plotted on the right.

In all 3 modes, the error is highest at the absorber and the fuel, which is similar to the error

of the solution itself in Figure 56. The error of the interpolated mode 1 and 3 are substantially higher than the error of mode 2. There is also noticeable noise in the first mode, which could have been overlooked in the mode analysis chapter. Therefore, to check if that occurring noise is a true property of the solution at that parameter combination, the true modes from the SVD are plotted for this parameter combination in Figure 61.

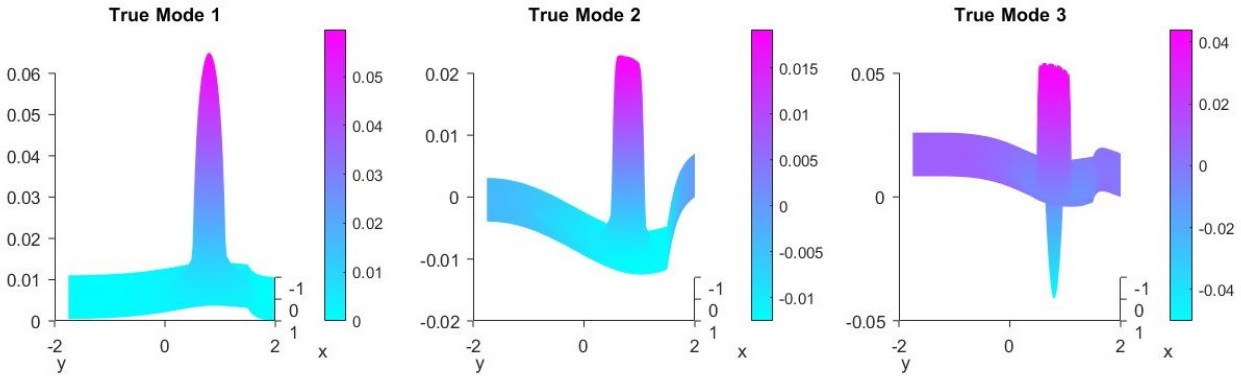


Figure 61: Plots of the true modes 1-3 that come out of the SVD of the true solution at the parameter combination where the highest interpolation error was achieved.

The noise does not occur in the true modes, meaning that this is a byproduct of the exponential map, which is the orthogonal transformation of the true basis. The product of the true orthogonally transformed $\mathbf{U}$, the true coupling matrix $\mathbf{C}$, and the true orthogonally transformed matrix $\mathbf{V}$ does give back the same solution as the product of the regular matrices from the SVD. This means that the added noise does not actually pose a problem for the interpolation itself, as the combination of these matrices will give back the same result. The accuracy of the interpolation is therefore mainly dependent on the number of points that were sampled and the combination of the RBF interpolator and the subdomain size. In Figure 62, the interpolated POD coefficients are plotted with the percentage difference.

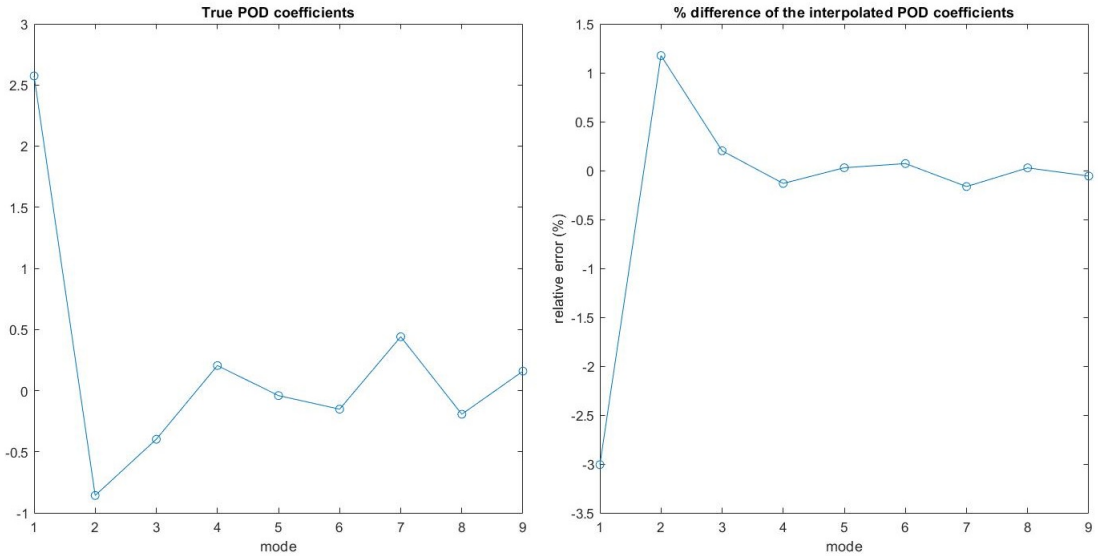


Figure 62: The true POD coefficients and the relative percentage error of the interpolated POD coefficients of the solution with the highest error.

The interpolated POD coefficient of mode 1 is 3% lower than its respective true positive POD coefficient, while the interpolated POD coefficient of mode 2 is more than 1% larger than its

respective true positive POD coefficient. The percentage difference of mode 3 is relatively low, which indicates that the sharp flux minimum in the interpolated solution in Figure 56 is mainly caused by the amplitude of mode 1 not being high enough to counterbalance the contribution of mode 2 and 3, which cause the flux at the absorber to be reduced. The true and interpolated modes 1-3 of the solution with the highest L_2 error in experiment S.σ_f.C are also plotted in Figure 63.

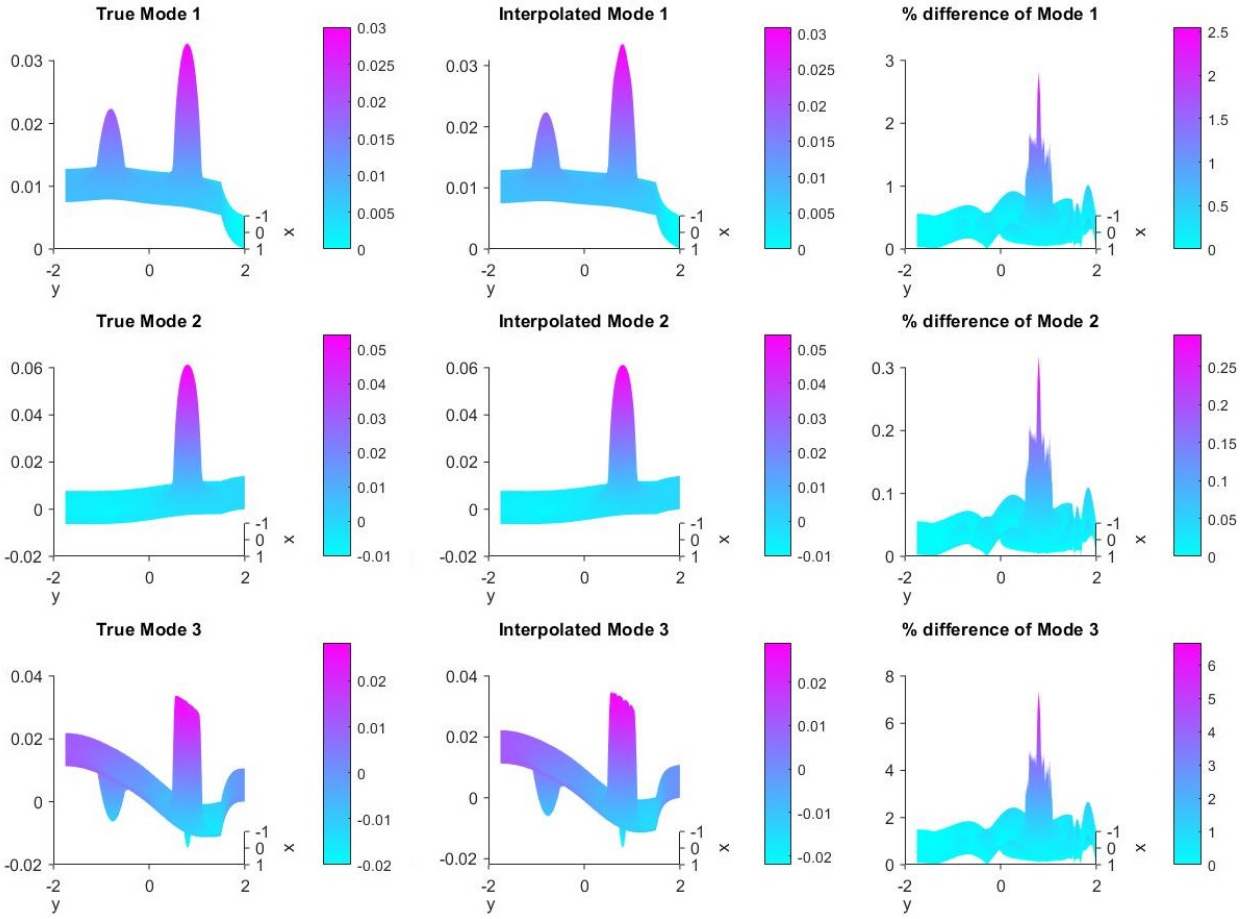


Figure 63: Plots of the true and interpolated modes 1 to 3 of the interpolated solution with the highest error $\epsilon_{L_2,max}$. The percentage difference between the two is plotted on the right.

The first mode does not contain any noise compared to the first mode in Figure 60. This means that the orthogonal transformation is different depending on the parameter combination and the tangent plane. The error of the interpolated modes at the absorber is very low compared to the interpolated modes in Figure 60, even though the modes represent a solution with a positive source, which can be seen from the first mode. This is completely opposite to the error of the interpolated modes in Figure 60, which represent the solution at a minimal source value. However, the error of the modes at the absorber is high in both the solution with a low and high source.

In Table 8 it can be observed that in experiment S.σ_c, 8 different tangent planes were drawn. These were drawn only because of different amounts of modes being needed in the parameter domain. The resulting sparse grids and error distribution of experiment S.σ_c is given in Figure 64. The number of modes in the lower left part of parameter domain changes, as multiple tangent planes were drawn to stay within the range defined by $[\epsilon_{tol,min}, \epsilon_{tol,max}]$. In this case the difference between the number of modes at the tangent planes is only one mode. The distribution of tangent planes follows the hierarchy of the sparse grids, where each tangent plane in each dimensions has a neighbour that can be used to define the overlapping training subdomains. Even though 8 different tangent planes were drawn, the error distribution is still uniform, where there are no

specific subdomains where the error is very high.

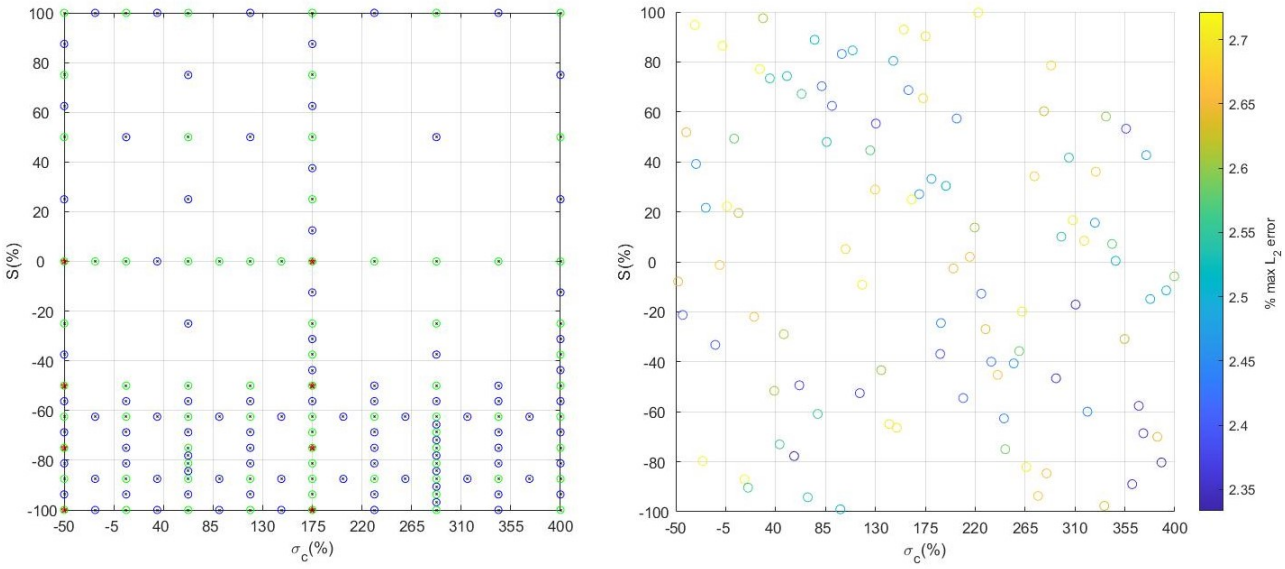


Figure 64: The resulting sparse grid of experiment S.σ_c on the left, with the error distribution on the right. The green points are the important points that are included in the ROM, the blue points are the inactive points where the interpolation error was low enough and the red points indicate a tangent plane.

5. Discussion

5.1 Effect of subdomain size and noise on the interpolation scheme

The findings in the experiments on the Burgers equation indicated that it is preferred to use smaller training subdomains when using the multiquadric RBF and vice versa when using the cubic RBF. This can be explained by looking at the shapes of the multiquadric and cubic RBF, which are plotted in Figure 65:

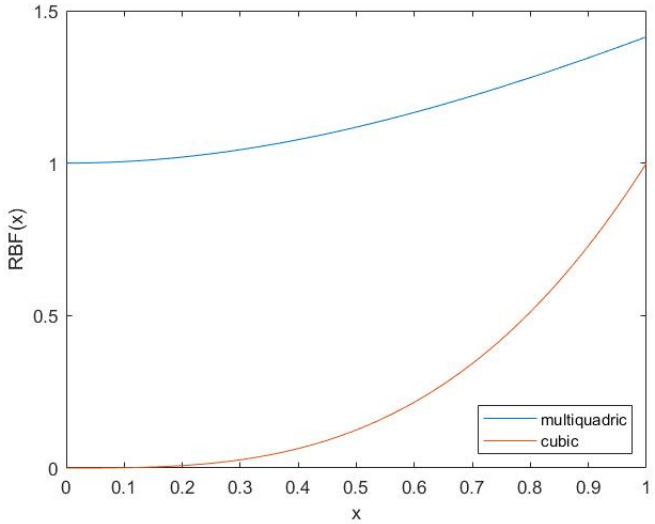


Figure 65: Plots of the multiquadric and cubic RBF functions.

As the derivative of the multiquadric RBF increases very slowly compared to the cubic RBF, the difference between the weights of points with increasing radial distance in the multiquadric RBF is much smaller than the cubic RBF. This means that for larger subdomains, very far distanced points to a point of interpolation interest will have a larger influence on the interpolation, compared to the cubic RBF where the weight of the far distance points decrease very fast to zero.

The main reason for the interpolation failing in Figure 38 at very low Reynolds numbers is that the RBF interpolation method fails around that region. Namely, ill-conditioning can occur near the boundaries as radial basis function interpolation relies on the radial symmetry of the basis functions. Basis functions centered at points on or close to the boundaries of the interpolation space become asymmetric. This mainly occurs when the density of training points is high on the boundaries of the domain, resulting in the radial distance matrix $\mathbf{D}$ in Equation (61) becoming almost singular, which can lead to an inaccurate inverse matrix. This also happened during this experiment. The interpolation failing in Figure 39 could be caused by the horizontal lift having to represent a velocity vector of a geodesic that leads to a point in a region on the Stiefel manifold that is very random due to the noise. It should be noted that the bases that include noise, still are legitimate points on the Stiefel manifold. The noise therefore becomes input to the logarithmic map in Equation (47), which results in a noisy horizontal lift matrix. It can be expected that trying to interpolate such noisy data will lead to an inaccurate interpolation of the horizontal lift, which then additionally must be mapped back via the exponential map in Equations (49) and (50).

From Figure 38 it can be stated that it is possible to include more higher order modes than necessary without making the tangent plane interpolation method more difficult. The interpolation of the

lower order modes does not deteriorate as more higher order modes are added. However, Figure 39 indicates that if a too high number of modes is interpolated in a domain where that number would be in a numerical noise range, then the interpolation completely deteriorates. Therefore, it is possible to neglect $\epsilon_{tol,min}$, given that no numerical noise is included into the matrices $\mathbf{U}$ and $\mathbf{V}$. However, a lower first principal angle tolerance $\theta_{1,tol}$ should be chosen to draw more tangent planes leading to smaller subdomains for the RBF interpolator. A better choice would be to not choose the lowest number of modes resulting in an approximation error below $\epsilon_{tol,max}$, but add more modes to not draw too many tangent planes in Figure 42.

5.2 Evaluation of the space-time coupled matrix interpolation scheme in the Molenkamp test

The main cause of the high error in the Smooth and Steep.A experiment could be explained by looking at the space-time coupled interpolation method itself and the previous method based on the global basis. The main difference between the interpolators of both algorithms is that the algorithm based on the global basis interpolates the individual time dependent coefficients of the respective modes for each individual state vector, while the algorithm based on the local bases does an interpolation of the coefficients for a whole-time evolution, by interpolating the full matrices of $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{C}$. This means that the choice of the interpolation accuracy threshold γ_{int} should be chosen lower, to be more forgiving of the fact that a full-time evolution is being interpolated instead of the individual time steps in the algorithm with the global basis.

Another point to address is the large difference in sampled values of λ_3, which is a parameter that leads to a decaying exponential dependence of the solution. The reason for this could be seen by looking at how accurate the coefficients of $\mathbf{V}$ are interpolated in Figures 44 and 51. It seems to be the case that the time dependent behaviour can be interpolated very easily in the space-time coupled approach. As the coefficients in $\mathbf{V}$ are essentially part of the information in the POD coefficients and are split off and interpolated separately with high accuracy, it can be stated that the coupling matrix coefficients in $\mathbf{C}$ are less complex than the POD coefficients. Interesting research would be to also analyze the difference in number of points that are needed to separately interpolate the $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{C}$ matrices with a high enough accuracy. This allows for a clear distinction in the dependence of these different matrices. Additionally, more analysis will have to be made on the difference between the local and global basis in certain problems with regards to how the overall physics is summarized over the whole parameter domain in the global basis compared to how the physics is represented via the local basis interpolation scheme.

It cannot be expected that the coupling coefficients in $\mathbf{C}$ scale in the same way compared to the values in the $\mathbf{Z}^U$ and $\mathbf{Z}^V$ matrices of which the interpolation should result in a velocity vector representing a geodesic that stays within the manifold. Therefore, entry-by-entry interpolation might the best solution for the coupling coefficients using local linear basis functions, instead of interpolating the full coupling matrices using the RBF interpolator. Another benefit of the local linear basis functions can be seen when looking at the difference in unique values of λ_1 and λ_2, which can be attributed to the way the RBF interpolator works compared to the one based on local linear basis functions. The RBF weights are calculated based on radial distances without considering any directions in the parameter domain. On the other hand, the algorithm based on local linear basis functions draws linear hat functions at the sampled points in each parameter dimension in the domain. These can then be combined to form a multivariate interpolant in which the dependencies of the function on each separate parameter are combined. For instance, the local linear interpolant would be able to do an extrapolation to the point $[1; 1]$ by forming an interpolant of two hat functions going from points $[0; 0]$ to $[0; 1]$ and $[0; 0]$ to $[1; 0]$. This is not possible with an RBF interpolation as the weights are only calculated based on a distance measure without taking into consideration what the dependencies are in each parameter direction.

Another problem of the RBF interpolator is that the test points must be surrounded by training points to have an accurate interpolation. This basically means that in every problem, no matter

what dimensions the parameter domain has, the corners will have to be sampled, which leads to a lot of extra evaluations being needed to cover the dynamics in the parameter domain. It can also be argued that the RBF interpolator cannot recognise linear dependencies very well, as it essentially draws RBF functions, between the sampled points that are not linear by themselves such as multiquadric or cubic. An option would be to change the sampling strategy, by filling the parameter space with cuboids in [34] in a hierarchical way. In this sampling strategy the algorithm starts with training points at the vertices of the parameter domain and tests the middle points of the domain. Then in subsequent iterations, the parameter domain is split up in smaller cuboids in which an interpolation is done to the center of these smaller cuboids. As the vertices of the corners of the cuboids can be used as training points, the test points will always be surrounded, resulting in an optimal RBF. In this case, the tangent plane can be drawn on one of the vertices. The question, however, is whether this would lead to more points being sampled compared to the sparse grid.

5.3 Computational cost and scaling of the algorithm

With the inclusion of drawing tangent planes in a hierarchical manner and overlapping the subdomains, it could be argued that a lot of extra computational resources and steps are needed to compute the ROM compared to the algorithm based on the global basis. Namely, in each single subdomain, it is checked whether the number of modes or first principal angle is compatible. Then, as new tangent planes are drawn and the subdomains are overlapped, many SVD calculations are made to calculate the horizontal lifts. However, the time needed to do these calculations is negligible compared to the time that it takes to do the numerical simulations in a 2D neutron diffusion problem. The main problem is the amount of data that is used to represent the full ROM. Namely, at points that lie in multiple tangent plane training subdomains due to the overlap, as many $\mathbf{Z^U}$ matrices must be stored as the number of tangent planes that point is connected to. For a high dimensional parameter domain this can be problematic. Moreover, each subdomain can use a different number of modes, which leads to many horizontal lifts of different dimensions that must be stored. Therefore, it is much harder to use conventional arrays to store the data compared to in the situation where only a single global basis and a matrix of coefficient must be stored.

Another main problem with this algorithm is how it scales for problems in which there is a large parameter domain with varying dependencies, from linear to very non-linear. As was already stated in the discussion of the experiments on the Molenkamp test, the RBF interpolator does not have any sense of direction in terms of being able to interpolate the separate dependencies in each dimension. One could apply entry-by-entry interpolation for the coupling coefficients $\mathbf{C}$ and the coefficients in $\mathbf{Z^V}$, but not for the values in $\mathbf{Z^U}$ as the spatial discretization leads to too many degrees of freedom compared to the temporal discretization. Therefore, future research should investigate whether there is a matrix interpolation method that has the same property of as the local linear basis functions, that takes into account the dependencies in different dimensions and can be used to extrapolate to corner points of the domain.

The last thing to point out is that it is difficult to use the space-time coupled approach in a time-adaptive way, instead of only in a parameter-adaptive way. With the algorithm based on the global basis, all the parameter and time dependent information is in the POD coefficients which can be interpolated separately using local linear basis functions. One idea would be to keep using the previous algorithm based on the global ROM, but when a new parameter combination is being sampled in time, the global basis can be locally enriched with new local basis vectors that are interpolated to that new parameter combination. Another possibility would be to just use the local interpolated basis at that new parameter combination and start sampling in time, with the initial interpolated local basis. The first snapshot is then projected on this local interpolated basis which essentially leads to local POD coefficients, instead of global POD coefficients. As more snapshots are added in time at the new parameter combination, the local basis can be updated with the true modes that come out of the SVD at that point.

6. Conclusion & recommendations

6.1 Conclusion

In this research a local basis interpolation method for time dependent parametric problems was successfully set up and combined with a locally adaptive sampling strategy based on sparse grids. The resulting algorithm was tested on 3 different models.

As the maximally achieved interpolation errors in the experiments on the Burgers equation were all low, it can be concluded that in a 1D parameter setting, the algorithm can adaptively draw new tangent planes in a hierarchical manner and can accurately interpolate withing the overlapping subdomains. Due to the fact that different radial basis functions scale differently for further distanced points from the point of interpolation interest, it can be stated that the interpolation performance of the RBF interpolator depends on the subdomain size, which is determined by how many tangent planes are drawn and how much the subdomains overlap. As the interpolation accuracy of lower order modes does not deteriorate as more higher order modes are added, a conclusion can be drawn that the minimum error tolerance, which is used to determine how many modes are needed for the interpolation within a subdomain, can be neglected, given that no numerical noise is included in the bases and coefficients.

Judging from the results on the Molenkamp problem, it can be concluded that the algorithm based on the local basis performed worse than the algorithm based on the global basis, for an equally set interpolation accuracy tolerance. The main cause of this is that full matrix interpolation is done with the coupling coefficient matrices $\mathbf{C}$, of which it cannot be guaranteed that the coupling coefficient in the matrix scale similarly. Another cause of this is that the RBF interpolator is not able to separate the dependencies in each parameter direction as it only relies on distances to do the interpolation. However, it can also be concluded that the algorithm based on the local bases can perform similarly well on the Molenkamp problem, given that the interpolation accuracy tolerance is set lower than the algorithm based on the global basis. This is done to be more forgiving of the fact that full matrix interpolation is being done.

The main conclusion that can be drawn from the 2D neutron diffusion problem is that also for a 2D parameter setting in a numerically solved problem, the algorithm can adaptively draw new tangent planes in a hierarchical manner and can successfully interpolate within the overlapping subdomains. It can also be concluded that the first principal angle can act as an indicator of which parts of the parameter domain contain more relatable physics than other parts of the domain.

This thesis demonstrated a novel method generalizing reduced order modelling of time and parameter dependent problems on sparse grids, by using multiple local bases instead of a fixed global basis to represent the underlying physics of a model. The local basis interpolation scheme can compete with the global basis approach on test problems, indicating that manifold methods such as ITSGM have a lot of potential in reduced order models. However, more research is needed in matrix interpolation methods to increase the algorithm's overall performance, applicability, and efficiency.

6.2 Recommendations

The first recommendation for future research is to develop a way to interpolate the horizontal lifts $\mathbf{Z}^U$ and $\mathbf{Z}^V$ in a similar way to interpolating coefficients with local linear basis functions. This should in turn reduce the number of evaluations that have to be made, due to being able to interpolate to the corner points of the domain. Local linear basis functions could also be used to interpolate the coupling coefficients entry by entry instead of the full matrices $\mathbf{C}$ themselves, which

might result in a better interpolation accuracy, and therefore less points having to be sampled. Another recommendation is to try the sampling scheme in which hierarchical cuboids are used instead of the sparse grids. It would be interesting to see if the space-time coupled approach would work better in the context of this sampling scheme. Another problem that will have to be resolved, is the storage of the data that represents the ROM. This would allow for the algorithm to be tested on larger problems like the Molenkamp problem with 101 time points or problems with higher dimensional parameter domains, which would give more insight into the scaling and performance of this algorithm. Also, more analysis will have to be done on the separate dependencies of the $\mathbf{U}$, $\mathbf{C}$ and $\mathbf{V}$ matrices throughout the domain and how the local basis really differs from the global basis in certain numerical problems. The last suggestion would be to use local basis interpolation in combination with the global basis by enriching the global basis with local basis vectors or using only the local basis when the time points are sampled at new parameter combinations, thereby leading to local POD coefficients instead of global POD coefficients.

Bibliography

[1] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.

[2] Christophe Audouze, Florian De Vuyst, and Prasanth Nair. Nonintrusive reduced-order modeling of parametrized time-dependent partial differential equations. *Numerical Methods for Partial Differential Equations*, 29(5):1587–1628, 2013.

[3] *A Reduced Order Model for Multi-Group Time-Dependent Parametrized Reactor Spatial Kinetics*, volume Volume 5: Innovative Nuclear Power Plant Design and New Technology Application; Student Paper Competition of *International Conference on Nuclear Engineering*, 2014. V005T17A048.

[4] Fahad Alsayyari. *Adaptive data-driven reduced-order modelling techniques for nuclear reactor analysis*. PhD thesis, Delft University of Technology, 2020.

[5] Fahad Alsayyari, Marco Tiberga, Zoltán Perkó, Danny Lathouwers, and Jan Leen Kloosterman. A nonintrusive adaptive reduced order modeling approach for a molten salt reactor system. *Annals of Nuclear Energy*, 141:107321, 2020.

[6] Jan Hesthaven and Stefano Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, 2018.

[7] Konstantinos Vlachas, Konstantinos Tatsis, Konstantinos Agathos, Adam Brink, and Eleni Chatzi. A local basis approximation approach for nonlinear parametric model order reduction. *Journal of Sound and Vibration*, page 116055, 2021.

[8] Si Hun Lee, Kijoo Jang, Haeseong Cho, Haedong Kim, and Sang Joon Shin. Parametric non-intrusive model order reduction for flow-fields using unsupervised machine learning. *Computer Methods in Applied Mechanics and Engineering*, 384:113999, 2021.

[9] Di Xiao, Feng Fang, Chistopher Pain, and Ionel Michael Navon. A parameterized non-intrusive reduced order model and error analysis for general time-dependent nonlinear partial differential equations and its applications. *Computer Methods in Applied Mechanics and Engineering*, 317:868–889, 2017.

[10] Mengwu Guo and Jan Hesthaven. Data-driven reduced order modeling for time-dependent problems. *Computer Methods in Applied Mechanics and Engineering*, 345:75–99, 2019.

[11] Joan Baiges, Ramon Codina, and Sergio Idelsohn. A domain decomposition strategy for reduced order models. application to the incompressible navier–stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 267:23–42, 2013.

[12] Wil Schilders, Henk Van der Vorst, and Joost Rommes. *Model Order Reduction: Theory, Research Aspects and Applications*, volume 13. 01 2008.

[13] Toby Phillips, Claire Heaney, Brendan Tollit, Paul Smith, and Christopher Pain. Reduced-Order Modelling with Domain Decomposition Applied to Multi-Group Neutron Transport. *Energies*, 14(5):1–25, March 2021.

[14] Alejandro Marquez, Jairo Espinosa Oviedo, and Darci Odloak. Model reduction using proper orthogonal decomposition and predictive control of distributed reactor system. *Journal of Control Science and Engineering*, 2013, 01 2013.

[15] Claire Heaney, Andrew Buchan, Christopher Pain, and Simon Jewer. Reduced-order modelling applied to the multigroup neutron diffusion equation using a nonlinear interpolation method for control-rod movement. *Energies*, 14(5), 2021.

[16] Claire Heaney, Andrew Buchan, Christopher Pain, and Simon Jewer. Reduced-order modelling applied to the multigroup neutron diffusion equation using a nonlinear interpolation method for control-rod movement. *Energies*, 14(5), 2021.

[17] Vladimir Buljak. Inverse analyses with model reduction: Proper orthogonal decomposition in structural mechanics, 2012.

[18] Umit Sayin. A short introduction to system theory: Indispensible postulate systems and basic structures of the systems in quantum physics, biology and neuroscience. *NeuroQuantology*, 14, 2015.

[19] Joel Robbin, Uw Madison, and Dietmar Salamon. Introduction to differential geometry. 2011.

[20] Rob van der Vorst. Lecture notes: Introduction to differentiable manifolds, 2012.

[21] John Lee. *Introduction to smooth manifolds. 2nd revised ed*, volume 218. 2012.

[22] Ralf Zimmermann. Manifold interpolation and model reduction. *arxiv.org*, 2019. 37 pages, 4 figures, featured chapter of upcoming "Handbook on Model Order Reduction", to appear at De Gruyter.

[23] Sylvain Calinon. Gaussians on riemannian manifolds: Applications for robot learning and adaptive control, 2020.

[24] Thomas Bendokat, Ralf Zimmermann, and Pierre-Antoine Absil. A Grassmann manifold handbook: Basic geometry and computational aspects, 2020.

[25] Orestis Friderikos, Emmanuel Baranger, Marc Olive, and David Néron. On the stability of POD basis interpolation via Grassmann manifolds for parametric model order reduction in hyperelasticity, 2020.

[26] Orestis Friderikos, Marc Olive, Emmanuel Baranger, Dimitrios Sagris, and Constantine David. A Space-Time POD Basis Interpolation on Grassmann Manifolds for Parametric Simulations of Rigid-Viscoplastic FEM. In *7th International Conference of Materials and Manufacturing Engineering (ICMMEN 2020)*, volume 318 of *7th International Conference of Materials and Manufacturing Engineering (ICMMEN 2020)*, page 01043, Thessaloniki, Greece, 2020. EDP sciences.

[27] Pierre-Antoine Absil, Rober Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, USA, 2007.

[28] Alan Edelman, Tomás Arias, and Steve Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, 20:303–353, 1998.

[29] Ke Ye and Lek-Heng Lim. Schubert varieties and distances between subspaces of different dimensions, 2014.

[30] Timothy Marrinan. *Grassmann, flag, and Schubert varieties in applications.* PhD thesis, 01 2017.

[31] David Amsallem and Charbel Farhat. Interpolation method for adapting reduced-order models and application to aeroelasticity. *Aiaa Journal - AIAA J*, 46:1803–1813, 2008.

[32] Dimitris Giovanis and Michael Shields. Uncertainty quantification for complex systems with very high dimensional response using Grassmann manifold variations. *Journal of Computational Physics*, 364, 2017.

[33] Suraj Pawar, Shady Ahmed, Omer San, and Adil Rasheed. Data-driven recovery of hidden physics in reduced order modeling of fluid flows. *Physics of Fluids*, 32(3):036602, 2020.

[34] Ye Lu, Nawfal Blal, and Anthony Gravouil. Space–time POD based computational vademecums for parametric studies: application to thermo-mechanical problems. *Advanced modeling and simulation in engineering sciences*, 5(1):1–27, 2018.

[35] Mourad Oulghelou and Cyrille Allery. Non intrusive method for parametric model order reduction using a bi-calibrated interpolation on the Grassmann manifold. *Journal of Computational Physics*, 426:109924, 2021.

[36] Mourad Oulghelou and Cyrille Allery. Hyper bi-calibrated interpolation on the Grassmann manifold for near real time flow control using genetic algorithm, 2019.

[37] Nguyen Thanh Son, Pierre-Yves Gousenbourger, Estelle Massart, and Pierre-Antoine Absil. Online balanced truncation for linear time-varying systems using continuously differentiable interpolation on Grassmann manifold. In *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 165–170, 2019.

[38] Dimitris Giovanis and Michael Shields. Data-driven surrogates for high dimensional models using gaussian process regression on the Grassmann manifold. *Computer Methods in Applied Mechanics and Engineering*, 370:113269, 2020.

[39] Rolando Mosquera, Abdallah El Hamidi, Aziz Hamdouni, and Antoine Falaize. Generalization of the neville–aitken interpolation algorithm on Grassmann manifolds: Applications to reduced order model. *International Journal for Numerical Methods in Fluids*, 93(7):2421–2442, 2021.

[40] Rolando Mosquera, Aziz Hamdouni, Abdallah El Hamidi, and Cyrille Allery. POD basis interpolation via inverse distance weighting on Grassmann manifolds. *Discrete Continuous Dynamical Systems - S*, 12(6):1743–1759, 2019.

[41] Ralf Zimmermann. A locally parametrized reduced-order model for the linear frequency domain approach to time-accurate computational fluid dynamics. *SIAM Journal on Scientific Computing*, 36, 2014.

[42] Caroline Shouraboura and Pavel Bleher. Placement of applications in computing clouds using voronoi diagrams. *J. Internet Services and Applications*, 2:229–241, 2011.

[43] James Duderstadt and Louis Hamilton. *Nuclear reactor analysis / James J. Duderstadt, Louis J. Hamilton.* Wiley New York, 1976.

[44] MATLAB. *version 7.10.0 (R2010a).* The MathWorks Inc., Natick, Massachusetts, 2010.

7. Appendix

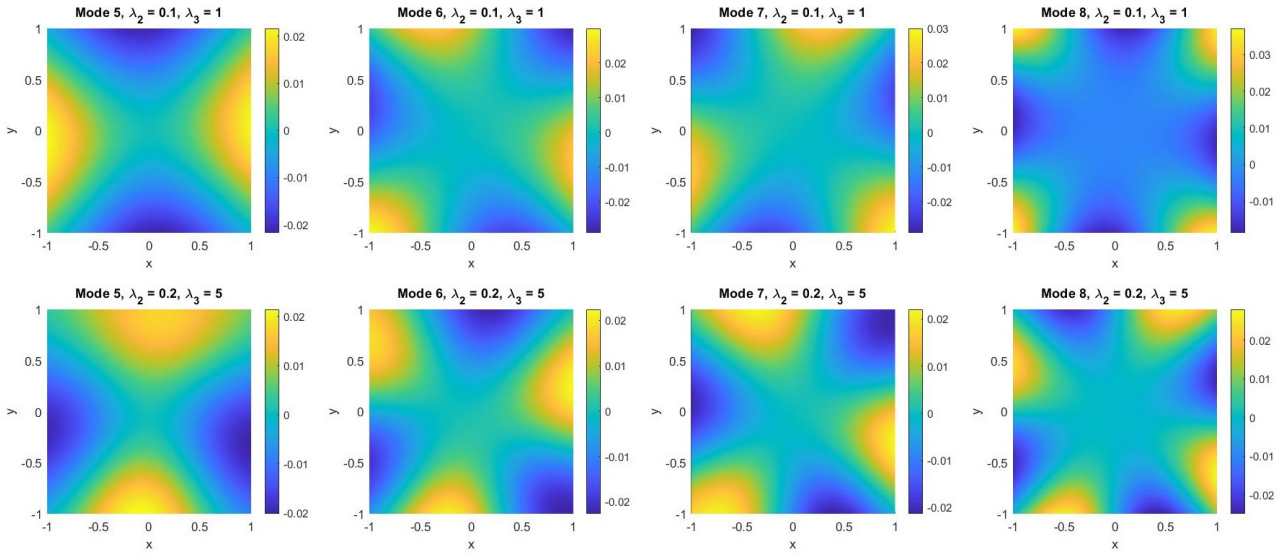


Figure A1: Modes 5-8 of the smooth Molenkamp test for different steepness and exponential decay values.

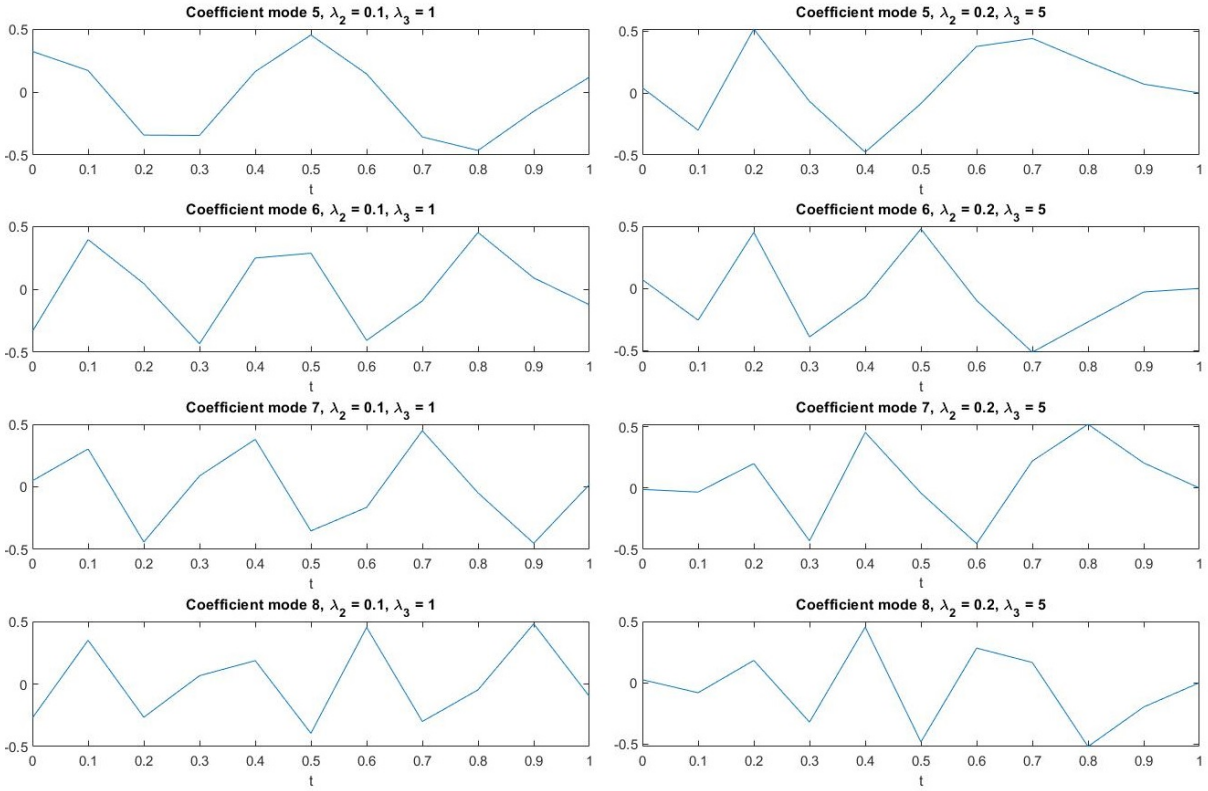


Figure A2: Coefficient 5-8 of the smooth Molenkamp test for different steepness and exponential decay values.

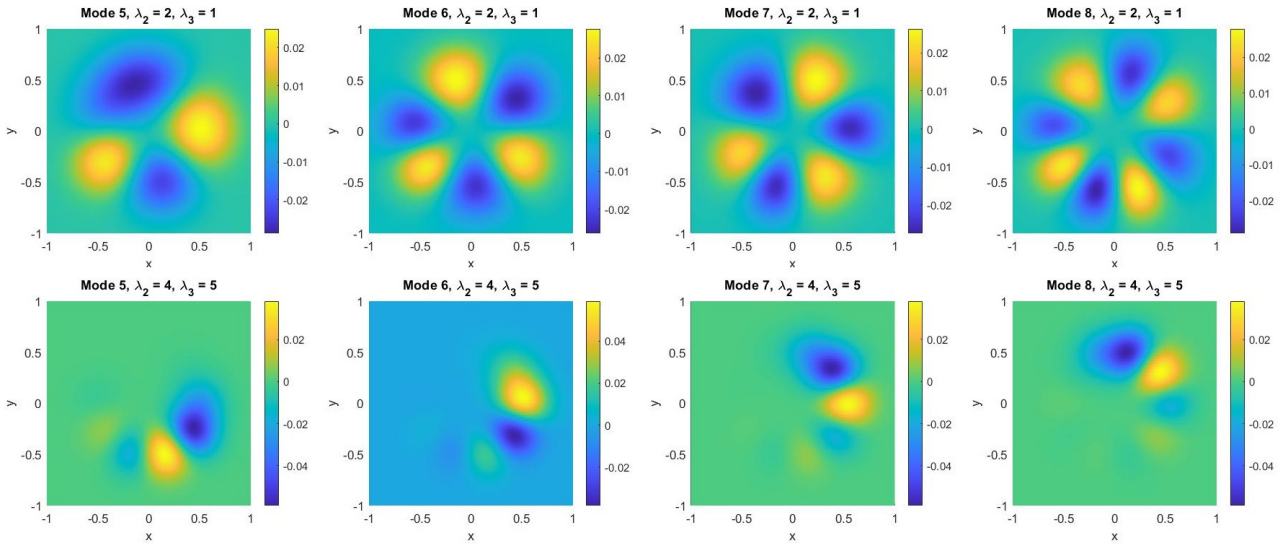


Figure A3: Modes 5-8 of the steep Molenkamp test for different steepness and exponential decay values.

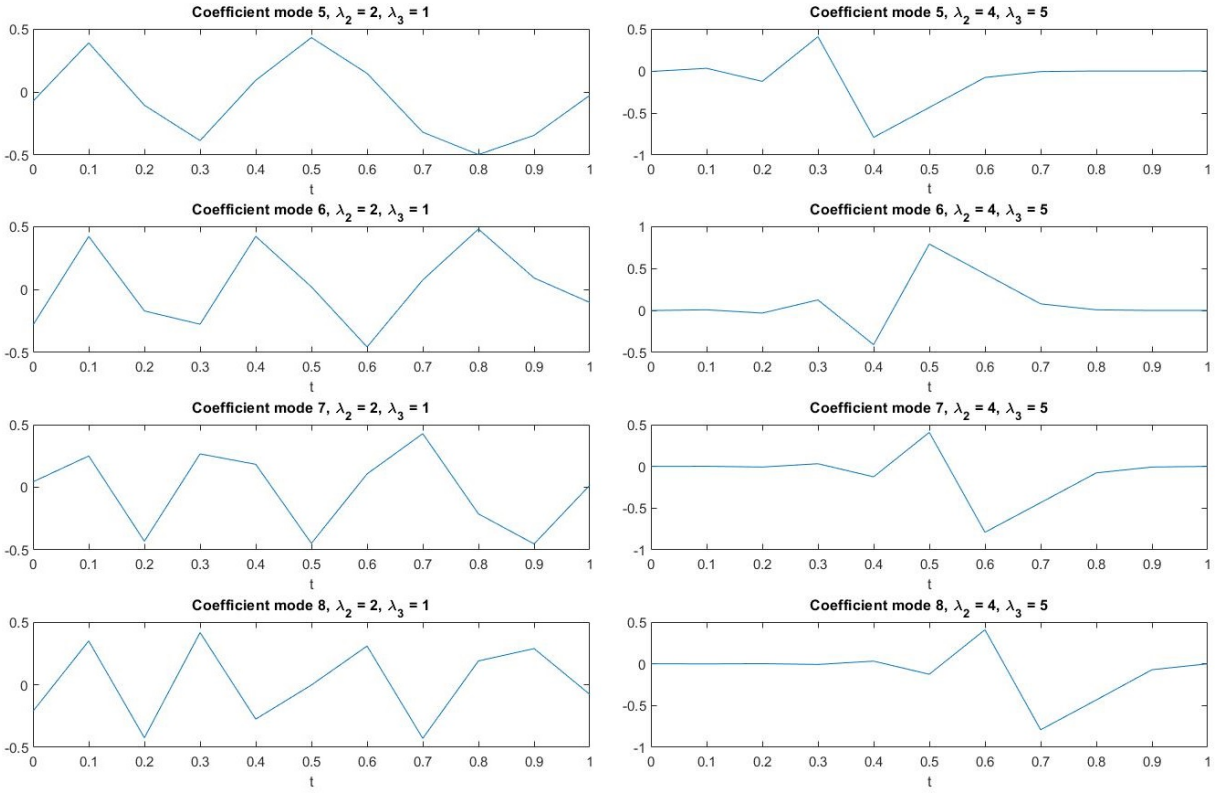


Figure A4: Coefficient 5-8 of the steep Molenkamp test for different steepness and exponential decay values.

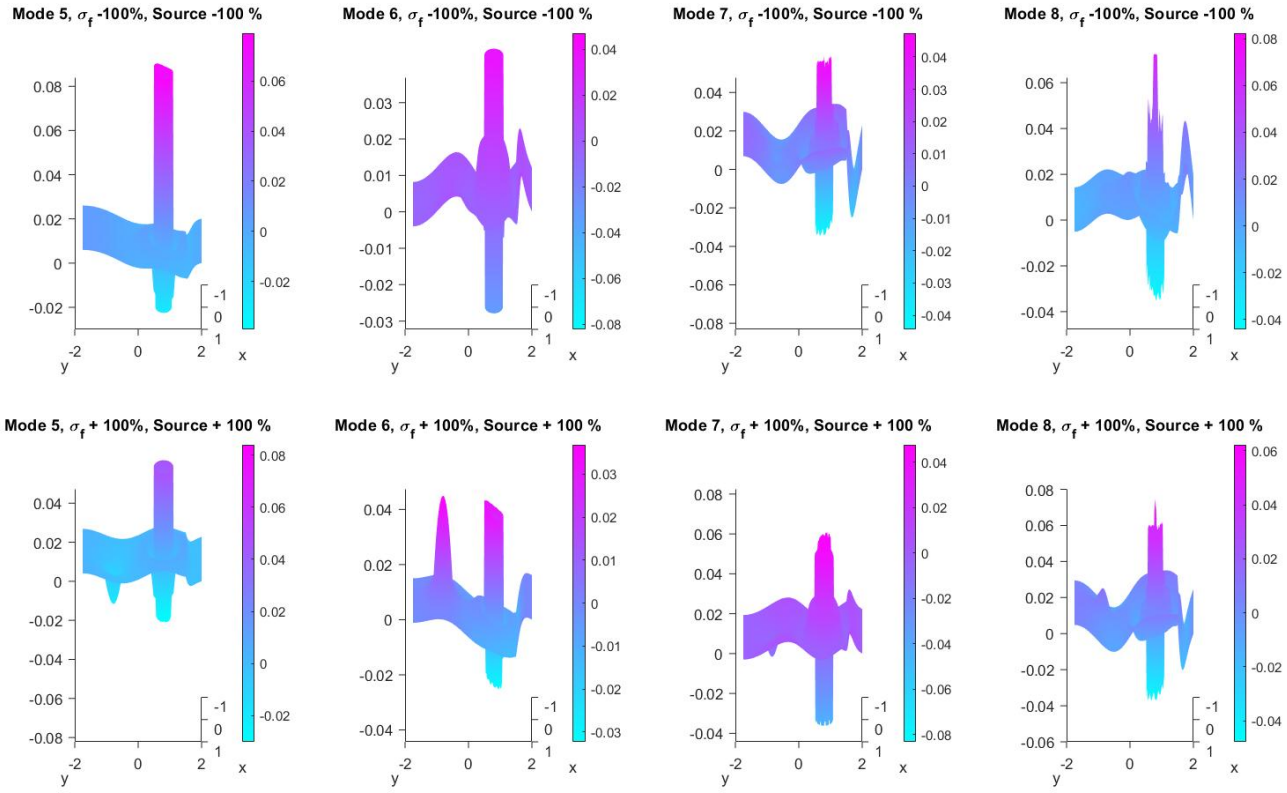


Figure A5: Modes 5-8 of the 2D neutron diffusion problem for different source values and fission cross sections σ_f.

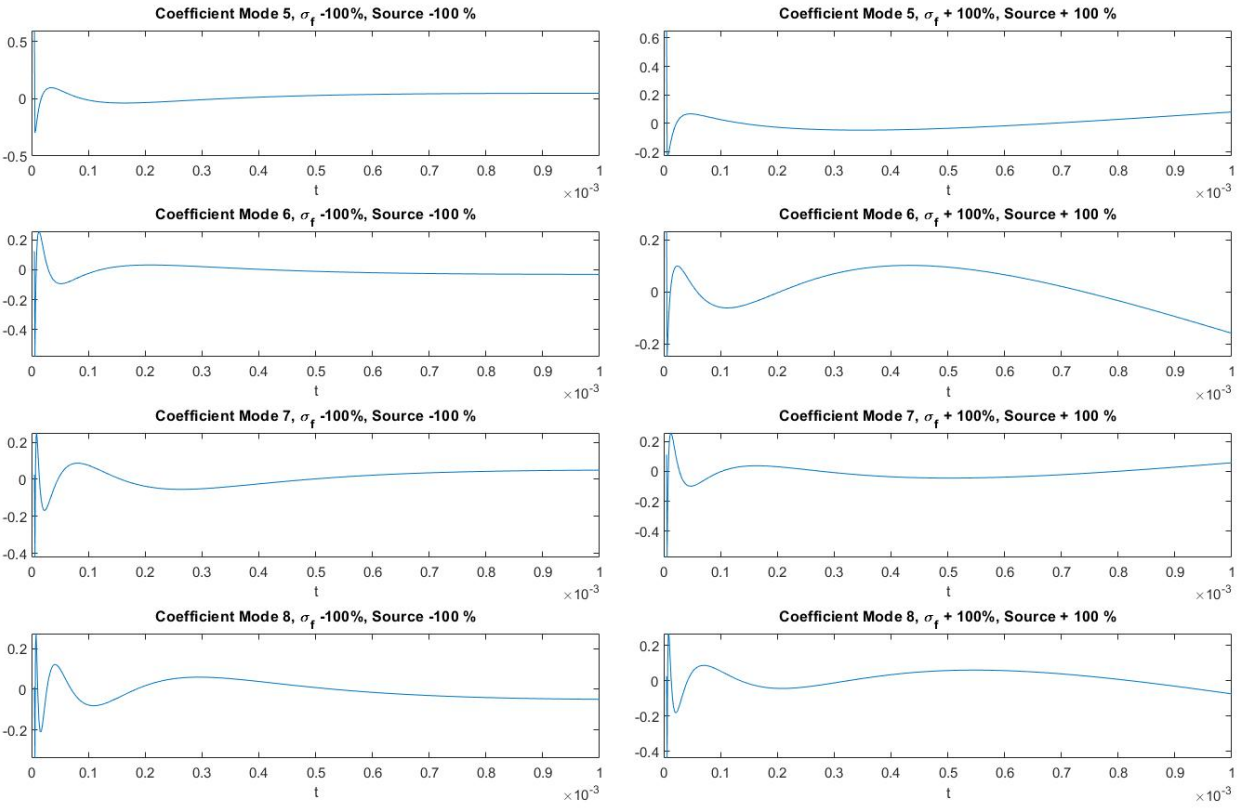


Figure A6: Coefficients 5-8 of the 2D neutron diffusion problem for different source values and fission cross sections σ_f.

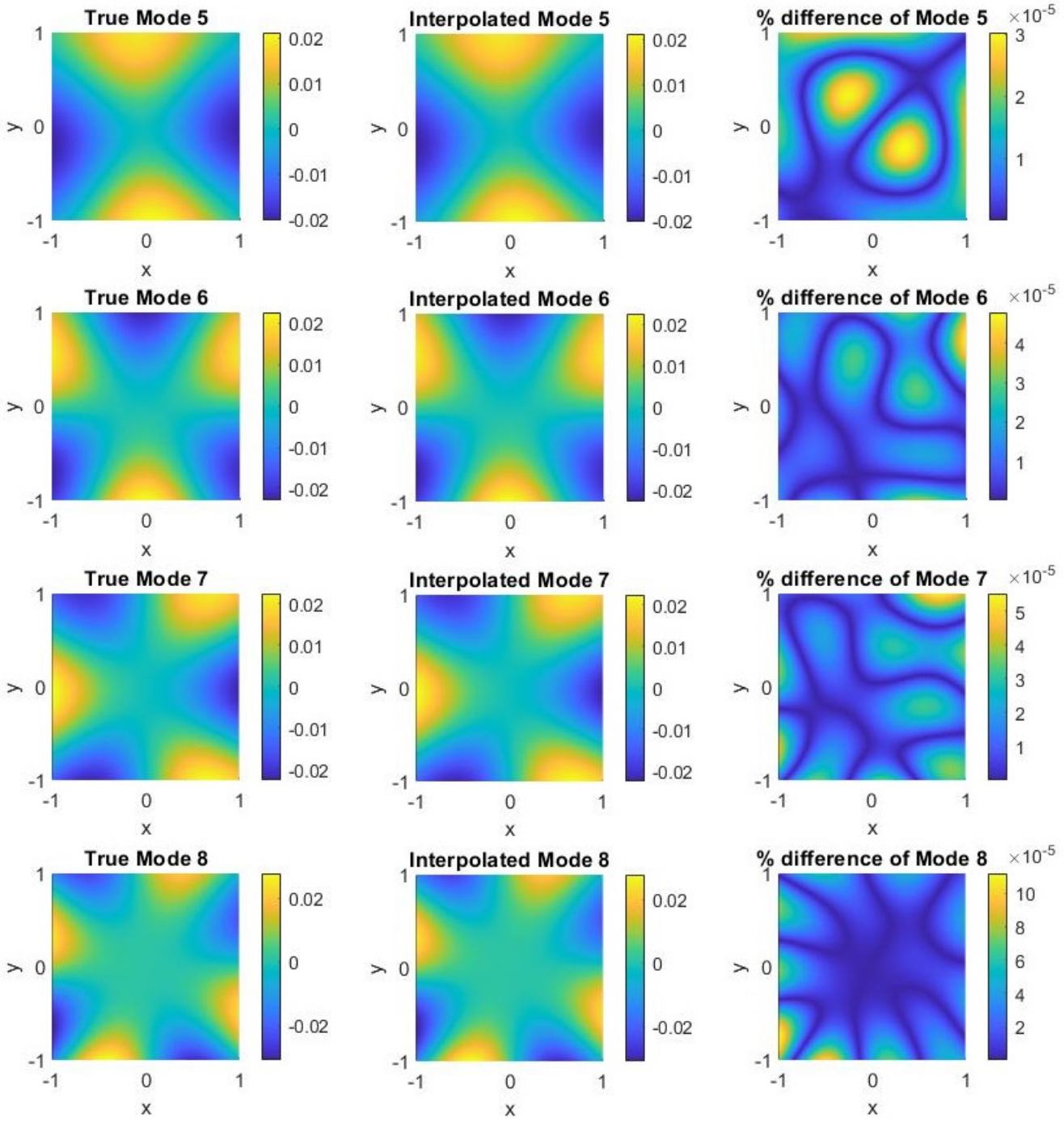


Figure A7: True and interpolated modes 5-8 of the smooth Molenkamp test with the given relative percentage difference, for the solution that achieved the highest L_2 error in experiments Smooth.

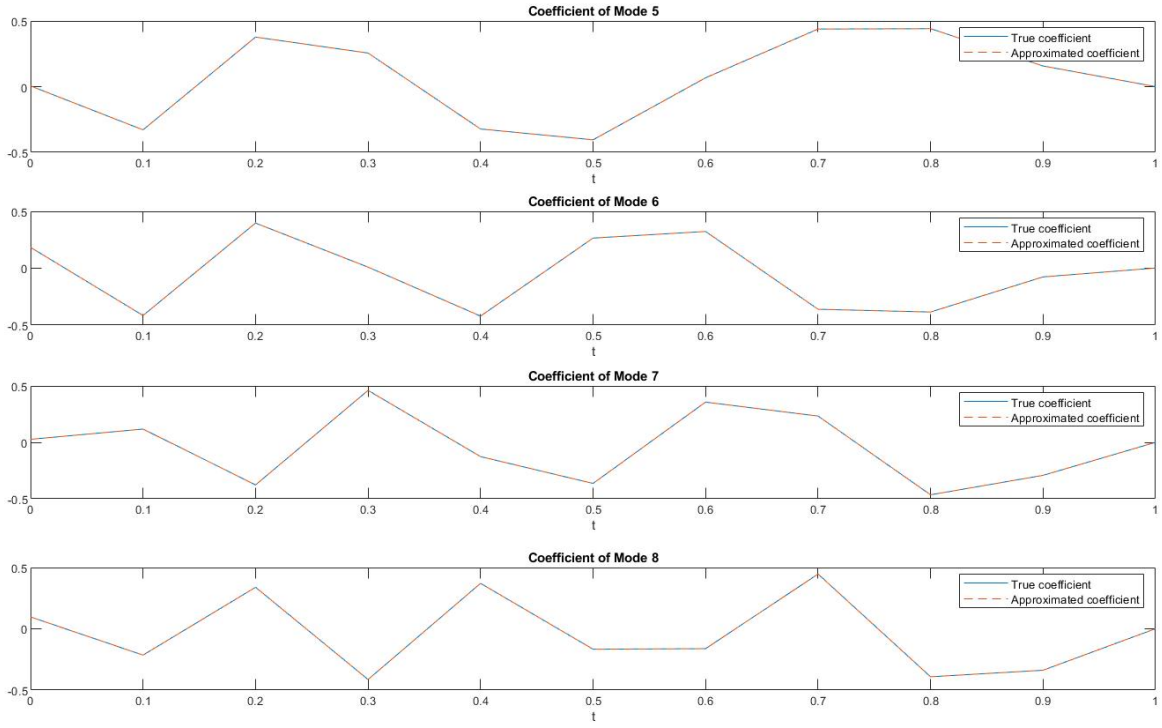


Figure A8: True and interpolated coefficients of modes 5-8 of the smooth Molenkamp test with the given relative percentage difference, for the solution that achieved the highest L_2 error in experiments Smooth.