Low-rank tensor decompositions for surrogate modeling in forward and inverse problems

Surrogatmodellierung von Vorwärts- und Inversen Problemen mittels Niedrig Rang Tensor Zerlegung

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
Genehmigte Dissertation von Ion Gabriel Ion aus Bukarest, Rümanien
Tag der Einreichung: 17. October 2022, Tag der Prüfung: 18. January 2023

1. Gutachten: Prof. Dr.-Ing. Herbert De Gersem
2. Gutachten: Prof. Dr.-Ing. Ulrich Römer
Darmstadt – D17

TECHNISCHE UNIVERSITÄT DARMSTADT

Electrical Engineering and Information Technology Department

Institute for Accelerator Science and Electromagnetic Fields

Low-rank tensor decompositions for surrogate modeling in forward and inverse problems
Surrogatmodellierung von Vorwärts- und Inversen Problemen mittels Niedrig Rang Tensor Zerlegung

Accepted doctoral thesis by Ion Gabriel Ion

Date of submission: 17. October 2022
Date of thesis defense: 18. January 2023

Darmstadt – D17

Bitte zitieren Sie dieses Dokument als:
URN: urn:nbn:de:tuda-tuprints-266782
URL: http://tuprints.ulb.tu-darmstadt.de/

Dieses Dokument wird bereitgestellt von tuprints,
E-Publishing-Service der TU Darmstadt
http://tuprints.ulb.tu-darmstadt.de
tuprints@ulb.tu-darmstadt.de

Erklärungen laut Promotionsordnung

§ 8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§ 8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§ 9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§ 9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 17. October 2022

I.G. Ion

Zusammenfassung

Diese Thesis behandelt das Thema der Surrogatmodellierung von Vorwärts- und Inversen Problemen im Rahmen von parameterabhängige Systemen, welche von Differentialgleichungen beschreiben sind. Surrogatmodelle werden genutzt um eine genaue Approximation des parameterabhängigen Systemzustandes zu gewinnen und werden verwendet um sowohl Vorwärtsmodellauswertungen als auch Bayessche Inversion zu beschleunigen. Im Folgenden werden zwei Anwendungen betrachtet, nämlich die parameterabhängige Chemische Mastergleichung als auch eine elliptische partielle Differentialgleichungen mit parameterabhängigen Gebieten. In beiden Fällen wird eine Tensorproduktbasisdarstellung verwendet um den Parameterzusammenhang darzustellen was dazu führt, dass sich die Dimension des Lösungstensors mit der Anzahl der Parameter erhöht, was vorallem mit vielen Parametern im Bezug zur Zeitkomplexität sehr teuer wird. Um diese Problematik anzugehen, werden Niedrig-Rang Tensor Zerlegungen, insbesonders das "Tensor-Train" Format, angewendet um die Speicher- und Zeitkomplexität von hochdimensionale Tensoren zu reduzieren. Um den Tensor der Freiheitsgrade im komprimierten Format zu gewinnen, wird ein speziell für diesen Zweck bestimmter Löser verwendet. Darin liegt die grösste Herausforderung in der Konstruktion der diskreten Systeme direkt in dem "Tensor-Train" Format für den gesamten Zustandsraum der Parameter.

Die Lösung der Chemischen Mastergleichung kann auf natürliche Art und Weise in einem Tensorformat dargestellt werden, und die Niedrig-Rang Formate können direkt angewendet werden um eine Reduktion der Rechenzeit zu erziehlen. Um eine effiziente Konstruktion der diskreten Operatoren zu bekommen, wird ein Algorithmus für den gesamten Zustand-Parameter-Zeit Tensor vorgestellt und erfolgreich verwendet um Bayessche Inferenz Aufgaben, wie zum Beispiel Zustandsrekonstruktion und Parameteridentifikation, durchhzuführen.

Im Falle von partiellen Differentialgleichungen, ist die Tensorproduktstruktur der diskretisierten Lösung nicht mehr vorhanden. Daher muss das Diskretisierungsverfahren so gewählt werden, dass eine Tensorproduktstruktur vorliegt, was speziell im Kontext der Isogeometrischen Analyse der Fall ist, da die im Referenzgebiet dargestellte Lösung entsprechendes Format hat. Das "Tensor-Train" Format kann somit für die Darstellung der Lösung verwendet werden. Die aus der schwachen Formulierung resultierende Integrale können mithilfe der Substitutionsregel über ein fixes Referenzgebiet mit einem modifizierten Metrik durchgeführt werden, anstatt über das parameterabhängige Gebiet zu integrieren. Die Anzahl an Dimensionen wird gleichzeitig erhöht um die Parameterzusammenhang einzufügen.

In beiden Anwendungen, führt das Niedrig-Rang "Tensor-Train" Format zu einer genauen Approximation der Lösung, mit reduzierter Rechenkomplexität. Die vorgeschlagenen Methoden besitzen die Fähigkeit hochdimensionale Probleme mit grossem Speicherbedarf zu lösen, was vorallem im Fall von Mastergleichungen mit vielen Reaktionsnetzerken der Fall ist. Die Speicherreduktion bringt auch eine Beschleunigung der Laufzeit der Simulationen, was besonders im Fall der Konstruktion diskreter Operatoren für parameterabhängigen Probleme in Augenschein tritt. Hier ist die vorgeschlagene Methode asymptotisch effizienter und um Größenordnungen schneller.

Abstract

This thesis addresses the topic of surrogate modeling for forward and inverse problems, in the context of parameter dependent systems described by differential equations. The surrogate model is an accurate approximation of the parameter dependent quantity of interest and is used to accelerate both forward model evaluations and Bayesian inversion. Two applications are considered: the parameter dependent chemical master equation (CME) and elliptic partial differential equations (PDEs) with parameter dependent computational domains. In both cases, a tensor product basis expansion is used to accommodate the parameter dependence, thus increasing the number of dimensions of the tensor used to store the approximation's basis coefficients. Low-rank tensor decompositions, in particular the tensor-train (TT) format, are used to reduce the computational costs of storing and handling large high-dimensional tensors. A dedicated solver is then used to obtain the coefficient tensor of the basis expansion in the low-rank format. The main challenge is the construction of the discrete systems directly in the low-rank format for the combined state-parameter space.

The solution of the CME is naturally represented as a tensor and the low-rank format can be directly applied to enhance the solver's performance. An algorithm for assembling the discrete operators for the joint state-parameter-time is presented. The computational complexity of this step is linear with respect to the number of dimensions. The developed framework is used for efficiently solving Bayesian inference tasks such as state reconstruction and parameter identification.

When dealing with PDEs, the tensor product structure of the discrete solution space is no longer a natural assumption. The discretization method in this case is the isogeometric analysis (IGA), since it leads to a tensor product structure of the solution in the reference domain. The TT format can then be used to represent the solution. When writing the weak formulation, the integral over the parameter dependent domain is transformed to an integral over a fixed reference domain with parameter dependent metric. The dimensionality of the solution tensor is then increased to accommodate the parameters.

In both cases, the use of the low-rank TT decomposition leads to accurate results at significantly reduced computational cost. The proposed tensor-train (TT) based frameworks are able to tackle high dimensional problems that would require a prohibitive amount of memory. This can be especially noticed when dealing with large reaction networks. The storage reduction brings a speedup of the runtime of the simulation. As an example, the complexity of constructing the discrete IGA operators in the TT format is asymptotically more efficient and orders of magnitude faster.

Contents

1. Introduction

We begin the thesis with an introduction in the topic and a literature survey. The contribution of the thesis is then stated, together with an overview of the content.

1.1. Motivation and related work

Over the past few decades, numerical simulations of physical systems have become a central topic in engineering. The increasing complexity of the models leads to higher demand of computing power, which is especially problematic when considering parameter dependent problems. Such problems often arise when performing optimization of design parameters, uncertainty quantification (UQ) studies, optimal control, or in general when exploring the behavior of the model for variable input data. Of particular interest in this work is the field of Bayesian inverse problems. In a nutshell, the goal of Bayesian inverse problems is to give a probabilistic description of the governing parameters of a system when certain output quantities of the system are observed. Applications can be found in fields such as computer vision and image processing [Bar18, GI15], electromagnetics [Che07, IWL+21], geosciences [Mal02, CFO11], computational chemistry [NDM+09, IWL+21], biomedicine [RWGG21], and many more. One common issue of Bayesian inversion and other UQ tasks is the high number of solver calls for different input parameter realizations. This motivates the use of computationally inexpensive surrogate models in the context of inverse problems.

A surrogate model is a function that maps a set of parameters to the corresponding value for the quantity of interest of the system. The approximation should be accurate and at the same time fast to evaluate. Once such an approximation is available, it can be used as a function handle to replace the solver in the computationally intensive tasks. A common problem when constructing surrogates is the increase of the computational complexity with respect to the number of parameters. This is known in the literature under the name of "curse of dimensionality" [BCC57]. One simple example is the interpolation of a multivariate function using a tensor product basis representation. The dimension of the basis grows exponentially with the number of variables. Moreover, in order to find the coefficients of the basis representation, an equal number of function evaluations is needed [Xiu10]. Several classes of methods have been proposed to circumvent this bottleneck: dimension adaptive basis expansions [GG03, LD20], sparse grids [BNR00, Bun04], low-rank tensor decomposition methods [Ose11b, OT10] and neural networks [TB18, YZ19b, HRM+21]. While the first two techniques rely on reducing the dimension of the basis expansion and therefore the number of degrees of freedom (DoFs), low-rank tensor decomposition methods start from the full tensor product basis ansatz and use compression methods to reduce the storage needed for the DoF tensor. Several tensor compression formats have been proposed in the literature [GKT13, Liu21], each having different advantages. Among them, the TT format [Ose11b, OT09] offers the best compromise between storage efficiency and robustness [Hac12]. The TT format has proven to be free of the curse of dimensionality in many surrogate modeling tasks [EMM20, KKNS14, DAIFS20, ZYO+14, SO11, DKO12, LRC+18, DKLM15, IWL+21, ILDG22]. Moreover, the result of most multilinear algebra operations can be directly expressed in the TT format if the operands are given in the TT format [Ose11b, SO11, OD12, DS14].

A further issue in the field of surrogate modeling is the computational complexity of computing the DoFs required for approximating of the quantity of interest. Depending on the available information about the

underlying model, we differentiate between black-box and white-box surrogate modeling. The black-box approach implies the model is evaluated for different parameter realizations. The parameter realizations together with the corresponding values of the quantity of interest form a label dataset. This falls under the umbrella of supervised learning and several methods have been proposed, from regression [LGDG20] to collocation [BNT07]. Moreover, adaptive input parameter sampling methods [GG03, LDG19] have also been proposed to reduce the number of solver calls. The main advantage of black-box surrogate modeling techniques is their versatility: constructing a surrogate model needs only a function handle of the parameter dependent problem.

As a comparison, white-box modeling techniques take advantage of the full description of the system and the solver is adapted to directly return the parameter dependent quantity of interest or solution. A typical approach is to extend the space where the quantity of interest lives in order to account for the parameter dependence. In the context of systems governed by differential equations, the Kronecker [VL00] product is used to construct the joint space. The corresponding DoFs can be recovered by solving the system that arises when performing Galerkin projection over the joint space [GS91, BTZ04]. Since the focus of the thesis lies on low-rank tensor decomposition methods, a tensor product basis is used over the parameter space. Moreover, it is beneficial for the discrete solution space to have a tensor product structure as well. In order to truly avoid the curse of dimensionality, all operations, from assembling the system to solving it and post-processing, have to be performed within the TT framework. The thesis addresses two different classes of problems: large systems of ordinary differential equations (ODEs) arising from the field of reaction kinetics [IWL$^+$21] and parameter dependent partial differential equations (PDEs) [ILDG22]. For both cases, the TT format was used to perform all the necessary steps.

The first application considered in this thesis is solving the chemical master equation (CME). The CME is a fundamental equation from the field of reaction kinetics that probabilistically describes the state of chemical reaction networks [Gil92]. Despite being simple at its core, the CME is prone to the curse of dimensionality since adding a new species results in a multiplication of the computational costs of solving the equation. This is exacerbated when parameter dependent CMEs are considered. Truncating the state space of the CME to a Cartesian box domain results in a tensor format of the solution. The TT compression scheme can therefore be used for the representation [KKNS14, DK15, DS20, IWL$^+$21]. Solving the parameter dependent CME in the TT format has been addressed in [DK15, IWL$^+$21]. In our contribution [IWL$^+$21], the solver was extended to directly perform parameter inference tasks.

The second application concerns the solution of parameter dependent boundary value problems (BVPs). The parameters range from material coefficients to variables in the geometry description. Conventional discretization techniques such as the finite element method (FEM) typically represent the solution on unstructured meshes [M$^+$03] making it difficult to cast the DoFs into a tensor. Therefore, the method of choice is IGA [HCB05, CHB09]. IGA is a very popular method that is applied in a variety of fields such as structural analysis [WWS13, CRBH06, MAB$^+$15, Rea06, SKBW10], electromagnetic field simulation [BCdF$^+$20, BSV10, DKSW19a, DKSW19b, SBdFS20], and fluid mechanics [ABKF11, BH08, GPC19, HAB11, LBJ19, WWX$^+$17]. The main idea behind it is to represent the geometry using B-spline or NURBS parametrizations. The geometry parametrization maps points from a reference domain (usually chosen as the unit cube) to points in the physical domain [HCB05]. The solution can also be defined on the reference domain, where a tensor product B-spline basis is used for its discretization. When performing the Galerkin projection, the integration over the parameter dependent geometry is transformed to integration over the reference domain with a modified metric. Therefore, the geometry information is moved into the coefficients of a modified PDE defined on the reference domain. Using a parameter dependent geometry map has been analyzed by several authors in the literature [CCNT16, CCNT21, HSS08, HSSS18, EMM20]. TT compression can be used to store the DoFs of the basis expansion as well as to construct the discrete Galerkin operators [ABC$^+$15, Hof18, MJKL17, ILDG22]. The assembly of the discrete operators for the

joint geometry-parameter domain in the TT format is more efficient than conventional assembly methods for a single parameter realization [ILDG22]. This holds for the system solver as well.

1.2. Contribution

The first main contribution of this thesis work is the development of the TT-CME framework for parameter dependent CMEs [IWL+21]. An explicit representation of the CME generator is given in the TT format without assembling the full operator. The parameter dependence is tackled by considering a tensor product basis expansion to obtain a joint state-parameter space. This increases the dimensionality of the underlying tensors. However, the assembly in the TT format still remains efficient. Compared to [DK15] where a collocation based method is used for the parameter dependence, in this work we employ the Galerkin approach to derive an extended system for the joint state-parameter domain. The presented framework is not restricted to handle only the reaction rates as parameters and can accommodate general propensity functions with different types of parameters as well as parameter dependent initial conditions (ICs). The alternating minimal energy (AMEn) solver is used to get the solution of the system for the joint state-parameter-time domain obtained after the time-domain discretization. Moreover, we leveraged the computationally efficient multilinear algebra operations in the TT format to address relevant inference problems from the field of chemical kinetics. The forward-backward algorithm is implemented in the TT format for state filtering and smoothing as well as for Bayesian parameter inference. Several examples are presented to demonstrate the efficiency of the method.

The second main contribution is the development of the TT-IGA framework [ILDG22]. The novelty of this work consists in deriving a TT representation for the tensor operators arising from the IGA discretization and using the AMEn solver to get the solution in the low-rank format. Compared to [MJKL17], the main focus is on tackling parametric geometry deformations, but the framework can also accommodate parameters governing the material laws or the boundary considerations. A recipe for constructing all the necessary discrete operators in the TT format is given. Collocation is used to handle the parameter dependency and, similarly to the TT-CME solver, the dimensionality of the tensors is increased. Moreover, both NURBS and B-spline geometry representations are taken into consideration in the framework. The method is benchmarked in the numerical results section and the obtained surrogates are further used to solve inverse problems.

All the developed methods are implemented in the `Python` programming language as two software packages: `tt-cme`[1] and `tt-iga`[2]. Supporting software was implemented in the `torchTT` library[3].

1.3. Structure of the thesis

The remaining of this thesis is structured as it follows: In Chapter 2, an introduction to tensor algebra is given. First, the general notion of tensor is defined and then, relevant topics from the field of multilinear algebra are recalled. In Chapter 3, the TT format is presented in detail, together with all the relevant algorithms needed in this work. Chapter 4 offers an overview of the field of Bayesian inverse problems. A brief introduction in probability theory is offered before presenting the Bayesian inversion formalism. In Chapter 5, the first application of the TT framework is presented in the context of reaction dynamics. The TT-CME solver for parameter dependent problems is introduced and numerical tests are performed for inference tasks. Chapter 6 concerns the use of the TT based surrogates for the parameter dependent

[1] https://github.com/ion-g-ion/tt-cme
[2] https://github.com/ion-g-ion/tt-iga
[3] https://github.com/ion-g-ion/torchTT

systems governed by PDEs. Numerical examples are also shown to showcase the performance of the method. Finally, the conclusion and the outlook are given in Chapter 7.

2. Fundamentals of tensor algebra

2.1. Introduction to tensors

The concept of a tensor has different definitions in different fields: as a multidimensional array, as a multilinear map over a cross product of vector spaces and their dual or by constructing the space using a tensor product of bases. In this chapter, we first introduce the concept of a tensor product of vector spaces. In the second part, the focus is put on tensors as multidimensional arrays. The notation is first introduced and then the relevant multilinear algebra operations are presented.

2.1.1. Tensor product

Before proceeding to the definition of the tensor product, we recall the concept of a multilinear map.

Definition 2.1.1 (Multilinear map)**.** Let $V_1, \dots, V_d$ be d vector spaces over a field $\mathbb{K}$. A map $\phi : \bigtimes_{k=1}^{d} V_k \to \mathbb{K}$ is called multilinear if

$$\begin{aligned}\phi(v_1, \dots, \alpha v_k + \beta v_k', \dots, v_d) =& \alpha\phi(v_1, \dots, v_k, \dots, v_d) + \beta\phi(v_1, \dots, v_k', \dots, v_d),\\ &\forall \alpha, \beta \in \mathbb{K}, v_k \in V_k, v_k' \in V_k, k \in \{1, \dots, d\}.\end{aligned} \tag{2.1}$$

The first definition of the tensor product is given using the universal property. The goal is to define a vector space S formed from 2 vector spaces U and V together with a map $\otimes : U \times V \to S$ with the property that it is "bilinear and nothing more" [Rom08].

Definition 2.1.2 (Universal property definition of tensor product)**.** Let U and V be two vector spaces over $\mathbb{K}$. The vector space denoted by $U \otimes V$ together with a bilinear mapping $\otimes : U \times V \to \mathbb{K}$ is called a tensor product space if it fulfills the universal property: for every bilinear mapping $\psi : U \times V \to \mathbb{K}$ there exists a unique linear map $\psi' : U \otimes V \to \mathbb{K}$ such that

$$\psi = \psi' \circ \otimes. \tag{2.2}$$

In Figure 2.1, a graphical representation of the universal property is given. The definition based on the universal property can be used to derive properties of the tensor product space, however it is not a constructive one. In the following, a constructive and coordinate-free definition of the tensor product of two vector spaces is given.

Construction of the tensor product

Let U, V be two vector spaces, $\mathcal{V}(U \times V)$ the space spanned by all elements from $U \times V$. Since a pair $(u, v) \in U \times V$ is a basis of the space $\mathcal{V}(U \times V)$, it cannot be represented as a linear combination of elements from $U \times V$. This violates the universal property stated in Definition 2.1.2. In order to exemplify this,

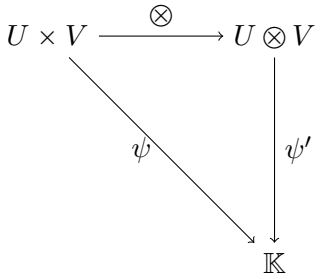


Figure 2.1.: Illustration of the universal property.

we consider 3 basis vectors (u, v), $(-u, v)$ and $(u - u, v) = (0, v)$ from $\mathcal{V}(U \times V)$. Since they are linearly independent, it holds

$$(0, v) \neq (u, v) + (-u, v). \tag{2.3}$$

In order to fulfill the universal property, additional relations have to be imposed. To this end, the subspace $\mathcal{Z}$ of $\mathcal{V}(U \times V)$ is defined as

$$\begin{aligned} \mathcal{Z} = &\{\alpha(u_1, v) + \beta(u_2, v) - (\alpha u_1 + \beta u_2, v) \ : \ u_1, u_2 \in U, v \in V, \alpha, \beta \in \mathbb{K}\} \cup \\ &\{\alpha(u, v_1) + \beta(u, v_2) - (u, \alpha v_1 + \beta v_2) \ : \ u \in U, v_1, v_2 \in V, \alpha, \beta \in \mathbb{K}\}. \end{aligned} \tag{2.4}$$

The tensor product of U and V is defined as the set quotient space

$$U \otimes V = \mathcal{V}(U \times V)/\mathcal{Z}. \tag{2.5}$$

The idea behind this is to restrict the larger vector space $\mathcal{V}(U \times V)$ by introducing the equivalence relations from (2.4). The construction fulfills the universal property introduced in Definition 2.1.2 (see [Hac12, Chapter 3] for the proof). For $u \in U$ and $v \in V$, the equivalence class constructed from u and v is denoted with $u \otimes v$.

Properties of the tensor product

Without proof, we present the following properties of the tensor product [Rom08]:

- $(\alpha u) \otimes v = u \otimes (\alpha v) = \alpha(u \otimes v), \forall \alpha \in \mathbb{K}, u \in U, v \in V.$
- $(u + u') \otimes v = u \otimes v + u' \otimes v, \forall u, u' \in U, v \in V,$
- $u \otimes (v + v') = u \otimes v + u \otimes v', \forall u \in U, v, v' \in V,$
- $0_V \otimes v = 0_{U \otimes V}, \forall v \in V$ and $u \otimes 0_V = 0_{U \otimes V}, \forall u \in U$, where 0_U, 0_V and $0_{U \otimes V}$ are the 0 tensors in the individual spaces.

Basis representation

For a vector space V with $\dim(V) = n < \infty$ and a particular choice of basis, an isomorphism between V and the $\mathbb{K}^n$ exists. This result can be generalized to a tensor product space $V_1 \otimes V_2 \otimes \cdots \otimes V_d$ with $\dim(V_k) = n_k < \infty, k \in \{1, \ldots, d\}$ [Hac12, Chapter 3]. Let $\{e_k^{(1)}\}_{k=1}^{n_k}$, $\{e_k^{(2)}\}_{k=1}^{n_k}$, …, $\{e_k^{(d)}\}_{k=1}^{n_k}$ be the bases of the individual spaces $V_1, V_2, \ldots, V_d$. The following statements hold [Hac12, Chapter 3]:

1. The set $\{e_{i_1}^{(1)} \otimes \cdots \otimes e_{i_d}^{(d)} \ : \ i_1 = 1, \ldots, n_1, \ldots, i_d = 1, \ldots, n_d\}$ is a basis of the tensor product space $\bigotimes_{k=1}^{d} V_k$ and any element $t \in \bigotimes_{k=1}^{d} V_k$ can be represented as

$$t = \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \mathsf{x}_{i_1 \ldots i_d} e_{i_1}^{(1)} \otimes \cdots \otimes e_{i_d}^{(d)}, \quad \mathbf{x} \in \mathbb{K}^{n_1 \times \cdots \times n_d}. \tag{2.6}$$

2. $\dim(\bigotimes_{k=1}^{d} V_k) = \prod_{k=1}^{d} n_k$.

3. For a fixed basis choice, the tensor product space $\bigotimes_{k=1}^{d} V_k$ is isomorphic to $\mathbb{K}^{n_1 \times \cdots \times n_d}$.

Let $U_1, \ldots, U_d$ and $V_1, \ldots, V_d$ be finite dimensional vector spaces with $\dim(V_k) = m_k$ and $\dim(U_k) = n_k$. We consider the vector space $\mathcal{L}(U_1 \otimes \cdots \otimes U_d, V_1 \otimes \cdots \otimes V_d)$, where $\mathcal{L}(A, B)$ denotes the set of all linear maps between the vector spaces A and B. Let $\phi \in \mathcal{L}(U_1 \otimes \cdots \otimes U_d, V_1 \otimes \cdots \otimes V_d)$. Since every element of the vector space $V_1 \otimes \cdots \otimes V_d$ can be expressed in terms of the basis $\{e_{i_1 \ldots i_d}^{V_1 \otimes \cdots \otimes V_d}\}_{i_1 \ldots i_d}$, the result of the function ϕ applied on the basis element $e_{j_1 \ldots j_d}^{U_1 \otimes \cdots \otimes U_d}$ can be expressed using the map $\beta : \left(\times_{k=1}^{d} \{1, \ldots, m_k\}\right) \times \left(\times_{k=1}^{d} \{1, \ldots, n_k\}\right) \to \mathbb{K}$

$$\phi(e_{j_1 \ldots j_d}^{U_1 \otimes \cdots \otimes U_d}) = \sum_{i_1, \ldots, i_d} \beta(i_1, \ldots, i_d, j_1, \ldots, j_d) e_{i_1 \ldots i_d}^{V_1 \otimes \cdots \otimes V_d}, \tag{2.7}$$

where $\{e_{j_1 \ldots j_d}^{U_1 \otimes \cdots \otimes U_d}\}_{j_1 \ldots j_d}$ is the basis of $U_1 \otimes \cdots \otimes U_d$. For every element $u \in U_1 \otimes \cdots \otimes U_d$ one can use its basis representation and the linearity of ϕ to obtain a representation in terms of the basis from $V_1 \otimes \cdots \otimes V_d$

$$\begin{aligned} \phi(u) = \phi\Big(\sum_{j_1 \ldots j_d} \mathsf{u}_{j_1 \ldots j_d} e_{j_1 \ldots j_d}^{U_1 \otimes \cdots \otimes U_d}\Big) = \sum_{j_1 \ldots j_d} \alpha(j_1, \ldots, j_d) \phi(e_{i_1 \ldots i_d}^{U_1 \otimes \cdots \otimes U_d}) = \\ \sum_{i_1, \ldots, i_d} \sum_{j_1, \ldots, j_d} \beta(i_1, \ldots, i_d, j_1, \ldots, j_d) \mathsf{u}_{j_1 \ldots j_d} e_{i_1 \ldots i_d}^{v_1 \otimes \cdots \otimes V_d}. \end{aligned} \tag{2.8}$$

This shows that once the bases are fixed, any element of $\mathcal{L}(U_1 \otimes \cdots \otimes U_d, V_1 \otimes \cdots \otimes V_d)$ can be described using a multidimensional array $\mathbf{B} \in \mathbb{K}^{m_1 \times \cdots \times m_d \times n_1 \times \cdots \times n_d}$. An element $\phi \in \mathcal{L}(U_1 \otimes \cdots \otimes U_d, V_1 \otimes \cdots \otimes V_d)$ is called in the following a **tensor operator**. Moreover, the following equality holds [Hac12]

$$\mathcal{L}\left(\bigotimes_{k=1}^{d} U_k, \bigotimes_{k=1}^{d} V_k\right) = \bigotimes_{k=1}^{d} \mathcal{L}(U_k, V_k). \tag{2.9}$$

2.2. Tensors as multidimensional arrays

As stated at the end of the previous section, tensor product spaces are isomorphic to the space of multidimensional arrays over the field of real numbers ($\mathbb{K} = \mathbb{R}$). In the following, we will refer to tensors as multidimensional arrays over the field of real numbers, such that $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ (or mapping from $\times_{k=1}^{d} \{1, \ldots, n_k\} \to \mathbb{R}$), unless otherwise stated. The dimensions of a tensor are also referred to as modes. As a convention, 1-dimensional tensors (vectors) are denoted with bold lower case letters, e.g. $\boldsymbol{a}, \boldsymbol{b}$, 2-dimensional tensors (matrices) are denoted by bold capital letters, e.g. $\boldsymbol{P}, \boldsymbol{C}$ and tensors with 3 or more dimensions are represented using lower case bold serif letters $\mathbf{x}, \mathbf{y}$. For tensor operators, upper case bold

serif letters are used (e.g. $\mathbf{A}, \mathbf{B}$). The multiindex notation is used to refer to an entry of a tensor, such that $\mathsf{x}_{i_1 \dots i_d} = \mathsf{x}_{\boldsymbol{i}}$, $\boldsymbol{i} = (i_1, \dots i_d) \in \times_{k=1}^{d} \{1, \dots, n_k\}$. In the case of tensor operators between 2 d-dimensional tensor product spaces, a comma is used to separate the first d indices, e.g. $\mathsf{A}_{\boldsymbol{i},\boldsymbol{j}}$.

Subtensors can be obtained by fixing or restricting some of the indices of a tensor. The colon is used to mark an index that is not restricted. For example, the j-th column of a matrix $\boldsymbol{A}$ is denoted by $A_{:j}$. By fixing all the indices but one, the resulting vector is called a fiber. If two of the indices are kept free and the rest are fixed, the resulting matrix is called a slice of the tensor. If we consider the tensor given in Figure 2.2, we have the following fibers and slices as examples:

$$\mathsf{x}_{1:3} = \begin{pmatrix} 3 \\ 7 \end{pmatrix}, \quad \mathsf{x}_{:1:} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 9 & 10 & 11 & 12 \\ 17 & 18 & 19 & 20 \end{pmatrix} \quad \mathsf{x}_{1::} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}. \tag{2.10}$$

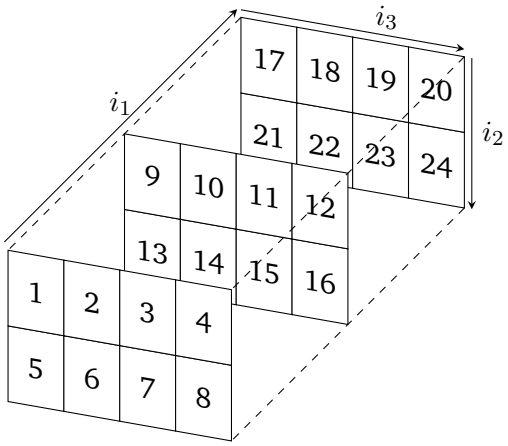


Figure 2.2.: 3-dimensional tensor $\mathsf{x} \in \mathbb{R}^{3\times2\times4}$.

2.2.1. Matricization

In the following, the matricization of a tensor is discussed. Before introducing the matricization operation, we present the multiindex raveling operation $\iota : \times_{k=1}^{d} \{1, \dots, n_k\} \to \{1, 2, \dots, n_1 \cdots n_d\}$. The mapping ι counts the multiindices of a given tensor and, using the big-endian convention, ι is defined as

$$\iota(i_1, \dots, i_d) = 1 + \sum_{k=1}^{d} \left((i_k - 1) \prod_{l=k+1}^{d} n_l \right). \tag{2.11}$$

The mapping ι is bijective with the inverse (also called index unraveling) given as

$$\iota_k^{-1}(i) = ((i-1) \,\%\, (n_2 \cdots n_d) \,\%\, (n_3 \cdots n_d) \,\%\, \cdots \,\%\, (n_k \cdots n_d)) \div \left(\prod_{l=k+1}^{d} n_l \right) + 1, \tag{2.12}$$

where $\%$ denotes the modulo operation (remainder of integer division) and $\div$ is the Euclidean division. In order to simplify the notation, the index raveling operation will be denoted with an overline $\overline{i_1 i_2 \dots i_d} := \iota(i_1, i_2, \dots, i_d)$. Moreover, it can also be used on subsets of indices of a tensor. Using the mapping ι, the

vectorization of a tensor (also known as unfolding or flattening in the literature) [KB09] is introduced. Given a tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, the vectorization $\mathcal{V}\mathbf{x} \in \mathbb{R}^{n_1 n_2 \cdots n_d}$ has the following elementwise definition

$$(\mathcal{V}\mathbf{x})_{\overline{i_1 \ldots i_d}} = \mathsf{x}_{i_1 \ldots i_d}. \tag{2.13}$$

The matrix unfolding $\mathcal{M}^{\leqslant k}\mathbf{x} \in \mathbb{R}^{(n_1 \cdots n_k) \times (n_{k+1} \cdots n_d)}$ of a tensor combines together the first k dimensions into the row mode of a matrix and the remaining $d-k$ into the column mode

$$(\mathcal{M}^{\leqslant k}\mathbf{x})_{\overline{i_1 i_2 \ldots i_k}, \overline{i_{k+1} \ldots i_d}} = \mathsf{x}_{i_1 i_2 \ldots i_d}. \tag{2.14}$$

The mode-k unfolding $\mathcal{M}^{=k}\mathbf{x}$ reshapes a tensor to a matrix by keeping the k-th mode as a leading mode and raveling the remaining dimensions into the second mode of a matrix

$$(\mathcal{M}^{=k}\mathbf{x})_{i_k, \overline{i_1 \ldots i_{k-1} i_{k+1} \ldots i_d}} = \mathsf{x}_{i_1 i_2 \ldots i_d}. \tag{2.15}$$

When dealing with a tensor operator $\mathbf{A} \in \mathbb{R}^{(m_1 \times \cdots \times m_d) \times (n_1 \times \cdots \times n_d)}$, we define the matricization operation $\mathcal{M}\mathbf{A} \in \mathbb{R}^{(m_1 \cdots m_d) \times (n_1 \cdots n_d)}$ elementwise as

$$\mathcal{M}\mathbf{A}_{\overline{i_1 i_2 \ldots i_d}, \overline{j_1 j_2 \ldots j_d}} = \mathsf{A}_{i_1 i_2 \ldots i_d, j_1 j_2 \ldots j_d}. \tag{2.16}$$

Example

As an example, we consider again the $3 \times 2 \times 4$ tensor illustrated in Figure 2.2. In this case, the unfolding of the tensor is given by

$$\mathcal{V}\mathbf{x} = (1, 2, 3, \ldots, 24)^\top, \tag{2.17}$$

the two matricizations are

$$\mathcal{M}^{\leqslant 1}\mathbf{x} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \end{pmatrix} \tag{2.18}$$

$$\mathcal{M}^{\leqslant 2}\mathbf{x} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 \end{pmatrix} \tag{2.19}$$

and the mode-2 unfolding is

$$\mathcal{M}^{=2}\mathbf{x} = \begin{pmatrix} 1 & 2 & 3 & 4 & 9 & 10 & 11 & 12 & 17 & 18 & 19 & 20 \\ 5 & 6 & 7 & 8 & 13 & 14 & 15 & 16 & 21 & 22 & 23 & 24 \end{pmatrix}. \tag{2.20}$$

2.2.2. Tensor algebra

In the following, an overview of the basic (multi)linear algebra operations (such as Kronecker product, elementwise addition, elementwise multiplication/Hadamard product, scalar product and norm and products between tensor operators) is given.

Kronecker product

Definition 2.2.1 (Kronecker product)**.** Let $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and $\mathbf{y} \in \mathbb{R}^{m_1 \times \cdots \times m_{d'}}$ be two tensors. The Kronecker product $\mathbf{x} \otimes \mathbf{y} \in \mathbb{R}^{n_1 \times \cdots \times n_d \times m_1 \times \cdots \times m_{d'}}$ is a $(d + d')$-dimensional tensor whose entries are given by

$$(\mathbf{x} \otimes \mathbf{y})_{\boldsymbol{ij}} = \mathsf{x}_{\boldsymbol{i}} \mathsf{y}_{\boldsymbol{j}}, \tag{2.21}$$

where $i_k \in \{1, \ldots, n_k\}, k \in 1, \ldots, d$ and $j_l \in \{1, \ldots, m_l\}, l \in 1, \ldots, d'$.

Elementwise addition

Definition 2.2.2 (Elementwise addition)**.** Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be two d-dimensional tensors. The sum $\mathbf{x} + \mathbf{y} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is defined elementwise as

$$(\mathbf{x} + \mathbf{y})_{\boldsymbol{i}} = \mathsf{x}_{\boldsymbol{i}} + \mathsf{y}_{\boldsymbol{i}}, \tag{2.22}$$

where $i_k \in \{1, \ldots, n_k\}, k \in 1, \ldots, d$.

When dealing with two tensor operators, the addition is performed analogously: $(\mathbf{A} + \mathbf{B})_{\boldsymbol{i},\boldsymbol{j}} = \mathsf{A}_{\boldsymbol{i},\boldsymbol{j}} + \mathsf{B}_{\boldsymbol{i},\boldsymbol{j}}$. The set of tensors with fixed modes together with the elementwise addition form an Abelian group. The neutral element is the zero (all entries are 0) tensor denoted in the following with $\mathbf{0}$.

Elementwise multiplication

Definition 2.2.3 (Elementwise multiplication)**.** Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be two d-dimensional tensors. The multiplication $\mathbf{x} \odot \mathbf{y} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is elementwise defined as

$$(\mathbf{x} \odot \mathbf{y})_{\boldsymbol{i}} = \mathsf{x}_{\boldsymbol{i}} \mathsf{y}_{\boldsymbol{i}}, \tag{2.23}$$

where $i_k \in \{1, \ldots, n_k\}, k \in 1, \ldots, d$.

In this work as well as in the specialized literature, the elementwise multiplication is also called Hadamard product.

Scalar product and Frobenius norm

Definition 2.2.4 (Scalar product)**.** Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be two d-dimensional tensors. The scalar product denoted by $\mathbf{x} \cdot \mathbf{y} \in \mathbb{R}$ is defined as

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \mathsf{x}_{\boldsymbol{i}} \mathsf{y}_{\boldsymbol{i}} = \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}} \mathsf{x}_{\boldsymbol{i}} \mathsf{y}_{\boldsymbol{i}}. \tag{2.24}$$

The norm induced by the presented scalar product is called the Frobenius norm $||\mathbf{x}||_F := \sqrt{\mathbf{x} \cdot \mathbf{x}}$. It can also be defined for tensor operators as $||\mathbf{A}||_F^2 = \sum_{\boldsymbol{i},\boldsymbol{j}} \mathsf{A}_{\boldsymbol{i},\boldsymbol{j}}^2$.

Product with a tensor operator

Similarly to the matrix-vector product, a generalization for tensors using multiindex notation can be given. The two presented products are the analog of matrix-vector/matrix-matrix products and are equivalent under the unfolding operation.

Definition 2.2.5 (Tensor operator product)**.** Given two tensor operators $\mathbf{A} \in \mathbb{R}^{m_1 \times \cdots \times m_d \times n_1 \times \cdots \times n_d}$, $\mathbf{B} \in \mathbb{R}^{n_1 \times \cdots \times n_d \times p_1 \times \cdots \times p_d}$ and a tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. The product $\mathbf{Ax} \in \mathbb{R}^{m_1 \times \cdots \times m_d}$ is elementwise defined as

$$(\mathbf{Ax})_{\boldsymbol{i}} = \sum_{\boldsymbol{j}} \mathsf{A}_{\boldsymbol{i},\boldsymbol{j}} \mathsf{x}_{\boldsymbol{j}}, \tag{2.25}$$

for $i_k \in \{1, \ldots, m_k\}, k \in 1, \ldots, d$. The product $\mathbf{AB} \in \mathbb{R}^{m_1 \times \cdots \times m_d \times p_1 \times \cdots \times p_d}$ between $\mathbf{A}$ and $\mathbf{B}$ is defined as

$$(\mathbf{AB})_{\boldsymbol{i},\boldsymbol{j}} = \sum_{\boldsymbol{l}} \mathsf{A}_{\boldsymbol{i},\boldsymbol{l}} \mathsf{B}_{\boldsymbol{l},\boldsymbol{j}}, \tag{2.26}$$

where $i_k \in \{1, \ldots, n_k\}$ and $j_k \in \{1, \ldots, p_k\}, k \in 1, \ldots, d$.

With the operations defined in (2.13) and (2.16), the following holds

$$\mathcal{V}(\mathbf{Ax}) = (\mathcal{M}\mathbf{A})\mathcal{V}\mathbf{x}, \quad \mathcal{M}(\mathbf{AB}) = (\mathcal{M}\mathbf{A})(\mathcal{M}\mathbf{B}). \tag{2.27}$$

Using the defined products, multilinear systems $\mathbf{Ax} = \mathbf{b}$ as well as the inverse $\mathbf{A}^{-1}$ of a tensor operator $\mathbf{A}$ can be introduced:

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}. \tag{2.28}$$

n-mode product

In addition to the previously presented products, we also define the product between a tensor and a matrix, called the n-mode product.

Definition 2.2.6 (n-mode product)**.** Let $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a d-dimensional tensor, $q \in \{1, \ldots, d\}$ an index and $\boldsymbol{A} \in \mathbb{R}^{m_q \times n_q}$ a matrix. The product along the n-th mode $\mathbf{x} \times_q \boldsymbol{A} \in \mathbb{R}^{n_1 \times \cdots \times n_{q-1} \times m_q \times n_{q+1} \times \cdots \times n_d}$ is elementwise defined as

$$(\mathbf{x} \times_q \boldsymbol{A})_{i_1 \ldots i_{q-1} j i_{q+1} \ldots i_d} = \sum_{i_q} \mathsf{x}_{i_1 \ldots i_q \ldots i_d} A_{j i_q}, \tag{2.29}$$

where $i_k \in \{1, \ldots, n_k\}, k \in \{1, \ldots, d\}, k \neq q, j \in \{1, \ldots, m_q\}$.

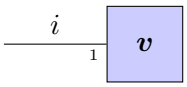
If the tensor $\mathbf{x}$ is the basis representation of a multilinear operator ϕ, the n-mode product across dimension q represents a basis change. When computing the n-mode product with a vector, the resulting tensor is $(d-1)$-dimensional

$$(\mathbf{x} \times_q \boldsymbol{a})_{i_1 \ldots i_{q-1} i_{q+1} \ldots i_d} = \sum_{i_q} \mathsf{x}_{i_1 \ldots i_{q-1} i_q i_{q+1} \ldots i_d} a_{i_q}. \tag{2.30}$$

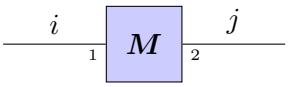
If the vector is $\boldsymbol{a} = \mathbf{1}$, the operation performs the contraction of the q-th index (summing across the q-th dimension).

2.2.3. Tensor diagram notation

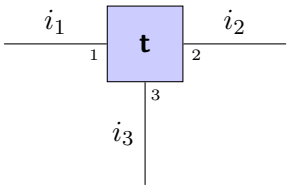
First introduced in [Pen71], the tensor diagram notation is used to graphically represent index contraction applied to tensors. In the literature, it is also found under the name of tensor networks representation [Pen71]. The main idea behind it is very simple: tensors are denoted by nodes in a graph (see Figure 2.3) and a connection between nodes represents an index contraction. On every end of a link connected to a tensor, a number is specified, representing the number of the mode. Additionally, indices that are not contracted are represented using a line that is linked to only one tensor (see Figure 2.3). The number of links that are connected to a tensor is equal to the number of dimensions of the respective tensor. On every link, the contraction index or the mode size is specified.



(a) Representation of a vector $\boldsymbol{v}$. The index denoted with i is in this case the first and only index and is denoted with 1.



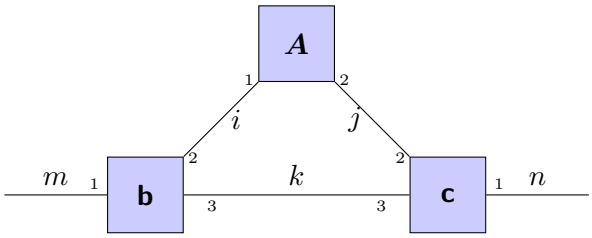
(b) Representation of a matrix $\boldsymbol{M}$. The order of the two indices i and j is specified by the numbers on the edges.



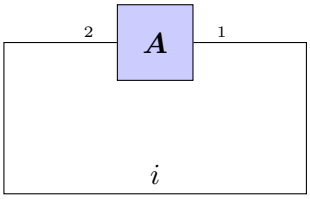
(c) Representation of a 3-dimensional tensor **t**.

Figure 2.3.: Graphical representation of a vector, a matrix and a 3-dimensional tensor. The tensors are represented by the blue squares and the indices are denoted by the one-way links.

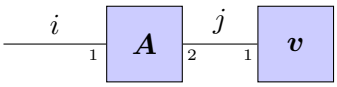
In Figure 2.4a, the tensor that results from $\sum_{i,j,k} A_{ij}\mathsf{b}_{mik}\mathsf{c}_{njk}$ is illustrated. The resulting tensor has only 2 dimensions, denoted with the indices m and n (the edges that are connected to a node at only one end). The sum is performed over the indices i, j, k as denoted by the graph edges. Using the tensor network notation, the trace of a matrix can also be represented (see Figure 2.4b) by constructing an edge that starts and ends in the same tensor. One more example, depicted in Figure 2.4c, is the well known matrix-vector product.



(a) Representation of index contraction $\sum_{i,j,k} A_{ij} \mathsf{b}_{mik} \mathsf{c}_{njk}$.



(b) Representation of the trace $\sum_i A_{ii}$ of a matrix $\boldsymbol{A}$.



(c) Graphical representation of the matrix vector product $\sum_j A_{ij} v_j$.

Figure 2.4.: Example of tensor diagram notation for visualizing index contractions.

3. Low-rank tensor decomposition

As already stated in the previous chapter, storing all the entries of a tensor in the memory can become problematic even for a moderate number of dimensions due to the curse of dimensionality. Data compression is therefore needed to reduce the computational costs. Compared to conventional source coding methods from information theory, where the compression is done using the binary representation of the data, the tensor compression methods used in this work rely on approximating a tensor using a series of algebraic operations on several smaller tensors. As it will be shown in the following for the TT format, the advantage is that the basic linear algebra operations defined on the full tensors can be performed on their the low-rank representations. In this chapter, we first give an overview of two of the first and most popular low-rank decomposition formats. The TT format is then presented in detail.

3.1. Overview of low-rank tensor decomposition formats

3.1.1. Canonical polyadic decomposition

Introduced in [Hit] under the name of polyadic decomposition, the canonical polyadic decomposition (CPD) format can be found under different names in the literature: CANDECOMP or PARAFAC [KB09]. Before defining the canonical format, we introduce the concept of rank-1 tensors.

Definition 3.1.1 (Rank-1 tensor)**.** A tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is called rank-one if it can be expressed as the Kronecker product of d vectors $\boldsymbol{v}^{(k)} \in \mathbb{R}^{n_k}, k = 1, \ldots, d$

$$\mathbf{x} = \boldsymbol{v}^{(1)} \otimes \cdots \otimes \boldsymbol{v}^{(d)}. \tag{3.1}$$

Definition 3.1.2 (CPD)**.** The CPD of a d-dimensional tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is defined as the superposition of r rank-1 tensors

$$\mathbf{x} = \sum_{k=1}^{r} \boldsymbol{v}^{(k,1)} \otimes \cdots \otimes \boldsymbol{v}^{(k,d)}, \tag{3.2}$$

where $r \in \mathbb{N}$ and $\{\boldsymbol{v}^{(k,l)}\}_{k,l}$ are the CPD factors. The minimum number of rank-1 tensors to represent $\mathbf{x}$ is called the tensor rank and is denoted by rank($\mathbf{x}$).

The storage complexity for storing a rank r CPD is $\mathcal{O}(dnr)$, being linear w.r.t. the number of dimensions. The main drawback of the CPD is finding the best rank r approximation of a given tensor. As shown in [Hå90], the tensor rank cannot be found in a finite number of steps. This arises due to the fact that there can exist a sequence of tensors with a CPD of rank r' converging to a tensor of rank $r > r'$ [KB09]. Moreover, for practical purposes, the CPD is not necessarily the most expressive way of compressing tensors (i.e. a high rank might be needed in order to obtain a good approximation).

3.1.2. Tucker decomposition

First introduced in [Tuc66] and also known in the literature under the name of higher-order SVD (HOSVD) representation [DLDMV00a], the Tucker decomposition takes advantage of the n-mode product in order to perform a basis transformation of a tensor.

Definition 3.1.3 (Tucker decomposition)**.** A tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is in the Tucker format if it can be represented as

$$\mathbf{x} = \mathbf{g} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} \cdots \times_d \boldsymbol{U}^{(d)}, \tag{3.3}$$

where $\mathbf{g} \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ is the Tucker core, $\boldsymbol{U}^{(k)} \in \mathbb{R}^{n_k \times r_k}, k = 1, \ldots, d$ and $\boldsymbol{r} = (r_1, \ldots, r_d)$ is the Tucker rank of the decomposition.

A representation of the Tucker decomposition in the tensor network language is given in Figure 3.1 for the case $d = 4$. The Tucker rank of a given tensor $\mathbf{x}$ is a tuple $(r_1, \ldots, r_d)$ containing the ranks of the matricizations along all the modes, i.e., $r_k = \operatorname{rank}(\mathcal{M}^k\mathbf{x})$. Any tensor admits a decomposition in the form of (3.3) with the rank equal to its Tucker rank [DLDMV00a]. The storage requirement for the Tucker format is $\mathcal{O}(r^d + dnr)$, $r = \max_k r_k$, which is exponential w.r.t. the number of dimensions. The Tucker format does not overcome the curse of dimensionality due to the need of storing the Tucker core. However, if the Tucker rank is small enough, a significant reduction of the storage is obtained. The CPD is a special case of the Tucker decomposition if the factor matrices $\boldsymbol{U}$ contain the CPD vectors as columns and the Tucker core is the diagonal tensor.

Computing an exact Tucker decomposition can be performed using the HOSVD algorithm [DLDMV00a]. The factor matrices $\boldsymbol{U}^{(k)}$ are obtained by iteratively performing the singular value decompositions (SVDs) on the matricizations $\mathcal{M}^k\mathbf{x}$ and taking the lead left singular vectors (with nonzero corresponding singular values). The Tucker core is then obtained using the n-mode product $\mathbf{g} = \mathbf{x} \times_1 \boldsymbol{U}^{(1)\top} \cdots \times_d \boldsymbol{U}^{(d)\top}$. However, the HOSVD is not suited to perform best Tucker approximation of a tensor within a prescribed accuracy. To this end, De Lathauwer, De Moor, and Vandewalle developed the higher-order orthogonal iteration (HOOI) method [DLDMV00b] to find the best Tucker approximation.

3.2. Tensor-train format

One popular format in terms of computational complexity and robustness is the tensor-train (TT) format [Ose11b], which can also be found in the literature under the name of matrix-product states (MPS) [VC06, Vid03]. It combines the excellent linear scaling w.r.t. number of dimensions of the CPD format with the robustness of the Tucker format. At its core, the TT decomposition is the generalization of the separation of variables applied on multiindices.

Definition 3.2.1 (Tensor-train format)**.** A tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is said to be in the TT format if its elements can be expressed as

$$\mathrm{x}_{\boldsymbol{i}} = \sum_{s_1=1}^{r_1} \cdots \sum_{s_{d-1}=1}^{r_{d-1}} \prod_{k=1}^{d} \mathrm{g}^{(k)}_{s_{k-1} i_k s_k} = (\mathcal{TT}(\mathbf{g}^{(1)}, \ldots, \mathbf{g}^{(d)}))_{\boldsymbol{i}}, \tag{3.4}$$

where $\mathbf{g}^{(k)} \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ are called the TT cores and the vector $\boldsymbol{r} = (r_0, r_1, \ldots, r_d)$ with $r_0 = r_d = 1$ is the TT rank.

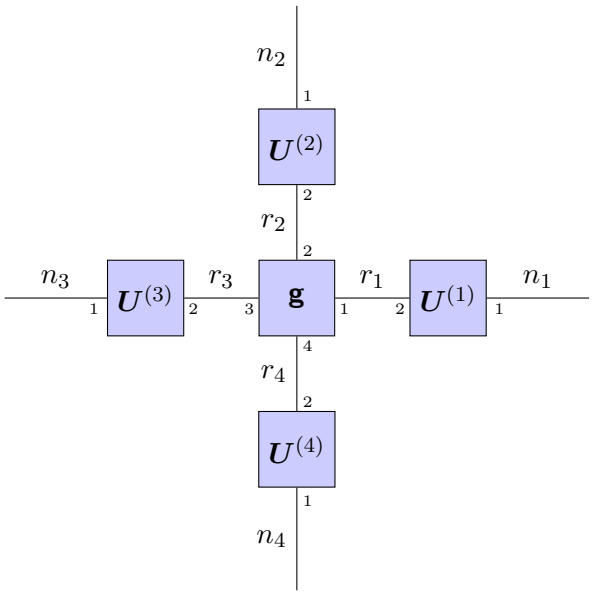
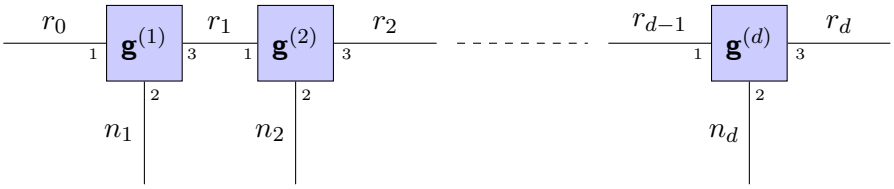
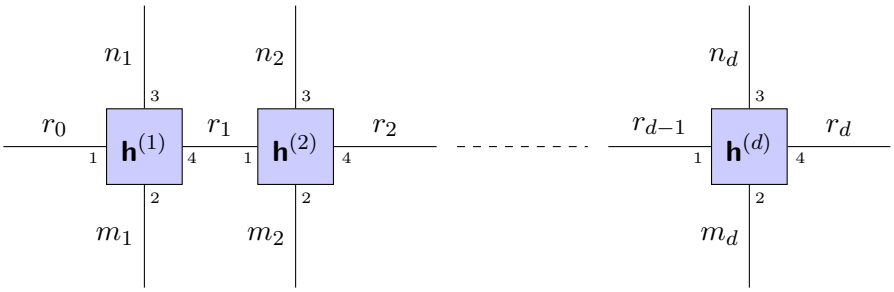


Figure 3.1.: Graphical representation of the Tucker decomposition of a 4-dimensional tensor. (r_1, r_2, r_3, r_4) represents the Tucker rank.



(a) Tensor train



(b) Tensor operator in TT format.

Figure 3.2.: Graphical representation of the TT format.

In Figure 3.2a, the tensor network representation of the TT decomposition is shown. The series connection of the TT cores (also called carriages) motivates the name tensor-train. Having to store only d TT cores, the storage requirement reduces from $\mathcal{O}(n^d)$ to $\mathcal{O}(dnr^2)$ where $r = \max_{k=1,\dots,d} r_k$. However, in practice most tensors do not possess an exact decomposition of type (3.4) with a low rank and an approximation

has to be made, so a trade-off between the TT rank and the accuracy has to be made.

A similar decomposition format can be defined for a tensor operator $\mathbf{A} \in \mathbb{R}^{m_1 \times \cdots \times m_d \times n_1 \times \cdots \times n_d}$ defined in Section 2.1.1:

$$\mathsf{A}_{\boldsymbol{i},\boldsymbol{j}} = \sum_{s_1=1}^{r_1} \cdots \sum_{s_{d-1}=1}^{r_{d-1}} \prod_{k=1}^{d} \mathsf{h}^{(k)}_{s_{k-1} i_k j_k s_k}. \tag{3.5}$$

In this case the TT cores $\mathbf{h}^{(k)} \in \mathbb{R}^{r_{k-1} \times m_k \times n_k \times r_k}$, $k \in 1, \ldots, d$ are 4d tensors, making the storage complexity $\mathcal{O}(dr^2mn)$ (a tensor network representation is given in Figure 3.2b). However, by permuting the dimensions of the tensor operator $\mathbf{A}$ and reshaping, one can obtain a d-dimensional tensor $\mathbf{a} \in \mathbb{R}^{m_1 n_1 \times \cdots \times m_d n_d}$ with

$$\mathsf{A}_{i_1 \ldots i_d, j_1 \ldots i_j} = \mathsf{a}_{\overline{i_1 j_1}\, \overline{i_2 j_2} \ldots \overline{i_d j_d}}. \tag{3.6}$$

If a TT decomposition of $\mathbf{a}$ is computed with the TT cores $\mathbf{g} \in \mathbb{R}^{r_{k-1} \times m_k n_k \times r_k}$, the cores of the TT decomposition of the tensor operator $\mathbf{A}$ can be obtained by reshaping the 3d cores into $r_{k-1} \times m_k \times n_k \times r_k$ tensors.

The TT decomposition is not unique. If a decomposition of a tensor $\mathbf{x}$ with the TT cores $\{\mathbf{g}^{(k)}\}_{k=1,\ldots,d}$ is found, one obtains the same full tensor after modifying any 2 consecutive cores

$$\tilde{\mathbf{g}}^{(k)}_{s_{k-1} i_k :} = \boldsymbol{R}^\top \mathbf{g}^{(k)}_{s_{k-1} i_k :}, \quad \tilde{\mathbf{g}}^{(k+1)}_{: i_k s_{k+1}} = \boldsymbol{R}^{-1} \mathbf{g}^{(k+1)}_{: i_k s_{k+1}}, \tag{3.7}$$

where $\boldsymbol{R} \in \mathbb{R}^{r_k \times r_k}$ is an invertible matrix. One further aspect of the TT format is the ordering of the indices. Compared to other low-rank formats (such as CPD and Tucker for example), permuting the indices does affect the TT rank of the decomposition. The intuition behind this is that the information has to flow through the cores that separate two modes. As an example, we consider the tensor $\mathbf{x} = \boldsymbol{I}_4 \otimes \mathbf{1}_4 \in \mathbb{R}^{4 \times 4 \times 4}$, where $\boldsymbol{I}_4 \in \mathbb{R}^{4 \times 4}$ is the identity matrix and $\mathbf{1}_4 \in \mathbb{R}^4$ is the one vector (alternatively one can write the tensor as x_{ijk}). An exact decomposition of $\mathbf{x}$ has the rank $\boldsymbol{r} = (1, 4, 1, 1)$. However, if the second and the third modes are permuted, the exact decomposition of the tensor now has the TT rank $\boldsymbol{r} = (1, 4, 4, 1)$.

In the following, the concept of orthonormal cores is introduced. Its importance will become clear in Section 3.2.2 where the rank rounding operation is defined as well as in Section 3.2.4.

Definition 3.2.2. Let $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a tensor with the TT decomposition given by the cores $\{\mathbf{g}^{(k)}\}_k$.

- The k-th core is called left-orthonormal if the rows of the left unfolding $\mathcal{L}\mathbf{g}^{(k)} = \mathcal{M}^{\leqslant 2}\mathbf{g}^{(k)} \in \mathbb{R}^{r_{k-1} n_k \times r_k}$ are orthonormal, i.e.

$$\left(\mathcal{L}\mathbf{g}^{(k)}\right)^\top \mathcal{L}\mathbf{g}^{(k)} = I \in \mathbb{R}^{r_k \times r_k}. \tag{3.8}$$

- The k-th core is called right-orthonormal if the columns of the right unfolding $\mathcal{R}\mathbf{g}^{(k)} = \mathcal{M}^{\leqslant 1}\mathbf{g}^{(k)} \in \mathbb{R}^{r_{k-1} \times n_k r_k}$ are orthonormal, i.e.

$$\mathcal{R}\mathbf{g}^{(k)} \left(\mathcal{R}\mathbf{g}^{(k)}\right)^\top = I \in \mathbb{R}^{r_{k-1} \times r_{k-1}}. \tag{3.9}$$

In practice, one can use the QR decomposition to orthonormalize the cores of a given TT representation. Let $k \in \{1, .., d-1\}$ be an index and $\boldsymbol{C} = \mathcal{L}\mathbf{g}^{(k)}$ the left unfolding of the k-th core. By performing the QR decomposition, the matrices $\boldsymbol{Q}, \boldsymbol{R}$ are obtained such that $\boldsymbol{QR} = \mathcal{L}\mathbf{g}^{(k)}$. The new left-orthonormal core $\tilde{\mathbf{g}}^{(k)}$ can be obtained by reshaping the $\boldsymbol{Q}$ matrix and the next core is updated as $\tilde{\mathbf{g}}^{(k+1)}_{: i_{k+1} s} = \boldsymbol{R}\mathbf{g}^{(k+1)}_{: i_{k+1} s}$. If this operation is applied sequentially, starting from $k = 1$ to $k = d - 1$, the resulting train has the first $d - 1$ TT cores left-orthogonal. The pseudocode is presented in Algorithm 1. Using slight modifications, one can also compute a train where all the cores but the first one are right-orthonormal (see Algorithm 2). In both cases, the complexity is $\mathcal{O}(dnr^3)$ [Ose11b] and during the process the rank can be decreased.

Algorithm 1 Left-to-right orthonormalization of the TT cores.

procedure ORTHOLR($\{\mathbf{g}^{(k)}\}_k, \boldsymbol{r}$)
  for $k \in \{1, \dots, d-1\}$ **do**
    $\boldsymbol{C} \leftarrow \text{reshape}(\mathbf{g}^{(k)}, r_{k-1}n_k \times r_k)$.
    $\boldsymbol{Q}, \boldsymbol{R} \leftarrow \text{QR}(\boldsymbol{C})$
    $r_k \leftarrow \#\text{columns}(\boldsymbol{Q})$.
    $\mathbf{g}^{(k)} \leftarrow \text{reshape}(\boldsymbol{Q}, r_{k-1} \times n_k \times r_k)$.
    $\mathrm{g}^{(k+1)}_{ijk} \leftarrow \sum_l R_{il}\mathrm{g}^{(k+1)}_{ljk}$.
return $\{\mathbf{g}^{(k)}\}_k, \boldsymbol{r}$.

Algorithm 2 Right-to-left orthonormalization of the TT cores.

procedure ORTHORL($\{\mathbf{g}^{(k)}\}_k, \boldsymbol{r}$)
  for $k \in \{d, \dots, 2\}$ **do**
    $\boldsymbol{C} \leftarrow \text{reshape}(\mathbf{g}^{(k)}, r_{k-1} \times n_k r_k)$.
    $\boldsymbol{Q}, \boldsymbol{R} \leftarrow \text{QR}(\boldsymbol{C}^\top)$
    $r_{k-1} \leftarrow \min(\#\text{columns}(\boldsymbol{Q}), \#\text{rows}(\boldsymbol{Q}))$.
    $\mathbf{g}^{(k)} \leftarrow \text{reshape}(\boldsymbol{Q}^\top, r_{k-1} \times n_k \times r_k)$.
    $\mathrm{g}^{(k-1)}_{ijk} \leftarrow \sum_l \mathrm{g}^{(k-1)}_{ijl} R_{il}$.
return $\{\mathbf{g}^{(k)}\}_k, \boldsymbol{r}$.

3.2.1. Conversion from full format

One important advantage of the TT format is the ability to compute a TT approximation of a given tensor within a prescribed relative accuracy. The problem is semidiscrete since the TT ranks as well as the TT cores have to be determined simultaneously.

Problem 3.2.1 (Conversion from full format to TT)**.** Given a tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and an $\epsilon \geqslant 0$, determine the TT cores $\{\mathbf{g}^{(k)}\}_k$ such that

$$||\mathbf{x} - \tilde{\mathbf{x}}||_F \leqslant \epsilon ||\mathbf{x}||_F, \quad \text{for } \tilde{\mathrm{x}}_{\boldsymbol{i}} = \sum_{s_1=1}^{r_1} \cdots \sum_{s_{d-1}=1}^{r_{d-1}} \prod_{k=1}^{d} \mathrm{g}^{(k)}_{s_{k-1} i_k s_k}, \tag{3.10}$$

where the TT rank of $\tilde{\mathbf{x}}$ has to be minimal.

Computing the TT decomposition can be performed by recursively applying the SVD on matrix unfoldings in order to individually separate the TT cores [Ose11b, OT09]. Let $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be the tensor that we want to decompose and $\epsilon \geqslant 0$ (the case $\epsilon = 0$ means an exact TT decomposition). If the matricization $\mathcal{M}^{\leqslant 1}\mathbf{x} \in \mathbb{R}^{n_1 \times (n_2 \cdots n_d)}$ is considered, one can apply the SVD

$$\mathcal{M}^{\leqslant 1}\mathbf{x} = \boldsymbol{U}\mathbf{diag}(\boldsymbol{\sigma}^2)\boldsymbol{V}^\top, \tag{3.11}$$

where $\boldsymbol{U} \in \mathbb{R}^{n_1 \times n_1}, \boldsymbol{V} \in \mathbb{R}^{n_1 \times (n_2 \cdots n_d)}$ and $\boldsymbol{\sigma} \in \mathbb{R}^{n_1}$ is the vector of singular values (in descending order). The factor matrices can be truncated by keeping only the first $r_1 \leqslant n_1$ columns of $\boldsymbol{U}, \boldsymbol{V}$ and the first singular values. The relative error in Frobenius norm is in this case determined as

$$||\boldsymbol{U}\mathbf{diag}(\boldsymbol{\sigma}^2)\boldsymbol{V}^\top - \tilde{\boldsymbol{U}}\mathbf{diag}(\tilde{\boldsymbol{\sigma}}^2)\tilde{\boldsymbol{V}}^\top||_F \leqslant \sigma^2_{r_1+1} + \cdots + \sigma^2_{n_1}, \tag{3.12}$$

where $\tilde{\boldsymbol{U}} \in \mathbb{R}^{n_1 \times r_1}$, $\tilde{\boldsymbol{V}} \in \mathbb{R}^{r_1 \times (n_2 \cdots n_d)}$, $\tilde{\boldsymbol{\sigma}}$ are the truncated factors. The first factor matrix can be identified with the first core of a TT decomposition if an additional mode of size 1 is added. The matrix $\mathbf{diag}(\tilde{\boldsymbol{\sigma}}^2)\tilde{\boldsymbol{V}}^\top$ can be reshaped to an $r_1 n_2 \times n_3 \cdots n_d$ matrix and an SVD can be further applied. A truncation to $r_2 \leqslant r_1 n_2$ can be performed, resulting to the factor matrices $\tilde{\boldsymbol{U}}' \in \mathbb{R}^{r_1 n_2 \times r_2}$, $\tilde{\boldsymbol{V}}' \in \mathbb{R}^{r_2 \times (n_3 \cdots n_d)}$ and the vector of singular values $\tilde{\boldsymbol{\sigma}} \in \mathbb{R}^{r_2}$. The matrix $\tilde{\boldsymbol{U}}'$ represents the second core of a TT decomposition after reshaping it to $r_1 \times n_2 \times r_2$. The process of performing truncated SVDs can be further applied in order to obtain the remaining cores. An example for the case $d = 3$ is presented in Figure 3.3, where the individual steps are illustrated using tensor network diagrams.

One important topic is how to truncate the rank such that the global relative error is smaller than the desired value. At the same time, the TT rank should not be overestimated. In [Ose11b], it has been proven that the optimal rank can be chosen according to

$$r_k = \operatorname*{argmax}_l \sqrt{\sigma_{k,l}^2 + \sigma_{k,l+1}^2 + \cdots} < \frac{\epsilon}{\sqrt{d-1}} \sqrt{\sum_s \sigma_s^2}\,, \tag{3.13}$$

where $\{\sigma_{k,l}\}_l$ are the singular values computed in the k-th step. The complete pseudocode is given in Algorithm 3. If the main loop is performed from right to left, i.e. starting with the last index, the TT cores are right orthogonal.

Algorithm 3 Conversion from full tensor to TT format.

procedure FULL2TT($\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, ϵ)
  $r_0 \leftarrow 1$.
  $\boldsymbol{X}^{(1)} \leftarrow \text{reshape}(\mathbf{x}, r_0 n_1 \times n_2 \cdots n_d)$.
  for $k \in \{1, \ldots, d-1\}$ **do**
    $\boldsymbol{U}, \boldsymbol{\sigma}, \boldsymbol{V} \leftarrow \text{SVD}(\boldsymbol{X}^{(k)})$.
    $r_k \leftarrow \max_l(\sigma_l^2 + \sigma_{l+1}^2 + \cdots) < \frac{\epsilon^2}{d-1}\left(\sum_s \sigma_s^2\right)$.
    $\tilde{\boldsymbol{U}}, \tilde{\boldsymbol{\sigma}}, \tilde{\boldsymbol{V}} \leftarrow \boldsymbol{U}_{:1:r_k}, \boldsymbol{\sigma}_{1:r_k}, \boldsymbol{V}_{:1:r_k}$.
    $\mathbf{g}^{(k)} \leftarrow \text{reshape}(\tilde{\boldsymbol{U}}, r_{k-1} \times n_k \times r_k)$.
    $\boldsymbol{X}^{(k+1)} \leftarrow \text{reshape}(\mathbf{diag}(\tilde{\boldsymbol{\sigma}}^2)\tilde{\boldsymbol{V}}^\top, r_k n_{k+1} \times n_{k+2} \cdots n_d)$.
  $r_d \leftarrow 1$.
  $\mathbf{g}^{(d)} \leftarrow \text{reshape}(\boldsymbol{X}^{(d)}, r_{d-1} \times n_d r_d)$.
return $\{\mathbf{g}^{(k)}\}_k$.

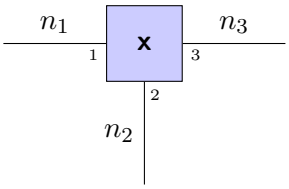
3.2.2. Rank rounding

One further operation in the TT format is the rank rounding, i.e., finding a more efficient TT representation of a tensor that is already given in the TT format. The formal problem statement is the following:

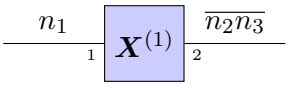
Problem 3.2.2 (TT rank rounding)**.** Given a tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ with a TT representation of rank $\boldsymbol{r}$ and $\epsilon > 0$, find a tensor $\tilde{\mathbf{x}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ with a TT representation of rank $\tilde{\boldsymbol{r}}$ with

$$||\mathbf{x} - \tilde{\mathbf{x}}||_F \leqslant \epsilon ||\mathbf{x}||_F, \quad r_k \leqslant \tilde{r}_k, \ \forall k. \tag{3.14}$$

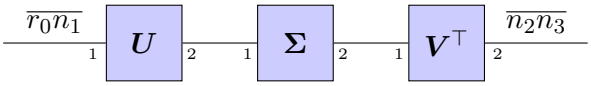
The naive approach would be to convert an existing TT decomposition to full format and then apply Algorithm 3 to obtain the new TT cores. However, this is not computationally feasible for large tensors. In [OT10], a more efficient way is proposed to apply Algorithm 3 for a tensor already represented in the TT



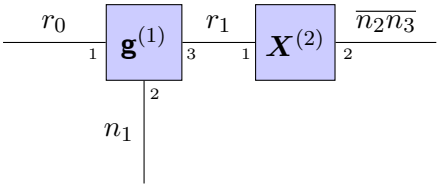
(a) The original 3-dimensional tensor.



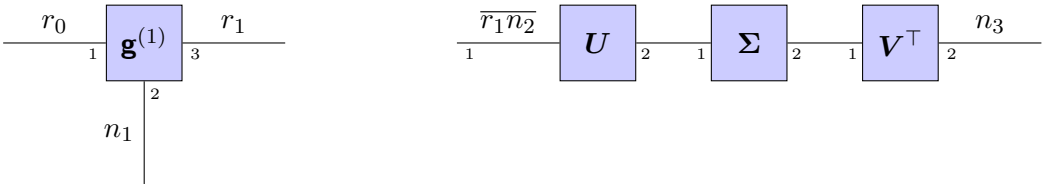
(b) The tensor is reshaped to an $n_1 \times (n_2 n_3)$ matrix.



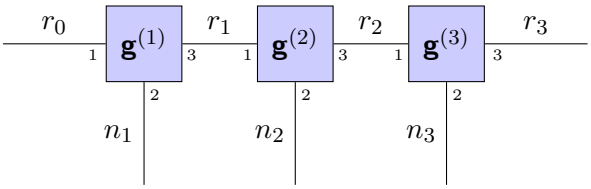
(c) SVD is applied on the matrix $\boldsymbol{X}^{(1)}$ from the previous step. The additional dimension corresponding to the first rank $r_0 = 1$ is added for completeness.



(d) The truncation of the singular values is performed, the truncated factor matrix $\boldsymbol{U}$ is reshaped into the first core while the rest is grouped in a matrix $\boldsymbol{X}^{(2)}$.



(e) The matrix $\boldsymbol{X}^{(2)}$ is reshaped to $(r_1 n_2) \times n_3$ and the SVD is applied on it to reveal the second TT rank.



(f) The truncated factor matrix is reshaped into the second core $\mathbf{g}^{(2)}$ and the remaining product is truncated and reshaped into the last core $\mathbf{g}^{(3)}$.

Figure 3.3.: Graphical illustration of the steps needed to decompose an $n_1 \times n_2 \times n_3$ tensor in the TT format. The steps are presented in the corresponding figures, in alphabetical order.

format. Let $\{\mathbf{g}^{(k)}\}_k$ be the TT cores of the tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ with suboptimal rank $\boldsymbol{r}$. During the first step of Algorithm 3, the ϵ-truncated SVD is applied on the matricization $\boldsymbol{X}^{(1)}$ to extract the first core $\tilde{\mathbf{g}}^{(1)}$

of the rounded TT representation. Using the TT representation of $\mathbf{x}$, the matricization can be written as

$$\boldsymbol{X}^{(1)} = \boldsymbol{G}^{(1)}\boldsymbol{G}^{\geqslant 2}, \quad G^{(1)}_{i_1 s_1} = \mathrm{g}^{(1)}_{1 i_1 s_1}, \quad G^{\geqslant 2}_{s_1 \overline{i_1 \dots i_d}} = \sum_{s_2,\dots,s_{d-1}} \mathrm{g}^{(2)}_{s_1 i_1 s_2} \cdots \mathrm{g}^{(d)}_{s_{d-1} i_d 1}. \tag{3.15}$$

If the train $\{\mathbf{g}^{(k)}\}_k$ is right-to-left orthonormalized, the matrix $\boldsymbol{G}^{\geqslant 2}$ also satisfies $\left(\boldsymbol{G}^{\geqslant 2}\boldsymbol{G}^{\geqslant 2}\right)^\top = \boldsymbol{I}$ [OT10]. A truncated SVD of $\boldsymbol{X}^{(1)}$ can therefore be computed by just applying the routine on the first core matricization $\boldsymbol{G}^{(1)}$. The first factor matrix $\tilde{\boldsymbol{U}} \in \mathbb{R}^{n_1 \times \tilde{r}_1}$ of the truncated SVD is reshaped to the new core $\mathbf{g}^{(1)}$ and the rank is updated. The singular values together with the right factor matrix $\tilde{\boldsymbol{V}}$ are multiplied to the second core (the cores indexed by $k \geqslant 3$ remain right orthonormal). This process repeats itself for the following cores (see Algorithm 4 for the detailed pseudocode). The procedure presented here can also be performed from right to left. In this case, the orthonormalization has to be done from left to right. A further improvement of the method is presented in [DBB22] where the tall-skinny QR decomposition [BDG+15] is used to parallelize the rounding procedure.

Algorithm 4 TT rank rounding.

function RoundRank($\{\mathbf{g}^{(k)}\}_k$, $\boldsymbol{r}$, ϵ)
  $\{\mathbf{g}^{(k)}\}_k, \boldsymbol{r} \leftarrow$ OrthoLR($\{\mathbf{g}^{(k)}\}_k, \boldsymbol{r}$).
  for $k \in \{d, \dots, 2\}$ **do**
    $\boldsymbol{C} \leftarrow$ reshape($\mathbf{g}^{(k)}, r_{k-1} \times n_k r_k$).
    $\boldsymbol{B} \leftarrow$ reshape($\mathbf{g}^{(k-1)}, r_{k-2} \times n_{k-1} r_{k-1}$).
    $\boldsymbol{U}, \boldsymbol{\sigma}, \boldsymbol{V} \leftarrow$ SVD($\boldsymbol{C}$).
    $r_k \leftarrow \max_l(\sigma_l^2 + \sigma_{l+1}^2 + \cdots) < \frac{\epsilon^2}{d-1}\left(\sum_s \sigma_s^2\right)$.
    $\tilde{\boldsymbol{U}}, \tilde{\boldsymbol{\sigma}}, \tilde{\boldsymbol{V}} \leftarrow \boldsymbol{U}_{:1:r_k}, \boldsymbol{\sigma}_{1:r_k}, \boldsymbol{V}_{:1:r_k}$.
    $\mathbf{g}^{(k)} \leftarrow$ reshape($\tilde{\boldsymbol{V}}^\top, r_{k-1} \times n_k \times r_k$).
    $\mathbf{g}^{(k-1)} \leftarrow$ reshape($\boldsymbol{B}\tilde{\boldsymbol{U}}\mathbf{diag}(\tilde{\boldsymbol{\sigma}}^2), r_{k-2} \times n_{k-1} \times r_{k-1}$).
return $\{\mathbf{g}^{(k)}\}_k$.

3.2.3. Linear algebra in the Tensor-Train format

An advantage of the TT format is that it can be used for most multilinear algebraic operations in a straightforward manner, the results of which are also expressed in the TT format. The construction of the cores for the basic linear algebra operations is presented in the following. For the case of tensor operators represented in the TT format, the algorithms can be easily applied to TT operators by reshaping the TT cores.

Addition

Given 2 tensors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, with the TT decomposition given by the cores $\mathbf{g}^{(\mathbf{x},k)}$ and $\mathbf{g}^{(\mathbf{y},k)}$ respectively, the TT cores of the pointwise addition $\mathbf{z} = \mathbf{x} + \mathbf{y}$ are

$$\begin{gathered}
\mathrm{g}^{(\mathbf{z},1)}_{1 i_1 :} = \begin{pmatrix} \mathrm{g}^{(\mathbf{x},1)}_{1 i_1 :} & \mathrm{g}^{(\mathbf{y},1)}_{1 i_1 :} \end{pmatrix}, \quad \forall i_1 \in \{1, \dots, n_1\}, \qquad \mathrm{g}^{(\mathbf{z},d)}_{: i_d 1} = \begin{pmatrix} \mathrm{g}^{(\mathbf{x},d)}_{: i_d 1} \\ \mathrm{g}^{(\mathbf{y},1)}_{: i_d 1} \end{pmatrix}, \quad \forall i_d \in \{1, \dots, n_d\} \\
\mathrm{g}^{(\mathbf{z},k)}_{: i_k :} = \begin{pmatrix} \mathrm{g}^{(\mathbf{x},k)}_{: i_k :} & \mathbf{0} \\ \mathbf{0} & \mathrm{g}^{(\mathbf{y},k)}_{: i_k :} \end{pmatrix}, \quad \forall k \in \{2, \dots, d-1\}, i_k \in \{1, \dots, n_k\}.
\end{gathered} \tag{3.16}$$

The TT rank of the resulting is the sum of the ranks of the involved operands $r_k^{\mathbf{z}} = r_k^{\mathbf{x}} + r_k^{\mathbf{y}}$, $k = 1, \ldots, d-1$. Since for some cases the actual TT rank of the result is smaller than the sum, the rank rounding procedure from Algorithm 4 should be applied in order to avoid the overestimation of the rank after successive operations. The asymptotic computational complexity of the addition step as described in (3.16) is $\mathcal{O}(dnr^2)$.

Elementwise multiplication

In the case of the elementwise multiplication (also known as the Hadamard product) of two tensors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, the cores of the product $\mathbf{z} = \mathbf{x} \odot \mathbf{y}$ are given by

$$\mathrm{g}^{(\mathbf{z},k)}_{s_{k-1} i_k s_k} = \mathrm{g}^{(\mathbf{x},k)}_{p_{k-1} i_k p_k} \mathrm{g}^{(\mathbf{y},k)}_{q_{k-1} i_k q_k}, \quad s_{k-1} = \overline{p_{k-1} q_{k-1}},\ s_k = \overline{p_k q_k}, \tag{3.17}$$

where $\{\mathbf{g}^{(\mathbf{x},k)}\}_{k=1,\ldots,d}$ and $\{\mathbf{g}^{(\mathbf{y},k)}\}_{k=1,\ldots,d}$ are the TT cores of the tensors $\mathbf{x}$ and $\mathbf{y}$, respectively. The cores $\mathbf{g}^{(\mathbf{z},k)}$ are $r^{\mathbf{x}}_{k-1} r^{\mathbf{y}}_{k-1} \times n_k \times r^{\mathbf{x}}_k r^{\mathbf{y}}_k$ tensors (the rank of the result is the product of the ranks of the operands), thus making the computational complexity $\mathcal{O}(dnr^4)$.

Tensor operator product

The product between a tensor operator and a tensor (or between 2 tensor operators) can also be computed directly in the TT format by core operations. Let $\mathbf{A} \in \mathbb{R}^{m_1 \times \cdots \times m_d \times n_1 \times \cdots \times n_d}$ be a tensor operator and $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a tensor with the TT decompositions given by the cores $\{\mathbf{g}^{(\mathbf{A},k)}\}_k$ and $\{\mathbf{g}^{(\mathbf{x},k)}\}_k$ respectively. The TT representation of the tensor $\mathbf{z} = \mathbf{A}\mathbf{x}$ is

$$\mathrm{g}^{(\mathbf{z},k)}_{s_{k-1} i_k s_k} = \sum_{j_k} \mathrm{g}^{(\mathbf{A},k)}_{p_{k-1} i_k j_k p_k} \mathrm{g}^{(\mathbf{x},k)}_{q_{k-1} j_k q_k}, \quad s_{k-1} = \overline{p_{k-1} q_{k-1}}, s_k = \overline{p_k q_k}. \tag{3.18}$$

Similar to the elementwise product, the rank of the result is the product of the ranks of the inputs. A more efficient method to approximate the product has been proposed in [Ose11a] which reduces the computational complexity to $\mathcal{O}(dnr^4)$, under the assumption that the result has a comparable rank with the rank of the input tensor. This method is based on the density matrix renormalization group (DMRG) method (first proposed in [Whi93] and is also used to solve multilinear systems in TT format in [OD12]). Contrary to elementwise addition, where the subtraction is also straightforward to perform by manipulating the TT cores, there is no known algorithm involving direct core manipulation in order to compute the elementwise division in the TT format. In order to accomplish this task, a minimization of the residual w.r.t. the TT cores has to be performed. The task is similar to solving a multilinear system with a diagonal tensor operator and right-hand side equal to 1 (more details about multilinear solvers are given in Section 3.2.4).

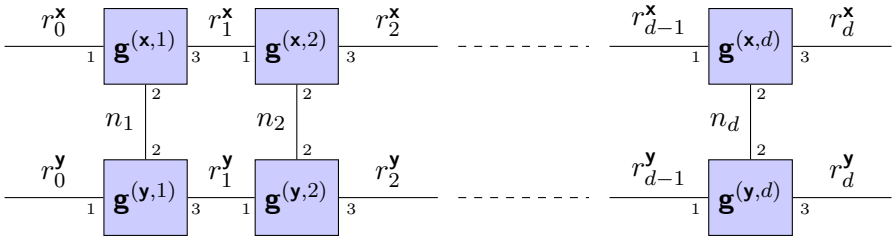
Scalar product and Frobenius norm

One further linear algebra operation that can be performed using basic core manipulation is the scalar product defined in (2.24). Given two tensors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ with the TT cores $\{\mathbf{g}^{(\mathbf{x},k)}\}_k$ and $\{\mathbf{g}^{(\mathbf{y},k)}\}_k$

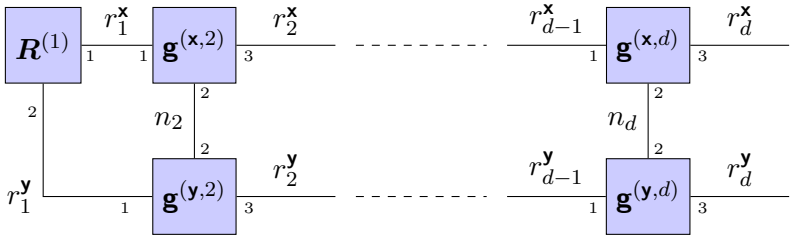
respectively. The scalar product can be obtained using successive contraction of the TT cores:

$$\begin{aligned}
\mathbf{x}\cdot\mathbf{y} &= \sum_{i_1,\dots,i_d}\left(\sum_{q_1,\dots,q_{d-1}} \mathrm{g}^{(\mathbf{x},1)}_{1 i_1 q_1}\cdots \mathrm{g}^{(\mathbf{x},d)}_{q_{d-1} i_d 1}\right)\left(\sum_{s_1,\dots,s_{d-1}} \mathrm{g}^{(\mathbf{y},1)}_{1 i_1 s_1}\cdots \mathrm{g}^{(\mathbf{y},d)}_{s_{d-1} i_d 1}\right) = \\
&\sum_{i_2,\dots,i_d}\sum_{\substack{q_1,\dots,q_{d-1}\\ s_1,\dots,s_{d-1}}} \underbrace{\left(\sum_{i_1} \mathrm{g}^{(\mathbf{x},1)}_{1 i_1 q_1}\mathrm{g}^{(\mathbf{y},1)}_{1 i_1 s_1}\right)}_{=:\boldsymbol{R}^{(1)}} \mathrm{g}^{(\mathbf{x},2)}_{q_1 i_2 q_2}\cdots \mathrm{g}^{(\mathbf{x},d)}_{q_{d-1} i_d 1}\mathrm{g}^{(\mathbf{y},2)}_{s_1 i_2 s_2}\cdots \mathrm{g}^{(\mathbf{y},d)}_{s_{d-1} i_d 1} = \\
&\sum_{i_3,\dots,i_d}\sum_{\substack{q_2,\dots,q_{d-1}\\ s_2,\dots,s_{d-1}}} \underbrace{\left(\sum_{i_2,s_1,q_1} R^{(1)}_{q_1 s_1}\mathrm{g}^{(\mathbf{x},2)}_{q_1 i_2 q_2}\mathrm{g}^{(\mathbf{y},2)}_{s_1 i_2 s_2}\right)}_{=:\boldsymbol{R}^{(2)}} \mathrm{g}^{(\mathbf{x},3)}_{q_2 i_3 q_3}\cdots \mathrm{g}^{(\mathbf{x},d)}_{q_{d-1} i_d 1}\mathrm{g}^{(\mathbf{y},3)}_{s_2 i_3 s_3}\cdots \mathrm{g}^{(\mathbf{y},d)}_{s_{d-1} i_d 1} = \cdots = \boldsymbol{R}^{(d)}\,.
\end{aligned} \tag{3.19}$$

The last remaining matrix $\boldsymbol{R}^{(d)}$ contains the resulting scalar product. In Figure 3.4, the scalar product of 2 tensors in the TT format as well as the first step of the algorithm are presented using the tensor network notation. Algorithm 5 presents the procedure in detail. Using the dot product, one can compute the Frobenius norm $||\mathbf{x}||_F^2 = \mathbf{x}\cdot\mathbf{x}$. An alternative way to compute the Frobenius norm is to make use of the core orthonormality property [Ose11b]. If the train is left-orthonormalized, the first $d-1$ contractions of the cores yield the identity matrix. In this case the Frobenius norm is the Frobenius norm of the last core from the orthonormal train.



(a) Graphical representation of $\mathbf{x}\cdot\mathbf{y}$ if both tensors are in the TT format.



(b) In the first step, the first 2 cores $\mathbf{g}^{(\mathbf{x},1)}$ and $\mathbf{g}^{(\mathbf{y},1)}$ are contracted along i_1.

Figure 3.4.: Graphical representation of performing the dot product in the TT format.

Algorithm 5 Scalar product in the TT format.

function ScalarProduct($\{\mathbf{g}^{(\mathbf{x},k)}\}_k, \{\mathbf{g}^{(\mathbf{y},k)}\}_k$ TT cores of $\mathbf{x}$ and $\mathbf{y}$)
  $R_{11}^{(0)} \leftarrow 1$, $\boldsymbol{R}^{(1)}$ is 1×1 matrix.
  for $k \in \{1, \dots, d\}$ **do**
    $R_{pq}^{(k)} \leftarrow \sum_{i,j,k} R_{ij}^{(k-1)} \mathrm{g}_{ikp}^{(\mathbf{x},k)} \mathrm{g}_{jkq}^{(\mathbf{x},k)}$, $\forall i \in \{1, \dots, r_k^{\mathbf{x}}\}, j \in \{1, \dots, r_k^{\mathbf{y}}\}$.
  $\mathbf{x} \cdot \mathbf{y} \leftarrow R_{11}^{(d)}$.
return $\mathbf{x} \cdot \mathbf{y}$.

n-mode product and index contraction

Of interest when performing inference tasks is the ability to sum over certain modes. If a tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ has the TT representation given by the cores $\{\mathbf{g}^{(k)}\}_g$, performing the n-mode product $\mathbf{z} = \mathbf{x} \times_q \boldsymbol{A}$ affects the q-th core only

$$\mathrm{g}_{s_{q-1} j s_q}^{(\mathbf{z},q)} = \sum_{i_q} \mathrm{g}_{s_{q-1} i_q s_q}^{(q)} A_{j i_q}, \quad \mathbf{g}^{(\mathbf{z},k)} = \mathbf{g}^{(k)}, k \neq q, \tag{3.20}$$

where $\mathbf{g}^{(\mathbf{z},q)}$ is the q-th core of the product. The rank remains in this case unchanged. If the n-mode product is performed with a vector $\boldsymbol{a}$, the resulting TT core becomes 2-dimensional and it has to be merged with one of its neighbors. If $r_{q-1} < r_q$ and $q < d$, the core is merged with its right neighbor $q+1$

$$\mathrm{g}_{s_{q-1} i_{q+1} s_{q+1}}^{(\mathbf{z},q+1)} = \sum_{i_q, s_q} \mathrm{g}_{s_{q-1} i_q s_q}^{(q)} a_{i_q} \mathrm{g}_{s_q i_{q+1} s_{q+1}}^{(q+1)}, \tag{3.21}$$

and if $r_{q-1} > r_q$ and $q > 1$, the core is merged with the left neighbor $q-1$

$$\mathrm{g}_{s_{q-2} i_{q-1} s_q}^{(\mathbf{z},q-1)} = \sum_{i_q, s_{q-1}} \mathrm{g}_{s_{q-2} i_{q-1} s_{q-1}}^{(q-1)} a_{i_q} \mathrm{g}_{s_{q-1} i_q s_q}^{(q)}, \tag{3.22}$$

allowing to keep the smallest ranks in the resulting train.

Reshaping and quantization

Of interest in this work is the reshaping of tensors represented in the TT format. In the case of working with full tensors, the reshaping of a tensor $\mathbf{x}$ with the mode sizes $\boldsymbol{n} = (n_1, \dots, n_d)$ to a tensor $\mathbf{x}'$ with mode sizes $\boldsymbol{n}' = (n'_1, \dots, n'_{d'})$, $n_1 n_2 \cdots n_d = n'_1 n'_2 \cdots n'_d$ implies creating a different view of the entries in the memory or it creates a copy with permuted entries, depending on the implementation. If the tensor is already represented in the TT format, the naive approach for reshaping would be computing the full tensor, reshaping it and converting it back to the TT format. This is however not efficient and therefore a different approach is presented (the pseudocode is given in Algorithm 6). The main idea is to iterate over the target mode sizes $n'_k, k = 1, \dots, d'$ and to group consecutive cores of the original tensor until the product of their modes (stored in the variable m) is divisible by the target n'_k. In the pseudocode, the index i iterates through the cores of the original tensor and the grouped cores are expanded into a tensor $\mathbf{c}$. The truncated SVD is then used to extract the TT core $\mathbf{g}'^{(k)}$ of the reshaped train from the tensor $\mathbf{c}$. The tensor $\mathbf{c}$ is updated using the truncated factor matrix $\mathbf{diag}(\tilde{\boldsymbol{\sigma}})\tilde{\boldsymbol{V}}^\top$. The cores of the original tensor are further accumulated in the newly obtained tensor $\mathbf{c}$ until the next TT core of the reshaped tensor can be split.

The quantized tensor-train (QTT) is one particular case of reshaping a tensor in the TT format. Quantization refers to increasing the number of dimensions by reshaping tensors with mode sizes that are powers of 2 ($\log_2 n_k \in \mathbb{N}$) into $(\sum_k \log_2 n_k)$-dimensional ones with mode sizes 2. This procedure has been proven to be effective for reducing the storage requirements and the computation time for solving the CME in the TT format in different fields, such as computational chemistry [KO10, KO11, KKNS14, DS20, IWL+21], partial differential equations [DKO12, ILDG22] and machine learning [LWY+21, SCS+22]. If the ranks of the quantized tensor-train (QTT) decomposition remain bounded, the storage complexity is $\mathcal{O}(d \log_2 n)$ [Kho11]. The QTT format will be later used in Chapters 5 and 6 in order to accelerate the AMEn solver and the construction of the tensor operators.

Algorithm 6 Reshaping a tensor in the TT format.

function RESHAPETT($\{\mathbf{g}^{(k)}\}_k$ TT cores of $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, $\boldsymbol{n}' = (n'_1, \ldots, n'_{d'})$, ϵ)
  Ensure $\prod_k n_k = \prod_k n'_k$.
  $r'_0 \leftarrow 1$.
  $k \leftarrow 1$.
  $\mathbf{c} \leftarrow \mathbf{g}^{(1)}$.
  $i \leftarrow 2$.
  while $k \leqslant d'$ **do**
    if $m/n'_k \in \mathbb{N}$ **then**
      $\boldsymbol{C} \leftarrow \text{reshape}(\mathbf{c}, r'_{k-1} n'_k \times (m/n'_k) r_i)$.
      $\boldsymbol{U}, \boldsymbol{\sigma}, \boldsymbol{V} \leftarrow \text{SVD}(\boldsymbol{C})$.
      $r'_k \leftarrow \max_l (\sigma_l^2 + \sigma_{l+1}^2 + \cdots) < \frac{\epsilon^2}{d'-1} \left(\sum_s \sigma_s^2 \right)$.
      $\tilde{\boldsymbol{U}}, \tilde{\boldsymbol{\sigma}}, \tilde{\boldsymbol{V}} \leftarrow \boldsymbol{U}_{:(1:r'_k)}, \boldsymbol{\sigma}_{(1:r'_k)}, \boldsymbol{V}_{:(1:r'_k)}$.
      $\mathbf{g}'^{(k)} \leftarrow \text{reshape}(\tilde{\boldsymbol{U}}, r'_{k-1} \times n'_k \times r'_k)$.
      $\mathbf{c} \leftarrow \text{reshape}(\mathbf{diag}(\tilde{\boldsymbol{\sigma}}) \tilde{\boldsymbol{V}}^\top, r'_k \times (m/n'_k) \times r_i)$.
      $m \leftarrow m/n'_k$.
    else
      $\mathsf{c}_{uvwz} \leftarrow \sum_s \mathsf{c}_{uvs} \mathsf{g}^{(i)}_{swz}$.
      $m \leftarrow m n_i$.
      $\mathbf{c} \leftarrow \text{reshape}(\mathbf{c}, r'_{k-1} \times m \times r_i)$.
      $i \leftarrow i + 1$.
return $\{\mathbf{g}'^{(k)}\}$.

3.2.4. Multilinear systems in the TT format

One very important step when dealing with high dimensional differential equations is solving multilinear systems. One approach is to adapt the classical iterative methods for solving linear systems to work with tensors in the TT format [HKT08]. In [Dol13], the generalized minimal residual method (GMRES) algorithm has been extended to the TT format. A significant drawback of adapting Krylov space methods to the TT format is that large tensors have to be handled during the iterations. Improvements compared to the TT-GMRES method introduced in [Dol13] have been presented in [ADL21].

An alternative is to perform optimization of quadratic functions (residual, energy) in the TT format. For

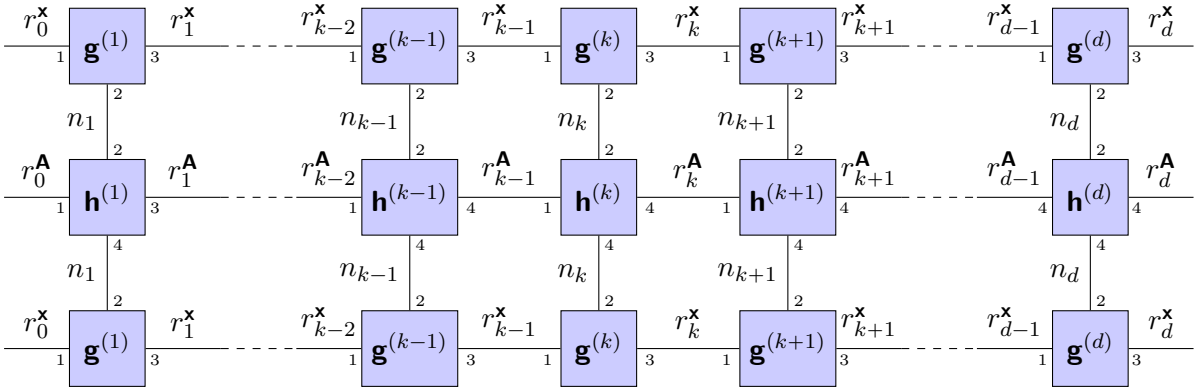


Figure 3.5.: Graphical representation of the bilinear form $\mathbf{x} \cdot (\mathbf{A}\mathbf{x})$ for both $\mathbf{A}$ and $\mathbf{x}$ represented in the TT format with the cores $\mathbf{h}^{(k)}$ and $\mathbf{g}^{(k)}$ respectively.

solving multilinear systems, the squared norm of the residual is minimized

$$\min_{\mathbf{x} \in \mathcal{X}} \ ||\mathbf{A}\mathbf{x} - \mathbf{b}||_F^2, \tag{3.23}$$

where $\mathcal{X}$ is the set of tensors with low TT rank. However, if the tensor operator is symmetric and positive definite, solving the system is equivalent to finding the minimizer of the following optimization problem

$$\min_{\mathbf{x} \in \mathcal{X}} \ \mathbf{x} \cdot (\mathbf{A}\mathbf{x}) - 2\mathbf{x} \cdot \mathbf{b}, \tag{3.24}$$

where the loss function is denoted as $J_{\mathbf{A},\mathbf{b}}$ in the following. The optimization problem is nonlinear and nonconvex w.r.t. all TT cores of the decomposition and the rank of the solution is not known in most of the cases, leading to a mixed-integer optimization problem. In the case of CPD, several attempts have been made to solve the minimization problem by applying Newton's method [KT10]. In the case of the TT format, Riemannian optimization has also been proposed to solve general minimization problems using automatic differentiation [HRS10a, NRO22]. This however only works on manifolds of tensors with fixed TT ranks.

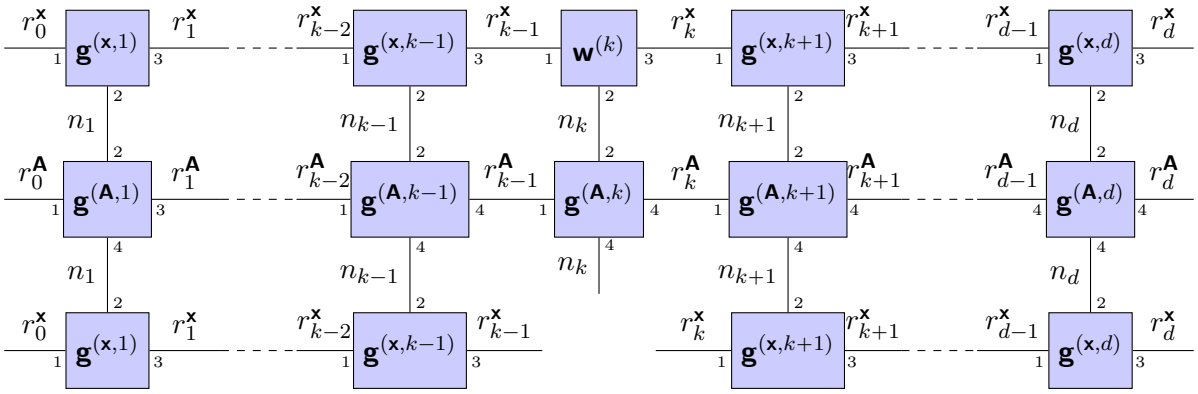
Alternating least squares (ALS)

One method to reduce the complexity of the optimization problem is to take advantage of the fact that the map $\mathcal{TT}$ is linear with respect to an individual core, provided that the remaining TT cores are fixed. Fixing all cores but one transforms the highly nonlinear problem (3.23) into a quadratic optimization problem. In the literature, this method is found under the name of ALS [HRS12]. The idea is to iteratively optimize w.r.t. only one core at the time in order to obtain a new core that decreases the loss function $J_{\mathbf{A},\mathbf{b}}$. If only the k-th TT core is considered as variable and the rest are fixed, the following linear operator is defined

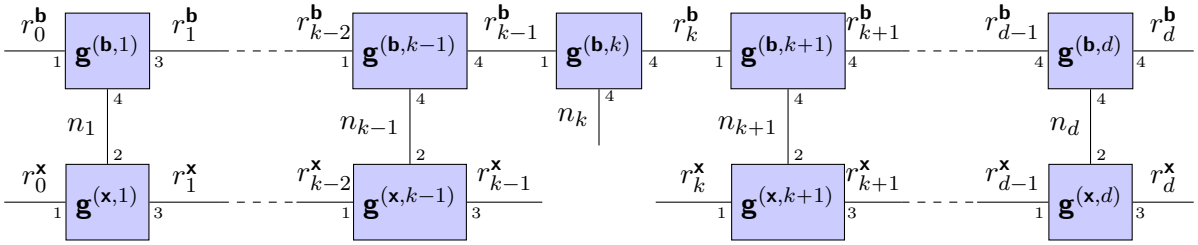
$$\mathbf{Q}^{(k)}(\mathbf{w}) = \mathcal{TT}(\mathbf{g}^{(1)}, \dots, \mathbf{g}^{(k-1)}, \mathbf{w}, \mathbf{g}^{(k+1)}, \dots, \mathbf{g}^{(d)}) \in \mathbb{R}^{n_1 \times \cdots \times n_d}, \quad \mathbf{w} \in \mathbb{R}^{r_{k-1}^{(\mathbf{x})} \times n_k \times r_k^{(\mathbf{x})}}, \tag{3.25}$$

which fixes all the cores but the k-th. The composition with the loss function $J_{\mathbf{A},\mathbf{b}}$ is in this case

$$(J_{\mathbf{A},\mathbf{b}} \circ \mathbf{Q}^{(k)})(\mathbf{w}) = \mathbf{Q}^{(k)}(\mathbf{w}) \cdot \left(\mathbf{A}\mathbf{Q}^{(k)}(\mathbf{w})\right) - 2\mathbf{b} \cdot \mathbf{Q}^{(k)}(\mathbf{w}). \tag{3.26}$$



(a) The tensor operator $\mathbf{L}^{(k)}\mathbf{w}^{(k)}$ of the local subsystem.



(b) Graphical representation of $\mathbf{f}^{(k)}$.

Figure 3.6.: Tensor network representation of the local system $\mathbf{L}^{(k)}\mathbf{w}^{(k)} = \mathbf{f}^{(k)}$.

As represented Figure 3.5, the fixed TT cores as well as the cores of $\mathbf{A}$ can be contracted together to obtain a single tensor operator $\mathbf{L}^{(k)} \in \mathbb{R}^{(r_{k-1}\times n_k \times r_k)\times(r_{k-1}\times n_k \times r_k)}$. The same can be performed for the second dot product from (3.26). The function from (3.26) takes the following form

$$\mathbf{Q}^{(k)}(\boldsymbol{w}) = \mathbf{w}\cdot(\mathbf{L}^{(k)}\mathbf{w}) - 2\mathbf{f}^{(k)}\cdot\mathbf{w}, \tag{3.27}$$

where $\mathbf{L}$ and $\mathbf{f} \in \mathbb{R}^{r^{(\mathbf{x})}_{k-1}\times n_k \times r^{(\mathbf{x})}_k}$ are computed by contracting the edges in Figure 3.6a and 3.6b, respectively. Since a symmetric positive definite $\mathbf{A}$ is considered, the operator $\mathbf{L}^{(k)}$ is also symmetric positive definite and the minimizer of the functional w.r.t. k-th core can be computed as the solution of the system

$$\nabla(J_{\mathbf{A},\mathbf{b}}\circ\mathbf{Q}^{(k)})(\mathbf{w}) = \mathbf{L}^{(k)}\mathbf{w} - \mathbf{f} = \mathbf{0}. \tag{3.28}$$

The solution of the system is the new core of the ALS iteration and the process is repeated for the next core. An efficient way of computing $\mathbf{L}^{(k)}$ and $\mathbf{f}^{(k)}$ at every iteration is to recursively contract the cores $1,\dots,k-1$ into a left interface tensor and the cores $\{k+1,\dots,d\}$ into a right interface tensor. As a result, the system can be assembled as (see the tensor diagrams from Figures 3.7a and 3.7b).

$$\mathsf{L}^{(k)}_{p_{k-1}i_kp_k,s_{k-1}j_ks_k} = \sum_{q_{k-1},q_k} \Phi^{(\mathbf{A},k)}_{p_{k-1}q_{k-1}s_{k-1}} \Psi^{(\mathbf{A},k)}_{p_kq_ks_k} \mathsf{g}^{(\mathbf{A},k)}_{q_{k-1}i_kj_kq_k}, \tag{3.29}$$

$$\mathsf{f}^{(k)}_{q_{k-1}i_kq_k} = \sum_{p_{k-1},p_k} \Phi^{(\mathbf{b},k-1)}_{p_{k-1}q_{k-1}} \Psi^{(\mathbf{b},k+1)}_{p_kq_k} \mathsf{g}^{(\mathbf{b},k)}_{p_{k-1}i_kp_k}, \tag{3.30}$$

where the following recursive relations can be used to update the left and right interfaces $\mathbf{\Phi}$ and $\mathbf{\Psi}$

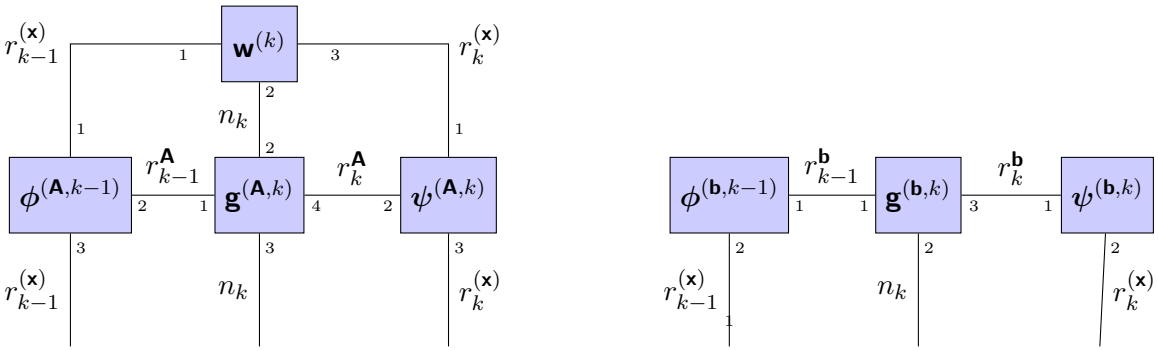
$$\Phi^{(\mathbf{A},k)}_{s'p'q'} = \sum_{s,p,q,m,n} \Phi^{(\mathbf{A},k-1)}_{spq} \mathrm{g}^{(\mathbf{A},k)}_{pmnp'} \mathrm{g}^{(\mathbf{x},k)}_{sms'} \mathrm{g}^{(\mathbf{x},k)}_{qnq'}, \qquad \mathbf{\Phi}^{(\mathbf{A},k)} \in \mathbb{R}^{r^{(\mathbf{x})}_k \times n_k \times r^{(\mathbf{x})}_k}, \quad \mathbf{\Phi}^{(\mathbf{A},0)} = 1, \tag{3.31}$$

$$\Psi^{(\mathbf{A},k)}_{spq} = \sum_{s',p',q',m,n} \Psi^{(\mathbf{A},k+1)}_{spq} \mathrm{g}^{(\mathbf{A},k)}_{pmnp'} \mathrm{g}^{(\mathbf{x},k)}_{sms'} \mathrm{g}^{(\mathbf{x},k)}_{qnq'}, \qquad \mathbf{\Psi}^{(\mathbf{A},k)} \in \mathbb{R}^{r^{(\mathbf{x})}_k \times n_k \times r^{(\mathbf{x})}_k}, \quad \mathbf{\Psi}^{(\mathbf{A},d)} = 1, \tag{3.32}$$

$$\Phi^{(\mathbf{b},k)}_{p'q'} = \sum_{m,p,q} \Phi^{(\mathbf{b},k-1)}_{pq} \mathrm{g}^{(\mathbf{b},k)}_{pmp'} \mathbf{g}^{(\mathbf{x},k)}_{qmq'}, \qquad \mathbf{\Phi}^{(\mathbf{b},k)} \in \mathbb{R}^{r^{(\mathbf{b})}_k \times r^{(\mathbf{x})}_k}, \quad \mathbf{\Phi}^{(\mathbf{b},0)} = 1, \tag{3.33}$$

$$\Psi^{(\mathbf{b},k)}_{pq} = \sum_{m,p',q'} \Phi^{(\mathbf{b},k+1)}_{p'q'} \mathrm{g}^{(\mathbf{b},k)}_{pmp'} \mathbf{g}^{(\mathbf{x},k)}_{qmq'}, \qquad \mathbf{\Psi}^{(\mathbf{b},k)} \in \mathbb{R}^{r^{(\mathbf{b})}_k \times r^{(\mathbf{x})}_k}, \quad \mathbf{\Phi}^{(\mathbf{b},d)} = 1. \tag{3.34}$$

The pseudocode of the ALS method is presented in Algorithm 7. The local system $\mathbf{L}^{(k)}\mathbf{w}^{(k)} = \mathbf{f}^{(k)}$ can be solved either using a direct solver, if the size of the tensor is smaller than a predefined threshold, or using iterative solvers. The latter avoids building the operator $\mathbf{L}$ by directly computing the product $\mathbf{Lw}$. Despite being simple to implement, the ALS method has two main drawbacks: the slow convergence [RU13] and the fact that the TT-ranks have to be known in advance.



(a) Tensor diagram representation of the product $\mathbf{L}^{(k)}\mathbf{w}^{(k)}$.

(b) Tensor diagram representation of the right-hand-side $\mathbf{f}^{(k)}$ of the local system.

Figure 3.7.: Assembly of the local system. This is obtained by contracting the left part of the tensor networks in Figure 3.6

DMRG

The lack of the rank adaptation of the ALS method can be overcome using the DMRG method. First used to minimize the Rayleigh quotient [Whi93], the DMRG was first used in [Jec02] in the context of linear systems. Instead of minimizing w.r.t. individual cores, the DMRG joins together 2 consecutive cores $\mathbf{g}^{(k)}$ and $\mathbf{g}^{(k+1)}$ in order to obtain a so called supercore $\mathbf{w}^{(k,k+1)} \in \mathbb{R}^{r^{(\mathbf{x})}_{k-1} \times n_k \times n_{k+1} \times r^{(\mathbf{x})}_{k+1}}$ by contracting across the common rank. The minimization problem at the step k is in this case

$$\min_{\mathbf{w}^{(k,k+1)}} J_{\mathbf{A},\mathbf{b}}(\mathcal{TT}(\mathbf{g}^{(1)}, \ldots, \mathbf{g}^{(k-1)}, \mathbf{w}^{(k,k+1)}, \mathbf{g}^{(k+2)}, \ldots, \mathbf{g}^{(d)})). \tag{3.35}$$

Similarly to the ALS, the optimal supercore fulfills the local system $\mathbf{L}^{(k,k+1)}\mathbf{w} = \mathbf{f}^{(k)}$. The tensor operator $\mathbf{L}^{(k,k+1)} \in \mathbb{R}^{(r^{\mathbf{x}}_{k-1} \times n_k \times n_{k+1} \times r^{\mathbf{x}}_{k+1}) \times (r^{(\mathbf{x})}_{k-1} \times n_k \times n_{k+1} \times r^{\mathbf{x}}_{k+1})}$ and the right-hand side $\mathbf{f} \in \mathbb{R}^{r^{(\mathbf{x})}_{k-1} \times n_k \times n_{k+1} \times r^{(\mathbf{x})}_{k+1}}$ can

Algorithm 7 ALS for solving linear systems in the TT format.

function ALS($\{\mathbf{g}^{(\mathbf{A},k)}\}_k, \{\mathbf{g}^{(\mathbf{b},k)}\}_k$ TT cores of $\mathbf{A}$ and $\mathbf{b}$)
  Choose random initial TT cores $\{\mathbf{g}^{(\mathbf{x},k)}\}_k$.
  for $i \in \{1, 2, \dots\}$ **do**
    $\boldsymbol{\Psi}^{(\mathbf{A},d)} \leftarrow 1$.
    $\boldsymbol{\Psi}^{(\mathbf{f},d)} \leftarrow 1$.
    for $k \in \{d, \dots, 1\}$ **do**
      Update $\boldsymbol{\Psi}^{(\mathbf{A},k-1)}$ and $\boldsymbol{\Psi}^{(\mathbf{f},k-1)}$ using (3.32) and (3.34).
      Orthogonalize the core $\mathbf{g}^{(\mathbf{x},k)}$ (right to left).
    for $k \in \{1, \dots, d\}$ **do**
      Solve the system $\mathbf{L}^{(k)}\mathbf{g}^{(\mathbf{x},k)} = \mathbf{f}^{(k)}$ (direct solver of iterative solver).
      Update $\boldsymbol{\Phi}^{(\mathbf{A},k)}$ and $\boldsymbol{\Phi}^{(\mathbf{f},k)}$ using (3.31) and (3.33).
  return $\{\mathbf{g}^{(\mathbf{x},k)}\}_k$.

be obtained using the tensors from (3.31), (3.32), (3.33) and (3.34) as shown in Figure 3.8. The newly obtained supercore $\mathbf{w}^{(k,k+1)}$ can be split into two TT cores using the SVD. One possible way to truncate the SVD and reveal the rank r_k is to use the residual of the system [OD12]. The supercore $\mathbf{w}^{(k,k+1)}$ is reshaped to an $(r_{k-1}n_k) \times (n_{k+1}r_{k+1})$ matrix $\boldsymbol{W}^{(k,k+1)}$ and the SVD is used to obtain $\boldsymbol{W}^{(k,k+1)} = \boldsymbol{U}\mathbf{diag}(\boldsymbol{\sigma})\boldsymbol{V}^\top$. Several strategies for truncating the decomposition have been proposed [OD12]. One of the most efficient truncation strategies is based on the residual of the local systems. This means finding the best residual $||\boldsymbol{L}\boldsymbol{w} - \boldsymbol{f}|| \leqslant \epsilon||\boldsymbol{L}\tilde{\boldsymbol{w}} - \boldsymbol{f}||$. An additional trick to overcome local minima is to extend the cores using a random component [OD12]. Let $\mathbf{w}^{(k)}$ and $\mathbf{w}^{(k+1)}$ be the TT cores obtained after splitting the supercore $\mathbf{w}^{(k,k+1)}$. The TT cores after the enrichment with a random component are

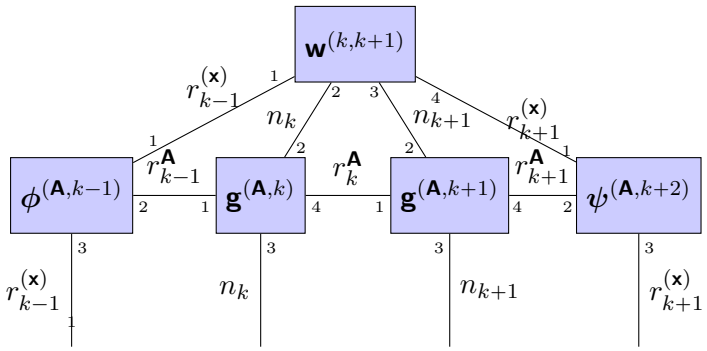
$$\mathsf{g}^{(\mathbf{x},k)}_{:i_k:} = \begin{pmatrix} \mathsf{w}^{(k)}_{:i_k:} & \mathsf{s}^{(k)}_{:i_k:} \end{pmatrix}, \quad \mathsf{g}^{(\mathbf{x},k+1)}_{:i_{k+1}:} = \begin{pmatrix} \mathsf{w}^{(k)}_{:i_k:} \\ \mathbf{0} \end{pmatrix}, \tag{3.36}$$

where $\mathbf{s}^{(k)} \in \mathbb{R}^{r^{\mathbf{x}}_{k-1} \times n_k \times r'}$ is a random tensor orthogonalized to $\mathbf{w}^{(k)}$ and $r' \in \mathbb{N}$ is the "kick rank" [OD12]. Despite the basis enrichment, the DMRG method remains stuck in local minima in some practical applications [DS14]. One further disadvantage of the DMRG scheme is the size of the local systems. Compared to the classical ALS, where the local system is solved for $\mathcal{O}(nr^2)$ entries, in the case of DMRG the unknown supercore has the storage complexity $\mathcal{O}(n^2r^2)$.

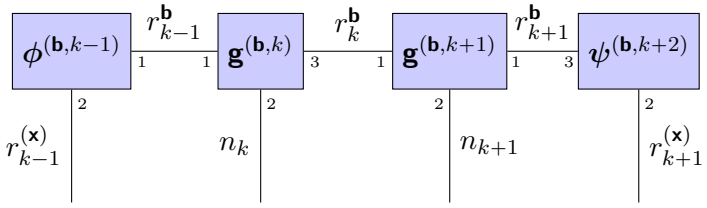
AMEn

The AMEn method aims to combine the ALS scheme with the classical Krylov-subspace based methods. The main idea of the AMEn method is to use the TT representation of the residual of the system $\mathbf{b} - \mathbf{A}\mathbf{x}$ for the basis enrichment (the residual coincides with $-\nabla J_{\mathbf{A},\mathbf{b}}(\mathbf{x})$ in this case). The following modifications to the ALS iteration [DS14] are made for the AMEn method:

1. The local system $\mathbf{L}^{(k)}\mathbf{w}^{(k)} = \mathbf{f}^{(k)}$ is solved.
2. A truncated SVD is performed on the matricization $\mathcal{M}^{\leqslant 2}\mathbf{w}^{(k)}$.
3. If $k < d$, the TT core $\mathbf{z}^{(k)}$ of the residual $\mathbf{b} - \mathbf{A}\tilde{\mathbf{w}}^{(k)}$ is computed ($\tilde{\mathbf{w}}^{(k)}$ is the core reconstructed after the SVD truncation)
4. $\tilde{\mathbf{w}}^{(k)}$ is enriched using $\mathbf{z}^{(k)}$ and the orthogonality is enforced on the newly obtained core $\mathbf{g}^{(\mathbf{x},k)}$.



(a) Tensor diagram representation of the product $\mathbf{L}^{(k,k+1)}\mathbf{w}^{(k,k+1)}$



(b) Tensor diagram representation of $\mathbf{f}^{(k,k+1)}$

Figure 3.8.: Local system computation for the DMRG method.

An in depth analysis of the AMEn algorithm is presented in [DS14]. Due to the reduced computational complexity when compared to the DMRG and the ability to escape local minima, AMEn is the method of choice in this work when dealing with multilinear systems.

3.2.5. Interpolation in the TT format

In Section 3.2.1, the decomposition method from full format has been introduced. However, this assumes that all the entries of the tensor are known. This requirement quickly becomes computationally unattractive even for small tensors. One question that immediately comes into mind is whether one can decompose a tensor with a low rank structure without explicitly storing all its entries in the memory. For $\boldsymbol{n} \in \mathbb{N}^d$ and $I_k = \{1, \dots, n_k\}, k = 1, \dots, d$ and a given a function $f : \times_{k=1}^{d} I_k \to \mathbb{R}$, the goal is to find a TT representation of the tensor $\mathsf{y}_{\boldsymbol{i}} = f(i_1, \dots, i_d)$ without explicitly computing the image of f. In the case of matrices, this problem has been addressed in [Tyr00, Beb00, GT01], where a low rank approximation of a matrix is computed by taking into account only selected rows and columns (crosses). The generalization to the TT format has been introduced in [Ose10] and later extended in [SO11]. The latter introduces an adaptive method based on DMRG minimization and the idea will be briefly presented in the following.

The problem of finding the TT approximation can be formally written as a minimization problem

$$\min_{\text{TT cores } \{\mathbf{g}^{(k)}\}_k} ||\mathbf{y} - \mathcal{TT}(\mathbf{g}^{(1)}, \dots, \mathbf{g}^{(d)})||_F^2. \tag{3.37}$$

If all the cores but the k-th are fixed, the following holds:

$$M^{<k}_{i_1 \dots i_{k-1}} . \mathbf{g}^{(k)}_{:i_k:} M^{>k}_{:i_{k+1} \dots i_d} \approx f(i_1, \dots, i_{k-1}, i_k, i_{k+1}, \dots, i_d), \tag{3.38}$$

where $\boldsymbol{M}^{<k} \in \mathbb{R}^{n_1 \times \cdots \times n_{k-1} \times r_{k-1}}$ and $\boldsymbol{M}^{>k} \in \mathbb{R}^{r_{k+1} \times n_{k+1} \times \cdots \times n_d}$ are obtained by contracting together the cores $\{\mathbf{g}^{(1)}, \ldots, \mathbf{g}^{(k-1)}\}$ and the core $\{\mathbf{g}^{(k+1)}, \ldots, \mathbf{g}^{(d)}\}$, respectively. To obtain the core $\mathbf{g}^{(k)}$, the pseudoinverse of the interface matrices $\boldsymbol{M}^{<k}$ and $\boldsymbol{M}^{>k}$ have to be computed. If the first $k-1$ cores are kept left-orthonormal and the cores $\{\mathbf{g}^{(k+1)}, \ldots, \mathbf{g}^{(d)}\}$ are kept right-orthonormal, then the matrices $\boldsymbol{M}^{<k}$ and $\boldsymbol{M}^{>k}$ have orthonormal columns and rows, respectively. The core $\mathbf{g}^{(k)}$ can be therefore recovered by multiplying the right-hand side with $(\boldsymbol{M}^{<k})^\top$ to the left and $(\boldsymbol{M}^{>k})^\top$ to the right. In [SO11], the cross approximation for matrices is used for evaluating the matrices $\boldsymbol{Y}_{:i_k:}, Y_{\overline{i_1,\ldots,i_{k-1}} i_k \overline{i_{k+1},\ldots,i_d}}, i_k = 1, \ldots, n_k$. As a result, the system in (3.38) takes the form

$$\sum_{s_{k-1}, s_k} M^{<k}_{\mathcal{I}(s_{k-1}) s_{k-1}} \mathrm{g}^{(k)}_{s_{k-1} i_k s_k} M^{>k}_{s_k \mathcal{J}(s_k)} = f(I_{1,s_{k-1}}, \ldots, I_{k-1,s_{k-1}}, i_k, J_{k+1,s_k}, \ldots, J_{d,s_k}), \quad \forall i_k = 1, \ldots, n_k, \tag{3.39}$$

where $\boldsymbol{I}$ and $\boldsymbol{J}$ are the sets of r_{k-1} and r_k indices where the function is evaluated (crosses of the matrices $\boldsymbol{Y}_{:i_k:}$) and $\mathcal{I}(s_{k-1}) = \overline{I_{1,s_{k-1}}, \ldots, I_{k-1,s_{k-1}}}$, $\mathcal{J}(s_k) = \overline{J_{k+1,s_k}, \ldots, J_{d,s_k}}$. The submatrices of $\boldsymbol{M}^{<k}$ and $\boldsymbol{M}^{>k}$ should be chosen such that they are well-conditioned. One way of constructing index sets is to use the maxvol method presented in [GOS+10, CMI09]. After the system (3.39) is solved, the index sets are updated and the system for the next cores is assembled. The procedure presented above can be used in ALS schemes (or AMEn) or can be extended to DMRG schemes [SO11, OT10]. In [QLG+22], it is shown that the cross approximation provides a stable TT decomposition within a given error.

4. Bayesian Inverse Problems

Solving an inverse problem concerns the determination of the underlying state or parameters of a system by taking into account possibly noisy observations of an output state. A model that connects the output quantity to the underlying state is assumed to be known. Determining the output state of the system given the hidden state (or parameters) is a *well-posed* problem, if it fulfills the following three conditions:

- *existence* — there exists a solution of the problem,
- *uniqueness* — the solution is unique,
- *stability* — the solution depends continuously on the data.

Predicting the observation from a known state is also known as the *forward problem*. Even if the forward problem is assumed to be well-posed, its inverse is not necessary well-posed. In most of the cases the forward map is not injective, and therefore the underlying state cannot be uniquely determined from a set of observations. An additional problem might occur when the observations are affected by noise. A graphical representation is shown in Figure 4.1, where the unknown state u is used to predict $z = \mathscr{L}(u)$. The observed quantity y is obtained as a noisy version of the prediction z and does not necessarily belong of the image of $\mathscr{L}$ anymore.

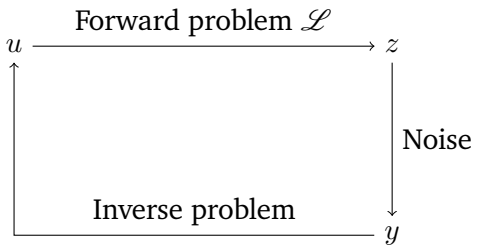


Figure 4.1.: Inverse problem.

One way of solving an inverse problem is to cast it as an optimization task. An error metric between the observation and the prediction is minimized with respect to the underlying state in order to find a solution of the inverse problem. However, this approach is affected by the lack of uniqueness of the minimizer. To this end, additional regularization of the underlying state is introduced in order to enforce uniqueness [EHN00, Bar18]. One disadvantage of classical optimization approaches is that the stochastic information offered by the noise model is lost. This work focuses on using Bayesian methods for solving inverse problems. In the Bayesian framework, the state and often also parameters as well as the observations are modeled as random variables (RVs) and the goal is to infer the distribution of the state. The noise model can therefore be incorporated in the framework. One common issue of both approaches is that the forward problem needs to be solved for several parameter combinations in order to obtain the solution. To this end, *surrogate models*, i.e., models that approximate the solution of the forward problem and are computationally inexpensive to evaluate, represent an important topic in the field of inverse problems [FMWW10].

For the rest of the chapter, we concentrate on parameter identification, i.e., the state as well as the observations belong to a finite dimensional space. The infinite dimensional case, where the state is represented as a function, can be discretized using the Karhunen-Loève expansion [OL79].

4.1. Introduction to probability theory

Before formalizing the inverse problems, we recall some important concepts from probability theory. The first concept for describing an experiment that depends on randomness is the probability space, which is a triplet $(\Omega, \Sigma, \mathbb{P})$, where

- Ω is a set of outcomes,
- Σ is a σ-algebra over Ω containing all the events that are of interest for the experiment,
- $\mathbb{P} : \Sigma \to [0, 1]$ is the probability function and has the property that it is σ-additive with $\mathbb{P}(\Omega) = 1$.

Let $(\Omega, \Sigma, \mathbb{P})$ be a probability space, $\mathcal{X}$, $\dim(\mathcal{X}) < \infty$ a measurable space with the Borel algebra $(\mathcal{X}, \mathcal{B}_\mathcal{X})$. A *random variable (RV)* is a measurable function $\boldsymbol{X} : \Omega \to \mathcal{X}$, i.e., for every set $B \in \mathcal{B}_\mathcal{X}$, the preimage $\boldsymbol{X}^{-1}(B) = \{\omega \in \Omega : \boldsymbol{X}(\omega) \in B\}$ belongs to the event space Σ. Common choices for the space $\mathcal{X}$ in this work are $\mathbb{R}^d$ and $\mathbb{N}^d$. The RVs are denoted with capital letters and, for $d > 1$, bold capital letters are used. A realization $\boldsymbol{x} = \boldsymbol{X}(\omega), \omega \in \Omega$, of a RV is denoted as a bold lowercase letter.

If the space $\mathcal{X}$ is countable, any RV $\boldsymbol{X} : \Omega \to \mathbb{X}$ is called a discrete RV. In this case, the *probability mass function (PMF)* is defined as the function

$$p_{\boldsymbol{X}}(\boldsymbol{x}) = \mathbb{P}(\{\omega \in \Omega \ : \ \boldsymbol{x} = \boldsymbol{X}(\omega)\}). \tag{4.1}$$

For a continuous RV, $\mathcal{X}, \Omega$ are uncountable and the probability mass function (PMF) is no longer the proper way of describing the RV, since the probability of unitary events is 0. Therefore, the *probability density function (PDF)* is used. Let $(\Omega, \Sigma, \mathbb{P})$ be a probability space and $\boldsymbol{X} : \Omega \to \mathcal{X}$ be a multivariate random variable. The function $p_{\boldsymbol{X}} : \mathcal{X} \to [0, \infty)$ is called the probability density function (PDF) of the RV $\boldsymbol{X}$ if it holds:

$$\mathbb{P}(\{\omega \in \Omega : \boldsymbol{X}(\omega) \in \mathcal{S}\}) = \int_{\mathcal{S}} p_{\boldsymbol{X}}(\boldsymbol{x}) \mathrm{d}\mu(\boldsymbol{x}), \quad \forall \mathcal{S} \in \mathcal{B}_\mathcal{X}, \tag{4.2}$$

for $(\mathcal{X}, \mathcal{B}_\mathcal{X})$ a Borel algebra equipped with a measure $\mu : \mathcal{B}_\mathcal{X} \to [0, \infty)$. The concept of PDF is a generalization of the PMF and for every discrete RV, that is, a PDF can be written as a superposition of Dirac-delta distributions.

Since using the PDF to describe RVs is not always computationally attractive, the expected value is an often used tool for characterizing RVs, defined as

$$\mathbb{E}(\boldsymbol{X}) = \int_{\Omega} \boldsymbol{X}(\omega) \mathrm{d}\mathbb{P}(\omega) = \int_{\mathcal{X}} \boldsymbol{x} p_{\boldsymbol{X}}(\boldsymbol{x}) \mathrm{d}\mu(\boldsymbol{x}) \in \mathcal{X}, \tag{4.3}$$

where the second part of the equation is valid if a PDF exists. In the discrete case, the expected value has the form

$$\mathbb{E}(\boldsymbol{X}) = \sum_{\boldsymbol{x} \in \mathcal{X}} \boldsymbol{x} p_{\boldsymbol{X}}(\boldsymbol{x}). \tag{4.4}$$

Two main properties are the linearity and the composition law $\mathbb{E}(f(\boldsymbol{X})) = \int_{\mathcal{X}} f(\boldsymbol{x}) p_{\boldsymbol{X}}(\boldsymbol{x}) \mathrm{d}\mu(\boldsymbol{x})$. One further instrument to characterize the distribution is the covariance, explained in the following.

Given two probability spaces, $(\Omega, \Sigma_\Omega, \mathbb{P}_\Omega)$ and $(\Pi, \Sigma_\Pi, \mathbb{P}_\Pi)$, and two RVs, $\boldsymbol{X} : \Omega \to \mathcal{X}$ and $\boldsymbol{Y} : \Pi \to \mathcal{Y}$, the joint RV is defined using the probability space $(\Omega \times \Pi, \Sigma_\Omega \times \Sigma_\Pi, \mathbb{P}_{\Sigma \times \Pi})$ as $(\boldsymbol{X}, \boldsymbol{Y}) : \Omega \times \Pi \to \mathcal{X} \otimes \mathcal{Y}$. The joint PDF is defined as the function $p_{\boldsymbol{X},\boldsymbol{Y}} : \mathcal{X} \otimes \mathcal{Y} \to [0, \infty)$ such that

$$\mathbb{P}_{\Omega \times \Pi}(\{(\omega, \pi) \in \Omega \times \Pi : (\boldsymbol{X}(\omega), \boldsymbol{Y}(\pi)) \in \mathcal{S}\}) = \int_{\mathcal{S}} p_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\mu(\boldsymbol{x}, \boldsymbol{y}), \quad \forall \mathcal{S} \in \mathcal{B}_{\mathcal{X} \times \mathcal{Y}}, \tag{4.5}$$

where μ is an appropriate measure defined on $\mathcal{X} \times \mathcal{Y}$. If the PDF of the joint RV is known, the PDFs of the individual variables can be obtained by marginalization:

$$p_{\boldsymbol{X}}(\boldsymbol{x}) = \int_{\mathcal{Y}} p_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\nu(\boldsymbol{y}), \tag{4.6}$$

where the expression for $p_{\boldsymbol{Y}}$ is similarly obtained and ν is a measure defined on $\mathcal{Y}$. For a RV that takes values in a tensor product space $\mathcal{X} = \mathcal{X}_1 \otimes \cdots \otimes \mathcal{X}_d$, the objects $p_{\boldsymbol{X}} : \mathcal{X} \to \mathbb{R}$ and $p_{X_1,\ldots,X_d} : \mathcal{X}_1 \times \cdots \times \mathcal{X}_d \to \mathbb{R}$ are equivalent. In the case of two jointly distributed RVs, their covariance is defined as

$$\mathrm{Cov}(\boldsymbol{X}, \boldsymbol{Y}) = \int_{\mathcal{X} \otimes \mathcal{Y}} (\boldsymbol{x} - \mathbb{E}(\boldsymbol{X})) \otimes (\boldsymbol{y} - \mathbb{E}(\boldsymbol{Y})) p_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\mu(\boldsymbol{x}, \boldsymbol{y}) \ \in \ \mathcal{X} \otimes \mathcal{Y}, \tag{4.7}$$

where the case $\mathrm{Cov}(X, X)$ is called the variance and is denoted as $\mathrm{Var}(X)$. The covariance $\mathrm{Cov}(\boldsymbol{X}, \boldsymbol{X}) \in \mathcal{X} \otimes \mathcal{X}$ is isomorphic to a tensor operator from $\mathcal{L}(\mathcal{X}, \mathcal{X})$.

Examples of distributions:

Some well-known and relevant distributions used in this work are:

- A RV X is uniform distributed over the interval $[a, b]$ (also written as $X \sim \mathcal{U}(a, b)$) if its PDF is

$$p_X(x) = \frac{1}{b - a}, \quad x \in [a, b], \tag{4.8}$$

  with $\mathbb{E}(X) = \frac{a+b}{2}$ and $\mathrm{Var}(X) = \frac{(b-a)^2}{12}$.

- A RV is said to be Gaussian (normal) distributed with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{d \times d}$ if its PDF is

$$p_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^d \det \boldsymbol{\Lambda}}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{\Lambda}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right), \tag{4.9}$$

  where the covariance matrix is symmetric positive definite. A Gaussian distributed RV is written as $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda})$ and has the expected value $\mathbb{E}(\boldsymbol{X}) = \boldsymbol{\mu}$ and covariance $\mathrm{Cov}(\boldsymbol{X}, \boldsymbol{X}) = \boldsymbol{\Lambda}$.

- A univariate RV X defined on $(0, \infty)$ is Gamma distributed ($X \sim \Gamma(\alpha, \beta)$) with the shape parameter $\alpha > 0$ and the inverse scale parameter $\beta > 0$ if its PDF is

$$p_X(x) = \frac{x^{\alpha-1} e^{-\beta x} \beta^\alpha}{\int_0^\infty x^{\alpha-1} e^{-\beta x} \beta^\alpha \mathrm{d}x}, \quad x > 0, \tag{4.10}$$

  with $\mathbb{E}(X) = \frac{\alpha}{\beta}$ and $\mathrm{Var}(X) = \frac{\alpha}{\beta^2}$.

- The RV $X \in [0,1]$ is said to be Beta distributed with the shape parameters $\alpha, \beta > 0$ if it has the PDF

$$p_X(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 x^{\alpha-1}(1-x)^{\beta-1}\mathrm{d}x}, \tag{4.11}$$

  with $\mathbb{E}(X) = \frac{\alpha}{\alpha+\beta}$ and $\mathrm{Var}(X) = \frac{\alpha\beta}{(\alpha+\beta)^2(1+\alpha+\beta)}$.

4.1.1. Conditioning and Bayes theorem

One important concept in the field of Bayesian inverse problems is the conditional probability. If two events A and B are dependent, one can ask how does the probability of observing A changes if the event B is observed. Intuitively, this is obtained as the probability of both events A, B to occur normalized to the probability of B to occur and is formalized in the following definition:

Definition 4.1.1 (Conditional probability)**.** Let $(\Omega, \Sigma, \mathbb{P})$ be a probability space. The conditional probability of A given that B is also observed is defined as

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cup B)}{\mathbb{P}(B)}, \quad \mathbb{P}(B) \neq 0. \tag{4.12}$$

By writing the definition for both $\mathbb{P}(A|B)$ and $\mathbb{P}(B|A)$ and eliminating the term $\mathbb{P}(A \cup B)$, the following relation is obtained:

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}, \quad \mathbb{P}(B) \neq 0. \tag{4.13}$$

This simple but very powerful formula is called the *Bayes theorem* (also known as Bayes rule) and it is the core of many machine learning methods [Jay03, Bar12, The15]. It relates the probability of a hypothesis A given some data B is observed to the product between the likelihood $\mathbb{P}(B|A)$ of the data to be observed, given that the hypothesis A is true, multiplied by the probability of the hypothesis $\mathbb{P}(A)$ without knowing the observation. The term $\mathbb{P}(B)$ is the probability that the data is observed and has the role of a normalization constant.

Using the Bayes rule for probabilities is not feasible in the context of inverse problems with continuous RVs, since conditioning on pointwise observations yields a zero normalization constant. To this end, the concept of conditioning has to be extended to PDFs. According to the definition from (4.2), the PDF of a RV $\boldsymbol{X}$ conditioned on an observation of a dependent RV has to fulfill the following equation:

$$\mathbb{P}(\boldsymbol{X} \in \mathcal{S}|\boldsymbol{Y} = \boldsymbol{y}) = \int_{\mathcal{S}} p_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y})\mathrm{d}\mu(\boldsymbol{x}), \quad \forall \mathcal{S} \text{ outcome set.} \tag{4.14}$$

This requirement is fulfilled by the following definition:

Definition 4.1.2 (Conditional PDF)**.** Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be two RVs with the joint PDF $p_{\boldsymbol{X},\boldsymbol{Y}}$. The PDF of $\boldsymbol{X}$ conditioned on the observation $\boldsymbol{Y} = \boldsymbol{y}$ is

$$p_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y}) = \frac{p_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})}{\int_{\mathcal{X}} p_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\mu(\boldsymbol{x})} = \frac{p_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})}{p_{\boldsymbol{Y}}(\boldsymbol{y})}, \quad p_{\boldsymbol{Y}}(\boldsymbol{y}) \neq 0, \tag{4.15}$$

where $\mathcal{X}$ is the image of $\boldsymbol{X}$ and μ is a measure.

With the definition of the conditional PDF, the Bayes rule in terms of density functions can be expressed as

$$p_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y}) = \frac{p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x})p_{\boldsymbol{X}}(\boldsymbol{x})}{p_{\boldsymbol{Y}}(\boldsymbol{y})}, \quad p_{\boldsymbol{Y}}(\boldsymbol{y}) \neq 0. \tag{4.16}$$

Note that if two RVs are independent, i.e., $p_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y}) = p_{\boldsymbol{X}}(\boldsymbol{x})p_{\boldsymbol{Y}}(\boldsymbol{y})$, conditioning gives $p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x}) = p_{\boldsymbol{Y}}(\boldsymbol{y})$.

4.2. Inverse problems in the Bayesian setup

As previously mentioned, in the Bayesian setup, the parameter vector is considered to be a RV, denoted in the following as $\boldsymbol{U} : \Omega \to \mathcal{U}$, where $(\Omega, \Sigma, \mathbb{P})$ is the probability space used to describe the entire experiment [Stu10, MZR$^+$16]. The prediction is also a RV and is obtained by the composition $\boldsymbol{Z} = \mathscr{L} \circ \boldsymbol{U}$. The observation noise is modeled using the RV $\boldsymbol{Y}$ defined in the general case as

$$\boldsymbol{Y}(\omega, \pi) = \boldsymbol{Y}(\mathscr{L}(\boldsymbol{U}(\omega)), \pi), \quad \boldsymbol{Y} : \Omega \times \Pi \to \mathcal{Y}, \tag{4.17}$$

where $(\Pi, \Sigma_\Pi, \mathbb{P}_\Pi)$ is a probability space and $\mathcal{Y}$ is a suitable space where the observations take values. Since the observation is just a noisy version of the prediction, it is reasonable to assume that the RV also takes values in $\mathcal{Y}$, $\mathrm{Im}(\boldsymbol{Z}) \subseteq \mathcal{Y}$. It relates the distribution of the parameter $\boldsymbol{U}$ to the observation $\boldsymbol{Y}$ using the forward problem. If a realization of the parameter is available, the forward problem is used to obtain a prediction and the prediction is modified according to the noise model. This is graphically illustrated in Figure 4.2. Solving the inverse problem means in this framework conditioning the RV $\boldsymbol{U}$ on realizations of $\boldsymbol{Y}$ [MZRL16, VM18, Stu10]. One particular model also used in this work is the additive noise. In this case, the RV $\boldsymbol{Y}$ has the form

$$\boldsymbol{Y}(\omega, \pi) = \mathscr{L}(\boldsymbol{U}(\omega)) + \boldsymbol{\Upsilon}(\pi), \tag{4.18}$$

where $\boldsymbol{\Upsilon}$ is a random variable representing the discrepancy between the output and the observation (measurement error).

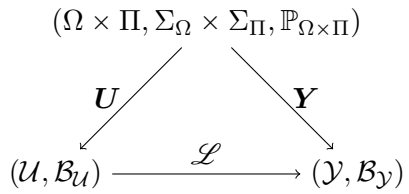


Figure 4.2.: Diagram illustrating the relation between the RVs. The image of the map $\mathscr{L}$ is assumed to be a subset of the set $\mathcal{Y}$.

The treatment of the inverse problem simplifies when expressed in terms of PDFs [Stu10]. Due to the connection between the observation and the parameter by means of the forward problem, the observation and the parameter are jointly distributed. Under the assumption that the joint PDF exists, the conditional PDF of $\boldsymbol{U}$ given the observation $\hat{\boldsymbol{y}}$ is well-defined and the Bayes law can be applied to obtain the following expression for the *posterior PDF* $p_{\boldsymbol{U}|\boldsymbol{Y}}$:

$$p_{\boldsymbol{U}|\boldsymbol{Y}}(\boldsymbol{u}|\hat{\boldsymbol{y}}) = \frac{p_{\boldsymbol{Y}|\boldsymbol{U}}(\hat{\boldsymbol{y}}|\boldsymbol{u}) p_{\boldsymbol{U}}(\boldsymbol{u})}{p_{\boldsymbol{Y}}(\hat{\boldsymbol{y}})}. \tag{4.19}$$

The PDF $p_{\boldsymbol{U}}$, also called *prior*, incorporates an initial knowledge about the underlying parameter, the conditional PDF $p_{\boldsymbol{Y}|\boldsymbol{U}}$ is called *likelihood* and the normalization constant $p_{\boldsymbol{Y}}(\hat{\boldsymbol{y}})$ is called *evidence*. The underlying physical model given by the forward problem is included in the likelihood. Therefore, for every evaluation of the posterior, a forward problem has to be solved. One concrete example is the likelihood for the additive noise case from (4.18), where the likelihood is expressed as

$$p_{\boldsymbol{Y}|\boldsymbol{U}}(\hat{\boldsymbol{y}}|\boldsymbol{u}) = p_{\boldsymbol{\Upsilon}}(\mathcal{L}(\boldsymbol{u}) - \hat{\boldsymbol{y}}), \tag{4.20}$$

for $p_{\boldsymbol{\Upsilon}}$ the PDF of the error.

4.2.1. Hidden Markov models and recursive state estimation

Evaluating and numerically approximating the posterior becomes computationally unattractive even for a small number of dimensions. When dealing with time dependent inverse problems, where the state evolves with time, the dimensionality of the joint density scales linearly with the number of timesteps. Moreover, since in practical cases not all measurements are available at once, some factorization of the joint has to be assumed [Che03].

Consider a sequence of RVs $\{\boldsymbol{U}\}_{k=1}^{n}$, each of them corresponding to the state of the system at discrete timesteps indexed by k. At every timestep, the indirect observation of the state is modeled by the RV sequence $\{\boldsymbol{U}\}_{k=1}^{n}$. The unobserved underlying state is also called *hidden* or *latent* state. One first assumption is that the k-th observation $\boldsymbol{Y}^{(k)}$ depends only on the current state $\boldsymbol{U}^{(k)}$. A second assumption is that the current state depends only on the previous state. These two assumptions are formalized in the following definition, with the notation $\boldsymbol{U}^{(<k)} = \boldsymbol{U}^{(k-1)}, \boldsymbol{U}^{(k-2)}, \dots$ and $\boldsymbol{U}^{(\leqslant k)} = \boldsymbol{U}^{(k)}, \boldsymbol{U}^{(k-1)}, \dots$.

A graphical representation of the hidden Markov model (HMM) is shown in Figure 4.3.

Definition 4.2.1 (Hidden Markov model)**.** Let $\{\boldsymbol{U}^{(k)}\}_{k=1}^{n}$ and $\{\boldsymbol{Y}^{(k)}\}_{k=1}^{n}$ be sequences of jointly distributed state and observation RVs. The family of RVs build a *hidden Markov model (HMM)* if the following factorization holds:

$$p_{\boldsymbol{Y}^{(k)}|\boldsymbol{U}^{(\leqslant k)},\boldsymbol{Y}^{(<k)}}(\boldsymbol{y}^{(k)}|\boldsymbol{u}^{(k)},...,\boldsymbol{u}^{(0)},\boldsymbol{y}^{(k)},...,\boldsymbol{y}^{(1)}) = p_{\boldsymbol{Y}^{(k)}|\boldsymbol{U}^{(k)}}(\boldsymbol{y}^{(k)},\boldsymbol{u}^{(k)}),$$
$$\forall \boldsymbol{u}^{(k)},...,\boldsymbol{u}^{(0)},\boldsymbol{y}^{(k)},...,\boldsymbol{x}^{(1)}, k \in \{1,...,n\}. \tag{4.21}$$

A HMM is called a *hidden Markov chain* if it fulfills the additional property

$$p_{\boldsymbol{U}^{(k)}|\boldsymbol{U}^{(<k)}}(\boldsymbol{u}^{(k)}|\boldsymbol{u}^{(k-1)},...,\boldsymbol{u}^{(0)}) = p_{\boldsymbol{U}^{(k)}|\boldsymbol{U}^{(k-1)}}(\boldsymbol{u}^{(k)}|\boldsymbol{u}^{(k-1)}), \quad \forall \boldsymbol{x}^{(k)},...,\boldsymbol{x}^{(0)}, k \in \{1,...,n\}. \tag{4.22}$$

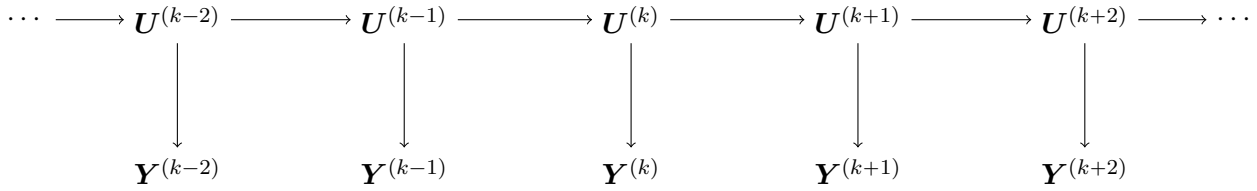


Figure 4.3.: Diagram illustrating the relation between the RVs in a HMM. The arrows represent the conditional dependencies of the RVs.

Inference tasks on HMMs

One relevant inference task on a HMM is the state prediction. If the PDF of the state at the step $k-1$ is known, the distribution of $\boldsymbol{U}^{(k)}$ is obtained using marginalization and the definition of the conditional PDF

$$\begin{aligned} p_{\boldsymbol{U}^{(k)}}(\boldsymbol{u}^{(k)}) &= \int_{\mathcal{U}} p_{\boldsymbol{U}^{(k)},\boldsymbol{U}^{(k-1)}}(\boldsymbol{u}^{(k)},\boldsymbol{u}^{(k-1)})\mathrm{d}\boldsymbol{u}^{(k-1)} \\ &= \int_{\mathcal{U}} p_{\boldsymbol{U}^{(k)}|\boldsymbol{U}^{(k-1)}}(\boldsymbol{u}^{(k)}|\boldsymbol{u}^{(k-1)}) p_{\boldsymbol{U}^{(k-1)}}(\boldsymbol{u}^{(k-1)})\mathrm{d}\boldsymbol{u}^{(k-1)}, \end{aligned} \tag{4.23}$$

where $p_{\boldsymbol{U}^{(k)}|\boldsymbol{U}^{(k-1)}}$ is the transition PDF which incorporates the time evolution equations and captures the process noise. A similar expression can be derived for the conditional $p_{\boldsymbol{U}^{(k)}|\boldsymbol{Y}^{(<k)}}$ in terms of $p_{\boldsymbol{U}^{(k-1)}|\boldsymbol{Y}^{(<k)}}$.

When a new measurement is taken into account, the Bayes rule can be used to relate the so-called filter PDF to the prediction using the noise model $p_{\boldsymbol{Y}^{(k)}|\boldsymbol{X}^{(k)}}$

$$p_{\boldsymbol{U}^{(k)}|\boldsymbol{Y}^{(\leqslant k)}}(\boldsymbol{u}^{(k)}|\boldsymbol{y}^{(k)},...) \propto p_{\boldsymbol{Y}^{(k)}|\boldsymbol{U}^{(k)}}(\boldsymbol{Y}^{(k)}|\boldsymbol{U}^{(k)}) p_{\boldsymbol{U}^{(k-1)}|\boldsymbol{Y}^{(<k)}}(\boldsymbol{u}^{(k)}|\boldsymbol{y}^{(k-1)},...). \tag{4.24}$$

Equations (4.23) and (4.24) can be recurrently used to incorporate new measurements into the state estimation starting from the prior $p_{\boldsymbol{U}^{(0)}}$. Smoothing is the third task and consists in describing the state RV given that all measurements have been accounted for, i.e., the PDFs $p_{\boldsymbol{U}^{(k)}|\boldsymbol{Y}^{\leqslant n}}, k \in \{1,...,n\}$. In Sections 5.4.1 and 5.4.2, the algorithm to compute the smoothing will be presented in detail.

4.2.2. Posterior approximation

In the case the system state is described by a vector of real numbers $\mathcal{U} = \mathbb{R}^d$, the prior and posterior PDFs are multivariate real-valued functions and therefore can be approximated using classical interpolation techniques. In this work, we concentrate on interpolation using tensor product grids. The reason for this is the ability to compress the DoFs tensor using the TT format presented in the previous chapter. An approximation from a tensor product space $\bigotimes_{k=1}^{d} \text{span}(b_i^{(k)} \ : \ i = 1,...,n_k)$ is obtained from univariate bases $b_i^{(k)}$. Regarding the choice of the basis, B-splines (presented in Appendix A) are used in this work due to their numerical stability and their locality [IWL+21]. The discretized posterior is expressed as a linear combination

$$p_{\boldsymbol{U}|\boldsymbol{Y}}(\boldsymbol{u}|\hat{\boldsymbol{y}}) \approx \sum_{\boldsymbol{k}} \mathsf{p}_{\boldsymbol{k}} b_{\boldsymbol{k}}(\boldsymbol{u}) = \sum_{\boldsymbol{k}} \mathsf{p}_{\boldsymbol{k}} b_{k_1}^{(1)}(u_1) \cdots b_{k_d}^{(d)}(u_d), \tag{4.25}$$

where $\mathbf{p} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ are the DoFs and can be obtained by interpolation. To this end, a collocation grid is defined as a Cartesian product of univariate grids $\times_{k=1}^{d} \{u_{i_k}^{C,k}\}_{i_k=1}^{n_k}$. Recovering the DoF tensor in the TT format can be achieved by solving the following linear system

$$\sum_{\boldsymbol{k}} \mathsf{p}_{\boldsymbol{k}} b_{\boldsymbol{k}}(\boldsymbol{u}_{\boldsymbol{i}}^{C}) = p_{\boldsymbol{U}|\boldsymbol{Y}}(\boldsymbol{u}_{\boldsymbol{i}}^{C}|\hat{\boldsymbol{y}}), \quad \forall \boldsymbol{i}, \tag{4.26}$$

where the choice of the univariate grids has to ensure that the matrices with the entries $B_{ij}^{k} = b_j^{(k)}(u_j^{C,k})$ are invertible. Representing the right-hand in the TT format can be achieved using the interpolation method presented in Section 3.2.5 in order to avoid constructing the full tensor. In the case where a surrogate model of the forward problem is available, this task becomes computationally inexpensive. After having an accurate approximation of the posterior, UQ tasks such as computing moments, modes, calculating marginals and sampling [DAIFS20] can be efficiently accomplished. The two error sources are the truncation of the space $\mathcal{U}$ and the discretization error introduced by the finite dimensional basis. The latter can be controlled by refinement strategies.

Alternatively, one can directly compute moments of the posterior using the TT cross approximation using tensor product numerical quadrature schemes [Xiu10], such that

$$\mathbb{E}(g(\boldsymbol{U})) = \int_{\mathcal{U}} g(\boldsymbol{u}) p_{\boldsymbol{U}|\boldsymbol{Y}}(\boldsymbol{u}|\hat{\boldsymbol{y}}) \mathrm{d}\boldsymbol{u} = \sum_{\boldsymbol{i}} \mathsf{w}_{\boldsymbol{i}} g(\boldsymbol{u}_{\boldsymbol{i}}^{Q}) p_{\boldsymbol{U}|\boldsymbol{Y}}(\boldsymbol{u}_{\boldsymbol{i}}^{Q}|\hat{\boldsymbol{y}}), \tag{4.27}$$

where $\boldsymbol{u}_{\boldsymbol{i}}^{Q} = (u_{i_1}^{Q,1},...,u_{i_d}^{Q,d})$ is the grid obtained from the individual univariate quadrature points with the corresponding weights forming the rank-1 tensor $\mathbf{w} = \boldsymbol{w}^{Q,1} \otimes \cdots \otimes \boldsymbol{w}^{Q,d}$. The evaluation of the integrand is obtained again using the cross interpolation method. In [Xiu10], several choices of quadrature points and

weights are proposed for the case when the density function has some canonical format: Gauss-Hermite quadrature points are used for Gaussian distributions, Gauss-Legendre quadrature for uniform distributions, etc. Since the posterior does not necessarily belong to the family of known PDFs, it is usual to use the quadrature scheme corresponding to the prior with a sufficiently high order [VM18].

4.2.3. Overview of Bayesian inversion methods

One common issue of all the Bayesian inversion methods is the need for a high number of evaluations of the posterior and thus the forward problem solver. Especially for systems described by differential equations, where a simulation for a single parameter can even take days to complete, the Bayesian inversion methods become inefficient. This motivates the need of using surrogate models in order to avoid a high number of expensive simulations. In the following, we recall three classes of methods:

Sampling based methods

As the name suggests, they rely on drawing a sample from the posterior distribution using Markov chain Monte Carlo (MCMC) [Tie94, BGJM11]. Having a sample from the posterior, one can compute moments or even histograms of marginals. Their advantage is that they do not require a smooth forward map and they are suited for high dimensional parameter spaces [Tar05, Bar18]. As a main disadvantage, the high number of posterior PDF evaluations makes the method unusable for systems with computationally expensive forward problems. To address this problem, some authors proposed the use of surrogate models [LM14, MX09, FLU$^+$20, YZ19b, YZ21, YZ19a].

Pointwise estimators

One very popular technique is the maximum a posteriori estimation (MAP), which searches for the maximizer of the posterior [Bar18, DG08]. The optimization problem can be difficult to solve especially for multimodal distributions or high number of dimensions. Moments of the posterior can as well be computed by numerical integration [VM18, MZRL16]. In the context of recursive state estimation for time dependent processes, the most known method is the Kalman filter introduced in [Kal60]. The initial variant was developed for linear inverse problems. For a Gaussian noise model, extensions have been made for nonlinear systems [Che03]. The result is the posterior mean and covariance at every time step. While the pointwise estimations are sufficient for a large class of applications, they are not good at capturing multimodal PDFs.

Variational inference

In variational inference, the posterior is approximated from a family of known distributions [BKM16]. The basic idea is to search for an approximation of the intractable posterior within a tractable family of distributions. As a discrepancy metric, some distance is chosen, typically the Kullback-Leibler divergence. Since the computation of the Kullback-Leibler divergence requires numerical integration, this method is affected by the curse of dimensionality. Some applications of the variational inference for inverse problems are found in [TRT$^+$20, AIJZ17, MTRP09, GWE$^+$15, aSYK$^+$04].

5. Tensor-Train for the Chemical Master Equation

When dealing with biological processes involving a low number of molecule instances, stochastic kinetics models have to be used [SYSY02]. The CME is a fundamental equation that probabilistically describes such systems under certain assumptions [Gil92]. Despite being just a linear initial value problem, the CME suffers from the curse of dimensionality, making the naive integration computationally expensive even for simple models. Since the solution of the CME is represented as a tensor, it is natural to directly apply tensor decomposition methods. Compressing the solution alone is not sufficient if the solver works with full high dimensional tensors. To this end, the so called CME generator has to be constructed in the TT format and the time integration has to be performed in the low-rank format as well. For the latter, the so called time dependent AMEn solver is used [Dol18]. Adding parameter dependence to the CME is resolved by combining the state space and the parameter space into a higher-dimensional tensor product space. Galerkin projection is then used derive a multilinear system in TT for the combined state-parameter DoF tensor.

The process of parameter identification can be modeled as an HMM where the hidden state is the joint state-parameter distribution at different time steps. In this case, the transition between two consecutive hidden states k and $k+1$ is performed by solving the CME for the corresponding time interval. The observation operator can also be computed in the TT format. After performing filtering for all observations, the resulting state-parameter joint is marginalized to obtain the posterior distribution over the parameter space given all observations. The fast and efficient algebra in the TT format makes it possible to perform all operations with linear complexity w.r.t. the number of dimensions.

The content presented in this chapter is based on our work [IWL+21] and is structured as follows: In Section 5.1, the CME is introduced together with an overview of its derivation. Next, in Section 5.2, we show how the TT format is employed to construct the CME generator, to solve the equation, and to tackle the parameter dependence. In Section 5.4, we apply the CME solver to perform Bayesian inference tasks such as filtering, smoothing, and parameter inference. Numerical results are shown in Section 5.5 to validate the proposed method.

5.1. Chemical Master Equation

We consider a system consisting of molecules of d species denoted by $\{S_1, ..., S_d\}$. The molecule copies are involved in n_R reactions of type:

$$R_m \quad : \quad q_{m,1}S_1 + \cdots + q_{m,d}S_d \longrightarrow s_{m,1}S_1 + \cdots + s_{m,d}S_d, \quad m = 1, \ldots, n_\mathrm{R} \tag{5.1}$$

where $q_{m,k} \in \mathbb{N}_0, k = 1, \ldots, d$, represent the number of molecules from each species that react and $s_{m,k} \in \mathbb{N}_0$, $k = 1, \ldots, d$, is the number of produced molecules after a reaction R_m takes place. Furthermore, we assume that all instances of the mentioned species are contained inside a constant volume. The state of the system is described at a macroscopic level as the number of the individuals per species at every time point $t \geqslant 0$ and is denoted by the vector $\boldsymbol{x}(t) \in \mathbb{N}_0^d$. In case the m-th reaction occurs, the state vector change is given by $\boldsymbol{x} \to \boldsymbol{x} + \boldsymbol{\nu}^{(m)}$, where the stoichiometric change vector $\boldsymbol{\nu}^{(m)} \in \mathbb{Z}^d$ is elementwise defined

as $\nu_k^{(m)} = s_{m,k} - q_{m,k}$. Since the position and the velocity of the individual molecules are considered to be RVs, an exact tracking of the evolution and the collisions that lead to the occurrence of the chemical reactions is not feasible. Such a system is therefore modeled as a continuous-time jump process $\{\boldsymbol{X}(t)\}_{t\geqslant 0}$ and the goal is to describe the time evolution of the PMF describing the process

$$p(\boldsymbol{x}, t) = p_t(\boldsymbol{x}) = \mathbb{P}(\boldsymbol{X}(t) = \boldsymbol{x}) = \mathbb{P}(X_1(t) = x_1, \ldots, X_d(t) = x_d). \tag{5.2}$$

Under further assumptions regarding the positions and the velocities of the individual species instances, an ODE can be derived for the PMF in (5.2). The first assumption is that the molecules are **well-stirred**, meaning that the position of all molecule copies are considered to be uniformly distributed RVs. The second assumption is that the system is at **thermal equilibrium**, implying that the components of the velocities are normally distributed with mean 0 and constant variance. In [Gil92], using rigorous microscopical considerations, it is shown that the probability of the m-th reaction to occur in the infinitesimal interval $[t, t + \mathrm{d}t)$, given that the current state of the system $\boldsymbol{x}$, can be expressed in terms of the propensity function

$$\alpha_m(\boldsymbol{x})\mathrm{d}t = c_m h^{(m)}(\boldsymbol{x})\mathrm{d}t. \tag{5.3}$$

The constant c_m is called the reaction rate constant, summarizes the chemical properties of the reactants as well as the temperature and the volume of the domain [Gil92] and has the meaning that $c_m \mathrm{d}t$ is the probability that a random combination of reactants will collide and react in the interval $[t, t + \mathrm{d}t)$. The term $h_m(\boldsymbol{x})$ counts the possible combinations of molecules that are eligible for a reaction

$$h^{(m)}(\boldsymbol{x}) = \prod_{k=1}^{d} \frac{x_k!}{q_{m,k}!(x_k - q_{m,k})!}, \tag{5.4}$$

Using the defined propensity function, in an infinitesimal interval $[t, t + \mathrm{d}t)$, the following events can occur:

- The probability of one reaction R_m occurring in the interval $[t, t + \mathrm{d}t)$ is

$$\mathbb{P}(\text{one } R_m \text{ occurs in } [t, t + \mathrm{d}t)) = \alpha_m(\boldsymbol{x}(t))\mathrm{d}t + \mathcal{O}(\mathrm{d}t^2). \tag{5.5}$$

- The probability that no reaction takes place in the interval $[t, t + \mathrm{d}t)$ is

$$\mathbb{P}(\text{no reaction in } [t, t + \mathrm{d}t)) = 1 - \sum_{m=1}^{n_\mathrm{R}} \alpha_m(\boldsymbol{x}(t))\mathrm{d}t + \mathcal{O}(\mathrm{d}t^2). \tag{5.6}$$

- The probability that more than one reaction occur in the time interval $[t, t + \mathrm{d}t)$ is

$$\mathbb{P}(\text{more than one}) = \mathcal{O}(\mathrm{d}t^2). \tag{5.7}$$

Since the events mentioned above are mutually exclusive in the interval $[t, t + \mathrm{d}t)$ [Gil92], the evolution of the PMF is

$$\begin{aligned} p(\boldsymbol{x}, t + \mathrm{d}t) =& p(\boldsymbol{x}, t)\left(1 - \sum_{m=1}^{n_\mathrm{R}} \alpha_m(\boldsymbol{x}(t))\mathrm{d}t + \mathcal{O}(\mathrm{d}t^2)\right) + \\ & \sum_{m=1}^{n_\mathrm{R}} p(\boldsymbol{x} - \boldsymbol{\nu}^{(m)}, t)\left(\alpha_m(\boldsymbol{x}(t) - \boldsymbol{\nu}^{(m)})\mathrm{d}t + \mathcal{O}(\mathrm{d}t^2)\right) + \mathcal{O}(\mathrm{d}t^2). \end{aligned} \tag{5.8}$$

Rearranging the terms, dividing by $\mathrm{d}t$, and taking the limit $\mathrm{d}t \to 0$ yields the CME

$$\frac{\mathrm{d}p_t(\boldsymbol{x})}{\mathrm{d}t} = \sum_{m=1}^{n_\mathrm{R}} \left\{ \alpha_m(\boldsymbol{x} - \boldsymbol{\nu}^{(m)}) p_t(\boldsymbol{x} - \boldsymbol{\nu}^{(m)}) - \alpha_m\left(\boldsymbol{x}\right) p_t\left(\boldsymbol{x}\right) \right\}, \tag{5.9a}$$

$$p_0(\boldsymbol{x}) = P^{(0)}(\boldsymbol{x}), \tag{5.9b}$$

where the PMF at the time $t = 0$ is assumed to be known. As an alternative to solving the CME to obtain the PMF, the stochastic simulation algorithm (SSA) algorithm has been developed to compute realizations of the stochastic process $\{\boldsymbol{X}(t)\}_{t \geqslant 0}$ [Gil76]. A sufficient number of realizations provides an accurate description of the system. However, events occurring with a low probability (rare events) need numerous realizations.

5.2. Chemical Master Equation in Tensor Notation

In the previous section, the solution of the CME was defined as the PMF $p_t : \mathbb{N}_0^d \to \mathbb{R}$. Since an infinite state-space is computationally intractable, an approximation has to be performed. To that end, a restriction of the definition domain of the continuous-time jump process $\{\boldsymbol{X}(t)\}_{t \geqslant 0}$ is needed, such that $x_k < n_k$, $k = 1, \ldots, d$. We denote the truncated state space as

$$\mathcal{X} = \left\{ \boldsymbol{x} \in \mathbb{N}_0^d \,|\, x_k < n_k, k = 1, \ldots, d \right\}. \tag{5.10}$$

The box domain structure of $\mathcal{X}$ is necessary in order to use the TT format for approximating the solution. Some methods such as the sliding window approach [WGMH10] adapt the bounds of the tensor product grid in (5.10) to capture only the relevant parts of the PMF. Before introducing the tensor notation, we note that all states in $\mathcal{X}$ are labeled using the bijection $\boldsymbol{x}(\boldsymbol{i})$, $x_k(i_k) = i_k - 1$, $i_k = 1, \ldots, n_k$, where $\boldsymbol{i} = (i_1, \ldots, i_d) \in \mathbb{N}^d$ is a multiindex.

The solution of the CME can therefore be represented by the time dependent tensor $\mathsf{p}\,(t) \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, with the elements

$$\mathsf{p}_{\boldsymbol{i}}(t) = p(\boldsymbol{x}(\boldsymbol{i}), t). \tag{5.11}$$

The differential equation from (5.9a) is also expressed as a linear ODE

$$\frac{\mathrm{d}\mathbf{p}(t)}{\mathrm{d}t} = \mathbf{A}\mathbf{p}(t), \tag{5.12}$$

where $\mathbf{A} \in \mathbb{R}^{(n_1 \times \cdots \times n_d) \times (n_1 \times \cdots \times n_d)}$ is a tensor operator with the elementwise definition

$$\mathsf{A}_{\boldsymbol{i},\boldsymbol{j}} = \sum_{m=1}^{n_\mathrm{R}} \left(\alpha_m(\boldsymbol{x}(\boldsymbol{i}) - \boldsymbol{\nu}^{(m)}) \delta^{\boldsymbol{x}(\boldsymbol{j})}_{\boldsymbol{x}(\boldsymbol{i}) - \boldsymbol{\nu}^{(m)}} - \alpha_m\left(\boldsymbol{x}(\boldsymbol{i})\right) \delta^{\boldsymbol{x}(\boldsymbol{j})}_{\boldsymbol{x}(\boldsymbol{i})} \right) \mathbb{I}_{\boldsymbol{n}}(\boldsymbol{x}(\boldsymbol{j}) + \boldsymbol{\nu}^{(m)}), \tag{5.13}$$

where $\delta^{\boldsymbol{j}}_{\boldsymbol{i}} = \delta^{j_1}_{i_1} \cdots \delta^{j_d}_{i_d}$ is the multiindex Kronecker delta and the indicator function $\mathbb{I}_{\boldsymbol{n}}(\boldsymbol{x})$ takes the value one if $x_k \leqslant n_k, k = 1, ..., d$, and zero otherwise. The addition of the indicator function allows for a boundary condition of the new state space $\mathcal{X}$ such that the mass is preserved [MK06]. The storage complexity for the solution $\mathbf{p}(t)$ at any given time point is exponential with respect to the number of species, i.e., $\mathcal{O}(n^d)$, where $n = \max_k \{n_k\}$. This makes solving the CME prohibitive even for a small number of species. One way to deal with the curse of dimensionality is to employ the already presented low rank tensor format presented in Section 3.2.

5.3. Solving the Chemical Master Equation in the Tensor-Train Format

5.3.1. Low-rank TT representation of the CME operator

The first step needed for solving the CME in the TT format is to obtain a low rank representation of the CME generator. In Section 5.1, the general form of a mass-action propensity function has been given. The structure of the mass-action propensity function can be leveraged to directly obtain a low-rank TT representation of the CME operator without having the need to work with full tensors [DK15, HG11].

We first compute the generator for one reaction since the final operator can be obtained as the sum of the tensor-operators corresponding to the individual reactions. For the m-th reaction, the mass action propensity from (5.4) can be decomposed as

$$\alpha_m(\boldsymbol{x}) = c_m h_1^{(m)}(x_1)\cdots h_d^{(m)}(x_d), \tag{5.14a}$$

$$h_k^{(m)}(x_k) = \frac{x_k!}{q_{m,k}!\,(x_k - q_{m,k})!}. \tag{5.14b}$$

In order to use the rank-1 structure of the decomposed propensity, the CME operator corresponding to the m-th reaction in (5.13) is expressed as the difference between two tensor operators

$$\mathsf{A}_{\boldsymbol{i},\boldsymbol{j}} = \mathsf{B}_{\boldsymbol{i},\boldsymbol{j}} - \mathsf{C}_{\boldsymbol{i},\boldsymbol{j}}, \tag{5.15}$$

where

$$\mathsf{B}_{\boldsymbol{i},\boldsymbol{j}} = \alpha_m(\boldsymbol{x}(\boldsymbol{i}) - \boldsymbol{\nu}^{(m)})\delta^{\boldsymbol{x}(\boldsymbol{j})}_{\boldsymbol{x}(\boldsymbol{i})-\boldsymbol{\nu}^{(m)}}\mathbb{I}_{\boldsymbol{n}}(\boldsymbol{x}(\boldsymbol{j}) + \boldsymbol{\nu}^{(m)}), \tag{5.16a}$$

$$\mathsf{C}_{\boldsymbol{i},\boldsymbol{j}} = \alpha_m\left(\boldsymbol{x}(\boldsymbol{i})\right)\delta^{\boldsymbol{x}(\boldsymbol{j})}_{\boldsymbol{x}(\boldsymbol{i})}\mathbb{I}_{\boldsymbol{n}}(\boldsymbol{x}(\boldsymbol{j}) + \boldsymbol{\nu}^{(m)}). \tag{5.16b}$$

Using the factorization (5.14b), the TT-cores of the two rank-1 tensor-operators $\mathbf{B}, \mathbf{C} \in \mathbb{R}^{(n_1\times\cdots\times n_d)\times(n_1\times\cdots\times n_d)}$ are given as

$$\mathsf{g}^{(\mathbf{C},k)}_{1 i_k j_k 1} = h_k^{(m)}(x_k(i_k))\delta^{j_k}_{i_k}\mathbb{I}_{n_k}(x_k(j_k) + \nu_k^{(m)}), \tag{5.17a}$$

$$\mathsf{g}^{(\mathbf{B},k)}_{1 i_k j_k 1} = h_k^{(m)}(x_k(j_k))\delta^{j_k}_{i_k-\nu_k}\mathbb{I}_{n_k}(x_k(j_k) + \nu_k^{(m)}), \tag{5.17b}$$

where the indicator functions $\mathbb{I}_{n_k}$ return 1 if $x \leqslant n_k$ and 0 otherwise. Since both of the tensors are rank-1, the result of their addition has the TT-rank $(1, 2, ..., 2, 1)$. If we consider the CME generator for a network with N_r reactions, the maximum rank of its TT representation is at most $2N_\mathrm{r}$. Performing TT-rank rounding (Algorithm 4) has been shown to return an approximation $\widetilde{\mathbf{A}} \approx \mathbf{A}$ with a lower rank for a very high accuracy (for example $\epsilon = 10^{-12}$).

5.3.2. Solving the CME in the TT format

The usual finite-difference schemes can be used to solve the CME in the TT format [Gel17]. The general form of a finite-difference scheme applied to the CME in the TT format is

$$\cdots + \alpha_{-1}\mathbf{p}_{:,j-1} + \alpha_0\mathbf{p}_{:,j} + \alpha_1\mathbf{p}_{:,j+1} + \cdots = \mathbf{A}\left(\cdots + \beta_{-1}\mathbf{p}_{:,j-1} + \beta_0\mathbf{p}_{:,j} + \beta_1\mathbf{p}_{:,j+1} + \cdots\right), \tag{5.18}$$

where $\mathbf{p}_{:,j}$ represents the slice of the tensor corresponding to the j-th timestep and the DoFs are defined as $\mathbf{p}_{\boldsymbol{i}j} = p(\boldsymbol{x}(\boldsymbol{i}), t^{(j)})$ for an equidistant time grid $\{t^{(j)}\}_{j=1}^{n_T}$ containing n_T timesteps. In the case of the explicit Euler method, the only nonzero coefficients are $\alpha_0 = 1$, $\alpha_{-1} = -1$, $\beta_1 = \Delta t$, and for the Crank-Nicolson

method we have $\alpha_1 = 1$, $\alpha_0 = -1$, $\beta_0 = \beta_1 = \Delta t/2$. An alternative method [Dol18] is to reformulate the finite-difference scheme into a larger system, such that the $(d+1)$-dimensional tensor $\mathsf{p} \in \mathbb{R}^{n_1 \times \cdots \times n_d \times n_T}$ is formed. Writing (5.18) for $j = 1, ..., n_t$ and using Kronecker product, one can derive a multilinear system for the extended tensor $\mathsf{p} \in \mathbb{R}^{n_1 \times \cdots \times n_d \times n_T}$

$$\mathsf{M}\mathsf{p} = \mathsf{f}, \tag{5.19}$$

for

$$\mathsf{M} = \mathsf{I}_{\boldsymbol{n}} \otimes \boldsymbol{S} - (\mathsf{I}_{\boldsymbol{n}} \otimes \boldsymbol{P})(\mathsf{A} \otimes \boldsymbol{I}_T), \tag{5.20a}$$

$$\mathsf{f} = \mathsf{p}^{(0)} \otimes (\boldsymbol{S}\mathbf{1}), \tag{5.20b}$$

where $\mathbf{1} \in \mathbb{R}^{n_T}$ is the one vector and $\boldsymbol{S}, \boldsymbol{P} \in \mathbb{R}^{n_T \times n_T}$. The matrices $\boldsymbol{S}$ and $\boldsymbol{P}$ are band diagonal and are constructed from the coefficients α_i and β_i, respectively. After solving the system, the time-continuous solution is recovered by interpolation. As an example, we consider the explicit Euler method on an equidistant grid with $\Delta t = \frac{t_{\text{end}} - t_{\text{start}}}{T-1}$ and $t_j = j\Delta t$, where the only nonzero coefficients are $\alpha_0 = 1$, $\alpha_{-1} = -1$ and $\beta_0 = \Delta t$ [But16]. In this case, the finite-difference matrices read

$$\boldsymbol{S} = \begin{pmatrix} 1 & & & & & \\ -1 & 1 & & & & \\ & -1 & 1 & & & \\ & & & \ddots & & \\ & & & & -1 & 1 \end{pmatrix}, \qquad \boldsymbol{P} = \Delta t \boldsymbol{I}_{n_T}. \tag{5.21}$$

An advantage of the described procedure is that the probability mass is conserved if $\mathsf{A}^\top \mathbf{1}_{\boldsymbol{n}} = \mathbf{0}$ regardless of the step size [Dol18].

An alternative method [DK15, Dol18] is to employ a basis representation of the time dependent solution over an interval $[0, \Delta T]$, such that

$$p(\boldsymbol{x}(\boldsymbol{i}), t) = \sum_{j=1}^{n_T} \mathsf{p}_{\boldsymbol{i}j} b_j(t), \tag{5.22}$$

where $b_j(t)$ are basis functions with the inputs scaled to the interval $[0, \Delta T]$, spanning a space $\mathcal{S} = \text{span}\{b_j\}_{j=1}^{n_T}$. Some examples are the Chebyshev polynomials [MH02], Lagrange polynomials, or hat functions. Using the scalar product $\langle \cdot, \cdot \rangle$ on $\mathbb{R}^{\boldsymbol{n}} \otimes \mathcal{S}$, defined as

$$\langle \mathsf{p}(\cdot), \mathsf{q}(\cdot) \rangle = \int_0^{\Delta T} \sum_{\boldsymbol{i}} \mathsf{p}_{\boldsymbol{i}}(t) \mathsf{q}_{\boldsymbol{i}}(t) \mathrm{d}t, \tag{5.23}$$

the Galerkin formulation searches for the solution $\mathsf{p}(\cdot) \in \mathbb{R}^{n_1 \times \cdots \times n_d} \otimes \mathcal{S}$ that satisfies

$$\langle \frac{\mathrm{d}\mathsf{p}}{\mathrm{d}t}, \mathsf{q} \rangle = \langle \mathsf{A}\mathsf{p}, \mathsf{q} \rangle + (\mathsf{p}^{(0)} - \mathsf{p}(0)) \cdot \mathsf{q}(0), \quad \forall \mathsf{q} \in \mathbb{R}^{n_1 \times \cdots \times n_d} \otimes \mathcal{S}, \tag{5.24}$$

where the term $(\mathsf{p}^{(0)} - \mathsf{p}(0)) \cdot \mathsf{q}(0)$ enforces the initial condition. Choosing as the test function $\mathsf{q}^{(\boldsymbol{i}m)}(t)_{\boldsymbol{l}} = \delta_{\boldsymbol{i}}^{\boldsymbol{l}} b_m(t)$, the following discrete problem is obtained:

$$\begin{aligned} \int_0^{\Delta T} \sum_{\boldsymbol{l},k} \mathsf{p}_{\boldsymbol{l}k} b_k'(t) b_m(t) \delta_{\boldsymbol{l}}^{\boldsymbol{i}} \mathrm{d}t = \int_0^{\Delta T} \sum_{\boldsymbol{l},k} \sum_{\boldsymbol{j}} \mathsf{A}_{\boldsymbol{l},\boldsymbol{j}} \mathsf{p}_{\boldsymbol{j}k} b_k(t) b_m(t) \delta_{\boldsymbol{l}}^{\boldsymbol{i}} \mathrm{d}t + \\ \sum_{\boldsymbol{l}} \left(\mathsf{p}_{\boldsymbol{l}}^{(0)} \delta_{\boldsymbol{l}}^{\boldsymbol{i}} b_m(0) - \sum_k \mathsf{p}_{\boldsymbol{l}k} b_k(0) \delta_{\boldsymbol{l}}^{\boldsymbol{i}} b_m(0) \right), \quad \forall \boldsymbol{i}, m. \end{aligned} \tag{5.25}$$

Using the linearity of the integral and introducing the stiffness matrix $\boldsymbol{S} \in \mathbb{R}^{n_T \times n_T}$ and the mass matrices $\boldsymbol{P} \in \mathbb{R}^{n_T \times n_T}$ as well as $\boldsymbol{V} \in \mathbb{R}^{n_T \times n_T}$, $\boldsymbol{v} \in \mathbb{R}^{n_T}$, such that

$$S_{ij} = \int_0^{\Delta T} b_i(t) b_j'(t) \mathrm{d}t, \quad P_{ij} = \int_0^{\Delta T} b_i(t) b_j(t) \mathrm{d}t, \tag{5.26a}$$

$$V_{ij} = b_i(0) b_j(0), \quad v_i = b_i(0), \tag{5.26b}$$

the relation from (5.25) is written as

$$\sum_{\boldsymbol{l},k} \delta_{\boldsymbol{l}}^{\boldsymbol{i}} S_{mk} \mathsf{p}_{\boldsymbol{l}k} + \sum_{\boldsymbol{l},k} \delta_{\boldsymbol{l}}^{\boldsymbol{i}} V_{mk} \mathsf{p}_{\boldsymbol{l}k} - \sum_{\boldsymbol{j},k} \left(\sum_{\boldsymbol{l},s} \delta_{\boldsymbol{l}}^{\boldsymbol{i}} P_{ms} \mathsf{A}_{\boldsymbol{l},\boldsymbol{j}} \delta_s^k \right) \mathsf{p}_{\boldsymbol{j}k} = \mathsf{p}_{\boldsymbol{i}}^{(0)} v_m, \quad \forall \boldsymbol{i}, m. \tag{5.27}$$

With the Kronecker product, we can write the equation as a system $\mathbf{M}\mathbf{p} = \mathbf{f}$ with the tensor operator and right-hand side

$$\mathbf{M} = \mathbf{I}_{\boldsymbol{n}} \otimes (\boldsymbol{S} + \boldsymbol{V}) - (\mathbf{I}_{\boldsymbol{n}} \otimes \boldsymbol{P})(\mathbf{A} \otimes \boldsymbol{I}_{n_T}), \tag{5.28a}$$

$$\mathbf{f} = \mathbf{p}^{(0)} \otimes \boldsymbol{v}. \tag{5.28b}$$

Similar to the previous case, the system can then be solved as described in Section 5.3. In order to avoid a rank increase when simulating over long periods of time, the time interval can be divided into smaller subintervals and the presented method can be applied on smaller subdomains. The initial condition of one subinterval is the end state of the previous one. In the literature, this is also known as the Discontinuous Galerkin method [DHT81]. For a constant subinterval length and if the solution is smooth, the convergence is exponential in n_T [Tre00], that is

$$\sup_{t \in [0, \Delta T]} ||\mathbf{p}(t) - \bar{\mathbf{p}}(t)||_F \leqslant C \exp(-n_T), \tag{5.29}$$

where $\bar{\mathbf{p}}$ is the exact solution and $C > 0$ a constant.

One further issue is controlling the error during the timesteps. One error indicator consists in evaluating the norm of the residual of the system (5.20b) for an enriched basis [Dol18]:

$$\varepsilon(n_T, \Delta T) = ||\mathbf{M}^{\text{enr}} \mathbf{Q} \mathbf{p} - \mathbf{f}^{\text{enr}}||_F, \tag{5.30}$$

where $\mathbf{M}^{\text{enr}}$ and $\mathbf{f}^{\text{enr}}$ are constructed using (5.28a) and (5.28b) for an enriched basis with $n_T^{\text{enr}} = 2n_T$. The tensor operator $\mathbf{Q} \in \mathbb{R}^{(n_1 \times \cdots \times n_d \times 2n_T) \times (n_1 \times \cdots \times n_d \times n_T)}$ interpolates the coarse grid solution on a finer basis with $n_T^{\text{enr}} = 2n_T$. This error estimate can be used to modify the subdomain length if $\varepsilon(n_T, \Delta T)$ is larger than a prescribed value ε_{tol} [Dol18], such that

$$\Delta T' = \left(\frac{\varepsilon_{\text{tol}}}{\varepsilon(n_T, \Delta T)} \right)^{\frac{1}{n_T}} \Delta T, \tag{5.31}$$

where $\Delta T'$ is the length of the modified subdomain. Since the AMEn method is used to obtain the solution, the relative tolerance of the solver acts like a lower bound for the error. This fact will be shown in the numerical results section.

A significant speedup of the presented CME solver is obtained when using the QTT format. The CME generator as well as the initial condition tensor and the observation operator can be reshaped as described in Section 3.2.3. The resulting quantized tensors are passes to the AMEn solver and the solution of the system is obtained in the QTT format as well. Reducing the mode sizes while increasing the number of dimensions has proven to be an effective way of further decreasing the computational complexity of the CME solver [KKNS14, DK15, IWL+21]. In [DKO12], it has been shown that the storage requirement scales with $\mathcal{O}(d \log_2 N)$ under the assumption that the TT rank remains bounded.

5.3.3. Parameter dependent CME

In the following we consider a parameter dependent reaction network. The n_p parameters are concatenated into a vector $\boldsymbol{\theta} \in \mathbb{R}^{n_p}$ and they are assumed to belong to the tensor-product space $\Xi = [\theta_1^{\min}, \theta_1^{\max}] \times \cdots \times [\theta_{n_p}^{\min}, \theta_{n_p}^{\max}]$. Parameters can be the reaction rates, or they can be part of the propensity functions in some cases. The topology of the network (number of reaction and stoichiometric vectors) is considered to be fixed. The parameter dependent CME can be solved in order to obtain the time dependent PMF as function of $\boldsymbol{\theta}$

$$\frac{\mathrm{d}\mathbf{p}(\boldsymbol{\theta})}{\mathrm{d}t} = \mathbf{A}\left(\boldsymbol{\theta}\right)\mathbf{p}(\boldsymbol{\theta}). \tag{5.32}$$

In this framework, a parameter dependence of the initial condition can also be easily handled.

Starting from a fixed initial condition $\mathsf{p}_{\boldsymbol{i}}(t_0, \boldsymbol{\theta}) = p_{t_0}(\boldsymbol{x}(\boldsymbol{i}))$ and solving the CME for every parameter $\boldsymbol{\theta} \in \Xi$ yields the conditional $\mathsf{p}_{\boldsymbol{i}}(t, \boldsymbol{\theta}) = p_t(\boldsymbol{x}(\boldsymbol{i})|\boldsymbol{\theta})$. This holds as well for the case when the initial condition is conditioned on the parameter $\mathsf{p}_{\boldsymbol{i}}(t_0, \boldsymbol{\theta}) = p_0(\boldsymbol{x}|\boldsymbol{\theta})$. If the parameter is treated as a RV and the CME is solved starting from $p_0(\cdot, \boldsymbol{\theta})$ for every $\boldsymbol{\theta} \in \Xi$, then the result is the joint PDF $p_t(\boldsymbol{x}, \boldsymbol{\theta})$ over the combined state-parameter space.

In order to discretize the dependence on the continuous parameter vectors, a basis expansion is used to construct an approximation of the joint PDF, such that

$$p_t\left(\boldsymbol{x}(\boldsymbol{i}), \boldsymbol{\theta}\right) \approx \sum_{\boldsymbol{l}} \mathsf{p}_{\boldsymbol{i}\boldsymbol{l}}(t) L_{\boldsymbol{l}}(\boldsymbol{\theta}), \tag{5.33}$$

where $\mathbf{p} \in \mathbb{R}^{n_1 \times \cdots \times n_d \times \ell_1 \times \cdots \times \ell_{n_p}}$ is the tensor of DoFs and $\{L_{\boldsymbol{l}}\}_{\boldsymbol{l}=\mathbf{1}}^{\boldsymbol{\ell}}$ is a tensor-product basis $L_{\boldsymbol{l}}(\boldsymbol{\theta}) = L^{(1)}(\theta_1) \cdots L^{(n_p)}(\theta_{n_p})$. The solution at a given time point t belongs therefore in the discrete space $\mathbb{R}^{n_1 \times \cdots \times n_d} \otimes \mathcal{P}$, where $\mathcal{P} = \mathrm{span}\{L_{\boldsymbol{l}} \ : \ l_k = 1, ..., \ell_k, k = 1, ..., n_p\}$. There exist several options for the basis, however, B-spline basis functions are used in this work regarding the parameter dependence [dB78]. Reason for that is the compact support and the locality of B-splines.

Different ways to retrieve the DoFs can be found in the literature. The first is collocation based and implies solving the problem over a finite parameter grid and computing the approximation using interpolation [BNT07]. The second approach is to use the Galerkin formulation in order to obtain a linear system for the DoFs. In this work, the Galerkin approach is chosen. We choose the test functions $\mathbf{q}(\boldsymbol{\theta})$ from the same space, which yields the formulation

$$\left\langle \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t}, \mathbf{q} \right\rangle = \left\langle \mathbf{A}\mathbf{p}, \mathbf{q} \right\rangle, \quad \forall \mathbf{q} \in \mathbb{R}^{n_1 \times \cdots \times n_d} \otimes \mathcal{P}, \tag{5.34}$$

where $\langle \cdot, \cdot \rangle$ is the inner product of the space $\mathbb{R}^{n_1 \times \cdots \times n_d} \otimes \mathcal{P}$, defined as

$$\left\langle \mathbf{p}(\cdot), \mathbf{q}(\cdot) \right\rangle = \int_{\Xi} \sum_{\boldsymbol{i}} \mathsf{p}_{\boldsymbol{i}}(\boldsymbol{\theta}) \mathsf{q}_{\boldsymbol{i}}(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}. \tag{5.35}$$

Testing with the basis elements $\mathsf{q}_{\boldsymbol{j}}^{(\boldsymbol{i},\boldsymbol{m})}(\boldsymbol{\theta}) = \delta_{\boldsymbol{j}}^{\boldsymbol{i}} L_{\boldsymbol{m}}(\boldsymbol{\theta})$, one obtains

$$\int_{\Xi} \sum_{\boldsymbol{j},\boldsymbol{l}} \frac{\mathrm{d}\mathsf{p}_{\boldsymbol{j}\boldsymbol{l}}}{\mathrm{d}t} L_{\boldsymbol{l}}(\boldsymbol{\theta}) \delta_{\boldsymbol{j}}^{\boldsymbol{i}} L_{\boldsymbol{m}}(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} = \int_{\Xi} \sum_{\boldsymbol{j},\boldsymbol{l}} \sum_{\boldsymbol{s}} \mathsf{A}_{\boldsymbol{j},\boldsymbol{s}}(\boldsymbol{\theta}) \mathsf{p}_{\boldsymbol{s}\boldsymbol{l}} L_{\boldsymbol{l}}(\boldsymbol{\theta}) \delta_{\boldsymbol{j}}^{\boldsymbol{i}} L_{\boldsymbol{m}}(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}, \quad \forall \boldsymbol{i}, \boldsymbol{m}. \tag{5.36}$$

The multilinear system

$$\mathbf{M} \frac{\mathrm{d}\mathbf{p}(\boldsymbol{\theta})}{\mathrm{d}t} = \mathbf{K}\mathbf{p}, \tag{5.37}$$

is then derived, with the mass and stiffness operators

$$\mathsf{M}_{\boldsymbol{im},\boldsymbol{jl}} = \delta^{\boldsymbol{i}}_{\boldsymbol{j}} \int_{\Xi} L_{\boldsymbol{l}}(\boldsymbol{\theta}) L_{\boldsymbol{m}}(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}, \tag{5.38}$$

$$\mathsf{K}_{\boldsymbol{im},\boldsymbol{sl}} = \int_{\Xi} \mathsf{A}_{\boldsymbol{i},\boldsymbol{s}}(\boldsymbol{\theta}) L_{\boldsymbol{l}}(\boldsymbol{\theta}) L_{\boldsymbol{m}}(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}. \tag{5.39}$$

The mass tensor-operator has TT rank $\boldsymbol{r} = \mathbf{1}$ and can be compactly written using the Kronecker product as

$$\mathbf{M} = \mathbf{I}_{\boldsymbol{n}} \otimes \boldsymbol{M}^{(1)} \otimes \cdots \otimes \boldsymbol{M}^{(n_p)}, \tag{5.40}$$

where $\boldsymbol{M}^{(k)} \in \mathbb{R}^{\ell_k \times \ell_k}, k = 1, ..., n_p$, are the mass matrices of the univariate bases and $\mathbf{I}_{\boldsymbol{n}}$ is the identity tensor-operator. Its inverse is also a rank-$\mathbf{1}$ TT-operator given by

$$\mathbf{M}^{-1} = \mathbf{I}_{\boldsymbol{n}} \otimes \left(\boldsymbol{M}^{(1)}\right)^{-1} \otimes \cdots \otimes \left(\boldsymbol{M}^{(n_p)}\right)^{-1}. \tag{5.41}$$

In the case of the stiffness matrix, the rank-1 structure is lost due to the dependence of the CME generator on the parameters. Therefore, a tensor product quadrature grid $\Theta = \left\{\theta^{\mathrm{Q},1}_{i_1}\right\}_{i_1=1}^{q_1} \times \left\{\theta^{\mathrm{Q},2}_{i_2}\right\}_{i_2=1}^{q_2} \times \cdots \times \left\{\theta^{\mathrm{Q},n_p}_{i_{n_p}}\right\}_{i_{n_p}=1}^{q_{n_p}}$ is constructed to perform the integration

$$\mathsf{K}_{\boldsymbol{im},\boldsymbol{sl}} \approx \sum_{\boldsymbol{o}} \mathsf{w}_{\boldsymbol{o}} \bar{\mathsf{A}}_{\boldsymbol{io},\boldsymbol{so}} L_{\boldsymbol{m}}(\boldsymbol{\theta}^{\mathrm{Q}}_{\boldsymbol{o}}) L_{\boldsymbol{l}}(\boldsymbol{\theta}^{\mathrm{Q}}_{\boldsymbol{o}}), \tag{5.42}$$

where $\mathsf{w}_{\boldsymbol{o}} = w^{(1)}_{o_1} \cdots w^{(n_p)}_{o_{n_p}}$ is the rank-1 weight tensor and $\bar{\mathsf{A}}_{\boldsymbol{ik},\boldsymbol{jl}} = \mathsf{A}_{\boldsymbol{i},\boldsymbol{j}}\left(\boldsymbol{\theta}_{\boldsymbol{k}}\right) \delta^{\boldsymbol{k}}_{\boldsymbol{l}}$. If the CME generator can be evaluated in the TT format for the constructed quadrature grid, then the TT cores of $\mathbf{K}$ are given as

$$\mathsf{g}^{(\mathbf{K},k)}_{s_{k-1} i_k j_k s_k} = \begin{cases} \mathsf{g}^{(\bar{\mathbf{A}},k)}_{s_{k-1} i_k j_k s_k}, & k \leqslant d, \\ \sum_{\alpha} \mathsf{g}^{(\bar{\mathbf{A}},k)}_{s_{k-1} \alpha \alpha s_k} w^{(k-d)}_{\alpha} L_{i_k}(\boldsymbol{\theta}^{\mathrm{Q},k-d}_{\alpha}) L_{j_k}(\boldsymbol{\theta}^{\mathrm{Q},k-d}_{\alpha}), & \text{otherwise}, \end{cases} \tag{5.43}$$

where $\{\mathbf{g}^{(\mathbf{K},k)}\}_k$ and $\{\mathbf{g}^{(\bar{\mathbf{A}},k)}\}_k$ are the TT cores of $\mathbf{K}$ and $\bar{\mathbf{A}}$, respectively.

In the case where the parameters are the reaction rates, i.e., $\boldsymbol{\theta} = (c_1, c_2, \dots)$, the construction of the tensor $\bar{\mathbf{A}}$ can be easily done using the Kronecker product, such that

$$\begin{aligned} \bar{\mathbf{A}} = \mathbf{A}^{(1)} \otimes \left(\mathrm{diag}\left(\theta^{\mathrm{Q},1}_1, ..., \theta^{\mathrm{Q},1}_{q_1}\right) \otimes \mathbf{I}_{\ell_2} \otimes \mathbf{I}_{\ell_3} \otimes \cdots\right) + \\ \mathbf{A}^{(2)} \otimes \left(\mathbf{I}_{\ell_1} \otimes \mathrm{diag}\left(\theta^{\mathrm{Q},2}_1, ..., \theta^{\mathrm{Q},2}_{q_2}\right) \otimes \mathbf{I}_{\ell_3} \otimes \cdots\right) + ..., \end{aligned} \tag{5.44}$$

where $\mathbf{A}^{(m)} \in \mathbb{R}^{(n_1 \times \cdots \times n_d) \times (n_1 \times \cdots \times n_d)}$ is the CME generator for the m-th reaction if the reaction rate is assumed to be one.

In some special cases, the propensity does no longer have a decomposable format and the parameter dependence can affect the propensity function and not only the reaction rate. In this case, equation (5.44) can no longer be used to construct the extended operator, and the process of obtaining a parameter dependent generator is similar to the one presented in Section 5.3.1. We consider an individual reaction m with its corresponding parameter dependent propensity function $\alpha_m(\boldsymbol{x}, \boldsymbol{\theta})$ and stoichiometric vector $\boldsymbol{\nu}^{(m)}$. Using the cross-interpolation in the TT format, one can obtain a low-rank representation of the tensor as

$$\mathsf{a}_{\boldsymbol{ij}} = \alpha_m(\boldsymbol{x}(\boldsymbol{i}), \theta^{\mathrm{Q},1}_{j_1}, ..., \theta^{\mathrm{Q},n_p}_{j_{n_p}}). \tag{5.45}$$

Performing a split as in (5.15), the low-rank representation in terms of TT cores of the two operators is

$$\mathrm{g}^{(\mathbf{C},k)}_{s_{k-1}i_kj_ks_k} = \begin{cases} \mathrm{g}^{(\mathbf{a},k)}_{s_{k-1}i_ks_k}\delta^{j_k}_{i_k}\mathbb{I}_{n_k}(x_k(j_k)+\nu_k^{(m)}), & k \leqslant d, \\ \mathrm{g}^{(\mathbf{a},k)}\delta^{j_k}_{i_k}, & \text{otherwise}, \end{cases} \tag{5.46a}$$

$$\mathrm{g}^{(\mathbf{B},k)}_{s_{k-1}i_kj_ks_k} = \begin{cases} \mathrm{g}^{(\mathbf{a},k)}_{s_{k-1}i_ks_k}\delta^{j_k}_{i_k-\nu_k}\mathbb{I}_{n_k}(x_k(j_k)+\nu_k^{(m)}), & k \leqslant d, \\ \mathrm{g}^{(\mathbf{a},k)}\delta^{j_k}_{i_k}, & \text{otherwise}, \end{cases} \tag{5.46b}$$

where $\mathbf{g}^{(\mathbf{a},k)}$ are the TT cores of the low-rank decomposition from (5.45). This way, propensities that are no longer rank-1 decomposable can also be used to build the generator in the TT format directly.

5.4. Bayesian Inference for the Chemical Master Equation with Parameter Dependencies

With the TT-CME solver presented, we now proceed to performing Bayesian inference tasks on systems governed by the CME. Due to the fast linear algebra operations in the TT format, computing marginals as well as performing multiplications of high dimensional objects becomes computationally affordable. Two cases are presented in the following: state filtering/smoothing and parameter inference. As will be mentioned, both are instances of performing inference on HMMs. In both cases, the TT-CME solver is used to compute the state transition on the HMM.

5.4.1. Filtering and smoothing in the TT format

One inference task that falls under the category of inverse problems is the estimation of the system state given observations. An example would be the reconstruction of the gene population dynamics given indirect measurements via a fluorescent reporter protein. We consider N_o indirect observations $\{\hat{\boldsymbol{y}}^{(j)}\}_{j=1}^{N_\mathrm{o}}$ of the trajectory $\boldsymbol{x}(t), t \in [0, t_{N_\mathrm{o}}]$, at discrete time points $t_1 < t_2 < \cdots < t_{N_\mathrm{o}}$. It is assumed that $\boldsymbol{x}(t), t \in [0, t_{N_\mathrm{o}}]$, is a realization of a jump process $\{\boldsymbol{X}(t)\}_{t\geqslant 0}$ whose time dependent PMF fulfills the CME. The observed quantities are assumed to be realizations of the RVs $\{\boldsymbol{Y}^{(j)}\}_{j=1}^{N_\mathrm{o}}$, which are connected to the underlying state. The RVs $\left\{\boldsymbol{Y}^{(j)}\right\}_{j=1}^{N_\mathrm{o}}$ are assumed to be mutually independent and for every $j = 1, ..., N_\mathrm{o}$, the RV $\boldsymbol{Y}^{(j)}$ is dependent only on the state RV $\boldsymbol{X}(t_j)$, with a known conditional PDF $p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x})$. Additional dependence on the time point can be added in the observation model and the presented framework, however, to simplify the notation, the time dependence of $p_{\boldsymbol{Y}|\boldsymbol{X}}$ will be omitted. Some popular examples are additive Gaussian or multiplicative lognormal noise, but the presented method is not restricted to them.

The following derivation of the time evolution of the smoothed PDF is based on [HPZ+16]. The conditional PMF $\mathbb{P}(\boldsymbol{X}(t) = \boldsymbol{x}|\boldsymbol{Y}^{(1)} = \hat{\boldsymbol{y}}^{(1)}, ..., \boldsymbol{Y}^{(j)} = \hat{\boldsymbol{y}}^{(j)})$ for $j = \max\{k \in \mathbb{N}|t_k < t\}$ can be obtained by solving the CME while incorporating the observations using the Bayes rule [HPZ+16], i.e., it satisfies the unconditional CME

$$\frac{\mathrm{d}p_t(\boldsymbol{x})}{\mathrm{d}t} = \sum_{m=1}^{M}\left\{\alpha_m(\boldsymbol{x}-\boldsymbol{\nu}^{(m)})p_t(\boldsymbol{x}-\boldsymbol{\nu}^{(m)}) - \alpha_m(\boldsymbol{x})\,p_t(\boldsymbol{x})\right\}, \tag{5.47a}$$

together with the reset conditions

$$p_{t_j}(\boldsymbol{x}) = \frac{1}{Z_j}p_{t_j^-}(\boldsymbol{x})p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}^{(j)}|\boldsymbol{x}), \quad Z_j = \sum_{\boldsymbol{x}} p_{t_j^-}(\boldsymbol{x})p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}^{(j)}|\boldsymbol{x}), \tag{5.47b}$$

where $p_{t_j^-}(\boldsymbol{x}) = \lim_{t \to t_j, t<t_j} p_t(\boldsymbol{x})$ is the PMF before the observation is taken into account. As shown in [HPZ+16], the PMF $\tilde{p}_t(\boldsymbol{x}) = \mathbb{P}(\boldsymbol{X}(t) = \boldsymbol{x} | \boldsymbol{Y}^{(1)} = \hat{\boldsymbol{y}}^{(1)}, ..., \boldsymbol{Y}^{(N_o)} = \hat{\boldsymbol{y}}^{(N_o)})$ of the smoothed distribution, for which all the measurements are taken into account, has the form

$$\tilde{p}_t(\boldsymbol{x}) = p_t(\boldsymbol{x}) \beta_t(\boldsymbol{x}), \tag{5.48}$$

where $\beta_t(\boldsymbol{x})$ satisfies the equation

$$\frac{\mathrm{d}\beta_t(\boldsymbol{x})}{\mathrm{d}t} = \sum_{m=1}^{M} \left\{ \alpha_m(\boldsymbol{x}) \beta_t(\boldsymbol{x}) - \alpha_m(\boldsymbol{x}) \beta_t(\boldsymbol{x} + \boldsymbol{\nu}^{(m)}) \right\}, \tag{5.49a}$$

with the terminal value $\beta(\boldsymbol{x})_{t_{N_o}} = 1$ and reset conditions

$$\beta_{t_j^-}(\boldsymbol{x}) = \frac{1}{Z_j} p_{t_j}(\boldsymbol{x}) p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}^{(j)} | \boldsymbol{x}). \tag{5.50}$$

Equation (5.49a), also called the backward master equation, is just a CME with the generator $-\mathbf{A}^\top$. Its solution can be interpreted as the density of the joint observation RVs following after the time point t

$$\beta_t(\boldsymbol{x}) \propto p_{\boldsymbol{Y}^{(\geqslant j)}|\boldsymbol{X}(t)}(\hat{\boldsymbol{y}}^{(j)}, ..., \hat{\boldsymbol{y}}^{(N_o)} | \boldsymbol{X}(t) = \boldsymbol{x}), \tag{5.51}$$

where $j = \min\{k \in \mathbb{N} | t_k > t\}$. The main result of [HPZ+16] is the derivation of an equation for the PMF $\tilde{p}_t(\boldsymbol{x})$ which shall satisfy

$$\frac{\mathrm{d}\tilde{p}_t(\boldsymbol{x})}{\mathrm{d}t} = \sum_{m=1}^{M} \left\{ \tilde{\alpha}_m(\boldsymbol{x} - \boldsymbol{\nu}^{(m)}, t) \tilde{p}_t(\boldsymbol{x} - \boldsymbol{\nu}^{(m)}) - \tilde{\alpha}\left(\boldsymbol{x}, t\right) \tilde{p}_t\left(\boldsymbol{x}\right) \right\}, \tag{5.52}$$

where the propensities are time dependent and have the form

$$\tilde{\alpha}_m(\boldsymbol{x}, t) = \alpha_m(\boldsymbol{x}) \frac{\beta(\boldsymbol{x} + \boldsymbol{\nu}^{(m)}, t)}{\beta(\boldsymbol{x}, t)}. \tag{5.53}$$

On a discrete-time level, we deal with a HMM where the hidden states are the RVs $\{\boldsymbol{X}(t_j)\}_{j=1}^{N_o}$. One well-known method to compute the posterior marginal distributions of all hidden states of a HMM is the forward-backward algorithm, also referred to as belief propagation in a HMM [LJ09]. Having a TT solver for the CME and fast multilinear algebra routines, all operations can be performed in the low-rank format (see Algorithm 8). The forward message, representing the solution $p_{t_j} = \mathbb{P}(\boldsymbol{X}(t_j) = \boldsymbol{x} | \boldsymbol{Y}^{(1)} = \hat{\boldsymbol{y}}^{(1)}, \ldots, \boldsymbol{Y}^{(j)} = \hat{\boldsymbol{y}}^{(j)})$ of the CME (5.47a), is denoted by the tensor $\mathbf{a}^{(j)} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. The CME is solved on the interval $[t_j, t_{j+1})$ to obtain a prediction $\mathbf{a}^{(j \to j+1)}$. Using the observation model, the tensor $\mathbf{p}^{\text{obs}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, $\mathsf{p}_{\boldsymbol{i}}^{\text{obs}} = p_{\boldsymbol{Y}|\boldsymbol{X}}(\hat{\boldsymbol{y}}^{(j)} | \boldsymbol{x}(\boldsymbol{i}))$ is assembled by using the TT cross-approximation or, in some cases, by directly constructing a TT representation. The latter, for example, is applicable when the observation model can be factorized into univariate PDFs. In this case, $\mathbf{p}^{\text{obs}}$ is a rank-1 tensor. The IC (reset condition) is calculated as the elementwise multiplication $\mathbf{a}^{(j+1)} = \mathbf{p}^{\text{obs}} \odot \mathbf{a}^{(j \to j+1)}$. In case of the backward pass, the message represented by the tensor $\mathbf{b}^{(j)}$ corresponds to $\beta(\boldsymbol{x})_{t_j}$, and is propagated by solving the CME with the transposed operator $\mathbf{A}^\top$, starting from the IC $\mathbf{b}^{(j+1)} \odot \mathbf{p}^{\text{obs}}$ to obtain $\mathbf{b}^{(j+1 \to j)}$. Having all forward and backward messages, the smoothing posterior is obtained by multiplying the messages together, such that $p_{\boldsymbol{X}(t_j)|\boldsymbol{Y}^{(\leqslant N_o)}}(\boldsymbol{x}(\boldsymbol{i}) | \hat{\boldsymbol{y}}^{(1)}, ..., \boldsymbol{y}^{(N_o)}) = Z^{-1} \mathsf{a}_{\boldsymbol{i}}^{(j)} \mathsf{b}_{\boldsymbol{i}}^{(j)}$, $Z = \mathbf{a} \cdot \mathbf{b}$. The advantage of using the basis expansion over the time domain is that the smoothing PMF can be calculated at any time point between the observations.

Algorithm 8 Forward-backward algorithm for trajectory estimation.

Input: Sample $\{\hat{\boldsymbol{y}}^{(j)}\}_{j=0}^{N_\mathrm{o}}$, initial PMF $\mathbf{p}^{(0)}$
$\mathbf{a}^{(0)} \leftarrow \mathbf{p}^{(0)}$
for $j = 1, ..., N_\mathrm{o}$ **do**
    Solve the CME with the IC $\mathbf{a}^{(j-1)}$ to obtain $\mathbf{a}^{(j-1\to j)}$.
    Compute $\mathbf{p}^{\mathrm{obs}}$ for $\hat{\boldsymbol{y}}^{(j)}$ in the TT format.
    $\mathbf{a}^{(j)} \leftarrow \mathbf{p}^{\mathrm{obs}}_{\boldsymbol{i}} \odot \mathbf{a}^{(j-1\to j)}$
$\mathbf{b}^{(N_\mathrm{obs})} \leftarrow \mathbf{1}$
for $j = N_\mathrm{obs} - 1, \ldots, 0$ **do**
    Compute $\mathbf{p}^{\mathrm{obs}}$ for $\hat{\boldsymbol{y}}^{(j+1)}$ in the TT format.
    Solve the CME with the generator $\mathbf{A}^\top$ and IC $Z^{-1}\mathbf{b}^{(j+1)} \odot \mathbf{p}^{\mathrm{obs}}$ to obtain $\mathbf{b}^{(j+1\to j)}$.
    $\mathbf{b}^{(j)} \leftarrow \mathbf{p}^{\mathrm{obs}} \odot \mathbf{b}^{(j+1\to j)}$
for $j = 0, ..., N_\mathrm{obs}$ **do**
    $\mathbf{p}^{(j)} \leftarrow Z^{-1}\mathbf{a}^{(j)} \odot \mathbf{b}^{(j)}$
Output: $\mathbf{p}^{(j)}$ for $j = 0, ..., N_\mathrm{obs}$

5.4.2. Bayesian parameter inference in the TT format

The second Bayesian inference task considered in this chapter, is parameter identification from a trajectory observation. Compared to Section 5.4.1, where a realization of a fully known system is considered, in this section, the observation sample $\left\{\boldsymbol{y}^{(j)}\right\}_{j=1}^{N_\mathrm{o}}$ is considered to be a realization of a parameter dependent jump process $\boldsymbol{X}(t, \hat{\boldsymbol{\theta}})$, where $\hat{\boldsymbol{\theta}}$ contains the parameters governing the system. The evolution of the PMF corresponding to the parameter dependent jump process is given by the parameter dependent CME. The parameter is considered as a RV $\boldsymbol{\vartheta}$ and finding $\hat{\boldsymbol{\theta}}$ is transformed to an inference task: approximate the conditional PDF $p_{\vartheta|\boldsymbol{Y}^{(\leqslant N_\mathrm{o})}}$ given a prior distribution over the parameter space $p_{\boldsymbol{\vartheta}}$ and the observations. The assumptions on the observation model $p_{\boldsymbol{Y}|\boldsymbol{X}}$ are identical to the ones from the previous section. The parameter inference task can be cast into the previously defined filtering framework by considering the joint $\{(\boldsymbol{X}^{(j)}, \boldsymbol{\vartheta}^{(j)})\}_{j=1}^{N_\mathrm{o}}$ as latent RVs in the HMM [IWL+21]. In the time-continuous case, this corresponds to the filtering of an augmented process $\{\mathbf{X}(t), \boldsymbol{\vartheta}(t)\}_{t\geqslant 0}$, where the posterior $p_{\vartheta|\boldsymbol{Y}^{(\leqslant N_\mathrm{o})}}$ is obtained by marginalizing the state at $t = t_{N_\mathrm{o}}$. No backward pass is needed in this case.

The method is presented in Algorithm 9. The prediction step is obtained by marginalization and using the Markov property

$$p_{\boldsymbol{X}^{(j)},\boldsymbol{\vartheta}^{(j)}|\boldsymbol{Y}^{(<j)}}\left(\boldsymbol{x}^{(j)}, \boldsymbol{\theta}^{(j)}|\boldsymbol{y}^{(0)}, ..., \boldsymbol{y}^{(j-1)}\right) = \sum_{\boldsymbol{x}^{(j-1)}} \int_{\Xi} \left(p_{j|j-1}\left(\boldsymbol{x}^{(j)}, \boldsymbol{\theta}^{(j)}|\boldsymbol{x}^{(j-1)}, \boldsymbol{\theta}^{(j-1)}\right)\right.$$
$$\left. p_{\boldsymbol{X}^{(j)},\boldsymbol{\vartheta}^{(j)}|\boldsymbol{Y}^{(\leqslant j)}}\left(\boldsymbol{x}^{(j-1)}, \boldsymbol{\theta}^{(j-1)}|\boldsymbol{y}^{(1)}, ..., \boldsymbol{y}^{(j-1)}\right)\right) \mathrm{d}\boldsymbol{\theta}^{(j-1)}, \tag{5.54}$$

where the conditional PDF $p_{j|j-1}$ is given as the solution of the parameter dependent CME in the interval $[t_{j-1}, t_j)$. Marginalization implies summation and integration over the state $j-1$, which is computationally inefficient. Alternatively, the prediction step can be performed by solving the parameter dependent CME with the IC $p_{\boldsymbol{X}^{(j)},\boldsymbol{\vartheta}^{(j)}|\boldsymbol{Y}^{(\leqslant j)}}(\boldsymbol{x}^{(j-1)}, \boldsymbol{\theta}^{(j-1)}|\hat{\boldsymbol{y}}^{(1)}, ..., \hat{\boldsymbol{y}}^{(j-1)})$. On the discrete level, the posterior at step j is represented by the tensor $\mathbf{p}^{(j)} \in \mathbb{R}^{n_1\times\cdots\times n_d\times \ell_1\times\cdots\times\ell_{n_p}}$, such that

$$p_{\boldsymbol{X}^{(j)},\boldsymbol{\vartheta}^{(j)}|\boldsymbol{Y}^{(\leqslant j)}}\left(\boldsymbol{x}^{(j)}(\boldsymbol{i}), \boldsymbol{\theta}^{(j)}|\hat{\boldsymbol{y}}^{(1)}, ..., \hat{\boldsymbol{y}}^{(j)}\right) = \sum_{\boldsymbol{l}} \mathsf{p}^{(j)}_{\boldsymbol{il}} L_{\boldsymbol{l}}\left(\boldsymbol{\theta}^{(j)}\right), \quad j = 1, ..., N_\mathrm{o}, \tag{5.55}$$

where $\mathbf{p}^{(j)}$ is stored in the TT format. The TT-CME solver with the IC $\mathbf{p}^{(j)}$ is applied in order the get the DoF tensor $\mathbf{p}^{(j\to j+1)}$ of the PDF $p_{\boldsymbol{X}^{(j)},\boldsymbol{\vartheta}^{(j)}|\boldsymbol{Y}^{(<j)}}$. The update step

$$p_{\boldsymbol{X}^{(j)},\boldsymbol{\vartheta}^{(j)}|\boldsymbol{Y}^{(\leqslant j)}}\left(\boldsymbol{x}^{(j)},\boldsymbol{\theta}^{(j)}|\hat{\boldsymbol{y}}^{(1)},...,\hat{\boldsymbol{y}}^{(j)}\right)=\frac{1}{Z}p_{\boldsymbol{Y}|\boldsymbol{X}}\left(\boldsymbol{y}^{(j)}|\boldsymbol{x}^{(j)}\right)$$
$$p_{\boldsymbol{X}^{(j)},\boldsymbol{\vartheta}^{(j)}|\boldsymbol{Y}^{(<j)}}\left(\boldsymbol{x}^{(j)},\boldsymbol{\theta}^{(j)}|\hat{\boldsymbol{y}}^{(1)},...,\hat{\boldsymbol{y}}^{(j-1)}\right), \tag{5.56}$$

is accomplished in the tensor format as

$$\mathbf{p}^{(j+1)}=Z^{-1}(\mathbf{p}^{\text{obs}}\otimes\mathbf{1}_{\boldsymbol{\ell}})\odot\mathbf{p}^{(j\to j+1)},\quad Z=\mathbf{p}^{(j+1)}\cdot(\mathbf{1}_{\boldsymbol{n}}\otimes\mathbf{w}_{\boldsymbol{l}}), \tag{5.57}$$

where $\mathbf{1}_{\boldsymbol{\ell}}\in\mathbb{R}^{\ell_1\times\cdots\times\ell_{n_p}}$ and $\mathbf{1}_{\boldsymbol{n}}\in\mathbb{R}^{n_1\times\cdots\times n_d}$ are 1-tensors of appropriate sizes and $\mathbf{w}\in\mathbb{R}^{\ell_1\times\cdots\times\ell_{n_p}}$ is a rank-1 tensor containing the weights that result from integrating the parameter-space basis. Since the process is iterative, the tensor $\mathbf{p}^{(0)}$ has to be specified. Given the DoFs of the prior $p_{\boldsymbol{\vartheta}}$ and the initial PMF $\mathbf{p}(0)$, the initial tensor is $\mathbf{p}^{(0)}\leftarrow\mathbf{p}^{(0)}\odot\mathbf{p}^{\text{prior}}$.

After the last observation is used, the marginalization of the state

$$p_{\boldsymbol{\vartheta}|\boldsymbol{Y}^{(\leqslant N_\text{o})}}\left(\boldsymbol{\theta}|\hat{\boldsymbol{y}}^{(1)}...,\hat{\boldsymbol{y}}^{(j)}\right)=\sum_{\boldsymbol{x}}p_{\boldsymbol{X}^{(n_\text{o})},\boldsymbol{\vartheta}^{(n_\text{o})}|\boldsymbol{Y}^{(\leqslant N_\text{o})}}\left(\boldsymbol{x},\boldsymbol{\theta}|\hat{\boldsymbol{y}}^{(1)}...,\hat{\boldsymbol{y}}^{(j)}\right), \tag{5.58}$$

is computed on the multilinear algebra level as $\mathbf{p}^{\text{post}}=\sum_{\boldsymbol{i}}\mathbf{p}^{(N_\text{o})}_{\boldsymbol{i}:}$.

Algorithm 9 Parameter identification in the TT-CME framework.

1: **Input:** Sample $\left\{\hat{\boldsymbol{y}}^{(j)}\right\}_{j=0}^{N_\text{o}}$, initial PMF $\mathbf{p}(0)$, prior over the parameter space $\mathbf{p}^{\text{prior}}$
2: $\mathbf{p}^{(0)}\leftarrow\mathbf{p}^{(0)}\odot\mathbf{p}^{\text{prior}}$
3: **for** $j=1,...,N_\text{o}$ **do**
4: Solve the CME with $\mathbf{p}^{(j-1)}$ as initial condition to obtain the solution $\mathbf{p}^{(j-1\to j)}$.
5: Compute $\mathbf{p}^{\text{obs}}$ for $\hat{\boldsymbol{y}}^{(j)}$ in TT.
6: $\mathbf{p}^{(j+1)}=Z^{-1}(\mathbf{p}^{\text{obs}}\otimes\mathbf{1}_{\boldsymbol{\ell}})\odot\mathbf{p}^{(j\to j+1)}$ for $Z=\mathbf{p}^{(j+1)}\cdot(\mathbf{1}_{\boldsymbol{n}}\otimes\mathbf{w}_{\boldsymbol{l}})$.
7: $\mathbf{p}^{\text{post}}\leftarrow\sum_{\boldsymbol{i}}\mathbf{p}^{(N_\text{o})}_{\boldsymbol{i}:}$
8: **Output:** $\mathbf{p}^{\text{post}}$

5.5. Numerical Experiments

In the rest of the chapter, numerical investigations are performed to showcase the performance of the presented TT-CME framework. First, a convergence study is performed to assess the correctness of the solver for a simple gene expression model. A more complicated model is also investigated, namely, the so-called SEIR model [Het00]. Second, the state filtering is exemplified on the SEIR model. The remaining three examples address the parameter identification task in the TT format. All presented results have been obtained with a standard workstation. The memory requirements are low enough to fit on every modern laptop or workstation. The framework has been implemented in the `tt-cme`[1] `Python` package. For the multilinear algebraic operations in the TT format, the `torchtt`[2] package is used.

[1] `https://github.com/ion-g-ion/tt-cme`
[2] `https://github.com/ion-g-ion/torchTT`

Reaction	$\alpha_m(\boldsymbol{x})$	Rates c_i	Description
mRNA $\rightarrow \varnothing$	$c_1 x_1$	0.002	mRNA degradation
mRNA $\rightarrow$ mRNA + Protein	$c_2 x_1$	0.015	Translation
$\varnothing \rightarrow$ mRNA	c_3	0.1	Transcription
Protein $\rightarrow \varnothing$	$c_4 x_2$	0.01	Protein degradation

Table 5.1.: Reactions of the simple gene expression model.

5.5.1. Validation of the TT-based CME solver

Two-dimensional simple gene expression model

We first perform a convergence study to validate the TT-CME solver. The simple gene expression model is used [AJL+02]. Since the model is only 2d, an accurate reference solution can be easily obtained by conventional ODE solvers. In Table 5.1, the four reactions involving the two species are given, i.e., messenger ribonucleic acid (mRNA) and protein. The IC is $p_0(\boldsymbol{x}) = \delta^2_{x_1}\delta^4_{x_2}$, i.e., $\boldsymbol{x}^{(0)} = (2,4)^\top$ with probability 1. The relevant simulation interval is $[0, 1024]$ time units, which is split into 8 subintervals of length 128 time units. In the following, several solver settings are varied as part of the convergence study. As a metric, the maximum relative error between TT-CME solution and the reference is computed:

$$\epsilon_{\max} = \frac{\max_{\boldsymbol{x}} \left| p^{(\mathrm{ref})}_{t_{\mathrm{end}}}(\boldsymbol{x}) - p_{t_{\mathrm{end}}}(\boldsymbol{x}) \right|}{\max_{\boldsymbol{x}} \left| p_{t_{\mathrm{end}}}(\boldsymbol{x}) \right|}, \tag{5.59}$$

where $t_{\mathrm{end}} = 1024$ and $p^{(\mathrm{ref})}_{t_{\mathrm{end}}}(\boldsymbol{x})$ is the reference solution computed by integrating the ODE in the full format over a very fine grid.

The first numerical experiment addressed the dependency of the maximum relative error on the dimension n_T of the basis expansion (5.22), which is used for the time dependency. The relative residual of the AMEn solver was fixed to 10^{-13} during this test and no rank truncation was done. In Figure 5.1, the results are shown for three different choices of time integration schemes: Chebyshev polynomial basis and the classical finite-difference schemes such as implicit Euler and Crank-Nicolson. As expected from the theory, the convergence of the classical schemes is $\mathcal{O}(n_T^{-1})$ for the implicit Euler and $\mathcal{O}(n_T^{-2})$ for Crank-Nicolson [But16]. The convergence of the Chebyshev method is also the one expected from the theory, namely exponential [Tre00]. However, the Chebyshev polynomial method reaches stagnation after $n_T = 8$. This bottleneck is caused by the choice of the relative residual of the iterative TT solver.

A further investigation is performed to address the effect of the basis dimension and the relative residual of the AMEn solver ϵ. To this end, the simulation is run for $(\epsilon, n_T) \in \{10^{-1}, 10^{-2}, \ldots, 10^{-12}\} \times \{2, 3, \ldots, 8\}$ and the results are displayed in Figure 5.2. The stagnation of $\epsilon_{\max}$ due to the relative residual ϵ can be clearly observed in the plot. Therefore, a balance between the n_T and the relative residual has to be found and the computational cost has to be taken into consideration as well.

Four-dimensional SEIR model

In this section, a more complicated model is studied. For the simple gene expression model previously introduced, the reference solution was easily computed using a conventional ODE solver. A more complicated model is investigated in this section, namely the so-called SEIR model. The SEIR model is a 4-dimensional virus spreading model [Het00] for which the standard ODE solvers applied on full tensors result in a high computational cost. The individuals of the virus spreading model are separated into four distinct categories (species):

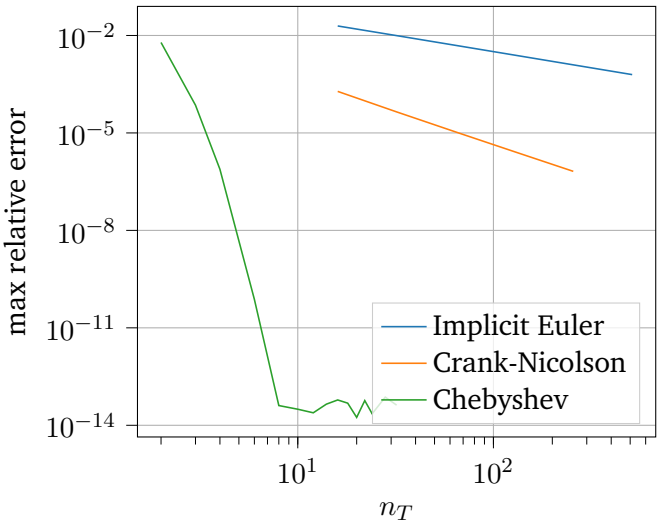


Figure 5.1.: Convergence of the TT-solver with respect to the dimension of the time basis.

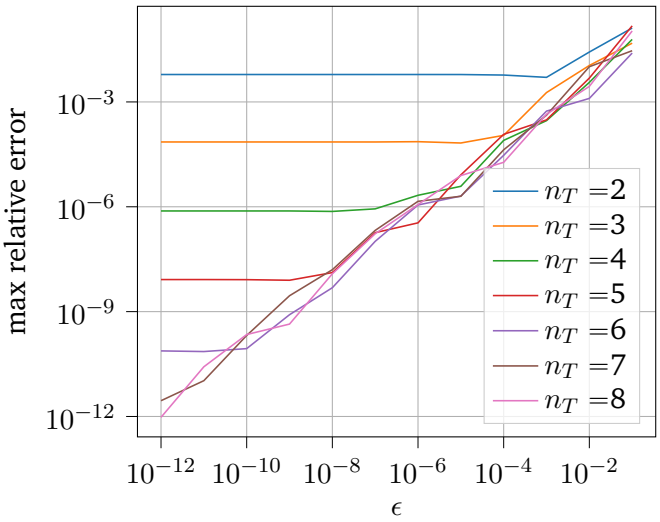


Figure 5.2.: Error versus solver accuracy for different sizes of the basis.

1. Susceptible (S), i.e., healthy individuals who are prone to getting infected.
2. Exposed (E), i.e., individuals exposed to the virus, but cannot yet spread the virus.
3. Infected (I), i.e., individuals who carry the virus and are contagious.
4. Recovered (R), i.e., healthy individuals that are immune to the virus.

In Table 5.2, the 7 reactions between the 4 categories of individuals are presented. For the numerical solution, the state space truncation is $\boldsymbol{n} = (n_1, n_2, n_3, n_4) = (128, 128, 64, 64)$ and the IC is $\boldsymbol{x}(0) = (50, 4, 0, 0)^\top$ with probability 1. A fixed subinterval length of 0.5 and a basis dimension of $n_T = 8$ are chosen in order to solve the CME in the time interval $[0, 8]$. The reference solution is computed by numerically solving the CME without the TT-decomposition for a very fine time grid.

Building the CME generator in the TT format for the given state truncation results in a TT rank of $\boldsymbol{r} = (1, 5, 6, 3, 1)$, which accounts for ≈ 2.3 MB of storage. As a comparison, the same tensor operator in

Reaction	$\alpha_m(\boldsymbol{x})$	Rate c_i	Description
$S + I \to E + I$	$c_1 x_1 x_3$	0.1	Susceptible meets infected and becomes exposed
$E \to I$	$c_2 x_2$	0.5	Exposed is infected and contagious
$I \to S$	$c_3 x_3$	1.0	Infected recovers without immunity
$S \to \varnothing$	$c_4 x_1$	0.01	Susceptible dies
$E \to \varnothing$	$c_5 x_2$	0.01	Exposed dies
$I \to R$	$c_6 x_3$	0.01	Infected recovers with immunity
$\varnothing \to S$	c_7	0.4	New susceptible is born

Table 5.2.: Reactions, propensities, and reaction rates of the SEIR model.

full format would require ≈ 2.1 GB, even if a sparse format is used. The TT representation needs in this case only 0.11% of the storage required to store all the nonzero elements of the full generator. Using the QTT format cuts the storage requirements to around 38 KB, thus resulting in a further decrease compared to the TT format. The QTT also speeds up the time domain solver, with the solution being obtained in approximately 3 min, while the solution in the TT format requires 4.4 min. The reference solution obtained with a conventional ODE integrator is computed in approximately 4 hours.

In order to illustrate the evolution of the PMF, several solution snapshots of the marginal EI distribution are shown in Figure 5.3. The pointwise error between the TT based solution and the reference is displayed in Figure 5.3f. The maximum and mean errors at the end of the simulation time $t_{\text{end}} = 8$ are

$$\epsilon_{\max} = \frac{\max_{\boldsymbol{x}} |p^{(\text{ref})}_{t_{\text{end}}}(\boldsymbol{x}) - p_{t_{\text{end}}}(\boldsymbol{x})|}{\max_{\boldsymbol{x}} |p^{(\text{ref})}_{t_{\text{end}}}(\boldsymbol{x})|} = 2.9 \cdot 10^{-5},$$

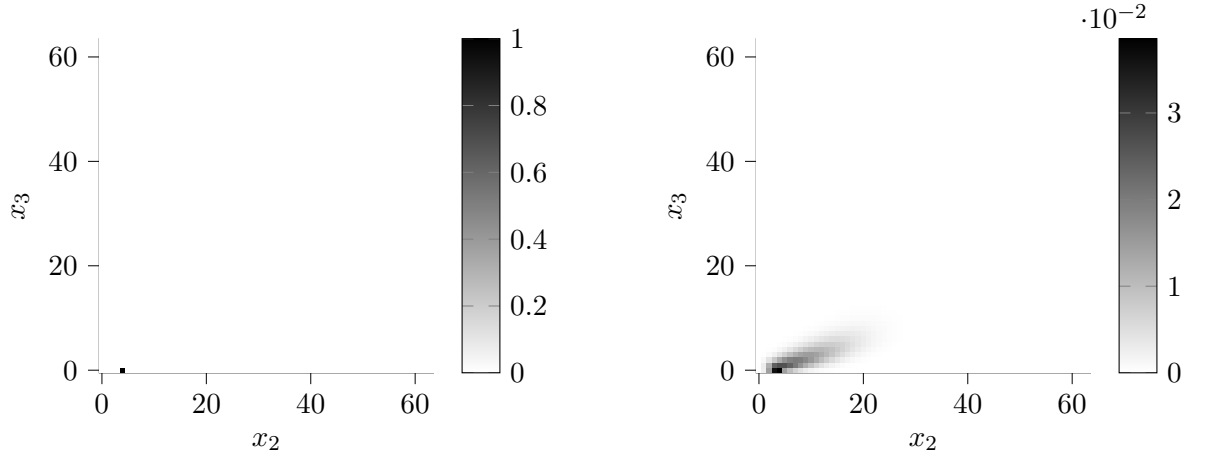
$$\epsilon_{\text{mean}} = \frac{\frac{1}{N^4} \sum_{\boldsymbol{x}} |p^{(\text{ref})}_{t_{\text{end}}}(\boldsymbol{x}) - p_{t_{\text{end}}}(\boldsymbol{x})|}{\max_{\boldsymbol{x}} |p^{(\text{ref})}_{t_{\text{end}}}(\boldsymbol{x})|} = 2.539 \cdot 10^{-9}.$$

The rank for representing the solution in the TT format at $t = t_{\text{end}}$ is $\boldsymbol{r} = (1, 46, 106, 14, 1)$, which accounts for 3.32 MB of storage (1.89 MB for QTT), that is, only 1.23% (0.70% for QTT) of the storage needed for storing the full PMF. During the TT-CME time stepping, the largest tensor stored for the basis expansion over the time domain took only 4.4 MB (2.5 MB for QTT). For this example, the TT-CME solver proves to be a considerably more efficient alternative compared to standard solvers.

The ordering S, E, I, R has been chosen based on the interactions between the species. Reordering the species has been shown to have an effect on the TT ranks of the CME operator and the solution. This issue has been addressed in [GKMS17]. The species that are highly correlated have to be close to each other in the train, otherwise the rank of the cores in between must carry the information. This is problematic since it increases the rank and thus the computational expense.

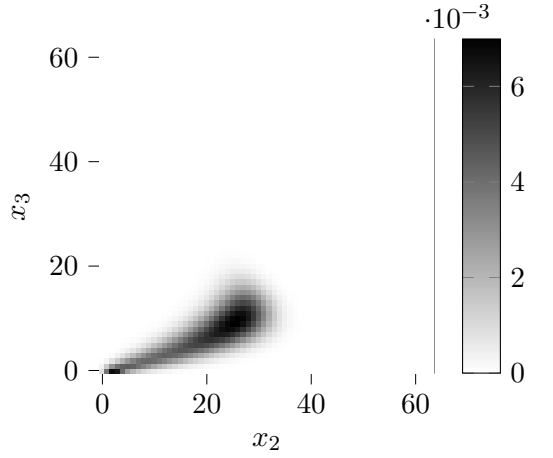
5.5.2. Filtering and smoothing

We now apply the state filtering and smoothing algorithm (Algorithm 8) presented in Section 5.4.1 on the SEIR model from Section 5.5.1. Using the SSA [Gil76], a realization of the system is computed for the time interval $[0, 10]$. From the reference trajectory, we choose $N_{\text{o}} = 33$ equidistant observations with $\Delta t = 0.3125$ (see the blue solid line in Figure 5.4). Lognormal noise is then added to the observations with variance 0.1 for S, E, and I, and 0.05 for R. In Figure 5.4, the trajectory for S, E and I is plotted (blue

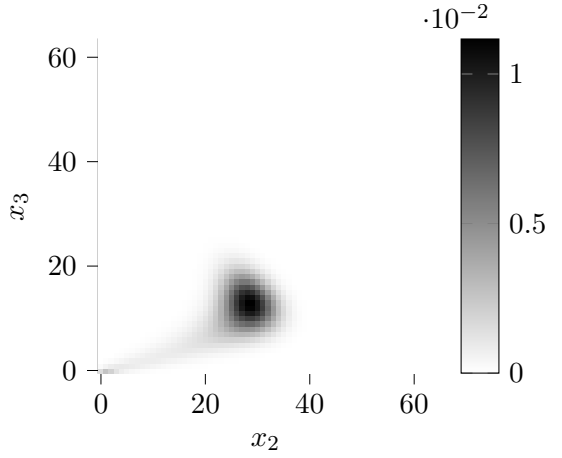


(a) $t = 0$.

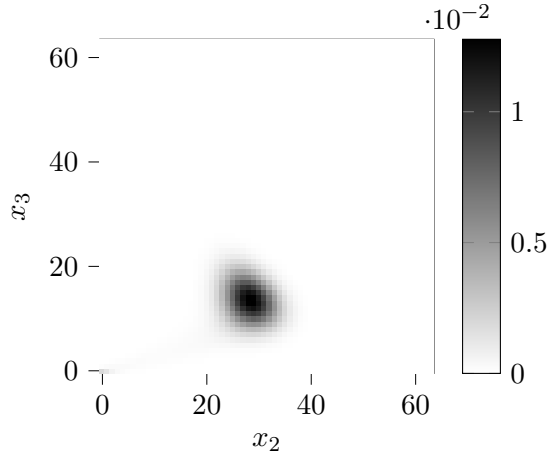
(b) $t = 2$.



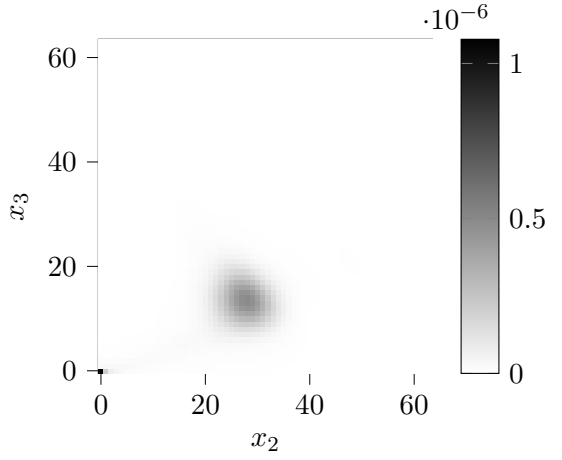
(c) $t = 4$.



(d) $t = 6$.



(e) $t = 8$.



(f) Pointwise error at $t = 8$.

Figure 5.3.: The time dependent Exposed-Infected (EI) marginal PMF for $t \in \{0, 2, 4, 6, 8\}$. In 5.3f, we compute the pointwise absolute error between the TT solution and the reference obtained by integrating the CME over a fine time grid. Figures adapted from [IWL+21].

solid line) together with the noisy observations (marked with the black $\times$ symbol). The noise model in this case represented as a tensor is

$$\mathsf{p}_{\boldsymbol{i}}^{\text{obs}}(\boldsymbol{y}) = \prod_{k=1}^{d} \frac{1}{y_k \sigma_k \sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(\log y_k - \log(x_k(i_k)+1))^2}{\sigma_k^2}\right), \tag{5.60}$$

where $\boldsymbol{y}$ is the observation and $\sigma_k, k = 1,2,3,4$ are the corresponding variances. As it can be seen the $\mathbf{p}^{\text{obs}}$ is a rank-1 tensor. If reshaped to QTT, the TT rank still remains low (maximum rank 8 and mean rank less than 2 for a truncation of $\epsilon = 10^{-12}$). The state truncation is $\boldsymbol{n} = (128, 128, 64, 32)$ and the Chebyshev differentiation scheme is used for the timestepping.

The runtime of the experiment is 10 minutes for the forward pass and 13 minutes for the backward pass (the relative residual of the AMEn solver is set to 10^{-6}). Both TT and QTT are investigated and it has been again shown that the QTT outperforms the TT in terms of computational cost (both space and time). With respect to the storage requirements, storing the PMFs in the QTT format reduces the memory usage to approximately ≈ 105 MB for the forward propagating messages and 130 MB for the backward propagating messages, which is an advantage since both type of messages have to be saved for the smoothing. Having the time dependent PMF $\mathbb{P}(\boldsymbol{X}(t) = \boldsymbol{x} | \boldsymbol{Y}^{(1)} = \hat{\boldsymbol{y}}^{(1)}, ..., \boldsymbol{Y}^{(N_\text{o})} = \hat{\boldsymbol{y}}^{(N_\text{o})})$ of the smooth distribution, we compute the first two moments in order to quantify the obtained reconstruction of the state. In Figure 5.4, the red discontinuous line is the expected value and the standard deviation is represented by the gray envelope. The moments are efficiently computed in the TT format since they are obtained using scalar product with rank-1 tensors.

For this example, applying the reset conditions in order to incorporate the observations leads to an improvement of the overall error. Moreover, the TT ranks also decreases after the reset condition is imposed. This behavior is illustrated in Figure 5.5, where the maximum TT rank of the train is represented over the simulation time. A decrease of up to 3 times is observed.

5.5.3. Bayesian parameter inference

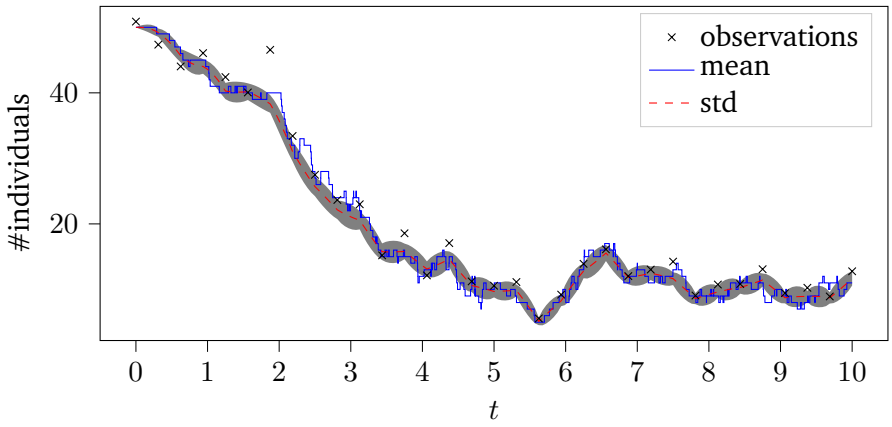
Simple gene expression model

Algorithm 9 is now applied in order to identify the four reaction rates $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4) = (c_1, c_4, c_3, c_4)$ of the simple gene expression model described by the reactions in Table 5.4. Using the SSA, a realization is drawn for the time interval $[0, 1024]$. From the reference trajectory, $N_\text{o} = 64$ equidistant observations are chosen, with $\delta t = 4$ time units. Gaussian noise with the standard deviation $\sigma = 0.5$ is added, with the noise model represented as the rank-1 tensor

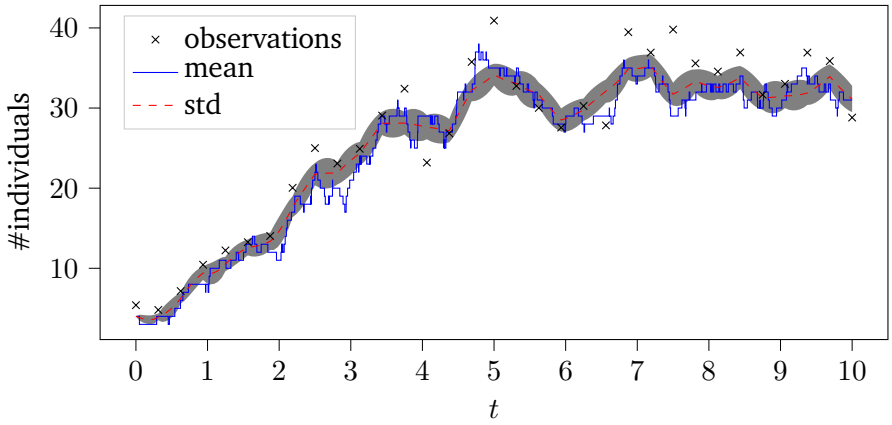
$$\mathsf{p}_{\boldsymbol{i}}^{\text{obs}}(\boldsymbol{y}) = \prod_{k=1}^{2} \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(y_k - x_k(i_k))^2}{\sigma^2}\right), \tag{5.61}$$

where $\boldsymbol{y}$ is the observation. Regarding the prior, the parameters are considered to be independent Gamma distributed. Since the model is computationally affordable even for the classical solvers, the Metropolis–Hastings (MH) algorithm [BGJM11] is used to draw a sample of size $5 \cdot 10^5$ from the posterior. To this end, the CME was solved for different parameter realizations using the built-in `Python` ODE solver. The runtime of the MH is ≈ 1.5 days, which is orders of magnitude slower than the TT based solver, as will be shown later.

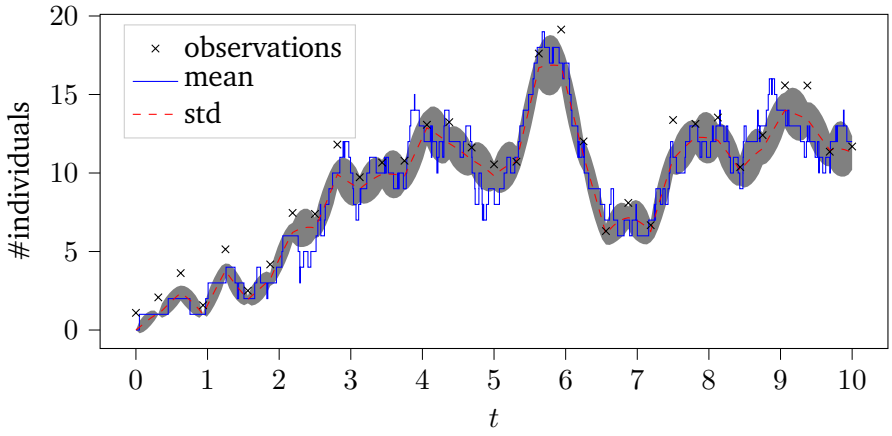
To discretize the parameter space, we use tensor product quadratic B-splines with uniform knots of equal size, i.e., $\boldsymbol{\ell} = (64, 64, 6, 4, 64)$. The parameters are truncated to $\theta_i \in [0, 6c_i], i = 1, \ldots, 4$. Regarding the time discretization, the Chebyshev scheme is used with $n_T = 8$ and a maximum subinterval size of 0.5 time



(a) Susceptible individuals.



(b) Exposed individuals.



(c) Infected individuals.

Figure 5.4.: Forward-backward algorithm applied on the SEIR model with N_o time-discrete observations. The ground truth is the solid blue line, while the noisy observations are marked with the "$\times$" symbol. The expected value of the posterior is represented by the red dashed line and the corresponding standard deviation by the gray envelope. Figure adapted from [IWL+21].

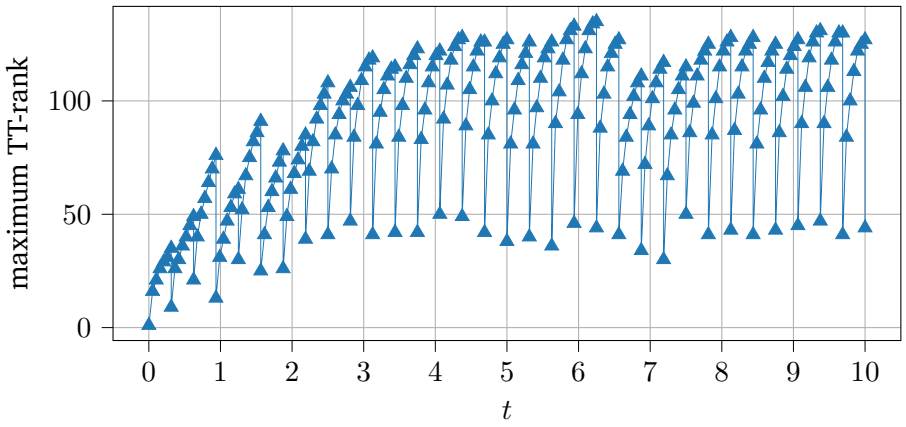


Figure 5.5.: TT ranks of the forward propagating messages (represented by the triangle markers).

units. For a state truncation of $\boldsymbol{n} = (64, 64)$, the QTT format is used to run the parameter inference. The total runtime is ≈ 21 min and the 6d tensor of mode size 64 representing the joint over the state-parameter space requires ≈ 9.2 MB of storage. If the same tensor is stored in the full format, it would require more than 500 GB. An even better compression is attained for the CME generator, requiring only ≈ 128 KB of storage. The 4d posterior cannot be directly visualized and therefore marginals $p_{\Theta_j,\Theta_i}, i, j = 1, \ldots, 4$, are computed and graphically represented in Figure 5.6. As a comparison, 1d and 2d histograms are also represented for a visual verification, where the exact parameters are marked by the red dashed lines. A visual match can be observed between the two methods. Posterior moments such as the expected value and the covariance are also computed:

$$\mathbb{E}(\boldsymbol{\vartheta}) = (0.001925, 0.01512, 0.09988, 0.01058),$$

$$\mathrm{Cov}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}) = \begin{pmatrix} 1.0351 \cdot 10^{-6} & -1.4000 \cdot 10^{-8} & 1.0560 \cdot 10^{-5} & 2.2160 \cdot 10^{-10} \\ -1.4000 \cdot 10^{-8} & 8.6669 \cdot 10^{-6} & -2.5506 \cdot 10^{-7} & 4.9247 \cdot 10^{-6} \\ 1.0560 \cdot 10^{-5} & -2.5506 \cdot 10^{-7} & 5.4189 \cdot 10^{-4} & 1.7087 \cdot 10^{-8} \\ 2.2160 \cdot 10^{-10} & 4.9247 \cdot 10^{-6} & 1.7087 \cdot 10^{-8} & 7.6244 \cdot 10^{-6} \end{pmatrix}.$$

The mean and covariance of the MH sample are:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}} = (0.001922, 0.01507, 0.09992, 0.01052),$$

$$\boldsymbol{C}_{\boldsymbol{\theta}} = \begin{pmatrix} 9.9755 \cdot 10^{-7} & -1.6217 \cdot 10^{-8} & 9.7124 \cdot 10^{-6} & 3.7631 \cdot 10^{-9} \\ -1.6217 \cdot 10^{-8} & 8.4986 \cdot 10^{-6} & -8.3867 \cdot 10^{-8} & 4.8476 \cdot 10^{-6} \\ 9.7124 \cdot 10^{-6} & -8.3867 \cdot 10^{-8} & 5.2334 \cdot 10^{-4} & 1.0908 \cdot 10^{-7} \\ 3.7631 \cdot 10^{-9} & 4.8476 \cdot 10^{-6} & 1.0908 \cdot 10^{-7} & 7.5183 \cdot 10^{-6} \end{pmatrix}.$$

As seen in the structure of the covariance and the 2d marginals, the second and fourth parameters are positively correlated. When looking at Table 5.4, the second and the fourth reactions govern the creation and the destruction of the proteins. An increase in the protein creation rate is therefore “cancelled” by an increase in the degradation rate.

A hyperparameter study is done to investigate the impact of several solver settings on the quality of the solution. The references in this case are the MH solution and a solution computed with a fine grid TT-CME solver with $\epsilon = 10^{-7}$, $\ell = 64$, $n_T = 16$. As a first study, we investigate how the relative residual ϵ affects inference by performing a sweep $\epsilon \in \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. In Table 5.3, the results are reported in terms of relative error of the mean, runtime and storage requirement. When comparing to the MCMC

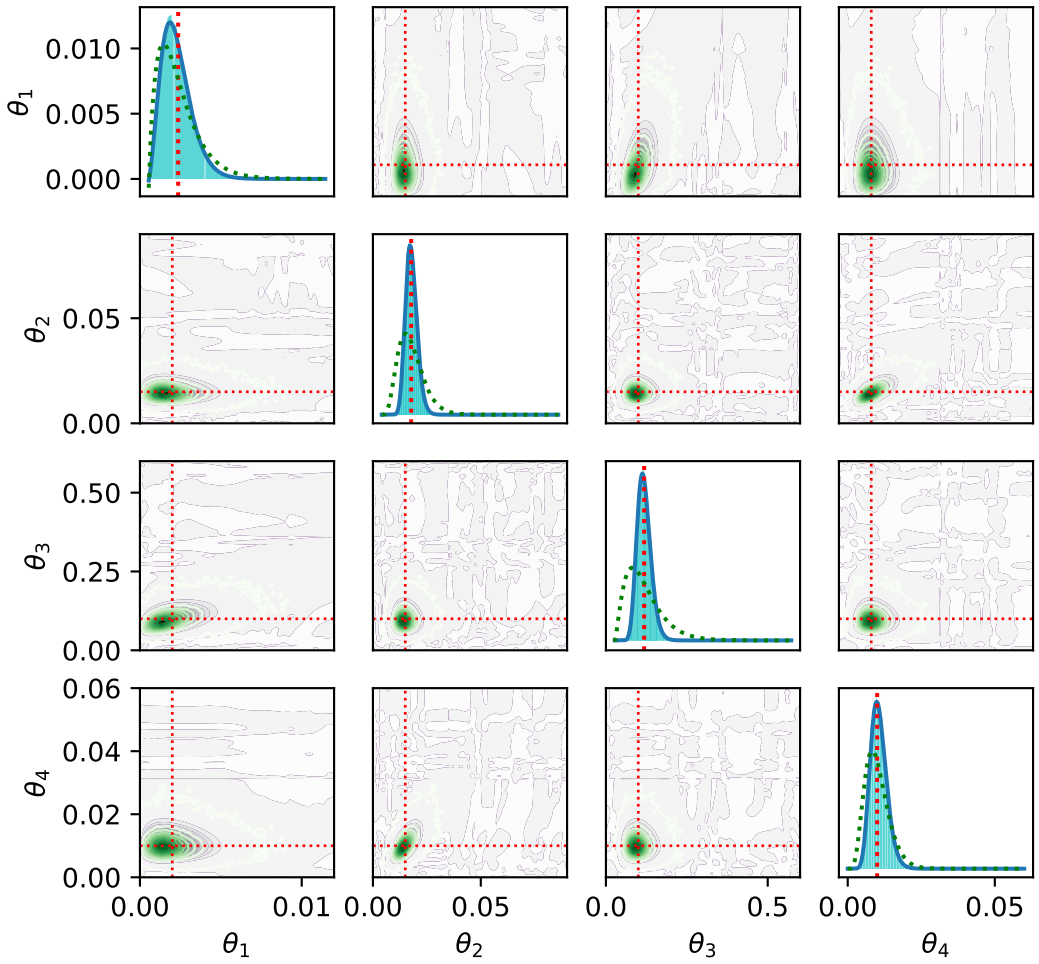


Figure 5.6.: Simple gene expression model: marginalized posterior PDFs over the parameter space. The black regions correspond to high density of the PDF and the green contour lines represent the 2d histograms obtained from the MCMC. The exact parameters are marked with the red dashed lines. In the case of the 1d marginals on the main diagonal, the prior is also represented (green dashed line). Figure adapted from [IWL+21].

based method, the stagnation of the error is achieved for $\epsilon = 10^{-4}$ and smaller. This is not the case when comparing to the fine grid solution. On the other side, a lower relative residual leads to higher computational costs in terms of runtime and storage.

Additional tests regarding the dimension of the univariate bases were conducted. A sweep $\ell \in \{16, 32, 64\}$ is performed to assess how the refinement of the B-splines affects the accuracy. For the case $\ell = 16$, the tensor product basis is unable to properly represent the joint due to oscillations. This is caused by the small posterior variance compared to the knot spacing. However, for the prior, no approximation problems appear. Refining the basis to $\ell = 32$ leads to a smoother approximation of the posterior, since the resolution of the

ϵ	Error w.r.t. MH	Error w.r.t. fine grid	Runtime [min]	Storage [MB]
10^{-3}	3.656×10^{-3}	2.678×10^{-3}	7	0.76
10^{-4}	2.284×10^{-3}	1.429×10^{-3}	10	2.93
10^{-5}	3.530×10^{-3}	1.338×10^{-4}	21	7.37
10^{-6}	3.620×10^{-3}	4.301×10^{-5}	54	14.75

Table 5.3.: Simple gene expression model: TT-CME solver performance for different values of the relative residual ϵ.

basis is sufficient to accommodate the narrow posterior. The conclusion of the study is that the accuracy of the AMEn solver is the limiting factor when performing inference tasks. It has also been noticed that a relative residual of $\epsilon = 10^{-5}$ is a satisfactory trade off between accuracy and solver runtime.

Gene expression model with feedback

A more complicated model is the 3-stage gene expression model with feedback loop [SS08]. The 4 species are: a gene in activated form (G), a gene in inactivated form (G^*), mRNA, and protein. The species are involved in 6 reactions, presented in Table 5.4. Similarly to the previous sections, the SSA is used to generate a reference trajectory for the nominal reaction rates given in Table 5.4. $N_o = 45$ equidistant observations are considered and Gaussian noise is added (see Figure 5.7).

Reaction	$\alpha_m(\boldsymbol{x})$	Rate c_i
$G \to G + M$	$c_1 x_1$	4.0
$M \to M + P$	$c_2 x_2$	10.0
$M \to \varnothing$	$c_3 x_2$	1.0
$G + P \to G^*$	$c_4 x_1 x_3$	0.2
$G^* \to G + P$	$c_5 x_4$	0.6
$P \to \varnothing$	$c_6 x_3$	1.0

Table 5.4.: Reactions of the 3 stage gene expression model.

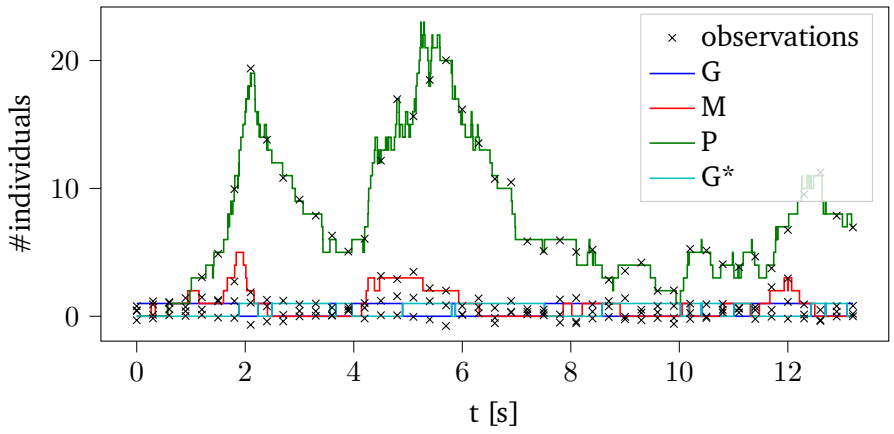


Figure 5.7.: 3 stage gene expression model: 45 noisy observations (marked with the "×" symbol) used for inferring the parameters and the reference trajectory. Figure adapted from [IWL+21].

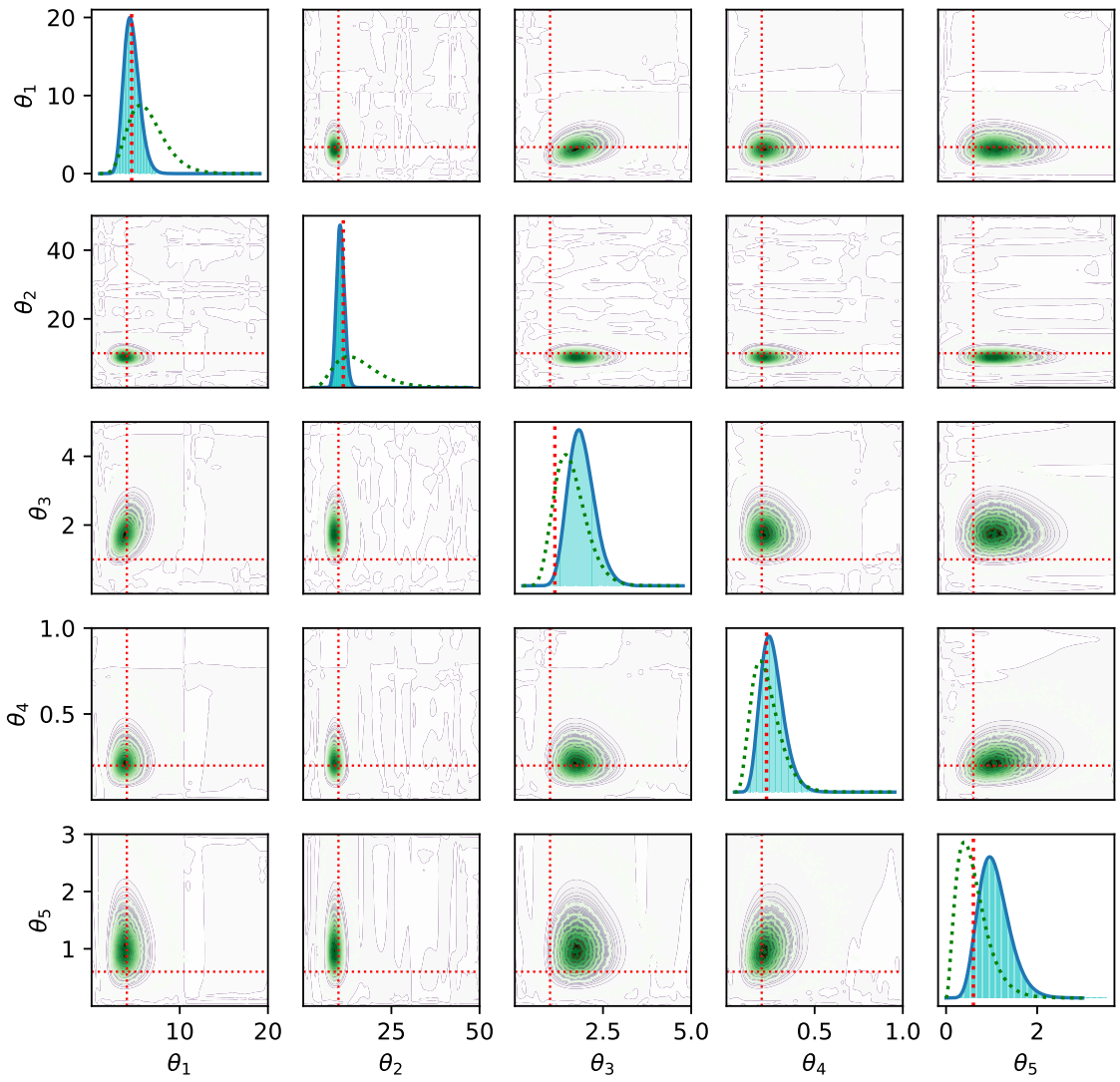


Figure 5.8.: 3 stage gene expression model: marginalized posterior PDFs over the parameter space. The black regions correspond to high density of the PDF and the green contour lines represent the 2d histograms obtained from the MCMC. The exact parameters are marked with the red dashed lines. In the case of the 1d marginals on the main diagonal, the prior is also represented (green dashed line). Figure adapted from [IWL+21].

Only the first 5 parameters are identified, i.e., $\boldsymbol{\theta} = (c_1, c_2, c_3, c_4, c_5)$, and the parameter space is restricted to $\Xi = \times_{i=1}^{5}[0, 5c_i]$. The prior distributions over the parameter space are chosen to be independent Gamma and are truncated within the parameter range. The parameter dependence is approximated using a tensor

product basis of univariate quadratic B-splines with dimension 64. The tolerance of the TT-solver is set to 10^{-5} as a relative residual.

Using the parameter identification algorithm presented in Section 5.4.2, an approximation of the posterior is computed. For the parameter dependence a tensor product of quadratic B-splines of dimensions $\ell = (64, 64, 64, 64, 64)$ is chosen and the relative residual of the AMEn solver is set to 10^{-5}. The marginalized PDFs are reported in Figure 5.8 together with the histograms obtained from the MCMC. On the main diagonal, the one-dimensional marginals $p_{\vartheta_k|\boldsymbol{Y}^{(\leqslant N_\mathrm{o})}}, k \in \{1, 5\}$ are plotted (blue solid lines) together with the histogram and the prior (green dashed line). The non-diagonal plots represent the two-dimensional marginals $p_{\vartheta_k,\vartheta_l|\boldsymbol{Y}^{(\leqslant N_\mathrm{o})}}, k, l \in \{1, 5\}, k \neq l$, together with the histogram (green contour lines). A visual match between the 2 methods can be observed. The expected value and the variance of the obtained posterior are

$$\mathbb{E}(\boldsymbol{\vartheta}) = (4.0237, 9.1566, 1.8336, 0.2369, 1.0655),$$

$$\mathrm{Cov}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}) = \begin{pmatrix} 9.4956 \cdot 10^{-1} & -7.9391 \cdot 10^{-2} & 1.3203 \cdot 10^{-1} & 2.2193 \cdot 10^{-3} & -1.3762 \cdot 10^{-3} \\ -7.9391 \cdot 10^{-2} & 1.5116 & -8.1799 \cdot 10^{-3} & -8.8107 \cdot 10^{-4} & -5.6806 \cdot 10^{-4} \\ 1.3203 \cdot 10^{-1} & -8.1799 \cdot 10^{-3} & 1.6569 \cdot 10^{-1} & -8.1715 \cdot 10^{-4} & 9.2572 \cdot 10^{-4} \\ 2.2193 \cdot 10^{-3} & -8.8107 \cdot 10^{-4} & -8.1715 \cdot 10^{-4} & 5.1425 \cdot 10^{-3} & 4.8380 \cdot 10^{-3} \\ -1.3762 \cdot 10^{-3} & -5.6806 \cdot 10^{-4} & 9.2572 \cdot 10^{-4} & 4.8380 \cdot 10^{-3} & 1.1611 \cdot 10^{-1} \end{pmatrix}.$$

The mean and variance of the MCMC parameter sample are

$$\boldsymbol{\mu}_{\boldsymbol{\theta}} = (4.0503, 9.1995, 1.8443, 0.2379, 1.0680),$$

$$\boldsymbol{C}_{\boldsymbol{\theta}} = \begin{pmatrix} 9.7501 \cdot 10^{-1} & -7.5935 \cdot 10^{-2} & 1.4018 \cdot 10^{-1} & 3.2855 \cdot 10^{-3} & -6.5464 \cdot 10^{-5} \\ -7.5935 \cdot 10^{-2} & 1.5294 & 4.2049 \cdot 10^{-3} & -1.7015 \cdot 10^{-3} & -7.1941 \cdot 10^{-3} \\ 1.4018 \cdot 10^{-1} & 4.2049 \cdot 10^{-3} & 1.7003 \cdot 10^{-1} & -3.8162 \cdot 10^{-4} & 1.4135 \cdot 10^{-3} \\ 3.2855 \cdot 10^{-3} & -1.7015 \cdot 10^{-3} & -3.8162 \cdot 10^{-4} & 5.1862 \cdot 10^{-3} & 5.1926 \cdot 10^{-3} \\ -6.5464 \cdot 10^{-5} & -7.1941 \cdot 10^{-3} & 1.4135 \cdot 10^{-3} & 5.1926 \cdot 10^{-3} & 1.2025 \cdot 10^{-1} \end{pmatrix}.$$

When compared, the relative error between the TT-CME solution and the MCMC is 10^{-3} for the mean and 10^{-2} for the variance, being limited by the small MCMC sample size. Regarding the runtime, the TT-CME solver needs 50 min. As a comparison, the MCMC simulation took approximately 2.5 days to complete for a sample size equal to $5 \cdot 10^5$. Regarding storage needs, storing a TT representation of the parameter dependent generator (inverse of the mass multiplied with the stiffness) requires approximately 11 MB. When using the QTT format, only approximately 0.2 MB are needed. As in the previous cases, the memory consumption for the TT format can be reduced if sparse tensors are used for storing the cores. During the parameter identification, storing the joint over the states and the parameters required at most 10 MB for a $4 \times 32 \times 128 \times 4 \times 64 \times 64 \times 64 \times 64 \times 64$ tensor.

SEIQR model

The final example considered in this chapter is the 5-dimensional SEIQR virus spreading model, containing one additional species compared to the SEIR model: quarantined (Q). The quarantined individuals carry the virus but are not able to spread it further. They also benefit from a maximum survival rate. The 9 reactions are described in Table 5.5. Only the first four parameters are considered as uncertain, i.e., $\boldsymbol{\theta} = (c_1, c_2, c_3, c_4)$. The $N_\mathrm{o} = 45$ observations are represented in Figure 5.9 along with the reference trajectory. A lognormal noise model is assumed for the species Susceptible, Exposed and Infected, while Quarantined and Recovered are exactly observed.

Reaction	$\alpha_m(\boldsymbol{x})$	Rate c_i	Description
$S + I \to E + I$	$c_1 x_1 x_3$	0.04	Susceptible meets infected and becomes exposed
$E \to I$	$c_2 x_2$	0.4	Exposed becomes infected
$I \to Q$	$c_3 x_3$	0.4	Infected becomes quarantined
$I \to \varnothing$	$c_4 x_3$	0.004	Infected individual dies
$I \to R$	$c_5 x_3$	0.12	Infected recovers with immunity
$Q \to R$	$c_6 x_4$	0.8765	Quarantined recovers with immunity
$I \to S$	$c_7 x_3$	0.01	Infected recovers without immunity
$Q \to S$	$c_8 x_4$	0.01	Quarantined recovers without immunity
$\varnothing \to S$	c_9	0.01	New susceptible individual

Table 5.5.: Reactions of the SEIQR model.

The solution is obtained using the TT-CME solver in approximately 42 min for a relative accuracy of $\epsilon = 10^{-5}$ and tensor product quadratic B-splines for the parameter dependence. The posterior is stored using ≈ 30 MB in the QTT format for a state truncation of $\boldsymbol{n} = (128, 64, 64, 32, 32)$, while the CME generator uses a mere 200 KB. The solution in the full format on the other side needs ≈ 4.2 GB for a single parameter.

The posterior marginals are shown in Figure 5.10. For this example, a MCMC reference is computationally unfeasible due to the long simulation time for the single parameter case. The posterior variance is lower than the prior, indicating a larger confidence in estimating the parameters. The expected value and the covariance matrix are:

$$\mathbb{E}(\boldsymbol{\vartheta}) = \begin{pmatrix} 0.03640125 & 0.37381211 & 0.47709099 & 0.00400664 \end{pmatrix}, \tag{5.62}$$

$$\mathrm{Cov}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}) = \begin{pmatrix} 1.7127 \cdot 10^{-5} & 2.0241 \cdot 10^{-6} & 7.9286 \cdot 10^{-6} & 2.0921 \cdot 10^{-8} \\ 2.0241 \cdot 10^{-6} & 2.9069 \cdot 10^{-3} & -7.5025 \cdot 10^{-6} & 1.7820 \cdot 10^{-7} \\ 7.9286 \cdot 10^{-6} & -7.5025 \cdot 10^{-6} & 9.4759 \cdot 10^{-3} & -1.1212 \cdot 10^{-7} \\ 2.0921 \cdot 10^{-8} & 1.7820 \cdot 10^{-7} & -1.1212 \cdot 10^{-7} & 4.0046 \cdot 10^{-7} \end{pmatrix}. \tag{5.63}$$

If the posterior variance becomes lower, the uniformly distributed knots of the B-splines are no longer able to properly capture the steep increase of the PDF. In order to avoid the oscillations, the size of the basis is set to $\boldsymbol{\ell} = (64, 64, 64, 64)$. Refining the basis, changing the position of the knots or adaptively truncating the parameter space are some possible solutions to improve the solver. A variance based criterion can be used to control the adaptivity.

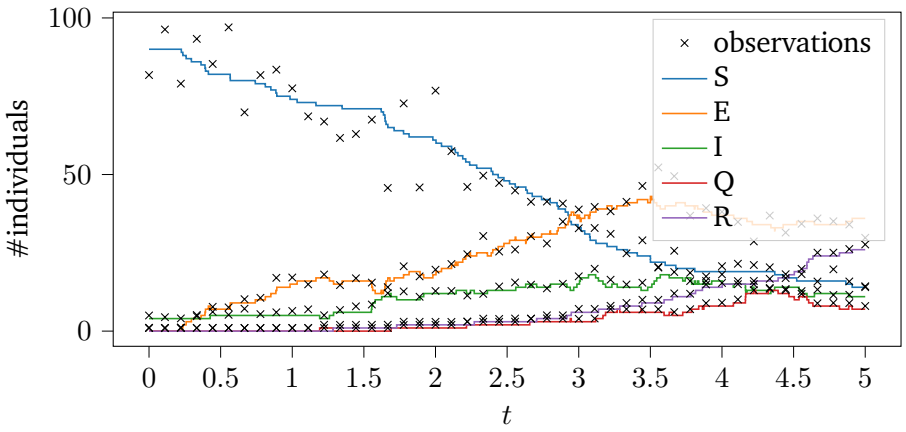


Figure 5.9.: SEIQR model: 45 noisy observations (marked with the "×" symbol) used for inferring the parameters and the reference trajectory. Figure adapted from [IWL+21].

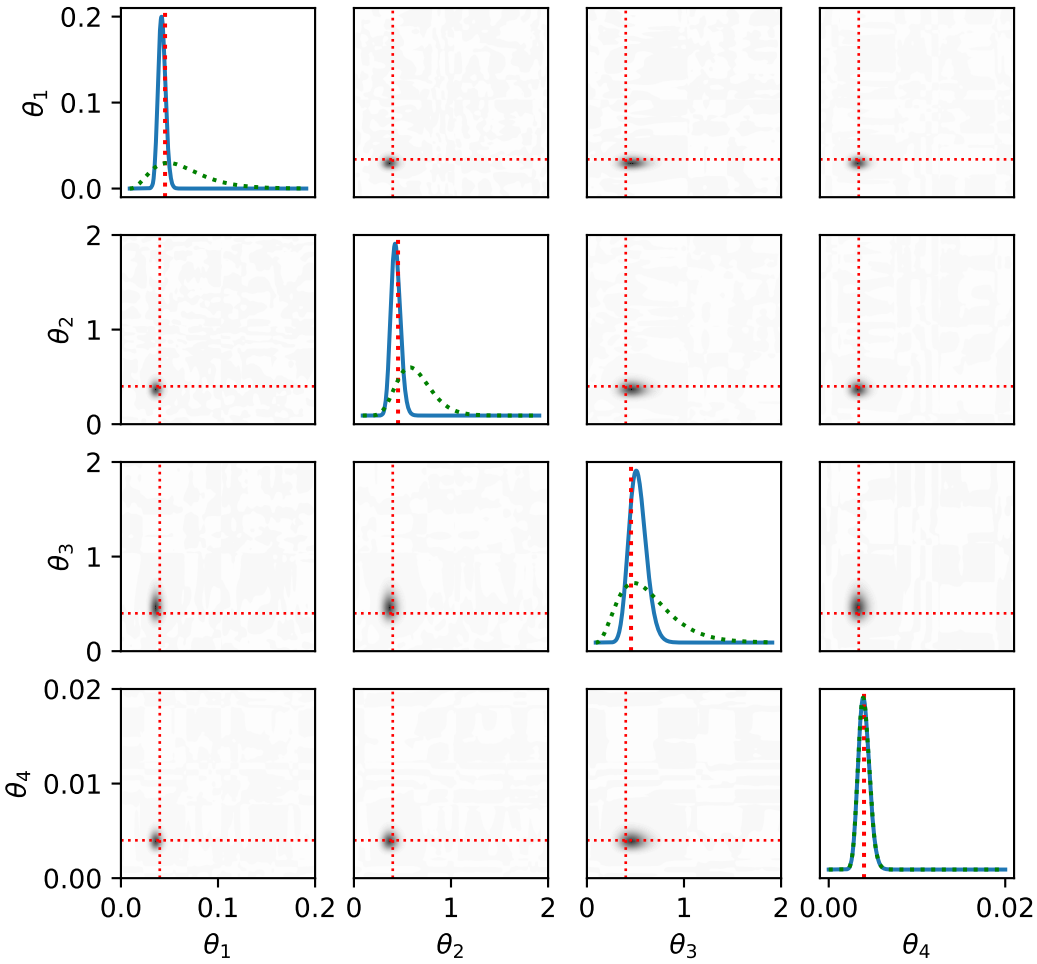


Figure 5.10.: SEIQR model: marginalized posterior PDFs over the parameter space. The black regions correspond to high density of the PDF. The exact parameters are marked with the red dashed lines. In the case of the 1d marginals on the main diagonal, the prior is also represented (green dashed line). Figure adapted from [IWL+21].

6. Tensor-Train Isogeometric Analysis

The focus of this chapter is on the development of surrogate models for parameter dependent systems governed by BVPs. The input parameters are assumed to be variable and to affect the geometry of the model, the material relation or the force term. Once the surrogate model is constructed, tasks such as optimization, control, uncertainty quantification or solving inverse problems can be easily performed by considering the surrogate as a black box model and applying the methods to it. The presented method to compute the surrogate is however white box.

The method of choice for discretizing the PDE is the IGA FEM. Introduced by Tom Hughes in 2005 [HCB05], the main idea of IGA is to represent the geometry using B-spline or NURBS parametrizations and to approximate the solution in the reference domain (the definition domain of the parametrization). The combination of the FEM with geometry description techniques based on computer-aided design (CAD) allows for the exact representation of complex geometries [CHB09, NABR15]. Moreover, compared to ordinary FEM, a reduced number of DoFs is needed for obtaining a similar accuracy [DVBSV14]. The main advantage exploited in this work is the tensor product structure of the reference domain, which makes storing the DoFs in tensor format a natural choice. Due to the mentioned advantages, the IGA has been successfully used to solve BVPs in a variety of fields such as structural analysis [WWS13, CRBH06, MAB$^+$15, Rea06, SKBW10], electromagnetic field simulation [BCdF$^+$20, BSV10, DKSW19a, DKSW19b, SBdFS20], and fluid mechanics [ABKF11, BH08, GPC19, HAB11, LBJ19, WWX$^+$17]. A main disadvantage of IGA is the computationally expensive assembly of the discrete Galerkin operators for the 3-dimensional case [HCB05]. This disadvantage is critical when IGA solvers are used for variable design parameters. This is the case in the fields of shape optimization [FSV15, MGS21, WWXP18, PBC$^+$15], UQ [GACS19, ZS19] or shape morphing [ZGAS22].

To address this problem, we propose a white box IGA solver for parameter dependent BVPs. The parameters can be either geometry design variables or they can affect the material coefficients. The parameter dependence is resolved by interpolating the solution over a tensor-product collocation grid. The space where the solution lives is therefore extended using the tensor product. On the linear algebra level, the DoFs as well as the discrete operators are represented using the tensor formulation presented in the previous chapters. The parameters introduce additional dimensions of the DoF tensor. Exploiting the tensor product structure [ABC$^+$15] of the bilinear forms arising from the IGA discretization allows us to directly construct the operators as well as the right-hand side in the TT format [ILDG22]. The multilinear solver presented in Section 3.2.4 is employed to obtain the solution DoFs in the TT-format as well. The obtained surrogate can be treated as a black box function handle to solve inverse problems using conventional techniques such as MCMC [BGJM11], computing the mode and moments of the posterior or directly approximating the posterior [IWL$^+$21, FLU$^+$20].

The content presented in this chapter is based on our work [ILDG22] and is structured as follows: the first section introduces the model problem as well as the assumptions made. The following section presents the discretization of the geometry using B-splines and NURBS. In the third section, the discretization of the solution space as well as the parameter dependence is explained. In Section 6.4, the construction of all discrete operators in the TT format is described. Numerical experiments are performed and discussed in Section 6.5 to validate and showcase the performance of the method. First a convergence test is performed

on an example problem. The second test assesses the efficiency of the suggested TT-IGA solver for a case where the number of parameters is increased. Two more examples are presented where an inverse problem is solved using the surrogate constructed by the TT-IGA method.

6.1. Problem statement

Let $D(\boldsymbol{\theta}) \subset \mathbb{R}^d$, $d \in \{2, 3\}$ be a domain (in following also referred to as the physical domain) depending on the parameter vector $\boldsymbol{\theta} = (\theta_1, \dots, \theta_{N_\mathrm{p}}) \in \Xi$ for a bounded box-domain $\Xi = \Xi_1 \times \cdots \times \Xi_{N_\mathrm{p}} \subset \mathbb{R}^{N_\mathrm{p}}$ (Ξ_k are intervals). In order to use the IGA framework, some restrictions on the choice of the domain are made. It is assumed that the physical domain is given as the image of an injective map (also called parametrization) $G : [0, 1]^d \times \Xi \to \mathbb{R}^d$:

$$D(\boldsymbol{\theta}) = \mathrm{Im}\{G(\boldsymbol{y}, \boldsymbol{\theta}) \ : \ \boldsymbol{y} \in (0, 1)^d\}, \quad \forall \boldsymbol{\theta} \in \Xi. \tag{6.1}$$

An example is shown in Figure 6.1, where a Cartesian mesh in the reference domain is deformed using a parameter dependent parametrization. In a more general setup, the domain can be represented using several parametrizations such that their images construct a fully connected set and overlap only on the boundary (multipatch approach). In this work, however, only a single patch is considered. Moreover, the following assumptions on the map G are made:

1. For every parameter $\boldsymbol{\theta} \in \Xi$, the restriction $G(\cdot, \boldsymbol{\theta}) : [0, 1]^d \to \mathbb{R}^d$ is Lipschitz continuous.
2. For every point $\boldsymbol{y} \in (0, 1)^d$, the function $G(\boldsymbol{y}, \cdot)$ is assumed to be sufficiently smooth for polynomial interpolation.

On the parameter dependent domain, we consider the following Helmholtz BVP for the scalar field $u : D(\boldsymbol{\theta}) \times \Xi \to \mathbb{R}$:

$$-\nabla \cdot (\kappa(\cdot, \boldsymbol{\theta}) \nabla u(\cdot, \boldsymbol{\theta})) + \rho u(\cdot, \boldsymbol{\theta}) = f(\cdot, \boldsymbol{\theta}), \quad \text{in } D(\boldsymbol{\theta}), \tag{6.2a}$$

$$u(\cdot, \boldsymbol{\theta}) = g(\cdot, \boldsymbol{\theta}), \quad \text{on } \Gamma_\mathrm{D}(\boldsymbol{\theta}), \tag{6.2b}$$

$$\kappa(\cdot, \boldsymbol{\theta}) \partial_{\boldsymbol{\nu}} u(\cdot, \boldsymbol{\theta}) = 0, \quad \text{on } \Gamma_\mathrm{N}(\boldsymbol{\theta}), \tag{6.2c}$$

where $\rho \in \mathbb{R}$ and $\Gamma_\mathrm{D}(\boldsymbol{\theta})$, $\Gamma_\mathrm{N}(\boldsymbol{\theta})$ are the Dirichlet and Neumann boundaries of $D(\boldsymbol{\theta})$, respectively and $\partial_{\boldsymbol{\nu}}$ is the normal derivative. Regarding the coefficient function $\kappa \in L^\infty(D(\boldsymbol{\theta}))$, it is assumed to be uniformly bounded from below for every $\boldsymbol{\theta} \in \Xi$. For the right-hand side, we have $f(\cdot, \boldsymbol{\theta}) \in L^2(D(\boldsymbol{\theta}))$, $\forall \boldsymbol{\theta} \in \Xi$. Under these assumptions, the problem is well-posed for every parameter $\boldsymbol{\theta} \in \Xi$ with the solution $u(\cdot, \boldsymbol{\theta}) \in H^1(D(\boldsymbol{\theta}))$ [Ste07].

The main focus of the work consists in parameter dependent geometry deformations, however, the BVP as presented in (6.2) allows for certain parameters from the vector to affect the coefficient function (parameters that govern the material laws for example).

6.2. Geometry parametrizations using B-splines and NURBS

Spline based representations are the core of most CAD tools due to their numerical stability as well as the ability to model a large variety of shapes [HCB05, CHB09]. Moreover, creating and manipulating shapes represented by B-splines and NURBS is a very intuitive process for humans. This section introduces both B-spline and NURBS representations for the computational domain $D(\boldsymbol{\theta})$.

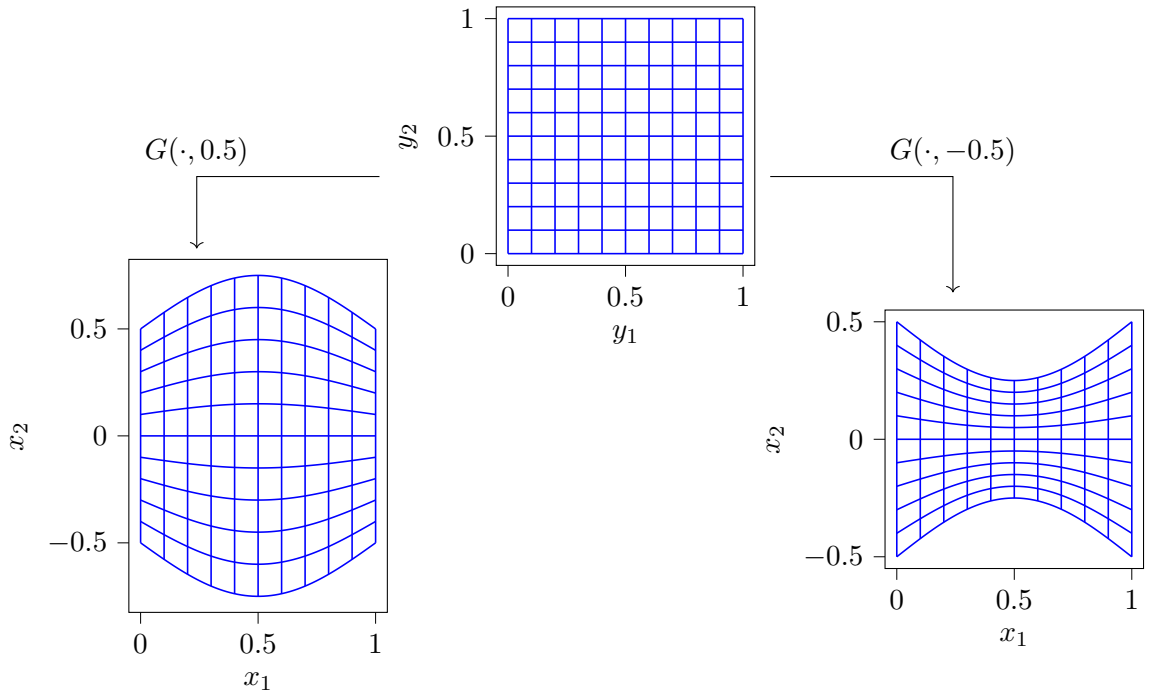


Figure 6.1.: Image of the Cartesian mesh through the mapping $G(\boldsymbol{y}, \theta) = (y_1, (y_2 - 0.5)(1 + \theta \sin \pi y_1))^\top$. The deformations for $G(\cdot, 0.5)$ and $G(\cdot, -0.5)$ are plotted.

As already stated in the previous section, the computational domain is chosen as the image of a map $G(\cdot, \boldsymbol{\theta}) : [0,1]^d \to \mathbb{R}^d, d \in \{2, 3\}$. Moreover, a parametric dependence of the maps and therefore of the computational domain is assumed. Since the map is defined on a hypercube, also called reference domain in the following, it is natural to represent the components of the map $G = (G_1, ..., G_d)$ with functions from the tensor product space of univariate B-splines bases $\mathcal{S}(\boldsymbol{\zeta^{(1)}}, p_1) \otimes \cdots \otimes \mathcal{S}(\boldsymbol{\zeta^{(d)}}, p_d)$ with knot vectors $\boldsymbol{\zeta^{(1)}}, \ldots, \boldsymbol{\zeta^{(d)}}$ and degrees $p_1, \ldots, p_d$, such that

$$G_s(\boldsymbol{y}, \boldsymbol{\theta}) = \sum_{\boldsymbol{k}} \mathsf{p}_{s\boldsymbol{k}}(\boldsymbol{\theta}) b_{\boldsymbol{k}}(\boldsymbol{y}), \quad s \in \{1, ..., d\}, \tag{6.3}$$

where the entries of the parameter dependent tensor $\mathbf{p}(\boldsymbol{\theta}) \in \mathbb{R}^{d \times n_1 \times \cdots \times n_d}$, $n_k = \dim(\mathcal{S}(\boldsymbol{\zeta}^{(k)}, p_k))$, are called control points. Detailed information on how to obtain the control points from a given parametrization will be presented in Section 6.4.1. If $d' < d$ univariate B-splines bases are used to construct the tensor product space $\mathcal{S}(\boldsymbol{\zeta^{(1)}}, p_1) \otimes \cdots \otimes \mathcal{S}(\boldsymbol{\zeta^{(d')}}, p'_d)$, the parametrization returns lower dimensional manifolds embedded in $\mathbb{R}^d$. For example, $d' = 1$ for parametrized curves and $d' = 2$ for surfaces in $\mathbb{R}^3$. As later shown in Section 6.5.4, increasing the multiplicity of certain knots allows for the relaxation of the parametrization's continuity at certain points (edges can be handled).

The second representation used in this work, NURBS, represent a generalization of the B-splines.

Definition 6.2.1 (NURBS basis functions)**.** Let $\{b_{k,p}\}_{k=1}^n$ be a B-spline basis function of dimension n and degree p and $\boldsymbol{w} \in \mathbb{R}^n$ a vector of positive values called the weights. The NURBS basis functions are defined as

$$N_{k,p}(y) = \frac{w_k b_{k,p}(y)}{\sum_{i=1}^n w_i b_{i,p}(y)}. \tag{6.4}$$

Using the defined NURBS basis functions, one can define NURBS curves as

$$F_s(y) = \sum_k \mathbf{p}_{sk} N_{k,p}(y), \tag{6.5}$$

where $\mathbf{p}$ are the control points. In Figure 6.2, an example is illustrated for different degrees of the B-spline basis as well as for increased knot multiplicity.

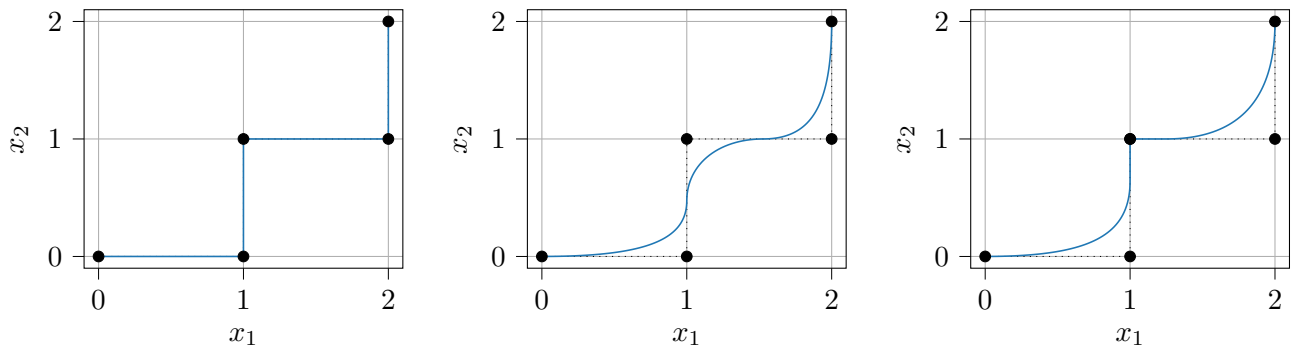
In the case of B-splines, the extension to the multidimensional case has been done by means of tensor product. In the case of NURBS, we first choose the multivariate basis $b_{\boldsymbol{k},\boldsymbol{p}}(\boldsymbol{y}) = b_{k_1,p_1}(y_1) \cdots b_{k_d,p_d}(y_d)$ as the tensor product of d univariate B-splines of dimensions $\boldsymbol{n} = (n_1, ..., n_d)$ and degrees $\boldsymbol{p} = (p_1, ..., p_d)$. Secondly, the weights are represented as a d-dimensional tensor $\boldsymbol{w} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. The multivariate NURBS functions are given by

$$N_{\boldsymbol{k},\boldsymbol{p}}(\boldsymbol{y}) = \frac{\mathsf{w}_{\boldsymbol{k}} b_{\boldsymbol{k},\boldsymbol{p}}(\boldsymbol{y})}{\sum_{\boldsymbol{i}} \mathsf{w}_{\boldsymbol{i}} b_{\boldsymbol{i},\boldsymbol{p}}(\boldsymbol{y})}. \tag{6.6}$$

Using the multivariate NURBS, surfaces and volumes can be parametrized over a mesh of control points. These points are not necessarily part of the domain. In case of a domain $D(\boldsymbol{\theta}) \subset \mathbb{R}^d, d \in \{2, 3\}$, the NURBS representation is

$$G_s(\boldsymbol{y}) = \frac{\sum_{\boldsymbol{k}} \mathsf{p}_{s\boldsymbol{k}} \mathsf{w}_{\boldsymbol{k}} b_{\boldsymbol{k},\boldsymbol{p}}(\boldsymbol{y})}{\sum_{\boldsymbol{i}} \mathsf{w}_{\boldsymbol{i}} b_{\boldsymbol{i},\boldsymbol{p}}(\boldsymbol{y})}, \tag{6.7}$$

where $\mathbf{p} \in \mathbb{R}^{d \times n_1 \times n_2 \times n_3}$ and the multiindex $\boldsymbol{p}$ representing the degree of the involved B-spline bases is dropped to simplify the notation. The parameter dependence of the domain is captured in the control points and in the weights. In case of NURBS, the differentiability of the parametrization can be relaxed by also repeating the control points. This is illustrated in Figure 6.2c, where repeating the control point $(1, 1)$ returns a curve that is continuous but not differentiable for $y = 0.5$.



(a) Linear B-spline basis for $\zeta = (0, 0, 1/4, 1/2, 3/4, 1, 1)$.

(b) Quadratic B-splines with $\zeta = (0, 0, 0, 3/8, 5/8, 1, 1, 1)$

(c) Quadratic B-splines with $\zeta = (0, 0, 0, 3/8, 1/2, 5/8, 1, 1, 1)$

Figure 6.2.: Controlling the smoothness of a curve parametrization by choosing the degree of the basis as well as repeating control points (in the last figure the control point $(1, 1)$ is repeated).

6.3. Discretization of the problem

In the following, the discretization of the parameter dependent solution of the BVP (6.2) is formalized. A Galerkin projection is used to discretize the bilinear forms. For the parameter dependence, the semidiscrete solution is collocated on a tensor product grid.

6.3.1. Galerkin discretization via IGA

The weak formulation of the problem presented in Section 6.1 is used. Converting the bilinear form over the infinite dimensional Sobolev spaces is done in the context of FEM using the Galerkin method [M+03]. For a fixed parameter, the solution lives in the appropriate Sobolev space, in our case $H^1(D(\boldsymbol{\theta}))$. In the IGA, the discrete trial and test spaces are chosen as the span of the B-spline basis composed with the inverse of the geometry mapping [HCB05]

$$\mathcal{V}_{h,\boldsymbol{\theta}} = \text{span}\{b_{\boldsymbol{k}} \circ G^{-1}(\cdot,\boldsymbol{\theta}) \ : \ k_1 = 1,...,n_1 \ , \ k_2 = 1,...,n_2 \ , \ k_3 = 1,...,n_3\}, \tag{6.8}$$

where $G^{-1}(\cdot,\boldsymbol{\theta}) : D(\boldsymbol{\theta}) \to [0,1]^d$ denotes the inverse map for a fixed parameter $\boldsymbol{\theta} \in \Xi$ and the index h is used to mark the discrete spaces. The candidate solutions are therefore expressed using the basis expansion

$$u(\boldsymbol{x},\boldsymbol{\theta}) = \sum_{\boldsymbol{k}} \mathsf{u}_{\boldsymbol{k}}(\boldsymbol{\theta}) b_{\boldsymbol{k}} \circ G^{-1}(\boldsymbol{x},\boldsymbol{\theta}), \quad \boldsymbol{x} \in D(\boldsymbol{\theta}), \tag{6.9}$$

where the parameter dependent tensor $\mathbf{u}(\boldsymbol{\theta}) \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ represents the DoFs. Introducing the solution ansatz and testing with the basis from (6.8) returns the following discrete counterparts of the continuous operators:

$$\mathsf{f}_{\boldsymbol{m}}(\boldsymbol{\theta}) = l_{\boldsymbol{\theta}}(\phi_{\boldsymbol{m}}(\cdot,\boldsymbol{\theta})) = \int_{D(\boldsymbol{\theta})} \phi_{\boldsymbol{m}}(\boldsymbol{x},\boldsymbol{\theta}) f(\boldsymbol{x},\boldsymbol{\theta}) \mathrm{d}\boldsymbol{x}, \tag{6.10}$$

$$\mathsf{M}_{\boldsymbol{m},\boldsymbol{k}}(\boldsymbol{\theta}) = a^{\mathsf{M}}_{\boldsymbol{\theta}}(\phi_{\boldsymbol{m}}(\cdot,\boldsymbol{\theta}), \phi_{\boldsymbol{k}}(\cdot,\boldsymbol{\theta})) = \int_{D(\boldsymbol{\theta})} \phi_{\boldsymbol{m}}(\boldsymbol{x},\boldsymbol{\theta}) \phi_{\boldsymbol{k}}(\boldsymbol{x},\boldsymbol{\theta}) \mathrm{d}\boldsymbol{x}, \tag{6.11}$$

$$\mathsf{S}_{\boldsymbol{m},\boldsymbol{k}}(\boldsymbol{\theta}) = a^{\mathsf{S}}_{\boldsymbol{\theta}}(u_{\boldsymbol{m}}(\cdot,\boldsymbol{\theta}), v_{\boldsymbol{k}}(\cdot,\boldsymbol{\theta})) = \int_{D(\boldsymbol{\theta})} \kappa(\boldsymbol{x},\boldsymbol{\theta}) \nabla u_{\boldsymbol{m}}(\boldsymbol{x},\boldsymbol{\theta}) \cdot \nabla v_{\boldsymbol{k}}(\boldsymbol{x},\boldsymbol{\theta}) \mathrm{d}\boldsymbol{x}, \tag{6.12}$$

where $\phi_{\boldsymbol{i}} = b_{\boldsymbol{i}} \circ G^{-1}(\cdot,\boldsymbol{\theta})$. By applying the substitution theorem for integrals, the integration can be performed over the reference cube and the information about the geometry parametrization is included in the metric tensor. For the presented bilinear forms and the right-hand side this leads to

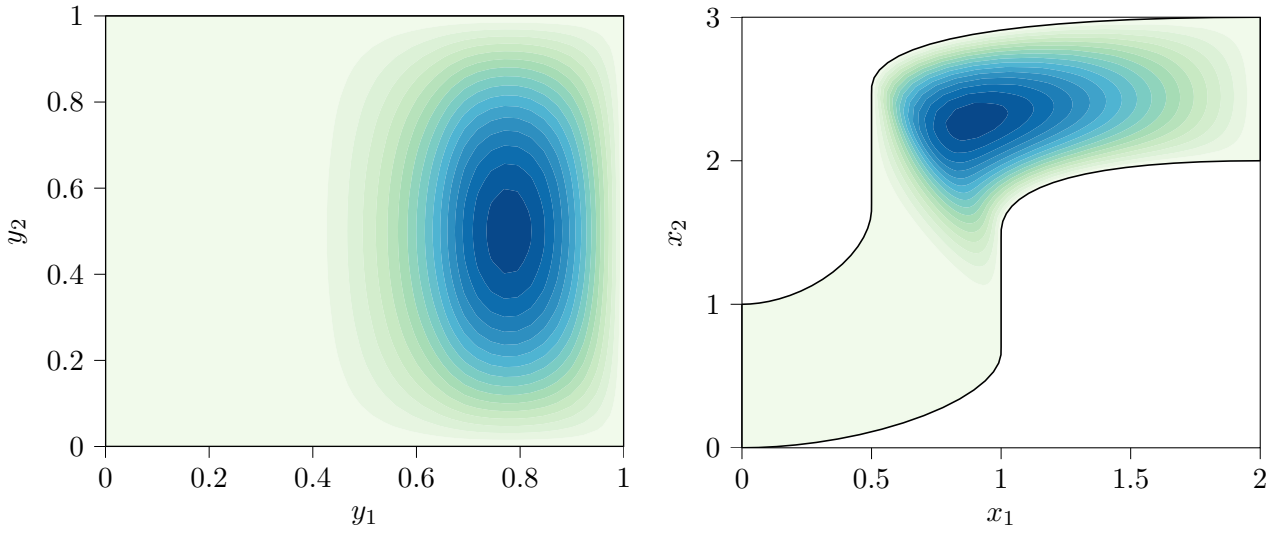
$$\mathsf{f}_{\boldsymbol{m}}(\boldsymbol{\theta}) = \int_{[0,1]^3} b_{\boldsymbol{m}}(\boldsymbol{y}) f(G(\boldsymbol{y},\boldsymbol{\theta})) D(\boldsymbol{y},\boldsymbol{\theta}) \mathrm{d}\boldsymbol{y}, \quad \omega(\boldsymbol{y},\boldsymbol{\theta}) = |\det D_{\boldsymbol{y}} G(\boldsymbol{y},\boldsymbol{\theta})|, \tag{6.13}$$

$$\mathsf{M}_{\boldsymbol{m},\boldsymbol{k}}(\boldsymbol{\theta}) = \int_{[0,1]^3} b_{\boldsymbol{m}}(\boldsymbol{y}) b_{\boldsymbol{k}}(\boldsymbol{y}) \omega(\boldsymbol{y},\boldsymbol{\theta}) \mathrm{d}\boldsymbol{y}, \tag{6.14}$$

$$\mathsf{S}_{\boldsymbol{m},\boldsymbol{k}}(\boldsymbol{\theta}) = \int_{[0,1]^3} \nabla b_{\boldsymbol{m}}(\boldsymbol{y})^\top \boldsymbol{K}(\boldsymbol{y},\boldsymbol{\theta}) \nabla b_{\boldsymbol{k}}(\boldsymbol{y}) \hat{\kappa}(\boldsymbol{y},\boldsymbol{\theta}) \mathrm{d}\boldsymbol{y}, \tag{6.15}$$

$$\boldsymbol{K}(\boldsymbol{y},\boldsymbol{\theta}) = D_{\boldsymbol{y}} G(\boldsymbol{y},\boldsymbol{\theta})^{-\top} D_{\boldsymbol{y}} G(\boldsymbol{y},\boldsymbol{\theta})^{-1} \omega(\boldsymbol{y},\boldsymbol{\theta}),$$

where $D_{\boldsymbol{y}} G(\boldsymbol{y},\boldsymbol{\theta}) \in \mathbb{R}^{d \times d}$ is the Jacobian of the geometry mapping G with respect to the reference coordinates and $\hat{\kappa}(\boldsymbol{y},\boldsymbol{\theta}) = \kappa(G(\boldsymbol{y},\boldsymbol{\theta}),\boldsymbol{\theta})$ is the coefficient function represented in the reference domain. Solving the BVP on the parameter dependent domain $D(\boldsymbol{\theta})$ is therefore equivalent to solving a BVP on the reference cube $[0,1]^d$ for a solution with modified coefficients. The solution on the reference domain, denoted in the following as $\hat{u}(\cdot,\boldsymbol{\theta})$, belongs to the Sobolev space $H^1([0,1]^3)$ and is approximated using a



(a) Reference domain.

(b) Physical domain.

Figure 6.3.: Representation of a basis function in the reference domain as well as the physical domain.

function from $\mathcal{S}(\boldsymbol{\zeta}, p)$, given as

$$\hat{u}(\boldsymbol{y}, \boldsymbol{\theta}) \approx \sum_{\boldsymbol{k}} \mathsf{u}_{\boldsymbol{k}}(\boldsymbol{\theta}) b_{\boldsymbol{k}}(\boldsymbol{y}). \tag{6.16}$$

In Figure 6.3, a quadratic bivariate B-spline basis function is represented in the reference domain and in the physical domain as an example.

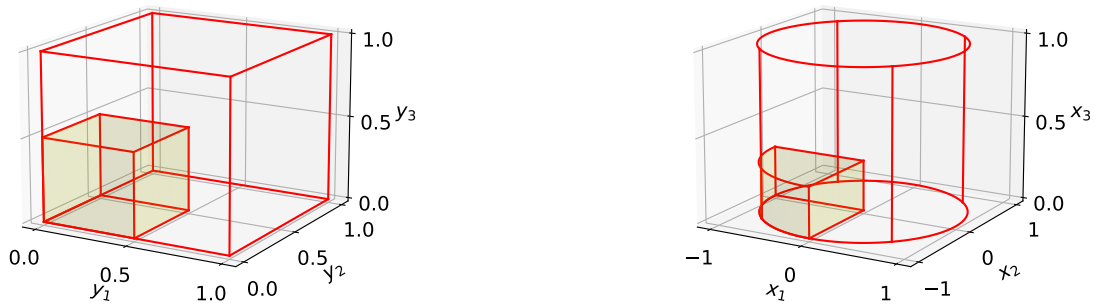
Basis refinement

In (6.8), the chosen basis coincides with the B-spline basis used for the geometry parametrization. However, for practical cases, the basis used to represent the solution should be larger (in terms of the dimension of the univariate bases) than the one used for the parametrization. This can be the case for NURBS parametrizations, where the basis used for representing the geometry is usually coarse.

To this end, the basis used for approximating the solution must be extended. One strategy, called h refinement [HCB05], implies enlarging the knot vectors of the univariate bases. Let $\bigotimes_{k=1}^{d} \mathcal{S}(\boldsymbol{\zeta}^{(k)}, p_k)$, $k \in \{2, 3\}$ be the discrete B-spline space used for the geometry. A finer space for approximating the solution can be obtained by performing knot insertion. Uniform refinement implies adding the knots $(\zeta_{i+1}^{(k)} + \zeta_i^{(k)})/2$, $\forall k, i$ if $\zeta_{i+1}^{(k)} > \zeta_i^{(k)}$.

Increasing the multiplicity of a knot $\zeta_i^{(k)}$ results in relaxing the continuity of basis $\{b_i^{(k)}\}$ at $\zeta_i^{(k)}$. This can be especially useful when considering jumps in the coefficient function κ from (6.2). We restrict ourselves to the case where the material coefficient represented in the reference domain $\hat{\kappa}$ is smooth over Cartesian partitions of the reference domain $[0, 1]^d$, i.e., there exist d vectors $\boldsymbol{\eta}^{(k)} \subset [0, 1]$ with elements in ascending order and containing 0, 1 as first and last element such that

$$\kappa(G(\boldsymbol{y}, \boldsymbol{\theta}), \boldsymbol{\theta}) \text{ is smooth over } \boldsymbol{y} \in (\eta_{i_1}^{(1)}, \eta_{i_1+1}^{(1)}) \times \cdots \times (\eta_{i_d}^{(d)}, \eta_{i_d+1}^{(d)}), \forall \boldsymbol{i}, \boldsymbol{\theta}, \tag{6.17}$$



(a) Reference domain.

(b) Physical domain.

Figure 6.4.: The function κ is not continuous at the interface between the yellow region and the remaining physical domain. In the reference domain, this region is given by $[0, 0.5] \times [0, 0.5] \times [0, 0.5]$. The corresponding B-spline basis must contain the value 0.5 d-times in the knot vector.

where the indices i_k loop over all the intervals of consecutive points constructed using the vectors $\boldsymbol{\eta}$. The discontinuity of $\hat{\kappa}$ along the boundaries of the Cartesian partitions implies a lack of smoothness for the solution along those surfaces. In order to accommodate for the lack of the solution's smoothness, the points that define the Cartesian partition must be part of the knot vectors $\boldsymbol{\eta}^{(k)} \subset \boldsymbol{\zeta}^{(k)}, \forall k$ and their multiplicity has to be equal to the degree of the basis.

A relevant example is provided in Figure 6.4. The Cartesian reference subdomain $[0, 0.5]\times[0, 0.5]\times[0, 0.5]$, which is marked with yellow in Figure 6.4(a), is mapped through a B-spline-based parametrization to the corresponding physical domain marked with yellow in Figure 6.4(b). In this particular case, the value 0.5 must be in the knot vector of the univariate B-spline bases with the appropriate multiplicity in order to accommodate the solution.

6.3.2. Parameter space discretization

In the previous section, the semidiscrete problem was introduced. The goal of the work is to give a surrogate model of the parameter dependent solution, i.e., an approximation of the solution over the joint physical-parameter domain that can be efficiently evaluated. One common way is to perform a polynomial approximation over the parameter space. In this work, a tensor product polynomial approximation is used since we want to take advantage of applying low-rank tensor decomposition methods. Alternative approaches include sparse polynomial spaces for approximation [ABW22, BCM17, BCDM17] or piecewise polynomial approximations [CDFS13, HKT05, Ric85]. Furthermore, the discrete system is obtained using a collocation based method [BNT07, CCNT16], as compared to stochastic Galerkin methods [GS91, EMM20, KS11, BTZ04].

The first step is collocating the geometry parametrization on a tensor product grid of parameters. For a B-spline geometry we have

$$\Theta = \left\{\theta_{i_1}^{\mathrm{C},1}\right\}_{i_1=1}^{\ell_1} \times \left\{\theta_{i_2}^{\mathrm{C},2}\right\}_{i_2=1}^{\ell_2} \times \cdots \times \left\{\theta_{i_{N_\mathrm{p}}}^{\mathrm{C},N_\mathrm{p}}\right\}_{i_{N_\mathrm{p}}=1}^{\ell_{N_\mathrm{p}}} \subset \Xi, \tag{6.18}$$

where the superscript "C" is used to denote the collocation points. In this work, the univariate grids consist of Gauss-Legendre nodes, however, other collocation points can be chosen as well, e.g., Chebyshev nodes. The geometry parametrization collocated on the grid Θ is modified to

$$G_s(\boldsymbol{y}, \boldsymbol{\theta_i}) \approx \sum_{\boldsymbol{k}} \mathsf{p}_{s\boldsymbol{k}}(\boldsymbol{\theta_i}) b_{\boldsymbol{k}}(\boldsymbol{y}) = \sum_{\boldsymbol{k}} \mathsf{p}_{s\boldsymbol{ki}} b_{\boldsymbol{k}}(\boldsymbol{y}), \tag{6.19}$$

for $\boldsymbol{\theta_i} = (\theta^{\mathrm{C},1}_{i_1}, ..., \theta^{\mathrm{C},N_\mathrm{p}}_{i_{N_\mathrm{p}}})$. The DoFs are now a $(1 + 3 + N_\mathrm{p})$-dimensional tensor $\mathbf{p} \in \mathbb{R}^{3 \times n_1 \times n_2 \times n_3 \times \ell_1 \times \cdots \times \ell_{N_\mathrm{p}}}$, where the last N_p modes correspond to the parameter space. That is, if we take a point $\boldsymbol{\theta_i} \in \Theta$, the corresponding control points are the slice $\mathbf{p}_{:\boldsymbol{i}}$. In case of a NURBS parametrization, the dimensionality of the weights tensor must be accordingly increased, such that

$$G_s(\boldsymbol{y}, \boldsymbol{\theta_i}) \approx \frac{\sum_{\boldsymbol{k}} \mathsf{p}_{s\boldsymbol{k}} \mathsf{w}_{\boldsymbol{k}}(\boldsymbol{\theta_i}) b_{\boldsymbol{k}}(\boldsymbol{y})}{\sum_{\boldsymbol{l}} \mathsf{w}_{\boldsymbol{l}}(\boldsymbol{\theta_i}) b_{\boldsymbol{l}}(\boldsymbol{y})} = \frac{\sum_{\boldsymbol{k}} \mathsf{p}_{s\boldsymbol{ki}} \mathsf{w}_{\boldsymbol{ki}} b_{\boldsymbol{k}}(\boldsymbol{y})}{\sum_{\boldsymbol{l}} \mathsf{w}_{\boldsymbol{li}} b_{\boldsymbol{l}}(\boldsymbol{y})}. \tag{6.20}$$

The discretized geometry mapping can be evaluated between the points from Θ using interpolation. In case of a B-spline parametrization, the interpolation is given as

$$G_s(\boldsymbol{y}, \boldsymbol{\theta}) \approx \sum_{\boldsymbol{k},\boldsymbol{i}} \mathsf{p}_{s\boldsymbol{ki}} b_{\boldsymbol{k}}(\boldsymbol{y}) L^{(1)}_{i_1}(\theta_1) \cdots L^{(N_\mathrm{p})}_{i_{N_\mathrm{p}}}(\theta_{N_\mathrm{p}}) = \sum_{\boldsymbol{k},\boldsymbol{i}} \mathsf{p}_{s\boldsymbol{ki}} b_{\boldsymbol{k}}(\boldsymbol{y}) L_{\boldsymbol{i}}(\boldsymbol{\theta}), \tag{6.21}$$

where $\left\{ P^{(k)}_{i_k} \right\}_{i_k=1}^{\ell_k}$ are Lagrange polynomials corresponding to the collocation points.

The parameter dependent solution formally belongs to the space $L^2(\Xi, G, 1)$ defined as

$$L^2(\Xi, G, \alpha) = \{u(\cdot,\cdot) \,:\, u(\cdot, \boldsymbol{\theta}) \in H^\alpha(D(\boldsymbol{\theta})),\ \forall \boldsymbol{\theta} \in \Xi \text{ and } ||u||^2_{L^2(\Xi,G,\alpha)} < \infty\}, \quad \alpha \in \mathbb{Z}, \tag{6.22}$$

with the corresponding norm

$$||u||^2_{L^2(\Xi,G,\alpha)} = \int_\Xi ||u||^2_{H^\alpha(D(\boldsymbol{\theta}))} \mathrm{d}\boldsymbol{\theta}. \tag{6.23}$$

Since in the IGA we search for the solution represented in the reference domain, the discretization of $\hat{u}(\boldsymbol{y}, \boldsymbol{\theta})$ is chosen from a finite dimensional subspace of the tensor product space $H^1([0,1]^d) \otimes L^2(\Xi), d \in \{1, 2\}$. The discrete subspace is chosen as the tensor product space $\mathcal{V}_{\boldsymbol{\zeta},\boldsymbol{\ell}} = \mathcal{S}(\boldsymbol{\zeta}, p) \otimes \mathcal{P}_{\boldsymbol{\ell}}$, where $\mathcal{P}_{\boldsymbol{\ell}} = \mathcal{P}_{\ell_1} \otimes \cdots \otimes \mathcal{P}_{\ell_{N_p}}$. The dimension of this space is $n_1 n_2 n_3 \ell_1 \cdots \ell_{N_p}$ and the basis representation for the solution $\hat{u}$ is used, such that

$$\hat{u}(\boldsymbol{y}, \boldsymbol{\theta}) \approx \hat{u}_{\boldsymbol{n},\boldsymbol{\ell}}(\boldsymbol{y}, \boldsymbol{\theta}) = \sum_{\boldsymbol{k},\boldsymbol{i}} \mathsf{u}_{\boldsymbol{ki}} b_{\boldsymbol{k}}(\boldsymbol{y}) L_{\boldsymbol{i}}(\boldsymbol{\theta}), \tag{6.24}$$

where $\mathbf{u} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times \ell_1 \times \cdots \times \ell_{N_\mathrm{p}}}$ are the DoFs. The following holds for every collocation point $\boldsymbol{\theta_i} \in \Theta$

$$u(\boldsymbol{x}(\boldsymbol{y}), \boldsymbol{\theta_i}) = \hat{u}(\boldsymbol{y}, \boldsymbol{\theta_i}) \approx \sum_{\boldsymbol{k}} \mathsf{u}_{\boldsymbol{ki}} b_{\boldsymbol{k}}(\boldsymbol{y}). \tag{6.25}$$

Due to the interpolating property of the chosen representation, the fully-discretized solution can be recovered by solving $\ell_1 \ell_2 \cdots \ell_{N_\mathrm{p}}$ multilinear systems

$$\mathbf{D}(\boldsymbol{\theta_i}) \mathbf{u}_{:\boldsymbol{i}} = \mathbf{f}(\boldsymbol{\theta_i}), \quad \forall \boldsymbol{i} \in \bigtimes_{k=1}^{N_\mathrm{p}} \{1, ..., \ell_k\}, \tag{6.26}$$

where $\mathbf{f} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is the parameter dependent right-hand side and $\mathbf{D}(\boldsymbol{\theta}) \in \mathbb{R}^{(n_1 \times n_2 \times n_3) \times (n_1 \times n_2 \times n_3)}$ is the discrete operator obtained from the Galerkin discretization and enforcing the boundary conditions. It contains the previously defined mass and stiffness terms (depending on the problem to be solved). The $\prod_k \ell_k$ systems from (6.26) can be cast to an extended system $\mathbf{Du} = \mathbf{f}$ that can be solved to obtain the DoF tensor $\mathbf{u} \in \mathbb{R}^{(n_1 \times n_2 \times n_3) \times (\ell_1 \times \cdots \times \ell_{N_\mathrm{p}})}$ using one single solver call. The tensor operators are in this case

$$\mathsf{D}_{\boldsymbol{mi},\boldsymbol{kq}} = \delta_{\boldsymbol{i}}^{\boldsymbol{q}} \mathsf{D}_{\boldsymbol{m},\boldsymbol{k}}(\boldsymbol{\theta_i}), \quad \mathbf{D} \in \mathbb{R}^{(n_1 \times n_2 \times n_3 \times \ell_1 \times \cdots \times \ell_{N_\mathrm{p}}) \times (n_1 \times n_2 \times n_3 \times \ell_1 \times \cdots \times \ell_{N_\mathrm{p}})}, \tag{6.27}$$

$$\mathsf{f}_{\boldsymbol{mi}} = \mathsf{f}_{\boldsymbol{m}}(\boldsymbol{\theta_i}), \quad \mathbf{f} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times \ell_1 \times \cdots \times \ell_{N_\mathrm{p}}}. \tag{6.28}$$

Under certain assumptions regarding the geometry map, the determinant of the Jacobian $\omega(\boldsymbol{y}, \boldsymbol{\theta})$ and the coefficient $\kappa(\boldsymbol{x}, \boldsymbol{\theta})$, an a priori error estimate for the elliptic case $\rho = 0$ is shown in [CCNT16]. The proof uses the existence of a map between the reference domain and the physical domain. In case of the IGA, this map is the B-spline or the NURBS representation. The information of the domain deformation is therefore included in a density term in the integrals arising from Galerkin FEM discretization. This way, the problem is reduced to the one studied in [BNT07], where the dependence on random right-hand side and coefficients is studied.

Estimating the error between the actual solution and the approximation can be split in the error arising from the IGA-FEM discretization and the error introduced by the parameter space discretization. The first error term concerning the IGA-FEM is denoted with $\epsilon_{\mathrm{IGA}}(h, p)$ and it decreases with $\mathcal{O}(h^{(p+1)})$, where p is the degree of the B-splines used (assumed to have the same degree) and h is the maximum grid step [DVBSV14]. For the error due to the parameter space discretization, the following estimate holds: $\epsilon_\Xi \leqslant C \sum_k e^{-\alpha_k \ell_k}$ for positive α_k and $C > 0$ [BNT07, CCNT16]. Combining the two together, we obtain the error estimate in the norm defined in (6.23)

$$||\hat{u} - \hat{u}_{\boldsymbol{n},\boldsymbol{\ell}}||_{L^2(\Xi,G,1)} \leqslant \epsilon_{\mathrm{IGA}}(h, p) + \epsilon_\Xi(\boldsymbol{\ell}). \tag{6.29}$$

6.4. TT IGA

In the previous sections, the IGA discretization for the parameter dependent BVP has been presented. One disadvantage of IGA FEM is the computational complexity of assembling the discrete operators. The number of nonzero elements in the mass and stiffness matrices is $\mathcal{O}(n^d p^d)$, where d is the number of dimensions, n is the maximum size of the univariate B-spline bases and p is the maximum degree of the B-splines. If we also consider the additional dimensions due to the parameter dependence, the computational complexity for storing the solution and the operator worsens even further. Several methods have been proposed to make the assembly step efficient, ranging from optimal quadrature rules [HRS10b, ACH$^+$12, RP12], to using lookup tables [MJ15] and taking advantage of the tensor product structure of the solution space [ABC$^+$15, Hof18, MJKL17].

In this section, we present a framework to directly assemble all tensor-operators (stiffness $\mathbf{S}$, mass $\mathbf{M}$), as well as the right-hand side, in order to obtain a multilinear system in the TT format for the fully discretized solution. Having all tensors in the TT format, the multilinear solver is used to obtain a TT approximation of the parameter dependent solution u. For the entire section, the case $d = 3$ is chosen. For the case $d = 2$, the derivations are analogous.

6.4.1. Geometry interpolation

As later required for constructing the discrete operators in the TT format, a low-rank representation of the control points (and weights) from the representations (6.19) and (6.20) is needed. As a first step,

the B-spline interpolation of the geometry map $G(\boldsymbol{y}, \boldsymbol{\theta})$ is considered. A popular technique for B-spline interpolation is to use the Greville abscissae [Joh05]. The Greville abscissae of a B-spline basis $\{b_i\}_i$ of degree p and with the knot vector $\boldsymbol{\zeta}$ are defined as the average of p consecutive knots

$$y_i^{\mathrm{G}} = \frac{\zeta_{i+1} + \cdots + \zeta_{i+p+1}}{p}, \quad i = 1, ..., n, \tag{6.30}$$

where n is the dimension of the B-spline basis. One property of the Greville abscissae is that the matrix $\boldsymbol{B}^{(\mathrm{G})}$ with the entries $B_{ij}^{(\mathrm{G})} = b_i(y_j^{(\mathrm{G})})$ is invertible [Joh05]. This implies that evaluating a function over the Greville grid is sufficient to build a B-spline interpolation of it. For the multidimensional case, a Cartesian product between the individual Greville points of the B-spline bases and the parameter grid Θ is constructed and the three components of the map G are evaluated, resulting in the tensor

$$\mathsf{g}_{s\boldsymbol{mi}} = G_s \left(y_{m_1}^{\mathrm{G},1}, y_{m_2}^{\mathrm{G},2}, y_{m_3}^{\mathrm{G},3}, \theta_{i_1}^{\mathrm{C},1}, ..., \theta_{i_{N_\mathrm{p}}}^{\mathrm{C},N_\mathrm{p}} \right), \tag{6.31}$$

where $\{y_{m_1}^{\mathrm{G},1}\}_{m_1}$, $\{y_{m_2}^{\mathrm{G},2}\}_{m_2}$, $\{y_{m_3}^{\mathrm{G},3}\}_{m_3}$ are the Greville abscissae for the univariate B-spline bases. In general, a TT representation of $\mathbf{g} \in \mathbb{R}^{3 \times n_1 \times n_2 \times n_3 \times \ell_1 \times \cdots \times \ell_{N_p}}$ cannot be given in closed form and therefore the cross-approximation method [OT10] must be used. By enforcing the B-spline representation to coincide with the given geometry parametrization on the joint Greville-parameter grid, the following multilinear system is obtained for the control points:

$$\mathsf{B}\mathsf{p}_{s::} = \mathbf{g}_{s::}, \quad \mathsf{B} = \boldsymbol{B}^{(\mathrm{G},1)} \otimes \boldsymbol{B}^{(\mathrm{G},2)} \otimes \boldsymbol{B}^{(\mathrm{G},3)} \otimes \mathsf{I}_{\boldsymbol{\ell}}. \tag{6.32}$$

The tensor operator B has the TT-rank $\boldsymbol{r} = \mathbf{1}$, thus allowing for a fast computation of the system's solution in the TT format, due to the fact that the inverse B^{-1} can be computed easily.

When using the NURBS representation of the geometry, the control points as well as the weights are usually constructed by CAD tools. However, in certain cases, adding parameter dependency to control points and to the weights is of interest for performing optimization tasks or in the field of uncertainty quantification [GACS19]. The setup considered in this work starts from a known dependence of the control points $\mathsf{p}_{k\boldsymbol{j}}(\boldsymbol{\theta})$ and the weights $\mathsf{w}_{\boldsymbol{j}}(\boldsymbol{\theta})$ w.r.t. the parameters. In this case, the user must provide the trajectory of the individual control points within the parameter space Ξ as a function handle. The TT representation of the collocated control points and weights can be computed using the cross approximation in the TT format, such that

$$\begin{aligned} \mathsf{p}_{k\boldsymbol{ji}} &= \mathsf{p}_{k\boldsymbol{j}}(\boldsymbol{\theta}_{\boldsymbol{i}}^{\mathrm{C}}), \\ \mathsf{w}_{\boldsymbol{ji}} &= \mathsf{w}_{\boldsymbol{j}}(\boldsymbol{\theta}_{\boldsymbol{i}}^{\mathrm{C}}). \end{aligned}$$

If a geometry is already given in NURBS format, a B-spline representation can be easily obtained by evaluating the NURBS parametrization in the TT format to obtain the tensor $\mathbf{g}$.

Independent of the geometry representation, a function $f(\cdot, \boldsymbol{\theta}) \in H^1(D(\boldsymbol{\theta}))$ can be interpolated to obtain its representation $\hat{f}$ in the reference domain $[0,1]^3 \times \Xi$ by solving the system

$$\sum_{\boldsymbol{n},\boldsymbol{q}} \mathsf{B}_{\boldsymbol{mi},\boldsymbol{nq}} \mathsf{f}_{\boldsymbol{nq}} = f(\mathsf{g}_{\boldsymbol{mi}}^1, \mathsf{g}_{\boldsymbol{mi}}^2, \mathsf{g}_{\boldsymbol{mi}}^3, \boldsymbol{\theta}_{\boldsymbol{i}}^{\mathrm{C}}), \tag{6.33}$$

where the TT representation of the right-hand side can be done using the cross approximation for nontrivial examples.

6.4.2. Discrete operators

One of the main results of this chapter is the construction of the IGA operators directly in the TT format. In this section, the TT representation of the operators is given. For all discrete operators, the integration over the reference domain $[0,1]^3$ is achieved by constructing a tensor-product grid of univariate quadrature points

$$\left\{y^{\mathrm{Q},1}_{i_1}\right\}_{i_1} \times \left\{y^{\mathrm{Q},2}_{i_2}\right\}_{i_2} \times \left\{y^{\mathrm{Q},3}_{i_3}\right\}_{i_3}, \tag{6.34}$$

together with the corresponding quadrature weights $\mathsf{w}_{\boldsymbol{i}} = w^{(1)}_{i_1} w^{(2)}_{i_2} w^{(3)}_{i_3}$. Since the B-spline bases are piecewise polynomials, the univariate quadrature grids are chosen as a concatenation of Gauss-Legendre quadrature points for the intervals defined by consecutive knots. In this case, they allow for exact polynomial integration in between the knots of the basis.

Let us first consider the integral in the mass tensor operator

$$\mathsf{M}_{\boldsymbol{mi},\boldsymbol{lq}} = \delta^{\boldsymbol{i}}_{\boldsymbol{q}} \int_U \omega(\boldsymbol{y}, \boldsymbol{\theta}^{\mathrm{C}}_{\boldsymbol{i}}) b_{\boldsymbol{m}}(\boldsymbol{y}) b_{\boldsymbol{l}}(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} \approx \delta^{\boldsymbol{i}}_{\boldsymbol{q}} \sum_{\boldsymbol{j}} \mathsf{w}_{\boldsymbol{j}} \mathsf{o}_{\boldsymbol{ji}} b_{m_1}(y^{\mathrm{Q},1}_{j_1}) \cdots b_{m_3}(y^{\mathrm{Q},3}_{j_3}) b_{l_1}(y^{\mathrm{Q},1}_{j_1}) \cdots b_{l_3}(y^{\mathrm{Q},3}_{j_3}), \tag{6.35}$$

where the tensor $\mathbf{o}$ contains the value of the function ω defined in formula (6.14), evaluated on the Cartesian product between the quadrature grid and the collocation grid, that is

$$\mathsf{o}_{\boldsymbol{ji}} = \omega\left(y^{\mathrm{Q},1}_{j_1}, y^{\mathrm{Q},2}_{j_2}, y^{\mathrm{Q},3}_{j_3}, \boldsymbol{\theta}^{\mathrm{C}}_{\boldsymbol{i}}\right). \tag{6.36}$$

In the following, the tensor $\mathbf{o}$ is assumed to have a TT representation with TT cores $\{\mathbf{g}^{(\mathbf{o},k)}\}_{k=1}^{3+N_\mathrm{p}}$. It will be later shown how to obtain this representation. The discrete mass operator can be written in this case as

$$\mathsf{M}_{\boldsymbol{mi},\boldsymbol{lq}} \approx \delta^{\boldsymbol{i}}_{\boldsymbol{q}} \sum_{\boldsymbol{j}} \mathsf{w}_{\boldsymbol{j}} \sum_{\boldsymbol{s}} \left(\prod_{k=1}^{3} \mathsf{g}^{(\mathbf{o},k)}_{s_{k-1} j_k s_k} \prod_{k=4}^{3+N_\mathrm{p}} \mathsf{g}^{(\mathbf{o},k)}_{s_{k-1} i_{k-3} s_k} \right) b_{m_1}(y^{\mathrm{Q},1}_{j_1}) \cdots b_{m_3}(y^{\mathrm{Q},3}_{j_3}) b_{l_1}(y^{\mathrm{Q},1}_{j_1}) \cdots b_{l_3}(y^{\mathrm{Q},3}_{j_3}), \tag{6.37}$$

where $\boldsymbol{s}$ is used for summing along the ranks of $\mathbf{o}$. Expressing the $\mathbf{w}$ and the Kroneker-delta as rank-1 tensors and rearranging the terms yields

$$\mathsf{M}_{\boldsymbol{mi},\boldsymbol{lq}} \approx \sum_{\boldsymbol{s}} \prod_{k=1}^{3} \underbrace{\left(\sum_{j_k} \mathsf{g}^{(\mathbf{o},k)}_{s_{k-1} j_k s_k} w^{(k)}_{j_k} b_{m_k}(y^{\mathrm{Q},k}_{j_k}) b_{l_k}(y^{\mathrm{Q},k}_{j_k}) \right)}_{\mathbf{g}^{(\mathbf{M},k)},\leqslant 3} \left(\prod_{k=4}^{3+N_\mathrm{p}} \underbrace{\mathsf{g}^{(\mathbf{o},k)}_{s_{k-1} i_{k-3} s_k} \delta^{q_{k-3}}_{i_{k-3}}}_{\mathbf{g}^{(\mathbf{M},k)},k>3} \right), \tag{6.38}$$

where $\{\mathbf{g}^{(\mathbf{M},k)}\}_{k=1}^{3+N_\mathrm{p}}$ can be identified as the TT cores of the mass operator. The stiffness tensor-matrix can be constructed in the TT format in a similar way, such that

$$\begin{aligned} \mathsf{S}_{\boldsymbol{mi},\boldsymbol{lq}} &= \delta^{\boldsymbol{i}}_{\boldsymbol{q}} \int_{[0,1]^3} \nabla b_{\boldsymbol{m}}(\boldsymbol{y})^\top \boldsymbol{K}(\boldsymbol{y}, \boldsymbol{\theta}^{\mathrm{C}}_{\boldsymbol{i}}) \nabla b_{\boldsymbol{l}}(\boldsymbol{y}) \hat{\kappa}(\boldsymbol{y}, \boldsymbol{\theta}^{\mathrm{C}}_{\boldsymbol{i}}) \mathrm{d}\boldsymbol{y} \\ &\approx \delta^{\boldsymbol{i}}_{\boldsymbol{q}} \sum_{\alpha,\beta=1}^{3} \sum_{\boldsymbol{j}} \mathsf{w}_{\boldsymbol{j}} \partial_{y_\alpha} b_{\boldsymbol{m}}(y^{\mathrm{Q},1}_{j_1}, y^{\mathrm{Q},2}_{j_2}, y^{\mathrm{Q},3}_{j_3}) \partial_{y_\beta} b_{\boldsymbol{l}}(y^{\mathrm{Q},1}_{j_1}, y^{\mathrm{Q},2}_{j_2}, y^{\mathrm{Q},3}_{j_3}) \mathsf{k}^{(\alpha,\beta)}_{\boldsymbol{ji}} \hat{\mathsf{k}}_{\boldsymbol{ji}}, \end{aligned} \tag{6.39}$$

where $\mathsf{k}^{(\alpha,\beta)}_{\boldsymbol{ji}} = K_{\alpha\beta}(y^{\mathrm{Q},1}_{j_1}, y^{\mathrm{Q},2}_{j_2}, y^{\mathrm{Q},3}_{j_3}, \theta^{\mathrm{C},1}_{i_1}, ..., \theta^{\mathrm{C},N_\mathrm{p}}_{i_{N_\mathrm{p}}})$, $\hat{\mathsf{k}}_{\boldsymbol{ji}} = \hat{\kappa}(y^{\mathrm{Q},1}_{j_1}, y^{\mathrm{Q},2}_{j_2}, y^{\mathrm{Q},3}_{j_3}, \theta^{\mathrm{C},1}_{i_1}, ..., \theta^{\mathrm{C},N_\mathrm{p}}_{i_{N_\mathrm{p}}})$ and ∂_{y_α} is the partial derivative with respect to y_α. For the first tensor $\mathbf{k}$, we assume for now that it has a low rank TT

representation which will be given in detail in the following section. The construction of the latter tensor can be done directly in the TT format by either decomposing the full tensor (if computationally tractable) or using the TT cross approximation method [DS19]. The stiffness can be expressed as the sum of 9 tensor operators:

$$\mathsf{S}_{\boldsymbol{mi},\boldsymbol{lq}} = \sum_{\alpha,\beta=1}^{3} \mathsf{S}^{(\alpha,\beta)}_{\boldsymbol{mi},\boldsymbol{lq}}, \tag{6.40}$$

where the TT representations can be obtained similarly to the mass operator for $k \in 1,2,3$

$$\mathsf{g}^{(\mathsf{S}^{(\alpha,\beta)},k)}_{s_{k-1}m_k l_k s_k} = \begin{cases} \sum_{j_k} w^{(k)}_{j_k} b'_{m_k}(y^{\mathrm{Q},k}_{j_k}) b'_{l_k}(y^{\mathrm{Q},k}_{j_k}) \mathsf{g}^{(\mathbf{k}^{(\alpha,\beta)}\hat{\mathbf{k}},k)}_{s_{k-1}j_k s_k}, & k=\alpha, k=\beta, \\ \sum_{j_k} w^{(k)}_{j_k} b'_{m_k}(y^{\mathrm{Q},k}_{j_k}) b_{l_k}(y^{\mathrm{Q},k}_{j_k}) \mathsf{g}^{(\mathbf{k}^{(\alpha,\beta)}\hat{\mathbf{k}},k)}_{s_{k-1}j_k s_k}, & k=\alpha, k\neq\beta, \\ \sum_{j_k} w^{(k)}_{j_k} b_{m_k}(y^{\mathrm{Q},k}_{j_k}) b'_{l_k}(y^{\mathrm{Q},k}_{j_k}) \mathsf{g}^{(\mathbf{k}^{(\alpha,\beta)}\hat{\mathbf{k}},k)}_{s_{k-1}j_k s_k}, & k\neq\alpha, k=\beta, \\ \sum_{j_k} w^{(k)}_{j_k} b_{m_k}(y^{\mathrm{Q},k}_{j_k}) b_{l_k}(y^{\mathrm{Q},k}_{j_k}) \mathsf{g}^{(\mathbf{k}^{(\alpha,\beta)}\hat{\mathbf{k}},k)}_{s_{k-1}j_k s_k}, & k\neq\alpha, k\neq\beta, \end{cases} \tag{6.41}$$

and for $k > 3$

$$\mathsf{g}^{(\mathsf{S}^{(\alpha,\beta)},k)}_{s_{k-1}m_k l_k s_k} = \mathsf{g}^{(\mathbf{k}^{(\alpha,\beta)}\hat{\mathbf{k}},k)}_{s_{k-1}i_{k-3}s_k} \delta^{q_{k-3}}_{i_{k-3}}, \tag{6.42}$$

where $\mathbf{g}^{(\mathbf{k}^{(\alpha,\beta)}\hat{\mathbf{k}},k)}$ are the TT cores of the product $\mathbf{k}^{(\alpha,\beta)}\hat{\mathbf{k}}$. Due to the summation over the indices α,β, the assembly complexity of the stiffness is higher than for the mass tensor, due to the fact that additional rounding operations must be performed. Moreover, as will be shown in the following section, the construction of the tensor $\mathbf{k}$ requires elementwise inversion and thus no analytical rank bound can be derived. During the construction process, it can be seen that the 4d cores of the operators have band diagonal structure. We therefore employ this property in order to decrease the complexity of the additions and rounding operations.

6.4.3. Construction of the metric tensors

The computation of the TT representations from the previous section is independent of the choice of the discretization (NURBS or B-splines). The parametrization information is included in the tensors $\mathbf{o}$ and $\mathbf{k}$. If the control points of the geometry discretization are given in the TT format as described in Section 6.4.1, then the tensor $\mathbf{o}$ can also be represented in the TT format as well, such that

$$\mathsf{o}_{\boldsymbol{ji}} = \omega\left(y^{\mathrm{Q},1}_{j_1}, y^{\mathrm{Q},2}_{j_2}, y^{\mathrm{Q},3}_{j_3}, \boldsymbol{\theta}^{\mathrm{C}}_{\boldsymbol{i}}\right) = \sum_{\text{perm. }\sigma} \operatorname{sgn}(\sigma) \prod_{s=1}^{3} \left(D_{\boldsymbol{y}} G(\boldsymbol{y}^{\mathrm{Q}}_{\boldsymbol{j}}, \boldsymbol{\theta}^{\mathrm{C}}_{\boldsymbol{i}})\right)_{s\sigma(s)} = \sum_{\text{perm. }\sigma} \operatorname{sgn}(\sigma) \prod_{s=1}^{3} \frac{\partial G_s(\boldsymbol{y}^{\mathrm{Q}}_{\boldsymbol{j}}, \boldsymbol{\theta}^{\mathrm{C}}_{\boldsymbol{i}})}{\partial y_{\sigma(s)}}, \tag{6.43}$$

where $\sigma(s)$ are permutations of the tuple $(1,2,3)$ and $D_{\boldsymbol{y}}$ is the Jacobian matrix. In this case, G is represented as a linear combination of B-spline shape functions, the tensor $\mathbf{o}$ has the form

$$\mathsf{o}_{\boldsymbol{ji}} = \sum_{\text{perm. }\sigma} \operatorname{sgn}(\sigma) \prod_{s=1}^{3} \left((\partial_{\sigma(s)}\mathbf{B})\mathbf{p}_{s:\boldsymbol{i}}\right)_{\boldsymbol{j}}, \tag{6.44}$$

where $\left(\partial_{\sigma(s)}\mathbf{B}\right)_{\boldsymbol{j},\boldsymbol{i}} = \partial y_{\sigma(s)} b_{\boldsymbol{i}}(y_{j_1}^{\mathsf{Q},1}, y_{j_2}^{\mathsf{Q},2}, y_{j_3}^{\mathsf{Q},3})$ are the components of a rank-$\mathbf{1}$ TT-operator. The TT rank of the tensor is in this case bounded by $6r^3$, where r is the maximum rank of the geometry parametrization. In practice, however, a lower TT rank is observed for common domains.

In case of a NURBS parametrization defined by the control points $\mathbf{p}$ and the weights $\mathbf{w}$, the geometry mapping has the form

$$G_s(\boldsymbol{y}_{\boldsymbol{j}}^{\mathsf{Q}}, \boldsymbol{\theta}_{\boldsymbol{i}}^{\mathsf{C}}) = \sum_{\boldsymbol{m}} \mathsf{p}_{s\boldsymbol{m}\boldsymbol{i}} \frac{\mathsf{w}_{\boldsymbol{m}\boldsymbol{i}} b_{m_1}^{(1)}(y_{j_1}^{\mathsf{Q},1}) b_{m_2}^{(2)}(y_{j_2}^{\mathsf{Q},2}) b_{m_3}^{(3)}(y_{j_3}^{\mathsf{Q},3})}{\sum_{\boldsymbol{l}} \mathsf{w}_{\boldsymbol{l}\boldsymbol{j}} b_{l_1}^{(1)}(y_{j_1}^{\mathsf{Q},1}) b_{l_2}^{(2)}(y_{j_2}^{\mathsf{Q},2}) b_{l_3}^{(3)}(y_{j_3}^{\mathsf{Q},3})} = \frac{\mathsf{a}_{s\boldsymbol{j}\boldsymbol{i}}}{\mathsf{d}_{\boldsymbol{j}\boldsymbol{i}}}, \tag{6.45}$$

with the tensors $\mathbf{a}$ and $\mathbf{d}$ defined as

$$\mathbf{a} = (\mathbf{p} \odot (\mathbf{1} \otimes \mathbf{w})) \times_2 \boldsymbol{B}^{(1)} \times_3 \boldsymbol{B}^{(3)} \times_4 \boldsymbol{B}^{(3)}, \tag{6.46}$$

$$\mathbf{d} = \mathbf{w} \times_1 \boldsymbol{B}^{(1)} \times_2 \boldsymbol{B}^{(3)} \times_3 \boldsymbol{B}^{(3)}, \tag{6.47}$$

where $B_{\alpha\beta}^{(k)} = b_{\beta}^{(k)}(y_{\alpha}^{\mathsf{Q},k})$, $\mathbf{1} \in \mathbb{R}^3$, is the one vector and $\times_k$ is the n-mode product. By applying the chain rule on G_s, the required derivatives for building the tensor $\mathbf{o}$ are given by

$$\frac{\partial G_s(\boldsymbol{y}_{\boldsymbol{j}}^{\mathsf{Q}}, \boldsymbol{\theta}_{\boldsymbol{i}})}{\partial y_k} = \frac{(\partial_{y_k}\mathsf{a})_{s\boldsymbol{j}\boldsymbol{i}}\mathsf{d}_{\boldsymbol{j}\boldsymbol{i}} - \mathsf{a}_{s\boldsymbol{j}\boldsymbol{i}}(\partial_{y_k}\mathsf{d})_{\boldsymbol{j}\boldsymbol{i}}}{\mathsf{d}_{\boldsymbol{j}\boldsymbol{i}}^2}, \tag{6.48}$$

where $\partial_{y_k}\mathbf{a}$ and $\partial_{y_k}\mathbf{d}$ denote the tensors obtained by applying the derivative to the k-th basis. In order to obtain a TT representation for this case, the elementwise inversion of the tensor $\mathbf{d}$ has to be performed in the TT format. If the weights are a rank-1 tensor, this process is trivial, since the elementwise inverse is also a rank-1 tensor. Otherwise, the AMEn algorithm has to be used to perform this operation.

One further point is the construction of the tensor $\mathbf{k}$ from (6.39). The tensors are obtained by evaluating the metric matrix over the joint quadrature-collocation grid

$$\boldsymbol{K}(\boldsymbol{y}_{\boldsymbol{j}}^{\mathsf{Q}}, \boldsymbol{\theta}_{\boldsymbol{i}}^{\mathsf{C}}) = D_{\boldsymbol{y}} G(\boldsymbol{y}_{\boldsymbol{j}}^{\mathsf{Q}}, \boldsymbol{\theta}_{\boldsymbol{i}})^{-\top} D_{\boldsymbol{y}} G(\boldsymbol{y}_{\boldsymbol{j}}^{\mathsf{Q}}, \boldsymbol{\theta}_{\boldsymbol{i}}^{\mathsf{C}})^{-1} \omega(\boldsymbol{y}_{\boldsymbol{j}}^{\mathsf{Q}}, \boldsymbol{\theta}_{\boldsymbol{i}}^{\mathsf{C}}). \tag{6.49}$$

Using Cramer's rule [SSSS93] for 3×3 matrices, the inverse can be written in terms of the cofactor matrix and the determinant

$$D_{\boldsymbol{y}} G(\boldsymbol{y}_{\boldsymbol{j}}^{\mathsf{Q}}, \boldsymbol{\theta}_{\boldsymbol{i}}^{\mathsf{C}})^{-\top} = \frac{\left(\mathsf{h}_{\boldsymbol{j}\boldsymbol{i}}^{(\alpha,\beta)}\right)_{\alpha,\beta=1}^{3}}{\mathsf{o}_{\boldsymbol{j}\boldsymbol{i}}}. \tag{6.50}$$

Multiplying the transpose of the inverse with the inverse yields in this case the following expression for the tensor

$$\mathsf{k}_{\boldsymbol{j}\boldsymbol{i}}^{(\alpha,\beta)} = \frac{\mathsf{h}_{\boldsymbol{j}\boldsymbol{i}}^{(\alpha,1)}\mathsf{h}_{\boldsymbol{j}\boldsymbol{i}}^{(\beta,1)} + \mathsf{h}_{\boldsymbol{j}\boldsymbol{i}}^{(\alpha,2)}\mathsf{h}_{\boldsymbol{j}\boldsymbol{i}}^{(\beta,2)} + \mathsf{h}_{\boldsymbol{j}\boldsymbol{i}}^{(\alpha,3)}\mathsf{h}_{\boldsymbol{j}\boldsymbol{i}}^{(\beta,3)}}{\mathsf{o}_{\boldsymbol{j}\boldsymbol{i}}}. \tag{6.51}$$

If the elements of the Jacobian are represented as tensors in the TT format, computing the low-rank representation of $\mathbf{k}^{(\alpha,\beta)}$ requires multiplications and elementwise inversions, where again the AMEn algorithm can be used. Having discussed the construction of the tensors $\mathbf{o}$ and $\mathbf{k}$, the detailed procedure to construct the stiffness in the TT format is presented in Algorithm 10.

Algorithm 10 Construction of the stiffness operator in the TT format (for both $d = 2$ and $d = 3$).

Input: Univariate B-spline bases $\{b_i^{(k)}\}_i$, $k = 1, \dots, d$, TT representation of the control points $\{\mathbf{g}^{(\mathsf{p},k)}\}_{k=1}^{4+N_p}$ (and weights for the NURBS case), tensor product collocation grid for the parameter space $\Theta = \left\{\theta_{i_1}^{\mathrm{C},1}\right\}_{i_1=1}^{\ell_1} \times \left\{\theta_{i_2}^{\mathrm{C},2}\right\}_{i_2=1}^{\ell_2} \times \cdots \times \left\{\theta_{i_{N_\mathrm{p}}}^{\mathrm{C},N_\mathrm{p}}\right\}_{i_{N_\mathrm{p}}=1}^{\ell_{N_\mathrm{p}}}$, coefficient function $\hat{\kappa}$, rounding accuracy ϵ.
Construct the quadrature grid $\left\{\theta_{i_1}^{\mathrm{Q},1}\right\}_{i_1} \times \cdots \times \left\{\theta_{i_{N_\mathrm{p}}}^{\mathrm{C},N_\mathrm{p}}\right\}_{i_d}$ tailored to the B-splines.
Compute $\mathbf{h}^{(\alpha,\beta)}$, $\alpha, \beta = 1, \dots, d$ in the TT format.
Use cross approximation to evaluate $\hat{\mathbf{k}}$.
Compute $\mathbf{o}$ in the TT format according to (6.43).
Initialize $\mathbf{S} \leftarrow 0$ in the TT format.
for $\alpha = 1, \dots, d$ **do**
    for $\beta = 1, \dots, d$ **do**
        Perform the elementwise division from (6.51) in the TT format.
        Multiply $\mathbf{k}$ and $\hat{\mathbf{k}}$ in the TT format.
        for $k = 1, \dots, d$ **do**
            Assemble $\mathbf{g}^{(\mathbf{S}^{(\alpha,\beta)},k)}$ using (6.41).
        for $k = d+1, \dots, d+N_p$ **do**
            Assemble $\mathbf{g}^{(\mathbf{S}^{(\alpha,\beta)},k)}$ using (6.42).
        Perform $\mathbf{S} \leftarrow \mathbf{S} + \mathbf{S}^{(\alpha,\beta)}$ in the TT format.
        Round $\mathbf{S}$ with relative accuracy ϵ.
Output: The TT representation of the stiffness $\{\mathbf{g}^{(\mathbf{S},k)}\}_{k=1}^{d+N_p}$.

6.4.4. Enforcing the boundary conditions

In the presented framework, we consider that the image of a face of the reference hypercube is either a Dirichlet or a Neumann boundary for every $\boldsymbol{\theta} \in \Xi$. The boundary conditions are applied by enforcing the values of the DoFs that contribute to the approximation along the boundary. For a B-spline as presented in Appendix A, the only basis elements that are nonzero at the interval boundaries are the first and the last. For a univariate basis of size n, the discrete projection operator $\boldsymbol{M} = \mathrm{diag}(0, 1, ..., 1, 0)$ is defined in order to enforce the DoFs corresponding to the boundaries to be 0. If however only the coefficient of the first (or the last) basis has to be set to 0, the diagonal matrix has the form $\mathrm{diag}(0, 1, ..., 1)$ ($\mathrm{diag}(1, ..., 1, 0)$ for the last basis). If no restriction on the DoFs is required, the projection is the identity matrix.

When enforcing the boundary conditions on the BVP (6.2), the operator is constructed using the Kronecker product as

$$\mathbf{P}^{\mathrm{in}} = \boldsymbol{M}^{(1)} \otimes \cdots \otimes \boldsymbol{M}^{(d)} \otimes \boldsymbol{I}_{\boldsymbol{\ell}}, \tag{6.52}$$

where the matrices $\boldsymbol{M}^{(k)}$ are defined as

$$\boldsymbol{M}^{(k)} = \begin{cases} \mathrm{diag}(0, 1, ..., 1, 0), & \text{Dirichlet BC for both facets } y_k = 0 \text{ and } y_k = 1 \\ \mathrm{diag}(0, 1, ..., 1), & \text{Dirichlet BC for the facet } y_k = 0, \text{ Neumann BC for the facet } y_k = 1, \\ \mathrm{diag}(1, ..., 1, 0), & \text{Neumann BC for the facet } y_k = 0, \text{ Dirichlet BC for the facet } y_k = 1, \\ \mathrm{diag}(1, ..., 1), & \text{Neumann BC for both facets } y_k = 0 \text{ and } y_k = 1. \end{cases} \tag{6.53}$$

Its complement $\mathbf{P}^{bd}$ only selects the DoFs that only affect the Dirichlet boundaries:

$$\mathbf{P}^{\text{bd}} = \mathbf{I} - \mathbf{P}^{\text{in}}. \tag{6.54}$$

With the discrete projection operators, the BVP (6.2) can be written as the system

$$\left(\mathbf{P}^{\text{in}}\left(\mathbf{S} + \rho\mathbf{M}\right) + \alpha\mathbf{P}^{\text{bd}}\right)\mathbf{u} = \mathbf{P}^{\text{in}}\mathbf{f} + \alpha\mathbf{P}^{\text{bd}}\mathbf{f}^{\text{D}}, \tag{6.55}$$

where $\alpha \neq 0$ is a constant. The tensor $\mathbf{f}^{\text{D}}$ approximates the prescribed Dirichlet conditions along the boundaries $\Gamma_D(\boldsymbol{\theta})$, such that

$$g(\boldsymbol{x}, \boldsymbol{\theta}) \approx \sum_{\boldsymbol{k}} \mathrm{f}^{\text{D}}_{\boldsymbol{ki}} b_{\boldsymbol{k}}(G^{-1}(\boldsymbol{x}, \boldsymbol{\theta})), \quad \forall \boldsymbol{\theta} \in \Xi, \boldsymbol{x} \in \Gamma_D(\boldsymbol{\theta}). \tag{6.56}$$

The choice of the constant α is arbitrary, however, it affects the condition number of the final tensor-operator and the convergence of the iterative solver. The choice of the constant in this work is $\alpha = 1/n$, where n is the maximum size of the univariate B-spline bases. For a more detailed derivation of imposing the boundary conditions as well as the interface conditions between several IGA patches we refer to [LMMT14].

6.4.5. Quantized tensor train decomposition

One way to speed up the computations in the TT format is to use the QTT format presented in Section 3.2.3. In many cases, the prior reshaping of the tensors increase the efficiency of the linear algebra operations (especially for the AMEn solver) [Kho11, KO10, IWL+21]. Therefore, the QTT format is used during the construction of the stiffness tensor-operator (especially for the elementwise inversion of $\mathbf{o}$). Since the choice of the univariate bases is not always a power of 2, the reshaping is performed using the prime factorization of the modes. Additionally, the QTT format is also used when solving the multilinear system (6.28).

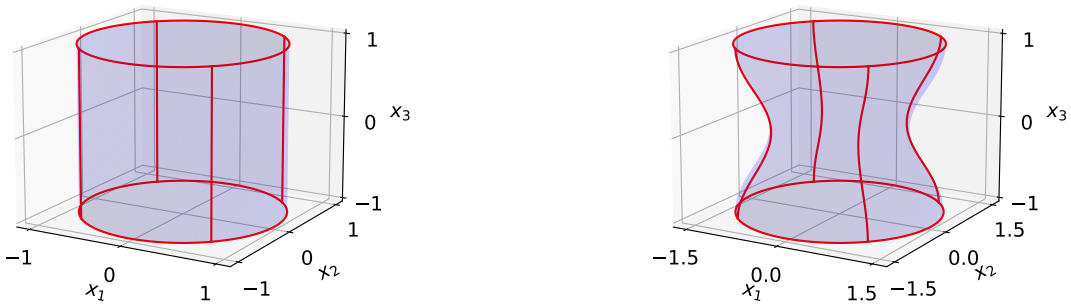
6.5. Numerical experiments

In the rest of the chapter, numerical investigations are performed to showcase the performance of the presented TT-IGA framework. The proposed method is also compared against other well established methods (FEM). First, a convergence study is performed to assess the correctness of the solver. Next, the efficiency of the TT solver as well as the operator construction is studied for an increasing number of parameters. The remaining examples address solving inverse problems with the help of a surrogate model obtained from the TT-IGA solver. One example is a section of a quadrupole model (2d magnetostatic problem) and the second example is a waveguide structure (3d scalar Helmholtz equation). The third test case concerns a parameter dependent material jump within the computational domain. All presented results are run on a standard workstation. The presented framework is implemented in the `Python` programming language in the form of the `Python` package `tt-iga`[1]. For the multilinear algebra operations in the TT format, the `Python` package `torchtt`[2] is used. Both packages have been developed by the author.

6.5.1. Convergence study

The first numerical test is a convergence study to assess the behavior of the error while refining the discretization. Along with the error of the TT-IGA solution, the computational cost is also investigated in

[1] `https://github.com/ion-g-ion/tt-iga`
[2] `https://github.com/ion-g-ion/torchTT`



(a) Initial geometry ($\theta = 0$).

(b) Fully deformed geometry ($\theta = 1$).

Figure 6.5.: Deformed cylinder (parametrization is given in (6.57)). Figures adapted from [ILDG22].

terms of storage needs, solver runtime and assembly time for the system operators. For this study, the computational domain is a cylinder which is deformed by varying a single parameter. In Figure 6.5, the initial and the fully deformed geometry are shown. The geometry is given by the parametrization

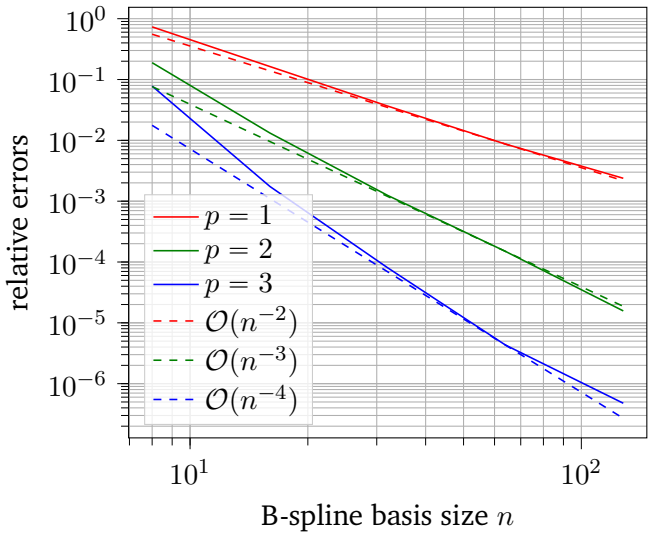
$$G(\boldsymbol{y}, \theta) = \begin{pmatrix} (2y_1 - 1)\sqrt{1 - \frac{(2y_2-1)^2}{2}\left(\frac{1-\cos((2y_3-1)\pi)}{2}\theta + 1\right)} \\ (2y_2 - 1)\sqrt{1 - \frac{(2y_1-1)^2}{2}\left(\frac{1-\cos((2y_3-1)\pi)}{2}\theta + 1\right)} \\ 2y_3 - 1 \end{pmatrix}, \quad y_1, y_2, y_3, \theta \in [0, 1]. \tag{6.57}$$

Using the method of manufactured solutions, the following Laplace BVP problem is solved over the parameter dependent domain $D(\boldsymbol{\theta})$:

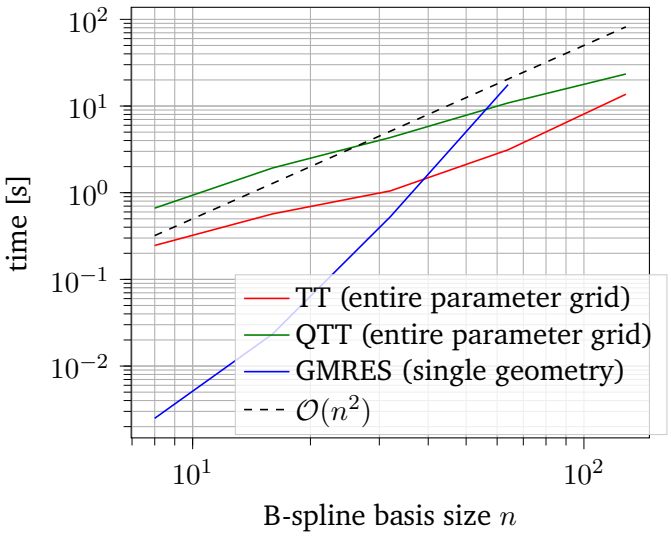
$$\Delta u(\boldsymbol{x}, \theta) = 0, \qquad \boldsymbol{x} \in D(\theta), \tag{6.58a}$$
$$u(\boldsymbol{x}, \theta) = \cos(3x_1)\cos(4x_2)\exp(-5x_3), \qquad \boldsymbol{x} \in \partial D(\theta), \tag{6.58b}$$

with the analytical solution represented in the physical domain $u(\boldsymbol{x}, \theta) = \cos(3x_1)\cos(4x_2)\exp(-5x_3)$. The geometry approximation is done using B-splines which are chosen to be identical (same degree and same knots) for all 3 dimensions with the degree p and the size n, such that $\boldsymbol{n} = (n, n, n)$.

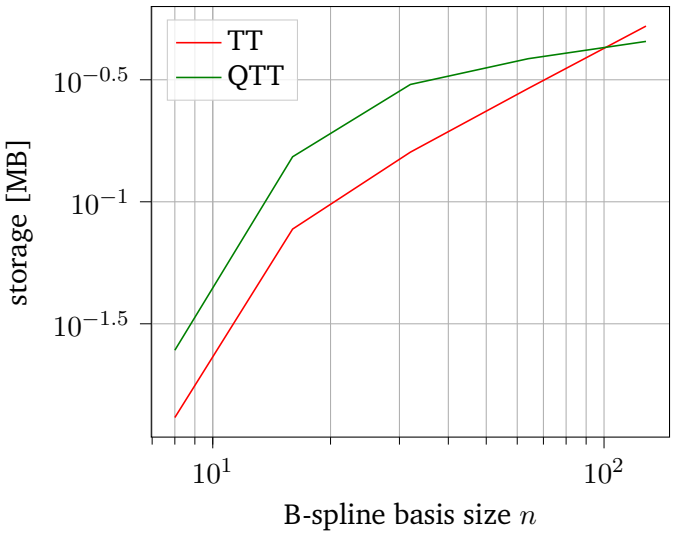
The first part of the study concerns the convergence with respect to refining the B-spline bases while keeping the size of the collocation grid fixed to $\ell = 8$. In this case, the expected convergence order for the L^2 error is $\mathcal{O}(n^{-(p+1)})$. In Figure 6.6a, the convergence of the TT-IGA solver is shown for gradually refined B-spline bases for $p \in \{1, 2, 3\}$ (linear, quadratic, cubic). For verification, the $\mathcal{O}(n^{-(p+1)})$ lines are also plotted. In the Figures 6.6b and 6.6c, the QTT format is compared against the TT format and the conventional GMRES solver. To this end, only quadratic B-splines are used, however, the results are however similar for linear and cubic B-splines. As seen in Figure 6.6b, the runtime of the AMEn solver scales better (similar to $\mathcal{O}(n^2)$) with the size of the basis, compared to using the GMRES for the linear system. Even though the system matrix is stored in sparse format, the computational complexity of GMRES solver equals the complexity of storing the solution in the full format ($\mathcal{O}(n^3)$). Despite scaling worse than the AMEn, the GMRES is faster for small values of n. As a note, the GMRES solver is used for resolving one single parameter, while the TT-IGA solver offers the solution for $\ell = 8$ parameters at once. Regarding the



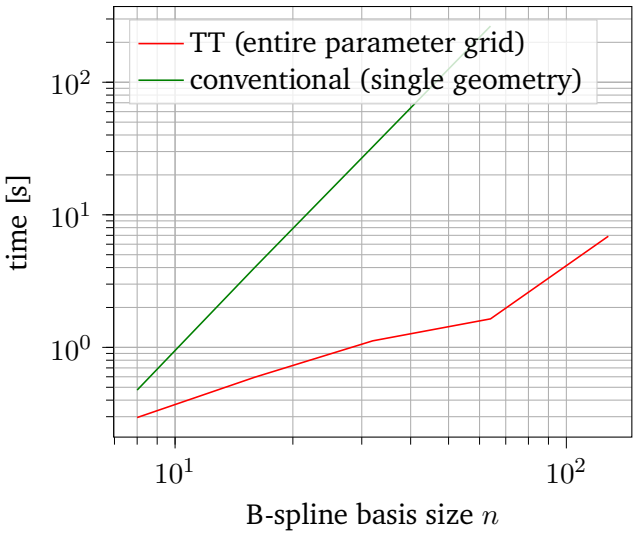
(a) Convergence study for $p \in \{1, 2, 3\}$ and $n \in \{8, 16, 32, 64, 128\}$.



(b) Comparison between the AMEn and GMRES solvers in terms of runtime for increasing n.



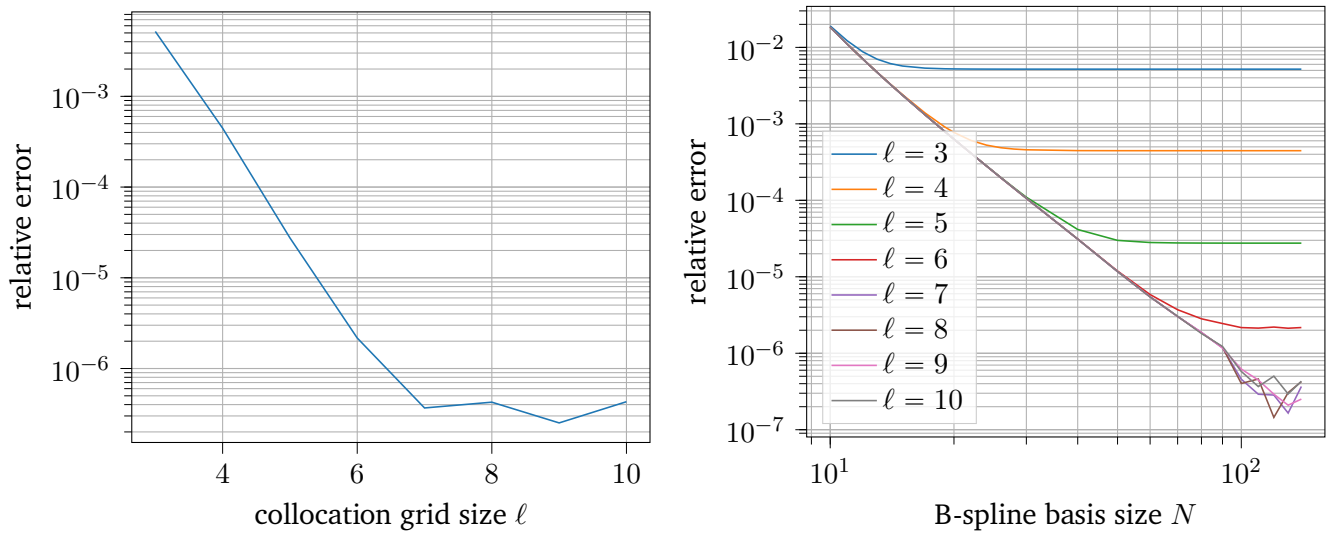
(c) Storage requirements for the solution for increasing n ($p = 2$).



(d) Assembly time of the stiffness operator as function of n ($p = 2$).

Figure 6.6.: Performance of the TT-IGA solver in terms of convergence, runtime (solver as well as stiffness assembler) and memory consumption for a fixed parameter grid ($\ell = 8$). Figures adapted from [ILDG22].

choice between the TT and the QTT format, it has been observed that both have a similar increase in the complexity. However, the TT is slightly faster for this example. In both cases, it has been observed that the TT-rank stagnates after a certain n. With respect to memory consumption, both TT and QTT achieve a very good compression ratio, however, the QTT format uses less memory for storing the TT-cores after the dimension of the univariate bases exceeds $n = 60$. As already mentioned in this chapter, the assembly of



(a) Solution convergence as function of the size of the collocation grid ℓ (cubic B-spline basis with fixed size $n = 128$).

(b) Solution convergence with respect to n for different values of ℓ.

Figure 6.7.: Convergence of the solution for increasing values of ℓ. Figures adapted from [ILDG22].

the mass and stiffness operators is a computationally expensive process in IGA-FEM. Even iterating through all the DoF in order to populate the matrix has the complexity $\mathcal{O}(n^3)$. The TT-IGA assembly routine is benchmarked against the naive integration as shown in Figure 6.6d. In this case, the proposed low-rank method is orders of magnitude faster with a better asymptotic complexity ($\mathcal{O}(n^2)$ compared to $\mathcal{O}(n^3)$ for the classical assembler). Note that the results for the TT based method are for the entire parameter grid while the conventional assembler is used for only one parameter realization.

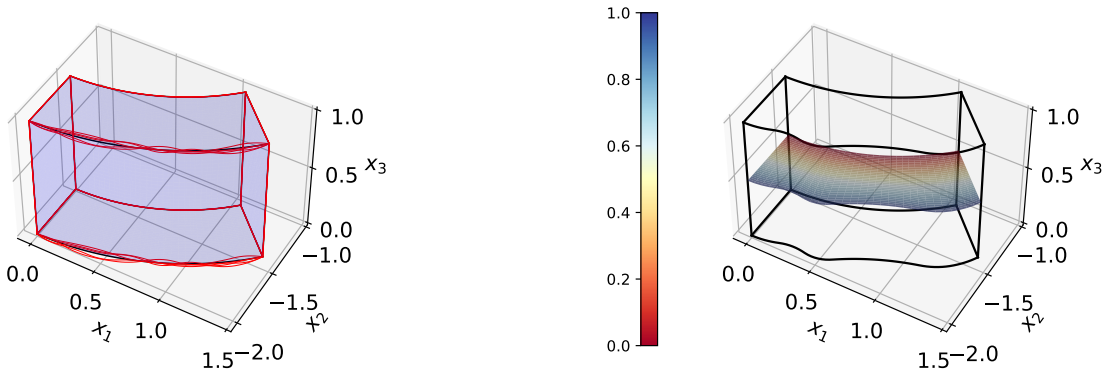
In the error estimate from (6.29), the error of the parameter space discretization is also accounted for. In the rest of the section, the convergence of the parameter dependent solution with respect to the size ℓ of the collocation grid is investigated. In Figure 6.7a, the exponential decrease of the error is observed for an increasing ℓ for fixed cubic B-spline basis of $n = 128$. Since the global error is bounded by both the IGA and the parameter space discretization, it can be seen that the IGA discretization becomes the limiting factor for $\ell \geqslant 7$. This behavior is also shown in Figure 6.7b, where the B-splines are refined for several values of ℓ.

6.5.2. Performance of the TT-IGA method for multiple parameter dependencies

In the previous test case, the geometry was controlled using only one parameter. One natural question that comes into mind is, how does the solver perform for an increasing number of geometry parameters. We use a quarter of a C-shape domain with the inner radius $r_{\text{in}} = 1.5$ and a parameter dependent outer radius

$$r_{\text{out}}(\alpha, \boldsymbol{\theta}) = 2 + \sum_{k=1}^{N_{\text{p}}} \theta_k b_k^{(\text{out})}(\alpha), \quad \theta_k \in [-0.0.5, 0.05], \tag{6.59}$$

where $\alpha \in (0, 1)$ is a scaled angular span, N_{p} is the number of parameters and $\left\{b_k^{(\text{out})}\right\}_{k=1}^{N_{\text{p}}}$ is a B-spline basis. An illustration of the geometry for several parameter realizations is given in Figure 6.8a. The



(a) The geometry for several parameter combinations in the case $N_\mathrm{p} = 8$.

(b) Solution in the $\boldsymbol{y} \in (0,1) \times (0,1) \times \{0.5\}$ plane. The color red corresponds to the maximum value (in this case 1) and the color blue corresponds to the smallest value (in this case 0).

Figure 6.8.: Quarter of a C-shaped domain perturbed outer radius $r_\mathrm{out}(\alpha, \boldsymbol{\theta})$ (both figures are adapted from [ILDG22]).

parametrization of the C-shape quarter with perturbed outer radius is given by

$$G(\boldsymbol{y}, \boldsymbol{\theta}) = \begin{pmatrix} (y_1(r_\mathrm{out}(y_2, \boldsymbol{\theta}) - r_\mathrm{in}) + r_\mathrm{in}) \cos(y_2 \pi/4) \\ (y_1(r_\mathrm{out}(y_2, \boldsymbol{\theta}) - r_\mathrm{in}) + r_\mathrm{in}) \sin(y_2 \pi/4) \\ y_3 \end{pmatrix}, \quad \boldsymbol{y} \in [0,1]^3. \tag{6.60}$$

In the domain $D(\boldsymbol{\theta})$, the following Laplace equation is solved

$$\Delta u(\cdot, \boldsymbol{\theta}) = 0, \quad \text{in } D(\boldsymbol{\theta}), \tag{6.61}$$
$$\partial_{\boldsymbol{\nu}} u = 0, \quad \text{on } \Gamma_\mathrm{N}(\boldsymbol{\theta}), \tag{6.62}$$
$$u = 1, \quad \text{on } \Gamma_{\mathrm{D}^+}, \tag{6.63}$$
$$u = 0, \quad \text{on } \Gamma_{\mathrm{D}^-}, \tag{6.64}$$

where the Dirichlet conditions are imposed along the inner radius $\Gamma_{\mathrm{D}^-} = \{G(\boldsymbol{y}, \boldsymbol{\theta}) : y_1 = 0\}$ and the perturbed outer radius $\Gamma_{\mathrm{D}^+} = \{G(\boldsymbol{y}, \boldsymbol{\theta}) : y_1 = 1\}$, while the normal derivative is along the remaining part of the boundary $\Gamma_\mathrm{N} = \partial D(\boldsymbol{\theta}) \cap \{G(\boldsymbol{y}, \boldsymbol{\theta}) : 0 < y_1 < 1\}$. In Figure 6.8b the solution in the plane corresponding to $\boldsymbol{y} \in (0,1) \times (0,1) \times \{0.5\}$ is plotted for a randomly chosen parameter and $N_\mathrm{p} = 8$. The geometry is represented using quadratic B-splines of fixed size $\boldsymbol{n} = (40, 20, 80)$. The discretization of the parameter space is done using a tensor product of identically sized grids of size $\ell_k = \ell = 8, k = 1, ..., N_\mathrm{p}$.

The BVP is solved for an increasing number of parameters $N_\mathrm{p} \in \{2, 3..., 10\}$. In order to obtain the best performance in terms of computational time and memory consumption, the QTT format is used for the construction of the stiffness tensor-operator. The resulting stiffness is however stored in the TT format. The results are shown in Table 6.1. The AMEn solver is applied to the system in the TT format and the storage requirements are reported without the use of quantization. In addition to the runtime on the central processing unit (CPU), the AMEn solver is also run on the graphics processing unit (GPU)[1], with an increase in speed of about 11 times. The error compared to a reference solution is checked as well, to ensure that it does not explode with the growing number of parameters.

[1]Nvidia Tesla P100

N_p	Assembly time [s]	AMEn solver runtime [s]		Operator storage [MB]	Solution storage [MB]
		CPU	GPU		
2	15.55	1.62	0.75	14.06	0.20
3	22.79	4.59	1.40	31.75	0.46
4	37.30	10.24	2.05	50.21	0.86
5	57.85	27.40	3.92	73.52	1.41
6	83.42	54.05	5.91	94.42	2.12
7	122.92	106.88	10.79	133.41	2.80
8	175.25	188.73	20.85	164.40	4.23
9	255.85	340.97	32.50	199.05	4.97
10	406.98	549.64	49.14	245.35	7.14

Table 6.1.: Computational complexity (runtime as well as storage) for an increasing number of the parameters N_p. The stiffness assembly time and the solver runtime are reported separately. Table data is adapted from [ILDG22].

As seen from the table, the time complexity of the TT-IGA has a slightly exponential behavior, however, the base is smaller than 2 ($\sim 1.6^{N_\mathrm{p}}$ for the stiffness assembly and $\sim 1.7^{N_\mathrm{p}}$ for the solver). The linear complexity with respect to the number of dimensions is not observed because the ranks also depend on the number of parameters. Despite this fact, the TT-IGA solver remains a very efficient approach for dealing with a moderate number of parameters. The quadratic growth of the storage requirement with respect to the mode size is observed for the tensor operators. They require two orders of magnitude more space than the solution of the problem. One way to reduce this is to take advantage of the sparse structure of the TT cores.

6.5.3. Quadrupole section

The model considered in this section is a quadrupole accelerator magnet [DGGI+20]. The model is 2d and the geometry can be found in Figure 6.9. Due to geometry and excitation symmetry, only $1/8$-th of the cross-section is considered (red contour in Figure 6.9). A surrogate model of the BVP is computed in order to be used to solve an inverse problem. The 4 geometry parameters are inferred from field observations in the air gap. The section is divided into two parts: computing the surrogate model (forward problem) and approximating the posterior PDF (inverse problem).

Forward problem

The detailed sketch of the geometry is presented is Figure 6.10a. The computational domain is divided into three subdomains, each corresponding to one of the three different material regions: air, iron and copper. Parametric deformations of the subdomains are also considered (represented in Figure 6.10b):

- Parameter θ_1 corresponds to the radius of the iron yoke's interior surface.
- Parameter θ_2 corresponds to the position along the x_1-axis of the left boundary of the copper region.
- Parameter θ_3 corresponds to the position along the x_1-axis of the right boundary of the copper region.
- Parameter θ_4 corresponds to the height of the copper region.

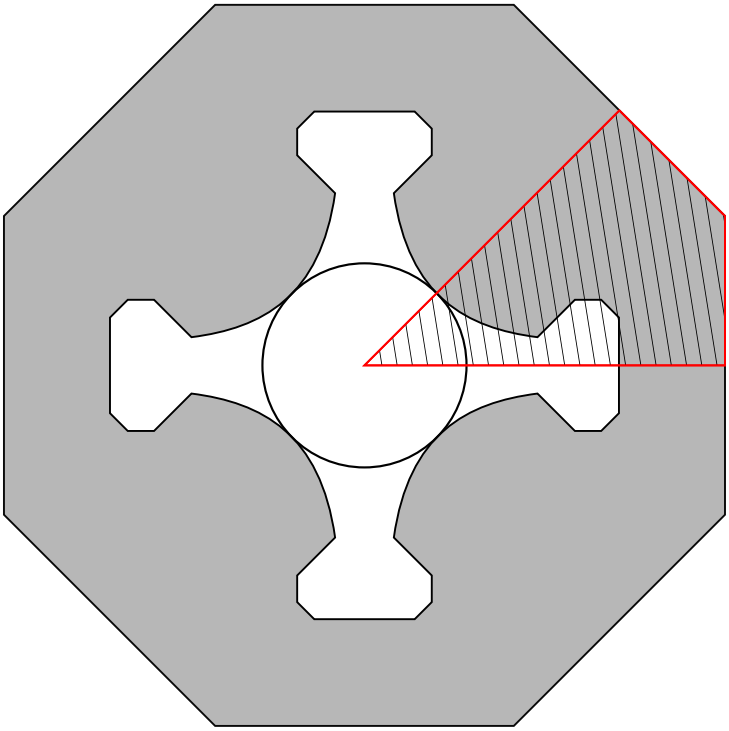


Figure 6.9.: Cross-section of the quadrupole accelerator magnet. Figure from [DGGI+20].

For each parameter, the following bounds are defined: $\theta_k \in [-0.75, 0.75]$ mm, $k \in \{1, 2, 3, 4\}$.

The geometry is represented using a single NURBS patch and the parameter dependence only affects the interior control points, while the image of the parametrization is the same for every parameter realization. The exact geometry description (control points, weights and B-spline basis) is presented in detail in Appendix C. The NURBS parametrization is chosen such that the different material regions correspond to Cartesian subpartitions in the reference domain for every parameter realization $\boldsymbol{\theta} \in \Xi$. The copper subdomain, for example, is given as the image of $[0.4, 0.6] \times [0, 0.5]$, while the air gap is the image of $[0, 0.4] \times [0, 0.3]$. The B-spline basis is chosen to allow for C^0 continuity across the interfaces between the subdomains.

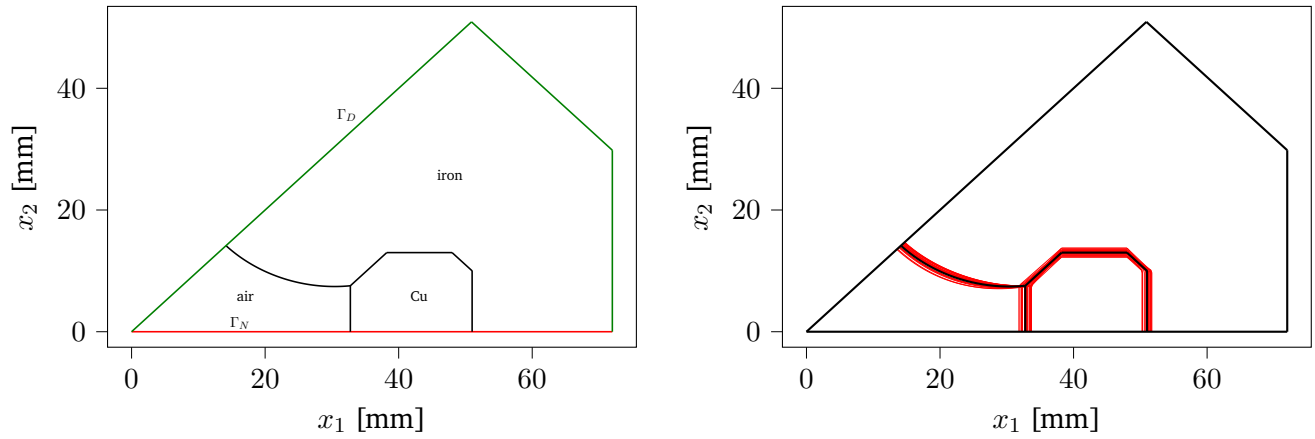
Dealing with a magnetostatic problem, the governing equations are the laws of Gauß and Ampère

$$\nabla \cdot \boldsymbol{B} = \boldsymbol{0}, \tag{6.65}$$
$$\nabla \times \boldsymbol{H} = \boldsymbol{J}, \tag{6.66}$$

where $\boldsymbol{B}$ is the magnetic flux density, $\boldsymbol{H}$ is the magnetic field strength, $\boldsymbol{J}$ is the current density, $\nabla\times$ is the curl operator and $\nabla\cdot$ is the divergence operator. The material relation is assumed to be linear, i.e., $\boldsymbol{H}(\boldsymbol{x}) = \nu(\boldsymbol{x})\boldsymbol{B}(\boldsymbol{x})$, but dependent on the spatial position. In order to derive a single PDE, the magnetic vector potential formulation $\boldsymbol{B} = \nabla \times \boldsymbol{A}$ is used to obtain the curl-curl equation

$$\nabla \times (\nu \nabla \times \boldsymbol{A}) = \boldsymbol{J}. \tag{6.67}$$

A further simplification is obtained when considering that the problem is 2d. In order to fully describe the solution in the x_1x_2-plane, only the third component of the magnetic vector potential is considered, such



(a) The computational domain together with the 3 subdomains.

(b) The subdomain boundaries for 30 different parameter realizations.

Figure 6.10.: Computational domain of the 2d quadrupole model: nominal geometry and the subdomain boundaries for different parameter realizations (red contours).

that $\boldsymbol{A} = (0, 0, A_3)$. The curl-curl equation is therefore reduced to the BVP

$$-\nabla \cdot (\mu(\cdot, \boldsymbol{\theta})^{-1} \nabla A_3(\cdot, \boldsymbol{\theta})) = J_3(\cdot, \boldsymbol{\theta}), \quad \text{in } D, \tag{6.68}$$

$$A_3 = 0, \quad \text{on } \Gamma_D, \tag{6.69}$$

$$\mu(\cdot, \boldsymbol{\theta})^{-1} \partial_{\boldsymbol{\nu}} A_3 = 0, \quad \text{on } \Gamma_N, \tag{6.70}$$

where Γ_D and Γ_N are the Dirichlet and Neumann boundaries of the domain, respectively (see Figure 6.10a). They are motivated by the symmetry of the model as well as by the fact that the permeability of the iron is high enough such that the field exiting the outer boundary can be neglected. The total current flowing through the copper region is set to be $I = 1080$ A (24 turns and a current of 45 A per turn). Due to the choice of the parametrization, the current density has the following expression in the reference domain:

$$\hat{J}(\boldsymbol{y}, \boldsymbol{\theta}) = \begin{cases} \frac{I}{S(\boldsymbol{\theta})}, & 0.4 < y_1 < 0.6 \text{ and } 0 < y_2 < 0.5, \\ 0, & \text{otherwise}, \end{cases} \tag{6.71}$$

where $S(\boldsymbol{\theta})$ is the surface of the copper region. The permeability is similarly expressed in the reference domain as

$$\hat{\mu}(\boldsymbol{y}, \boldsymbol{\theta}) = \begin{cases} \mu_0, & 0.4 < y_1 < 0.6 \text{ and } 0 < y_2 < 0.5 \text{ or } 0 < y_1 < 0.4 \text{ and } 0 < y_2 < 0.3, \\ \mu_0 \mu_r, & \text{otherwise}, \end{cases} \tag{6.72}$$

where μ_0 is the vacuum permeability and $\mu_r = 1500$ is the relative permeability.

The TT-IGA solver is applied on the parameter dependent BVP. Quadratic B-spline bases with the size $\boldsymbol{n} = (64, 64, 64)$ are chosen for the IGA discretization and $\ell = 8$ collocation points per dimension for the parameter space discretization. Constructing the stiffness TT operator up to a relative accuracy of $\epsilon = 10^{-10}$ in the Frobenius norm yields the TT rank $\boldsymbol{r} = (1, 92, 86, 61, 26, 7, 1)$ which corresponds to ≈ 268 MB of storage. The assembly time for the extended stiffness tensor operator is approximately 50 s, compared to 1.5 s needed for a FEM mesh of comparable size for one parameter realization. The AMEn solver is run with a relative residual accuracy of $\epsilon = 10^{-6}$. The solution is plotted for $\boldsymbol{\theta} = \boldsymbol{0}$ in Figure 6.11. The rank of the

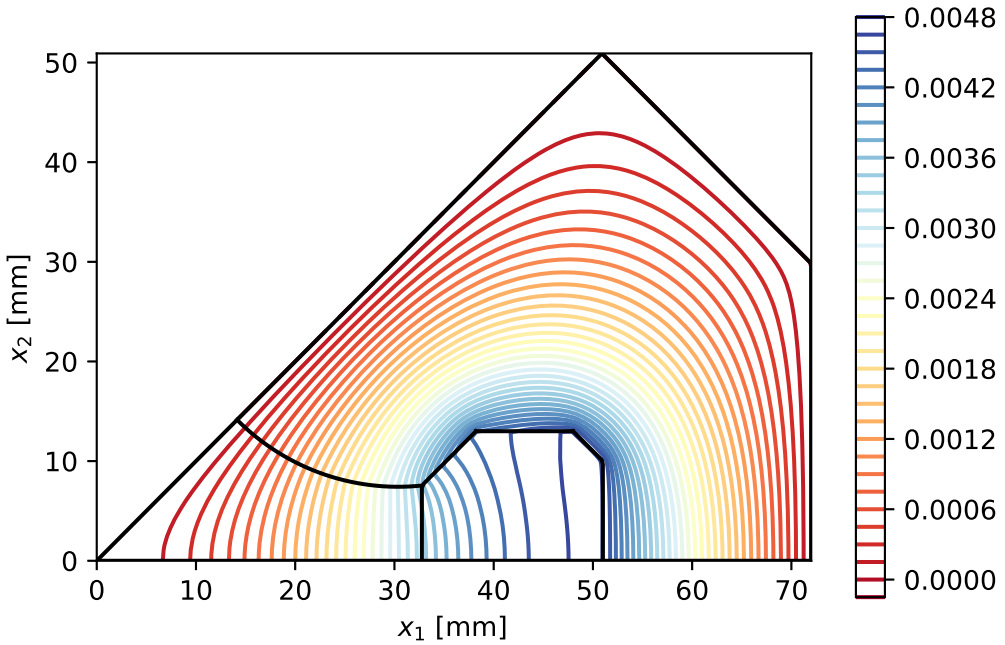


Figure 6.11.: Quadrupole section: solution for $\boldsymbol{\theta} = (0, 0, 0, 0)$. The contour lines of A_3 are plotted.

obtained solution is $\boldsymbol{r} = (1, 63, 141, 123, 54, 8, 1)$ and corresponds to ≈ 6.53 MB of storage. As a comparison, storing a solution as a full tensor for a fixed geometry only requires ≈ 0.032 MB. The relatively increased storage requirement for the 2d problem is caused by the bad choice of the geometry parametrization. The naive choice of the control points leads to high distortions across the domain. Optimization techniques similar to the ones presented in [PC19, JM17b] can be applied to obtain a better IGA parametrization of the domain. The parameter dependent solution obtained from the TT-IGA is validated against a fine grid FEM solution in terms of accuracy for multiple parameter combinations.

Inverse problem

For the inverse problem, we assume that the observed quantity is the A_3 component of the magnetic vector potential. The number of observations is $N_\mathrm{o} = 9$ and they are equidistantly spread along the line $x_1 \in [0, 16]$ mm, $x_2 = 0$ mm. The observations are generated from a high-fidelity FEM simulation for $\hat{\boldsymbol{\theta}} = \mathbf{0}$ and polluted with Gaussian additive noise. In the Bayesian setup, the vector $\boldsymbol{z} \in \mathbb{R}^{N_\mathrm{o}}$ containing the observations as well as the parameter vector $\boldsymbol{\theta}$ are modeled as random variables $\boldsymbol{Z}$ and $\boldsymbol{\vartheta}$, respectively. The additive Gaussian noise model is translated into the following likelihood function

$$p_{\boldsymbol{Z}|\boldsymbol{\vartheta}=\boldsymbol{\theta}}(\boldsymbol{z}) \propto \exp\left(-\frac{1}{2\sigma_\mathrm{n}^2}\sum_{k=1}^{N_\mathrm{o}}(z_k - A_3(\boldsymbol{x}^{(k)}, \boldsymbol{\theta}))^2\right), \tag{6.73}$$

where $\sigma_\mathrm{n} = 5 \cdot 10^{-6}$ is the noise standard deviation and $\boldsymbol{x}^{(k)}$ are the positions where the field is observed. Due to the choice of the parametrization G, the positions of the points $\boldsymbol{x}^{(k)}$ in the reference domain depend on the parameter $\boldsymbol{\theta}$. The dependence is however linear and no inversion of the map G is necessary

to evaluate the TT-IGA solution. Regarding the prior distribution, the parameters are considered to be independent and identically distributed beta RVs with $\alpha = \beta = 4$. Note that the definition domain of the beta PDF has to be scaled to the parameter range.

The posterior

$$p_{\boldsymbol{\vartheta}|\boldsymbol{Z}=\boldsymbol{z}}(\boldsymbol{\theta}) \propto p_{\boldsymbol{Z}|\boldsymbol{\vartheta}=\boldsymbol{\theta}}(\boldsymbol{z}) p_{\boldsymbol{\vartheta}}(\boldsymbol{\theta}), \tag{6.74}$$

is in this case interpolated using a tensor product quadratic B-spline basis defined over the domain Ξ with the size $(50, 25, 25, 25)$. The result is shown in Figure 6.12, where the 2d and 1d marginals of the joint posterior PDF is represented. Furthermore, the first and second order moments of the posterior are computed using numerical integration, which are given as

$$\mathbb{E}(\boldsymbol{\vartheta}) = (-5.2097 \cdot 10^{-6}, -1.1709 \cdot 10^{-6}, -5.2202 \cdot 10^{-9}, 7.3047 \cdot 10^{-9})\ \mathrm{m}, \tag{6.75}$$

$$\mathrm{Cov}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}) = \begin{pmatrix} 1.6124 \cdot 10^{-9} & 5.7094 \cdot 10^{-10} & -8.6323 \cdot 10^{-11} & -1.1170 \cdot 10^{-10} \\ 5.7094 \cdot 10^{-10} & 5.1130 \cdot 10^{-8} & 6.8685 \cdot 10^{-13} & 6.8260 \cdot 10^{-13} \\ -8.6323 \cdot 10^{-11} & 6.8685 \cdot 10^{-13} & 5.1138 \cdot 10^{-8} & -1.2719 \cdot 10^{-13} \\ -1.1170 \cdot 10^{-10} & 6.8260 \cdot 10^{-13} & -1.2719 \cdot 10^{-13} & 5.1138 \cdot 10^{-8} \end{pmatrix} \mathrm{m}^2. \tag{6.76}$$

As it can be observed, the posterior only offers significant information about the parameter θ_1, which corresponds to the radius of the yoke. This parameter has a higher influence on the predicted measurements since it directly affects the distribution of the field along the horizontal line. The remaining parameters govern the surface of the current excitation and have a small contribution to the predicted measurements. As seen from the diagonal dominant structure of the covariance matrix, the parameters are statistically independent of each other.

6.5.4. Helmholtz equation within a waveguide structure

In this test case, a 3d model is studied. Similarly to the previous case, the surrogate model is built and then used to solve an inverse problem.

Forward problem

The geometry in this case is a 3d waveguide structure (see Figure 6.13a). The section is rectangular with the height $h = 0.5$ and the width variable width $w \in [2, 3]$. The volume is created by translating the cross-section along a curve. In Figure 6.13a, the coordinates of some relevant points (blue dots) are given. The three points correspond in the reference domain to the points $y_1 = 0, y_2 = 1, y_3 \in \{0, 0.25, 0.5, 0.75, 1\}$. The computational domain depends on three parameters: $\theta_1 \in [-0.2, 0.2]$ which affects the radius of the circular section and $\theta_2, \theta_3 \in [-0.3, 0.3]$ which determines the height of the inverted truncated pyramid part. In Figure 6.13b, the geometry is represented for several parameter realizations $\boldsymbol{\theta} \in \{-0.2, 0.2\} \times \{-0.3, 0.3\} \times \{-0.3, 0.3\}$. The parameter dependent geometry map that describes the domain is approximated using a B-spline representation similar to the one given in (6.3).

Within the presented computational domain $D(\boldsymbol{\theta})$, the following scalar Helmholtz BVP is solved:

$$\Delta u(\cdot, \boldsymbol{\theta}) + \rho u(\cdot, \boldsymbol{\theta}) = 0, \quad \text{in } D(\boldsymbol{\theta}), \tag{6.77a}$$

$$u(\cdot, \boldsymbol{\theta}) = g(\cdot), \quad \text{on } \partial D(\boldsymbol{\theta}) \cap \{(x_1, x_2, x_3) : x_3 = -3\}, \tag{6.77b}$$

$$u(\cdot, \boldsymbol{\theta}) = 0, \quad \text{on } \partial D(\boldsymbol{\theta}) \backslash \{(x_1, x_2, x_3) : x_3 = -3\}, \tag{6.77c}$$

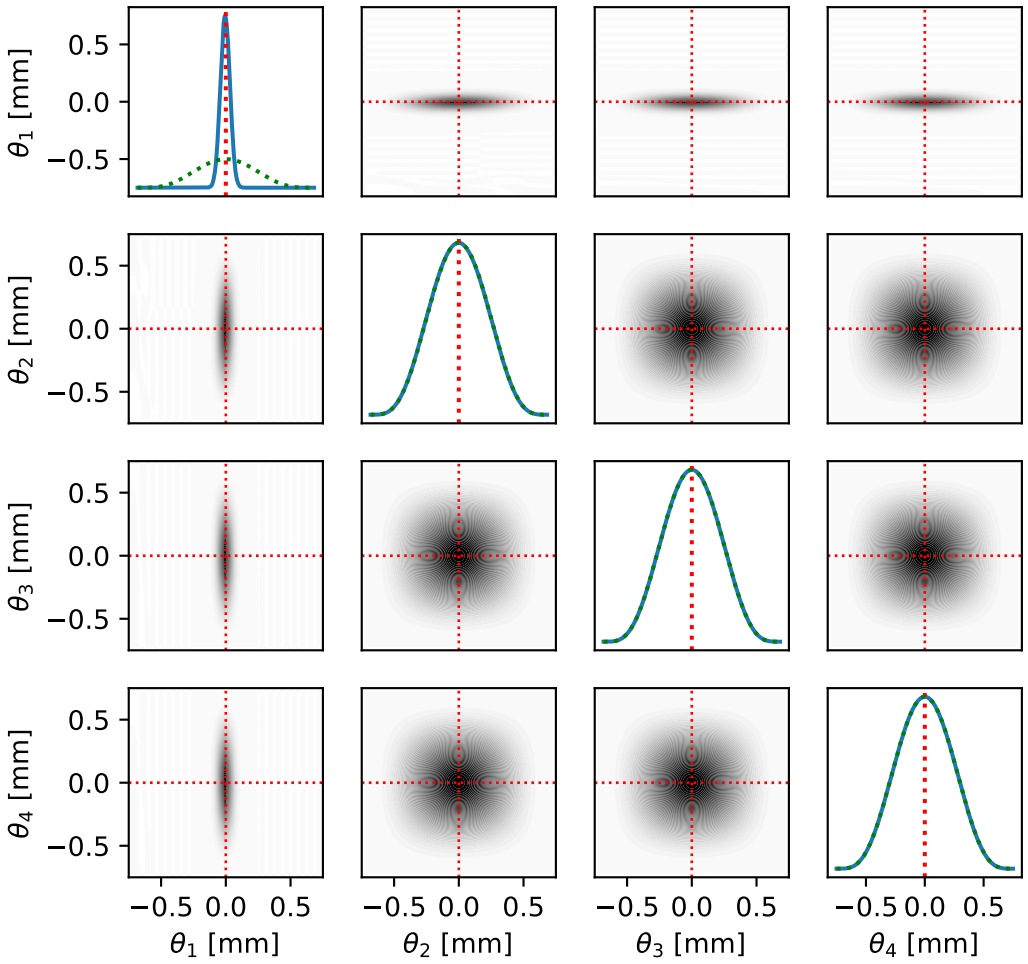
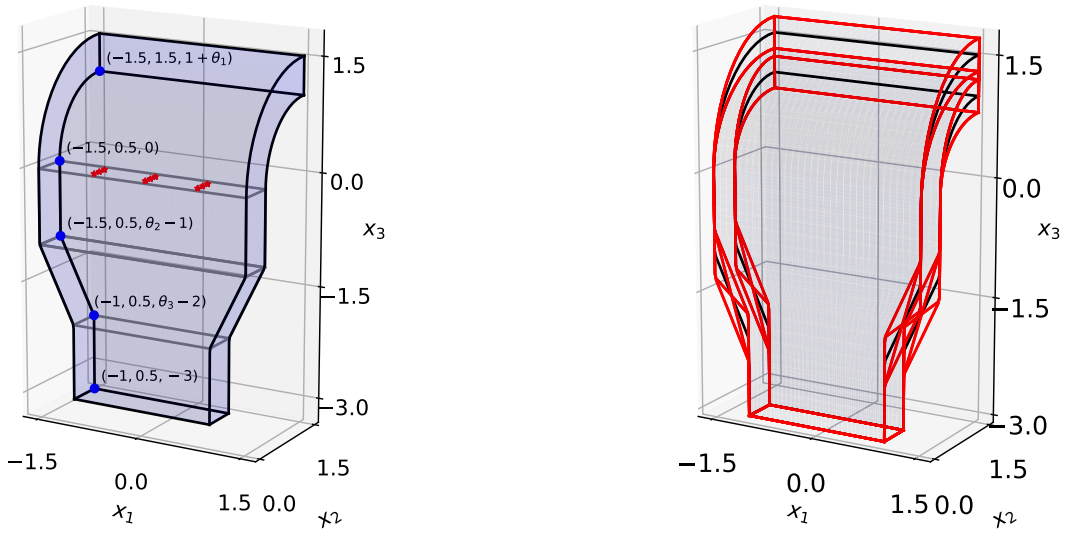


Figure 6.12.: Quadrupole section: posterior PDF for the inverse problem. Marginalization is performed to obtain the graphical representation. On the main diagonal, the prior (green dashed line) is plotted together with the posterior (blue solid line). The red vertical lines represent the ground truth $\hat{\boldsymbol{\theta}} = \mathbf{0}$.

where the boundary condition $g(x_1, x_2) = \cos(\pi x_1)\sin(\pi x_2)$ is imposed at the plane $x_3 = -3$, the zero Dirichlet boundary condition is imposed on the remaining part of the boundary $\partial D(\boldsymbol{\theta}) \cap \{(x_1, x_2, x_3) : x_3 = -3\}$ and $\rho = 49$. The solution is plotted in Figure 6.14 for $\theta_1 = \theta_2 = \theta_3 = 0$.

The BVP is discretized using tensor product of uniform quadratic B-splines of dimension $\boldsymbol{n} = (64, 64, 128)$ for the spatial dimensions and a tensor product of Chebyshev nodes of size $\boldsymbol{\ell} = (8, 8, 8)$ for the parameter space. Due to the low rank structure of the parameter dependent geometry, the system operators (stiffness and mass) are constructed in ≈ 2 s for a relative accuracy of $\epsilon = 10^{-12}$ and $\epsilon = 10^{-10}$ for the mass and stiffness, respectively. The TT rank of the stiffness tensor operator is $\boldsymbol{r}^{\mathsf{S}} = (1, 5, 21, 12, 6, 8, 1)$, which corresponds to only approximately 35 MB of storage space. The solution is obtained using the AMEn solver



(a) The geometry of the waveguide. The cross-section is rectangular with height $h = 0.5$ and width ranging from 2 to 3. The positions of the blue points is also displayed as well as their parameter dependence. Red markers represent the position of the observations for the inverse problem.

(b) Waveguide geometry for different parameter combinations $\boldsymbol{\theta} \in \{-0.2, 0.2\} \times \{-0.3, 0.3\} \times \{-0.3, 0.3\}$.

Figure 6.13.: Geometry of the waveguide structure with rectangular section.

for a relative accuracy of the residual of $\epsilon = 10^{-7}$. To verify the correctness of the TT-IGA solver, the solution is compared against a FEM solver for some parameter realizations. The GMRES solver used by the FEM solver suffers from the lack of preconditioning. The AMEn solver on the other hand benefits from the preconditioning of the local subsystems. The TT-IGA solver is again faster than the conventional FEM solver for a similar discretization level. Regarding the low rank approximation of the parameter dependent solution, its TT rank is $\boldsymbol{r}^{\mathbf{u}} = (1, 64, 81, 81, 66, 10, 1)$, corresponding to around 9 MB. As a comparison, storing the full tensor of shape $(64, 64, 128)$ corresponding to a fixed parameter requires ≈ 4 MB.

Inverse problem

Similar to the previous test case, an inverse problem is solved using the surrogate model of the solution of (6.77). The observed quantity is the solution u at a discrete number of points. The observations are generated using a FEM simulation and correspond to the parameter $\hat{\boldsymbol{\theta}} = \mathbf{0}$. The $z_k = u(\boldsymbol{x}^{(k)}, \hat{\boldsymbol{\theta}}) + \varepsilon_k$ where ε_k is a sequence of i.i.d. Gaussian RVs with mean 0 and variance σ_{n}^2. The likelihood in this case is

$$p_{\boldsymbol{Z}|\boldsymbol{\vartheta}=\boldsymbol{\theta}}(\boldsymbol{z}) \propto \exp\left(-\frac{1}{2\sigma_{\mathrm{n}}^2}\sum_{k=1}^{N_{\mathrm{o}}}(z_k - u(\boldsymbol{x}^{(k)}, \boldsymbol{\theta}))^2\right), \tag{6.78}$$

where $\boldsymbol{x}^{(k)}$ are the positions in the physical domain where the field is observed (represented with the red markers in Figure 6.13a). Compared to the previous case, the points $\boldsymbol{x}^{(k)}$ correspond to the same points in the reference domain for every parameter combination.

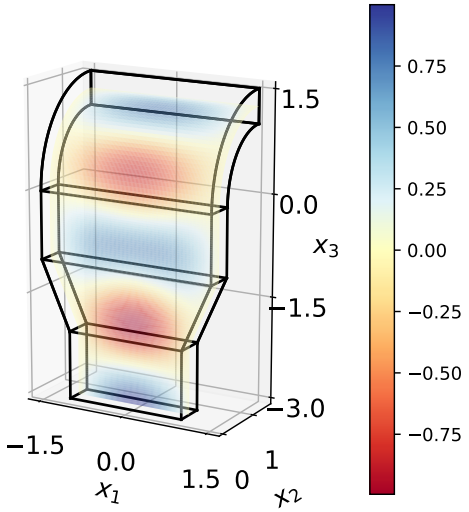


Figure 6.14.: Waveguide problem: the solution for $\theta_1 = \theta_2 = \theta_3 = 0$ and $\boldsymbol{y} \in [0,1] \times \{0.5\} \times [0,1]$.

Using the surrogate model, the posterior

$$p_{\boldsymbol{\vartheta}|\boldsymbol{Z}=\boldsymbol{z}}(\boldsymbol{\theta}) \propto p_{\boldsymbol{Z}|\boldsymbol{\vartheta}=\boldsymbol{\theta}}(\boldsymbol{z}) p_{\boldsymbol{\vartheta}}(\boldsymbol{\theta}) \tag{6.79}$$

is interpolated using a tensor product quadratic B-spline basis of size $\boldsymbol{n} = (64, 64, 64)$. The prior of each parameter is a Gamma distributed with $\alpha = \beta = 5$ (scaled to the parameter space Ξ). The results for $\sigma_{\mathrm{n}} = 0.02$ are showed in Figure 6.15, where the individual parameters are marginalized to obtain a graphical representation of the posterior. Furthermore, the first and second order moments of the posterior are computed using numerical integration:

$$\mathbb{E}(\boldsymbol{\vartheta}) = (0.0067, -0.0337, -0.0031), \tag{6.80}$$

$$\mathrm{Cov}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}) = \begin{pmatrix} 4.4079 \cdot 10^{-5} & 7.6069 \cdot 10^{-5} & -4.8618 \cdot 10^{-5} \\ 7.6069 \cdot 10^{-5} & 5.5687 \cdot 10^{-3} & 5.8178 \cdot 10^{-4} \\ -4.8618 \cdot 10^{-5} & 5.8178 \cdot 10^{-4} & 9.6319 \cdot 10^{-3} \end{pmatrix}. \tag{6.81}$$

The parameter that is identified with the highest confidence is θ_1, since the variance of its posterior is orders of magnitude lower compared to the variance of the prior. The reason for this is that the first parameter has the largest impact on the length of the waveguide structure and causes the highest variation of the field at the given observation points. Between the second and third parameter, a correlation is observed. This is motivated by the fact that the height of the truncated pyramid part of the waveguide plays a more significant role than its position.

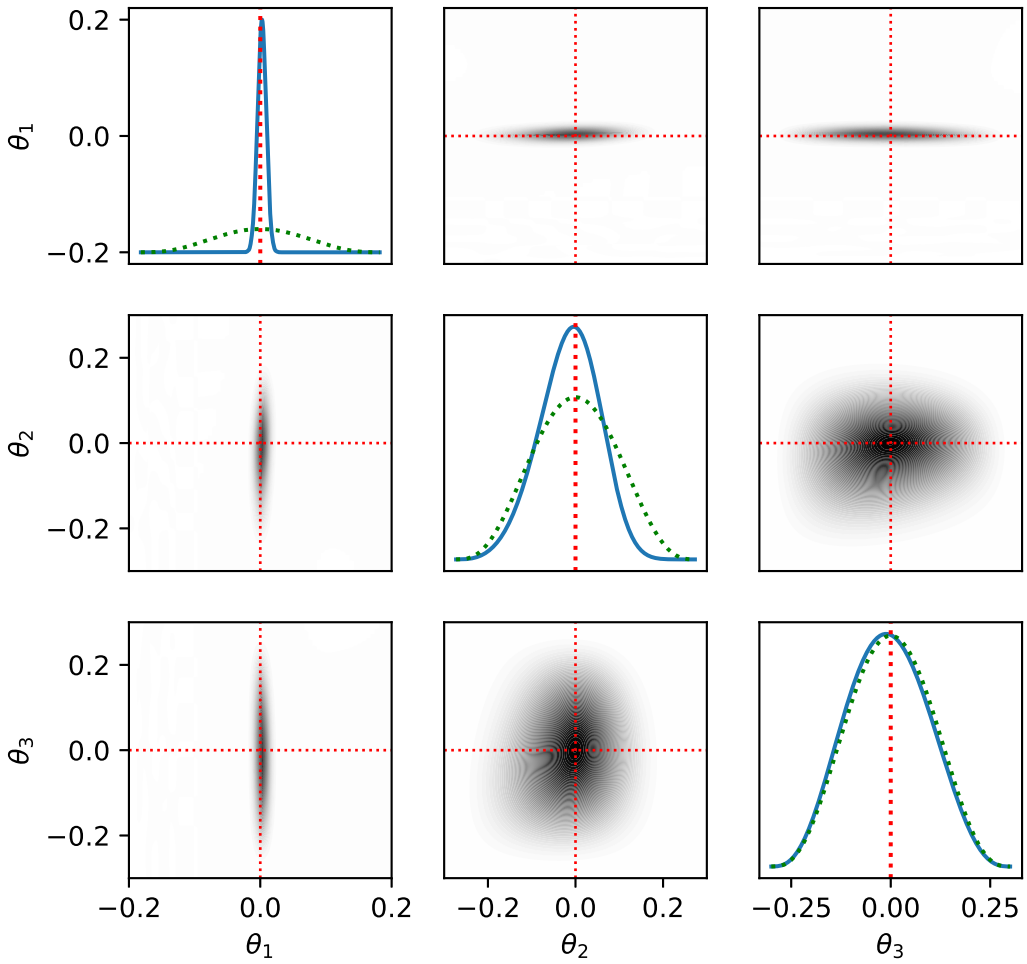


Figure 6.15.: Waveguide structure with rectangular section: posterior PDF for the inverse problem. Marginalization is performed to obtain the graphical representation. On the main diagonal, the prior (green dashed line) is plotted together with the posterior (blue solid line). The red vertical lines represent the ground truth $\hat{\theta} = \mathbf{0}$.

7. Conclusion and Outlook

The final chapter of this thesis consists of an overview and some concluding remarks. Furthermore, several possible continuations of the work are suggested.

7.1. Conclusion

In this work, we addressed the construction of surrogate models using the TT format. Two white-box surrogate modeling methods were presented: TT-CME and TT-IGA. Despite the applications coming from different fields, the idea remains the same: add additional dimensions to the DoF tensor to accommodate the parameter dependence. An extended system is then built for recovering the DoF tensor. In both cases, all relevant tensor operators and the right-hand side are directly computed in the TT format using TT cores manipulation and multilinear algebra operations. The AMEn solver is then used to directly obtain the solution DoFs in the TT format. Despite being prohibitive when dealing with full tensors, the idea of increasing the dimensionality can be beneficial in the TT format since the dependence between the parameters can be captured by interactions between the cores. This effect is the so-called “blessing of dimensionality” [Kho12].

The TT-CME has been proven to be a viable alternative to the conventional methods. A general method for constructing the CME generator directly in the TT format was presented. The time dependency is resolved by basis expansion over the time interval. Thus, an additional dimension is added to the involved tensors. When adding parameters, the dimensionality is increased even further in order to obtain an approximation over the joint state-parameter-time domain. The extended system is then obtained using Galerkin projection. The accuracy and efficiency of the solver is then assessed in the numerical results section. Bayesian inference tasks such as filtering, smoothing and parameter identification are conducted using the presented framework, thanks to the fast and efficient linear algebra operations in TT. By using the TT-CME solver, a drastic improvement in the storage requirements as well as the computational time is obtained. In some cases, storing the full solution would exceed the memory available on conventional workstations, while the TT solver is able to run on a normal laptop computer.

For the TT-IGA solver, the main difficulty consists in constructing the TT representation of the discretized weak formulation. The proposed construction method has a lower computational complexity compared to conventional assemblers. As shown in the numerical findings, the standard IGA assembly techniques are slower than the proposed TT based method for the combined physical and parameter space. The AMEn solver is also faster and more memory efficient than the conventional GMRES solver. The first two numerical experiments validate the solver and investigate its behavior for an increasing number of parameters, while the remaining two examples address the usage of surrogates for solving synthetic inverse problems. Due to the fast evaluations of the surrogate model, the TT cross approximation method can be used to interpolate the posterior PDF.

Compared to black-box surrogate modeling techniques, the specialized TT based solvers offer a significant improvement in the computational time. It is no longer bounded by the number of model calls needed for constructing the training set. In some cases, building a TT surrogate is faster than using conventional solvers for a single parameter realization. Moreover, the low-rank representations of the parameter dependent

solution are also efficient during the post-processing step. Numerical integration, pointwise evaluation, differentiation can be easily done on the surrogate model.

7.2. Outlook

The following extensions and improvements, which have not been addressed in this work, are considered to be important for future research works:

- The TT-CME solver presented in Chapter 5 assumes a fixed basis to represent the parameter dependency for all filtering steps. As already stated in the results section, the posterior PDF over the parameter space shrinks and the relevant part becomes concentrated in a small region of the initial truncated parameter space. As a result, a uniform distribution of the knots of the B-splines does not offer a high enough resolution. Further research should be conducted for performing adaptive basis refinement or to adaptively adjust the knots of the B-splines depending on where the information is concentrated.

- In Chapter 6, the geometry was defined by either interpolating a given parametrization or by providing the control points of the NURBS representation. Since the choice of the geometry parametrization affects the TT rank of the control points and therefore the rank of the discrete tensor operators, we are interested in finding the best parametrization in order to minimize the computational burden of the TT solver. The image of the improved parametrization must still remain the same physical domain. Several metrics have been proposed to quantify the goodness of a parametrization [PCT20]. The method introduced in [JM17a] should be extended to constructing efficient TT-representations of domains starting from the (parameter dependent) boundary patches [PCT20]. This has been addressed in [PC19] for fixed geometries and using the CPD format.

- One possible extension of the TT-IGA solver is to address nonlinear problems. This, for example, is of great interest for the quadrupole magnet analyzed in Section 6.5.3. In classical FEM, the most common approach when dealing with nonlinear PDEs is to apply the Newton solver on the nonlinear system derived from the weak formulation [Wri08]. This requires a reconstruction of the discrete operators at every step. This procedure can be also extended to the IGA [CHB09, DZW+20, WWS13]. In order to integrate this in the TT based solver, the construction of the discrete operators, especially the stiffness operator, has to be adapted to account for nonlinearities.

- The emerging field of quantum computing has gained a lot of attention over the past few years. Development of linear solvers for quantum computers [PPM+22] motivate the idea of developing quantum algorithms for solving PDEs. A further continuation of the work is to investigate the possibility of implementing the presented TT based solvers on quantum computing systems.

- The main application of the proposed surrogate modeling techniques is solving inverse problems. However, the white-box solvers are not limited to this application. A further continuation is to diversify the use cases to other fields such as forward UQ, design optimization, or control problems.

A. B-spline bases

Of relevance for this work, especially for the IGA solver but also for representing the parameter dependence of the CME solution are the B-spline bases. In this section, we define the B-spline basis, and we will give some of their important properties relevant to this thesis.

Definition A.0.1 (B-spline shape functions)**.** Let $\boldsymbol{\zeta} = [\zeta_1, \dots, \zeta_{n+p+1}]$ with $\zeta_k \leqslant \zeta_{k+1}$, $\zeta_1 = \cdots = \zeta_{p+1} = 0$, and $\zeta_{n+1} = \cdots = \zeta_{n+p+1} = 1$, be a knot vector, where n denotes the space dimension and p the polynomial degree. Then, the B-spline functions $\{b_{k,p}\}_{k=1}^{n}$ of degree p are defined via the Cox–de Boor recursion formula [PT96]

$$b_{k,p}(x) = \frac{x - \zeta_k}{\zeta_{k+1} - \zeta_k} b_{k,p-1}(x) + \frac{\zeta_{k+p+1} - x}{\zeta_{k+p+1} - \zeta_{k+1}} b_{k+1,p-1}(x), \quad p > 0, \tag{A.1}$$

$$b_{k,0}(x) = \begin{cases} 1, & x \in [\zeta_k, \zeta_{k+1}), \\ 0, & \text{otherwise.} \end{cases} \tag{A.2}$$

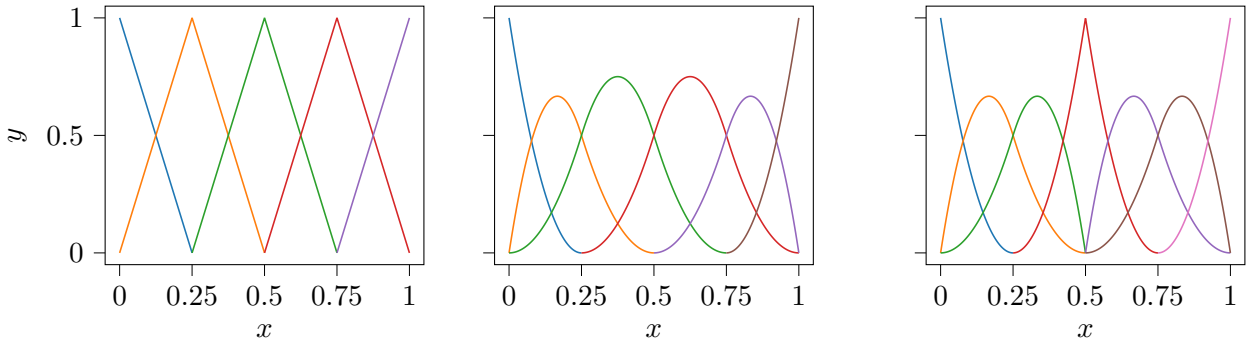
where the convention $0/0 = 0$ is used. Furthermore, we introduce the space $\mathcal{S}(\boldsymbol{\zeta}, p) = \text{span}\{b_{k,p} \ : \ k = 1, ..., n\}$.

The main properties relevant in this work are [PT96] :

1. The B-spline functions are piecewise polynomials of degree p between the knots.
2. The space $\mathcal{S}(\boldsymbol{\zeta}, p)$ has the dimension n.
3. If a knot $\zeta_k \in (0, 1)$ has the multiplicity m_k (its value appears m_k times), then the B-splines are at most $p - m_k$ continuously differentiable at the point ζ_k ($p - m_k = 0$ just continuity holds at ζ_k). See Figure A.1c for one example.
4. The compact support of the k-th basis is (ζ_k, ζ_{k+p+1}).
5. The derivatives of the B-splines are again piecewise polynomials and can be expressed as

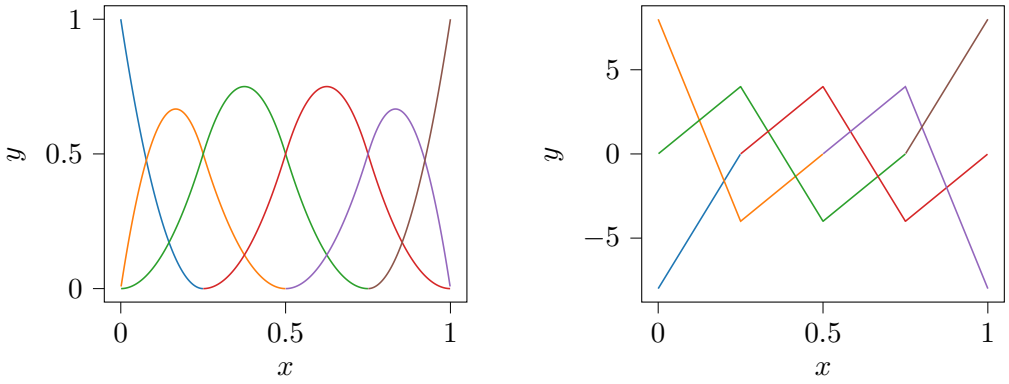
$$\frac{\mathrm{d}b_{k,p}(x)}{\mathrm{d}x} = \frac{p}{\zeta_{k+1} - \zeta_k} b_{k,p-1}(x) - \frac{p}{\zeta_{k+p+1} - \zeta_{k+1}} b_{k+1,p-1}(x), \quad p > 0. \tag{A.3}$$

   with the same convention for the division by 0. In Figure A.2b, the B-spline basis functions for $p = 2$ and $\boldsymbol{\zeta} = (0, 0, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 1, 1)$ is plotted along with their derivatives.



(a) $\zeta = (0, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 1)$ (b) $\zeta = (0, 0, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 1, 1)$ (c) $\zeta = (0, 0, 0, \frac{1}{4}, \frac{1}{2}, \frac{1}{2}, \frac{3}{4}, 1, 1, 1)$

Figure A.1.: Different B-spline bases for $p = 1$, $p = 2$, and $p = 2$ with increased multiplicity for the knot $1/2$ (only the compact support is represented).



(a) B-spline basis (b) The derivative of the B-splines

Figure A.2.: The quadratic B-spline basis for the knots $\zeta = (0, 0, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 1, 1)$ and the first order derivatives (only the compact support is represented).

B. Metric coefficients

In Section 6.4.3, the tensors $\mathbf{h}^{(\alpha,\beta)}$ were used compute the metric coefficients for the stiffness integral. By defining the tensors

$$\mathrm{g}_{\boldsymbol{ij}}^{(\alpha,\beta)} = \frac{\partial G_\alpha(\boldsymbol{y}_{\boldsymbol{j}}^{\mathrm{Q}}, \boldsymbol{\theta_i})}{\partial y_\beta}, \quad \alpha,\beta \in \{1,\dots,d\}, \tag{B.1}$$

the tensors $\mathbf{h}^{(\alpha,\beta)}$ are calculated as it follows for the case $d = 3$:

$$\begin{pmatrix} \mathbf{h}^{(1,1)} & \mathbf{h}^{(1,2)} & \mathbf{h}^{(1,3)} \\ \mathbf{h}^{(2,1)} & \mathbf{h}^{(2,2)} & \mathbf{h}^{(2,3)} \\ \mathbf{h}^{(3,1)} & \mathbf{h}^{(3,2)} & \mathbf{h}^{(3,3)} \end{pmatrix} =$$

$$\begin{pmatrix} \mathbf{g}^{(2,2)} \odot \mathbf{g}^{(3,3)} - \mathbf{g}^{(2,3)} \odot \mathbf{g}^{(3,2)} & \mathbf{g}^{(1,3)} \odot \mathbf{g}^{(3,2)} - \mathbf{g}^{(1,2)} \odot \mathbf{g}^{(3,3)} & \mathbf{g}^{(1,2)} \odot \mathbf{g}^{(2,3)} - \mathbf{g}^{(1,3)} \odot \mathbf{g}^{(2,2)} \\ \mathbf{g}^{(2,3)} \odot \mathbf{g}^{(3,1)} - \mathbf{g}^{(2,1)} \odot \mathbf{g}^{(3,3)} & \mathbf{g}^{(1,1)} \odot \mathbf{g}^{(3,3)} - \mathbf{g}^{(1,3)} \odot \mathbf{g}^{(3,1)} & \mathbf{g}^{(1,3)} \odot \mathbf{g}^{(2,1)} - \mathbf{g}^{(1,1)} \odot \mathbf{g}^{(2,3)} \\ \mathbf{g}^{(2,1)} \odot \mathbf{g}^{(3,2)} - \mathbf{g}^{(2,2)} \odot \mathbf{g}^{(3,1)} & \mathbf{g}^{(1,2)} \odot \mathbf{g}^{(3,1)} - \mathbf{g}^{(1,1)} \odot \mathbf{g}^{(3,2)} & \mathbf{g}^{(1,1)} \odot \mathbf{g}^{(2,2)} - \mathbf{g}^{(1,2)} \odot \mathbf{g}^{(2,1)} \end{pmatrix}. \tag{B.2}$$

If $d = 2$, the tensors are given as

$$\begin{pmatrix} \mathbf{h}^{(1,1)} & \mathbf{h}^{(1,2)} \\ \mathbf{h}^{(2,1)} & \mathbf{h}^{(2,2)} \end{pmatrix} = \begin{pmatrix} \mathbf{g}^{(2,2)} & -\mathbf{g}^{(1,2)} \\ -\mathbf{g}^{(2,1)} & \mathbf{g}^{(1,1)} \end{pmatrix}. \tag{B.3}$$

C. Quadrupole magnet model

In this part, the geometry of the quadrupole magnet is described in detail. First, the following geometry constants are defined:

- $D_\mathrm{o} = 72$ mm,
- $D_\mathrm{i} = 51$ mm,
- $h_\mathrm{i} = 12$ mm,
- $b_\mathrm{i} = 3$ mm,
- $D_\mathrm{c} = 35$ mm,
- $h_\mathrm{c} = 6$ mm,
- $r_\mathrm{i} = 20$ mm,
- $r_\mathrm{a} = 18$ mm,
- $b_\mathrm{c} = h_\mathrm{i} - h_\mathrm{c}$.

In addition to the defined constants, the following functions are introduced:

- $r_\mathrm{O}(D_\mathrm{c}, h_\mathrm{c}, r_\mathrm{i}) = \frac{D_\mathrm{c}D_\mathrm{c}+h_\mathrm{c}h_\mathrm{c}-r_\mathrm{i}r_\mathrm{i}}{D_\mathrm{c}\sqrt{(2)}+h_\mathrm{c}\sqrt{(2)}-2r_\mathrm{i}}$ represents the radius of the circular section.
- The control point controlling the circle arch has the following coordinates C_x and C_y given explicitly as

$$\begin{pmatrix} \frac{r_\mathrm{O}(D_\mathrm{c},h_\mathrm{c},r_\mathrm{i})-r_\mathrm{i}}{\sqrt{(2)}} & \frac{r_\mathrm{O}(D_\mathrm{c},h_\mathrm{c},r_\mathrm{i})-r_\mathrm{i}}{\sqrt{(2)}} \\ \frac{r_\mathrm{O}(D_\mathrm{c},h_\mathrm{c},r_\mathrm{i})}{\sqrt{2}} - D_\mathrm{c} & \frac{r_\mathrm{O}(D_\mathrm{c},h_\mathrm{c},r_\mathrm{i})}{\sqrt{2}} - h_\mathrm{c} \end{pmatrix} \begin{pmatrix} C_x(D_\mathrm{c}, h_\mathrm{c}, r_\mathrm{i}) \\ C_y(D_\mathrm{c}, h_\mathrm{c}, r_\mathrm{i}) \end{pmatrix} = \begin{pmatrix} \frac{r_\mathrm{O}(D_\mathrm{c},h_\mathrm{c},r_\mathrm{i})-r_\mathrm{i}}{\sqrt{(2)}} \frac{2r_\mathrm{i}}{\sqrt{2}} \\ \frac{r_\mathrm{O}(D_\mathrm{c},h_\mathrm{c},r_\mathrm{i})-r_\mathrm{i}}{\sqrt{(2)}}(D_\mathrm{c} - h_\mathrm{c}) \end{pmatrix} \tag{C.1}$$

The parameter dependent weights are:

$$\mathsf{w}_{i_1 i_2} = \begin{cases} \sin\left(\frac{\pi - \arcsin\left(\frac{h_\mathrm{c} - r_\mathrm{O}(D_\mathrm{c}+\theta_2, h_\mathrm{c}, r_\mathrm{i}+\theta_1)/\sqrt{(2)}}{r_\mathrm{O}(D_\mathrm{c}+\theta_2, h_\mathrm{c}, r_\mathrm{i}+\theta_1) - r_\mathrm{i} - \theta_1}\right) + \frac{3\pi}{4}}{2}\right) & , i_1 = 2, i_2 = 3 \\ 1 & , \text{ otherwise.} \end{cases}$$

Finally, the following B-spline space is used:

$$\mathcal{S}((0,0,0,0.4,0.4,0.6,0.6,1,1,1)^\top, 2) \otimes \mathcal{S}((0,0,0.15,0.3,0.5,1,1)^\top, 1).$$

The parameter dependent control points (also illustrated in Figure C.1 for $\boldsymbol{\theta} = (0,0,0,0)$) are given by:

$$\mathbf{p}_{:11}(\boldsymbol{\theta}) = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \mathbf{p}_{:12}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{r_\mathrm{i}+\theta_1}{2\sqrt{2}} \\ \frac{r_\mathrm{i}+\theta_1}{2\sqrt{2}} \end{pmatrix} \qquad \mathbf{p}_{:13}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{r_\mathrm{i}+\theta_1}{\sqrt{2}} \\ \frac{r_\mathrm{i}+\theta_1}{\sqrt{2}} \end{pmatrix}$$

$$\mathbf{p}_{:21}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{D_\mathrm{c}+\theta_2}{2} \\ 0 \end{pmatrix} \qquad \mathbf{p}_{:22}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{C_x(r_\mathrm{i},D_\mathrm{c},h_\mathrm{c})+D_\mathrm{c}}{2} \\ \frac{C_y(r_\mathrm{i},D_\mathrm{c},h_\mathrm{c})}{2} \end{pmatrix} \qquad \mathbf{p}_{:23}(\boldsymbol{\theta}) = \begin{pmatrix} C_x(r_\mathrm{i}+\theta_1, D_\mathrm{c}+\theta_2, h_\mathrm{c}) \\ C_x(r_\mathrm{i}+\theta_1, D_\mathrm{c}+\theta_2, h_\mathrm{c}) \end{pmatrix}$$

$$\mathbf{p}_{:31}(\boldsymbol{\theta}) = \begin{pmatrix} D_\mathrm{c}+\theta_2 \\ 0 \end{pmatrix} \qquad \mathbf{p}_{:32}(\boldsymbol{\theta}) = \begin{pmatrix} D_\mathrm{c}+\theta_2 \\ \frac{h_\mathrm{c}}{2} \end{pmatrix} \qquad \mathbf{p}_{:33}(\boldsymbol{\theta}) = \begin{pmatrix} D_\mathrm{c}+\theta_2 \\ h_\mathrm{c} \end{pmatrix}$$

$$\mathbf{p}_{:41}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{D_\mathrm{c}+D_\mathrm{i}}{2} \\ 0 \end{pmatrix} \qquad \mathbf{p}_{:42}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{D_\mathrm{c}+D_\mathrm{i}}{2} \\ \frac{h_\mathrm{i}-b_\mathrm{i}+h_\mathrm{c}}{4} \end{pmatrix} \qquad \mathbf{p}_{:43}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{D_\mathrm{c}+D_\mathrm{i}}{2} \\ \frac{h_\mathrm{i}-b_\mathrm{i}+h_\mathrm{c}}{2} \end{pmatrix}$$

$$\mathbf{p}_{:51}(\boldsymbol{\theta}) = \begin{pmatrix} D_\mathrm{i}+\theta_3 \\ 0 \end{pmatrix} \qquad \mathbf{p}_{:52}(\boldsymbol{\theta}) = \begin{pmatrix} D_\mathrm{i}+\theta_3 \\ \frac{h_\mathrm{i}-b_\mathrm{i}}{2} \end{pmatrix} \qquad \mathbf{p}_{:53}(\boldsymbol{\theta}) = \begin{pmatrix} D_\mathrm{i}+\theta_3 \\ h_\mathrm{i}-b_\mathrm{i} \end{pmatrix}$$

$$\mathbf{p}_{:61}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{D_\mathrm{o}+D_\mathrm{i}}{2} \\ 0 \end{pmatrix} \qquad \mathbf{p}_{:62}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{D_\mathrm{o}+D_\mathrm{i}}{2} \\ \frac{h_\mathrm{i}-b_\mathrm{i}}{2} \end{pmatrix} \qquad \mathbf{p}_{:63}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{D_\mathrm{o}+D_\mathrm{i}}{2} \\ h_\mathrm{i}-b_\mathrm{i} \end{pmatrix}$$

$$\mathbf{p}_{:71}(\boldsymbol{\theta}) = \begin{pmatrix} D_\mathrm{o} \\ 0 \end{pmatrix} \qquad \mathbf{p}_{:72}(\boldsymbol{\theta}) = \begin{pmatrix} D_\mathrm{o} \\ \frac{h_\mathrm{i}-b_\mathrm{i}}{2} \end{pmatrix} \qquad \mathbf{p}_{:73}(\boldsymbol{\theta}) = \begin{pmatrix} D_\mathrm{o} \\ h_\mathrm{i}-b_\mathrm{i} \end{pmatrix}$$

$$\mathbf{p}_{:14}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{0.75r_\mathrm{i}+0.25D_\mathrm{o}}{\sqrt{2}} \\ \frac{0.75r_\mathrm{i}+0.25D_\mathrm{o}}{\sqrt{2}} \end{pmatrix} \qquad \mathbf{p}_{:15}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{D_\mathrm{o}}{\sqrt{(2)}} \\ \frac{D_\mathrm{o}}{\sqrt{(2)}} \end{pmatrix}$$

$$\mathbf{p}_{:24}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{1}{2}\left(\frac{0.75r_\mathrm{i}+0.25D_\mathrm{o}}{\sqrt{2}} + D_\mathrm{c} + b_\mathrm{c}\right) \\ \frac{1}{2}\left(\frac{0.75r_\mathrm{i}+0.25D_\mathrm{o}}{\sqrt{2}} + h_\mathrm{i}\right) \end{pmatrix} \qquad \mathbf{p}_{:25}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{D_\mathrm{o}}{\sqrt{(2)}}\left(1-\frac{D_\mathrm{c}}{2D_\mathrm{o}}\right) + D_\mathrm{o}\frac{D_\mathrm{c}}{2D_\mathrm{o}} \\ \frac{D_\mathrm{o}}{\sqrt{(2)}}\left(1-\frac{D_\mathrm{c}}{2D_\mathrm{o}}\right) + D_\mathrm{o}\tan(\pi/8)\frac{D_\mathrm{c}}{2D_\mathrm{o}} \end{pmatrix}$$

$$\mathbf{p}_{:34}(\boldsymbol{\theta}) = \begin{pmatrix} D_\mathrm{c}+b_\mathrm{c} \\ h_\mathrm{i}+\theta_4 \end{pmatrix} \qquad \mathbf{p}_{:35}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{D_\mathrm{o}}{\sqrt{(2)}}\left(1-\frac{D_\mathrm{c}}{D_\mathrm{o}}\right) + D_\mathrm{o}\frac{D_\mathrm{c}}{D_\mathrm{o}} \\ \frac{D_\mathrm{o}}{\sqrt{(2)}}\left(1-\frac{D_\mathrm{c}}{D_\mathrm{o}}\right) + D_\mathrm{o}\tan(\pi/8)\frac{D_\mathrm{c}}{D_\mathrm{o}} \end{pmatrix}$$

$$\mathbf{p}_{:44}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{D_\mathrm{c}+b_\mathrm{c}+D_\mathrm{i}-b_\mathrm{i}}{2} \\ h_\mathrm{i}+\theta_4 \end{pmatrix} \qquad \mathbf{p}_{:45}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{D_\mathrm{o}}{\sqrt{(2)}}\left(1-\frac{D_\mathrm{i}+D_\mathrm{o}}{2D_\mathrm{o}}\right) + D_\mathrm{o}\frac{D_\mathrm{i}+D_\mathrm{c}}{2D_\mathrm{o}} \\ \frac{D_\mathrm{o}}{\sqrt{(2)}}\left(1-\frac{D_\mathrm{i}+D_\mathrm{o}}{2D_\mathrm{o}}\right) + D_\mathrm{o}\tan\pi/8\frac{D_\mathrm{i}+D_\mathrm{o}}{2D_\mathrm{c}} \end{pmatrix}$$

$$\mathbf{p}_{:54}(\boldsymbol{\theta}) = \begin{pmatrix} D_\mathrm{i}-b_\mathrm{i} \\ h_\mathrm{i}+\theta_4 \end{pmatrix} \qquad \mathbf{p}_{:55}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{D_\mathrm{o}}{\sqrt{(2)}}\left(1-\frac{D_\mathrm{i}}{D_\mathrm{o}}\right) + D_\mathrm{o}\frac{D_\mathrm{i}}{D_\mathrm{o}} \\ \frac{D_\mathrm{o}}{\sqrt{(2)}}\left(1-\frac{D_\mathrm{i}}{D_\mathrm{o}}\right) + D_\mathrm{o}\tan(\pi/8)\frac{D_\mathrm{i}}{D_\mathrm{o}} \end{pmatrix}$$

$$\mathbf{p}_{:64}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{D_\mathrm{i}+D_\mathrm{o}}{2} \\ h_\mathrm{i} \end{pmatrix} \qquad \mathbf{p}_{:65}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{D_\mathrm{o}}{\sqrt{2}}\left(1-\frac{D_\mathrm{i}+D_\mathrm{o}}{2D_\mathrm{o}}\right) + D_\mathrm{o}\frac{D_\mathrm{i}+D_\mathrm{o}}{2D_\mathrm{o}} \\ \frac{D_\mathrm{o}}{\sqrt{(2)}}\left(1-\frac{D_\mathrm{i}+D_\mathrm{o}}{2D_\mathrm{o}}\right) + D_\mathrm{o}\tan(\pi/8)\frac{(D_\mathrm{i}+D_\mathrm{o})}{2D_\mathrm{o}} \end{pmatrix}$$

$$\mathbf{p}_{:74}(\boldsymbol{\theta}) = \begin{pmatrix} D_\mathrm{o} \\ h_\mathrm{i} \end{pmatrix} \qquad \mathbf{p}_{:75}(\boldsymbol{\theta}) = \begin{pmatrix} D_\mathrm{o} \\ D_\mathrm{o}\tan(\pi/8) \end{pmatrix}$$

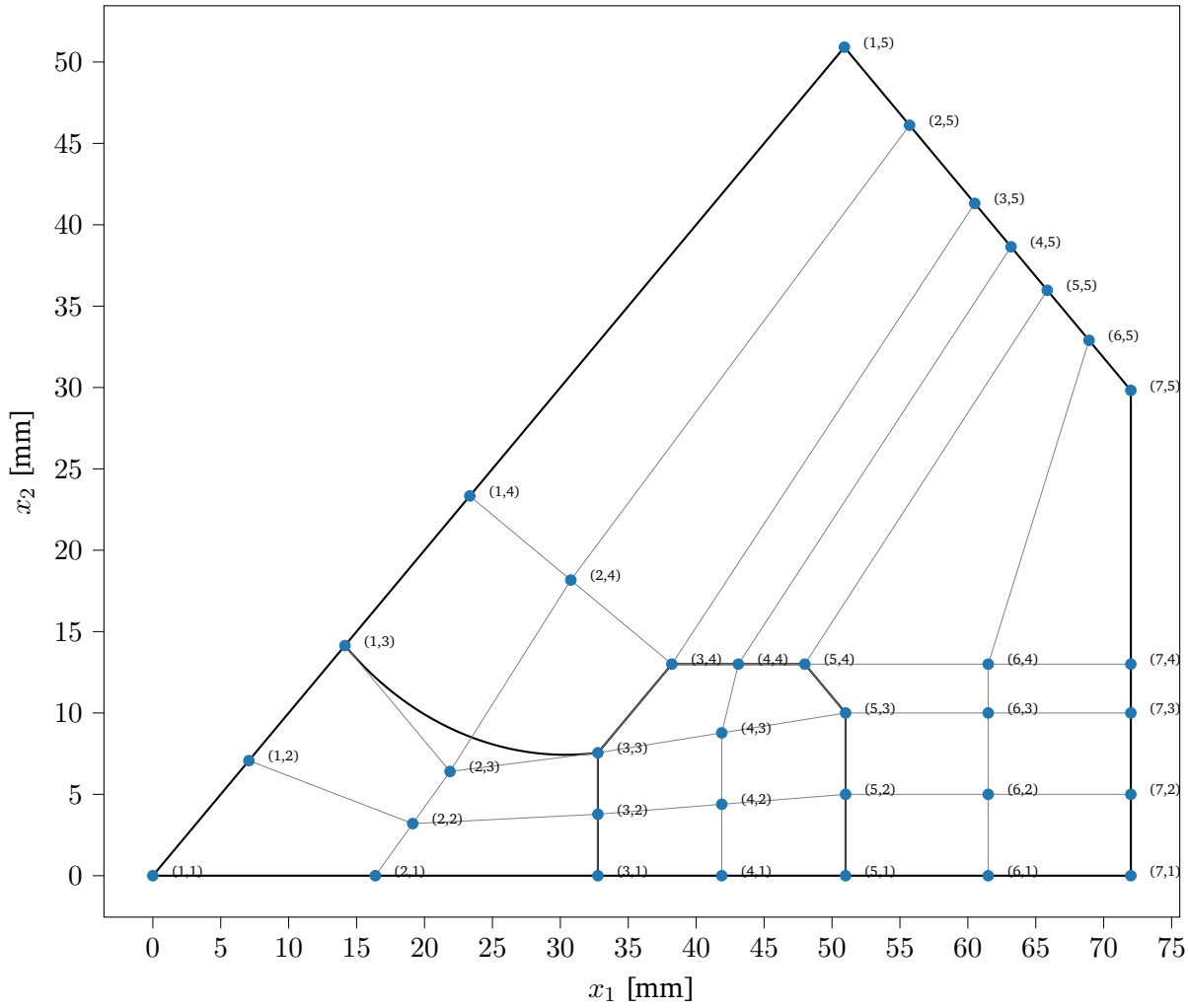


Figure C.1.: The control points (blue dots) of the quadrupole magnet model for $\boldsymbol{\theta} = (0, 0, 0, 0)$. The control point mesh is represented with gray lines and the indices of the control points $\mathbf{p}_{:i_1 i_2}$ are marked as (i_1, i_2).

Glossary

ALS alternating least squares. 27–30, 32, 113

AMEn alternating minimal energy. 3, 26, 30–32, 41, 46, 53, 57, 61, 63, 79, 81–83, 85, 86, 88, 91, 92, 95

BVP boundary value problem. 2, 67, 68, 70, 71, 75, 80–82, 85, 86, 88, 90, 91

CAD computer-aided design. 67, 68, 76

CME chemical master equation. vii, 2, 3, 26, 41, 43, 44, 46–57, 59, 61, 63, 64, 95–97, 107, 111, 113, 129

CPD canonical polyadic decomposition. 15, 16, 18, 27, 96

CPU central processing unit. 85

DMRG density matrix renormalization group. 23, 29–32, 107

DoF degree of freedom. 1, 2, 39, 41, 44, 47, 52, 67, 71, 74, 75, 80, 81, 84, 95

FEM finite element method. 2, 67, 71, 75, 81, 84, 89, 92, 96

GMRES generalized minimal residual method. 26, 82, 83, 92, 95

GPU graphics processing unit. 85

HMM hidden Markov model. ix, 38, 41, 49–51, 107

HOOI higher-order orthogonal iteration. 16

HOSVD higher-order SVD. 16

IC initial condition. 3, 50–54

IGA isogeometric analysis. vii, x, 2, 3, 67, 68, 71, 74, 75, 81–84, 86, 88–90, 92, 95–97, 108, 132

MAP maximum a posteriori estimation. 40

MCMC Markov chain Monte Carlo. 40, 59, 60, 62–64, 67, 107, 108

MH Metropolis–Hastings. 57, 59, 61

MPS matrix-product states. 16

mRNA messenger ribonucleic acid. 53, 61

NURBS non-uniform rational B-Spline. x, 2, 3, 67–70, 72, 74–76, 78–80, 87, 96

ODE ordinary differential equation. 2, 42, 43, 53, 55, 57

PDE partial differential equation. vii, 2, 4, 67, 87, 96

PDF probability density function. 34–40, 47, 49–52, 60, 62–64, 66, 86, 90, 91, 94–96, 107, 108

PMF probability mass function. 34, 42, 43, 47, 49–52, 55–57, 107

QTT quantized tensor-train. 26, 46, 55, 57, 59, 63, 64, 81–83, 85

RV random variable. 33–39, 42, 47, 49–51, 90, 92, 107

SSA stochastic simulation algorithm. 43, 55, 57, 61

SVD singular value decomposition. 16, 19–22, 25, 30

TT tensor-train. vii, ix, x, 1–3, 15–32, 39, 41, 43, 44, 46, 48–57, 59, 61, 63, 64, 67, 68, 75–86, 88–92, 95, 96, 107, 108, 111, 113, 132

UQ uncertainty quantification. 1, 39, 67, 96, 129

List of Figures

List of Tables

List of Algorithms

Bibliography

[ABC+15] P Antolin, Annalisa Buffa, F Calabro, M Martinelli, and G Sangalli. Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization. *Computer Methods in Applied Mechanics and Engineering*, 285:817–828, 2015.

[ABKF11] I Akkerman, Y Bazilevs, Chris E Kees, and Matthew W Farthing. Isogeometric analysis of free-surface flow. *Journal of Computational Physics*, 230(11):4137–4152, 2011.

[ABW22] Ben Adcock, Simone Brugiapaglia, and Clayton G Webster. *Sparse Polynomial Approximation of High-Dimensional Functions*, volume 25. SIAM, 2022.

[ACH+12] F Auricchio, Francesco Calabro, Thomas JR Hughes, A Reali, and G30030591348 Sangalli. A simple algorithm for obtaining nearly optimal quadrature rules for nurbs-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 249:15–27, 2012.

[ADL21] Hussam Al Daas and Damiano Lombardi. An extended krylov-like method for the solution of multi-linear systems. 2021.

[AIJZ17] Simon Arridge, Kazufumi Ito, Bangti Jin, and Chen Zhang. Variational gaussian approximation for poisson data. *Inverse Problems*, 34, 09 2017.

[AJL+02] B. Alberts, A. Johnson, J. Lewis, P. Walter, M. Raff, and K. Roberts. *Molecular Biology of the Cell 4th Edition: International Student Edition*. Routledge, 2002.

[aSYK+04] Masa aki Sato, Taku Yoshioka, Shigeki Kajihara, Keisuke Toyama, Naokazu Goda, Kenji Doya, and Mitsuo Kawato. Hierarchical bayesian estimation for meg inverse problem. *NeuroImage*, 23(3):806–826, 2004.

[Bar12] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[Bar18] J.M. Bardsley. *Computational Uncertainty Quantification for Inverse Problems*. Computer Science and Engineering. Society for Industrial and Applied Mathematics, 2018.

[BCC57] R. Bellman, Rand Corporation, and Karreman Mathematics Research Collection. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957.

[BCdF+20] Annalisa Buffa, Jacopo Corno, Carlo de Falco, Sebastian Schöps, and Rafael Vázquez Hernández. Isogeometric mortar coupling for electromagnetic problems. *SIAM Journal on Scientific Computing*, 42(1):B80–B104, 2020.

[BCDM17] Markus Bachmayr, Albert Cohen, Ronald DeVore, and Giovanni Migliorati. Sparse polynomial approximation of parametric elliptic pdes. part ii: lognormal coefficients. *ESAIM: Mathematical Modelling and Numerical Analysis*, 51(1):341–363, 2017.

[BCM17] Markus Bachmayr, Albert Cohen, and Giovanni Migliorati. Sparse polynomial approximation of parametric elliptic pdes. part i: affine coefficients. *ESAIM: Mathematical Modelling and Numerical Analysis*, 51(1):321–339, 2017.

[BDG+15] G. Ballard, J. Demmel, L. Grigori, M. Jacquelin, N. Knight, and H.D. Nguyen. Reconstructing householder vectors from tall-skinny qr. *Journal of Parallel and Distributed Computing*, 85:3–31, 2015. IPDPS 2014 Selected Papers on Numerical and Combinatorial Algorithms.

[Beb00] Mario Bebendorf. Approximation of boundary element matrices. *Numerische Mathematik*, 86(4):565–589, 2000.

[BGJM11] S. Brooks, A. Gelman, G. Jones, and X.L. Meng. *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC Press, 2011.

[BH08] Yuri Bazilevs and TJR Hughes. Nurbs-based isogeometric analysis for the computation of flows about rotating components. *Computational Mechanics*, 43(1):143–150, 2008.

[BKM16] David Blei, Alp Kucukelbir, and Jon McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112, 01 2016.

[BNR00] Volker Barthelmann, Erich Novak, and Klaus Ritter. High dimensional polynomial interpolation on sparse grids. *Adv. Comput. Math.*, 12:273–288, 03 2000.

[BNT07] Ivo Babuska, Fabio Nobile, and Raul Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM J. Numerical Analysis*, 45:1005–1034, 01 2007.

[BSV10] Annalisa Buffa, Giancarlo Sangalli, and Rafael Vázquez. Isogeometric analysis in electromagnetics: B-splines approximation. *Computer Methods in Applied Mechanics and Engineering*, 199(17-20):1143–1152, 2010.

[BTZ04] Ivo Babuska, Raúl Tempone, and Georgios E. Zouraris. Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, 42(2):800–825, 2004.

[Bun04] Hans-Joachim Bungartz. Sparse grids. *Acta Numerica*, 13:147 – 269, 2004.

[But16] J C Butcher. *Numerical Differential Equation Methods*. John Wiley and Sons, Ltd, 2016.

[CCNT16] Julio E Castrillon-Candas, Fabio Nobile, and Raul F Tempone. Analytic regularity and collocation approximation for elliptic pdes with random domain deformations. *Computers & Mathematics with Applications*, 71(6):1173–1197, 2016.

[CCNT21] Julio E Castrillón-Candás, Fabio Nobile, and Raúl F Tempone. A hybrid collocation-perturbation approach for pdes with random domains. *Advances in Computational Mathematics*, 47(3):1–35, 2021.

[CDFS13] Nabi Chegini, Stephan Dahlke, Ulrich Friedrich, and Rob Stevenson. Piecewise tensor product wavelet bases by extensions and approximation rates. *Mathematics of Computation*, 82(284):2157–2190, 2013.

[CFO11] Tiangang Cui, Colin Fox, and Michael O'Sullivan. Bayesian calibration of a large-scale geothermal reservoir model by a new adaptive delayed acceptance metropolis hastings algorithm. *Water Resources Research - WATER RESOUR RES*, 47, 10 2011.

[CHB09] J Austin Cottrell, Thomas JR Hughes, and Yuri Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[Che03] Zhe Chen. Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, 182, 01 2003.

[Che07] *Inverse Problems in Electrical Circuits and Electromagnetic Field Theory*, pages 1–46. Springer US, Boston, MA, 2007.

[CMI09] Ali Civril and Malik Magdon-Ismail. On selecting a maximum volume sub-matrix of a matrix and related problems. *Theoretical Computer Science*, 410(47):4801–4811, 2009.

[CRBH06] J Austin Cottrell, Alessandro Reali, Yuri Bazilevs, and Thomas JR Hughes. Isogeometric analysis of structural vibrations. *Computer methods in applied mechanics and engineering*, 195(41-43):5257–5296, 2006.

[DAIFS20] Sergey Dolgov, Karim Anaya-Izquierdo, Colin Fox, and Robert Scheichl. Approximation and sampling of multivariate probability distributions in the tensor train decomposition. *Statistics and Computing*, 30, 05 2020.

[dB78] Carl de Boor. *A Practical Guide to Spline*, volume Volume 27. 01 1978.

[DBB22] Hussam Al Daas, Grey Ballard, and Peter Benner. Parallel algorithms for tensor train arithmetic. *SIAM Journal on Scientific Computing*, 44(1):C25–C53, 2022.

[DG08] Guy Demoment and Yves Goussard. *Inversion within the Probabilistic Framework*, chapter 3, pages 59–78. John Wiley and Sons, Ltd, 2008.

[DGGI+20] Herbert De Gersem, Armin Galetzka, Ion Gabriel Ion, Dimitrios Loukrezis, and Ulrich Römer. Magnetic field simulation with data-driven material modeling. *IEEE Transactions on Magnetics*, 56(8):1–6, 2020.

[DHT81] Michel Delfour, W. Hager, and Francois Trochu. Discontinuous galerkin methods for ordinary differential equations. *Mathematics of Computation*, 36, 05 1981.

[DK15] Sergey Dolgov and B. Khoromskij. Simultaneous state-time approximation of the chemical master equation using tensor product formats. *Numerical Linear Algebra with Applications*, 22, 03 2015.

[DKLM15] Sergey Dolgov, Boris N Khoromskij, Alexander Litvinenko, and Hermann G Matthies. Polynomial chaos expansion of random coefficients and the solution of stochastic partial differential equations in the tensor train format. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):1109–1135, 2015.

[DKO12] Sergey V Dolgov, Boris N Khoromskij, and Ivan V Oseledets. Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the fokker–planck equation. *SIAM Journal on Scientific Computing*, 34(6):A3016–A3038, 2012.

[DKSW19a] Jürgen Dölz, Stefan Kurz, Sebastian Schöps, and Felix Wolf. Isogeometric boundary elements in electromagnetism: rigorous analysis, fast methods, and examples. *SIAM Journal on Scientific Computing*, 41(5):B983–B1010, 2019.

[DKSW19b] Jürgen Dölz, Stefan Kurz, Sebastian Schöps, and Felix Wolf. A numerical comparison of an isogeometric and a parametric higher order raviart–thomas approach to the electric field integral equation. *IEEE Transactions on Antennas and Propagation*, 68(1):593–597, 2019.

[DLDMV00a] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[DLDMV00b] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-(r1 ,r2 ,. . .,rn) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000.

[Dol13] Sergey Dolgov. TT-GMRES: Solution to a linear system in the structured tensor format. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 28, 04 2013.

[Dol18] Sergey Dolgov. A Tensor Decomposition Algorithm for Large ODEs with Conservation Laws. *Computational Methods in Applied Mathematics*, 19, 09 2018.

[DS14] Sergey V Dolgov and Dmitry V Savostyanov. Alternating minimal energy methods for linear systems in higher dimensions. *SIAM Journal on Scientific Computing*, 36(5):A2248–A2271, 2014.

[DS19] Sergey Dolgov and Robert Scheichl. A hybrid alternating least squares–tt-cross algorithm for parametric pdes. *SIAM/ASA Journal on Uncertainty Quantification*, 7(1):260–291, 2019.

[DS20] Trang Dinh and Roger B Sidje. An adaptive solution to the chemical master equation using quantized tensor trains with sliding windows. *Physical Biology*, 2020.

[DVBSV14] L Beirao Da Veiga, Annalisa Buffa, Giancarlo Sangalli, and Rafael Vázquez. Mathematical analysis of variational isogeometric methods. *Acta Numerica*, 23:157–287, 2014.

[DZW+20] Xiaoxiao Du, Gang Zhao, Wei Wang, Mayi Guo, Ran Zhang, and Jiaming Yang. Nliga: A matlab framework for nonlinear isogeometric analysis. *Computer Aided Geometric Design*, 80:101869, 2020.

[EHN00] H.W. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Mathematics and Its Applications. Springer Netherlands, 2000.

[EMM20] Martin Eigel, Manuel Marschall, and Michael Multerer. An adaptive stochastic Galerkin tensor train discretization for randomly perturbed domains. *SIAM/ASA Journal on Uncertainty Quantification*, 8(3):1189–1214, 2020.

[FLU+20] I.-G. Farcaş, J. Latz, E. Ullmann, T. Neckel, and H.-J. Bungartz. Multilevel adaptive sparse leja approximations for bayesian inverse problems. *SIAM Journal on Scientific Computing*, 42(1):A424–A451, 2020.

[FMWW10] M. Frangos, Youssef Marzouk, K. Willcox, and B. Waanders. *Surrogate and Reduced-Order Modeling: A Comparison of Approaches for Large-Scale Statistical Inverse Problems*, pages 123 – 149. 10 2010.

[FSV15] Daniela Fußeder, Bernd Simeon, and A-V Vuong. Fundamental aspects of shape optimization in the context of isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 286:313–331, 2015.

[GACS19] Niklas Georg, Wolfgang Ackermann, Jacopo Corno, and Sebastian Schöps. Uncertainty quantification for Maxwell's eigenproblem based on isogeometric analysis and mode tracking. *Computer Methods in Applied Mechanics and Engineering*, 350:228–244, 2019.

[Gel17] Patrick Gelß. *The Tensor-Train Format and Its Applications: Modeling and Analysis of Chemical Reaction Networks, Catalytic Processes, Fluid Flows, and Brownian Dynamics*. PhD thesis, 2017.

[GG03] Thomas Gerstner and Michael Griebel. Dimension adaptive tensor product quadrature. *Computing*, 71:65–87, 09 2003.

[GI15] Jean-François Giovannelli and Jérôme Idier. *Regularization and Bayesian Methods for Inverse Problems in Signal and Image Processing*. 02 2015.

[Gil76] Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of computational physics*, 22(4):403–434, 1976.

[Gil92] Daniel T Gillespie. A rigorous derivation of the chemical master equation. *Physica A: Statistical Mechanics and its Applications*, 188(1-3):404–425, 1992.

[GKMS17] Patrick Gelß, Stefan Klus, Sebastian Matera, and Christof Schütte. Nearest-neighbor interaction systems in the tensor-train format. *Journal of Computational Physics*, 341:140–162, 2017.

[GKT13] Lars Grasedyck, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

[GOS+10] Sergei Goreinov, Ivan Oseledets, D. Savostyanov, E. Tyrtyshnikov, and Nickolai Zamarashkin. How to find a good submatrix. *Matrix Methods: Theory, Algorithms and Applications*, 04 2010.

[GPC19] Daniel Garcia, David Pardo, and Victor M Calo. Refined isogeometric analysis for fluid mechanics and electromagnetics. *Computer Methods in Applied Mechanics and Engineering*, 356:598–628, 2019.

[GS91] Roger G. Ghanem and Pol D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Springer-Verlag, Berlin, Heidelberg, 1991.

[GT01] Sergei Goreinov and E. Tyrtyshnikov. The maximal-volume concept in approximation by low-rank matrices. *Contemporary Mathematics*, 208, 01 2001.

[GWE+15] Nilabja Guha, Xiaoqing Wu, Yalchin Efendiev, Bangti Jin, and Bani Mallick. A variational bayesian approach for inverse problems with skew-t error distributions. *Journal of Computational Physics*, 301, 08 2015.

[HAB11] Ming-Chen Hsu, Ido Akkerman, and Yuri Bazilevs. High-performance computing of wind turbine aerodynamics using isogeometric analysis. *Computers & Fluids*, 49(1):93–100, 2011.

[Hac12] Wolfgang Hackbusch. *Tensor spaces and numerical tensor calculus*, volume 42. Springer, 2012.

[HCB05] Thomas JR Hughes, John A Cottrell, and Yuri Bazilevs. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer methods in applied mechanics and engineering*, 194(39-41):4135–4195, 2005.

[Het00] Herbert Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42:599–653, 12 2000.

[HG11] Markus Hegland and Jochen Garcke. On the numerical solution of the chemical master equation with sums of rank one tensors. *ANZIAM Journal*, 52, 08 2011.

[Hit] Frank Lauren Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6:164–189.

[HKT05] Wolfgang Hackbusch, Boris N Khoromskij, and Eugene E Tyrtyshnikov. Hierarchical kronecker tensor-product approximations. 2005.

[HKT08] Wolfgang Hackbusch, B. Khoromskij, and E. Tyrtyshnikov. Approximate iterations for structured matrices. *Numerische Mathematik*, 109:365–383, 04 2008.

[Hof18] Clemens Hofreither. A black-box low-rank approximation algorithm for fast matrix assembly in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 333:311–330, 2018.

[HPZ+16] Lirong Huang, Loic Pauleve, Christoph Zechner, Michael Unger, Anders S Hansen, and Heinz Koeppl. Reconstructing dynamic molecular states from single-cell time series. *Journal of The Royal Society Interface*, 13(122):20160533, 2016.

[HRM+21] Ehsan Haghighat, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741, 2021.

[HRS10a] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. On manifolds of tensors of fixed tt-rank. *Numer. Math*, 120:701–731, 01 2010.

[HRS10b] T.J.R. Hughes, A. Reali, and G. Sangalli. Efficient quadrature for nurbs-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 199(5):301–313, 2010. Computational Geometry and Analysis.

[HRS12] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal on Scientific Computing*, 34(2):A683–A713, 2012.

[HSS08] Helmut Harbrecht, Reinhold Schneider, and Christoph Schwab. Sparse second moment analysis for elliptic problems in stochastic domains. *Numerische Mathematik*, 109:385–414, 04 2008.

[HSSS18] Ralf Hiptmair, Laura Scarabosio, Claudia Schillings, and Christoph Schwab. Large deformation shape uncertainty quantification in acoustic scattering. *Adv. Comput. Math.*, 44(5):1475–1518, 2018.

[Hå90] Johan Håstad. Tensor rank is np-complete. *Journal of Algorithms*, 11(4):644–654, 1990.

[ILDG22] Ion Gabriel Ion, Dimitrios Loukrezis, and Herbert De Gersem. Tensor train based isogeometric analysis for pde approximation on parameter dependent geometries. *arXiv preprint arXiv:2204.02843*, 2022.

[IWL+21] Ion Gabriel Ion, Christian Wildner, Dimitrios Loukrezis, Heinz Koeppl, and Herbert De Gersem. Tensor-train approximation of the chemical master equation and its application for parameter inference. *The Journal of Chemical Physics*, 155(3):034102, 2021.

[Jay03] E. T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[Jec02] Eric Jeckelmann. Dynamical density-matrix renormalization-group method. *Phys. Rev. B*, 66:045114, Jul 2002.

[JM17a] Bert Jüttler and Dominik Mokriš. Low rank interpolation of boundary spline curves. *Computer Aided Geometric Design*, 55:48–68, 2017.

[JM17b] Bert Jüttler and Dominik Mokriš. Low rank interpolation of boundary spline curves. *Computer Aided Geometric Design*, 55:48–68, 2017.

[Joh05] Richard W Johnson. Higher order b-spline collocation at the greville abscissae. *Applied Numerical Mathematics*, 52(1):63–75, 2005.

[Kal60] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960.

[KB09] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[Kho11] Boris N Khoromskij. O (d log n)-quantics approximation of n-d tensors in high-dimensional numerical modeling. *Constructive Approximation*, 34(2):257–280, 2011.

[Kho12] B. Khoromskij. Tensors-structured numerical methods in scientific computing: Survey on recent advances. *Chemometrics and Intelligent Laboratory Systems*, 110:1–19, 01 2012.

[KKNS14] Vladimir Kazeev, Mustafa Khammash, Michael Nip, and Christoph Schwab. Direct Solution of the Chemical Master Equation Using Quantized Tensor Trains. *PLOS Computational Biology*, 10(3):1–19, 03 2014.

[KO10] Boris N Khoromskij and I Oseledets. Quantics-TT collocation approximation of parameter-dependent and stochastic elliptic pdes. *Computational methods in applied mathematics*, 10(4):376–394, 2010.

[KO11] Boris N Khoromskij and Ivan V Oseledets. Qtt approximation of elliptic solution operators in higher dimensions. 2011.

[KS11] Boris N. Khoromskij and Christoph Schwab. Tensor-structured galerkin approximation of parametric and stochastic elliptic pdes. *SIAM Journal on Scientific Computing*, 33(1):364–385, 2011.

[KT10] V. Kazeev and E. Tyrtyshnikov. Structure of the hessian matrix and an economical implementation of newton's method in the problem of canonical approximation of tensors. *Computational Mathematics and Mathematical Physics*, 50:927–945, 06 2010.

[LBJ19] Ning Liu, Paul A Beata, and Ann E Jeffers. A mixed isogeometric analysis and control volume approach for heat transfer analysis of nonuniformly heated plates. *Numerical Heat Transfer, Part B: Fundamentals*, 75(6):347–362, 2019.

[LD20] Dimitrios Loukrezis and Herbert De Gersem. Approximation and Uncertainty Quantification of Systems with Arbitrary Parameter Distributions using Weighted Leja Interpolation. *Algorithms*, 2020.

[LDG19] Dimitrios Loukrezis and Herbert De Gersem. Adaptive sparse polynomial chaos expansions via leja interpolation. *arXiv preprint arXiv:1911.08312*, 2019.

[LGDG20] Dimitrios Loukrezis, Armin Galetzka, and Herbert De Gersem. Robust adaptive least squares polynomial chaos expansions in high-frequency applications. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, 33(6):e2725, 2020.

[Liu21] Y. Liu. *Tensors for Data Processing: Theory, Methods, and Applications*. Elsevier Science, 2021.

[LJ09] Stan Z. Li and Anil Jain, editors. *Forward-Backward Algorithm*, pages 580–580. Springer US, Boston, MA, 2009.

[LM14] Jinglai Li and Youssef M Marzouk. Adaptive construction of surrogates for the bayesian solution of inverse problems. *SIAM Journal on Scientific Computing*, 36(3):A1163–A1186, 2014.

[LMMT14] Ulrich Langer, Angelos Mantzaflaris, Stephen Moore, and Ioannis Toulopoulos. Multipatch discontinuous galerkin isogeometric analysis. *Lecture Notes in Computational Science and Engineering*, 107:1–32, 11 2014.

[LRC+18] Dimitrios Loukrezis, Ulrich Römer, Thorben Casper, Sebastian Schöps, and Herbert De Gersem. High-dimensional uncertainty quantification for an electrothermal field problem using stochastic collocation on sparse grids and tensor train decompositions. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, 31(2):e2222, 2018.

[LWY+21] Donghyun Lee, Dingheng Wang, Yukuan Yang, Lei Deng, Guangshe Zhao, and Guoqi Li. Qttnet: Quantized tensor train neural networks for 3d object and video recognition. *Neural Networks*, 141:420–432, 2021.

[M+03] P. Monk et al. *Finite Element Methods for Maxwell's Equations*. Numerical Mathematics and Scie. Clarendon Press, 2003.

[MAB+15] S Morganti, F Auricchio, DJ Benson, FI Gambarin, S Hartmann, TJR Hughes, and A Reali. Patient-specific isogeometric structural analysis of aortic valve closure. *Computer methods in applied mechanics and engineering*, 284:508–520, 2015.

[Mal02] Alberto Malinverno. Parsimonious bayesian markov chain monte carlo inversion in a nonlinear geophysical problem. *Geophysical Journal International*, 151:675 – 688, 11 2002.

[MGS21] Melina Merkel, Peter Gangl, and Sebastian Schöps. Shape optimization of rotating electric machines using isogeometric analysis. *IEEE Transactions on Energy Conversion*, 2021.

[MH02] J.C. Mason and D.C. Handscomb. *Chebyshev Polynomials*. Taylor & Francis, 2002.

[MJ15] Angelos Mantzaflaris and Bert Jüttler. Integration by interpolation and look-up for galerkin-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 284:373–400, 2015.

[MJKL17] Angelos Mantzaflaris, Bert Jüttler, Boris N Khoromskij, and Ulrich Langer. Low rank tensor methods in galerkin-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 316:1062–1085, 2017.

[MK06] Brian Munsky and Mustafa Khammash. The finite state projection algorithm for the solution of the chemical master equation. *The Journal of Chemical Physics*, 124(4):044104, January 2006.

[MTRP09] Clare McGrory, D. Titterington, Robert Reeves, and Anthony Pettitt. Variational bayes for estimating the parameters of a hidden potts model. *Statistics and Computing*, 19, 09 2009.

[MX09] Youssef Marzouk and Dongbin Xiu. A stochastic collocation approach to bayesian inference in inverse problems. *PRISM: NNSA Center for Prediction of Reliability, Integrity and Survivability of Microsystems*, 6, 10 2009.

[MZR+16] Hermann G. Matthies, Elmar Zander, Bojana V. Rosić, Alexander Litvinenko, and Oliver Pajonk. *Inverse Problems in a Bayesian Setting*, pages 245–286. Springer International Publishing, Cham, 2016.

[MZRL16] Hermann Matthies, Elmar Zander, Bojana Rosic, and Alexander Litvinenko. Parameter estimation via conditional expectation — a bayesian inversion. *Advanced Modeling and Simulation in Engineering Sciences*, 3, 06 2016.

[NABR15] Vinh Phu Nguyen, Cosmin Anitescu, Stéphane PA Bordas, and Timon Rabczuk. Isogeometric analysis: an overview and computer implementation aspects. *Mathematics and Computers in Simulation*, 117:89–116, 2015.

[NDM+09] H. N. Najm, B. J. Debusschere, Y. M. Marzouk, S. Widmer, and O. P. Le Maître. Uncertainty quantification in chemical systems. *International Journal for Numerical Methods in Engineering*, 80(6-7):789–814, 2009.

[NRO22] Alexander Novikov, Maxim Rakhuba, and Ivan Oseledets. Automatic differentiation for riemannian optimization on low-rank matrix and tensor-train manifolds. *SIAM Journal on Scientific Computing*, 44:A843–A869, 04 2022.

[OD12] Ivan V Oseledets and Sergey V Dolgov. Solution of linear systems and matrix inversion in the tt-format. *SIAM Journal on Scientific Computing*, 34(5):A2718–A2739, 2012.

[OL79] Steven Orey and Michel Loève. Probability theory ii (4th ed.). *Journal of the American Statistical Association*, 74:725, 1979.

[Ose10] Ivan V Oseledets. Approximation of $2^d \times 2^d$ matrices using tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2130–2145, 2010.

[Ose11a] Ivan Oseledets. Dmrg approach to fast linear algebra in the tt-format. *Computational Methods in Applied Mathematics*, 11(3):382–393, 2011.

[Ose11b] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[OT09] Ivan V Oseledets and Eugene E Tyrtyshnikov. Breaking the curse of dimensionality, or how to use SVD in many dimensions. *SIAM Journal on Scientific Computing*, 31(5):3744–3759, 2009.

[OT10] Ivan Oseledets and Eugene Tyrtyshnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88, 2010.

[PBC+15] Andreas Pels, Zeger Bontinck, Jacopo Corno, Herbert De Gersem, and Sebastian Schöps. Optimization of a stern-gerlach magnet by magnetic field-circuit coupling and isogeometric analysis. *IEEE Transactions on Magnetics*, 51, 07 2015.

[PC19] Maodong Pan and Falai Chen. Low-rank parameterization of volumetric domains for isogeometric analysis. *Computer-Aided Design*, 114:82–90, 2019.

[PCT20] Maodong Pan, Falai Chen, and Weihua Tong. Volumetric spline parameterization for isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 359:1–19, 02 2020.

[Pen71] Roger Penrose. Applications of negative dimensional tensors. *Combinatorial Mathematics and its Applications*, 1971.

[PPM+22] M. R. Perelshtein, A. I. Pakhomchik, A. A. Melnikov, A. A. Novikov, A. Glatz, G. S. Paraoanu, V. M. Vinokur, and G. B. Lesovik. Solving large-scale linear systems of equations by a quantum hybrid algorithm. *Annalen der Physik*, 534(7):2200082, 2022.

[PT96] Les Piegl and Wayne Tiller. *The NURBS book*. Springer Science & Business Media, 1996.

[QLG+22] Zhen Qin, Alexander Lidiak, Zhexuan Gong, Gongguo Tang, Michael Wakin, and Zhihui Zhu. Error analysis of tensor-train cross approximation, 07 2022.

[Rea06] Allessandro Reali. An isogeometric analysis approach for the study of structural vibrations. *Journal of Earthquake Engineering*, 10(spec01):1–30, 2006.

[Ric85] John R Rice. Adaptive tensor product grids for singular problems. 1985.

[Rom08] Steven Roman. *Tensor Products*, pages 355–409. Springer New York, New York, NY, 2008.

[RP12] D Rypl and B Patzák. Study of computational efficiency of numerical quadrature schemes in the isogeometric analysis. *Engineering Mechanics*, 304, 2012.

[RU13] Thorsten Rohwedder and André Uschmajew. On local convergence of alternating schemes for optimization of convex problems in the tensor train format. *SIAM Journal on Numerical Analysis*, 51(2):1134–1162, 2013.

[RWGG21] Anna Rörich, Tim A. Werthmann, Dominik Göddeke, and Lars Grasedyck. Bayesian inversion for electromyography using low-rank tensor formats. *Inverse Problems*, 37(5):055003, mar 2021.

[SBdFS20] Abele Simona, Luca Bonaventura, Carlo de Falco, and Sebastian Schöps. Isogeometric approximations for electromagnetic problems in axisymmetric domains. *Computer Methods in Applied Mechanics and Engineering*, 369:113211, 2020.

[SCS+22] Konstantin Sozykin, Andrei Chertkov, Roman Schutski, Anh-Huy Phan, Andrzej Cichocki, and Ivan Oseledets. Ttopt: A maximum volume quantized tensor train-based optimization and its application to reinforcement learning. *arXiv preprint arXiv:2205.00293*, 2022.

[SKBW10] Robert Schmidt, Josef Kiendl, K-U Bletzinger, and Roland Wüchner. Realization of an integrated structural design process: analysis-suitable geometric modelling and isogeometric analysis. *Computing and Visualization in Science*, 13(7):315–330, 2010.

[SO11] Dmitry Savostyanov and Ivan Oseledets. Fast adaptive interpolation of multi-dimensional arrays in tensor train format. In *The 2011 International Workshop on Multidimensional (nD) Systems*, pages 1–8. IEEE, 2011.

[SS08] Vahid Shahrezaei and Peter Swain. Analytic distributions for stochastic gene expression. *Proceedings of the National Academy of Sciences of the United States of America*, 105:17256–61, 12 2008.

[SSSS93] Gilbert Strang, Gilbert Strang, Gilbert Strang, and Gilbert Strang. *Introduction to linear algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA, 1993.

[Ste07] Olaf Steinbach. Numerical approximation methods for elliptic boundary value problems: Finite and boundary elements. 2007.

[Stu10] A. M. Stuart. Inverse problems: A Bayesian perspective. *Acta Numerica*, 19:451–559, 2010.

[SYSY02] R. SRIVASTAVA, L. YOU, J. SUMMERS, and J. YIN. Stochastic vs. deterministic modeling of intracellular viral kinetics. *Journal of Theoretical Biology*, 218(3):309–321, 2002.

[Tar05] Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, 2005.

[TB18] Rohit K. Tripathy and Ilias Bilionis. Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of Computational Physics*, 375:565–588, 2018.

[The15] Sergios Theodoridis. *Machine Learning: A Bayesian and Optimization Perspective*. Academic Press, Inc., USA, 1st edition, 2015.

[Tie94] Luke Tierney. Markov chains for exploring posterior distributions. *Annals of Statistics*, 22:1701–1728, 1994.

[Tre00] L.N. Trefethen. *Spectral Methods in MATLAB*. Software, Environments, and Tools. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2000.

[TRT+20] Francesco Tonolini, Jack Radford, Alex Turpin, Daniele Faccio, and Roderick Murray-Smith. Variational inference for computational imaging inverse problems. *J. Mach. Learn. Res.*, 21(1), jan 2020.

[Tuc66] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, September 1966.

[Tyr00] E. Tyrtyshnikov. Incomplete cross approximation in the mosaic-skeleton method. *Computing (Vienna/New York)*, 64:367–380, 06 2000.

[VC06] F. Verstraete and J. I. Cirac. Matrix product states represent ground states faithfully. *Phys. Rev. B*, 73:094423, Mar 2006.

[Vid03] Guifré Vidal. Efficient classical simulation of slightly entangled quantum computations. *Phys. Rev. Lett.*, 91:147902, Oct 2003.

[VL00] Charles F Van Loan. The ubiquitous kronecker product. *Journal of computational and applied mathematics*, 123(1-2):85–100, 2000.

[VM18] Jaroslav Vondrejc and Hermann Matthies. Accurate computation of conditional expectation for highly non-linear problems. 06 2018.

[WGMH10] Verena Wolf, Rushil Goel, Maria Mateescu, and Thomas A. Henzinger. Solving the chemical master equation using sliding windows. 4(1):42, April 2010.

[Whi93] Steven R. White. Density-matrix algorithms for quantum renormalization groups. *Phys. Rev. B*, 48:10345–10356, Oct 1993.

[Wri08] Peter Wriggers. *Nonlinear Finite Element Methods*, volume 4. 01 2008.

[WWS13] Oliver Weeger, U. Wever, and Bernd Simeon. Isogeometric analysis of nonlinear euler-bernoulli beam vibrations. *Nonlinear Dynamics*, 72, 01 2013.

[WWX+17] Chenglong Wang, Michael CH Wu, Fei Xu, Ming-Chen Hsu, and Yuri Bazilevs. Modeling of a hydraulic arresting gear using fluid–structure interaction and isogeometric analysis. *Computers & Fluids*, 142:3–14, 2017.

[WWXP18] Yingjun Wang, Zhenpei Wang, Zhaohui Xia, and Leong Hien Poh. Structural design optimization using isogeometric analysis: a comprehensive review. *Computer Modeling in Engineering & Sciences*, 117(3):455–507, 2018.

[Xiu10] Dongbin Xiu. *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press, Princeton, 2010.

[YZ19a] Liang Yan and Tao Zhou. Adaptive multi-fidelity polynomial chaos approach to bayesian inference in inverse problems. *Journal of Computational Physics*, 381:110–128, 2019.

[YZ19b] Liang Yan and Tao Zhou. An adaptive surrogate modeling based on deep neural networks for large-scale bayesian inverse problems. *arXiv preprint arXiv:1911.08926*, 2019.

[YZ21] Liang Yan and Tao Zhou. An acceleration strategy for randomize-then-optimize sampling via deep neural networks. *arXiv preprint arXiv:2104.06285*, 2021.

[ZGAS22] Anna Ziegler, Niklas Georg, Wolfgang Ackermann, and Sebastian Schöps. Mode recognition by shape morphing for maxwell's eigenvalue problem. *arXiv preprint arXiv:2203.00499*, 2022.

[ZS19] Hongguan Zhang and Tadahiro Shibutani. Development of stochastic isogeometric analysis (SIGA) method for uncertainty in shape. *International Journal for Numerical Methods in Engineering*, 118(1):18–37, 2019.

[ZYO+14] Zheng Zhang, Xiu Yang, Ivan V Oseledets, George E Karniadakis, and Luca Daniel. Enabling high-dimensional hierarchical uncertainty quantification by anova and tensor-train decomposition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(1):63–76, 2014.

Acknowledgements

Finally, I would like to express my gratitude to:

- Prof. Dr. Ing. Herbert De Gersem for his supervision and for offering me the opportunity to start the PhD.
- Dr. Ing. Dimitrios Loukrezis for the supervision and for all the fruitful discussion about science and many other things. His careful eye for typos, grammar mistakes and text inconsistencies was always appreciated (also for this thesis). Moreover, I would like to thank him for always having the time to reply to my questions.
- Prof. Dr. Ing. Ulrich Römer from TU Braunschweig for the fruitful discussions on UQ and for being co-rederee for my thesis.
- Prof. Dr. techn. Heinz Koeppl and Dr. Christian Wildner for the fruitful collaboration and for their valuable feedback regarding the CME work.
- Dr. Melvin Liebsch for providing the measurements and for his feedback regarding the magnetic field reconstruction project.
- My colleagues from the scientific machine learning group, Armin Galetzka and Moritz von Tresckow, for all the scientific and non-scientific discussions.
- Nu în ultimul rând aş dori să mulţumesc parinţilor mei şi Isabelei pentru tot suportul oferit pe parcursul celor 3 ani şi jumătate de doctorat.
- Last by not least, I should not forget to mention: This work has been supported by the Graduate School Computational Engineering within the Centre for Computational Engineering at the Technische Universität Darmstadt.

Curriculum Vitae

Education

- PhD Candidate in Electrical Engineering and Information Technology, May 2019 –now
  Technische Universität Darmstadt, Darmstadt, Germany,
- Master of Science in Computational Engineering, Apr 2017 –Apr 2019
  Technische Universität Darmstadt, Darmstadt, Germany,
- Bachelor of Science in Electrical Engineering, Oct 2014 –Mar 2017
  Technische Universität Darmstadt, Darmstadt, Germany,

Scholarships and Awards

- Young Scientist Award, International Union of Radio Science Sep 2020
- Best student award for the B.Sc. study Jul 2017
- Helmut Böhme Stipendium Oct 2014–Jul 2015

Publications

1. Ion, I.G., Bontinck, Z., Loukrezis, D., Römer, U., Lass, O., Ulbrich, S., Schöps, S. and De Gersem, H., 2018. Robust shape optimization of electric devices based on deterministic optimization methods and finite-element analysis with affine parametrization and design elements. Electrical Engineering, 100(4), pp.2635-2647.

2. De Gersem, H., Galetzka, A., Ion, I.G., Loukrezis, D. and Römer, U., 2020. Magnetic field simulation with data-driven material modeling. IEEE Transactions on Magnetics, 56(8), pp.1-6.

3. Ion, I.G., Wildner, C., Loukrezis, D., Koeppl, H. and De Gersem, H., 2021. Tensor-train approximation of the chemical master equation and its application for parameter inference. The Journal of Chemical Physics, 155(3), p.034102.

4. Ion, I.G., Liebsch, M., Simona, A., Loukrezis, D., Petrone, C., Russenschuck, S., De Gersem, H. and Schöps, S., 2021. Local field reconstruction from rotating coil measurements in particle accelerator magnets. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 1011, p.165580.

5. Ion, I.G., Loukrezis, D. and De Gersem, H., 2022. Tensor train based isogeometric analysis for PDE approximation on parameter dependent geometries. arXiv preprint arXiv:2204.02843.

Scientific software

- Python TT toolbox: `https://github.com/ion-g-ion/torchTT`
- Simulation of chemical reaction networks: `https://github.com/ion-g-ion/tt-cme`
- TT enhanced IGA solver: `https://github.com/ion-g-ion/tt-iga`
- Physics-informed (isogeometric) neural networks: `https://github.com/ion-g-ion/PINNs`