UNIVERSITY OF CALIFORNIA
SANTA CRUZ

TENSOR METHODS FOR HIGH-DIMENSIONAL PARTIAL DIFFERENTIAL EQUATIONS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

APPLIED MATHEMATICS

by

Alec Dektor

June 2023

The Dissertation of Alec Dektor
is approved:

Daniele Venturi, Chair

Nicholas Brummell

Dongwook Lee

Peter Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

Abstract

TENSOR METHODS FOR HIGH-DIMENSIONAL PARTIAL DIFFERENTIAL EQUATIONS

by

Alec Dektor

The numerical simulation of high-dimensional partial differential equations (PDEs) is a challenging and important problem in science and engineering. Classical methods based on tensor product representations are not viable in high-dimensions, as the number of degrees of freedom grows exponentially fast with the problem dimension. In this dissertation we present low-rank tensor methods for approximating high-dimensional PDEs, which have a number of degrees of freedom and computational cost that grow linearly with the problem dimension. These methods are based on projecting a given PDE onto a low-rank tensor manifold and then constructing an approximate PDE solution as a path on the manifold. In order to control the accuracy of the low-rank tensor approximation we present a rank-adaptive algorithm that can add or remove tensor modes adaptively from the PDE solution during time integration. We also present a tensor rank reduction method based on coordinate transformations that can greatly increase the efficiency of high-dimensional tensor approximation algorithms. The idea is to determine a coordinate transformation of a given functions domain so that the function in the new coordinate system has smaller tensor rank. We demonstrate each of the presented low-rank tensor methods by providing several numerical applications to multivariate functions and PDEs.

To my parents:

Mark and Angela

Acknowledgments

First I need to acknowledge my adviser Professor Daniele Venturi for everything he has taught me, for introducing me to this beautiful subject, and for inspiring me throughout the research process. Without his hard work, patience, and guidance this dissertation could not have been written. I need to acknowledge all faculty members of the mathematics and applied mathematics departments at UC Santa Cruz for everything they have taught me and for providing a welcoming environment to learn and grow. I also need to acknowledge Bram for his friendship and collaboration throughout our graduate studies.

I need to acknowledge the unconditional love and support I received from my friends and family. In particular my parents Mark and Angela who have always supported each and every one of my endeavors, my sister Rachael who inspires me in so many ways, and my brother Isaac who I am so lucky to be able to pass time with. I also need to acknowledge all the love and support I have received from my significant other Corrina during my graduate studies. Finally I would like to acknowledge all of the wonderful friendships I have found in the town of Santa Cruz during my time as an undergraduate and graduate student.

Chapter 1

Introduction

High-dimensional partial differential equations (PDEs) arise in engineering, physical sciences and mathematics as elegant formulations of many important phenomena. Classical examples are equations involving probability density functions such as the Fokker-Plank equation [91], the Liouville equation [112, 23], or the Boltzmann equation [17, 33]. Other examples of high-dimensional PDEs can be obtained as finite-dimensional approximations of functional differential equation [109, 111], such as the Hopf equation of turbulence [48, 49, 76] the Schwinger-Dyson equation [50], or functional formulations of classical statistical dynamics [74, 51, 52, 84]. Computing the solution to high-dimensional PDEs is a challenging problem that requires approximating high-dimensional functions, i.e., the solution to the PDE, and then developing appropriate numerical schemes to compute such functions accurately for a given PDE. Classical numerical methods based on tensor product representations are not viable in high-dimensions, as the number of degrees of freedom grows exponentially fast with the dimension. To address this problem there have been substantial research efforts in recent years on high-dimensional numerical approximation theory. Techniques such as sparse collocation [15, 22, 9, 37, 79], high-dimensional model representations (HDMR) [67, 16, 8] and, more recently, deep neural networks [86, 87, 114] and tensor methods [57, 7, 1, 43, 20, 64] were proposed to mitigate the exponential growth of the degrees of freedom, the computational cost and memory requirements.

In this dissertation we present methods based on tensor networks for solving high-dimensional PDEs. A tensor network is a factorization of an entangled object such as a multivariate function or an operator, into a set of simpler objects (e.g., low-dimensional functions or operators) which are amenable to efficient representation and computation. The process of building a tensor network relies on a hierarchical decomposition of the entangled object, which, can be visualized in terms of trees [105, 34, 7]. Such a decomposition is rooted in the spectral theory for linear operators [56], and it opens the possibility to approximate high-dimensional functions and compute the solution of high-dimensional PDEs at a cost that scales linearly with respect to the dimension of the object and polynomially with respect to the tensor rank. A key observation is that the collection of all tensors with a fixed rank possesses a smooth manifold structure [105]. This observation gives simple geometric meaning to many operations involving tensors and can be exploited in the development of numerical algorithms. Using this geometric structure we develop numerical integration schemes for computing solutions to high-dimensional nonlinear initial/boundary value problems of the form

$$\frac{\partial u(\boldsymbol{x},t)}{\partial t} = G(\boldsymbol{x}, u(\boldsymbol{x},t)), \qquad u(\boldsymbol{x},0) = u_0(\boldsymbol{x}). \tag{1.1}$$

Here $u : \Omega \times [0,T] \to \mathbb{R}$ is a d-dimensional (time-dependent) scalar field defined on some domain $\Omega \subseteq \mathbb{R}^d$ and G is a nonlinear operator which may depend on the spatial variables, and may incorporate boundary conditions. The storage and computational cost of these numerical integration schemes scales linearly in the problem dimension d and polynomially in the tensor rank, rendering them viable for high-dimensional problems.

This dissertation is organized as follows. In Chapter 2 we develop an approximation theory for high-dimensional multivariate functions based on recursive application of spectral decompositions. We focus specifically on the functional tensor train (FTT) format which provides a simple and effective ansatz for demonstrating low-rank tensor methods. We discuss the geometric structure of FTT tensors which lays the theoretical foundation used in the subsequent chapter to develop numerical integration schemes for PDEs. In Chapter 3 we develop numerical integration schemes for initial/boundary value problems of the form (1.1). We focus on two methods: dynamic approximation and step-truncation, and prove that these two methodologies are consistent with each other as the temporal step size is sent to zero. We also develop a rank-adaptive criterion for tensor integration that allows us to efficiently control the error of the low-rank approximation to the PDE solution. Numerical applications to various linear and nonlinear PDEs are presented. In Chapter 4 we discuss rank reducing coordinate transformations, i.e., coordinate systems which allow us to represent multivariate functions with smaller rank than the representation of the same function in Cartesian coordinates. In the case of linear coordinate transformations we show that this idea gives rise to a class of functions called tensor ridge functions. Then we develop an algorithm based on Riemannian gradient descent on matrix manifolds to compute linear rank reducing coordinate transformations. Finally we demonstrate the effectiveness of rank reducing coordinate transformations on a prototype function approximation problem and various linear and nonlinear PDEs.

Chapter 2

Low-rank tensor approximation of high-dimensional functions

We begin by introducing a mathematical setting that yields collections of functions amenable to low-rank tensor representations. Subsequently we present low-rank tensor approximation of multivariate functions, an effective ansatz for the numerical representation of high-dimensional functions in separable Hilbert spaces.

2.1 Separable Hilbert space

Let $\Omega \subseteq \mathbb{R}^d$ be a Cartesian product of d real intervals $\Omega_i = [a_i, b_i]$

$$\Omega = \bigtimes_{i=1}^{d} \Omega_i, \tag{2.1}$$

μ a finite product measure on Ω

$$\mu(\boldsymbol{x}) = \prod_{i=1}^{d} \mu_i(x_i), \tag{2.2}$$

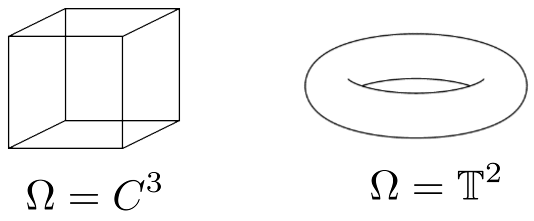


Figure 2.1: Geometrical interpretation of the domain Ω as a hyper-cube or torus.

and

$$H(\Omega) = L^2_\mu(\Omega) \tag{2.3}$$

the standard weighted Hilbert space[1] of square–integrable functions on Ω. Geometrically the domain Ω can be visualized as a hyper-cube or a high-dimensional torus (see Figure 2.1). It is convenient to define the following partial Cartesian products

$$\begin{aligned} \Omega_{\leq i} &= \Omega_1 \times \cdots \times \Omega_i, && i = 1, \ldots, d, \\ \Omega_{>i} &= \Omega_{i+1} \times \cdots \times \Omega_d, && i = 1, \ldots, d-1, \end{aligned} \tag{2.4}$$

the corresponding partial products of measures

$$\begin{aligned} \mu_{\leq i} &= \mu_1 \times \cdots \times \mu_i, && i = 1, \ldots, d, \\ \mu_{>i} &= \mu_{i+1} \times \cdots \times \mu_d, && i = 1, \ldots, d-1, \end{aligned} \tag{2.5}$$

and partial vectors

$$\begin{aligned} \boldsymbol{x}_{\leq i} &= \begin{bmatrix} x_1 & \cdots & x_i \end{bmatrix}^\top && i = 1, \ldots, d, \\ \boldsymbol{x}_{>i} &= \begin{bmatrix} x_{i+1} & \cdots & x_d \end{bmatrix}^\top && i = 1, \ldots, d. \end{aligned} \tag{2.6}$$

[1]Note that the Hilbert space H in equation (2.3) can be equivalently chosen to be a Sobolev space $W^{2,p}$ (see [29] for details).

The goal of low-rank tensors is to approximate high-dimensional $(d >> 2)$ functions $u(\boldsymbol{x}) \in H(\Omega)$ at a reasonable computational cost. It is well-known that the Hilbert space $H(\Omega)$ can be written as a tensor product of Hilbert spaces containing functions of one variable

$$H(\Omega) = H(\Omega_1) \otimes H(\Omega_2) \otimes \cdots \otimes H(\Omega_d). \tag{2.7}$$

Given bases $\{\psi_i(x_i; \alpha_i)\}_{\alpha_i=1}^{\infty}$ for each $H(\Omega_i)$ the tensor product basis $\{\psi_1(x_1; \alpha_1) \otimes \psi_2(x_2; \alpha_2) \otimes \cdots \otimes \psi_d(x_d; \alpha_d)\}_{\alpha_1,\ldots,\alpha_d=1}^{\infty}$ can be constructed for $H(\Omega)$. Expressing a function $u(\boldsymbol{x}) \in H(\Omega)$ relative to the tensor product basis results in a series expansion of the form

$$u(\boldsymbol{x}) = \sum_{\alpha_1,\ldots,\alpha_d=1}^{\infty} A_{\alpha_1,\ldots,\alpha_d} \psi_1(x_1; \alpha_1) \otimes \psi_2(x_2; \alpha_2) \otimes \cdots \otimes \psi_d(x_d; \alpha_d). \tag{2.8}$$

Truncating each summation appearing in infinite series (2.8) to a finite number of terms, r, we obtain the approximation

$$u(\boldsymbol{x}) \approx \sum_{\alpha_1,\ldots,\alpha_d=1}^{r} A_{\alpha_1,\ldots,\alpha_d} \psi_1(x_1; \alpha_1) \otimes \psi_2(x_2; \alpha_2) \otimes \cdots \otimes \psi_d(x_d; \alpha_d). \tag{2.9}$$

Observe that the number of coefficients (or degrees of freedom) $A_{\alpha_1,\ldots,\alpha_d}$ in the preceding approximation of $u(\boldsymbol{x})$ is r^d. Therefore the storage cost of approximations (2.9) constructed from a tensor product basis grows exponentially as the number of dimensions d increases. To illustrate the computational limitations of representing high-dimensional functions relative to a tensor product basis, consider $d = 8$ and $r = 30$ in (2.9). The number of coefficients required to store this approximation is $r^d = 6.56 \times 10^{11}$, which, if stored in double precision floating point format (IEEE 754, 64 bits/number), requires around 5 terabytes of memory.

Low-rank tensor formats are an alternative to the tensor product basis representation of high-dimensional functions which do not suffer from exponential scaling of computational cost with the the number of dimensions d. Such tensor formats are obtained by recursively applying spectral decompositions (also referred to as Schmidt decompositions, bi-orthogonal decompositions, or Karhunen-Loeve expansions depending on the application) to the function $u(\boldsymbol{x})$. One instance of such a spectral decomposition is realized by selecting a partition of the relevant variables, e.g.,

$$\{x_1, x_2, \dots, x_d\} = \{x_1, x_2, \dots, x_j\} \cup \{x_{j+1}, \dots, x_d\}, \qquad 1 \leq j \leq d-1, \tag{2.10}$$

and considering the linear operator

$$\begin{aligned} U : H(\Omega_{\leq j}) &\to H(\Omega_{>j}) \\ g(\boldsymbol{x}_{\leq j}) &\mapsto \int_{\Omega_{\leq j}} u(\boldsymbol{x}) g(\boldsymbol{x}_{\leq j}) d\mu_{\leq j}(\boldsymbol{x}_{\leq j}) \end{aligned} \tag{2.11}$$

with formal adjoint

$$\begin{aligned} U^* : H(\Omega_{>j}) &\to H(\Omega_{\leq j}) \\ h(\boldsymbol{x}_{>j}) &\mapsto \int_{\Omega_{>j}} u(\boldsymbol{x}) h(\boldsymbol{x}_{>j}) d\mu_{>j}(\boldsymbol{x}_{>j}). \end{aligned} \tag{2.12}$$

The operators U, U^* are linear, bounded, and compact since u is a Hilbert-Schmidt kernel. The composition operators

$$\begin{aligned} U^*U : H(\Omega_{\leq i}) &\to H(\Omega_{\leq i}), \\ UU^* : H(\Omega_{>i}) &\to H(\Omega_{>i}), \end{aligned} \tag{2.13}$$

are self-adjoint compact operators that share the same spectra $\{\sigma(\alpha)\}_{\alpha\in\mathbb{N}}$ which is countable with one accumulation point at 0 and satisfies

$$\sum_{\alpha\in\mathbb{N}} \sigma(\alpha) < \infty. \tag{2.14}$$

The eigenfunctions $\{\psi_{\leq i}(\boldsymbol{x}_{\leq i};\alpha)\}_{\alpha\in\mathbb{N}}$ (resp. $\{\psi_{>i}(\boldsymbol{x}_{>i};\alpha)\}_{\alpha\in\mathbb{N}}$) of U^*U (resp. UU^*) form an orthonormal basis for $H(\Omega_{\leq i})$ (resp. $H(\Omega_{>i})$). It is a classical result of functional analysis[2] that $u(\boldsymbol{x})$ admits an expansion relative to the orthonormal bases of eigenfunctions []

$$u(\boldsymbol{x}) = \sum_{\alpha=1}^{\infty} \sqrt{\sigma(\alpha)}\psi_{\leq i}(\boldsymbol{x}_{\leq i};\alpha)\psi_{>i}(\boldsymbol{x}_{>i};\alpha). \tag{2.15}$$

2.2 Tensor formats

Tensor formats are obtained by sequentially applying the spectral decomposition (2.15) to decompose a function $u(\boldsymbol{x}) \in H(\Omega)$ into a series expansion consisting of functions depending on a fewer number of variables than $u(\boldsymbol{x})$. The selection of variable partitions at each step of the spectral decomposition sequence defines the structure of the series expansion for $u(\boldsymbol{x})$ and is called a tensor format. To demonstrate the idea of applying sequences of Schmidt decompositions, let us consider a three dimensional function $u(x_1, x_2, x_3)$. First we partition the variables $\{x_1, x_2, x_3\}$ as $\{x_1\} \cup \{x_2, x_3\}$ and use the spectral decomposition to write

$$u(x_1, x_2, x_3) = \sum_{\alpha_1=1}^{\infty} \sqrt{\sigma_1(\alpha_1)}\psi_1(x_1;\alpha_1)\psi_{23}(\alpha_1; x_2, x_3). \tag{2.16}$$

Since $\psi_{23}(x_2, x_3; \alpha_1)$ belongs to the Hilbert space $L^2_{\mu_{\leq 2}}(\Omega_{\leq 2})$ for each α_1, we can partition the variables $\{x_2, x_3\}$ as $\{x_2\} \cup \{x_3\}$ and apply the spectral decomposition to ψ_{23} for each α_1

$$\psi_{23}(x_2, x_3; \alpha_1) = \sum_{\alpha_2=1}^{\infty} \sqrt{\sigma_2(\alpha_2)}\varphi_2(x_2;\alpha_1,\alpha_2)\varphi_3(x_3;\alpha_1,\alpha_2), \qquad \alpha_1 = 1, 2, \ldots. \tag{2.17}$$

[2]This spectral decomposition is a direct generalization of the singular value decomposition for matrices to separable Hilbert spaces. Indeed, a $m \times n$ real matrix $\boldsymbol{A}$ is a linear map from $\mathbb{R}^n \to \mathbb{R}^m$, its adjoint, $\boldsymbol{A}^*$, is a linear map $\mathbb{R}^m \to \mathbb{R}^n$. The singular vectors/values of $\boldsymbol{A}$ are the eigenvectors/eigenvalues of the composition maps $\boldsymbol{A}^*\boldsymbol{A} : \mathbb{R}^n \to \mathbb{R}^n$ and $\boldsymbol{A}\boldsymbol{A}^* : \mathbb{R}^m \to \mathbb{R}^m$.

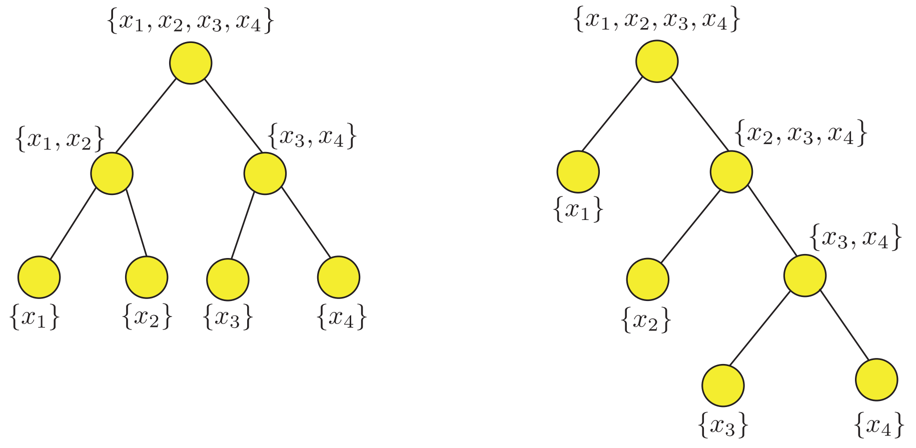


Figure 2.2: Two possible tensor formats for a four dimensional function. A balanced binary tree (left) and a unbalanced or tensor train binary tree (right).

Now we can represent $u(\boldsymbol{x})$ relative to the functions $\psi_1, \varphi_2, \varphi_3$ which depend only on 1 continuous variable

$$u(\boldsymbol{x}) = \sum_{\alpha_1,\alpha_2=1}^{\infty} \sqrt{\sigma_1(\alpha_1)\sigma_2(\alpha_2)}\psi_1(x_1;\alpha_1)\varphi_2(x_2;\alpha_1,\alpha_2)\varphi_3(x_3;\alpha_1,\alpha_2). \tag{2.18}$$

Comparing this expansion with the expansion in terms of tensor product basis functions ((2.8) with $d = 3$) we notice that both expansions are given in terms of univariate functions, however, (2.18) uses only two summations while (2.8) uses three summations. The construction of the expansion (2.18) required two spectral decompositions in sequence. Of course other variable partitions for each of these two spectral decomposition are possible, e.g., first using the partition $\{x_1, x_2, x_3\} = \{x_1, x_2\} \cup \{x_3\}$ and then using the partition $\{x_1, x_2\} = \{x_1\} \cup \{x_2\}$. Yet another variant of the sequential application of spectral decompositions can be realized by reconsidering the function $\psi_{23}(x_2, x_3; \alpha_1)$ as an element of $L^2_{\mu_{\leq 2}\times\tau}(\Omega_{\leq 2} \times \mathbb{N})$ where τ is the counting measure on $\mathbb{N}$. Applying the spectral decomposition with variable partition

$\{x_2, x_3, \alpha_1\} = \{x_2, \alpha_1\} \cup \{x_3\}$ yields the expansion

$$\psi_{23}(x_2, x_3; \alpha_1) = \sum_{\alpha_2=1}^{\infty} \sqrt{\sigma_2(\alpha_2)} \psi_2(x_2; \alpha_1, \alpha_2) \psi_3(\alpha_2; x_3). \tag{2.19}$$

Here ψ_3 does not depend on α_1 whereas φ_3 from (2.17) depends on α_1. Using (2.19) together with (2.16) we obtain the expansion of $u(\boldsymbol{x})$

$$u(\boldsymbol{x}) = \sum_{\alpha_1, \alpha_2=1}^{\infty} \sqrt{\sigma_1(\alpha_1)\sigma_2(\alpha_2)} \psi_1(x_1; \alpha_1) \varphi_1(x_2; \alpha_1, \alpha_2) \varphi_3(x_3; \alpha_2), \tag{2.20}$$

known as the three-dimensional functional tensor train (FTT) format. For a d-dimensional function $u(\boldsymbol{x})$ there are many possible ways to partition the variables in the sequence of spectral decompositions. The choice of partition at each step in the sequence (and therefore the tensor format) can be conveniently visualized using binary trees. In Figure 2.2 two possible binary trees are shown for a four dimensional function: a balanced tree (left) and an unbalanced or tensor train tree (right). The FTT format is described in detail for d-dimensional functions in section 2.3.

The spectral decomposition binary trees are a useful (constructive) graphical representation of tensor formats. Another less constructive (but useful) graphical representation of tensor formats is tensor network diagrams. In a tensor network diagram functions (called tensors) are represented by vertices. For a given vertex, each edge connected to that vertex denotes dependence on a variable (either continuous or discrete). Summations of products of tensors over a shared index (called tensor contractions) are represented by edges connecting vertices. Consequently, the number of free edges in the tensor network diagram indicates the dimension of the function the network represents. For example the tensor network diagram corresponding to the series expansion (2.20) is shown in Figure 2.3.

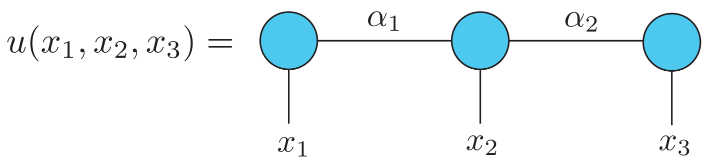


Figure 2.3: Tensor network diagram corresponding to the three-dimensional FTT format defined by the series expansion (2.20). The left vertex corresponds to $\psi_1(x_1; \alpha_1)$, the middle vertex corresponds to $\psi_2(x_2; \alpha_1, \alpha_2)$, and the right vertex corresponds to $\psi_3(x_3; \alpha_2)$.

2.3 Functional tensor train (FTT) format

We proceed by giving a detailed presentation of the functional tensor train (FTT) format, a recursive spectral decomposition for d-variate functions $u(\boldsymbol{x}) \in H(\Omega)$ based on the unbalanced binary tree Figure 2.2 (right). Define the operator

$$\begin{aligned} U_1 : L^2_{\mu_1}(\Omega_1) &\to L^2_{\mu_{>1}}(\Omega_{>1}) \\ g_1 &\mapsto \int_{\Omega_1} u(\boldsymbol{x}) g_1(x_1) d\mu_1(x_1). \end{aligned} \tag{2.21}$$

Such operator is linear, bounded, and compact since u is a Hilbert-Schmidt kernel. The formal adjoint operator of U_1 is given by

$$\begin{aligned} U_1^* : L^2_{\mu_{>1}}(\Omega_{>1}) &\to L^2_{\mu_1}(\Omega_1) \\ h_1 &\mapsto \int_{\Omega_{>1}} u(\boldsymbol{x}) h_1(x_2, \ldots, x_d) d\mu_{>1}(x_2, \ldots, x_d). \end{aligned} \tag{2.22}$$

The composition operator $U_1^* U_1 : L^2_{\mu_1}(\Omega_1) \to L^2_{\mu_1}(\Omega_1)$ is a self-adjoint compact Hermitian operator. The spectrum of $U_1^* U_1$, denoted as $\sigma(U_1^* U_1) = \{\lambda_1(1), \lambda_1(2), \ldots\}$, is countable with one accumulation point at 0 and satisfies

$$\sum_{\alpha_1=1}^{\infty} \lambda_1(\alpha_1) < \infty. \tag{2.23}$$

We denote by $\psi_1(\alpha_1; x_1) \in L^2_{\mu_1}(\Omega_1)$ a normalized eigenfunction of $U_1^* U_1$ corresponding to the eigenvalue $\lambda_1(\alpha_1)$, and construct an orthonormal basis of eigenfunctions $\{\psi_1(\alpha_1; x_1)\}_{\alpha_1=1}^{\infty}$ for the space $L^2_{\mu_1}(\Omega_1)$. The operator $U_1 U_1^* : L^2_{\mu_{>1}}(\Omega_{>1}) \to L^2_{\mu_{>1}}(\Omega_{>1})$ is also self-adjoint, compact, and Hermitian, and shares the same spectrum as $U_1 U_1^*$, i.e., $\sigma(U_1 U_1^*) = \sigma(U_1^* U_1)$. Its eigenfunctions[3] $\{\psi_{>1}(\alpha_1; x_2, \ldots, x_d)\}_{\alpha_1=1}^{\infty}$ form an orthonormal basis of $L^2_{\mu_{>1}}(\Omega_{>1})$. It is a classical result in functional analysis that $u(\boldsymbol{x})$ can be expanded as (see [42, 4, 5])

$$u(\boldsymbol{x}) = \sum_{\alpha_1=1}^{\infty} \sqrt{\lambda_1(\alpha_1)}\psi_1(\alpha_1; x_1)\psi_{>1}(\alpha_1; x_2, \ldots, x_d). \tag{2.24}$$

Next we consider $\psi_{>1}(\alpha_1; x_2, \ldots, x_d) \in L^2_{\tau\times\mu_{>1}}(\mathbb{N} \times \Omega_{>1})$ where τ denotes the counting measure on $\mathbb{N}$. From the orthonormality of $\{\psi_{>1}(\alpha_1; x_2, \ldots, x_d)\}_{\alpha_1=1}^{\infty}$ with respect to the inner product in $L^2_{\mu_{>1}}(\Omega_{>1})$ and the fact that $u \in L^2_{\mu}(\Omega)$ we have

$$\begin{aligned}
&\int_{\mathbb{N}\times\Omega_{>1}} |\sqrt{\lambda_1(\alpha_1)}\psi_{>1}(\alpha_1; x_2, \ldots, x_d)|^2 d\tau(\alpha_1) d\mu_{>1}(x_2, \ldots, x_d)\\
&= \sum_{\alpha_1=1}^{\infty} \lambda_1(\alpha_1) \int_{\Omega_{>1}} |\psi_{>1}(\alpha_1; x_2, \ldots, x_d)|^2 d\mu_{>1}(x_2, \ldots, x_d)\\
&= \sum_{\alpha_1=1}^{\infty} \lambda_1(\alpha_1) < \infty,
\end{aligned} \tag{2.25}$$

i.e., $(\sqrt{\lambda_1}\psi_{>1}) \in L^2_{\tau\times\mu_{>1}}(\mathbb{N} \times \Omega_{>1})$ is a Hilbert-Schmidt kernel. Thus the operators

$$\begin{aligned}
U_2 : L^2_{\tau\times\mu_2}(\mathbb{N} \times \Omega_2) &\to L^2_{\mu_{>2}}(\Omega_{>2})\\
g_2 &\mapsto \int_{\mathbb{N}\times\Omega_2} \sqrt{\lambda_1(\alpha_1)}\psi_{>1}(\alpha_1; x_2, \ldots, x_d) g_2(\alpha_1; x_2) d\tau(\alpha_1) d\mu_2(x_2)
\end{aligned} \tag{2.26}$$

[3]We will use the notation $\psi_{>i}$ to denote functions that depend on variables $(x_{i+1}, \ldots, x_d)$. Similarly, $\psi_{\leq i}$ denotes functions depending on variables $(x_1, \ldots, x_i)$ and ψ_i denotes functions depending on x_i.

and

$$\begin{aligned} U_2^* : L^2_{\mu_{>2}}(\Omega_{>2}) &\to L^2_{\tau\times\mu_2}(\mathbb{N}\times\Omega_2) \\ h_2 &\mapsto \int_{\Omega_{>2}} \sqrt{\lambda_1(\alpha_1)}\psi_{>1}(\alpha_1;x_2,\dots,x_d)h_2(\alpha_1;x_2)d\mu_{>2}(x_2,\dots,x_d) \end{aligned} \tag{2.27}$$

enjoy the same properties as U_1 and U_1^*, in particular they are linear, bounded and compact. Hence the composition operators $U_2^*U_2$ and $U_2U_2^*$ are self-adjoint compact Hermitian operators which share a countable spectrum $\sigma(U_2^*U_2) = \sigma(U_2U_2^*) = \{\lambda_2(\alpha_2)\}_{\alpha_2=1}^{\infty}$ that accumulates at zero and satisfies

$$\sum_{\alpha_2=1}^{\infty} \lambda_2(\alpha_2) < \infty. \tag{2.28}$$

We decompose $\sqrt{\lambda_1(\alpha_1)}\psi_{>1}(\alpha_1;x_2,\dots,x_d)$ as

$$\sqrt{\lambda_1(\alpha_1)}\psi_{>1}(\alpha_1;x_2,\dots,x_d) = \sum_{\alpha_2=1}^{\infty} \sqrt{\lambda_2(\alpha_2)}\psi_2(\alpha_1;x_2;\alpha_2)\psi_{>2}(\alpha_2;x_3,\dots,x_d), \tag{2.29}$$

where $\{\psi_2(\alpha_1;x_2;\alpha_2)\}_{\alpha_2=1}^{\infty}$ and $\{\psi_{>2}(\alpha_2;x_3,\dots,x_d)\}_{\alpha_2=1}^{\infty}$ are orthonormal bases of eigenfunctions corresponding to the operators $U_2^*U_2$ and $U_2U_2^*$, respectively. A substitution of (2.29) into (2.24) yields

$$u(\boldsymbol{x}) = \sum_{\alpha_1=1}^{\infty}\sum_{\alpha_2=1}^{\infty} \sqrt{\lambda_2}(\alpha_2)\psi_1(x_1;\alpha_1)\psi_2(\alpha_1;x_2;\alpha_2)\psi_{>2}(\alpha_2;x_3,\dots,x_d). \tag{2.30}$$

Proceeding recursively in this manner yields the following FTT expansion

$$u(\boldsymbol{x}) = \sum_{\alpha_1,\dots,\alpha_{d-1}=1}^{\infty} \psi_1(\alpha_0;x_1;\alpha_1)\psi_2(\alpha_1;x_2;\alpha_2)\cdots\psi_d(\alpha_{d-1};x_d;\alpha_d), \tag{2.31}$$

where $\alpha_0 = \alpha_d = 1$.

By truncating the expansion (2.31) so that the largest singular values are retained we obtain

$$u_{\boldsymbol{r}}(\boldsymbol{x}) = \sum_{\alpha_0,\dots,\alpha_d=1}^{\boldsymbol{r}} \psi_1(\alpha_0;x_1;\alpha_1)\psi_2(\alpha_1;x_2;\alpha_2)\cdots\psi_d(\alpha_{d-1};x_d;\alpha_d), \tag{2.32}$$

where $\boldsymbol{r} = (1, r_1, \ldots, r_{d-1}, 1)$ is the FTT-rank (or rank if the FTT format is clear from context). It is known that the truncated FTT expansion converges optimally with respect to the $L^2_\mu(\Omega)$ norm [13]. More precisely, for any given function $u \in L^2_\mu(\Omega)$ the FTT approximant (2.32) minimizes the residual $R_{\boldsymbol{r}} = \|u - u_{\boldsymbol{r}}\|_{L^2_\mu(\Omega)}$ relative to independent variations of the functions $\psi_i(\alpha_{i-1}; x_i; \alpha_i)$ on a tensor manifold with constant rank $\boldsymbol{r}$. It is convenient to write (2.32) in a more compact form as

$$u_{\boldsymbol{r}}(\boldsymbol{x}) = \boldsymbol{\Psi}_1(x_1)\boldsymbol{\Psi}_2(x_2)\cdots\boldsymbol{\Psi}_d(x_d), \tag{2.33}$$

where $\boldsymbol{\Psi}_i(x_i)$ is a $r_{i-1} \times r_i$ matrix with entries $[\boldsymbol{\Psi}_i(x_i)]_{jk} = \psi_i(j; x_i; k)$. The matrix-valued functions $\boldsymbol{\Psi}_i(x_i)$ will be referred to as FTT *cores*, and we denote by $M_{r_{i-1}\times r_i}(L^2_{\mu_i}(\Omega_i))$ the set of all $r_{i-1} \times r_i$ matrices with entries in $L^2_{\mu_i}(\Omega_i)$. To simplify notation, we will often suppress explicit tensor core dependence on the spatial variable x_i, allowing us to simply write $\boldsymbol{\Psi}_i = \boldsymbol{\Psi}_i(x_i)$ and $\psi_i(\alpha_{i-1}, \alpha_i) = \psi_i(\alpha_{i-1}; x_i; \alpha_i)$ as the spatial dependence is indicated by the tensor core subscript. Throughout this dissertation we also denote the FTT tensor $u_{\boldsymbol{r}}$ as u_{TT} whenever it is cumbersome to keep track of the rank $\boldsymbol{r}$ of the FTT tensor. Specifically the notation u_{TT} is useful for the rank-adaptive algorithms presented in Chapter 3. We may compute rank-$\boldsymbol{r}$ FTT decompositions at quadrature points by first discretizing $u(\boldsymbol{x})$ on a tensor product grid and then using a tensor product quadrature rule together with known algorithms for computing a discrete TT decomposition of a full tensor as discussed in [13]. This procedure for constructing low-rank FTT approximations requires evaluating $u(\boldsymbol{x})$ on a full tensor product grid in d-dimensions which is not feasible even for moderately large d. Alternative methods for constructing low-rank FTT approximations include the TT-cross algorithm [81] and the more recently developed TT-sketching algorithms [113].

2.3.1 Orthogonalization and truncation of FTT tensors

Many arithmetic operations on FTT tensors results in a significant increase of FTT rank. For example The sum of two FTT tensors $u_{\boldsymbol{r}}$ and $v_{\boldsymbol{s}}$ with ranks $\boldsymbol{r}$ and $\boldsymbol{s}$ respectively results in a new FTT tensor $u_{\boldsymbol{r}} + v_{\boldsymbol{s}}$ with (non-optimal) FTT rank $\boldsymbol{r} + \boldsymbol{s}$ that can often be compressed to a smaller FTT rank. Another example is the multiplication of the FTT tensors $u_{\boldsymbol{r}}$ and $v_{\boldsymbol{s}}$ which results in a FTT tensor $u_{\boldsymbol{r}} \cdot v_{\boldsymbol{s}}$ with FTT rank $\boldsymbol{r} \circ \boldsymbol{s}$ where $\circ$ denotes element-wise multiplication. Thus a key algorithm for working with FTT tensors is truncation which is based on truncating singular value spectra in the FTT tensor. The spectra of an FTT tensor can be effectively computed by performing orthogonalizations of FTT tensors based on recursive applications of matrix decompositions such as the QR factorization. Hereafter we describe in detail the orthogonalization and truncation of FTT tensors.

For any tensor core $\boldsymbol{\Psi}_i \in M_{r_{i-1}\times r_i}(L^2_{\mu_i}(\Omega_i))$ we define the matrix

$$\langle \boldsymbol{\Psi}_i^\top \boldsymbol{\Psi}_i \rangle_i \in M_{r_i \times r_i}(\mathbb{R}) \tag{2.34}$$

with entries[4]

$$\left\langle \boldsymbol{\Psi}_i^\top \boldsymbol{\Psi}_i \right\rangle_i (j,k) = \sum_{p=1}^{r_{i-1}} \int_{\Omega_i} \psi_i(p; x_i; j)\psi_i(p; x_i; k) d\mu_i(x_i). \tag{2.35}$$

The FTT representation (2.32) is given in terms of FTT cores $\boldsymbol{\Psi}_i$ satisfying[5]

$$\begin{aligned} \left\langle \boldsymbol{\Psi}_i^\top \boldsymbol{\Psi}_i \right\rangle_i &= \boldsymbol{I}_{r_i \times r_i}, \qquad i = 1, \dots, d-1, \\ \left\langle \boldsymbol{\Psi}_d \boldsymbol{\Psi}_d^\top \right\rangle_d &= \sqrt{\boldsymbol{\Lambda}}, \end{aligned} \tag{2.36}$$

[4]The averaging operation in (2.35) can be viewed as a an inner product on the space $\bigtimes_{n=1}^{r_{i-1}} L^2_{\mu_i}(\Omega_i)$.

[5]Equation (2.36) follows immediately from the orthogonality of $\{\psi_i(\alpha_{i-1}; x_i; \alpha_i)\}_{\alpha_i}$ relative to the inner product in $L^2_{\tau\times\mu_i}(\mathbb{N} \times \Omega_i)$.

where $\boldsymbol{\Lambda}$ is a diagonal matrix with entries $\lambda(\alpha_{d-1})$ ($\alpha_{d-1} = 1, \ldots, r_{d-1}$). Other orthogonal representations can be computed, e.g., based on recursive QR decompositions. To describe different orthogonalizations of FTT tensors, let $\boldsymbol{\Psi}_i \in M_{r_{i-1}\times r_i}(L^2_{\mu_i}(\Omega_i))$ and consider each column of $\boldsymbol{\Psi}_i$ as a vector in $\bigtimes_{n=1}^{r_{i-1}} L^2_{\mu_i}(\Omega_i)$. Performing an orthogonalization process (e.g. Gram-Schmidt) on the columns of the FTT core $\boldsymbol{\Psi}_i$ relative to the inner product (2.55) yields a QR-type decomposition of the form

$$\boldsymbol{\Psi}_i = \boldsymbol{Q}_i \boldsymbol{R}_i, \tag{2.37}$$

where $\boldsymbol{Q}_i$ is an $r_{i-1} \times r_i$ matrix with elements in $L^2_{\mu_i}(\Omega_i)$ satisfying $\left\langle \boldsymbol{Q}_i^\top \boldsymbol{Q}_i \right\rangle_i = \boldsymbol{I}_{r_i \times r_i}$, and $\boldsymbol{R}_i$ is an upper triangular $r_i \times r_i$ matrix with real entries. Next consider an arbitrary FTT tensor $u_{\boldsymbol{r}} = \boldsymbol{\Psi}_1 \boldsymbol{\Psi}_2 \cdots \boldsymbol{\Psi}_d$, where the matrix $\langle \boldsymbol{\Psi}_i^\top \boldsymbol{\Psi}_i \rangle_i$ may be singular. One way to orthogonalize $u_{\boldsymbol{r}}$ is by performing QR decompositions recursively from left to right as we will now describe. Begin by decomposing $\boldsymbol{\Psi}_1$ as

$$\begin{gathered}\boldsymbol{\Psi}_1 = \boldsymbol{Q}_1 \boldsymbol{R}_1, \qquad \boldsymbol{Q}_1 \in M_{r_0 \times r_1}(L^2_{\mu_1}(\Omega_1)), \quad \left\langle \boldsymbol{Q}_1^\top \boldsymbol{Q}_1 \right\rangle_1 = \boldsymbol{I}_{r_1 \times r_1}, \\ \boldsymbol{R}_1 \in M_{r_1 \times r_1}(\mathbb{R}) \text{ is upper triangular.}\end{gathered} \tag{2.38}$$

Now we may write $u_{\boldsymbol{r}} = \boldsymbol{Q}_1 \boldsymbol{R}_1 \boldsymbol{\Psi}_2 \cdots \boldsymbol{\Psi}_d$. Next, perform another QR decomposition

$$\begin{gathered}\boldsymbol{R}_1 \boldsymbol{\Psi}_2 = \boldsymbol{Q}_2 \boldsymbol{R}_2, \qquad \boldsymbol{Q}_2 \in M_{r_1 \times r_2}(L^2_{\mu_2}(\Omega_2)), \quad \left\langle \boldsymbol{Q}_2^\top \boldsymbol{Q}_2 \right\rangle_2 = \boldsymbol{I}_{r_2 \times r_2}, \\ \boldsymbol{R}_2 \in M_{r_2 \times r_2}(\mathbb{R}) \text{ is upper triangular.}\end{gathered} \tag{2.39}$$

Proceeding recursively in this way we obtain a representation for $u_{\boldsymbol{r}}$ of the form

$$u_{\boldsymbol{r}} = \boldsymbol{Q}_1 \cdots \boldsymbol{Q}_{d-1} \boldsymbol{Q}_d \boldsymbol{R}_d, \tag{2.40}$$

where each $\boldsymbol{Q}_i \in M_{r_{i-1}\times r_i}(L^2_{\mu_i}(\Omega_i))$ satisfies $\left\langle \boldsymbol{Q}_i^\top \boldsymbol{Q}_i \right\rangle_i = \boldsymbol{I}_{r_i \times r_i}$. We refer to such a representation as a left orthogonalization of $u_{\boldsymbol{r}}$. We may stop orthogonolizing at any step in the recursive process to obtain the partial left orthogonalization

$$u_{\boldsymbol{r}} = \boldsymbol{Q}_{\leq i}\boldsymbol{R}_i\boldsymbol{\Psi}_{>i}. \tag{2.41}$$

Similar to orthogonalizing from the left, we may also orthogonalize $u_{\boldsymbol{r}}$ from the right. To do so, begin by performing a QR decomposition

$$\begin{gathered}\boldsymbol{\Psi}_d^\top = \boldsymbol{K}_d\boldsymbol{W}_d, \qquad \boldsymbol{K}_d \in M_{r_d\times r_{d-1}}(L^2_{\mu_d}(\Omega_d)), \quad \left\langle \boldsymbol{K}_d^\top \boldsymbol{K}_d\right\rangle_d = \boldsymbol{I}_{r_{d-1}\times r_{d-1}}, \\ \boldsymbol{W}_d \in M_{r_{d-1}\times r_{d-1}}(\mathbb{R}) \text{ is upper triangular.}\end{gathered} \tag{2.42}$$

A substitution of (2.42) into (2.32) yields the expansion $u_{\boldsymbol{r}} = \boldsymbol{\Psi}_1\cdots\boldsymbol{\Psi}_{d-1}\boldsymbol{W}_d^\top\boldsymbol{K}_d^\top$. Next perform a QR decomposition

$$\begin{aligned}&\boldsymbol{W}_d\boldsymbol{\Psi}_{d-1}^\top = \boldsymbol{K}_{d-1}\boldsymbol{W}_{d-1}, \\ &\boldsymbol{K}_{d-1} \in M_{r_{d-1}\times r_{d-2}}(L^2_{\mu_{d-1}}(\Omega_{d-1})), \quad \left\langle \boldsymbol{K}_{d-1}^\top \boldsymbol{K}_{d-1}\right\rangle_{d-1} = \boldsymbol{I}_{r_{d-2}\times r_{d-2}}, \\ &\boldsymbol{W}_{d-1} \in M_{r_{d-2}\times r_{d-2}}(\mathbb{R}) \text{ is upper triangular.}\end{aligned} \tag{2.43}$$

Proceeding recusively in this way we obtain the right orthogonalization

$$u_{\boldsymbol{r}} = \boldsymbol{W}_1^\top\boldsymbol{K}_1^\top\cdots\boldsymbol{K}_d^\top. \tag{2.44}$$

We may have stopped the orthogonalization process at any point to obtain the partial right orthogonalization

$$u_{\boldsymbol{r}} = \boldsymbol{\Psi}_{\leq i}\boldsymbol{W}_{i+1}^\top\boldsymbol{K}_{>i}^\top. \tag{2.45}$$

It is also useful to orthogonalize from the left and right to obtain expansions of the form

$$u_{\boldsymbol{r}} = \boldsymbol{Q}_{\leq i}\boldsymbol{R}_i\boldsymbol{W}_{i+1}^\top\boldsymbol{K}_{>i}^\top, \tag{2.46}$$

where the rank of the matrix $\boldsymbol{R}_i\boldsymbol{W}_{i+1}^\top$ is the i-th component of the true FTT rank of the tensor $u_{\boldsymbol{r}}$.

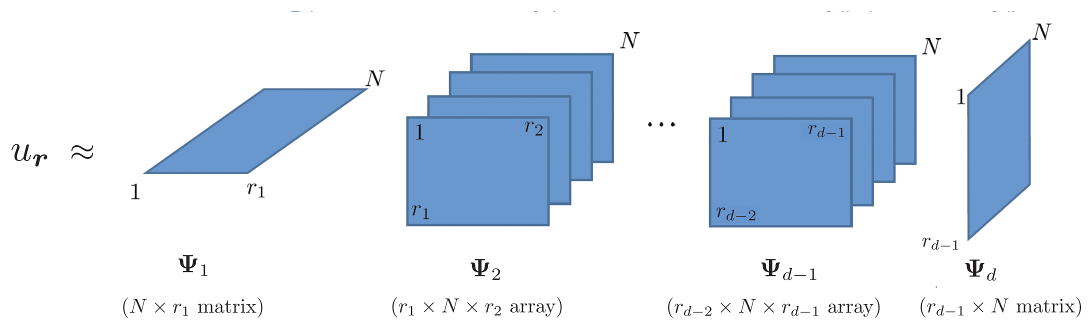


Figure 2.4: Discretization of the low-rank FTT tensor representation (2.33) using N points per dimension.

The truncation of FTT tensors based on their spectra is obtained by computing a singular value decomposition of the matrix $\boldsymbol{R}_i \boldsymbol{W}_{i+1}^\top$ in (2.46) for $i = 1, 2, \ldots, d-1$ and truncating the singular values and corresponding singular vectors up to a given tolerance. Efficient algorithms to perform this operation for TT tensors (a TT tensor is the discrete analogue of the FTT tensor) can be found in [82, section 3] and in [26, 27]. Such algorithms are easily adapted to FTT tensors by replacing QR decompositions of matrices with the QR of FTT cores given in (2.37) and SVD decomposition of matrices with Schmidt decompositions. In numerical implementations, this adaptation amounts to introducing appropriate quadrature weight matrices into the algorithms.

2.3.2 Discretization and computational cost

In order to work with the low-rank FTT representation (2.33) numerically, each continuous function appearing in the expansion must be discretized. One possible discretization[6] of low-rank FTT tensor representations of multivariate functions is obtained by discretizing each

[6]Other discretizations can be obtained by representing one-dimensional tensor cores relative to appropriate sets of basis functions [13].

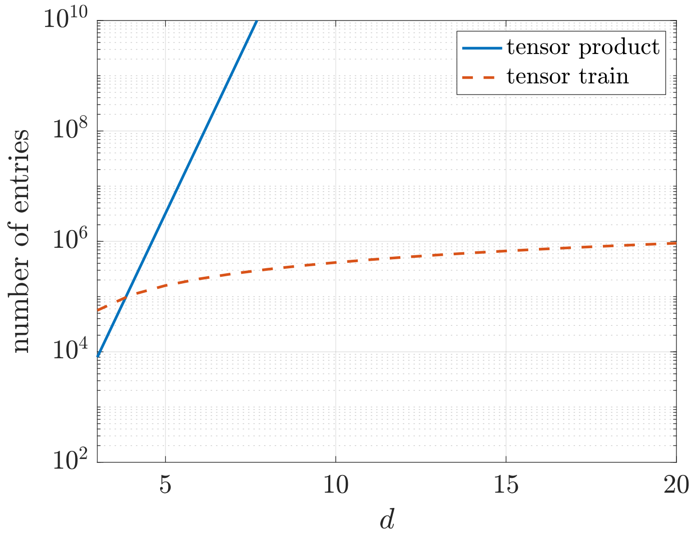


Figure 2.5: Number of entries in the discretized tensor product representation (2.9) and the low-rank FTT tensor representation (2.33) using $N = 128$ points per dimension and truncation rank $r = 20$.

one-dimensional domain Ω_i with N points and considering a collocation representation in each of the tensor cores $\mathbf{\Psi}_i$. Upon discretization, the FTT representation (2.33) becomes a discrete TT tensor which is depicted in Figure 2.4. Assuming that each entry of the FTT rank $\boldsymbol{r}$ (excluding the first entry and the last entry which are always equal to 1) is equal to the same number r, the total number of entries in a discrete TT representation with N points in each dimension is $2Nr + (d-2)Nr^2$, which, scales linearly in the number of dimensions d and quadratically in the rank r. In Figure 2.5 we compare the number of entries in the full tensor product representation of a multivariate function (2.9) with the number of entries in a discrete FTT representation for $N = 128$, $r = 20$ and various dimensions d. We observe that for $d > 3$ the low-rank FTT representation has significantly fewer entries. In addition to the storage cost scaling linear in dimension, the computational cost of many arithmetic operations with FTT format also scales linear with the number of dimensions d.

2.3.3 Numerical example

As a demonstration of the FTT format let us consider a three-dimensional ($d = 3$) function that we define as a sum of Gaussian functions

$$u(\boldsymbol{x}) = \sum_{i=1}^{N_g} w_i \exp\left(-\sum_{j=1}^{d} \frac{1}{\beta_{ij}} \left(\boldsymbol{R}_j^{(i)} \cdot \boldsymbol{x} + t_{ij}\right)^2\right). \tag{2.47}$$

Here N_g is the number of Gaussian functions, β_{ij} are positive real numbers, w_i are positive weights satisfying

$$\sum_{i=1}^{d} w_i = 1,$$

$\boldsymbol{R}_j^{(i)}$ is the j-th row of a $d \times d$ rotation matrix $\boldsymbol{R}^{(i)}$, and t_{ij} are translations. For our demonstration we set $N_g = 3$, $w_i = 1/3$,

$$\begin{array}{llllll}
\beta_{11} = 2, & \beta_{12} = 1/3, & \beta_{13} = 1/2, & t_{11} = 0, & t_{12} = 0, & t_{13} = 0, \\
\beta_{21} = 3, & \beta_{22} = 4, & \beta_{23} = 1/6, & t_{21} = -1, & t_{22} = 1/2, & t_{23} = -1/3, \\
\beta_{31} = 1, & \beta_{32} = 1/5, & \beta_{33} = 5, & t_{31} = 1/2, & t_{32} = -1/4, & t_{33} = 1,
\end{array} \tag{2.48}$$

and the rotation matrices

$$\boldsymbol{R}^{(i)} = \exp\left(\begin{bmatrix} 0 & \theta_i(1) & \theta_i(2) \\ -\theta_i(1) & 0 & \theta_i(3) \\ -\theta_i(2) & -\theta_i(3) & 0 \end{bmatrix}\right), \tag{2.49}$$

with

$$\boldsymbol{\theta}_1 = \begin{bmatrix} \pi/4 \\ \pi/3 \\ \pi/5 \end{bmatrix}, \quad \boldsymbol{\theta}_2 = \begin{bmatrix} \pi/3 \\ \pi/6 \\ \pi/4 \end{bmatrix}, \quad \boldsymbol{\theta}_3 = \begin{bmatrix} \pi/3 \\ \pi/3 \\ \pi/7 \end{bmatrix}. \tag{2.50}$$

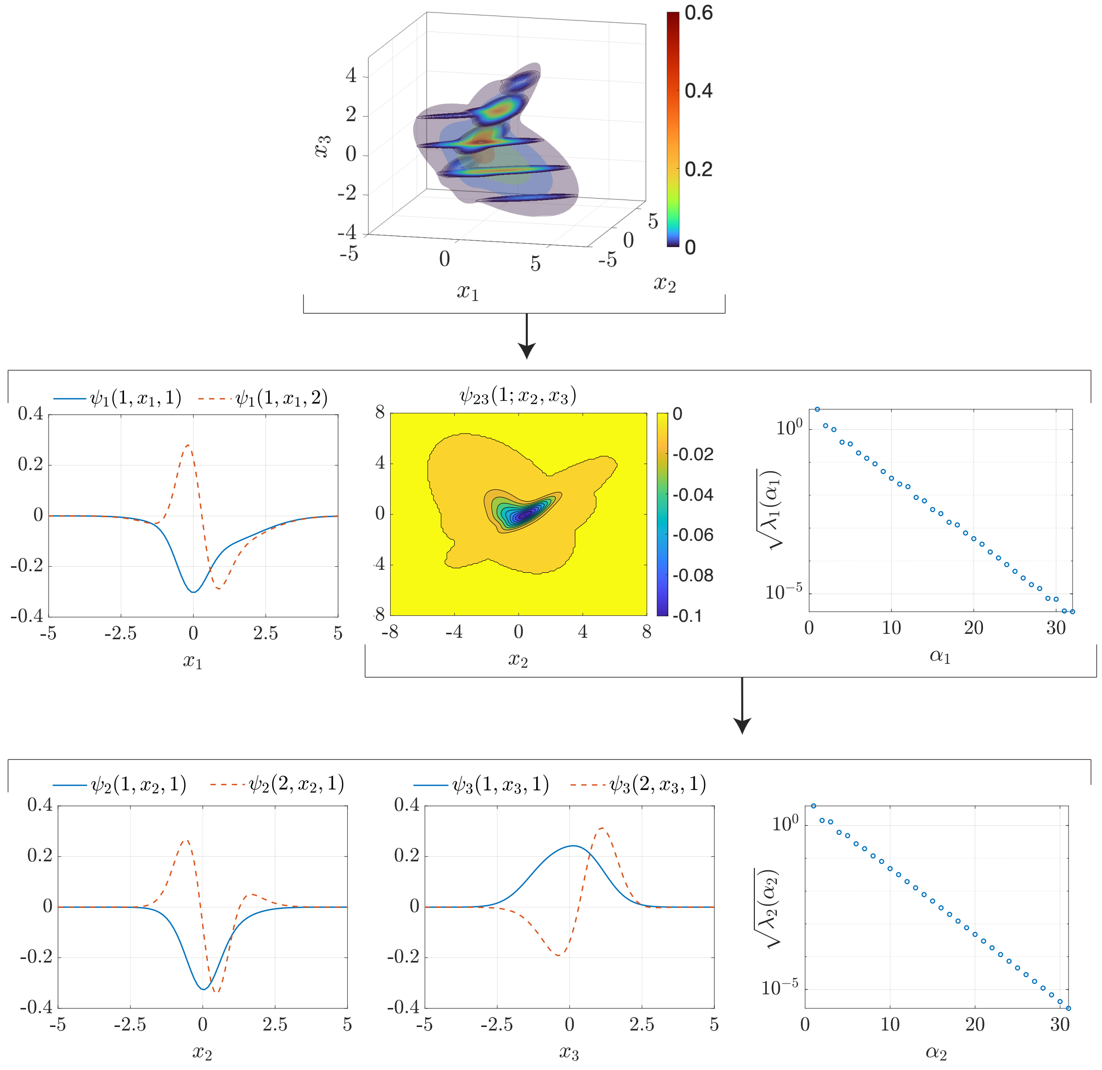


Figure 2.6: Numerical example of FTT decomposition applied to the three dimensional Gaussian mixture (2.47).

We discretize the Gaussian mixture (2.47) on the computational domain $[-12, 12]^3$ (which is large enough to enclose the numerical support of (2.47)) using $N = 200$ evenly-spaced points in each variable. In the top row of Figure 2.6 we provide a volumetric plot of the three-dimensional Gaussian mixture. From the discretization of (2.47) we compute the FTT decomposition $u_{\boldsymbol{r}}(\boldsymbol{x})$ using singular value decompositions recursively. After the first singular value decomposition we obtain a representation of the function $u(\boldsymbol{x})$ in terms of orthonormal x_1 tensor modes $\psi_1(x_1;\alpha_1)$, orthonormal x_2, x_3 tensor modes $\psi_{23}(\alpha_1; x_2, x_3)$ and singular values $\lambda_1(\alpha_1)$ $(\alpha_1 = 1, 2 \ldots, N)$

$$u(\boldsymbol{x}) = \sum_{\alpha_1=1}^{N} \sqrt{\lambda_1(\alpha_1)}\psi_1(x_1;\alpha_1)\psi_{23}(\alpha_1; x_2, x_3). \tag{2.51}$$

Then we truncate the expansion (2.51) to retain all singular values with square root larger than 10^{-6} resulting in the first element of the FTT rank $r_1 = 32$. In the second row of Figure 2.6 we plot the first two x_1 tensor modes, the first x_2, x_3 tensor mode, and the singular values which satisfy $\sqrt{\lambda_1(\alpha_1)} < 10^{-6}$. Next we perform a second singular value decomposition on the collection of x_1, x_2 tensor modes weighted by their corresponding singular values to obtain an expansion in terms of orthonormal x_2 tensor modes, orthonormal x_3 tensor modes, and singular values $\lambda_2(\alpha_2)$

$$\psi_{23}(\alpha_1; x_2, x_3) = \sum_{\alpha_2=1}^{N} \sqrt{\lambda_2(\alpha_2)}\psi_2(\alpha_1; x_2; \alpha_2)\psi_3(\alpha_2; x_3). \tag{2.52}$$

Then we truncate the expansion (2.52) to retain all singular values with square root larger than 10^{-6} resulting in the second element of the FTT rank $r_2 = 30$. In the third row of Figure 2.6 we plot two x_2 tensor modes, two x_3 tensor modes, and the singular values which satisfy $\sqrt{\lambda_2(\alpha_2)} < 10^{-6}$. The final FTT approximation of the Gaussian mixture (2.47) is then given in terms of the tensor modes depending on one spatial variable

$$u(\boldsymbol{x}) \approx u_{\boldsymbol{r}}(\boldsymbol{x}) = \sum_{\alpha_1=1}^{r_1}\sum_{\alpha_2=1}^{r_2} \psi_1(x_1;\alpha_1)\psi_2(\alpha_1; x_2; \alpha_2)\psi_3(\alpha_2; x_3). \tag{2.53}$$

2.4 The geometry of FTT tensors

A key feature of low-rank tensor formats is that collections of tensors of a given rank form a smooth submanifold of the corresponding ambient Hilbert space. Many algorithms involving low-rank tensors including the algorithms we present in Chapter 3 and Chapter 4 rely on this feature. In this section we prove that the space of constant rank FTT tensors is a smooth manifold, which therefore admits a tangent space and a normal space at each point. To prove that the space of constant rank FTT tensors is a smooth manifold, we follow a similar construction as presented in [78, 77]. Closely related work was presented in [21] in relation to Slater–type variational spaces in many particle Hartree–Fock theory. Also, the discrete analogues of the infinite-dimensional tensor manifolds discussed hereafter were studied in detail in [105, 47].

Let $\mathbf{\Phi}$ denote an arbitrary $s_1 \times s_2$ matrix with entries in $L^2_{\mu_i}(\Omega_i)$, $L^2_{\mu_{\leq i}}(\Omega_{\leq i})$ or $L^2_{\mu_{>i}}(\Omega_{>i})$. We write such a matrix as

$$\mathbf{\Phi} = \begin{bmatrix} \phi(1,1) & \cdots & \phi(1,s_2) \\ \vdots & \ddots & \vdots \\ \phi(s_1,1) & \cdots & \phi(s_1,s_2) \end{bmatrix}, \tag{2.54}$$

where $\phi(j,k)$ depends on variable x_i if $\phi(j,k) \in L^2_{\mu_i}(\Omega_i)$. Similarly, $\phi(j,k)$ depends on the variables $(x_1,\ldots,x_i)$ if $\phi(j,k) \in L^2_{\mu_{\leq i}}(\Omega_{\leq i})$ or on the variables $(x_{i+1},\ldots,x_d)$ if $\phi(j,k) \in$

$L^2_{\mu_{>i}}(\Omega_{>i})$. For the following discussion it is convenient to define the integrals

$$\begin{aligned}
\langle \mathbf{\Phi} \rangle_i &= \int_{\Omega_i} \mathbf{\Phi}(x_i) d\mu_i(x_i), \\
\langle \mathbf{\Phi} \rangle_{\leq i} &= \int_{\Omega_{\leq i}} \mathbf{\Phi}(x_1, \ldots, x_i) d\mu_{\leq i}(x_1, \ldots, x_i), \\
\langle \mathbf{\Phi} \rangle_{>i} &= \int_{\Omega_{>i}} \mathbf{\Phi}(x_{i+1}, \ldots, x_d) d\mu_{>i}(x_{i+1}, \ldots, x_d).
\end{aligned} \tag{2.55}$$

Denote by $V^{(i)}_{r_{i-1}\times r_i}$ the set of all FTT tensor cores $\mathbf{\Psi}_i \in M_{r_{i-1}\times r_i}(L^2_{\mu_i}(\Omega_i))$ with the property that the covariance matrix $\langle \mathbf{\Psi}_i^\top \mathbf{\Psi}_i \rangle_i \in M_{r_i \times r_i}(\mathbb{R})$ is invertible. We are interested in the subset $\mathcal{M}_{\boldsymbol{r}} \subseteq L^2_\mu(\Omega)$ consisting of rank-$\boldsymbol{r}$ FTT tensors in d dimensions

$$\mathcal{M}_{\boldsymbol{r}} = \{u \in L^2_\mu(\Omega) : \quad u = \mathbf{\Psi}_1 \mathbf{\Psi}_2 \cdots \mathbf{\Psi}_d, \quad \mathbf{\Psi}_i \in V^{(i)}_{r_{i-1}\times r_i}, \quad \forall i = 1, 2, \ldots, d\}. \tag{2.56}$$

The set

$$V = V^{(1)}_{r_0 \times r_1} \times V^{(2)}_{r_1 \times r_2} \times \cdots \times V^{(d)}_{r_{d-1}\times r_d} \tag{2.57}$$

can be interpreted as a latent space for $\mathcal{M}_{\boldsymbol{r}}$ via the mapping

$$\begin{aligned}
\pi : V &\to \mathcal{M}_{\boldsymbol{r}}, \\
(\mathbf{\Psi}_1, \mathbf{\Psi}_2, \ldots, \mathbf{\Psi}_d) &\mapsto \mathbf{\Psi}_1 \mathbf{\Psi}_2 \cdots \mathbf{\Psi}_d.
\end{aligned} \tag{2.58}$$

Each tensor $u_{\boldsymbol{r}} \in \mathcal{M}_{\boldsymbol{r}}$ has many representations in V, that is the map $\pi(\cdot)$ is not injective. The purpose of the following Lemma 2.4.1 and Proposition 2.4.1 is to characterize all elements of the space V which have the same image under π.

Lemma 2.4.1. *If $\{\psi_k(\alpha_i, \alpha_j)\}_{\alpha_j=1}^r$, $\{\tilde{\psi}_k(\alpha_i, \alpha_j)\}_{\alpha_j=1}^r$ are two bases for the same finite dimensional subspace of $L^2_{\tau\times\mu_k}(\mathbb{N}\times\Omega_k)$ and $\mathbf{\Psi}_k$, $\tilde{\mathbf{\Psi}}_k$ are the corresponding matrices* (2.54)*, then the matrix $\langle \mathbf{\Psi}_k^\top \tilde{\mathbf{\Psi}}_k \rangle_k$ is invertible.*

Proof. The matrix under consideration is given by

$$\langle \boldsymbol{\Psi}_k^\top \tilde{\boldsymbol{\Psi}}_k \rangle_k = \begin{bmatrix} \sum_{i=1}^{r} \left\langle \psi_k(i,1)\tilde{\psi}_k(i,1) \right\rangle_k & \cdots & \sum_{i=1}^{r} \left\langle \psi_k(i,1)\tilde{\psi}_k(i,r) \right\rangle_k \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{r} \left\langle \psi_k(i,r)\tilde{\psi}_k(i,1) \right\rangle_k & \cdots & \sum_{i=1}^{r} \left\langle \psi_k(i,r)\tilde{\psi}_k(i,r) \right\rangle_k \end{bmatrix}. \tag{2.59}$$

We will show that the columns of this matrix are linearly independent. To this end, consider the linear equation

$$\sum_{j=1}^{r} a_j \begin{bmatrix} \sum_{i=1}^{r} \left\langle \psi_k(i,1)\tilde{\psi}_k(i,j) \right\rangle_k \\ \vdots \\ \sum_{i=1}^{r} \left\langle \psi_k(i,r)\tilde{\psi}_k(i,j) \right\rangle_k \end{bmatrix} = \mathbf{0}, \qquad a_j \in \mathbb{R}, \tag{2.60}$$

the p-th row of which we may write as

$$\sum_{i=1}^{r} \left\langle \psi_k(i,p) \sum_{j=1}^{r} a_j \tilde{\psi}_k(i,j) \right\rangle_k = 0, \qquad p = 1, \ldots, r. \tag{2.61}$$

If not all the a_j are equal to zero then (2.61) implies that $\sum_{j=1}^{r} a_j \tilde{\psi}_k(i,j)$ is orthogonal to $\psi_k(i,p)$ in $L^2_{\tau\times\mu_k}(\mathbb{N}\times\Omega_k)$ and therefore linearly independent for all $p = 1, \ldots, r$. This contradicts the assumption that $\{\psi_k(i,j)\}_{j=1}^{r}$, $\{\tilde{\psi}_k(i,j)\}_{j=1}^{r}$ span the same finite dimensional subspace of $L^2_{\tau\times\mu_k}(\mathbb{N}\times\Omega_k)$. Hence a_j is equal to zero for all $j = 1, \ldots, r$.

□

Proposition 2.4.1. *Let $(\boldsymbol{\Psi}_i)_{i=1}^{d}$, $(\tilde{\boldsymbol{\Psi}}_i)_{i=1}^{d}$ be elements of V. Then*

$$\pi(\boldsymbol{\Psi}_1, \ldots, \boldsymbol{\Psi}_d) = \pi(\tilde{\boldsymbol{\Psi}}_1, \ldots, \tilde{\boldsymbol{\Psi}}_d) \tag{2.62}$$

if and only if there exist matrices $\boldsymbol{P}_i \in GL_{r_i\times r_i}(\mathbb{R})$ ($i = 0, 1, \ldots, d$) such that $\boldsymbol{\Psi}_i = \boldsymbol{P}_{i-1}^{-1}\tilde{\boldsymbol{\Psi}}_i\boldsymbol{P}_i$ with $\boldsymbol{P}_0, \boldsymbol{P}_d = 1$.

Proof. We prove the forward implication by induction on d. For $d = 2$ we have

$$\boldsymbol{\Psi}_1 \boldsymbol{\Psi}_2 = \tilde{\boldsymbol{\Psi}}_1 \tilde{\boldsymbol{\Psi}}_2 \tag{2.63}$$

implies

$$\boldsymbol{\Psi}_1 = \tilde{\boldsymbol{\Psi}}_1 \left\langle \tilde{\boldsymbol{\Psi}}_2 \boldsymbol{\Psi}_2^\top \right\rangle_2 \left\langle \boldsymbol{\Psi}_2 \boldsymbol{\Psi}_2^\top \right\rangle_2^{-1}. \tag{2.64}$$

Set $\boldsymbol{P}_1 = \left\langle \tilde{\boldsymbol{\Psi}}_2 \boldsymbol{\Psi}_2^\top \right\rangle_2 \left\langle \boldsymbol{\Psi}_2 \boldsymbol{\Psi}_2^\top \right\rangle_2^{-1}$ which is a change of basis matrix and therefore invertible. Substituting $\boldsymbol{\Psi}_1 = \tilde{\boldsymbol{\Psi}}_1 \boldsymbol{P}_1$ into (2.63) we see that

$$\tilde{\boldsymbol{\Psi}}_1 \boldsymbol{P}_1 \boldsymbol{\Psi}_2 = \tilde{\boldsymbol{\Psi}}_1 \tilde{\boldsymbol{\Psi}}_2, \tag{2.65}$$

which implies

$$\boldsymbol{\Psi}_2 = \boldsymbol{P}_1^{-1} \tilde{\boldsymbol{\Psi}}_2. \tag{2.66}$$

This proves the proposition for $d = 2$. Suppose the proposition holds true for $d - 1$ and that

$$\boldsymbol{\Psi}_1 \cdots \boldsymbol{\Psi}_d = \tilde{\boldsymbol{\Psi}}_1 \cdots \tilde{\boldsymbol{\Psi}}_d. \tag{2.67}$$

Then,

$$\boldsymbol{\Psi}_1 \cdots \boldsymbol{\Psi}_{d-1} = \tilde{\boldsymbol{\Psi}}_1 \cdots \tilde{\boldsymbol{\Psi}}_{d-1} \left\langle \tilde{\boldsymbol{\Psi}}_d \boldsymbol{\Psi}_d^\top \right\rangle_d \left\langle \boldsymbol{\Psi}_d \boldsymbol{\Psi}_d^\top \right\rangle_d^{-1}, \tag{2.68}$$

and we are gauranteed the existence of invertible matrices $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_{d-2}$ such that

$$\begin{aligned}
\boldsymbol{\Psi}_1 &= \tilde{\boldsymbol{\Psi}}_1 \boldsymbol{P}_1, \\
\boldsymbol{\Psi}_2 &= \boldsymbol{P}_1^{-1} \tilde{\boldsymbol{\Psi}}_2 \boldsymbol{P}_2, \\
&\vdots \\
\boldsymbol{\Psi}_{d-2} &= \boldsymbol{P}_{d-3}^{-1} \tilde{\boldsymbol{\Psi}}_{d-2} \boldsymbol{P}_{d-2}, \\
\boldsymbol{\Psi}_{d-1} &= \boldsymbol{P}_{d-2}^{-1} \tilde{\boldsymbol{\Psi}}_{d-1} \left\langle \tilde{\boldsymbol{\Psi}}_d \boldsymbol{\Psi}_d^\top \right\rangle_d \left\langle \boldsymbol{\Psi}_d \boldsymbol{\Psi}_d^\top \right\rangle_d^{-1}.
\end{aligned} \tag{2.69}$$

Let $\boldsymbol{P}_{d-1} = \left\langle \tilde{\boldsymbol{\Psi}}_d \boldsymbol{\Psi}_d^\top \right\rangle_d \left\langle \boldsymbol{\Psi}_d \boldsymbol{\Psi}_d^\top \right\rangle_d^{-1}$. Substituting the expressions for $\boldsymbol{\Psi}_i$ ($i = 1, \ldots, d-1$) in (2.69) into (2.67) yields

$$\tilde{\boldsymbol{\Psi}}_1 \cdots \tilde{\boldsymbol{\Psi}}_{d-2} \tilde{\boldsymbol{\Psi}}_{d-1} \boldsymbol{P}_{d-1} \boldsymbol{\Psi}_d = \tilde{\boldsymbol{\Psi}}_1 \cdots \tilde{\boldsymbol{\Psi}}_d, \tag{2.70}$$

which implies

$$\boldsymbol{P}_{d-1} \boldsymbol{\Psi}_d = \tilde{\boldsymbol{\Psi}}_d. \tag{2.71}$$

From the preceding equation we realize that the matrix $\boldsymbol{P}_{d-1}$ is a change of basis matrix and therefore invertible. Upon writing

$$\boldsymbol{\Psi}_d = \boldsymbol{P}_{d-1}^{-1} \tilde{\boldsymbol{\Psi}}_d, \tag{2.72}$$

we prove the forward implication. The backward implication is trivial.

□

With Proposition 2.4.1 in mind we define the group[7]

$$\mathcal{G} = \mathrm{GL}_{r_1 \times r_1}(\mathbb{R}) \times \mathrm{GL}_{r_2 \times r_2}(\mathbb{R}) \times \cdots \times \mathrm{GL}_{r_{d-1} \times r_{d-1}}(\mathbb{R}), \tag{2.73}$$

with group operation given by component-wise matrix multiplication. Let $\mathcal{G}$ act on V by

$$(\boldsymbol{P}_1, \ldots, \boldsymbol{P}_{d-1}) \cdot (\boldsymbol{\Psi}_1, \ldots, \boldsymbol{\Psi}_d) = (\boldsymbol{\Psi}_1 \boldsymbol{P}_1, \boldsymbol{P}_1^{-1} \boldsymbol{\Psi}_2 \boldsymbol{P}_2, \ldots, \boldsymbol{P}_{d-1}^{-1} \boldsymbol{\Psi}_d) \tag{2.74}$$

for all $(\boldsymbol{P}_1, \ldots, \boldsymbol{P}_{d-1}) \in \mathcal{G}$ and $(\boldsymbol{\Psi}_1, \ldots, \boldsymbol{\Psi}_d) \in V$. It is easy to see that this is action is free and transitive making $\mathcal{G}$, V, $\mathcal{M}_{\boldsymbol{r}}$ and π a principal $\mathcal{G}$-bundle [96]. In particular $V/\mathcal{G}$ is isomorphic to $\mathcal{M}_{\boldsymbol{r}}$ which allows us to equip $\mathcal{M}_{\boldsymbol{r}}$ with a manifold structure. Thus, we may define its tangent

[7]In equation (2.73) $\mathrm{GL}_{r_1 \times r_i}(\mathbb{R})$ denotes the general linear group of $r_i \times r_i$ invertible matrices with real entries, together with the operation of matrix multiplication.

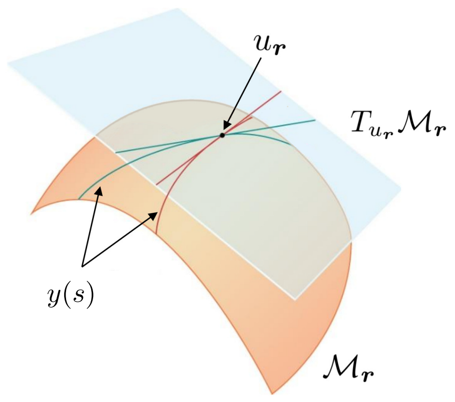

$u_{\boldsymbol{r}}$
$T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}$
$y(s)$
$\mathcal{M}_{\boldsymbol{r}}$

Figure 2.7: A few curves $y(s)$ on the tensor manifold $\mathcal{M}_{\boldsymbol{r}}$ parameterized by $s \in (-\delta, \delta)$ passing through $u_{\boldsymbol{r}}$. The velocities of all such curves define the tangent space $T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}$ to $\mathcal{M}_{\boldsymbol{r}}$ at the point $u_{\boldsymbol{r}}$.

space $T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}$ at a point $u_{\boldsymbol{r}} \in \mathcal{M}_{\boldsymbol{r}}$. We represent elements of the tangent space $T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}$ as equivalence classes of velocities of curves passing through the point $u_{\boldsymbol{r}}$

$$T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}} = \left\{ y'(s)|_{s=0} : \quad y \in \mathcal{C}^1\left((-\delta,\delta), \mathcal{M}_{\boldsymbol{r}}\right), \quad y(0) = u_{\boldsymbol{r}} \right\}. \tag{2.75}$$

Here $\mathcal{C}^1\left((-\delta,\delta), \mathcal{M}_{\boldsymbol{r}}\right)$ is the space of continuously differentiable functions from the interval $(-\delta,\delta)$ to the space of constant rank FTT tensors $\mathcal{M}_{\boldsymbol{r}}$. The following Lemma guarantees a convenient representation for each $u_{\boldsymbol{r}} \in \mathcal{M}_{\boldsymbol{r}}$ in the space V.

Lemma 2.4.2. *For any $u_{\boldsymbol{r}} \in \mathcal{M}_{\boldsymbol{r}}$ there exist $\boldsymbol{\Psi}_i \in V^{(i)}_{r_{i-1}\times r_i}$ $(i = 1, 2, \ldots, d)$ such that $u_{\boldsymbol{r}} = \boldsymbol{\Psi}_1\boldsymbol{\Psi}_2\cdots\boldsymbol{\Psi}_d$ and $\left\langle \boldsymbol{\Psi}_i^\top \boldsymbol{\Psi}_i \right\rangle_i = \boldsymbol{I}_{r_i\times r_i}$ for all $i = 1, \ldots, d-1$.*

Proof. Let us first represent $u_{\boldsymbol{r}} \in \mathcal{M}_{\boldsymbol{r}}$ relative to the tensor cores $\{\tilde{\boldsymbol{\Psi}}_1, \ldots, \tilde{\boldsymbol{\Psi}}_d\}$. Since $\left\langle \tilde{\boldsymbol{\Psi}}_1^\top \tilde{\boldsymbol{\Psi}}_1 \right\rangle_1$ is symmetric there exists an orthogonal matrix $\boldsymbol{P}_1$ such that $\boldsymbol{\Lambda}_1 = \boldsymbol{P}_1^\top \left\langle \tilde{\boldsymbol{\Psi}}_1^\top \tilde{\boldsymbol{\Psi}}_1 \right\rangle_1 \boldsymbol{P}_1$ is diagonal. Set $\boldsymbol{\Psi}_1 = \tilde{\boldsymbol{\Psi}}_1 \boldsymbol{P}_1 \boldsymbol{\Lambda}_1^{-1/2}$ and $\hat{\boldsymbol{\Psi}}_2 = \boldsymbol{\Lambda}_1^{1/2} \boldsymbol{P}_1^T \tilde{\boldsymbol{\Psi}}_2$ so that $\left\langle \boldsymbol{\Psi}_1^\top \boldsymbol{\Psi}_1 \right\rangle_1 = \boldsymbol{I}_{r_1\times r_1}$ and

$\boldsymbol{\Psi}_1 \hat{\boldsymbol{\Psi}}_2 \tilde{\boldsymbol{\Psi}}_3 \cdots \tilde{\boldsymbol{\Psi}}_d = \tilde{\boldsymbol{\Psi}}_1 \cdots \tilde{\boldsymbol{\Psi}}_d$. The matrix $\left\langle \hat{\boldsymbol{\Psi}}_2^\top \hat{\boldsymbol{\Psi}}_2 \right\rangle_2$ is symmetric so there exists an orthogonal matrix $\boldsymbol{P}_2$ such that $\boldsymbol{\Lambda}_2 = \boldsymbol{P}_2^\top \left\langle \hat{\boldsymbol{\Psi}}_2^\top \hat{\boldsymbol{\Psi}}_2 \right\rangle_2 \boldsymbol{P}_2$ is diagonal. Set $\boldsymbol{\Psi}_2 = \hat{\boldsymbol{\Psi}}_2 \boldsymbol{P}_2 \boldsymbol{\Lambda}_2^{-1/2}$ and $\hat{\boldsymbol{\Psi}}_3 = \boldsymbol{\Lambda}_2^{1/2} \boldsymbol{P}_2^T \tilde{\boldsymbol{\Psi}}_3$ so that $\left\langle \boldsymbol{\Psi}_2^\top \boldsymbol{\Psi}_2 \right\rangle_2 = \boldsymbol{I}_{r_2 \times r_2}$ and $\boldsymbol{\Psi}_1 \boldsymbol{\Psi}_2 \hat{\boldsymbol{\Psi}}_3 \tilde{\boldsymbol{\Psi}}_4 \cdots \tilde{\boldsymbol{\Psi}}_d = \tilde{\boldsymbol{\Psi}}_1 \cdots \tilde{\boldsymbol{\Psi}}_d$. We proceed recursively in this way until $\boldsymbol{\Psi}_1 \boldsymbol{\Psi}_2 \cdots \boldsymbol{\Psi}_{d-1} \hat{\boldsymbol{\Psi}}_d = \tilde{\boldsymbol{\Psi}}_1 \cdots \tilde{\boldsymbol{\Psi}}_d$ with $\left\langle \boldsymbol{\Psi}_i^\top \boldsymbol{\Psi}_i \right\rangle_i = \boldsymbol{I}_{r_i \times r_i}$, $i = 1, \ldots, d-1$. Upon setting $\boldsymbol{\Psi}_d = \hat{\boldsymbol{\Psi}}_d$, the required collection of cores $\{\boldsymbol{\Psi}_1, \ldots, \boldsymbol{\Psi}_d\}$ is obtained.

□

Since $L^2_\mu(\Omega)$ is an inner product space, for each $u \in L^2_\mu(\Omega)$ the tangent space $T_u L^2_\mu(\Omega)$ is canonically isomorphic to $L^2_\mu(\Omega)$. Moreover, for each $u_{\boldsymbol{r}} \in \mathcal{M}_{\boldsymbol{r}}$ the normal space to $\mathcal{M}_{\boldsymbol{r}}$ at the point $u_{\boldsymbol{r}}$, denoted by $N_{u_{\boldsymbol{r}}} \mathcal{M}_{\boldsymbol{r}}$, consists of all vectors in $L^2_\mu(\Omega)$ that are orthogonal to $T_{u_{\boldsymbol{r}}} \mathcal{M}_{\boldsymbol{r}}$ with respect to the inner product in $L^2_\mu(\Omega)$

$$N_{u_{\boldsymbol{r}}} \mathcal{M}_{\boldsymbol{r}} = \{ w \in L^2_\mu(\Omega) : \langle w, v \rangle_{L^2_\mu(\Omega)} = 0, \quad \forall v \in T_{u_{\boldsymbol{r}}} \mathcal{M}_{\boldsymbol{r}} \}. \tag{2.76}$$

Since the tangent space $T_{u_{\boldsymbol{r}}} \mathcal{M}_{\boldsymbol{r}}$ is closed, for each point $u_{\boldsymbol{r}} \in \mathcal{M}_{\boldsymbol{r}}$ the space $L^2_\mu(\Omega)$ admits a decomposition into tangential and normal components

$$L^2_\mu(\Omega) = T_{u_{\boldsymbol{r}}} \mathcal{M}_{\boldsymbol{r}} \oplus N_{u_{\boldsymbol{r}}} \mathcal{M}_{\boldsymbol{r}}. \tag{2.77}$$

The partition (2.77) will be useful for the development of rank-adaptive tensor integrators in Chapter 3.

Chapter 3

Low-rank tensor approximation of high-dimensional nonlinear PDEs

In this chapter we use FTT manifolds $\mathcal{M}_{\boldsymbol{r}}$ to approximate a function $u(\boldsymbol{x}, t)$ governed by the autonomous evolution equation (1.1). We present two methodologies for approximating $u(\boldsymbol{x}, t)$ with a time-dependent FTT tensor. This first is dynamic approximation which is based on projecting the PDE dynamics onto the tensor manifold tangent space at each time step. The second is step-truncation which is based on projecting the solution onto the tensor manifold at each time step. These methods are discussed in sections 3.2 and 3.3 respectively. In section 3.4 we show that the integration schemes resulting from these two methodologies are consistent as the time step size approaches zero. Before presenting the schemes we discuss representations of the operator $G : H(\Omega) \to H(\Omega)$ defining the dynamics of the PDE (1.1) that are compatible with FTT tensors $u_{\boldsymbol{r}}$.

3.1 Separable operators

As is well-known, solving (1.1) numerically involves repeated application of G. In particular, if the approximate solution of the given PDE is represented as a FTT tensor $u \approx u_{\text{TT}}$ then the operator G must be represented in a form that can take u_{TT} as an input and output another FTT tensor. If G is a linear operator then such a representation is given by the rank $\boldsymbol{g}$ FTT-operator (or TT-matrix after disceretization [82])

$$G(\cdot, \boldsymbol{x}) \approx G_{\text{TT}}(\cdot, \boldsymbol{x}) = \sum_{\alpha_1=1}^{g_1} \sum_{\alpha_2=1}^{g_2} \cdots \sum_{\alpha_{d-1}=1}^{g_{d-1}} \boldsymbol{A}_1(x_1; \alpha_1) \otimes \boldsymbol{A}_2(\alpha_1; x_2; \alpha_2) \otimes \cdots \otimes \boldsymbol{A}_d(\alpha_{d-1}; x_d), \tag{3.1}$$

where, for fixed α_{i-1} and α_i, $\boldsymbol{A}_j$ is a one-dimensional operator acting only on functions of x_j. The representation (3.1) is also known as matrix product operator (MPO) [75]. After applying G_{TT} to a FTT tensor u_{TT} with rank $\boldsymbol{r}$, the new FTT tensor $G_{\text{TT}}(u_{\text{TT}}, \boldsymbol{x})$ has rank determined by the element-wise (Hadamard) product of the two ranks $\boldsymbol{g} \circ \boldsymbol{r}$, which then has to be truncated. The computational cost of such a truncation scales cubically in the new FTT rank. If the product rank $\boldsymbol{g} \circ \boldsymbol{r}$ is prohibitively large then the FTT operator G_{TT} can be split into sums of low rank operators

$$G_{\text{TT}} = \sum_{k=1}^{n} G_{\text{TT}}^{(k)}, \tag{3.2}$$

where each $G_{\text{TT}}^{(k)}$ has FTT operator rank $\boldsymbol{g}^{(k)}$, which is less than $\boldsymbol{g}$. Hence, instead of applying G_{TT} directly to the solution tensor, we can apply each $G_{\text{TT}}^{(k)}$ ($k = 1, 2, \ldots, n$) to u_{TT}, truncate each $G_{\text{TT}}^{(k)}(u_{\text{TT}}, \boldsymbol{x})$, and then add them together. After the addition, one more truncation procedure must be performed to ensure the result of the FTT addition has optimal ranks. This procedure can be written mathematically as

$$G_{\text{TT}}(u_{\text{TT}}, \boldsymbol{x}) \approx \mathfrak{T}_\delta \left[\sum_{k=1}^{n} \mathfrak{T}_\delta \left(G_{\text{TT}}^{(k)}(u_{\text{TT}}, \boldsymbol{x}) \right) \right], \tag{3.3}$$

where $\mathfrak{T}_\delta$ is a truncation (or rounding) operator for FTT tensors with relative accuracy δ. Alternatively, one can use randomized algorithms, e.g., based on tensor sketching, for computing sums of many TT tensors [26]. This can increase efficiency of applying high rank FTT operators to FTT tensors.

3.2 Dynamic approximation methods

As mentioned above, the dynamic approximation approach for initial/boundary value problems of the form (1.1) aims at determining the vector in the tangent space of $\mathcal{M}_{\boldsymbol{r}}$ at the point $u_{\boldsymbol{r}}$ that best approximates $\partial u_{\boldsymbol{r}}/\partial t$ at each t in the time domain of interest $[0, T]$. One way to obtain the optimal tangent vector is by orthogonal projection which we now describe. For each $u \in L^2_\mu(\Omega)$, the tangent space $T_u L^2_\mu(\Omega)$ is canonically isomorphic to $L^2_\mu(\Omega)$. Moreover, for each $u_{\boldsymbol{r}} \in \mathcal{M}_{\boldsymbol{r}}$ the normal space to $\mathcal{M}_{\boldsymbol{r}}$ at the point $u_{\boldsymbol{r}}$, denoted by $N_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}$, consists of all vectors in $L^2_\mu(\Omega)$ that are orthogonal to $T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}$ with respect to the inner product in $L^2_\mu(\Omega)$. For each $u_{\boldsymbol{r}} \in \mathcal{M}_{\boldsymbol{r}}$ the space $L^2_\mu(\Omega)$ admits the decomposition into a tangential component and a normal component (see equation (2.77)). Assuming that the solution to the PDE (1.1) lives on the manifold $\mathcal{M}_{\boldsymbol{r}}$ at time t, we may write its velocity $\partial u_{\boldsymbol{r}}/\partial t = G(u_{\boldsymbol{r}})$ as a unique combination of tangential and normal components relative to $\mathcal{M}_{\boldsymbol{r}}$

$$G(u_{\boldsymbol{r}}) = v + w, \qquad v \in T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}, \quad w \in N_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}} \tag{3.4}$$

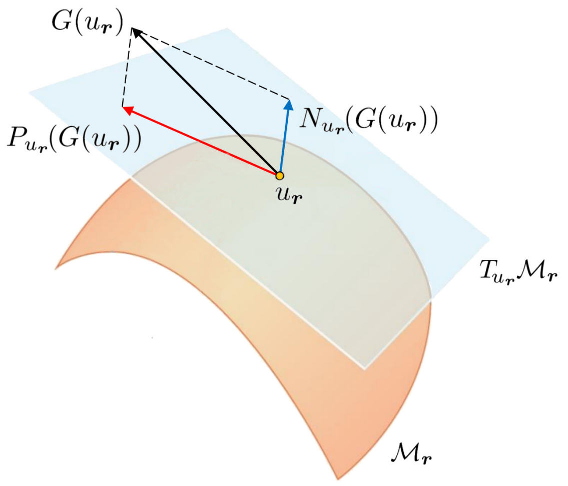

$G(u_{\boldsymbol{r}})$
$N_{u_{\boldsymbol{r}}}(G(u_{\boldsymbol{r}}))$
$P_{u_{\boldsymbol{r}}}(G(u_{\boldsymbol{r}}))$
$u_{\boldsymbol{r}}$
$T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}$
$\mathcal{M}_{\boldsymbol{r}}$

Figure 3.1: A sketch of the tangent and normal components of $\partial u_{\boldsymbol{r}}/\partial t = G\left(u_{\boldsymbol{r}}\right)$ to the manifold $\mathcal{M}_{\boldsymbol{r}}$ at the point $u_{\boldsymbol{r}}$.

(see Figure 3.1). The orthogonal projection we are interested in computing for dynamic approximation is

$$\begin{aligned} P_{u_{\boldsymbol{r}}} : L^2_{\mu}(\Omega) &\to T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}, \\ G(u_{\boldsymbol{r}}) &\mapsto P_{u_{\boldsymbol{r}}}G(u_{\boldsymbol{r}}). \end{aligned} \tag{3.5}$$

If the initial condition $u_0(\boldsymbol{x})$ is on the manifold[1] $\mathcal{M}_{\boldsymbol{r}}$, then the solution to the initial/boundary value problem

$$\begin{cases} \dfrac{\partial u_{\boldsymbol{r}}}{\partial t} = P_{u_{\boldsymbol{r}}}G(u_{\boldsymbol{r}}), \\ u(\boldsymbol{x}, 0) = u_0(\boldsymbol{x}), \end{cases} \tag{3.6}$$

remains on the manifold $\mathcal{M}_{\boldsymbol{r}}$ for all $t \geq 0$. The solution to (3.6) is known as a dynamic approximation to the solution of (1.1).

In practice, we can compute the image of $P_{u_{\boldsymbol{r}}}$ at each $t \in [0, T]$ by solving the

[1]If the initial condition $u_0(\boldsymbol{x})$ does not belong to $\mathcal{M}_{\boldsymbol{r}}$ then it can be projected onto $\mathcal{M}_{\boldsymbol{r}}$ using a truncation operator (see section 2.3.1

following minimization problem over the tangent space of $\mathcal{M}_{\boldsymbol{r}}$ at $u_{\boldsymbol{r}}$

$$\min_{v \in T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}} \left\| v(\boldsymbol{x},t) - \frac{\partial u_{\boldsymbol{r}}(\boldsymbol{x},t)}{\partial t} \right\|_{L^2_\mu(\Omega)}^2 = \min_{v \in T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}} \| v(\boldsymbol{x},t) - G(u_{\boldsymbol{r}}(\boldsymbol{x},t)) \|_{L^2_\mu(\Omega)}^2 . \tag{3.7}$$

We emphasize that (3.7) defines an infinite family of optimization problems parameterized by $t \in [0,T]$. Prior to solving the optimization problems in (3.7), let us establish that they admit a unique solution.

Proposition 3.2.1. *Fix $t \in [0,T]$. If $G(u_{\boldsymbol{r}}) \notin T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}$ then there exists a unique solution to the minimization problem* (3.7)*, i.e., a unique global minimum.*

Proof. We first notice that the feasible set $T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}$ is a real vector space and thus a convex set. Next we show that the functional $F[v] = \|v - G(u_{\boldsymbol{r}})\|_{L^2_\mu(\Omega)}^2$ is strictly convex. Indeed, take $v_1, v_2 \in T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}$ distinct and $q \in (0,1)$. Then for all $t \in [0,T]$

$$\begin{aligned}
(F[qv_1 + (1-q)v_2])^{1/2} &= \|qv_1 + (1-q)v_2 - G(u_{\boldsymbol{r}})\|_{L^2_\mu(\Omega)} \\
&= \|q(v_1 - G(u_{\boldsymbol{r}})) + (1-q)\left((v_2 - G(u_{\boldsymbol{r}})\right)\|_{L^2_\mu(\Omega)} \\
&\leq q\|v_1 - G(u_{\boldsymbol{r}})\| + (1-q)\|v_2 - G(u_{\boldsymbol{r}})\|_{L^2_\mu(\Omega)},
\end{aligned} \tag{3.8}$$

with equality if and only if there exists an $\alpha > 0$ such that $q(v_1 - G(u_{\boldsymbol{r}})) = \alpha(1-q)(v_2 - G(u_{\boldsymbol{r}}))$. However, the existence of such an α implies that $v_1 - \beta v_2 = (1-\beta)G(u_{\boldsymbol{r}})$ for some real number β, whence $G(u_{\boldsymbol{r}}) \in T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}$. Therefore if $G(u_{\boldsymbol{r}}) \notin T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}$ then the inequality in (3.8) is strict and the functional $(F[v])^{1/2}$ is strictly convex. Since the function x^2 is strictly increasing on the image of $F^{1/2}$ it follows that F is strictly convex and therefore admits a unique global minimum over the feasible set $T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}$.

□

It can easily be shown that the unique solution to each of the optimization problems in (3.7) is $P_{u_r}G(u_{\boldsymbol{r}})$. Hereafter we will compute the orthogonal projection $P_{u_{\boldsymbol{r}}}G(u_{\boldsymbol{r}})$ by solving the optimization problems (3.7) at each $t \in [0, T]$. Then we will use this tangent vector to integrate the solution of the PDE (1.1) forward in time on the manifold $\mathcal{M}_{\boldsymbol{r}}$. First, we must assume that the solution $u(\boldsymbol{x}, t)$ to the PDE at time t is on the manifold $\mathcal{M}_{\boldsymbol{r}}$. Under this assumption, the solution admits an expansion in terms of FTT cores

$$u_{\boldsymbol{r}}(\boldsymbol{x}, t) = \boldsymbol{\Psi}_1(t)\boldsymbol{\Psi}_2(t)\cdots\boldsymbol{\Psi}_d(t) \tag{3.9}$$

satisfying $\left\langle \boldsymbol{\Psi}_i^\top(t)\boldsymbol{\Psi}_i(t)\right\rangle_i = \boldsymbol{I}_{r_i\times r_i}$ for $i = 1, 2, \ldots, d-1$ (see Lemma 2.4.2). In order to solve the optimization problem (3.7) at time t, we expand an arbitrary $\mathcal{C}^1$ curve $y(t, s)$ on the manifold $\mathcal{M}_{\boldsymbol{r}}$ passing through $u_{\boldsymbol{r}}(\boldsymbol{x}, t) \in \mathcal{M}_{\boldsymbol{r}}$ at $s = 0$ in terms of s-dependent FTT cores (see Figure 2.7)

$$\begin{aligned} y(t,s) &= \boldsymbol{\Gamma}_1(t,s)\cdots\boldsymbol{\Gamma}_d(t,s) \\ &= \sum_{\alpha_0,\ldots,\alpha_d=1}^{\boldsymbol{r}} \gamma_1(t,s;\alpha_0,\alpha_1)\gamma_2(t,s;\alpha_1,\alpha_2)\cdots\gamma_d(t,s;\alpha_{d-1},\alpha_d). \end{aligned} \tag{3.10}$$

This allows us to represent any element of the tangent space $T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}$ at $u_{\boldsymbol{r}}(\boldsymbol{x}, t)$ as

$$v = \frac{\partial}{\partial s}\left[\boldsymbol{\Gamma}_1(t,s)\cdots\boldsymbol{\Gamma}_d(t,s)\right]_{s=0}, \tag{3.11}$$

with $\boldsymbol{\Gamma}_1(t, 0)\cdots\boldsymbol{\Gamma}_d(t, 0) = u_{\boldsymbol{r}}(\boldsymbol{x}, t)$. At this point we notice that minimizing the functional in (3.7) for a fixed $t \in [0, T]$ over the tangent space $T_{u_{\boldsymbol{r}}}\mathcal{M}_{\boldsymbol{r}}$ is equivalent to minimizing the same functional over the velocity of each of the FTT cores. In other words, rather than solving (3.7) we will instead solve

$$\min_{\left.\frac{\partial\boldsymbol{\Gamma}_1}{\partial s}\right|_{s=0},\ldots,\left.\frac{\partial\boldsymbol{\Gamma}_d}{\partial s}\right|_{s=0}} \left\| \left.\frac{\partial}{\partial s}\left(\boldsymbol{\Gamma}_1\cdots\boldsymbol{\Gamma}_d\right)\right|_{s=0} - G(u_{\boldsymbol{r}})\right\|_{L^2_\mu(\Omega)}^2 \tag{3.12}$$

at each $t \in [0,T]$. In view of Lemma 2.4.1, the FTT core velocities which solve the minimization problem (3.12) are not unique. In order to obtain a unique solution of the optimization problem (3.12) at each $t \in [0,T]$, we enforce constraints on the cores $\mathbf{\Gamma}_i(t,s)$. One set of constraints which allows us to obtain a unique solution can be written as

$$\left\langle \mathbf{\Gamma}_i^\top(t,s)\mathbf{\Gamma}_i(t,s)\right\rangle_i = \boldsymbol{I}_{r_i\times r_i}, \qquad \forall t\in[0,T], \quad \forall s\in(-\delta,\delta), \quad \forall i = 1,\ldots,d-1. \tag{3.13}$$

Note that any curve (3.10) admits an expansion in terms of cores satisfying (3.13) thanks to Lemma 2.4.2. In order to introduce these constraints into the minimization problem we first recast them as equations involving both the cores and their velocities as follows. Differentiate (3.13) with respect to s to obtain

$$\left\langle \frac{\partial \mathbf{\Gamma}_i^\top(t,s)}{\partial s}\mathbf{\Gamma}_i(t,s)\right\rangle_i = -\left\langle \mathbf{\Gamma}_i^\top(t,s)\frac{\partial \mathbf{\Gamma}_i(t,s)}{\partial s}\right\rangle_i, \tag{3.14}$$

which is attained when

$$\left\langle \frac{\mathbf{\Gamma}_i^\top(t,s)}{\partial s}\mathbf{\Gamma}_i(t,s)\right\rangle_i = \mathbf{0}_{r_i\times r_i}, \qquad \forall t\in[0,T], \quad \forall s\in(-\delta,\delta), \quad \forall i = 1,\ldots,d-1. \tag{3.15}$$

Hence the set of constraints

$$\begin{cases} \left\langle \dfrac{\mathbf{\Gamma}_i^\top(t,s)}{\partial s}\mathbf{\Gamma}_i(t,s)\right\rangle_i & = \mathbf{0}_{r_i\times r_i}, \qquad \forall t\in[0,T], \quad \forall s\in(-\delta,\delta), \quad \forall i = 1,\ldots,d-1, \\ \left\langle \mathbf{\Gamma}_i^\top(t,0)\mathbf{\Gamma}_i(t,0)\right\rangle_i & = \boldsymbol{I}_{r_i\times r_i}, \qquad \forall t\in[0,T], \quad \forall i = 1,\ldots,d-1, \end{cases} \tag{3.16}$$

are equivalent to the constraints in (3.13). Using the constraints given in (3.16), we will obtain the optimal core velocities $\partial\widetilde{\mathbf{\Gamma}}_i(t,0)/\partial s$ which minimize (3.12) at each time $t\in[0,T]$. These optimal core velocities define the optimal tangent vector (3.11) for the minimization problem (3.7), i.e., the orthogonal projection $P_{u_{\boldsymbol{r}}}(G(u_{\boldsymbol{r}}))$ (see equation (3.5)). To integrate the low-rank solution (3.9) forward in time, we set the time derivative of the FTT solution cores equal to the core velocities defining the optimal tangent vector

$$\frac{\partial \boldsymbol{\Psi}_i(t)}{\partial t} = \frac{\partial \widetilde{\boldsymbol{\Gamma}}_i(t,0)}{\partial s}, \qquad \forall i = 1,2,\dots,d. \tag{3.17}$$

This yields the minimization problem[2]

$$\begin{cases} \displaystyle\min_{\frac{\partial \boldsymbol{\Psi}_1}{\partial t},\dots,\frac{\partial \boldsymbol{\Psi}_d}{\partial t}} \left\| \frac{\partial}{\partial t}\left[\boldsymbol{\Psi}_1\boldsymbol{\Psi}_2\cdots\boldsymbol{\Psi}_d\right] - G(u_{\boldsymbol{r}})\right\|^2_{L^2_\mu(\Omega)} \\ \text{subject to:} \\ \displaystyle\left\langle \frac{\boldsymbol{\Psi}_i^\top(t)}{\partial t}\boldsymbol{\Psi}_i(t)\right\rangle_i = \mathbf{0}_{r_i\times r_i}, \quad \forall t\in[0,T], \quad \forall i = 1,2,\dots,d-1. \end{cases} \tag{3.18}$$

The solution to this problem provides the optimal time derivative of the FTT approximation of the solution to the PDE (1.1) on the manifold $\mathcal{M}_{\boldsymbol{r}}$. The minimization problem (3.18) is a convex optimization problem subject to linear equality constraints, which therefore is still convex. Hence, any local minimum is also a global minimum. To solve (3.18), it is convenient to construct an action functional $\mathcal{A}$ which introduces the constraints via Lagrange multipliers $\lambda^{(i)}_{\alpha_i\beta_i}$ and expands the products of FTT cores as summations of scalar functions. The action functional is given explicitly as

$$\begin{aligned} &\mathcal{A}\left(\frac{\partial \psi_1(\alpha_0,\alpha_1)}{\partial t},\dots,\frac{\partial \psi_d(\alpha_{d-1},\alpha_d)}{\partial t}\right) = \\ &\quad\left\| \frac{\partial}{\partial t}\left[\sum_{\alpha_0,\dots,\alpha_d=1}^{\boldsymbol{r}} \psi_1(\alpha_0,\alpha_1)\psi_2(\alpha_1,\alpha_2)\cdots\psi_d(\alpha_{d-1},\alpha_d)\right] - G(u_{\boldsymbol{r}})\right\|^2_{L^2_\mu(\Omega)} + \\ &\qquad\sum_{i=1}^{d-1}\sum_{\alpha_i,\beta_i=1}^{r_i}\lambda^{(i)}_{\alpha_i\beta_i}\left\langle \frac{\partial \psi_i(\alpha_{i-1},\alpha_i)}{\partial t}, \psi_i(\alpha_{i-1},\beta_i)\right\rangle_{L^2_{\tau\times\mu_i}(\mathbb{N}\times\Omega_i)}. \end{aligned} \tag{3.19}$$

[2]Note that if the FTT decomposition of the initial condition satisfies the second line of equality constraints in (3.16), i.e., $\left\langle \boldsymbol{\Psi}_i^\top(0)\boldsymbol{\Psi}_i(0)\right\rangle_i = \boldsymbol{I}_{r_i\times r_i}$ $(i = 1,2,\dots,d-1)$, then the constraints in (3.18) imply that $\left\langle \boldsymbol{\Psi}_i^\top(t)\boldsymbol{\Psi}_i(t)\right\rangle_i = \boldsymbol{I}_{r_i\times r_i}$ $(i = 1,2,\dots,d-1)$ for all $t\in[0,T]$.

At this point, we are ready to formulate the FTT propagator for the nonlinear PDE (1.1), which is the system of Euler-Lagrange equations corresponding to the unique global minimum of (3.19). Such propagator provides velocities of the FTT solution cores corresponding to the orthogonal projection (3.5). These core velocities determine the best dynamic approximation of the solution to (1.1) on a FTT tensor manifold with constant rank.

Theorem 3.2.1. *The unique global minimum of the functional* (3.19) *is attained at FTT tensor cores satisfying the PDE system*

$$\begin{aligned}
\frac{\partial \boldsymbol{\Psi}_1}{\partial t} &= \left[\left\langle G(u_{\boldsymbol{r}}) \boldsymbol{\Psi}_{>1}^\top \right\rangle_{>1} - \boldsymbol{\Psi}_1 \left\langle \left\langle \boldsymbol{\Psi}_1^\top G(u_{\boldsymbol{r}}) \right\rangle_1 \boldsymbol{\Psi}_{>1}^\top \right\rangle_{>1} \right] \left\langle \boldsymbol{\Psi}_{>1} \boldsymbol{\Psi}_{>1}^\top \right\rangle_{>1}^{-1}, \\
\frac{\partial \boldsymbol{\Psi}_k}{\partial t} &= \left[\left\langle \left\langle \boldsymbol{\Psi}_{\leq k-1}^\top G(u_{\boldsymbol{r}}) \right\rangle_{\leq k-1} \boldsymbol{\Psi}_{>k}^\top \right\rangle_{>k} \right. \\
&\qquad \left. - \boldsymbol{\Psi}_k \left\langle \left\langle \boldsymbol{\Psi}_{\leq k}^\top G(u_{\boldsymbol{r}}) \right\rangle_{\leq k} \boldsymbol{\Psi}_{>k}^\top \right\rangle_{>k} \right] \left\langle \boldsymbol{\Psi}_{>k} \boldsymbol{\Psi}_{>k}^\top \right\rangle_{>k}^{-1}, \qquad k = 2, 3, \ldots, d-1, \\
\frac{\partial \boldsymbol{\Psi}_d}{\partial t} &= \left\langle \boldsymbol{\Psi}_{\leq d-1}^\top G(u_{\boldsymbol{r}}) \right\rangle_{\leq d-1},
\end{aligned} \tag{3.20}$$

where $\boldsymbol{\Psi}_{\leq k} = \boldsymbol{\Psi}_1 \boldsymbol{\Psi}_2 \cdots \boldsymbol{\Psi}_k$ *and* $\boldsymbol{\Psi}_{>k} = \boldsymbol{\Psi}_{k+1} \cdots \boldsymbol{\Psi}_d$.

We prove Theorem 3.2.1 in Appendix 3.A. We refer to the PDE system (3.20) as the dynamically orthogonal [99] functional tensor train (DO-FTT) propagator. The DO-FTT system (3.20) involves several inverse covariance matrices $\left\langle \boldsymbol{\Psi}_{\geq k} \boldsymbol{\Psi}_{\geq k}^\top \right\rangle_{\geq k}^{-1}$, which can become poorly conditioned in the presence of tensor modes with small energy (i.e. autocovariance matrices with small singular values). This phenomenon has been shown to be a result of the fact that the curvature of the tensor manifold at a tensor is inversely proportional to the smallest singular value present in the tensor [61, section 4]. A slight improvement to the numerical stability of (3.20) can be obtained by right orthogonalizing (see section 2.3.1) the partial products

$$\boldsymbol{\Psi}_{\geq k} = \boldsymbol{R}_k^\top \boldsymbol{Q}_{\geq k}^\top, \qquad k = 2, \ldots, d. \tag{3.21}$$

Using the orthogonality of $\boldsymbol{Q}_k$ it can easily be verified that $\boldsymbol{R}_k = \left\langle \boldsymbol{\Psi}_{\geq k} \boldsymbol{\Psi}_{\geq k}^\top \right\rangle_{\geq k}^{1/2}$. With these right orthogonalized cores, the DO-FTT system (3.20) can be written as

$$\begin{aligned}
\frac{\partial \boldsymbol{\Psi}_1}{\partial t} &= \left[\langle G(u_{\boldsymbol{r}}) \boldsymbol{Q}_{>1} \rangle_{>1} - \boldsymbol{\Psi}_1 \left\langle \boldsymbol{\Psi}_1^\top G(u_{\boldsymbol{r}}) \boldsymbol{Q}_{>1} \right\rangle_{\geq 1} \right] \left\langle \boldsymbol{\Psi}_{>1} \boldsymbol{\Psi}_{>1}^\top \right\rangle_{\geq 2}^{-1/2}, \\
\frac{\partial \boldsymbol{\Psi}_k}{\partial t} &= \Bigg[\left\langle \boldsymbol{\Psi}_{\leq k-1}^\top G(u_{\boldsymbol{r}}) \boldsymbol{Q}_{>k} \right\rangle_{\leq k-1, >k} \\
&\qquad - \boldsymbol{\Psi}_k \left\langle \boldsymbol{\Psi}_{\leq k}^\top G(u_{\boldsymbol{r}}) \boldsymbol{Q}_{>k} \right\rangle_{\geq 1} \Bigg] \left\langle \boldsymbol{\Psi}_{>k} \boldsymbol{\Psi}_{>k}^\top \right\rangle_{>k}^{-1/2}, \qquad k = 2, 3, \ldots, d-1, \\
\frac{\partial \boldsymbol{\Psi}_d}{\partial t} &= \left\langle \boldsymbol{\Psi}_{\leq d-1}^\top G(u_{\boldsymbol{r}}) \right\rangle_{\leq d-1},
\end{aligned} \tag{3.22}$$

where $\left\langle \boldsymbol{\Psi}_{\geq k} \boldsymbol{\Psi}_{\geq k}^\top \right\rangle_{k,\ldots,d}^{-1/2}$ denotes the inverse of the matrix square root. Since the condition number of $\left\langle \boldsymbol{\Psi}_{\geq k} \boldsymbol{\Psi}_{\geq k}^\top \right\rangle_{\geq k}$ is larger than the condition number of $\left\langle \boldsymbol{\Psi}_{\geq k} \boldsymbol{\Psi}_{\geq k}^\top \right\rangle_{\geq k}^{1/2}$, we have that the inverse covariances at the right hand side of (3.22) can be computed more accurately than the ones in (3.20) in the presence of small singular values. To increase robustness of the DO-FTT system even further, they may be treated with operator splitting methods or unconventional integration schemes (see section 3.2.1 for more details).

We conclude this section with a few important remarks on the DO-FTT propagator (3.20). As the solution evolves in time on the tensor manifold $\mathcal{M}_{\boldsymbol{r}}$, it is possible for the solution rank to decrease. Consequently, the auto-correlation matrices $\left\langle \boldsymbol{\Psi}_{>k} \boldsymbol{\Psi}_{>k}^\top \right\rangle_{>k}$ become singular and the equations (3.20) are no longer valid. In this case, the solution lives on a tensor manifold of smaller rank, say $\mathcal{M}_{\boldsymbol{s}}$, where $s_i \leq r_i$ for all $i = 1, 2, \ldots, d$, and the dynamic tensor approximation can be constructed on $\mathcal{M}_{\boldsymbol{s}}$. On the other hand, if the PDE (1.1) is not well approximated on $\mathcal{M}_{\boldsymbol{r}}$, then one can increase the tensor rank $\boldsymbol{r}$ adaptively in time to retain accuracy, e.g. by thresholding the norm of the velocity vector $N_{u_{\boldsymbol{r}}}(G(u_{\boldsymbol{r}}))$ normal to the tensor manifold [28] (see section 3.5 for an in depth discussion of rank-adaptive tensor integration). Finally, we mention that it is possible to transform the dynamically orthogonal tensor cores $\boldsymbol{\Psi}_i$ into bi-orthogonal cores (with corresponding bi-orthogonal evolution equations) by adopting the proofs given in [29, 24]. In light of the discussion above on the optimality of the DO-FTT propagator on $\mathcal{M}_{\boldsymbol{r}}$ and Lemma 2.4.1, it is clear that FTT with bi-orthogonal cores is also an optimal[3] dynamic approximation on $\mathcal{M}_{\boldsymbol{r}}$.

3.2.1 Temporal integration using operator splitting methods

One of the challenges of dynamic approximation of PDEs on low-rank tensor manifolds relates to the curvature of the manifold, which is proportional to the inverse of the smallest singular value of $\left\langle \boldsymbol{\Psi}_{\geq k}\boldsymbol{\Psi}_{\geq k}^{\top}\right\rangle_{\geq k}$ [61, section 4]. Such curvature appears naturally at the right hand side of the DO-FTT system (3.20) in the form of inverse covariance matrices $\left\langle \boldsymbol{\Psi}_{\geq k}\boldsymbol{\Psi}_{\geq k}^{\top}\right\rangle_{\geq k}^{-1}$. Clearly, if the tensor solution is comprised of cores with small singular values, then the covariance matrices $\left\langle \boldsymbol{\Psi}_{\geq k}\boldsymbol{\Psi}_{\geq k}^{\top}\right\rangle_{\geq k}$ are ill-conditioned and therefore not easily invertible. Moreover, it is desirable to add and remove tensor modes adaptively during temporal integration, and adding a mode with zero energy immediately yields singular covariance

[3] By selecting a collection of time-dependent invertible matrices $\boldsymbol{P}_i(s) \in \mathrm{GL}_{r_i\times r_i}(\mathbb{R})$, $i = 2, 3, \dots, d-1$, defined by the matrix differential equation

$$\begin{cases} \dfrac{d\boldsymbol{P}_i(s)}{ds} = G_i(\boldsymbol{P}_i) \\ \boldsymbol{P}_i(0) = \boldsymbol{P}_{i,0} \end{cases} \tag{3.23}$$

it is possible to develop evolution equations other than (3.20) for $\boldsymbol{\Psi}_i(s)$, which still solve the minimization problem (3.12). The bi-orthogonal method discussed in [29, 24] is a result of one choice of matrix differential equation (3.23).

matrices (see [29]). The problem of inverting the covariance matrices $\left\langle \boldsymbol{\Psi}_{\geq k} \boldsymbol{\Psi}_{\geq k}^{\top} \right\rangle_{\geq k}$ when integrating (3.20) can be avoided by using projector-splitting methods. These methods were originally proposed for integration on tensor manifolds by Lubich *et. al* in [69, 59, 70]. The key idea is to apply an exponential operator splitting scheme, e.g., the Lie-Trotter scheme, directly to the projection operator onto the tangent space defining the dynamic approximation (see equation (3.5)). To describe the method, we begin by introducing a general framework for operator splitting of dynamics on the FTT tangent space. We first rewrite the orthogonal tangent space projection (3.5) as

$$\begin{aligned} P_{u_{\boldsymbol{r}}} G(u_{\boldsymbol{r}}) &= \dot{\boldsymbol{\Psi}}_1 \boldsymbol{\Psi}_{\geq 2} + \boldsymbol{\Psi}_1 \dot{\boldsymbol{\Psi}}_2 \boldsymbol{\Psi}_{\geq 3} + \cdots + \boldsymbol{\Psi}_{\leq d-1} \dot{\boldsymbol{\Psi}}_d \\ &= \dot{\boldsymbol{\Psi}}_1 \left\langle \boldsymbol{\Psi}_{\geq 2} \boldsymbol{\Psi}_{\geq 2}^{\top} \right\rangle_{\geq 2}^{1/2} \boldsymbol{Q}_{\geq 2}^{\top} + \boldsymbol{\Psi}_1 \dot{\boldsymbol{\Psi}}_2 \left\langle \boldsymbol{\Psi}_{\geq 3} \boldsymbol{\Psi}_{\geq 3}^{\top} \right\rangle_{\geq 3}^{1/2} \boldsymbol{Q}_{\geq 3}^{\top} + \cdots + \boldsymbol{\Psi}_{\leq d-1} \dot{\boldsymbol{\Psi}}_d, \end{aligned} \tag{3.24}$$

where in the second line we used the right orthogonalizations in equation (3.21) and we denoted a derivative with respect to time with a dot above the corresponding tensor core, i.e., $\dot{\boldsymbol{\Psi}}_i = \partial \boldsymbol{\Psi}_i / \partial t$. A substitution of the expressions for $\dot{\boldsymbol{\Psi}}_k$ we obtained in (3.22) into (3.24) yields

$$P_{u_{\boldsymbol{r}}} G(u_{\boldsymbol{r}}) = P_d^{+} G(u_{\boldsymbol{r}}) + \sum_{i=1}^{d-1} P_i^{+} G(u_{\boldsymbol{r}}) - P_i^{-} G(u_{\boldsymbol{r}}), \tag{3.25}$$

where we defined the following projection operators from $L^2_{\mu}(\Omega)$ onto $T_{u_{\boldsymbol{r}}} \mathcal{M}_{\boldsymbol{r}}$

$$\begin{aligned} P_k^{+} z(\boldsymbol{x}) &= \boldsymbol{\Psi}_{\leq k-1} \left\langle \boldsymbol{\Psi}_{\leq k-1}^{\top} z(\boldsymbol{x}) \boldsymbol{Q}_{\geq k+1} \right\rangle_{\leq k-1, \geq k+1} \boldsymbol{Q}_{\geq k+1}^{\top}, \qquad k = 1, \ldots, d, \\ P_k^{-} z(\boldsymbol{x}) &= \boldsymbol{\Psi}_{\leq k} \left\langle \boldsymbol{\Psi}_{\leq k}^{\top} z(\boldsymbol{x}) \boldsymbol{Q}_{\geq k+1} \right\rangle_{\geq 1} \boldsymbol{Q}_{\geq k+1}^{\top}, \qquad k = 1, \ldots, d-1, \end{aligned} \tag{3.26}$$

for any $z(\boldsymbol{x}) \in L^2_{\mu}(\Omega)$. Also we set $\boldsymbol{\Psi}_0 = 1$. The key point in (3.25) is that inverse covariance matrices no longer appear. To establish a general operator splitting framework, let us assume that there exists an evolution operator $\mathcal{E}_{P_{u_{\boldsymbol{r}}} G}$ for the solution of the initial/boundary value problem (3.6), where $P_{u_{\boldsymbol{r}}}G$ is given in (3.25). Such evolution operator $\mathcal{E}_{P_{u_{\boldsymbol{r}}}G} : L^2_\mu(\Omega) \times [0,T] \to L^2_\mu(\Omega)$ satisfies a semi-group property and it maps the initial condition $u_0(\boldsymbol{x})$ into the solution to (3.6) at a later time

$$u(\boldsymbol{x},t) = \mathcal{E}_{P_{u_{\boldsymbol{r}}}G}(u_0(\boldsymbol{x}),t). \tag{3.27}$$

We write such an evolution operator formally as an exponential operator with generator $D_{P_{u_{\boldsymbol{r}}}G}$ (see e.g. [63])

$$u(\boldsymbol{x},t) = e^{tD_{P_{u_{\boldsymbol{r}}}G}} u_0(\boldsymbol{x}), \qquad 0 \le t \le T, \tag{3.28}$$

where $D_{P_{u_{\boldsymbol{r}}}G}$ is the Lie derivative associated with $P_{u_{\boldsymbol{r}}}G$. We now discretize the temporal domain of interest $[0,T]$ into $N+1$ evenly-spaced time instants,

$$t_i = i\Delta t, \qquad \Delta t = \frac{T}{N}, \qquad i = 0,1,\ldots,N. \tag{3.29}$$

An approximation to the exact solution of (3.6) is then obtained by the recurrence relation

$$u_{\boldsymbol{r}}(\boldsymbol{x},t_{n+1}) \approx \mathcal{S}(\Delta t, u_{\boldsymbol{r}}(\boldsymbol{x},t_n)), \tag{3.30}$$

where $\mathcal{S}$ is an exponential operator splitting that approximates the exact evolution operator

$$\mathcal{S}(t,\cdot) = \prod_{i=1}^{s} \left[e^{\gamma_{i,d} t P_d^+ G} \prod_{j=1}^{d-1} \left(e^{\gamma_{i,j} t P_j^+ G} e^{\gamma_{i,j} t P_j^- G} \right) \right]. \tag{3.31}$$

Setting $s = 1$ and $\gamma_{1,j} = 1$ for all $j = 1, \ldots, d$ in (3.31) yields the well-known Lie-Trotter splitting, which is first-order in time. The discrete time version of this scheme can be written as

$$\begin{cases} u_1^+(t_{i+1}) = u_1^+(t_i) + \Delta t P_1^+ G(u_{\boldsymbol{r}}), \qquad u_1^+(t_i) = u_{\boldsymbol{r}}(t_i), \\ u_1^-(t_{i+1}) = u_1^-(t_i) - \Delta t P_1^- G(u_{\boldsymbol{r}}), \qquad u_1^-(t_i) = u_1^+(t_{i+1}), \\ \qquad\quad \vdots \\ u_j^+(t_{i+1}) = u_j^+(t_i) + \Delta t P_j^+ G(u_{\boldsymbol{r}}), \qquad u_j^+(t_i) = u_{j-1}^-(t_{i+1}), \\ u_j^-(t_{i+1}) = u_j^-(t_i) - \Delta t P_j^- G(u_{\boldsymbol{r}}), \qquad u_j^-(t_i) = u_j^+(t_{i+1}), \\ \qquad\quad \vdots \\ u_d^+(t_{i+1}) = u_d^+(t_i) + \Delta t P_d^+ G(u_{\boldsymbol{r}}), \qquad u_d(t_i) = u_{d-1}^-(t_{i+1}), \\ u_{\boldsymbol{r}}(t_{i+1}) = u_d^+(t_{i+1}). \end{cases} \tag{3.32}$$

This allows us to compute $u_{\boldsymbol{r}}(t_{i+1})$ given $u_{\boldsymbol{r}}(t_i)$. Although each equation in (3.32) involves a FTT tensor, it was shown in [69, Theorem 4.1] that each equation only updates one tensor core. Clearly this is computationally more efficient than updating a full tensor. Moreover, in (3.32) there is no need to invert covariance matrices, which is a distinct advantage over iterating a discrete form of (3.20) or (3.22).

Regarding computational cost, suppose we discretize the d-dimensional domain Ω using a tensor product grid with n points per dimension. It was pointed out in [69] that the computational complexity of the sweeping algorithm to update the tensor cores for the Lie-Trotter scheme (3.32) applied to a linear PDE (i.e. equation (3.6) with linear G) is linear in the dimension d but has high polynomial complexity in the tensor rank. On the other hand, discretizing such linear PDE on the same tensor product grid and performing one time step with a first-order time stepping scheme (e.g. Euler forward) has computational complexity which scales exponentially with the dimension d. Specifically, assuming that the operator G in (1.1) is linear with rank $\boldsymbol{g} = \begin{bmatrix} g & \dots & g \end{bmatrix}$ (see section 3.1), the computational cost of one time step of Euler forward is $dn^{d+1}g + n^d g$ floating point operations, hence exponential in d.

3.3 Step-truncation methods

Another methodology to integrate nonlinear PDEs on fixed-rank tensor manifolds $\mathcal{M}_{\boldsymbol{r}}$ is step-truncation [60, 92, 93]. The idea is to integrate the solution off of $\mathcal{M}_{\boldsymbol{r}}$ for short time, e.g., by performing one time step of the full equation with a conventional time-stepping scheme, followed by a truncation operation back onto $\mathcal{M}_{\boldsymbol{r}}$. To describe this method further let us define the truncation operator[4]

$$\begin{aligned} \mathfrak{T}_{\boldsymbol{r}} &: L^2_\mu(\Omega) \to \mathcal{M}_{\boldsymbol{r}} \\ \mathfrak{T}_{\boldsymbol{r}}(u) &= \underset{u_{\boldsymbol{r}} \in \mathcal{M}_{\boldsymbol{r}}}{\operatorname{argmin}} \|u - u_{\boldsymbol{r}}\|_{L^2_\mu(\Omega)}, \end{aligned} \tag{3.33}$$

which provides the best approximation of u on $\mathcal{M}_{\boldsymbol{r}}$. Such a map is known as a metric projection or closest point function and in general it may be multivalued, i.e., the set of $u_{\boldsymbol{r}} \in \mathcal{M}_{\boldsymbol{r}}$ which minimize $\|u - u_{\boldsymbol{r}}\|_{L^2_\mu(\Omega)}$ is not a singleton set. However, since $\mathcal{M}_{\boldsymbol{r}}$ is a smooth submanifold of $L^2_\mu(\Omega)$, we have by [97, Proposition 5.1] that for each $u_0 \in \mathcal{M}_{\boldsymbol{r}}$ there exists an open neighborhood U of u_0 such that $\mathfrak{T}_{\boldsymbol{r}}$ is well-defined and smooth on U. Let

$$u(\boldsymbol{x}, t_{k+1}) = u(\boldsymbol{x}, t_k) + \Delta t \Phi \left(G, u(\boldsymbol{x}, t_k), \Delta t\right) \tag{3.34}$$

[4]Throughout this discussion we will also use the truncation operator $\mathfrak{T}_\delta$ which adaptively selects the rank $\boldsymbol{r}$ of the output tensor to have relative accuracy δ, i.e., $\|\mathfrak{T}(u_{\mathrm{TT}}(\boldsymbol{x})) - u_{\mathrm{TT}}(\boldsymbol{x})\|_{H(\Omega)} \leq \delta \|u_{\mathrm{TT}}(\boldsymbol{x})\|_{H(\Omega)}$.

be a convergent one-step time integration scheme[5] approximating the solution to the initial value problem (1.1). Assume that the solution $u(\boldsymbol{x}, t_0)$ at time t_0 is on $\mathcal{M}_{\boldsymbol{r}}$.[6] In order to guarantee the solution $u(\boldsymbol{x}, t_k)$ at time step t_k is an element of the manifold $\mathcal{M}_{\boldsymbol{r}}$ for each $k = 1, 2, \ldots$, we apply the truncation operator to the right hand side (3.34). This yields the following step-truncation method

$$u_{\boldsymbol{r}}(\boldsymbol{x}, t_{k+1}) = \mathfrak{T}_{\boldsymbol{r}} \left(u_{\boldsymbol{r}}(\boldsymbol{x}, t_k) + \Delta t \Phi \left(G, u_{\boldsymbol{r}}(\boldsymbol{x}, t_k), \Delta t \right) \right). \tag{3.35}$$

For example the step-truncation Euler forward scheme is

$$u_{\text{TT}}(\boldsymbol{x}, t_{k+1}) = \mathfrak{T}_{\delta} \left[u_{\text{TT}}(\boldsymbol{x}, t_k) + \Delta t \mathfrak{T}_{\delta} \left(G_{\text{TT}}(u_{\text{TT}}(\boldsymbol{x}, t_k)) \right] , \tag{3.36}$$

and a step-truncation Adams-Bashforth 2 (AB2) scheme[7] is

$$\begin{aligned} &u_{\text{TT}}(\boldsymbol{x}, t_{k+1}) \\ &= \mathfrak{T}_{\delta} \left[u_{\text{TT}}(\boldsymbol{x}, t_k) + \Delta t \left(\frac{3}{2} \mathfrak{T}_{\delta} \left[G_{\text{TT}}(u_{\text{TT}}(\boldsymbol{x}, t_k)) \right] - \frac{1}{2} \mathfrak{T}_{\delta} \left[G_{\text{TT}}(u_{\text{TT}}(\boldsymbol{x}, t_{k-1})) \right] \right) \right]. \end{aligned} \tag{3.37}$$

In the step-truncation Euler scheme and AB2 scheme (3.36)-(3.37) we denote FTT tensors by u_{TT} instead of $u_{\boldsymbol{r}}$ in order to not specify the exact rank of the FTT solution at each time step. The rank at of $u_{\text{TT}}(\boldsymbol{x}, t_k)$ at each time step t_k is determined by the truncation operator $\mathfrak{T}_{\delta}$ with accuracy δ. See [92] for more details on rank-adaptive step-truncation integrators. In section 3.5 we present rank-adaptive methods for dynamic approximation integrators.

[5]Time stepping schemes of the form (3.34) include Runge-Kutta methods and linear multi-step methods [92].

[6]If $u(\boldsymbol{x}, t_0)$ is not on $\mathcal{M}_{\boldsymbol{r}}$ then it may be mapped onto $\mathcal{M}_{\boldsymbol{r}}$ by evaluating $\mathfrak{T}_{\boldsymbol{r}}(u(\boldsymbol{x}, t_0))$.

[7]Other step-truncation schemes can be obtained from the Adams-Bashforth 2 time discretization by inserting or removing truncation operations following operator applications or summations of tensors.

3.4 Consistency of dynamic approximation and step-truncation

Next we ask what happens in the step-truncation algorithm in the limit of time step Δt approaching zero. The result of such a limiting procedure results in a scheme which keeps the solution $u(\boldsymbol{x}, t)$ on the manifold $\mathcal{M}_{\boldsymbol{r}}$ for all time $t \geq t_0$ in an optimal way. We now show that this limiting procedure in fact results in precisely the dynamic approximation method described in section 3.2. In other words, by sending Δt to zero in (3.35) we obtain a solution of (3.6). For similar discussions connecting these two approximation methods in closely related contexts see [35, 36, 60]. To prove consistency between step-truncation and dynamic approximation methods we need to compute $\mathfrak{T}_{\boldsymbol{r}}(u(\boldsymbol{x}, t))$ for t infinitesimally close to t_0. Such quantity depends on the derivative

$$\left.\frac{\partial \mathfrak{T}_{\boldsymbol{r}}(u(\boldsymbol{x}, t))}{\partial t}\right|_{t=t_0} = \lim_{\Delta t \to 0} \frac{\mathfrak{T}_{\boldsymbol{r}}(u(\boldsymbol{x}, t)) - \mathfrak{T}_{\boldsymbol{r}}(u(\boldsymbol{x}, t_0))}{\Delta t}. \tag{3.38}$$

The following proposition provides a representation of the derivative $\partial \mathfrak{T}_{\boldsymbol{r}}(u(\boldsymbol{x}, t))/\partial t$ in terms of $G(u(\boldsymbol{x}, t))$ and the Fréchet derivative [111] of the operator $\mathfrak{T}_{\boldsymbol{r}}(u)$.

Proposition 3.4.1. *If the solution $u_0 = u(\boldsymbol{x}, t_0)$ to* (1.1) *at time t_0 is on the manifold $\mathcal{M}_{\boldsymbol{r}}$, then*

$$\left.\frac{\partial \mathfrak{T}_{\boldsymbol{r}}(u(\boldsymbol{x}, t))}{\partial t}\right|_{t=t_0} = (\mathfrak{T}_{\boldsymbol{r}})'_{u_0} G(u(\boldsymbol{x}, t)), \tag{3.39}$$

where $(\mathfrak{T}_{\boldsymbol{r}})'_{u_0}$ is the Fréchet derivative of the nonlinear operator $\mathfrak{T}_{\boldsymbol{r}}$ at the point u_0.

Proof. Express the solution of (1.1) at time $t \geq t_0$ as

$$u(\boldsymbol{x}, t) = u_0(\boldsymbol{x}) + h(\boldsymbol{x}, t), \tag{3.40}$$

where

$$h(\boldsymbol{x}, t) = \int_{t_0}^{t} G(u(\boldsymbol{x}, \tau)) d\tau. \tag{3.41}$$

Expanding $\mathfrak{T}_{\boldsymbol{r}}(u(\boldsymbol{x},t))$ in a Taylor series around $u_0(\boldsymbol{x})$ we obtain [80, Theorem 6.1]

$$\mathfrak{T}_{\boldsymbol{r}}(u(\boldsymbol{x},t)) = u_0(\boldsymbol{x}) + (\mathfrak{T}_{\boldsymbol{r}})'_{u_0} h(\boldsymbol{x},t) + \frac{1}{2}(\mathfrak{T}_{\boldsymbol{r}})''_{u_0} h(\boldsymbol{x},t)^2 + \cdots . \tag{3.42}$$

Differentiating (3.42) with respect to t and evaluating at $t = t_0$ we obtain

$$\left.\frac{\partial \mathfrak{T}_{\boldsymbol{r}}(u(\boldsymbol{x},t))}{\partial t}\right|_{t=t_0} = (\mathfrak{T}_{\boldsymbol{r}})'_{u_0} G(u(\boldsymbol{x},t_0)), \tag{3.43}$$

where we assumed that $\partial/\partial t$ commutes with $(\mathfrak{T}_{\boldsymbol{r}})'_{u_0}$ and used the fact that $\partial h(\boldsymbol{x},t)/\partial t = G(u(\boldsymbol{x},t))$ for the first order term. All of the higher order terms are seen to be zero by commuting $\partial/\partial t$ with $(\mathfrak{T}_{\boldsymbol{r}})^{(n)}_{u_0}$ and using chain rule.

$\square$

Since $\mathfrak{T}_{\boldsymbol{r}}(u(\boldsymbol{x},t))$ is an element of $\mathcal{M}_{\boldsymbol{r}}$ for all $t \geq t_0$, it follows that (3.39) is an element of $T_{u_0}\mathcal{M}_{\boldsymbol{r}}$. Arguing on the optimality of the tangent space element $(\mathfrak{T}_{\boldsymbol{r}})'_{u_0} G(u(\boldsymbol{x},t_0))$ it is seen that (3.43) is the same problem as dynamic approximation (3.6), i.e., $(\mathfrak{T}_{\boldsymbol{r}})'_{u_0} = P_{u_0}$. Now consider the scheme (3.35) and use a Taylor expansion of $\mathfrak{T}_{\boldsymbol{r}}$ around $u_{\boldsymbol{r}}(\boldsymbol{x},t_k)$ on the right hand side

$$u_{\boldsymbol{r}}(\boldsymbol{x},t_{k+1}) = u_{\boldsymbol{r}}(\boldsymbol{x},t_k) + \Delta t (\mathfrak{T}_{\boldsymbol{r}})' \Phi\left(G, u_{\boldsymbol{r}}(\boldsymbol{x},t_k), \Delta t\right) + O(\Delta t^2). \tag{3.44}$$

Discarding higher order terms in Δt yields

$$u_{\boldsymbol{r}}(\boldsymbol{x},t_{k+1}) \approx u_{\boldsymbol{r}}(\boldsymbol{x},t_k) + \Delta t P_{u_{\boldsymbol{r}}} \Phi\left(G, u_{\boldsymbol{r}}(\boldsymbol{x},t_k), \Delta t\right). \tag{3.45}$$

Moreover if the increment function Φ defines the Euler forward scheme

$$\Phi(G, u_{\boldsymbol{r}}(\boldsymbol{x},t_k), \Delta t) = G(u_{\boldsymbol{r}}(\boldsymbol{x},t_k)), \tag{3.46}$$

then the scheme (3.45) is equivalent to the scheme in (3.6). Thus, we just proved the following lemma.

Lemma 3.4.1. *Step-truncation and dynamic approximation methods are consistent at least to first-order in Δt.*

This Lemma applies to any first-order time integrator for dynamic approximation and step-truncation, including the Lie-Trotter splitting integrator we discussed in section 3.2.1.

3.5 Rank-adaptive time integration

The solution to the initial/boundary value problem (1.1) is often not accurately represented on a tensor manifold with fixed rank, even for short integration times. In this section we discuss effective methods to adaptively add and remove tensor modes from the solution based on appropriate criteria.

In the context of step-truncation algorithms, if the solution rank naturally decreases in time then the operator $\mathfrak{T}_{\boldsymbol{r}}$ in (3.35) is no longer well-defined. In this situation, replacing the operator $\mathfrak{T}_{\boldsymbol{r}}$ with $\mathfrak{T}_{\boldsymbol{s}}$ for an appropriate[8] $\boldsymbol{s} \leq \boldsymbol{r}$ allows for integration to continue. On the other hand, if the solution rank increases in during integration then the operator $\mathfrak{T}_{\boldsymbol{r}}$ will still be well-defined for small enough Δt but the approximation on $\mathcal{M}_{\boldsymbol{r}}$ will not retain accuracy. Rank-adaptive integration for step-truncation integrators can be built by using tolerance based truncation operators $\mathfrak{T}_{\delta}(\cdot)$. With properly chosen truncation tolerances δ the can be made to have a desired order of accuracy (see [92] for more details).

[8]Here $\leq$ denotes component-wise inequality of rank vectors, i.e., $\boldsymbol{s} < \boldsymbol{r}$ if and only if $s_i \leq r_i$ for all $i = 0, 1, \dots, d$.

In the context of dynamic approximation integrators criteria for adding and removing modes during rank-adaptive integration can be obtained by decomposing the velocity of the PDE solution $\partial u/\partial t$ into a tangential component and a normal component relative to the tensor manifold $\mathcal{M}_{\boldsymbol{r}}$. In the following subsection we present these rank-adaptive criteria and provide some analysis on the order of the resulting rank-adaptive integration schemes. For the remainder of this section let $u(\boldsymbol{x}, t)$ be the solution to (1.1) and $u_{\boldsymbol{r}}(\boldsymbol{x}, t) \in \mathcal{M}_{\boldsymbol{r}}$ an approximation of $u(\boldsymbol{x}, t)$ obtained by either the solution of the dynamical approximation problem (3.6) or step-truncation methods (see section 3.3).

3.5.1 Decreasing tensor rank

For decreasing tensor rank at time t, we are interested in determining if $u_{\boldsymbol{r}}(\boldsymbol{x}, t) \in \mathcal{M}_{\boldsymbol{r}}$ is close to an element $u_{\boldsymbol{s}}(\boldsymbol{x}, t) \in \mathcal{M}_{\boldsymbol{s}}$ for $\boldsymbol{s} \leq \boldsymbol{r}$. This can be achieved by simply performing a FTT truncation on $u_{\boldsymbol{r}}(\boldsymbol{x}, t)$ with small threshold ϵ_{dec}. Since the splitting integrator described in section 3.2.1 is robust to over approximation by tensor rank, it may not be strictly necessary to decrease rank during integration. However, it is desirable to have solutions of the lowest rank possible (while retaining accuracy) when solving high dimensional problems. For these reasons it is advisable not perform a FTT truncation at each time step (as this would be unnecessary and inefficient when using an operator splitting integrator) but only every once and a while. One may choose a criterion for when to check for rank decrease based on the problem, step size, current rank, and dimension. If one is using a step-truncation method with a tolerance based FTT truncation algorithm such as the one described in section 2.3.1 then rank decrease is already built into each time step.

3.5.2 Increasing tensor rank

As a general heuristic one would like to increase rank at the time when the error between the low-rank approximation $u_{\boldsymbol{r}}(\boldsymbol{x},t)$ and the PDE solution $u(\boldsymbol{x},t)$ will become large after the subsequent time step. Such critical time instant for rank increase can be determined by examining the normal component of the dynamics

$$N_{u_{\boldsymbol{r}}}(G(u_{\boldsymbol{r}})) = G(u_{\boldsymbol{r}}) - P_{u_{\boldsymbol{r}}}(G(u_{\boldsymbol{r}})). \tag{3.47}$$

To describe this situation further, suppose we are integrating one time step forward from t_i to t_{i+1}. The error at t_{i+1} is given by

$$\begin{aligned} E(t_i,t_{i+1}) &= u_{\boldsymbol{r}}(\boldsymbol{x},t_{i+1}) - u(\boldsymbol{x},t_{i+1}) \\ &= u(\boldsymbol{x},t_i) \\ &+ \int_{t_i}^{t_{i+1}} G(u(\boldsymbol{x},\tau))d\tau - \left(\mathfrak{T}_{\boldsymbol{r}}(u(\boldsymbol{x},t_i)) + \int_{t_i}^{t_{i+1}} P_{u_{\boldsymbol{r}}(\boldsymbol{x},\tau)} G(u_{\boldsymbol{r}}(\boldsymbol{x},\tau))d\tau\right). \end{aligned} \tag{3.48}$$

If $u(\boldsymbol{x},t_i) \in \mathcal{M}_{\boldsymbol{r}}$ then

$$E(t_i,t_{i+1}) = \int_{t_i}^{t_{i+1}} \left[G(u(\boldsymbol{x},\tau)) - P_{u_{\boldsymbol{r}}(\boldsymbol{x},\tau)} G(u_{\boldsymbol{r}}(\boldsymbol{x},\tau))\right] d\tau. \tag{3.49}$$

For small Δt the above integral can be approximated by the left endpoint

$$\begin{aligned} E(t_i,t_{i+1}) &= \Delta t \left(G(u_{\boldsymbol{r}}(\boldsymbol{x},t_i)) - P_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)} G(u_{\boldsymbol{r}}(\boldsymbol{x},t_i))\right) + O(\Delta t^2) \\ &= \Delta t N_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)}(G(u_{\boldsymbol{r}}(\boldsymbol{x},t_i))) + O(\Delta t^2), \end{aligned} \tag{3.50}$$

where $N_{u_{\boldsymbol{r}}(\boldsymbol{x},\tau)}$ denotes the orthogonal projection onto the normal space of $\mathcal{M}_{\boldsymbol{r}}$ at the point $u_{\boldsymbol{r}}(\boldsymbol{x},t)$. Hence, up to first-order in Δt we have that

$$\|E(t_i,t_{i+1})\| \simeq \Delta t \|N_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)}(G(u_{\boldsymbol{r}}(\boldsymbol{x},t_i)))\|. \tag{3.51}$$

From this approximation we see that a reasonable criterion for increasing rank at time t_i is when the norm of the normal component of $G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i))$ is larger than some threshold ϵ_{inc} (see Figure 3.1)

$$\|N_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)}(G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i)))\| > \epsilon_{\text{inc}}. \tag{3.52}$$

To efficiently compute the normal component $N_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)}(G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i)))$ at each time instant t_i we use the formula

$$N_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)}(G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i))) = G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i)) - P_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)}(G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i))), \tag{3.53}$$

where $N_{\boldsymbol{r}}(G(u_{\boldsymbol{r}}))$ and $T_{u_{\boldsymbol{r}}}(G(u_{\boldsymbol{r}}))$ represent the normal and tangential components of $G(u_{\boldsymbol{r}})$. The tangential component can be approximated at a low computational cost via backward differentiation formulas (BDF) as

$$\widetilde{P}^{(2)}_{u_{\boldsymbol{r}}} G(u_{\boldsymbol{r}}) = \frac{u_{\boldsymbol{r}}(\boldsymbol{x}, t_i) - u_{\boldsymbol{r}}(\boldsymbol{x}, t_{i-1})}{\Delta t} + O(\Delta t^2) \qquad \text{(two-point formula)}, \tag{3.54}$$

$$\widetilde{P}^{(3)}_{u_{\boldsymbol{r}}} G(u_{\boldsymbol{r}}) = \frac{3u_{\boldsymbol{r}}(\boldsymbol{x}, t_i) - 4u_{\boldsymbol{r}}(\boldsymbol{x}, t_{i-1}) + u_{\boldsymbol{r}}(\boldsymbol{x}, t_{i-2})}{2\Delta t} + O(\Delta t^3) \quad \text{(three-point formula)}, \tag{3.55}$$

$$\widetilde{P}^{(p)}_{u_{\boldsymbol{r}}} G(u_{\boldsymbol{r}}) = BD_p(\Delta t, u_{\boldsymbol{r}}(\boldsymbol{x}, t_i), \ldots, u_{\boldsymbol{r}}(\boldsymbol{x}, t_{i-p})) + O(\Delta t^p) \qquad (p\text{-point formula}). \tag{3.56}$$

With a p-point backward difference approximation of the tangent space projection available at t_i we easily obtain an approximation of the normal component of $G(u_{\boldsymbol{r}})$ at t_i

$$N_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)}(G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i))) = G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i)) - \widetilde{P}^{(p)}_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)} G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i)) + O(\Delta t^{p+1}), \tag{3.57}$$

which allows us to implement the criterion (3.52) for rank increase at time t_i. Clearly, the p-point formula (3.56), and the corresponding approximation of the normal component (3.57), are effectively of order p in Δt if and only if the time snapshots $u_{\boldsymbol{r}}(\boldsymbol{x}, t_i)$ are computed via a temporal integrator of order p. We emphasize that this method of using a finite difference stencil based on the temporal grid for approximating the tangential component of the dynamics (and thus the normal component) creates a lower bound for the choice of normal vector threshold ϵ_{inc}. In particular, we must have that $K_1(\Delta t)^p \geq \epsilon_{\text{inc}}$ for some constant K_1 otherwise the error incurred from our approximation of the normal component may trigger unnecessary mode addition. This approximation of the normal component is cheap but only informs on whether or not it is appropriate to add modes at time instant t_i.

The subsequent question is which entries of the rank vector $\boldsymbol{r}$ need to be increased. In order to make such a determination we expand the approximate solution at time t as

$$\begin{aligned} &u_{\boldsymbol{r}}(\boldsymbol{x}, t) = \boldsymbol{\Psi}_1(t) \cdots \boldsymbol{\Psi}_d(t) + \boldsymbol{\Gamma}_1(t) \cdots \boldsymbol{\Gamma}_d(t), \\ &\boldsymbol{\Psi}_i \in M_{r_{i-1} \times r_i}(L^2_{\mu_i}(\Omega_i)), \quad \boldsymbol{\Gamma}_i \in M_{f_{i-1} \times f_i}(L^2_{\mu_i}(\Omega_i)), \end{aligned} \tag{3.58}$$

where $\boldsymbol{\Gamma}_1(t) \cdots \boldsymbol{\Gamma}_d(t) = 0$ for all $t \in [0, T]$. Differentiating (3.58) with respect to time yields

$$\frac{\partial u_{\boldsymbol{r}}(\boldsymbol{x}, t)}{\partial t} = \frac{\partial}{\partial t} \left[\boldsymbol{\Psi}_1(t) \cdots \boldsymbol{\Psi}_d(t)\right] + \frac{\partial}{\partial t} \left[\boldsymbol{\Gamma}_1(t) \cdots \boldsymbol{\Gamma}_d(t)\right]. \tag{3.59}$$

Subtracting off the tangential component $\partial \left[\boldsymbol{\Psi}_1(t) \cdots \boldsymbol{\Psi}_d(t)\right] / \partial t$ we have the normal component at time t

$$N_{u_{\boldsymbol{r}}(\boldsymbol{x},t)} \left(\frac{\partial u_{\boldsymbol{r}}(\boldsymbol{x}, t)}{\partial t} \right) = \frac{\partial}{\partial t} \left[\boldsymbol{\Gamma}_1(t) \cdots \boldsymbol{\Gamma}_d(t)\right]. \tag{3.60}$$

Next, orthogonalize the partial product $\boldsymbol{\Gamma}_{\leq i-1}(t)$ from the left and the partial product $\boldsymbol{\Gamma}_{\geq i}(t)$ from the right to obtain

$$N_{u_{\boldsymbol{r}}(\boldsymbol{x},t)} \left(\frac{\partial u_{\boldsymbol{r}}(\boldsymbol{x}, t)}{\partial t} \right) = \frac{\partial}{\partial t} \left[\boldsymbol{\Gamma}_1(t) \cdots \boldsymbol{\Gamma}_{i-1}(t) \boldsymbol{C}_i(t) \boldsymbol{\Gamma}_i^{\top}(t) \cdots \boldsymbol{\Gamma}_d^{\top}(t)\right], \tag{3.61}$$

where $\boldsymbol{C}_i = \boldsymbol{0}_{r_{i-1}\times r_i}$ and $\left\langle \boldsymbol{\Gamma}_i^\top \boldsymbol{\Gamma}_i \right\rangle_i = \boldsymbol{I}$ for all $i = 1, 2, \ldots, d$. Expand (3.61) using a product rule and evaluate at $t = t_i$

$$\left[N_{u_{\boldsymbol{r}}(\boldsymbol{x},t)} \left(\frac{\partial u_{\boldsymbol{r}}(\boldsymbol{x},t)}{\partial t} \right) \right]_{t=t_i} = \boldsymbol{\Gamma}_1(t_i) \cdots \boldsymbol{\Gamma}_{j-1}(t_i) \left.\frac{\partial \boldsymbol{C}_j(t)}{\partial t}\right|_{t=t_i} \boldsymbol{\Gamma}_j(t_i) \cdots \boldsymbol{\Gamma}_d(t_i). \tag{3.62}$$

From the previous equation we see that the FTT autocorrelation matrices of the normal component at time instant t_i are the time derivatives of the zero energy modes in the current solution. Thus, if the normal component has FTT rank $\boldsymbol{n}$ then the solution $u_{\boldsymbol{r}}(\boldsymbol{x}, t)$ at time t_i should be represented by an FTT tensor of rank $\boldsymbol{r} + \boldsymbol{n}$. Certainly, the solution will be over represented at t_i with rank $\boldsymbol{r} + \boldsymbol{n}$. However, after one step of the splitting integrator the additional ranks will ensure that the low-rank solution $u_{\boldsymbol{r}+\boldsymbol{n}}(\boldsymbol{x}, t) \in \mathcal{M}_{\boldsymbol{r}+\boldsymbol{n}}$ retains its accuracy.

The main steps of the algorithm we propose to adaptively increase the tensor rank are summarized in Algorithm 1. The operation "$*$" appearing within the conditional statement if/end denotes scalar times FTT tensor, and is meant to indicate that the multiplication is done by scaling the first core of the tensor with the scalar 0 and leaving the remainder of the cores unchanged [82]. As we will demonstrate in section 3.6, Algorithm 1 is robust and it yields accurate results that do no require ad-hoc approximations such the matrix pseudo-inverse approximation introduced in [6].

3.5.3 Order of the rank-adaptive scheme

Let us choose the threshold ϵ_{inc} in (3.52) to satisfy

$$\epsilon_{\text{inc}} \leq K_2 \Delta t, \tag{3.63}$$

Algorithm 1: One step integration with adaptive rank increase.

```
Input:
$u_{\boldsymbol{r}}(\boldsymbol{x}, t_i), \dots, u_{\boldsymbol{r}}(\boldsymbol{x}, t_{i-p}) \rightarrow$ time snapshots of the PDE solution with rank $\boldsymbol{r}$,
$G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i)) \rightarrow$ velocity defined PDE (1.1) by at time $t_i$,
$\Delta t \rightarrow$ time step,
$\epsilon_{\text{inc}} \rightarrow$ threshold for the norm of normal component $N_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)}(G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i)))$.

Output:
$u_{\boldsymbol{r}+\boldsymbol{n}}(\boldsymbol{x}, t_{i+1}) \rightarrow$ PDE solution with rank $\boldsymbol{r}+\boldsymbol{n}$ at time $t_{i+1}$

Initialization:
    • Approximate the constant rank velocity vector via the BDF formula:
      $\widetilde{P}^{(p)}_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)} G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i)) = BD_p(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i), u_{\boldsymbol{r}}(\boldsymbol{x}, t_{i-1}), \dots, u_{\boldsymbol{r}}(\boldsymbol{x}, t_{i-p}))$
    • Compute the normal component:
      $N_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)} G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i)) = G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i)) - \widetilde{P}^{(p)}_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)} G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i))$

Runtime:
    • if $\| N_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)} G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i)) \| > \epsilon_{\text{inc}}$ then
        Compute the FTT decomposition of normal component:
        $N_{\text{TT}}(\boldsymbol{x}, t_i) = \text{FTT}(N_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)} G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i)))$
        Initialize additional tensor modes in $u_{\boldsymbol{r}}(\boldsymbol{x}, t_i)$, as many as the rank of $N_{\text{TT}}$ (say $\boldsymbol{n}$):
        $u_{\boldsymbol{r}+\boldsymbol{n}}(\boldsymbol{x}, t_i) = u_{\boldsymbol{r}}(\boldsymbol{x}, t_i) + 0 * N_{\text{TT}}(\boldsymbol{x}, t_i)$
      end
    • Use one step of splitting integrator to map $u_{\boldsymbol{r}+\boldsymbol{n}}(\boldsymbol{x}, t_i)$ into $u_{\boldsymbol{r}+\boldsymbol{n}}(\boldsymbol{x}, t_{i+1})$
```

and assume that the condition

$$\| N_{u_{\boldsymbol{r}}(\boldsymbol{x},t)}(G(u_{\boldsymbol{r}}(\boldsymbol{x}, t))) \| \le \epsilon_{\text{inc}} \tag{3.64}$$

is satisfied for all $t \in [0, T]$. Then we have the following bound for the local truncation error

$$\begin{aligned} \|E(t_i, t_{i+1})\| &= \left\| \int_{t_i}^{t_{i+1}} N_{u_{\boldsymbol{r}}(\boldsymbol{x},\tau)}(G(u_{\boldsymbol{r}}(\boldsymbol{x}, \tau)))d\tau \right\| \\ &\leq \int_{t_i}^{t_{i+1}} \|N_{u_{\boldsymbol{r}}(\boldsymbol{x},\tau)}(G(u_{\boldsymbol{r}}(\boldsymbol{x}, \tau)))\| d\tau \\ &\leq \int_{t_i}^{t_{i+1}} K_2 \Delta t d\tau \\ &= K_2 \Delta t^2. \end{aligned} \tag{3.65}$$

In particular, we have that the continuous-time rank-adaptive scheme is order one consistent in Δt if the normal vector threshold is set as in (3.63).

When implementing the adaptive scheme we usually discretize the time domain $[0, T]$ into a mesh of time instants as in (3.29). Therefore, we do not necessarily have control over the normal vector for all $t \in [0, T]$ but rather only at a finite number of time instants. However, an analogous argument as we have made for order one consistency in the continuous time rank-adaptive scheme holds for the discrete time rank-adaptive scheme by considering the first-order approximation of the local truncation error given in (3.50). In particular by using the equality in (3.50) and discrete time thresholding of the normal component

$$\|N_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)}(G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i)))\| \leq \epsilon_{\text{inc}}, \qquad \forall i = 0, 1, \dots, N, \tag{3.66}$$

we have that

$$\begin{aligned} \|E(t_i, t_{i+1})\| &= \left\| \Delta t N_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)}(G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i))) + O(\Delta t^2) \right\| \\ &\leq \left\| \Delta t N_{u_{\boldsymbol{r}}(\boldsymbol{x},t_i)}(G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i))) \right\| + \|O(\Delta t^2)\| \\ &= K_2 \Delta t^2 + O(\Delta t^2) \\ &= O(\Delta t^2). \end{aligned} \tag{3.67}$$

This proves that the discrete time rank-adaptive scheme with normal threshold given by (3.66) is consistent with order one in Δt. Higher-order consistency results can be obtained with higher-order time integration methods and higher-order estimators for the normal vector $N_{u_{\boldsymbol{r}}} G(u_{\boldsymbol{r}})$.

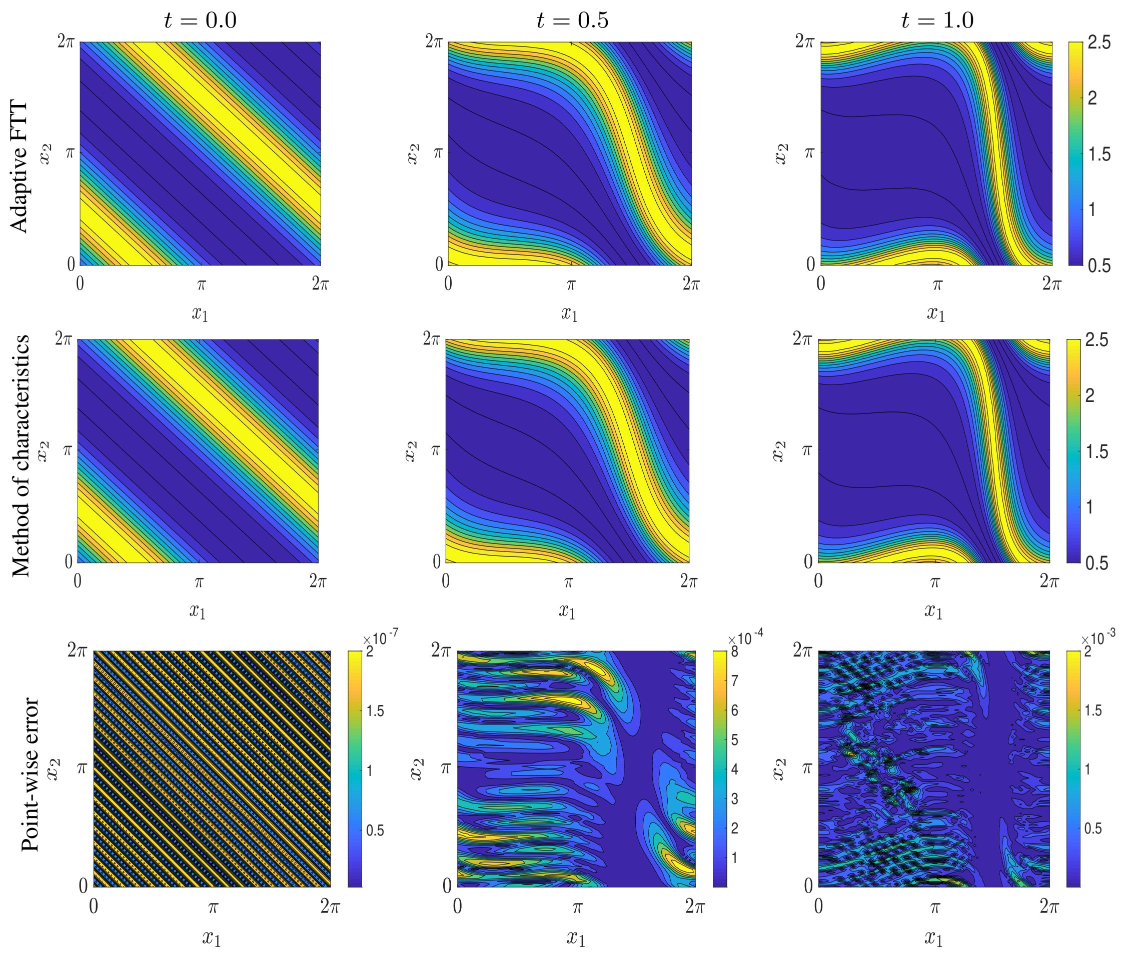


Figure 3.2: Variable coefficient advection equation (3.68). Time snapshots of the rank-adaptive FTT solution $u_{\boldsymbol{r}}(x_1, x_2, t)$ obtained with threshold $\epsilon_{\text{inc}} = 10^{-2}$ (top), the semi-analytical solution $u_{\text{ref}}(x_1, x_2, t)$ (middle), and the point-wise error between the two solutions (bottom).

3.6 Numerical examples

In this section we demonstrate the rank-adaptive FTT tensor method on linear and nonlinear PDEs. In all examples the rank-adaptive scheme relies on first-order Lie-Trotter operator splitting time integration (3.32), and the thresholding criterion (3.52). For each PDE we rigorously assess the accuracy of the proposed rank-adaptive tensor method by comparing it with benchmark solutions computed with well-established numerical methods.

3.6.1 Two-dimensional advection equation

Let us begin with the two-dimensional variable coefficient advection problem

$$\begin{cases} \dfrac{\partial u(x_1,x_2,t)}{\partial t} = (\sin(x_1)+\cos(x_2))\dfrac{\partial u(x_1,x_2,t)}{\partial x_1} + \cos(x_2)\dfrac{\partial u(x_1,x_2,t)}{\partial x_2}, \\ \\ u(x_1,x_2,0) = \exp[\sin(x_1+x_2)], \end{cases} \tag{3.68}$$

on the torus $\Omega = \mathbb{T}^2$. We have shown in previous work [29] that the tensor solution to the PDE (3.68) increases in rank as time increases. As is well known, the PDE (3.68) can be reduced to the trivial ODE $du/dt = 0$ along the flow generated by the dynamical system (see, e.g., [90])

$$\begin{cases} \dfrac{dx_1}{dt} = \sin(x_1)+\cos(x_2), \\ \dfrac{dx_2}{dt} = \cos(x_2). \end{cases} \tag{3.69}$$

With the flow $\{x_1(t,x_{01},x_{02}), x_1(t,x_{01},x_{02})\}$ available, we can write the analytical solution to (3.68) as

$$u_{\text{ref}}(x_1,x_2,t) = \exp\left[\sin(x_{01}(x_1,x_2,t)+x_{02}(x_1,x_2,t))\right], \tag{3.70}$$

where $\{x_{01}(x_1,x_2,t), x_{02}(x_1,x_2,t)\}$ denotes the inverse flow generated by (3.69). We obtain a semi-analytical solution to the PDE (3.68) by solving the characteristic system (3.69) numer-

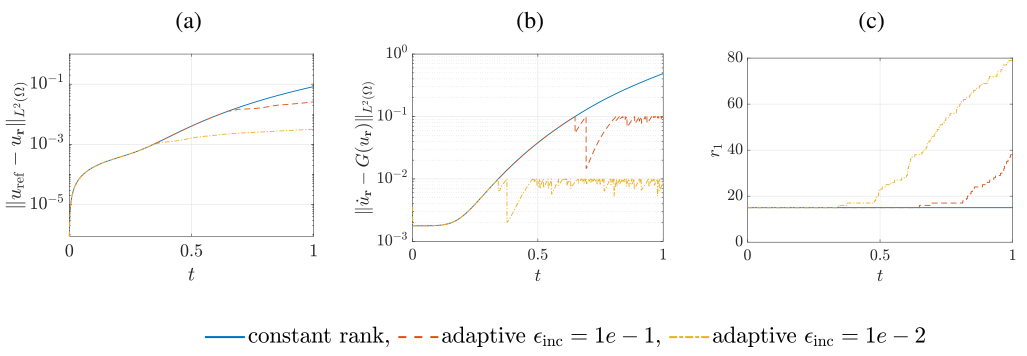


Figure 3.3: (a) Global $L^2(\Omega)$ error of the FTT solution $u_{\boldsymbol{r}}$ relative to the benchmark solution u_{ref}; (b) Norm of the two-point BDF approximation to the normal component $N_{u_{\boldsymbol{r}}} G(u_{\boldsymbol{r}}(\boldsymbol{x}, t))$ (note the effect of thresholding); (c) Tensor rank versus time of the constant-rank FTT solution and adaptive rank solutions with $\epsilon_{\text{inc}} = 10^{-1}$ and $\epsilon_{\text{inc}} = 10^{-2}$.

ically for different initial conditions and then evaluating (3.70). A few time snapshots of the semi-analytical solution (3.70) are plotted in Figure 3.2 (middle row).

We also solve the PDE (3.68) using the proposed rank-adaptive tensor method with first-order Lie-Trotter operator splitting and thresholding criterion (3.52) with $\epsilon_{\text{inc}} = 10^{-2}$. The initial condition is approximated by an FTT tensor $u_{\boldsymbol{r}}(x_1, x_2, 0)$ with multivariate rank $\boldsymbol{r} = \begin{bmatrix} 1 & 15 & 1 \end{bmatrix}$

$$u_{\boldsymbol{r}}(x_1, x_2, 0) = \boldsymbol{\Psi}_1(x_1)\sqrt{\boldsymbol{\Lambda}}\boldsymbol{\Psi}_2(x_2), \tag{3.71}$$

where

$$\boldsymbol{\Psi}_1(x_1) = \begin{bmatrix} \psi_1(1;x_1;1) & \cdots & \psi_1(1;x_1;15) \end{bmatrix}, \quad \sqrt{\boldsymbol{\Lambda}} = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_{15} \end{bmatrix},$$
$$\boldsymbol{\Psi}_2(x_2) = \begin{bmatrix} \psi_2(1;x_2;1) \\ \vdots \\ \psi_2(15;x_2;1) \end{bmatrix}. \tag{3.72}$$

Each tensor mode ψ_i is discretized on a grid of 81 evenly-spaced points in the interval $\Omega_i = [0, 2\pi]$. One-dimensional Fourier pseudo-spectral quadrature rules and differentiation matrices [45] are used to compute inner products and derivatives when needed. We run three simulations with the initial tensor decomposition (3.71) and time step $\Delta t = 10^{-4}$. In the first simulation we do not use any rank adaptation, in the second simulation we set the normal vector threshold to $\epsilon_{\text{inc}} = 10^{-1}$ and in the third simulation we set $\epsilon_{\text{inc}} = 10^{-2}$. At each time step the component of $G(u_{\boldsymbol{r}}(\boldsymbol{x}, t_i))$ normal to the tensor manifold is approximated with the two-point BDF formula (section 3.5.2). In Figure 3.4 we plot a few time snapshots of the singular values of the rank-adaptive FTT solution with $\epsilon_{\text{inc}} = 10^{-2}$. Figures 3.3(a)-(c) summarize the performance and accuracy of the proposed rank-adaptive FTT solver. In particular, in Figure 3.3(a) we plot the time-dependent $L^2(\Omega)$ error between the rank-adaptive FTT solution and the reference solution we obtained with method of characteristics. It is seen that decreasing the threshold ϵ_{inc} on the norm of the component of $G(u_{\boldsymbol{r}})$ normal to the FTT tensor manifold (Figure 3.3(b)) yields addition of more tensor cores to the FTT solution (Figure 3.3(c)). This, in turn, results in better accuracy as demonstrated in Figure 3.3(a).

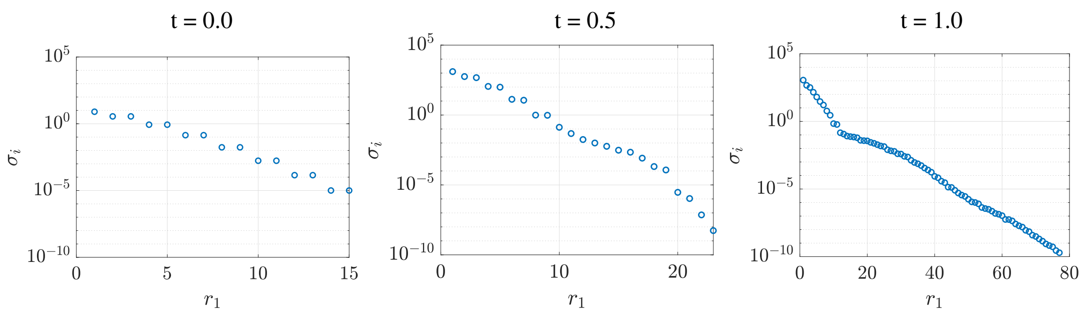


Figure 3.4: Time snapshots of the singular values of the rank-adaptive FTT solution with threshold $\epsilon_{\text{inc}} = 10^{-2}$.

3.6.2 Two-dimensional Kuramoto-Sivashinsky equation

Next we demonstrate the rank-adaptive FTT integrator on the two-dimensional Kuramoto-Sivashinsky equation [53]

$$\begin{cases} \dfrac{\partial}{\partial t}u(x_1,x_2,t) + \dfrac{1}{2}|\nabla_\nu u(x_1,x_2,t)|^2 + \Delta_\nu u(x_1,x_2,t) + \nu_1 \Delta_\nu^2 u(x_1,x_2,t) = 0, \\ u(x_1,x_2,0) = \sin(x_1+x_2) + \sin(x_1) + \sin(x_2), \end{cases} \tag{3.73}$$

where

$$\nabla_\nu = \left(\frac{\partial}{\partial x_1}, \nu\frac{\partial}{\partial x_2}\right), \qquad \Delta_\nu = \frac{\partial^2}{\partial x_1^2} + \nu\frac{\partial^2}{\partial x_2^2}. \tag{3.74}$$

Here, ν_1, ν_2 are bifurcation parameters and $\nu = \nu_2/\nu_1$. For our demonstration we set $\nu_1 = 0.25$, $\nu_2 = 0.04$ and solve (3.73) on the two-dimensional torus $\mathbb{T}^2$. The initial condition can be written as rank $\boldsymbol{r} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$ FTT tensor

$$u_0(x_1,x_2) = \psi_1(1;x_1;1)\psi_2(1;x_2;1)\sqrt{\lambda(1)} + \psi_1(1;x_1;2)\psi_2(2;x_2;1)\sqrt{\lambda(2)}, \tag{3.75}$$

where

$$\begin{aligned} \psi_1(1;x_1;1) &= \frac{\sin(x_1)}{\sqrt{\pi}}, & \psi_1(1;x_1;2) &= \frac{\cos(x_1)+1}{\sqrt{3\pi}}, \\ \psi_2(1;x_2;1) &= \frac{\cos(x_2)+1}{\sqrt{3\pi}}, & \psi_2(2;x_2;1) &= \frac{\sin(x_2)}{\sqrt{\pi}}, \end{aligned} \tag{3.76}$$

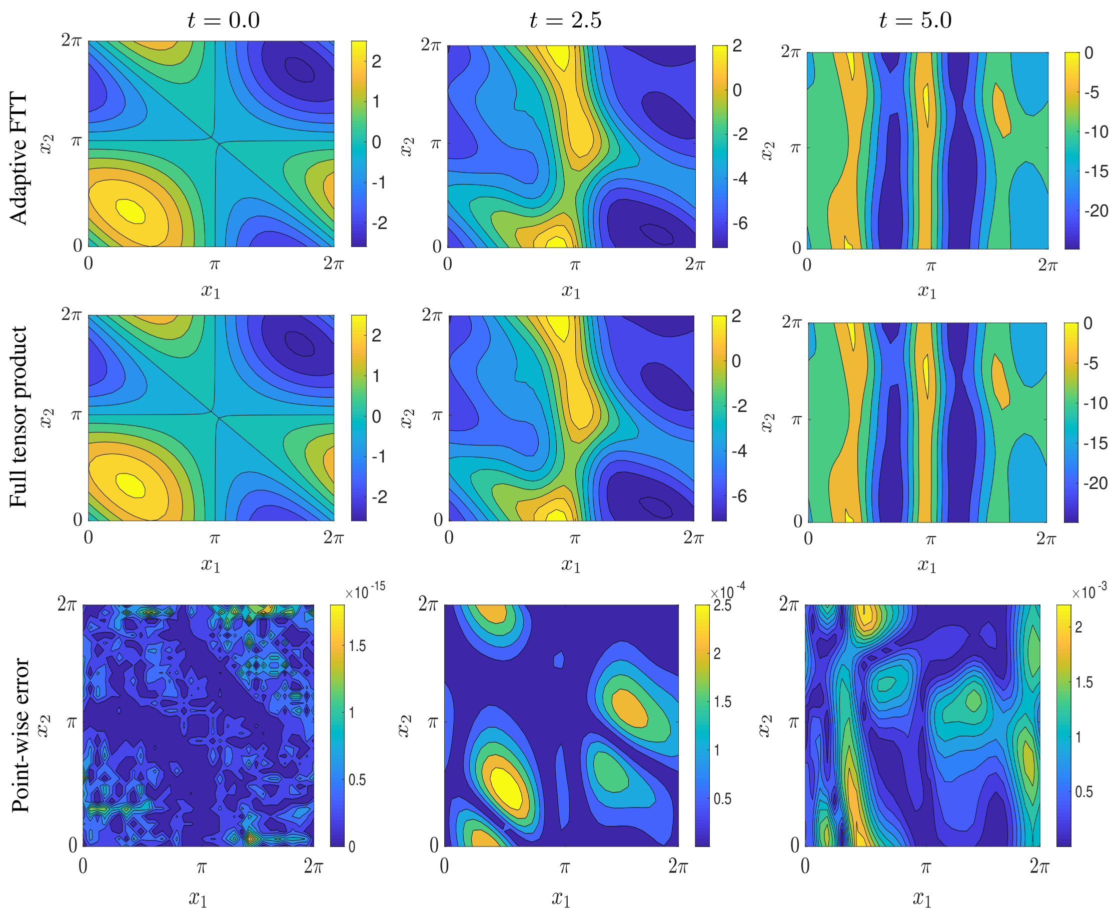


Figure 3.5: Kuramoto-Sivashinsky equation (3.73). Time snapshots of the rank-adaptive FTT solution $u_{\boldsymbol{r}}(x_1, x_2, t)$ obtained with threshold $\epsilon_{\text{inc}} = 10^{-2}$ (top), the Fourier pseudo-spectral solution $u_{\text{ref}}(x_1, x_2, t)$ (middle), and their point-wise error between the two solutions (bottom).

and

$$\sqrt{\lambda(1)} = \sqrt{\lambda(2)} = \sqrt{3}\pi. \tag{3.77}$$

We compute a benchmark solution by using a Fourier pseudo-spectral method [45] with 33 evenly-spaced grid points per spatial dimension (1089 total number of points). Derivatives and integrals are approximated with well-known pseudo-spectral differentiation matrices and Gauss quadrature rules. The resulting ODE system is integrated forward in time using an explicit fourth-order Runge-Kutta method with time step $\Delta t = 10^{-5}$.

As before, we performed multiple simulations using the proposed rank-adaptive FTT algorithm with different thresholds for the component of $G(u_{\boldsymbol{r}})$ normal to the tensor manifold. Specifically, we ran one simulation with no mode addition and three simulations with adaptive mode addition based on Algorithm 1, and thresholds set to $\epsilon_{\text{inc}} = 10$, $\epsilon_{\text{inc}} = 10^{-1}$, and $\epsilon_{\text{inc}} = 10^{-2}$. We used the two-point BDF formula (3.54) to approximate the component of the solution normal to the tensor manifold at each time step and the Lie-Trotter operator splitting scheme (3.32) with time step $\Delta t = 10^{-5}$ to integrate in time the rank-adaptive FTT solution. In Figure 3.5 we compare the time snapshots of the rank-adaptive FTT solution with $\epsilon_{\text{inc}} = 10^{-2}$ with the benchmark solution obtained by the Fourier pseudo-spectral method. As before, Figures 3.6(a)-(c) demonstrate that the rank-adaptive FTT algorithm is effective in controlling the $L^2(\Omega)$ error of the FTT solution. Interestingly, the solution to the PDE (3.73) has the property that any tensor approximation with sufficient rank yields a normal component that does not grow in time. In fact, as seen in Figure 3.6(b) the tensor rank becomes constant for each threshold ϵ_{inc} after a transient of approximately 0.5 dimensionless time units. In Figure 3.6 we observe that the error associated with the constant rank 2 FTT solution increases significantly during temporal integration. This suggests that projecting the nonlinear Kuramoto-Sivashinsky equation (3.73) onto a rank 2 FTT manifold yields a reduced-order PDE which does not accurately capture the dynamics of the full system. A similar phenomenon occurs in other areas of reduced-order modeling, e.g., when projecting nonlinear PDEs onto proper orthogonal decomposition (POD) bases [103].

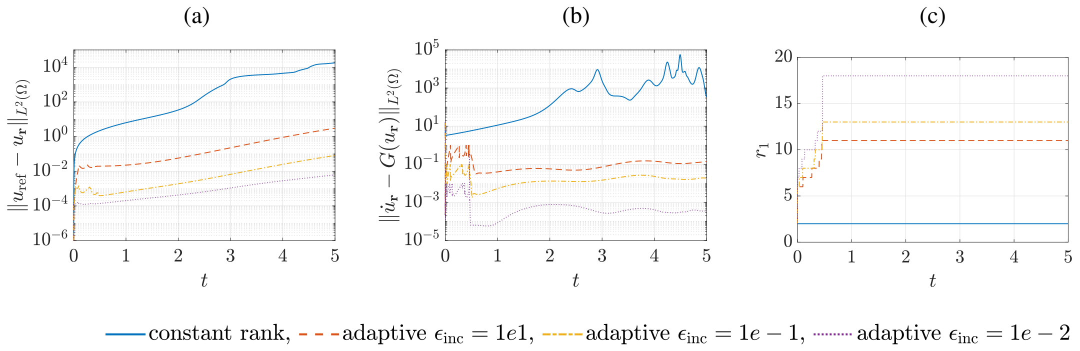


Figure 3.6: (a) Global $L^2(\Omega)$ error between the FTT solution $u_{\boldsymbol{r}}$ to equation (3.73) and the benchmark solution u_{ref}. (b) Norm of the approximation to $N_{u_{\boldsymbol{r}}}(G(u_{\boldsymbol{r}})) = \dot{u}_{\boldsymbol{r}} - G(u_{\boldsymbol{r}})$ where the tangent space projection is computed with a two-point BDF formula at each time. (c) Rank versus time of the constant rank FTT solution and rank-adaptive FTT solutions with $\epsilon_{\text{inc}} = 10, 10^{-1}, 10^{-2}$.

3.6.3 Four-dimensional Fokker-Planck equation

Finally, we demonstrate the proposed rank-adaptive FTT integrator on a four-dimensional Fokker–Planck equation with non-constant drift and diffusion coefficients. As is well known [91], the Fokker–Planck equation describes the evolution of the probability density function (PDF) of the state vector solving the Itô stochastic differential equation (SDE)

$$d\boldsymbol{X}_t = \boldsymbol{\mu}(\boldsymbol{X}_t, t)dt + \boldsymbol{\sigma}(\boldsymbol{X}_t, t)d\boldsymbol{W}_t. \tag{3.78}$$

Here, $\boldsymbol{X}_t$ is the d-dimensional state vector, $\boldsymbol{\mu}(\boldsymbol{X}_t, t)$ is the d-dimensional drift, $\boldsymbol{\sigma}(\boldsymbol{X}_t, t)$ is an $d \times m$ matrix and $\boldsymbol{W}_t$ is an m-dimensional standard Wiener process. The Fokker–Planck equation that corresponds to (3.78) has the form

$$\begin{cases} \dfrac{\partial p(\boldsymbol{x}, t)}{\partial t} = \mathcal{L}(\boldsymbol{x}, t)p(\boldsymbol{x}, t), \\ \\ p(\boldsymbol{x}, 0) = p_0(\boldsymbol{x}), \end{cases} \tag{3.79}$$

where $p_0(\boldsymbol{x})$ is the PDF of the initial state $\boldsymbol{X}_0$, $\mathcal{L}$ is a second-order linear differential operator defined as

$$\mathcal{L}(\boldsymbol{x},t)p(\boldsymbol{x},t) = -\sum_{k=1}^{d} \frac{\partial}{\partial x_k}\left(\mu_k(x,t)p(\boldsymbol{x},t)\right) + \sum_{k,j=1}^{d} \frac{\partial^2}{\partial x_k \partial x_j}\left(D_{ij}(\boldsymbol{x},t)p(\boldsymbol{x},t)\right), \tag{3.80}$$

and $\boldsymbol{D}(\boldsymbol{x},t) = \boldsymbol{\sigma}(\boldsymbol{x},t)\boldsymbol{\sigma}(\boldsymbol{x},t)^\top/2$ is the diffusion tensor. For our numerical demonstration we set

$$\boldsymbol{\mu}(\boldsymbol{x}) = \alpha \begin{bmatrix} \sin(x_1) \\ \sin(x_3) \\ \sin(x_4) \\ \sin(x_1) \end{bmatrix}, \qquad \boldsymbol{\sigma}(\boldsymbol{x}) = \sqrt{2\beta} \begin{bmatrix} g(x_2) & 0 & 0 & 0 \\ 0 & g(x_3) & 0 & 0 \\ 0 & 0 & g(x_4) & 0 \\ 0 & 0 & 0 & g(x_1) \end{bmatrix}, \tag{3.81}$$

where $g(x) = \sqrt{1 + k\sin(x)}$. With the drift and diffusion matrices chosen in (3.81) the operator (3.80) takes the form

$$\begin{aligned} \mathcal{L} = &- \alpha \left(\cos(x_1) + \sin(x_1)\frac{\partial}{\partial x_1} + \sin(x_3)\frac{\partial}{\partial x_2} + \sin(x_4)\frac{\partial}{\partial x_3} + \sin(x_1)\frac{\partial}{\partial x_4}\right) \\ &+ \beta\Bigg((1 + k\sin(x_2))\frac{\partial^2}{\partial x_1^2} + (1 + k\sin(x_3))\frac{\partial^2}{\partial x_2^2} \\ &+ (1 + k\sin(x_4))\frac{\partial^2}{\partial x_3^2} + (1 + k\sin(x_1))\frac{\partial^2}{\partial x_4^2}\Bigg). \end{aligned} \tag{3.82}$$

Clearly $\mathcal{L}$ is a linear, time-independent separable operator of rank 9, since it can be written as

$$\mathcal{L} = \sum_{i=1}^{9} L_i^{(1)} \otimes L_i^{(2)} \otimes L_i^{(3)} \otimes L_i^{(4)}, \tag{3.83}$$

where each $L_i^{(j)}$ operates on x_j only. Specifically, we have

$$
\begin{aligned}
&L_1^{(1)} = -\alpha\cos(x_1), && L_2^{(1)} = -\alpha\sin(x_1)\frac{\partial}{\partial x_1}, && L_3^{(2)} = -\alpha\frac{\partial}{\partial x_2}, && L_3^{(3)} = \sin(x_3),\\
&L_4^{(3)} = -\alpha\frac{\partial}{\partial x_3}, && L_4^{(4)} = \sin(x_4), && L_5^{(1)} = -\alpha\sin(x_1), && L_5^{(4)} = \frac{\partial}{\partial x_4},\\
&L_6^{(1)} = \beta\frac{\partial^2}{\partial x_1^2}, && L_6^{(2)} = 1 + k\sin(x_2), && L_7^{(2)} = \beta\frac{\partial^2}{\partial x_2^2}, && L_7^{(3)} = 1 + k\sin(x_3),\\
&L_8^{(3)} = \beta\frac{\partial^2}{\partial x_3^2}, && L_8^{(2)} = 1 + k\sin(x_4), && L_9^{(4)} = \beta\frac{\partial^2}{\partial x_4^2}, && L_9^{(1)} = 1 + k\sin(x_1),
\end{aligned}
\tag{3.84}
$$

and all other unspecified $L_i^{(j)}$ are identity operators. We set the parameters in (3.81) as $\alpha = 0.1$, $\beta = 2.0$, $k = 1.0$ and solve (3.79) on the four-dimensional torus $\mathbb{T}^4$. The initial PDF is set as

$$
p_0(\boldsymbol{x}) = \frac{\sin(x_1)\sin(x_2)\sin(x_3)\sin(x_4) + 1}{16\pi^4}. \tag{3.85}
$$

Note that (3.85) is a four-dimensional FTT tensor with multilinear rank $\boldsymbol{r} = \begin{bmatrix} 1 & 2 & 2 & 2 & 1 \end{bmatrix}$. Upon normalizing the modes appropriately we obtain the left orthogonalized initial condition required to begin integration

$$
\begin{aligned}
p_0(\boldsymbol{x}) = {} & \psi_1(1; x_1; 1)\psi_2(1; x_2; 1)\psi_3(1; x_3; 1)\psi_4(1; x_4; 1)\sqrt{\lambda(1)}\\
& + \psi_1(1; x_1; 2)\psi_2(2; x_2; 2)\psi_3(2; x_3; 2)\psi_4(2; x_4; 1)\sqrt{\lambda(2)},
\end{aligned}
\tag{3.86}
$$

where

$$
\psi_i(1; x_i; 1) = \frac{\sin(x_i)}{\sqrt{\pi}}, \qquad \sqrt{\lambda(1)} = \frac{1}{16\pi^2}. \tag{3.87}
$$

All other tensor modes are equal to $1/\sqrt{2\pi}$, and $\sqrt{\lambda(2)} = 1/(2\pi^2)$. To obtain a benchmark solution with which to compare the rank-adaptive FTT solution, we solve the PDE (3.79) using a Fourier pseudo-spectral method on the torus $\mathbb{T}^4$ with $21^4 = 194481$ evenly-spaced points. As before, the operator $\mathcal{L}$ is represented in terms of pseudo-spectral differentiation matrices

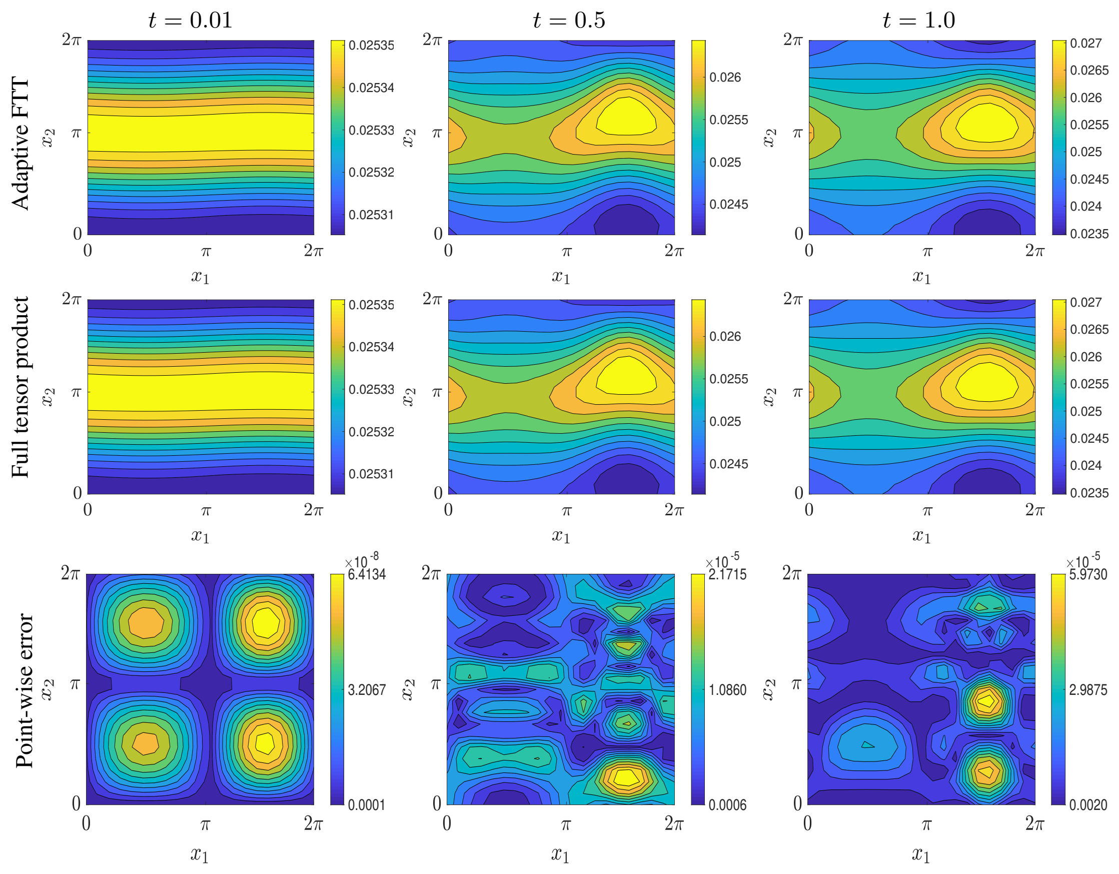


Figure 3.7: Time snapshots of marginal PDF $p_{\boldsymbol{r}}(x_1, x_2, t)$ corresponding to the solution to the Fokker-Planck equation (3.79). We plot marginals computed with the rank-adaptive FTT integrator using $\epsilon_{\text{inc}} = 10^{-4}$ (top row) and with the full tensor product Fourier pseudo-spectral method (middle row). We also plot the point-wise error between the two numerical solutions (bottom row). The initial condition is the FTT tensor (3.85).

[45], and the resulting semi-discrete approximation (ODE system) is integrated with an explicit fourth-order Runge Kutta method using time step $\Delta t = 10^{-4}$. The numerical solution we obtained in this way is denoted by $p_{\text{ref}}(\boldsymbol{x}, t)$. We also solve the Fokker-Planck using the proposed rank-adaptive FTT method with first-order Lie-Trotter time integrator (section 3.2.1) and normal vector thresholding (section 3.5.2). We run three simulations all with time step $\Delta t = 10^{-4}$: one with no rank adaption, and two with rank-adaptation and normal component

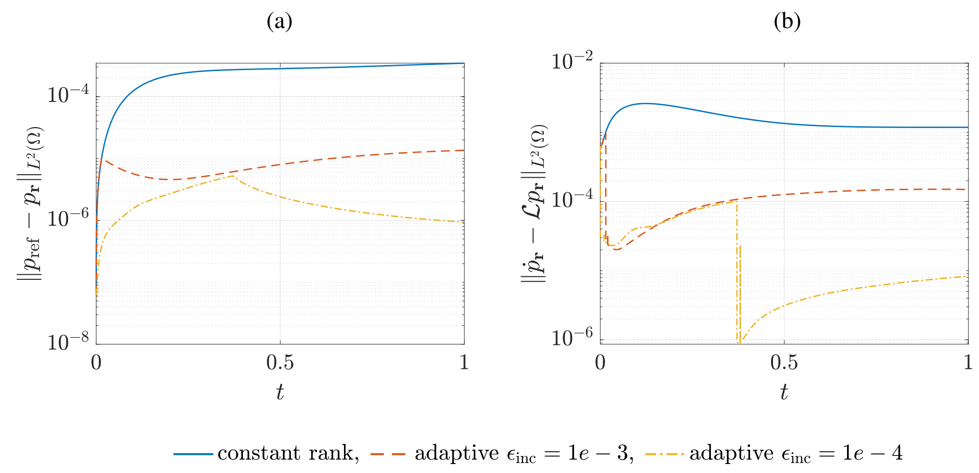


Figure 3.8: (a) The $L^2(\Omega)$ error of the FTT solution $p_{\boldsymbol{r}}(\boldsymbol{x}, t)$ relative to the benchmark solution $p_{\text{ref}}(\boldsymbol{x}, t)$ computed with a Fourier pseudo-spectral method on a tensor product grid. (b) Norm of the component of $\mathcal{L}p_{\boldsymbol{r}}$ normal to the tensor manifold (see Figure 3.1). Such component is approximated a two-point BDF formula at each time step.

thresholds set to $\epsilon_{\text{inc}} = 10^{-3}$ and $\epsilon_{\text{inc}} = 10^{-4}$. In Figure 3.7 we plot three time snapshots of the two-dimensional solution marginal

$$p(x_1, x_2, t) = \int_0^{2\pi} \int_0^{2\pi} p(x_1, x_2, x_3, x_4, t) dx_3 dx_4 \tag{3.88}$$

computed with the rank-adaptive FTT integrator ($\epsilon_{\text{inc}} = 10^{-4}$) and the full tensor product pseudo-spectral method (reference solution). In Figure 3.8(a) we compare the $L^2(\Omega)$ errors of the rank-adaptive method relative to the reference solution. It is seen that as we decrease the threshold the solution becomes more accurate. In Figure 3.8(b) we plot the component of $\mathcal{L}p_{\boldsymbol{r}}$ normal to the tensor manifold, which is approximated using the two-point BDF formula (3.54). Note that in the rank-adaptive FTT solution with thresholds $\epsilon_{\text{inc}} = 10^{-3}$ and $\epsilon_{\text{inc}} = 10^{-4}$ the solver performs both mode addition as well as mode removal. This is documented in Figure 3.9. The abrupt change in rank observed in Figure 3.9(a)-(c) near time $t = 0.4$ corresponding to the rank-adaptive solution with threshold $\epsilon inc = 10^{-4}$ is due to the time step size Δt being

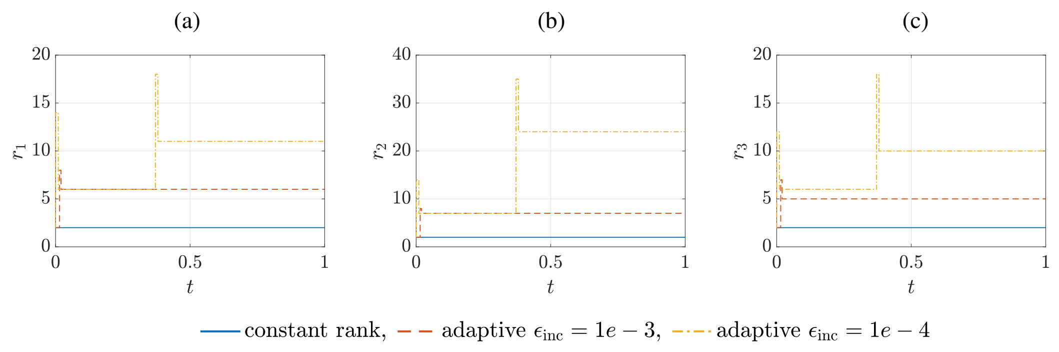


Figure 3.9: Tensor rank $\boldsymbol{r} = [1\, r_1\, r_2\, r_3\, 1]$ of adaptive FTT solution to the four dimensional Fokker-Planck equation (3.79).

equal to ϵ_{inc}. This can be justified as follows. Recall that the solution is first order accurate in Δt and therefore the approximation of the component of $\mathcal{L}p_{\boldsymbol{r}}$ normal to the tensor manifold $\mathcal{M}_{\boldsymbol{r}}$ is first-order accurate in Δt. If we set $\epsilon_{\text{inc}} \leq \Delta t$, then the rank-adaptive scheme may overestimate the number of modes needed to achieve accuracy on the order of Δt. This does not affect the accuracy of the numerical solution due to the robustness of the Lie-Trotter integrator to over-approximation [69]. Moreover we notice that the rank-adaptive scheme removes the unnecessary modes ensure that the tensor rank is not unnecessarily large (see section 3.5.1). In fact, the diffusive nature of the Fokker-Plank equation on the torus $\mathbb{T}^4$ yields relaxation to a statistical equilibrium state that depends on the drift and diffusion coefficients in (3.79). Such equilibrium state may be well-approximated by a low-rank FTT tensor.

Appendix 3.A Proof of Theorem 3.2.1

The functional $\mathcal{A}$ in (3.19) is convex and thus a critical point is necessarily a global minimum. To find such a critical point set the first variation of $\mathcal{A}$ with respect to $\partial\psi_j(\xi_{j-1}, \xi_j)/\partial t$

in the direction $\eta_j(\xi_{j-1}, \xi_j)$ ($j = 1, 2, \dots, d$, $\xi_j = 1, 2, \dots, r_j$)

$$\left[\frac{d}{d\epsilon}\mathcal{A}\left(\frac{\partial \psi_j(\xi_{j-1}, \xi_j)}{\partial t} + \epsilon \eta_j(\xi_{j-1}, \xi_j)\right)\right]_{\epsilon=0} \tag{3.89}$$

equal to zero for all $\eta_j(\xi_{j-1}, \xi) \in L^2_{\mu_j}(\Omega_j)$. Note that we have available the dynamic constraints

$$\left\langle \frac{\partial \psi_j(\cdot, \alpha_j)}{\partial t}, \psi_j(\cdot, \beta_j)\right\rangle_{L^2_{\tau\times\mu_j}(\mathbb{N}\times\Omega_j)} = 0, \qquad \forall j = 1, \dots, d-1, \quad \alpha_j, \beta_j = 1, \dots, r_j, \tag{3.90}$$

and the static constraints

$$\langle \psi_j(\cdot, \alpha_j), \psi_j(\cdot, \beta_j)\rangle_{L^2_{\tau\times\mu_j}(\mathbb{N}\times\Omega_j)} = \delta_{\alpha_j,\beta_j}, \qquad \forall j = 1, \dots, d-1, \quad \alpha_j, \beta_j = 1, \dots, r_j, \tag{3.91}$$

which are implied by the dynamic constraints as long as the cores $\boldsymbol{\Psi}_1(t), \dots, \boldsymbol{\Psi}_{d-1}(t)$ all have identity auto-correlation matrices at some time (say at $t = 0$). For $j = 1$ we obtain

$$\begin{aligned}
&\left[\delta_{\frac{\partial\psi_1(\xi_0,\xi_1)}{\partial t}}\mathcal{A}\right]\eta_1(\xi_0, \xi_1) \\
&= 2\left\langle \left(\frac{\partial}{\partial t}\left[\sum_{\alpha_0,\alpha_1=1}^{r_0,r_1}\psi_1(\alpha_0, \alpha_1)\psi_{>1}(\alpha_1)\right] - G(u_{\boldsymbol{r}})\right)\eta_1(\xi_0, \xi_1)\psi_{>1}(\xi_1)\right\rangle_{\leq d} \\
&\qquad\qquad + \sum_{\alpha_1=1}^{r_j}\lambda^{(1)}_{\xi_1\alpha_1}\left\langle \eta_1(\xi_0, \xi_1)\psi_1(\xi_0, \alpha_1)\right\rangle_1 \\
&= 0,
\end{aligned} \tag{3.92}$$

whence the fundamental lemma of calculus of variations implies

$$\begin{aligned}
&2\left\langle \left(\frac{\partial}{\partial t}\left[\sum_{\alpha_0,\alpha_1=1}^{r_0,r_1}\psi_1(\alpha_0, \alpha_1)\psi_{>1}(\alpha_1)\right] - G(u_{\boldsymbol{r}})\right)\psi_{>1}(\xi_1)\right\rangle_{>1} + \sum_{\alpha_1=1}^{r_j}\lambda^{(1)}_{\xi_1\alpha_1}\psi_1(\xi_0, \alpha_1) \\
&= 0.
\end{aligned} \tag{3.93}$$

Rearranging terms we obtain

$$\sum_{\alpha_0,\alpha_1=1}^{r_0,r_1}\left(\frac{\partial \psi_1(\alpha_0,\alpha_1)}{\partial t}\left\langle \psi_{>1}(\alpha_1),\psi_{>1}(\xi_1)\right\rangle_{2,\dots,d}+\psi_1(\alpha_0,\alpha_1)\left\langle \frac{\partial \psi_{>1}(\alpha_1)}{\partial t}\psi_{>1}(\xi_1)\right\rangle_{>1}\right)$$
$$=\left\langle G(u_{\boldsymbol{r}})\psi_{>1}(\xi_1)\right\rangle_{>1}-\frac{1}{2}\sum_{\alpha_1=1}^{r_j}\lambda^{(1)}_{\xi_1\alpha_1}\psi_1(\xi_0,\alpha_1). \tag{3.94}$$

Taking $\langle \cdot,\psi_1(\alpha_0,\xi_1')\rangle_{L^2_{\tau\times\mu_1}(\mathbb{N}\times\Omega_1)}$ of the previous equation and utilizing the dynamic and static constraints, we solve for the Lagrange multiplier

$$\lambda^{(1)}_{\xi_1\xi_1'}=\left\langle G(u_{\boldsymbol{r}})\psi_1(1,\xi_1')\psi_{>1}(\xi_1)\right\rangle_{\leq d}-\left\langle \frac{\partial \psi_{>1}(\xi_1')}{\partial t}\psi_{>1}(\xi_1)\right\rangle_{>1}. \tag{3.95}$$

Substituting (3.95) into (3.94) and rearranging terms we obtain

$$\begin{aligned}&\sum_{\alpha_1=1}^{r_1}\frac{\partial \psi_1(1,\alpha_1)}{\partial t}\left\langle \psi_{>1}(\alpha_1)\psi_{>1}(\xi_1)\right\rangle_{>1}\\&=\left\langle G(u_{\boldsymbol{r}})\psi_{>1}(\xi_1)\right\rangle_{>1}-\sum_{\alpha_1=1}^{r_1}\psi_1(1,\alpha_1)\left\langle G(u_{\boldsymbol{r}})\psi_1(1,\alpha_1)\psi_{>1}(\xi_1)\right\rangle_{\leq d}\end{aligned} \tag{3.96}$$

Using the matrix–vector notation for tensor cores and inverting the auto-correlation matrix on the left hand side yields the equation for $\partial\boldsymbol{\Psi}_1/\partial t$ in (3.20). For $j=2,\dots,d-1$ we have that

$$\begin{aligned}&\left[\delta_{\frac{\partial\psi_j(\xi_{j-1},\xi_j)}{\partial t}}\mathcal{A}\right]\eta_j(\xi_{j-1},\xi_j)\\&=2\Bigg\langle\left(\frac{\partial}{\partial t}\left[\sum_{\alpha_0,\dots,\alpha_j=1}^{r_0,\dots,r_j}\psi_1(\alpha_0,\alpha_1)\cdots\psi_j(\alpha_{j-1},\alpha_j)\psi_{>j}(\alpha_j)\right]-G(u_{\boldsymbol{r}})\right)\\&\qquad\sum_{\alpha_0,\dots,\alpha_{j-2}=1}^{r_0,\dots,r_{j-2}}\psi_1(\alpha_0,\alpha_1)\cdots\psi_{j-1}(\alpha_{j-2},\xi_{j-1})\eta_j(\xi_{j-1},\xi_j)\psi_{>j}(\xi_j)\Bigg\rangle_{\leq d}\\&\qquad+\sum_{\alpha_j=1}^{r_j}\lambda^{(j)}_{\xi_j\alpha_j}\left\langle\eta_j(\xi_{j-1},\xi_j)\psi_j(\xi_{j-1},\alpha_j)\right\rangle_j.\end{aligned} \tag{3.97}$$

Utilizing the fundamental lemma of calculus of variations and rearranging terms we obtain

$$\begin{aligned}
&\Bigg\langle \left(\sum_{\alpha_0,\ldots,\alpha_j=1}^{r_0,\ldots,r_j} \frac{\partial}{\partial t}\Big[\psi_1(\alpha_0,\alpha_1)\cdots\psi_j(\alpha_{j-1},\alpha_j)\psi_{>j}(\alpha_j)\Big]\right) \\
&\qquad\qquad \sum_{\alpha_0,\ldots,\alpha_{j-2}=1}^{r_0,\ldots,r_{j-2}} \psi_1(\alpha_0,\alpha_1)\cdots\psi_{j-1}(\alpha_{j-2},\xi_{j-1})\psi_{>j}(\xi_j)\Bigg\rangle_{\leq j-1,>j} \\
&= \Bigg\langle G(u_{\boldsymbol{r}}) \sum_{\alpha_0,\ldots,\alpha_{j-2}=1}^{r_0,\ldots,r_{j-2}} \psi_1(\alpha_0,\alpha_1)\cdots\psi_{j-1}(\alpha_{j-2},\xi_{j-1})\psi_{>j}(\xi_j)\Bigg\rangle_{\leq j-1,>j} \\
&\qquad\qquad -\frac{1}{2}\sum_{\alpha_j}\lambda^{(j)}_{\xi_j\alpha_j}\psi_j(\xi_{j-1},\alpha_j).
\end{aligned} \tag{3.98}$$

Utilizing the dynamic orthogonality condition (3.15) and the orthonormality for all t on the left hand side of (3.98) we obtain

$$\begin{aligned}
&\sum_{\alpha_j=1}^{r_j}\left(\frac{\psi_j(\xi_{j-1},\alpha_j)}{\partial t}\Big\langle \psi_{>j}(\alpha_j)\psi_{>j}(\xi_j)\Big\rangle_{>j} + \psi_j(\xi_{j-1},\alpha_j)\Big\langle\frac{\partial\psi_{>j}(\alpha_j)}{\partial t}\psi_{>j}(\xi_j)\Big\rangle_{>j}\right) \\
=&\Bigg\langle G(u_{\boldsymbol{r}}) \sum_{\alpha_0,\ldots,\alpha_{j-2}=1}^{r_0,\ldots,r_{j-2}} \psi_1(\alpha_0,\alpha_1)\cdots\psi_{j-1}(\alpha_{j-1},\xi_{j-1})\psi_{>j}(\xi_j)\Bigg\rangle_{\leq j-1,>j} \\
&-\frac{1}{2}\sum_{\alpha_j=1}^{r_j}\lambda^{(j)}_{\xi_j\alpha_j}\psi_j(\xi_{j-1},\alpha_j).
\end{aligned} \tag{3.99}$$

Taking $\langle\cdot,\psi_j(\alpha_{j-1},\xi_j')\rangle_{L^2_{\tau\times\mu_j}(\mathbb{N}\times\Omega_j)}$ of the previous equation and utilizing the constraints we find

$$\begin{aligned}
\lambda^{(j)}_{\xi_j\xi_j'} = 2\Bigg[&\sum_{\alpha_0,\ldots,\alpha_{j-1}=1}^{r_0,\ldots,r_{j-1}}\Big\langle G(u_{\boldsymbol{r}})\psi_1(\alpha_0,\alpha_1)\cdots\psi_{j-1}(\alpha_{j-2},\xi_{j-1})\psi_j(\alpha_{j-1},\xi_j')\psi_{>j}(\xi_j)\Big\rangle_{\leq d} \\
&-\Big\langle\frac{\partial\psi_{>j}(\xi_j')}{\partial t}\psi_{>j}(\xi_j)\Big\rangle_{>j}\Bigg]
\end{aligned} \tag{3.100}$$

Plugging (3.100) into (3.99) and simplifying we obtain

$$\begin{aligned}
&\sum_{\alpha_j=1}^{r_j} \frac{\partial \psi_j(\xi_{j-1},\alpha_j)}{\partial t} \Big\langle \psi_{>j}(\alpha_j)\psi_{>j}(\xi_j) \Big\rangle_{>j} \\
&= \sum_{\alpha_0,\ldots,\alpha_{j-2}=1}^{r_0,\ldots,r_{j-2}} \Big\langle G(u_{\boldsymbol{r}})\psi_1(\alpha_0,\alpha_1)\cdots\psi_{j-1}(\alpha_{j-2},\xi_{j-1})\psi_{>j}(\xi_j) \Big\rangle_{\leq j-1,>j} \\
&\quad - \sum_{\alpha_0,\ldots,\alpha_j=1}^{r_0,\ldots,r_j} \psi_j(\xi_{j-1},\alpha_j) \Big\langle G(u_{\boldsymbol{r}})\psi_1(\alpha_0,\alpha_1)\cdots\psi_{j-1}(\alpha_{j-2},\xi_{j-1})\psi_j(\alpha_{j-1},\alpha_j)\psi_{>j}(\xi_j) \Big\rangle_{\leq d}.
\end{aligned} \tag{3.101}$$

Using the matrix vector notation for tensor cores and inverting the auto-correlation matrix on the left hand side yields the equation for $\partial \boldsymbol{\Psi}_j/\partial t$ in (3.20). For $j = d$ we obtain

$$\begin{aligned}
&\left[\delta_{\frac{\partial \psi_d(\xi_0,\xi_1)}{\partial t}} \mathcal{A}\right] \eta_d(\xi_{d-1},\xi_d) \\
&= 2\Big\langle \left(\frac{\partial}{\partial t}\left[\sum_{\alpha_0,\ldots,\alpha_d=1}^{\boldsymbol{r}} \psi_1(\alpha_0,\alpha_1)\cdots\psi_d(\alpha_{d-1},\alpha_d) \right] - G(u_{\boldsymbol{r}}) \right) \\
&\quad \sum_{\alpha_0,\ldots,\alpha_{d-2}=1}^{r_0,\ldots,r_{d-1}} \psi_1(\alpha_0,\alpha_1)\cdots\psi_{d-1}(\alpha_{d-2},\xi_{d-1})\eta_d(\xi_{d-1},\xi_d) \Big\rangle_{\leq d},
\end{aligned} \tag{3.102}$$

whence the fundamental lemma of calculus of variations implies

$$\begin{aligned}
&\Big\langle \left(\frac{\partial}{\partial t}\left[\sum_{\alpha_0,\ldots,\alpha_d=1}^{\boldsymbol{r}} \psi_1(\alpha_0,\alpha_1)\cdots\psi_d(\alpha_{d-1},\alpha_d) \right] - G(u_{\boldsymbol{r}}) \right) \\
&\quad \sum_{\alpha_0,\ldots,\alpha_{d-2}=1}^{r_0,\ldots,r_{d-1}} \psi_1(\alpha_0,\alpha_1)\cdots\psi_{d-1}(\alpha_{d-2},\xi_{d-1}) \Big\rangle_{\leq d-1} = 0.
\end{aligned} \tag{3.103}$$

Rearranging terms we obtain

$$\begin{aligned}
&\Big\langle \frac{\partial}{\partial t}\left[\sum_{\alpha_0,\ldots,\alpha_d=1}^{\boldsymbol{r}} \psi_1(\alpha_0,\alpha_1)\cdots\psi_d(\alpha_{d-1},\alpha_d) \right] \\
&\quad \sum_{\alpha_0,\ldots,\alpha_{d-2}=1}^{r_0,\ldots,r_{d-2}} \psi_1(\alpha_0,\alpha_1)\cdots\psi_{d-1}(\alpha_{d-2},\xi_{d-1}) \Big\rangle_{\leq d-1} \\
&= \Big\langle G(u_{\boldsymbol{r}}) \sum_{\alpha_0,\ldots,\alpha_{d-2}=1}^{r_0,\ldots,r_{d-2}} \psi_1(\alpha_0,\alpha_1)\cdots\psi_{d-1}(\alpha_{d-2},\xi_{d-1}) \Big\rangle_{\leq d-1}.
\end{aligned} \tag{3.104}$$

Using the dynamic and static orthogonality constraints we obtain

$$\frac{\partial \psi_d(\xi_{d-1},1)}{\partial t} = \left\langle G(u_{\boldsymbol{r}}) \sum_{\alpha_0,\ldots,\alpha_{d-2}=1}^{r_0,\ldots,r_{d-2}} \psi_1(\alpha_0,\alpha_1)\cdots\psi_{d-1}(\alpha_{d-2},\xi_{d-1}) \right\rangle_{\leq d-1} \cdot \tag{3.105}$$

Writing this expression in matrix-vector notation the desired equation for $\partial \boldsymbol{\Psi}_d/\partial t$ is obtained.

Chapter 4

Rank reducing coordinate transformations

In this chapter we present an extension of low-rank tensor methods in which we approximate a given multivariate function with a low-rank tensor together with a coordinate transformation in order to obtain a more efficient low-rank representation. Given the fundamental importance of tensor rank in computations and its non-favorable scaling, rank reduction based on coordinate transformations can greatly increase the efficiency of high-dimensional tensor approximation algorithms. To describe the method, consider the scalar field $u(\boldsymbol{x})$, where $\boldsymbol{x} \in \Omega \subseteq \mathbb{R}^d$, $d > 1$. The idea is very simple: determine an invertible coordinate transformation $\boldsymbol{H} : \mathbb{R}^d \to \mathbb{R}^d$ so that the function

$$v(\boldsymbol{x}) = u(\boldsymbol{H}(\boldsymbol{x})) \tag{4.1}$$

has smaller tensor rank than $u(\boldsymbol{x})$. The concept of representing a function on a transformed coordinate system has proven to be a successful technique for a wide range of applications, e.g., [71, 108, 54]. Related ideas have been used in quantum man-body problems in which the discretization basis is rotated in order to reduce the rank of the solution representation [65].

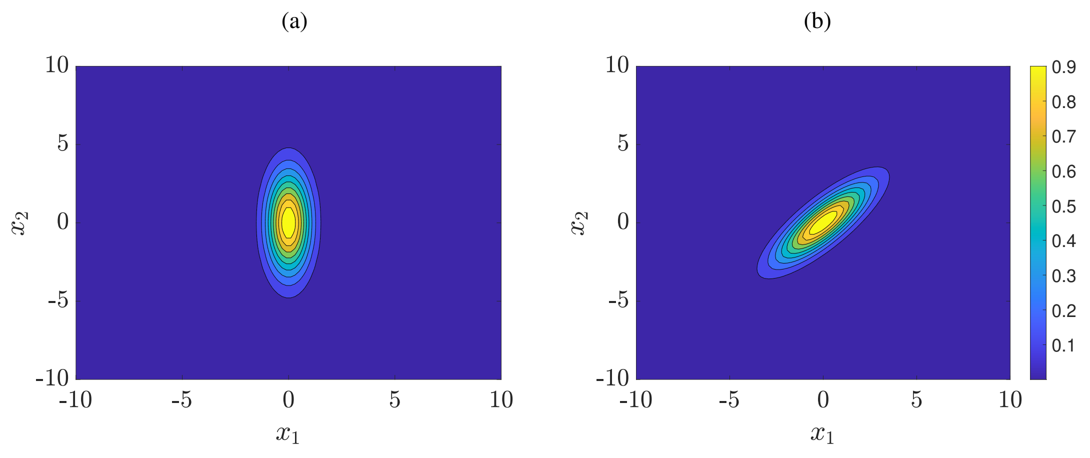
To illustrate the effects of coordinate transformations on tensor rank, in Figure 4.1 we show that a simple two-dimensional rotation can increase the rank of fully separated (i.e., rank one) Gaussian function significantly. Vice versa, the inverse rotation can transform a rotated Gaussian with high tensor rank into a rank one function. Similar results hold of course in higher dimensions.

Under mild assumptions on the function $u(\boldsymbol{x})$, one may argue, e.g. using nonlinear dynamics or the theory of optimal mass transport, that there always exists a transformation $\boldsymbol{H}$ such that $v(\boldsymbol{x}) = u(\boldsymbol{H}(\boldsymbol{x}))$ possesses the smallest possible multilinear rank among all tensors in a given format[1]. However, developing a computationally tractable algorithm for obtaining the transformation $\boldsymbol{H}$ given the function $u(\boldsymbol{x})$ is not an easy task. In this chapter we present a mathematical framework and computationally efficient algorithms for obtaining *quasi-optimal* tensor rank-reducing coordinate transformations $\boldsymbol{H}$. In particular, we restrict our analysis to linear coordinate transformations. In this setting, $\boldsymbol{H}$ can be represented by a matrix $\boldsymbol{A}$, which allows us to write (4.1) in the simplified form

$$v(\boldsymbol{x}) = u(\boldsymbol{A}\boldsymbol{x}). \tag{4.2}$$

The function $v(\boldsymbol{x})$ is known as a generalized ridge function [85]. If $u(\boldsymbol{x})$ is represented in a tensor format, i.e., a series of products of one-dimensional functions $\psi(x_i)$ called tensor modes, then $v(\boldsymbol{x})$ inherits a similar series expansion. However, under the action of the linear map $\boldsymbol{A}\boldsymbol{x}$ the tensor modes are no longer univariate. Instead, they take the form of ridge functions $\psi(\boldsymbol{a}_i \cdot \boldsymbol{x})$, where $\boldsymbol{a}_i$ is the i-th row of the matrix $\boldsymbol{A}$. Since these ridge tensor modes are now

[1]If we allow for nonlinear coordinate flows [44], then we can of course map any multivariate probability density function (PDF) into a target distribution that has rank-one. Similar results can be obtained via optimal mass transport, e.g., by suitable approximations of the Knöthe-Rosenblatt rearrangement [95, 104].



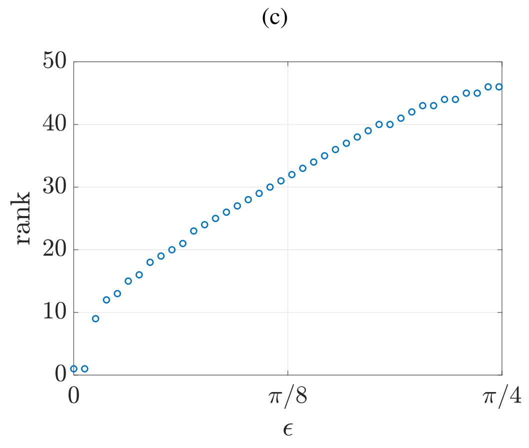


Figure 4.1: (a) Two-dimensional Gaussian stretched along the x_2-axis; (b) Two-dimensional rotated Gaussian (clockwise rotation by $\epsilon = \pi/4$ radians); (c) Rank of the the rotated Gaussian versus the clockwise rotation angle ϵ.

d-dimensional, the tensor compression which we had for $u(\boldsymbol{x})$ may be lost. The theoretical results we present for linear coordinate transformations provide a framework that be generalized to larger classes of nonlinear transformations, and open new pathways for tensor rank reduction that are complementary to classical rank reduction methods, e.g., based on hierarchical SVD [39, 41].

The remainder of this Chapter is organized as follows. In section 4.1 we introduce ridge tensor train and provide a simple example to illustrate the effects of coordinate flows on tensor rank. In section 4.2 we formulate the tensor rank reduction problem as a Riemannian optimization problem on the manifold of volume-preserving invertible coordinate maps. We also provide an efficient algorithm for computing the numerical solution to such optimization problem. In section 4.2.6 we demonstrate our proposed Riemannian gradient descent algorithm for computing a quasi-optimal linear coordinate map on a three-dimensional Gaussian mixture. In section 4.3 we apply the proposed rank reduction methods various linear and nonlinear PDEs. We also include two appendices in which we describe the Riemannian manifold of coordinate transformations, and provide theoretical results supporting the proposed mathematical framework for ridge tensors.

4.1 Tensor ridge functions

We begin by introducing a class of functions, which we call *tensor ridge functions*, that will be used in subsequent sections to build a tensor rank-reduction theory via linear coordinate mappings. To this end, consider the following invertible linear coordinate transformation

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}, \qquad \boldsymbol{A} : \mathbb{R}^d \to \mathbb{R}^d. \tag{4.3}$$

If we evaluate a function $u \in L^2_\mu(\Omega)$ at $\boldsymbol{y}$, we obtain the generalized ridge function $u(\boldsymbol{A}\boldsymbol{x})$ (e.g., [85]). Although the coordinate transformation $\boldsymbol{A}$ is linear, the evaluation of u on $\boldsymbol{A}\boldsymbol{x}$ defines a *nonlinear map* of functions

$$u(\boldsymbol{x}) \mapsto v(\boldsymbol{x}) = u(\boldsymbol{A}\boldsymbol{x}). \tag{4.4}$$

The image of a FTT tensor (2.32) under such a map has the form

$$u_{\text{TT}}(\boldsymbol{A}\boldsymbol{x}) = \\ \sum_{\alpha_0=1}^{r_0}\sum_{\alpha_1=1}^{r_1}\cdots\sum_{\alpha_d=1}^{r_d}\sqrt{\lambda(\alpha_{d-1})}\psi_1(\alpha_0;\boldsymbol{a}_1\cdot\boldsymbol{x};\alpha_1)\psi_2(\alpha_1;\boldsymbol{a}_2\cdot\boldsymbol{x};\alpha_2)\cdots\psi_d(\alpha_{d-1};\boldsymbol{a}_d\cdot\boldsymbol{x};\alpha_d), \tag{4.5}$$

which we call a *tensor ridge function*. In (4.5), $\boldsymbol{a}_i$ denotes the i-th row of the matrix $\boldsymbol{A}$ and "$\cdot$" is the standard dot product on $\mathbb{R}^d$, and $r_0 = r_d = 1$. When we consider $u_{\text{TT}}(\boldsymbol{A}\boldsymbol{x}) = u_{\text{TT}}(\boldsymbol{y})$ in coordinates $\boldsymbol{y}$, equation (4.5) is the FTT expansion for $u_{\text{TT}}(\boldsymbol{y})$. However, when we consider $v(\boldsymbol{x}) = u_{\text{TT}}(\boldsymbol{A}\boldsymbol{x})$ in coordinates $\boldsymbol{x}$ we have that each mode $\psi_i(\alpha_{i-1};\boldsymbol{a}_i\cdot\boldsymbol{x};\alpha_i)$ in the tensor ridge function (4.5) is no longer a univariate function of x_i as in (2.32), but rather a d-variate ridge function, which, has the property of being constant in all directions orthogonal to the vector $\boldsymbol{a}_i$ (e.g., [25, 85]). An important problem is determining the FTT expansion

$$v_{\text{TT}}(\boldsymbol{x}) = \sum_{\alpha_0=1}^{s_0}\sum_{\alpha_1=1}^{s_1}\cdots\sum_{\alpha_d=1}^{s_d}\sqrt{\theta(\alpha_{d-1})}\varphi_1(\alpha_0;x_1;\alpha_1)\varphi_2(\alpha_1;x_2;\alpha_2)\cdots\varphi_d(\alpha_{d-1};x_d;\alpha_d) \tag{4.6}$$

given the FTT expansion (4.5) for $u_{\text{TT}}(\boldsymbol{A}\boldsymbol{x})$. A naive approach to solve this problem would be to recompute the FTT expansion from scratch using the methods of section 2.3, i.e., treat $v_{\text{TT}}(\boldsymbol{x})$ as a multivariate function and solve a sequence of hierarchical eigenvalue problems. This is not practical even for a moderate number of dimensions d since the evaluation of $v(\boldsymbol{x})$ requires constructing a tensor product grid in d-dimensions, and each eigenvalue problem requires the computation of d-dimensional integrals. Another approach is to use TT-cross approximation [81], which provides an algorithm for interpolating d-dimensional black-box tensors in the tensor train format with computationally complexity that scales linearly with the dimension d. Hereafter, we develop a new approach to compute the FTT expansion (4.6) from (4.5) based on coordinate flows.

4.1.1 Computing tensor ridge functions via coordinate flows

Consider the non-autonomous linear dynamical system

$$\begin{cases} \dfrac{d\boldsymbol{y}(\epsilon)}{d\epsilon} = \boldsymbol{B}(\epsilon)\boldsymbol{y}(\epsilon), \\ \\ \boldsymbol{y}(0) = \boldsymbol{x}, \end{cases} \tag{4.7}$$

where $\boldsymbol{y}(\epsilon) \in \mathbb{R}^d$, and $\boldsymbol{B}(\epsilon)$ is a given $d \times d$ matrix with real entries for all $\epsilon \geq 0$. It is well-known that the solution to (4.7) can be written as

$$\boldsymbol{y}(\epsilon) = \boldsymbol{\Phi}(\epsilon)\boldsymbol{x}, \tag{4.8}$$

where

$$\boldsymbol{\Phi}(\epsilon) = e^{\boldsymbol{M}(\epsilon)} \tag{4.9}$$

is an invertible linear mapping on $\mathbb{R}^d$ for each $\epsilon \geq 0$. The matrix $\boldsymbol{M}(\epsilon)$ can be represented by the Magnus series (e.g., [14])

$$\boldsymbol{M}(\epsilon) = \int_0^{\epsilon} \boldsymbol{B}(\epsilon_1) d\boldsymbol{\epsilon}_1 - \frac{1}{2}\int_0^{\epsilon} \left\{ \int_0^{\epsilon_1} \boldsymbol{B}(\epsilon_2) d\epsilon_2, \boldsymbol{B}(\epsilon_1) \right\} d\epsilon_1 + \cdots, \tag{4.10}$$

where $\{\cdot,\cdot\}$ denotes the matrix commutator

$$\{\boldsymbol{P}, \boldsymbol{Q}\} = \boldsymbol{P}\boldsymbol{Q} - \boldsymbol{Q}\boldsymbol{P}. \tag{4.11}$$

Now that we have introduced coordinate flows and their connection to linear coordinate transformations, let us consider the problem of determining the FTT expansion of $u_{\text{TT}}(\boldsymbol{A}\boldsymbol{x})$ when $\boldsymbol{A}$

is generated by a coordinate flow, i.e., $\boldsymbol{A} = \boldsymbol{\Phi}(\epsilon)$ for some $\boldsymbol{\Phi}$ and some $\epsilon \geq 0$. Differentiating $v(\boldsymbol{x};\epsilon) = u_{\text{TT}}(\boldsymbol{\Phi}(\epsilon)\boldsymbol{x})$ with respect to ϵ yields the hyperbolic PDE

$$\begin{aligned}\frac{\partial v(\boldsymbol{x};\epsilon)}{\partial \epsilon} &= \frac{\partial u_{\text{TT}}(\boldsymbol{\Phi}(\epsilon)\boldsymbol{x})}{\partial \epsilon} \\ &= \nabla u_{\text{TT}}(\boldsymbol{\Phi}(\epsilon)\boldsymbol{x}) \cdot \left(\frac{\partial \boldsymbol{\Phi}(\epsilon)}{\partial \epsilon}\boldsymbol{x}\right) \\ &= \nabla u_{\text{TT}}(\boldsymbol{\Phi}(\epsilon)\boldsymbol{x}) \cdot \left(\boldsymbol{B}(\epsilon)\boldsymbol{\Phi}(\epsilon)\boldsymbol{x}\right),\end{aligned} \tag{4.12}$$

where in the second line we used the chain rule and in the third line we used equations (4.7) and (4.8). Recalling $\boldsymbol{\Phi}(0) = \boldsymbol{I}_{d\times d}$ (identity matrix), we see that the initial state $v(\boldsymbol{x};0) = u_{\text{TT}}(\boldsymbol{x})$ is in FTT format. Thus, we have derived the following hyperbolic initial value problem for the tensor ridge function $v(\boldsymbol{x};\epsilon)$

$$\begin{cases}\dfrac{\partial v(\boldsymbol{x};\epsilon)}{\partial \epsilon} = \nabla v(\boldsymbol{x};\epsilon) \cdot \left(\boldsymbol{B}(\epsilon)\boldsymbol{\Phi}(\epsilon)\boldsymbol{x}\right), \\ v(\boldsymbol{x};0) = u_{\text{TT}}(\boldsymbol{x}),\end{cases} \tag{4.13}$$

with an initial condition that is given in an FTT format.

Integrating the PDE (4.13) forward in ϵ on a FTT tensor manifold, e.g. using rank-adaptive step-truncation methods [93, 92, 60] or dynamic tensor approximation methods [30, 29, 28, 69, 62], results in a FTT approximation $v_{\text{TT}}(\boldsymbol{x};\epsilon)$ of the function $v(\boldsymbol{x};\epsilon)$ for all $\epsilon \geq 0$. The computational cost of this approach for computing the FTT expansion of a FTT ridge function is precisely the same cost as solving the hyperbolic PDE (4.13) in the FTT format, which in the case of step-truncation or dynamic approximation has computational complexity that scales linearly with d. Note that the accuracy of $v_{\text{TT}}(\boldsymbol{x};\epsilon)$ as an approximation of $v(\boldsymbol{x};\epsilon)$ depends on the ϵ step-size and order of integration scheme used to solve the PDE (4.13).

Using coordinate flows, it is straightforward to compute the FTT expansion of a tensor ridge function $u_{\text{TT}}(\boldsymbol{A}\boldsymbol{x})$ when the matrix $\boldsymbol{A}$ admits a real matrix logarithm $\boldsymbol{L}$. In this case, setting $\boldsymbol{B}(\epsilon) = \boldsymbol{L}$ yields $\boldsymbol{\Phi}(\epsilon) = e^{\epsilon \boldsymbol{L}}$, and therefore $\boldsymbol{A} = \boldsymbol{\Phi}(1)$. This means that we need to integrate (4.13) with $\boldsymbol{B}(\epsilon) = \boldsymbol{L}$ up to $\epsilon = 1$ to obtain the FTT approximation of the tensor ridge function $u_{\text{TT}}(\boldsymbol{A}\boldsymbol{x})$. Let us provide a simple example.

An example of application Consider the two-dimensional Gaussian function depicted in Figure 4.1(a), i.e.,

$$u_{\text{TT}}(\boldsymbol{x}) = e^{-x_1^2 - x_2^2/10}. \tag{4.14}$$

Clearly, (4.14) is the product of two univariate functions and therefore the FTT tensor representation coincides with (4.14) and has rank equal to one. Next, consider a simple linear coordinate transformation $\boldsymbol{\Phi}(\epsilon)$ which rotates the (x_1, x_2)-plane by an angle of ϵ radians. It is well-known that

$$\boldsymbol{\Phi}(\epsilon) = e^{\epsilon \boldsymbol{L}}, \tag{4.15}$$

where

$$\boldsymbol{L} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{4.16}$$

is the infinitesimal generator of the two-dimensional rotation. The dynamical system (4.7) defining the coordinate flow $\boldsymbol{y}(\epsilon) = \boldsymbol{\Phi}(\epsilon)\boldsymbol{x}$ can be written as

$$\begin{cases} \dfrac{d\boldsymbol{y}(\epsilon)}{d\epsilon} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \boldsymbol{y}(\epsilon), \\ \boldsymbol{y}(0) = \boldsymbol{x}. \end{cases} \tag{4.17}$$

The tensor ridge function corresponding to the coordinate map $\boldsymbol{\Phi}(\epsilon)$ is given analytically by

$$\begin{aligned} v(\boldsymbol{x};\epsilon) &= u_{\text{TT}}(\boldsymbol{\Phi}(\epsilon)\boldsymbol{x}) \\ &= e^{-(\Phi_{11}(\epsilon)x_1+\Phi_{12}(\epsilon)x_2)^2} e^{-(\Phi_{21}(\epsilon)x_1+\Phi_{22}(\epsilon)x_2)^2/10}. \end{aligned} \tag{4.18}$$

The hyperbolic PDE (4.13) for $v(\boldsymbol{x};\epsilon)$ in this case is given by

$$\begin{cases} \dfrac{\partial v(\boldsymbol{x};\epsilon)}{\partial \epsilon} = -x_2 \dfrac{\partial v(\boldsymbol{x};\epsilon)}{\partial x_1} + x_1 \dfrac{\partial v(\boldsymbol{x};\epsilon)}{\partial x_2}, \\ v(\boldsymbol{x};0) = u_{\text{TT}}(\boldsymbol{x}). \end{cases} \tag{4.19}$$

It is straightforward to verify that (4.18) satisfies (4.19). Note that $v(\boldsymbol{x};\epsilon)$ in (4.18) is not an FTT tensor if $\epsilon \neq \pi k/2$ and $k \in \mathbb{N}$. To compute the FTT representation of $v_{\text{TT}}(\boldsymbol{x};\epsilon)$ we can solve the PDE (4.19) on a tensor manifold using step-truncation or dynamic tensor approximation methods [28, 30, 92, 94]. Given the low dimensionality of the spatial domain in this example ($d = 2$), we can also evaluate (4.18) directly, and compute its FTT decomposition by solving an eigenvalue problem. In Figure 4.1 (b) we provide a contour plot of $v(\boldsymbol{x};\pi/4)$. To demonstrate the effect of rotations on tensor rank, in Figure 4.1(c) we plot the rank of $v(\boldsymbol{x};\epsilon)$ versus ϵ for all $\epsilon \in [0,\pi/4]$. Of course such a plot can be mirrored to obtain the rank for $\epsilon \in [\pi/4,\pi/2]$, $[\pi/2,3\pi/4]$, and $[3\pi/4,\pi]$.

4.2 Tensor rank reduction via coordinate flows

The coordinate flows we introduced in the previous section can be used to morph a given function into another one that has a faster decay rate of FTT singular values, i.e., a FTT tensor with lower rank (after truncation). A simple example is the coordinate flow that rotates the Gaussian function in Figure 4.1(b) back to the rank-one state depicted in Figure 4.1(a). This example and the examples documented in subsequent sections indicate that symmetry of the function relative to the new coordinate system plays an important role in reducing the tensor rank.

The problem of tensor rank reduction via linear coordinate flows can be formulated as follows: how do we choose an invertible linear coordinate transformation $\boldsymbol{A}$ so that $v_{\text{TT}}(\boldsymbol{x}) \approx u_{\text{TT}}(\boldsymbol{A}\boldsymbol{x})$ has smaller rank than $u_{\text{TT}}(\boldsymbol{x})$ (eventually minimum rank)? The mathematical statement of this optimization problem is

$$\boldsymbol{A} = \underset{\boldsymbol{A} \in \text{GL}_d(\mathbb{R})}{\text{argmin}} \ \text{rank}\left[v_{\text{TT}}(\boldsymbol{x})\right], \tag{4.20}$$

where $\text{GL}_d(\mathbb{R})$ denotes the set of $d \times d$ real invertible matrices, $\text{rank}[\cdot]$ is a metric related to the FTT rank, and $v_{\text{TT}}(\boldsymbol{x})$ is a FTT approximation of $u_{\text{TT}}(\boldsymbol{A}\boldsymbol{x})$. In equation (4.20) we have purposely left the cost function unspecified, as some care must be taken in its definition to ensure that the optimization problem is both feasible and computationally effective in reducing rank. One possibility is to define $\text{rank}[v_{\text{TT}}(\boldsymbol{x})]$ to return the sum of all d entries of the multilinear rank vector $\boldsymbol{r}$ corresponding to the FTT tensor $v_{\text{TT}}(\boldsymbol{x})$. While such a cost function is effective in measuring tensor rank it yields a NP-hard rank optimization problem [68].

4.2.1 Non-convex relaxation for the rank minimization problem

A common relaxation for rank minimization problems is to replace the rank cost function with the sum of the singular values. To describe this relaxation in the context of FTT tensors we first recall that any FTT tensor $u_{\text{TT}}(\boldsymbol{x})$ can be orthogonalized in the i-th variable as

(see Section 2.3.1)

$$u_{\text{TT}}(\boldsymbol{x}) = \boldsymbol{Q}_{\leq i}\boldsymbol{\Sigma}_i\boldsymbol{Q}_{>i}, \tag{4.21}$$

where

$$\boldsymbol{\Sigma}_i = \operatorname{diag}(\sigma_i(1), \sigma_i(2), \ldots, \sigma_i(r_i)), \tag{4.22}$$

is a diagonal matrix with real entries (singular values of u_{TT}). The matrices $\boldsymbol{Q}_{\leq i}$ and $\boldsymbol{Q}_{>i}$ are defined as partial products

$$\boldsymbol{Q}_{\leq i} = \boldsymbol{Q}_1\boldsymbol{Q}_2\cdots\boldsymbol{Q}_i, \qquad \boldsymbol{Q}_{>i} = \boldsymbol{Q}_{i+1}\boldsymbol{Q}_{i+2}\cdots\boldsymbol{Q}_d, \tag{4.23}$$

and they satisfy the orthogonality conditions

$$\left\langle \boldsymbol{Q}_{\leq i}^{\top}\boldsymbol{Q}_{\leq i}\right\rangle_{\leq i} = \boldsymbol{I}_{r_i\times r_i}, \qquad \left\langle \boldsymbol{Q}_{>i}\boldsymbol{Q}_{>i}^{\top}\right\rangle_{>i} = \boldsymbol{I}_{r_i\times r_i}. \tag{4.24}$$

Here, $\langle\cdot\rangle_{\leq i}$ and $\langle\cdot\rangle_{>i}$ are the averaging operators

$$\langle \boldsymbol{W}\rangle_{\leq i}(j,k) = \int_{\Omega_{\leq i}} w(j;\boldsymbol{x};k)d\mu_{\leq i}(\boldsymbol{x}_{\leq i}), \qquad \langle \boldsymbol{W}\rangle_{>i}(j,k) = \int_{\Omega_{>i}} w(j;\boldsymbol{x};k)d\mu_{>i}(\boldsymbol{x}_{>i}), \tag{4.25}$$

which map an arbitrary $r_i \times r_i$ matrix-valued function $\boldsymbol{W}(\boldsymbol{x})$ with entries $w(j;\boldsymbol{x};k)$ into another $r_i \times r_i$ matrix-valued function depending on a smaller number of variables. Using the orthogonalization (4.21) for each $i = 1, 2, \ldots, d-1$, we define the functions

$$\begin{aligned} S_i : L^2_{\mu}(\Omega) &\to \mathbb{R} \\ u_{\text{TT}}(\boldsymbol{x}) &\mapsto \sum_{\alpha_i=1}^{r_i} \sigma_i(\alpha_i), \end{aligned} \tag{4.26}$$

which returns the sum of the singular values corresponding to the i-th component of the multilinear rank vector. Using these functions we define

$$\begin{aligned} S : L^2_\mu(\Omega) &\to \mathbb{R} \\ u_{\text{TT}}(\boldsymbol{x}) &\mapsto \sum_{i=1}^{d-1} \sum_{\alpha_i=1}^{r_i} \sigma_i(\alpha_i), \end{aligned} \tag{4.27}$$

which is a relaxation of the rank cost function appearing in (4.20). An analogous relaxation of the rank cost function called the matrix nuclear norm has been studied extensively for matrix rank minimization problems [88, 68]. The function (4.27) has also been used as a relaxation of $\operatorname{rank}[\cdot]$ in tensor completion [10]. Next, we proceed by selecting an appropriate search space for the rank cost function. The largest search space we may choose is $\mathrm{GL}_d(\mathbb{R})$ as we have done in (4.20). This, however, is not a good choice since transformations in $\mathrm{GL}_d(\mathbb{R})$ are not volume-preserving and hence do not preserve L^2 norm, i.e., $\|u_{\text{TT}}(\boldsymbol{A}\boldsymbol{x})\|_{L^2_\mu} \neq \|u_{\text{TT}}(\boldsymbol{x})\|_{L^2_\mu}$. Transformations which do not preserve the norm of $u_{\text{TT}}(\boldsymbol{x})$ can reduce the rank cost function while having no impact on the tensor rank relative to the tensor's norm. Therefore we choose $\mathrm{SL}_d(\mathbb{R}) \subset \mathrm{GL}_d(\mathbb{R})$ as the search space, i.e., the collection of invertible matrices with determinant equal to one. These transformations are obviously volume-preserving, and therefore they preserve[2] the L^2 norm of u_{TT}.

Since the domain of S in (4.27) is $L^2_\mu(\Omega)$ and the cost function must be defined on

[2]It is straightforward to show with a simple change of variables that volume-preserving transformations preserve many quantities that are defined via an integral. For example, for any $u_{\text{TT}}(\boldsymbol{x}) \in L^2_\mu(\Omega)$ and $\boldsymbol{A} \in \mathrm{SL}_d(\mathbb{R})$ we have

$$\|u_{\text{TT}}(\boldsymbol{A}\boldsymbol{x})\|_{L^p_\mu\left(\boldsymbol{A}^{-1}\Omega\right)} = \|u_{\text{TT}}(\boldsymbol{x})\|_{L^p_\mu(\Omega)}, \qquad p = 1, 2. \tag{4.28}$$

the search space $\mathrm{SL}_d(\mathbb{R})$, we define an evaluation map E corresponding to $u_{\text{TT}}(\boldsymbol{x})$

$$\begin{aligned} E : \mathrm{SL}_d(\mathbb{R}) &\to L^2_\mu(\Omega) \\ \boldsymbol{A} &\mapsto u_{\text{TT}}(\boldsymbol{A}\boldsymbol{x}). \end{aligned} \tag{4.29}$$

Composing E with S yields the following (non-convex) relaxation of the cost function $\mathrm{rank}[\cdot]$ in (4.20)

$$C = S \circ E : \mathrm{SL}_d(\mathbb{R}) \to \mathbb{R}. \tag{4.30}$$

The function $C(\boldsymbol{A})$ returns the sum of the singular values of the tensor ridge function $u_{\text{TT}}(\boldsymbol{A}\boldsymbol{x})$, where $\boldsymbol{A}$ is an invertible matrix with determinant equal to one. The optimization problem corresponding to the cost function and search space discussed above is

$$\boldsymbol{A} = \underset{\boldsymbol{A} \in \mathrm{SL}_d(\mathbb{R})}{\operatorname{argmin}} \; C(\boldsymbol{A}). \tag{4.31}$$

While (4.31) is simpler than (4.20), it is still a computationally challenging problem for a number of reasons: First, it is non-convex. Second, when considered as a subset of all $d \times d$ matrices, the search space $SL_d(\mathbb{R})$ is subject to a non-trivial set of constraints, e.g., $\det(\boldsymbol{A}) = 1$. Third, we note that the set $SL_d(\mathbb{R})$ is unbounded (i.e. matrices with determinant 1 can have entries that are arbitrarily large) and thus it is important to monitor the size of the entries of the matrix $\boldsymbol{A}$ during optimization. Of course it is also possible to optimize over bounded subsets of $SL_d(\mathbb{R})$ such as the set of rotation matrices with determinant 1. Hereafter we develop a new method for obtaining a local minimum of (4.31). To handle the search space constraints, we give $\mathrm{SL}_d(\mathbb{R})$ a Riemannian manifold structure and perform gradient descent on this manifold [83]. To obtain the gradient of $C(\boldsymbol{A})$ efficiently we build its computation into the FTT truncation procedure at a negligible additional computational cost.

4.2.2 Riemannian gradient descent path

In Lemma B4 we show that if the FTT singular values of $u_{\text{TT}}(\boldsymbol{Ax})$ are simple (i.e., distinct) then the cost function $C(\boldsymbol{A})$ defined in (4.30) is differentiable in $\boldsymbol{A}$. With this sufficient condition for the smoothness of the cost function established, we can use the machinery of Riemannian geometry summarized in 4.A to construct a gradient descent path for the minimization of (4.31) on the search space $\text{SL}_d(\mathbb{R})$. To this end, let us denote by $\boldsymbol{\Gamma}(\epsilon)$ such gradient descent path. To build $\boldsymbol{\Gamma}(\epsilon)$, we start at the identity $\boldsymbol{\Gamma}(0) = \boldsymbol{I}$ and consider the matrix ordinary differential equation

$$\begin{cases} \dfrac{d\boldsymbol{\Gamma}(\epsilon)}{d\epsilon} = -\text{grad}\left[C\left(\boldsymbol{\Gamma}\left(\epsilon\right)\right)\right], \\ \boldsymbol{\Gamma}(0) = \boldsymbol{I}, \end{cases} \tag{4.32}$$

where $\text{grad}\left[C\left(\boldsymbol{\Gamma}\left(\epsilon\right)\right)\right]$ is the *Riemannian gradient* of the cost function C defined in (4.30). By construction, the vector $-\text{grad}\left[C(\boldsymbol{\Gamma}(\epsilon))\right]$ is tangent to the manifold $\text{SL}_d(\mathbb{R})$ at each point $\boldsymbol{\Gamma}\left(\epsilon\right)$, and thus $\boldsymbol{\Gamma}(\epsilon) \in \text{SL}_d(\mathbb{R})$ for all $\epsilon \geq 0$. Since $-\text{grad}\left[C\left(\boldsymbol{\Gamma}\left(\epsilon\right)\right)\right]$ points in the direction of steepest descent of the cost function C at the point $\boldsymbol{\Gamma}(\epsilon)$, the cost function is guaranteed to decrease along the path $\boldsymbol{\Gamma}(\epsilon)$ (or remain constant which implies we have obtained a local minimum). In Figure 4.2 we provide an illustration of the Riemannian gradient descent path $\boldsymbol{\Gamma}(\epsilon)$. For computational efficiency, it is essential to have a fast method for computing the Riemannian gradient of the cost function C at an arbitrary point $\boldsymbol{A} \in \text{SL}_d(\mathbb{R})$. The following Proposition provides an expression for such Riemannian gradient in terms of orthogonal FTT cores which we will use to efficiently compute the descent direction. The proof is given in 4.B.

Proposition 4.2.1. *The Riemannian gradient of the cost function* (4.30) *at the point* $\boldsymbol{A} \in$

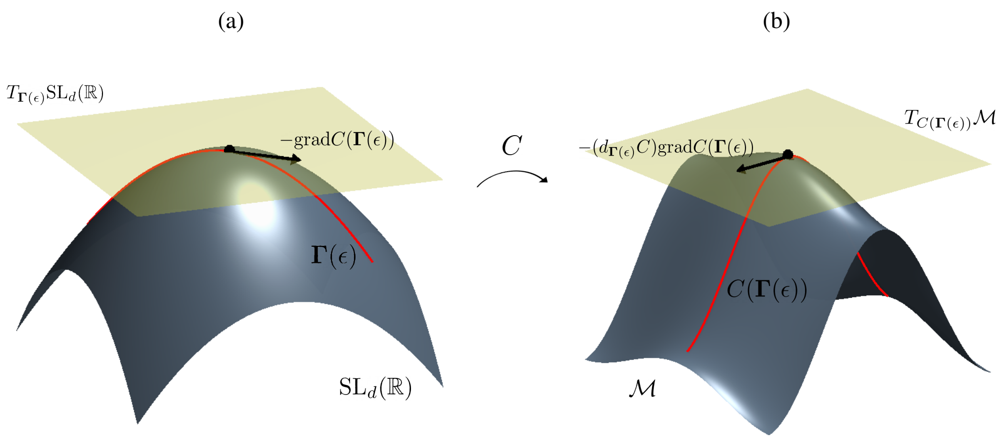

(a)
(b)
$T_{\mathbf{\Gamma}(\epsilon)}\mathrm{SL}_d(\mathbb{R})$
$-\mathrm{grad}C(\mathbf{\Gamma}(\epsilon))$
$\mathbf{\Gamma}(\epsilon)$
$\mathrm{SL}_d(\mathbb{R})$
C
$-(d_{\mathbf{\Gamma}(\epsilon)}C)\mathrm{grad}C(\mathbf{\Gamma}(\epsilon))$
$T_{C(\mathbf{\Gamma}(\epsilon))}\mathcal{M}$
$C(\mathbf{\Gamma}(\epsilon))$
$\mathcal{M}$

Figure 4.2: An illustration of a descent path $\mathbf{\Gamma}(\epsilon)$ for the cost function C defined in (4.30). At each point of the path $\mathbf{\Gamma}(\epsilon) \in \mathrm{SL}_d(\mathbb{R})$ we assign the tangent vector $-\mathrm{grad}\left[C(\mathbf{\Gamma}(\epsilon))\right] \in T_{\mathbf{\Gamma}(\epsilon)}\mathrm{SL}_d(\mathbb{R})$ which points in the direction of steepest descent $-(d_{\mathbf{\Gamma}(\epsilon)}C)\mathrm{grad}\left[C(\mathbf{\Gamma}(\epsilon))\right] \in T_{C(\mathbf{\Gamma}(\epsilon))}\mathcal{M}$ of the function C at the point $C(\mathbf{\Gamma}(\epsilon)) \in \mathcal{M}$.

$\mathrm{SL}_d(\mathbb{R})$ *is given by*

$$\mathrm{grad}\left[C\left(\boldsymbol{A}\right)\right] = \boldsymbol{D}\boldsymbol{A}, \tag{4.33}$$

where

$$\boldsymbol{D} = \sum_{i=1}^{d-1} \int_{\Omega} \boldsymbol{Q}_{\leq i}\boldsymbol{Q}_{>i} \left(\nabla v(\boldsymbol{x}) \left(\boldsymbol{A}\boldsymbol{x}\right)^{\top} - \frac{\nabla v(\boldsymbol{x})^{\top}\boldsymbol{A}\boldsymbol{x}}{d} \boldsymbol{I}_{d\times d} \right) d\mu(\boldsymbol{x}), \tag{4.34}$$

$v(\boldsymbol{x}) = u_{TT}(\boldsymbol{A}\boldsymbol{x})$ *and* $\boldsymbol{Q}_{\leq i}, \boldsymbol{Q}_{>i}$ *are tensor cores of the orthogonalized FTT*

$$v(\boldsymbol{x}) = \boldsymbol{Q}_{\leq i}\boldsymbol{\Sigma}_i\boldsymbol{Q}_{>i}. \tag{4.35}$$

Using the gradient descent path defined by (4.32) we differentiate the coordinate transformation

$$\boldsymbol{y}(\epsilon) = \mathbf{\Gamma}(\epsilon)\boldsymbol{x} \tag{4.36}$$

with respect to ϵ and use (4.32)-(4.33) to obtain

$$\begin{cases} \dfrac{d\boldsymbol{y}(\epsilon)}{d\epsilon} = -\boldsymbol{D}(\epsilon)\boldsymbol{y}(\epsilon), \\ \boldsymbol{y}(0) = \boldsymbol{x}. \end{cases} \tag{4.37}$$

Note that (4.37) has the same form as the ODE (4.7) we used to define coordinate flows. Hence, by construction, the flow map generated by (4.37) is the descent path $\mathbf{\Gamma}(\epsilon)$ on the manifold $\mathrm{SL}_d(\mathbb{R})$, which, converges to a local minimum of C as ϵ increases. By defining the function $v(\boldsymbol{x};\epsilon) = u_{\text{TT}}(\mathbf{\Gamma}(\epsilon)\boldsymbol{x})$ and differentiating it with respect to ϵ we obtain the hyperbolic PDE

$$\begin{cases} \dfrac{\partial v(\boldsymbol{x};\epsilon)}{\partial \epsilon} = -\nabla v(\boldsymbol{x};\epsilon) \cdot \left[\boldsymbol{D}(\epsilon)\mathbf{\Gamma}(\epsilon)\boldsymbol{x}\right], \\ v(\boldsymbol{x};0) = u_{\text{TT}}(\boldsymbol{x}). \end{cases} \tag{4.38}$$

Note that the evolution of $\boldsymbol{y}(\epsilon) = \mathbf{\Gamma}(\epsilon)\boldsymbol{x}$ at the right hand side of (4.38) is defined by (4.37). Integrating (4.37)-(4.38) forward in ϵ yields a rank-reducing linear coordinate transformation $\mathbf{\Gamma}(\epsilon)$ and the reduced rank function $v(\boldsymbol{x};\epsilon) = u_{\text{TT}}\left(\mathbf{\Gamma}(\epsilon)\boldsymbol{x}\right)$.

4.2.3 Numerical integration of the gradient descent equations

It is convenient to use a step-truncation method [92, 94, 60] to integrate the initial value problem (4.38) on a FTT tensor manifold. This is because applying the FTT truncation operation to $v(\boldsymbol{x};\epsilon)$ requires computing the orthogonalized tensor cores $\boldsymbol{Q}_{\leq i}(\epsilon), \boldsymbol{Q}_{>i}(\epsilon)$ $(i = 1, 2, \ldots, d-1)$ which can then be readily used to evaluate the matrix $\boldsymbol{D}(\epsilon)$ defining the Riemannian gradient (4.34). Moreover, the FTT truncation operation applied to $v(\boldsymbol{x};\epsilon)$ yields an expansion of the form (4.6), which is the desired tensor format. To describe the integration algorithm in more detail, let us discretize the interval $[0, \epsilon_f]$ into $N+1$ evenly-spaced points[3]

$$\epsilon_i = i\Delta\epsilon, \qquad \Delta\epsilon = \frac{\epsilon_f}{N}, \qquad i = 0, 1, \ldots, N, \tag{4.39}$$

[3]The right end-point ϵ_f will ultimately be determined by the stopping criterion for gradient descent.

and let $v_i, \boldsymbol{\Gamma}_i, \boldsymbol{D}_i$ denote $v(\boldsymbol{x}; \epsilon_i), \boldsymbol{\Gamma}(\epsilon_i), \boldsymbol{D}(\epsilon_i)$, respectively. Let

$$v_{i+1} = v_i + \Delta\epsilon \Phi \left(v_i, \boldsymbol{D}_i, \Delta\epsilon \right), \tag{4.40}$$

$$\boldsymbol{\Gamma}_{i+1} = \boldsymbol{\Gamma}_i + \Delta\epsilon \widehat{\boldsymbol{\Phi}} \left(\boldsymbol{\Gamma}_i, \boldsymbol{D}_i, \Delta\epsilon \right) \tag{4.41}$$

be one-step explicit integration schemes approximating the solution to the initial value problem (4.38) and the matrix ODE (4.32), respectively. In order to guarantee that the solution v_{i+1} is a low-rank FTT tensor, we apply a truncation operator to the right hand side of (4.40). This yields the step-truncation method [92]

$$v_{i+1} = \mathfrak{T}_\delta \left(v_i + \Delta\epsilon \Phi \left(v_i, \boldsymbol{D}_i, \Delta\epsilon \right) \right). \tag{4.42}$$

Here, $\mathfrak{T}_\delta$ denotes the standard FTT truncation operator with relative accuracy δ proposed in [82] modified to return the matrix $\boldsymbol{D}_{i+1}$. A detailed description of the modified tensor truncation algorithm is provided in section 4.2.4.

At this point, a few remarks regarding the integration scheme (4.40)-(4.42) are in order. First, we notice that at each step of the gradient descent algorithm we are computing the gradient (4.33) at the identity matrix since the current tensor v_i is the tensor ridge function $v_i(\boldsymbol{x}) = u_{\text{TT}}(\boldsymbol{\Gamma}_i \boldsymbol{x})$. Second, the accuracy of the tensor v_i approximating $u_{\text{TT}}(\boldsymbol{\Gamma}_i \boldsymbol{x})$ is determined by the chosen integration scheme and its relevant parameters (i.e., the function Φ, the step size $\Delta\epsilon$, and the accuracy of $\mathfrak{T}_\delta$). In a standard gradient descent algorithm, convergence can be expedited with a line-search routine that determines an appropriate step-size to take in the descent direction. However, in the proposed integration scheme the step-size determines the accuracy of the final tensor, thus we keep the step-size $\Delta\epsilon$ fixed during gradient descent. We set a stopping criterion for the integration of (4.40)-(4.42) based on the empirical observation that the cost function C does not decrease substantially along the descent path $\mathbf{\Gamma}(\epsilon)$ when $\mathbf{\Gamma}_i$ is close to a local minimum. Mathematically, this translates into the condition

$$\frac{dS(v_i)}{d\epsilon} > -\eta, \tag{4.43}$$

where η is some predetermined tolerance. Since the solution history $v_i, v_{i-1}, \dots$ is available during gradient descent, a simple method for approximating $dS(v_i)/d\epsilon$ is a p-point backwards difference stencil

$$\frac{dS(v_i)}{d\epsilon} \approx \mathrm{BD}^{(p)}(S(v_i), S(v_{i-1}), \dots, S(v_{i-p})). \tag{4.44}$$

In addition to the stopping criterion (4.43), we also set a maximum number of iterations M_{iter} to ensure that the Riemannian gradient descent algorithm halts within a reasonable amount of time. We summarize the proposed Riemannian gradient descent method to compute a local minimum of (4.31) in Algorithm 2.

4.2.4 Modified tensor truncation algorithm

To speed up numerical integration of the gradient descent equations (4.41)-(4.42) it is convenient to compute the matrix $\boldsymbol{D}_i$ defined in (4.34) during FTT truncation. The reason being that standard FTT truncation requires the computation of all orthogonal FTT cores $\boldsymbol{Q}_{\leq j}$ and $\boldsymbol{Q}_{>j}$, which, can be readily used to compute $\boldsymbol{D}_i$. To describe the modified tensor truncation algorithm, let $v_{\text{TT}}(\boldsymbol{x}) = \boldsymbol{\Psi}_1 \cdots \boldsymbol{\Psi}_d$ be an FTT tensor with non-optimized rank, e.g., $v_{\text{TT}}(\boldsymbol{x})$ is the result of adding two FTT tensors together. Denote by $\mathfrak{T}_\delta$ the modified truncation operator, where δ is the required relative accuracy, i.e.,

$$\|v_{\text{TT}}(\boldsymbol{x}) - \mathfrak{T}_\delta(v_{\text{TT}}(\boldsymbol{x}))\|_{L^2_\mu(\Omega)} \leq \delta \|v_{\text{TT}}(\boldsymbol{x})\|_{L^2_\mu(\Omega)}. \tag{4.45}$$

Algorithm 2: Riemannian gradient descent for computing rank-reducing linear coordinate transformations.

Input:

$u_{\text{TT}} \to$ initial FTT tensor,

$\Delta\epsilon \to$ step-size for gradient descent,

$\eta \to$ stopping tolerance,

$M_{\text{iter}} \to$ maximum number of iterations.

Output:

$\mathbf{\Gamma} \to$ rank-reducing linear coordinate transformation,

$v_{\text{TT}} \to$ reduced rank FTT tensor on transformed coordinates $v_{\text{TT}}(\boldsymbol{x}) = u_{\text{TT}}(\mathbf{\Gamma}\boldsymbol{x})$.

Runtime:

```
    $[v_0, S(v_0), \boldsymbol{D}_0] = \mathfrak{T}_\delta(u_{\text{TT}})$,
    $\mathbf{\Gamma}_0 = \boldsymbol{I}$,
    $\dot{S}(v_0) = -\infty$,
    $i = 0$.
    while $\dot{S}(v_i) < -\eta$ and $i \leq M_{\text{iter}}$
        $v_{i+1} = v_i + \Delta\epsilon\Phi(v_i, \boldsymbol{D}_i, \Delta\epsilon)$,
        $[v_{i+1}, S(v_{i+1}), \boldsymbol{D}_{i+1}] = \mathfrak{T}_\delta(v_{i+1})$,
        $\mathbf{\Gamma}_{i+1} = \mathbf{\Gamma}_i + \Delta\epsilon\widehat{\mathbf{\Phi}}(\mathbf{\Gamma}_i, \boldsymbol{D}_i, \Delta\epsilon)$,
        $\dot{S}(v_{i+1}) = \text{BD}^{(p)}(S(v_{i+1}), S(v_i), \dots, S(v_{i+1-p}))$,
        $i = i + 1$,
        $\mathbf{\Gamma} = \mathbf{\Gamma}_i$,
        $v_{\text{TT}} = v_i$.
    end
```

Left to right orthogonalization

$$\begin{tikzcd} 1) & \mathbf{\Psi}_1 \cdots \mathbf{\Psi}_i \cdots \mathbf{\Psi}_d \arrow[r, bend left] & \mathbf{Q}_1 \cdots \mathbf{Q}_i \mathbf{R}_i \mathbf{\Psi}_{i+1} \cdots \mathbf{\Psi}_d \arrow[r, bend left] & \mathbf{Q}_1 \cdots \mathbf{Q}_{d-1} \mathbf{\Psi}_d \end{tikzcd}$$

Right to left truncation

$$\begin{tikzcd} 2)\ (i) & \mathbf{Q}_1 \cdots \mathbf{Q}_{d-1} \mathbf{\Psi}_d \arrow[r, bend left] & \mathbf{Q}_1 \cdots \mathbf{Q}_i \mathbf{L}_i \mathbf{Q}_{i+1} \cdots \mathbf{Q}_d \arrow[r, bend left] & \mathbf{Q}_1 \cdots \mathbf{Q}_i \mathbf{\Sigma}_i \mathbf{Q}_{i+1} \cdots \mathbf{Q}_d \end{tikzcd}$$

Gradient computation

$$2)\ (ii) \qquad \mathbf{D}^{(i)} = \int_\Omega \mathbf{Q}_{\leq i} \mathbf{Q}_{>i} \left(\nabla v(\mathbf{x}) \left(\mathbf{A}\mathbf{x}\right)^{\mathrm{T}} - \frac{\nabla v(\mathbf{x}) \mathbf{A}\mathbf{x}}{d} \mathbf{I}_{d\times d} \right) d\mu(\mathbf{x})$$

Figure 4.3: A summary of the modified FTT truncation algorithm for computing the Riemannian gradient (4.34).

We also define

$$\hat{\delta} = \frac{\delta}{\sqrt{d-1}} \|v_{\mathrm{TT}}(\boldsymbol{x})\|_{L^2_\mu(\Omega)}, \tag{4.46}$$

which is the required accuracy for each SVD in the FTT truncation algorithm. As described in section 2.3.1, we may perform a functional analogue of the QR decomposition on the FTT tensor cores of $v_{\mathrm{TT}}(\boldsymbol{x})$

$$\mathbf{\Psi}_i = \boldsymbol{Q}_i \boldsymbol{R}_i, \tag{4.47}$$

where $\boldsymbol{Q}_i$ is a $r_{i-1} \times r_i$ matrix with elements in $L^2_{\mu_i}(\Omega_i)$ satisfying $\left\langle \boldsymbol{Q}_i^\top \boldsymbol{Q}_i \right\rangle_i = \boldsymbol{I}_{r_i \times r_i}$, and $\boldsymbol{R}_i$ is an upper triangular $r_i \times r_i$ matrix with real entries. Similarly, we can perform an LQ-factorization

$$\mathbf{\Psi}_i = \boldsymbol{L}_i \boldsymbol{Q}_i, \tag{4.48}$$

where $\boldsymbol{Q}_i$ is a $r_{i-1} \times r_i$ matrix with elements in $L^2_{\mu_i}(\Omega_i)$ satisfying $\left\langle \boldsymbol{Q}_i^\top \boldsymbol{Q}_i \right\rangle_i = \boldsymbol{I}_{r_i \times r_i}$, and $\boldsymbol{L}_i$ is a lower-triangular $r_i \times r_i$ matrix with real entries. The first procedure in the modified truncation routine is a left-to-right orthogonalization sweep in which we first perform the QR decomposition

$$\boldsymbol{\Psi}_1 = \boldsymbol{Q}_1 \boldsymbol{R}_1, \tag{4.49}$$

and then update the core $\boldsymbol{\Psi}_2$

$$\boldsymbol{\Psi}_2 = \boldsymbol{R}_1 \boldsymbol{\Psi}_2. \tag{4.50}$$

This process is repeated recursively

$$\boldsymbol{\Psi}_i = \boldsymbol{Q}_i \boldsymbol{R}_i, \qquad \boldsymbol{\Psi}_{i+1} = \boldsymbol{R}_i \boldsymbol{\Psi}_{i+1}, \qquad i = 2, \ldots, d-1, \tag{4.51}$$

resulting in the orthogonalization

$$v_{\text{TT}}(\boldsymbol{x}) = \boldsymbol{Q}_1 \boldsymbol{Q}_2 \cdots \boldsymbol{Q}_{d-1} \boldsymbol{\Psi}_d. \tag{4.52}$$

Next, we perform a right-to-left sweep which compresses $v_{\text{TT}}(\boldsymbol{x})$ and simultaneously computes each term appearing in the summation of $\boldsymbol{D}$ in equation (4.34). The first step of this procedure is to compute a LQ decomposition of $\boldsymbol{\Psi}_d$

$$\boldsymbol{\Psi}_d = \boldsymbol{L}_d \boldsymbol{Q}_d, \tag{4.53}$$

and then perform a truncated singular value decomposition of $\boldsymbol{L}_d$ with threshold $\hat{\delta}$

$$\boldsymbol{L}_d = \boldsymbol{U}_d \boldsymbol{\Sigma}_d \boldsymbol{V}_d^\top. \tag{4.54}$$

Substituting (4.53) and (4.54) into (4.52) yields

$$v_{\text{TT}}(\boldsymbol{x}) = \boldsymbol{Q}_1 \boldsymbol{Q}_2 \cdots \boldsymbol{Q}_{d-1} \boldsymbol{\Sigma}_d \boldsymbol{Q}_d, \tag{4.55}$$

where we re-defined

$$\boldsymbol{Q}_{d-1} = \boldsymbol{Q}_{d-1} \boldsymbol{U}_d, \qquad \boldsymbol{Q}_d = \boldsymbol{V}_d^\top \boldsymbol{Q}_d. \tag{4.56}$$

At this point we have truncated the FTT tensor $v_{\text{TT}}(\boldsymbol{x})$ in the d-th variable. The expansion (4.56) provides the FTT orthogonalization needed to compute the $(d-1)$-th term in the sum (4.34)

$$\boldsymbol{D}^{(d-1)} = \int_{\Omega} \boldsymbol{Q}_{\leq d-1} \boldsymbol{Q}_{>d} \left(\nabla v_{\text{TT}}(\boldsymbol{x}) \boldsymbol{x}^{\top} \right) d\mu(\boldsymbol{x}), \tag{4.57}$$

which, can be computed efficiently by applying one-dimensional differentiation matrices and quadrature rules to the FTT cores. We proceed in a recursive manner ($i = d-1, \ldots, 2$) with the same steps described above for $\boldsymbol{\Psi}_d$. First compute the LQ decomposition

$$(\boldsymbol{Q}_i \boldsymbol{\Sigma}_{i+1}) = \boldsymbol{L}_i \boldsymbol{Q}_i, \tag{4.58}$$

and then perform a singular value decomposition with threshold $\hat{\delta}$

$$\boldsymbol{L}_i = \boldsymbol{U}_i \boldsymbol{\Sigma}_i \boldsymbol{V}_i^{\top}. \tag{4.59}$$

Then rewrite $v_{\text{TT}}(\boldsymbol{x})$ as

$$v_{\text{TT}}(\boldsymbol{x}) = \boldsymbol{Q}_1 \boldsymbol{Q}_2 \cdots \boldsymbol{Q}_{i-1} \boldsymbol{\Sigma}_i \boldsymbol{Q}_i \cdots \boldsymbol{Q}_d, \tag{4.60}$$

where we re-defined

$$\boldsymbol{Q}_{i-1} = \boldsymbol{Q}_{i-1} \boldsymbol{U}_i, \qquad \boldsymbol{Q}_i = \boldsymbol{V}_i^{\top} \boldsymbol{Q}_i. \tag{4.61}$$

The expansion (4.60) provides the FTT orthogonalization required to compute the $(i-1)$-th term in the sum (4.34)

$$\boldsymbol{D}^{(i-1)} = \int_{\Omega} \boldsymbol{Q}_{\leq i-1} \boldsymbol{Q}_{>i} \left(\nabla v_{\text{TT}}(\boldsymbol{x}) \boldsymbol{x}^{\top} \right) d\mu(\boldsymbol{x}), \tag{4.62}$$

which, can be computed efficiently by applying one-dimensional differentiation matrices and quadrature rules to the FTT cores. Finally with all of the terms in (4.34) computed we simply sum them to obtain the matrix

$$\boldsymbol{D} = \sum_{i=1}^{d-1} \boldsymbol{D}^{(i)}. \tag{4.63}$$

We summarize the main steps of the modified tensor truncation in Figure 4.3 and in Algorithm 3. Clearly, the matrix $\boldsymbol{D}$ needs to be recomputed at each step ϵ_i, resulting in the matrix $\boldsymbol{D}_i$ appearing in the gradient descent equations (4.41)-(4.42).

4.2.5 Computational cost

One step of a first-order step-truncation integrator of the form (4.42) (e.g. Euler forward) without computing the left factor of the Riemannian gradient (4.34) requires one multiplication between a scalar and a TT tensor ($\mathcal{O}(nr^2)$ FLOPS), one addition between two TT tensors, and one FTT truncation ($\mathcal{O}(dnr^3)$ FLOPS) for a total computational complexity $\mathcal{O}(dnr^3)$. In the above estimate we assumed that each entry of the rank of the FTT tensor $v_i + \Delta\epsilon\Phi(v_i, \boldsymbol{D}_i, \Delta\epsilon)$ is bounded by r and v_i is discretized on a grid with n points in each variable. In addition to the cost of step-truncation, we must also compute the matrix $\boldsymbol{D}_i$ defined in (4.34) at each step. The orthogonal FTT cores $\boldsymbol{Q}_{\leq j}, \boldsymbol{Q}_{>j}$ needed for the computation of $\boldsymbol{D}_i$ are readily available during the tensor truncation procedure. For the computation of $\boldsymbol{D}_i$ we must compute the gradient of ∇v_i, which requires d matrix multiplications between a differentiation matrix of size $n \times n$ and a FTT core $\boldsymbol{\Psi}_i$ of size $n \times rn$ for total complexity of $\mathcal{O}(dn^3r)$. We also need to compute the outer product $(\nabla v_i)\boldsymbol{x}^\top$ where each entry of ∇v_i is in FTT format, and, for each of the d^2 entries in the matrix $(\nabla v_i)\boldsymbol{x}^\top$ compute an integral of FTT tensors requiring $\mathcal{O}(dnr^3)$ FLOPS. The final estimate for computing the matrix $\boldsymbol{D}_i$ is $\mathcal{O}(d^2n^3r^3)$, which dominates the cost of performing one-step of (4.42). We point out that the computation of $\boldsymbol{D}_i$

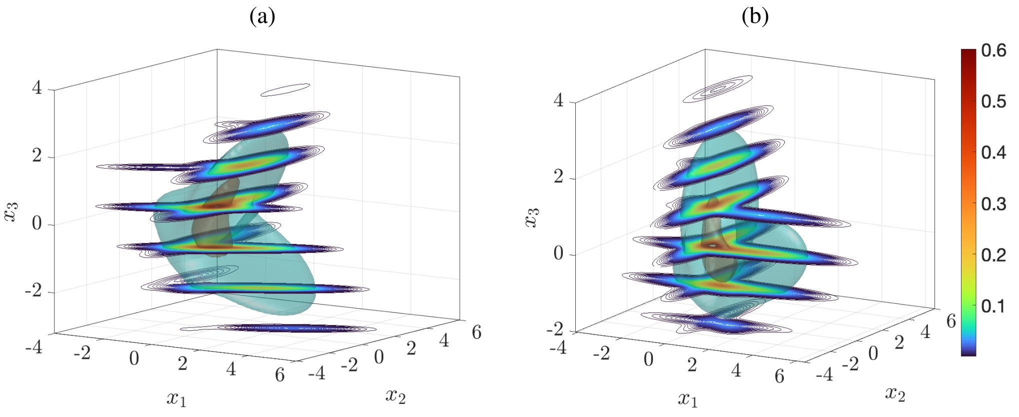


Figure 4.4: (a) Volumetric plot of the three-dimensional Gaussian mixture (2.47). (b) Volumetric plot of the corresponding reduced rank ridge tensor $v(\boldsymbol{x};\epsilon_f)$.

may be incorporated into high performance computing algorithms for tensor train rounding, e.g., [26, 102].

4.2.6 Application to a time-independent function

We now demonstrate the Riemannian gradient descent algorithm for generating rank-reducing linear coordinate transformations. Consider the three-dimensional ($d = 3$) Gaussian mixture (2.47) with parameters defined in (2.48)-(2.50). We discretize the Gaussian mixture (2.47) on the computational domain $[-12, 12]^3$ (which is large enough to enclose the numerical support of (2.47)) using 200 evenly-spaced points in each variable. From the discretization of (2.47) we compute the FTT decomposition $u_{\text{TT}}(\boldsymbol{x})$ using recursive SVDs. To integrate the PDE (4.38) for the reduced rank ridge tensor $v(\boldsymbol{x};\epsilon)$, we use the explicit two-step Adams-Bashforth step-truncation method with $\Delta\epsilon = 10^{-3}$ and relative FTT truncation accuracy $\delta = 10^{-6}$. All spatial derivatives and integrals are computed by applying one-dimensional pseudo-spectral Fourier differentiation matrices and quadrature weights [45] to the tensor modes. We integrate

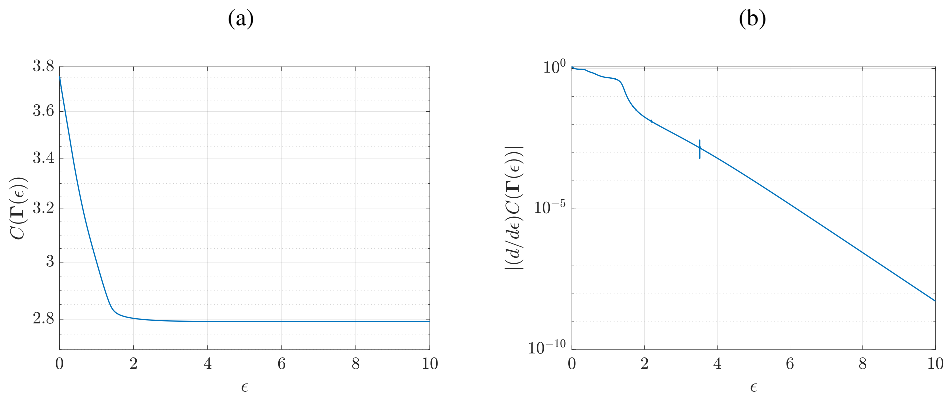


Figure 4.5: Rank reduction problem via coordinate flow for the three-dimensional Gaussian mixture (2.47). (a) Cost function (4.31) evaluated along a steepest descent mapping $\mathbf{\Gamma}(\epsilon)$ versus ϵ. (b) Absolute value of the derivative of the cost function in (4.31) versus ϵ. The derivative is computed with a second-order backwards finite difference stencil (4.44).

up to $\epsilon_f = 10$ which is sufficient to demonstrate convergence of the gradient descent method. In Figure 4.4 we provide volumetric plots of the of the function $v(\boldsymbol{x};\epsilon)$ for $\epsilon = 0$ and $\epsilon = \epsilon_f$. We observe that the reduced rank ridge tensor $v(\boldsymbol{x};\epsilon_f)$ appears to be more symmetrical with respect to the (x_1, x_2, x_3)-axes than the higher rank function $v(\boldsymbol{x}; 0)$. In Figure 4.5(a) we plot the cost function $C(\mathbf{\Gamma}(\epsilon))$ versus ϵ and in Figure 4.5(b) we plot the absolute value of the derivative of the cost function versus ϵ. Observe in Figure 4.5 that $\mathbf{\Gamma}(\epsilon)$ appears to be converging to a local minimum of C, and, the majority of the decrease in the cost function occurs in the interval $\epsilon \in [0, 2]$. Correspondingly, we notice that in Figure (4.6)(c) the majority of rank increase occurs in the interval $\epsilon \in [0, 2]$. Finally, in Figure 4.6(a)-(b) we plot the multilinear spectra of the function $v_{\text{TT}}(\boldsymbol{x};\epsilon)$ for $\epsilon = 0$ and $\epsilon = \epsilon_f$. The decay rate of the multilinear spectra corresponding to $v(\boldsymbol{x};\epsilon_f)$ is significantly faster than the decay rate of the multilinear corresponding to the initial function $v(\boldsymbol{x}; 0)$, resulting in a multilinear rank of about half the one in Cartesian coordinates. Thus, for any truncation tolerance δ, the FTT ridge tensor $v(\boldsymbol{x};\epsilon_f)$

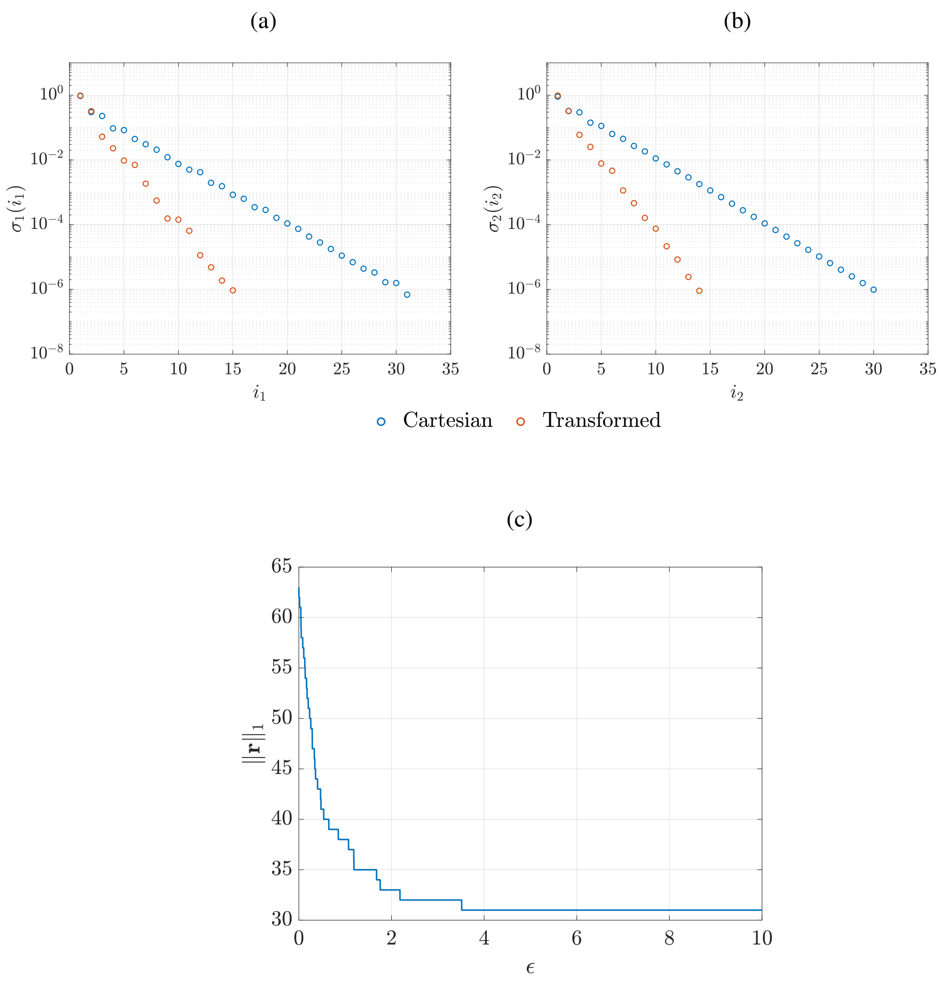


Figure 4.6: Multilinear spectra of the 3-dimensional Gaussian mixture $v(\boldsymbol{x}; 0)$ defined in (2.47) (Cartesian coordinates) and the corresponding reduced-rank ridge tensor $v(\boldsymbol{x}; \epsilon_f)$ (transformed coordinates). (a) Spectra σ_1 corresponding to multilinear rank r_1. (b) Spectra σ_2 corresponding to multilinear rank r_2. (c) 1-norm of the multilinear rank vector of $v_{\text{TT}}(\boldsymbol{x}; \epsilon)$ versus ϵ. It is seen that the coordinate flow $\boldsymbol{\Gamma}(\epsilon)\boldsymbol{x}$ reduces the multilinear rank of the Gaussian mixture (2.47) from about 63 (Cartesian coordinates) to 31 (transformed coordinates).

can be stored at a significantly lower cost than the original function. Intuitively, the savings that can be obtained by the coordinate flow in higher-dimensions are even more pronounced, since hierarchical SVDs with steeper spectra yield a much smaller number of tensor modes.

4.3 Application to PDEs

We now apply the proposed rank tensor reduction method to initial-value problems of the form (1.1). For a given PDE operator $G(\cdot, \boldsymbol{x})$ and linear coordinate transformation $\boldsymbol{y} = \boldsymbol{\Gamma}\boldsymbol{x}$, it is always possible to write G in coordinates $\boldsymbol{y}$ resulting in a new operator $G_{\boldsymbol{\Gamma}}$. If G acts on $u_{\text{TT}}(\boldsymbol{x}, t_k)$ then $G_{\boldsymbol{\Gamma}}$ acts on the transformed tensor $u_{\text{TT}}(\boldsymbol{y}, t_k) = v_{\text{TT}}(\boldsymbol{x}, t_k)$. Such an operator can be constructed using standard tools of differential geometry [2, 108], and usually has different FTT-operator rank than G. For example, consider the variable coefficient advection operator

$$G(u(\boldsymbol{x}, t), \boldsymbol{x}) = \sum_{i=1}^{d} f_i(\boldsymbol{x}) \frac{\partial u}{\partial x_i}. \tag{4.64}$$

The scalar field $u(\boldsymbol{x}, t_k)$ can be written in the new coordinate system $\boldsymbol{y}$ as

$$u(\boldsymbol{x}, t_k) = u(\boldsymbol{\Gamma}^{-1}\boldsymbol{y}, t_k) = U(\boldsymbol{y}, t_k) = U(\boldsymbol{\Gamma}\boldsymbol{x}, t_k) \tag{4.65}$$

which implies that

$$\frac{\partial u(\boldsymbol{x}, t)}{\partial x_j} = \sum_{k=1}^{d} \Gamma_{kj} \frac{\partial U(\boldsymbol{y}, t_k)}{\partial y_k}. \tag{4.66}$$

In this way, we can rewrite the operator (4.64) in coordinates $\boldsymbol{y} = \boldsymbol{\Gamma}\boldsymbol{x}$ as

$$\begin{aligned} G_{\boldsymbol{\Gamma}}\left(U(\boldsymbol{y}, t), \boldsymbol{y}\right) &= \sum_{i,j=1}^{d} \Gamma_{ij} f_j\left(\boldsymbol{\Gamma}^{-1}\boldsymbol{y}\right) \frac{\partial U(\boldsymbol{y}, t)}{\partial y_i} \\ &= \sum_{i=1}^{d} h_i\left(\boldsymbol{y}\right) \frac{\partial U(\boldsymbol{y}, t)}{\partial y_i}, \end{aligned} \tag{4.67}$$

where

$$h_i\left(\boldsymbol{y}\right) = \sum_{j=1}^{d} \Gamma_{ij} f_j\left(\boldsymbol{\Gamma}^{-1}\boldsymbol{y}\right). \tag{4.68}$$

Note that $G_{\boldsymbol{\Gamma}}$ has a relatively simple form due to the linearity[4] of the coordinate transformation. In this case, the rank of the operators G and $G_{\boldsymbol{\Gamma}}$ are determined by the FTT ranks of the variable

[4]For more general nonlinear coordinate transformations $\boldsymbol{y} = \boldsymbol{H}(\boldsymbol{x})$, the operator $G_{\boldsymbol{\Gamma}}$ includes the metric tensor of the coordinate change, which can significantly complicate the form of $G_{\boldsymbol{\Gamma}}$ (e.g., [2, 71]).

coefficients $f_i(\boldsymbol{x})$ and $h_i(\boldsymbol{y})$ $(i = 1, 2, \dots, d)$, respectively.

At any time step t_k during temporal integration of (1.1) using the methods discussed in Chapter 3 (e.g., the dynamical approximation (3.6) or step-truncation scheme (3.35)) we may compute a rank-reducing coordinate transformation $\boldsymbol{y} = \boldsymbol{\Gamma}\boldsymbol{x}$ and obtain a tensor ridge representation of the solution at time t_k. To integrate the initial boundary value problem (1.1) using the tensor ridge representation of the solution at time t_k, the operator G_{TT} may be rewritten as a new (FTT) operator $G_{\text{TT},\boldsymbol{\Gamma}}$ acting in the transformed coordinate system. With the operator $G_{\text{TT},\boldsymbol{\Gamma}}$ available, we can write the following PDE for $U(\boldsymbol{y}, t)$ corresponding to (1.1)

$$\begin{cases} \dfrac{\partial U(\boldsymbol{y}, t)}{\partial t} = G_{\text{TT},\boldsymbol{\Gamma}}(U(\boldsymbol{y}, t)), \qquad t \geq t_k, \\ U(\boldsymbol{y}, t_k) = v_{\text{TT}}(\boldsymbol{y}, t_k), \end{cases} \tag{4.69}$$

with initial condition given at time t_k. Time integration can then proceed in the transformed coordinate system using the PDE (4.69). It is well-known that the computational cost of the low-rank tensor integrators presented in Chapter 3 scales linearly in the problem dimension d and polynomially in the tensor rank of the solution and operator. However, in order to determine an optimal coordinate transformation for reducing the overall cost of temporal integration it is necessary to have more precise estimates on the computational cost of one time step. Such computational cost depends on many factors, e.g., the tensor integration scheme used, the separation rank of the PDE operator $G_{\boldsymbol{\Gamma}}$, the operator splitting (3.2) used, the FTT rank of the PDE solution $U(\boldsymbol{y}, t_k)$ at time t_k, the rank of the operator applied to the solution after truncation $\mathfrak{T}_\delta\left(G_{\boldsymbol{\Gamma}}(U(\boldsymbol{y}, t_k))\right)$, etc. From this observation it is clear that in order to obtain an optimal coordinate transformation for reducing the overall computational cost of temporal integration, the transformation must take into consideration both the solution rank and the operator rank. Determining a coordinate transformation $\mathbf{\Gamma}$ that controls the separation rank of a general operator $G_{\text{TT},\mathbf{\Gamma}}$ is a non-trivial problem that we do not address in this dissertation.

4.3.1 Coordinate-adaptive time integration

Next we develop coordinate-adaptive time integration schemes for PDEs on FTT manifolds that are designed to control the solution rank, the PDE operator rank, or the rank of the right hand side of the PDE. The first coordinate-adaptive algorithm (Algorithm 4) is designed to attempt a rank reducing coordinate transformation if the computational cost of time integration in the current coordinate system exceeds a predetermined threshold. The computational cost of one time step may be measured in different ways, e.g., by the CPU-time it takes to perform one time step, by the rank of the solution, or by the rank of the right hand side of the PDE (operator applied to the solution). In the second to last line of Algorithm 4, "time" denotes the computational time it takes to compute one time step and "rank" denotes either the solution rank, the rank of the PDE right hand side, or the maximum of the two.

The second algorithm (Algorithm 5) we propose for coordinate-adaptive tensor integration of PDEs is based on computing a small correction of the coordinate system at every time step. In practice, we compute one ϵ-step of (4.38) at every time step during temporal integration of the given PDE. This yields a PDE in which the operator (which depends on the coordinate system) changes at every time step, i.e., a time-dependent operator induced by the time-dependent coordinate change.

Hereafter we apply these coordinate-adaptive algorithms to five different PDEs and compare the results with conventional FTT integrators in fixed Cartesian coordinates. All numerical simulations were run in Matlab 2022a on a 2021 MacBook Pro with M1 chip and 16GB RAM, spatial derivatives and integrals were approximated with one-dimensional Fourier pseudo-spectral differentiation matrices and quadrature rules [45], and various explicit step-truncation time integration schemes were used.

4.3.2 2D linear advection equations

First we apply coordinate-adaptive tensor integration to the 2D linear advection equation

$$\begin{cases} \dfrac{\partial u(\boldsymbol{x},t)}{\partial t} = f_1(\boldsymbol{x})\dfrac{\partial u(\boldsymbol{x},t)}{\partial x_1} + f_2(\boldsymbol{x})\dfrac{\partial u(\boldsymbol{x},t)}{\partial x_2}, \\ u(\boldsymbol{x},0) = u_0(\boldsymbol{x}), \end{cases} \tag{4.70}$$

with two different sets of coefficients $f_i(\boldsymbol{x})$ specified hereafter. Each example is designed to demonstrate different features of the proposed coordinate-adaptive algorithms (Algorithm 4 and Algorithm 5).

In the first example, we generate the vector field $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))$ via the two-dimensional stream function [110]

$$\psi(x_1, x_2) = \Theta(x_1)\Theta(x_2) \tag{4.71}$$

with

$$\Theta(x) = \frac{\cos(\alpha x/L)}{\cos(\alpha/2)} - \frac{\cosh(\alpha x/L)}{\cosh(\alpha/2)}, \tag{4.72}$$

$L = 30$ and $\alpha = 4.73$. Such a stream function generates the divergence-free vector field (see

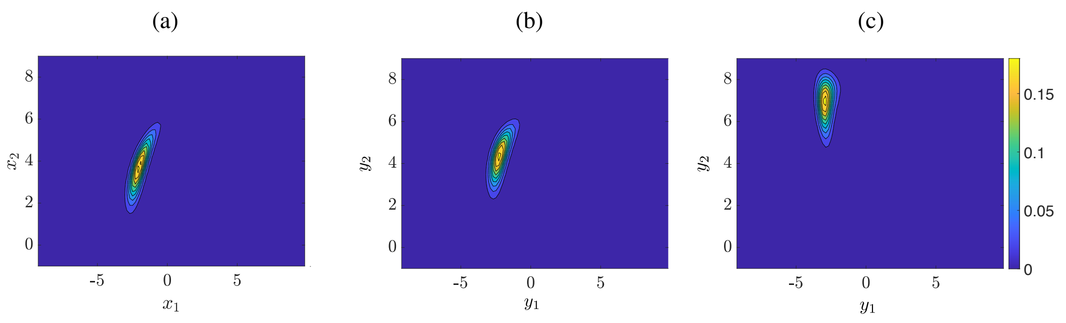


Figure 4.7: Solution to the linear advection equation (4.70) with coefficients (4.73) at time $t = 30$. (a) Cartesian coordinates, (b) tensor ridge simulation 1 using on Algorithm 4 with $\Delta\epsilon = 10^{-4}$, $M_{\text{iter}} = 2000$, and $\eta = 10^{-1}$, (c) tensor ridge simulation 2 using Algorithm 5 with $M_{\text{iter}} = 1$ and $\Delta\epsilon = 10^{-3}$. In all simulations the truncation tolerance is set to $\delta = 10^{-6}$.

Figure 4.9(a))

$$f_1(\boldsymbol{x}) = \frac{\partial \psi}{\partial x_2}, \qquad f_2(\boldsymbol{x}) = -\frac{\partial \psi}{\partial x_1}. \tag{4.73}$$

We set the initial condition

$$u_0(\boldsymbol{x}) = \frac{1}{m} \exp\left(-4(x_1-2)^2\right) \exp\left(-\frac{(x_2-2)^2}{2}\right), \tag{4.74}$$

where

$$m = \left\| \exp\left(-4(x_1-2)^2\right) \exp\left(-\frac{(x_2-2)^2}{2}\right) \right\|_{L^2(\Omega)} \tag{4.75}$$

is a normalization constant.

We first ran one FTT tensor simulation in fixed Cartesian coordinates. We then ran two coordinate-adaptive simulations that use rank-reducing coordinate transformations during time integration. In the first coordinate-adaptive simulation we use Algorithm 4 with $\max \operatorname{rank} = 15$ and $k_r = 0$ to initialize coordinate transformations during time integration. For the Riemannian gradient descent algorithm that computes the rank reducing coordinate transformation we set step-size $\Delta\epsilon = 10^{-4}$, maximum number of iterations $M_{\text{iter}} = 2000$ and stopping tolerance $\eta = 10^{-1}$. In the second coordinate-adaptive simulation we use Algorithm 5 with $M_{\text{iter}} = 1$ and $\Delta\epsilon = 10^{-3}$, i.e., the integrator performs one step of time integration followed by one step of the Riemannian gradient descent algorithm 2. In both coordinate-adaptive simulations we set the truncation threshold $\delta = 10^{-6}$.

In Figure 4.7 we plot the solutions obtained from each of the three simulations at time $t = 30$. In order to check the accuracy of integrating the PDE solution in the low-rank coordinate system we mapped the transformed solution back to Cartesian coordinates using a two-dimensional trigonometric interpolant and compared with the solution computed in Cartesian coordinates. In both coordinate-adaptive simulations we found that the global L^{∞} error is bounded by 8×10^{-4}, suggesting that the coordinate transformation does not affect accuracy significantly. In Figure 4.8 we plot the solution rank and the rank of the of the right hand side of the PDE (4.70) versus time for all three tensor simulations. We observe that in the coordinate-adaptive tensor ridge simulations the ranks of both the solution and the PDE right hand side are less than or equal to the corresponding ranks in Cartesian coordinates. We also observe that the adaptive simulation based on Algorithm 5 (denoted by “tensor ridge 2” in Figure 4.8) has significantly smaller rank than the adaptive simulation based on Algorithm 4 (denoted by “tensor ridge 1” in Figure 4.8).

Next, we demonstrate that linear coordinate transformations can be used to reduce the rank of a PDE operator and reduce the overall computational cost of temporal integration. To this end, consider again the two-dimensional advection equation (4.70) this time with advection

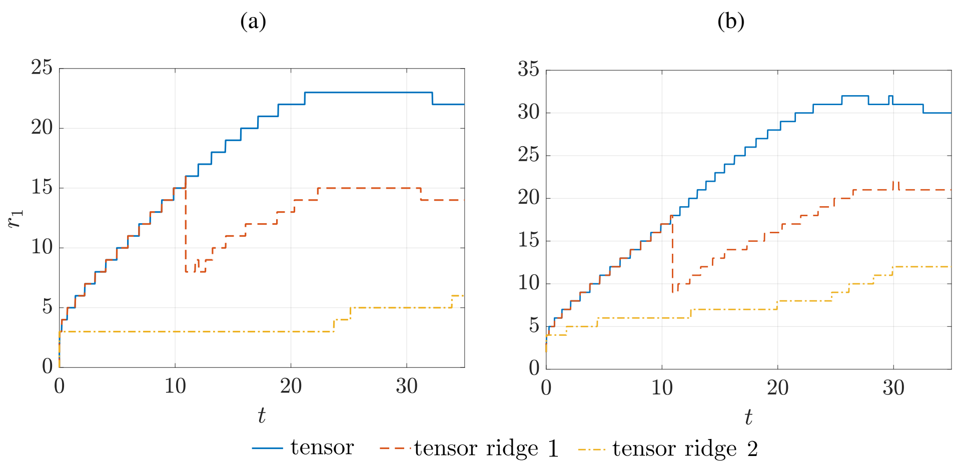


Figure 4.8: Rank (number of singular values larger than $\delta = 10^{-6}$) of the PDE solution (a) and velocity vector (b) (RHS of the PDE) for the advection problem (4.70) with coefficients given by (4.73). Tensor ridge 1 was computed using the coordinate adaptive algorithm 4 using a maximum solution rank threshold of 15 and tensor ridge 2 was computed using algorithm 5 which performs coordinate corrections at each time step.

coefficients

$$\begin{aligned} f_1(\boldsymbol{x}) &= \exp\left(-a_1 \left(\boldsymbol{R}_1 \cdot \boldsymbol{x}\right)^2\right) \exp\left(-a_2 \left(\boldsymbol{R}_2 \cdot \boldsymbol{x}\right)^2\right), \\ f_2(\boldsymbol{x}) &= \exp\left(-b_1 \left(\boldsymbol{R}_1 \cdot \boldsymbol{x}\right)^2\right) \exp\left(-b_2 \left(\boldsymbol{R}_2 \cdot \boldsymbol{x}\right)^2\right), \end{aligned} \tag{4.76}$$

where $\boldsymbol{R}_i$ is the i^{th} row of the matrix $\boldsymbol{R}$,

$$\boldsymbol{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{4.77}$$

We set parameters $\theta = \pi/4$,

$$a_1 = 1/20, \quad a_2 = 1/10, \quad b_1 = 1/10, \quad b_2 = 1/20, \tag{4.78}$$

and initial condition

$$u_0(\boldsymbol{x}) = \exp\left(-x_1^2/3\right) \exp\left(-x_2^2/3\right). \tag{4.79}$$

In Figure 4.9(b) we plot the vector field $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))$ defined by (4.76). Note that the initial condition $u_0(\boldsymbol{x})$ is rank 1. The rank of the linear advection operator defined on

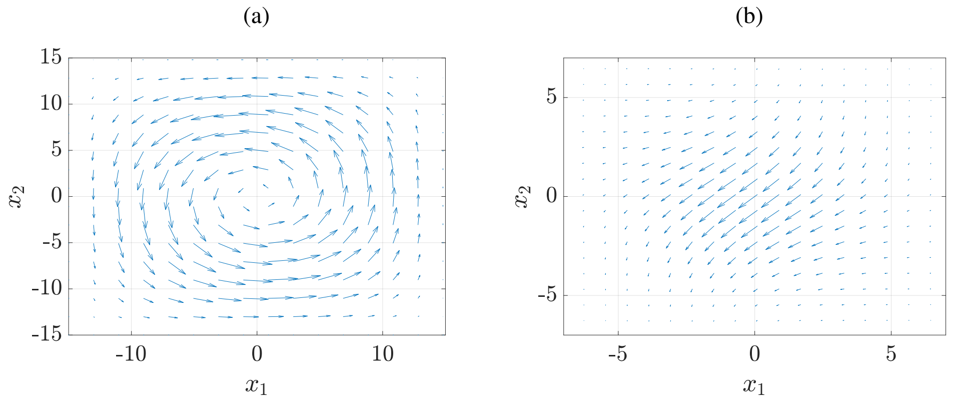


Figure 4.9: (a) Vector fields used as coefficients in the two-dimensional linear advection equation (4.70). The vector field defined in (4.73) is shown in (a) and the vector field defined in (4.76) is shown in (b).

the right hand side of (4.70) depends on the FTT truncation tolerance used to compress the multivariate functions $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$. If we choose the coordinate transformation $\boldsymbol{y} = \boldsymbol{\Gamma}\boldsymbol{x}$, where $\boldsymbol{\Gamma} = \boldsymbol{R}^{-1}$, then the initial condition remains rank 1, but the rank of the advection operator at the right hand side of (4.70) becomes 2, regardless of the FTT truncation tolerance used.

We ran two simulations of (4.70) with coefficients (4.76) using the step-truncation FTT integrator (3.35) based on Adams-Bashforth 3 with step-size $\Delta t = 10^{-3}$, truncation tolerance $\delta = 10^{-6}$, and final integration time $t = 5$. In the first simulation we solved the PDE with a step-truncation tensor method in fixed Cartesian coordinates. In the second simulation we solved the PDE in coordinates $\boldsymbol{y} = \boldsymbol{R}^{-1}\boldsymbol{x}$, using the same step-truncation tensor method. In order to verify the accuracy of our FTT simulations we also computed a benchmark solution on a full tensor product grid in two dimensions. We then mapped the transformed solution back to Cartesian coordinates at each time step and compared it with the benchmark solution. We found that the global L^∞ error of both low-rank simulations is bounded by 8×10^{-4}. In Figure

4.10 we plot the FTT solution in Cartesian coordinates and the FTT-ridge solution in low-rank coordinates at time $t = 5$. We observe that the PDE operator expressed in Cartesian coordinates advects the solution at an angle relative to the underling coordinate system while the operator expressed in coordinates $\boldsymbol{y} = \boldsymbol{R}^{-1}\boldsymbol{x}$ advects the solution directly along a coordinate axis, hence the low rank dynamics. In Figure 4.12(a) we plot the solution ranks versus time. Note that even though the operator in Cartesian coordinates has significantly larger rank than the operator in low-rank coordinates, the solution ranks follow the same trend during temporal integration with the FTT-ridge rank only slightly smaller than the FTT rank.

Computational cost The CPU-time of integrating the advection equation (4.70) with coefficients (4.76) from $t = 0$ to $t = 5$ is 72 seconds when computed with FTT in Cartesian coordinates and 31 seconds when computed with FTT-ridge in low-rank coordinates. Note that although the ranks of the low-rank solutions are similar at each time step (Figure 4.12(a)), the computational speed-up of the FTT-ridge simulation is due to the operator rank, which is 2 for FTT-ridge and 16 for FTT in Cartesian coordinates.

4.3.3 Allen-Cahn equation

Next we demonstrate coordinate-adaptive tensor integration on a simple nonlinear PDE. The Allen-Cahn eqaution is a reaction-diffusion PDE, which, in its simplest form includes a low-order polynomial non-linearity (reaction term) and a diffusion term [55]

$$\begin{cases} \dfrac{\partial u(\boldsymbol{x},t)}{\partial t} = \alpha \Delta u(\boldsymbol{x},t) + u(\boldsymbol{x},t) - u(\boldsymbol{x},t)^3, \\ \\ u(\boldsymbol{x},0) = u_0(\boldsymbol{x}). \end{cases} \tag{4.80}$$

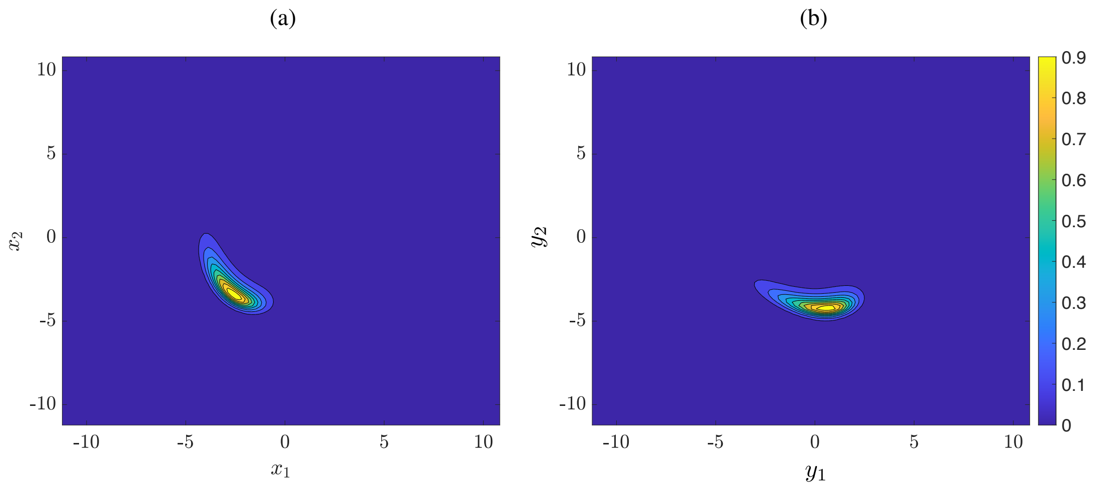


Figure 4.10: Solution to the two-dimensional advection equation (4.70) with coefficients (4.76) at time $t = 5$. (a) FTT solution computed in Cartesian coordinates. (b) FTT-ridge computed low-rank coordinates.

In two spatial dimensions the Laplacian in coordinates $\boldsymbol{y} = \boldsymbol{\Gamma}\boldsymbol{x}$ is given by

$$\Delta_{\boldsymbol{\Gamma}} = \left(\Gamma_{11}^2 + \Gamma_{12}^2\right) \frac{\partial^2}{\partial y_1^2} + \left(\Gamma_{21}^2 + \Gamma_{22}^2\right) \frac{\partial^2}{\partial y_2^2} + 2\left(\Gamma_{11}\Gamma_{21} + \Gamma_{12}\Gamma_{22}\right) \frac{\partial^2}{\partial y_1 \partial y_2} \tag{4.81}$$

which allows us to write the nonlinear PDE (4.80) in the coordinate system $\boldsymbol{y} = \boldsymbol{\Gamma}\boldsymbol{x}$ with only a small increase in the rank of the Laplacian operator[5]. The FTT rank of the cubic term appearing in the PDE operator of the Allen-Cahn equation (4.80) is determined by the rank of the FTT solution. Standard algorithms for multiplying two FTT tensors u_{TT} and v_{TT} with ranks $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ results in a FTT tensor with (non-optimal) rank equal to the Hadamard product of the two ranks $\boldsymbol{r}_1 \circ \boldsymbol{r}_2$. Hence, by reducing the solution rank with a coordinate transformation, we can reduce the computational cost of computing the nonlinear term in (4.80). We set the diffusion coefficient $\alpha = 0.2$ and the initial condition $u_0(\boldsymbol{x})$ as the rotated Gaussian from equation (4.18) with $\epsilon = \pi/3$.

[5]In general a d-dimensional Laplacian Δ is a rank-d operator and the corresponding operator $\Delta_{\boldsymbol{\Gamma}}$ in (linearly) transformed coordinates is rank $(d^2 + d)/2$.

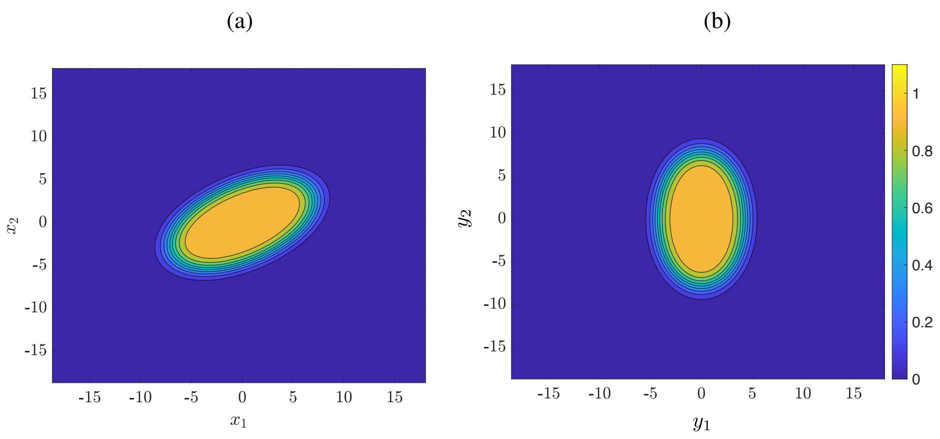


Figure 4.11: Solution to the two-dimensional Allen-Cahn equation (4.80) at time $t = 5$. (a) FTT solution computed in Cartesian coordinates and (b) FTT-ridge solution computed in low-rank coordinates.

We ran two FTT simulations of (4.80) for time $t \in [0, 5]$. The first simulation was computed in fixed Cartesian coordinates. In the second simulation we used the coordinate transformation $\boldsymbol{\Phi}(\pi/3)^{-1}$ (which in this case we have available analytically) to transform the initial condition into a rank 1 FTT-ridge function. We integrated the rank 1 FTT-ridge initial condition forward in time using the corresponding transformed PDE, i.e., using the transformed Laplacian (4.81). In order to verify the accuracy of our low-rank simulations we also computed a benchmark solution on a full tensor product grid in two dimensions and mapped the transformed solution back to Cartesian coordinates at each time step. We found that the global L^∞ error of both low-rank solutions is bounded 5×10^{-5}. In Figure 4.11 we plot the FTT solution in Cartesian coordinates and the FTT-ridge solution in low-rank coordinates at time $t = 5$. We observe that the profile of Gaussian functions are preserved as the solution moves from the unstable equilibrium at $u = 0$ to the stable equilibrium at $u = 1$. In Figure 4.12(b) we plot the solution ranks versus time. We observe that the FTT solution rank is larger than the FTT-ridge solution rank at each time.

Computational cost The CPU-time of integrating the Allen-Cahn equation (4.80) from $t = 0$ to $t = 5$ is 279 seconds when computed using FTT in Cartesian coordinates and 72 seconds when computed using FTT-ridge in low-rank coordinates. The optimal coordinate transformation at time $t = 0$ is known analytically so we do not need to compute it, thus these computational timings only include the temporal integration, and do not account for any computational time of changing the coordinate system. A significant amount of computational time in computing the FTT solutions comes from computing the cubic nonlinearity appearing in the Allen-Cahn equation at each time step. The lower rank FTT-ridge solution allows for this term to be computed significantly faster than the FTT solution in Cartesian coordinates.

4.3.4 3D and 5D linear advection equations

We also applied the rank-reducing coordinate-adaptive FTT integrators to the advection equation

$$\begin{cases} \dfrac{\partial u(\boldsymbol{x},t)}{\partial t} = \boldsymbol{f}(\boldsymbol{x}) \cdot \nabla u(\boldsymbol{x},t), \\ \\ u(\boldsymbol{x},0) = u_0(\boldsymbol{x}), \end{cases} \tag{4.82}$$

in dimensions three and five with initial condition $u_0(\boldsymbol{x})$ defined as a Gaussian mixture

$$u_0(\boldsymbol{x}) = \frac{1}{m} \sum_{i=1}^{N_g} \exp\left(- \sum_{j=1}^{d} \frac{1}{\beta_{ij}} \left(\boldsymbol{R}_j^{(i)} \cdot \boldsymbol{x} + t_{ij} \right)^2 \right), \tag{4.83}$$

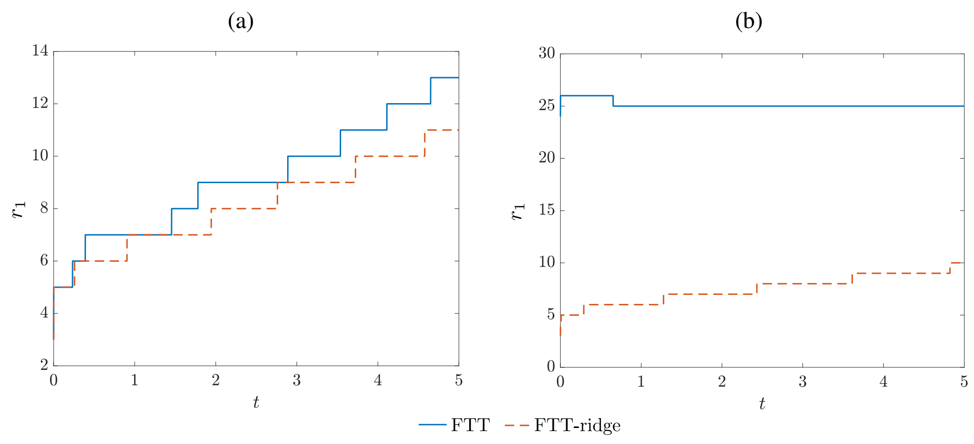


Figure 4.12: (a) Rank versus time for the FTT and FTT-ridge solutions to the advection PDE (4.70) with coefficients (4.76). (b) Rank versus time for the FTT and FTT-ridge solutions to the Allen-Cahn equation (4.80).

where $\boldsymbol{R}_j^{(i)}$ is the j-th row of a $d \times d$ rotation matrix $\boldsymbol{R}^{(i)}$, $\beta_i \geq 0$, t_{ij} are translations and m is the normalization factor

$$m = \left\| \sum_{i=1}^{N_g} \exp\left(-\sum_{j=1}^{d} \frac{1}{\beta_{ij}} \left(\boldsymbol{R}_j^{(i)} \cdot \boldsymbol{x} + t_{ij} \right)^2 \right) \right\|_{L^1(\mathbb{R}^d)} . \tag{4.84}$$

4.3.4.1 Three-dimensional simulation results

First we consider three spatial dimensions ($d = 3$) and set the coefficients in (4.82) as

$$\boldsymbol{f}(\boldsymbol{x}) = -\frac{1}{6} \begin{bmatrix} 2\sin(x_2) \\ 3\cos(x_3) \\ 3x_1 \end{bmatrix}, \tag{4.85}$$

resulting in the linear operator

$$\boldsymbol{f}(\boldsymbol{x}) \cdot \nabla = \frac{\sin(x_2)}{3} \frac{\partial}{\partial x_1} + \frac{\cos(x_3)}{2} \frac{\partial}{\partial x_2} + \frac{x_1}{2} \frac{\partial}{\partial x_3}. \tag{4.86}$$

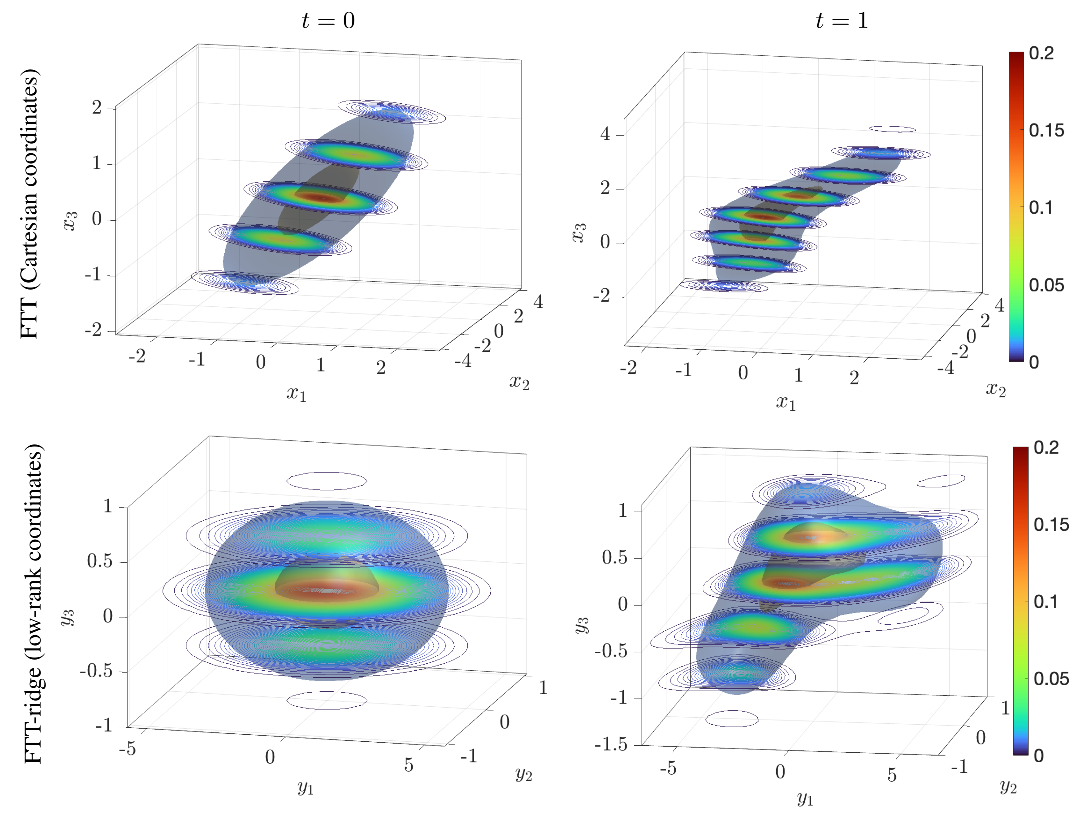


Figure 4.13: Volumetric plot of the FTT solutions to the 3D advection equation (4.82) at time $t = 0$ (left column) in Cartesian coordinates (top) and low-rank coordinates (bottom), and at time $t = 1$ (right column) in Cartesian coordinates (top) and low-rank coordinates (bottom).

In a previous work [29] we have demonstrated that variable coefficient advection problems with operators of the form (4.86) can have solutions with multilinear rank that grows significantly over time. We set the parameters in the initial condition (4.83) $N_g = 1$,

$$\boldsymbol{R}^{(1)} = \exp\left(\frac{1}{28}\begin{bmatrix} 0 & 7\pi & 4\pi \\ -7\pi & 0 & 7\pi \\ -4\pi & -7\pi & 0 \end{bmatrix}\right), \qquad \boldsymbol{\beta} = \begin{bmatrix} 3 & 1/10 & 3 \end{bmatrix}, \tag{4.87}$$

and $t_{ij} = 0$ for all i, j.

We ran three FTT simulations up to time $t = 1$. The first simulation is computed in

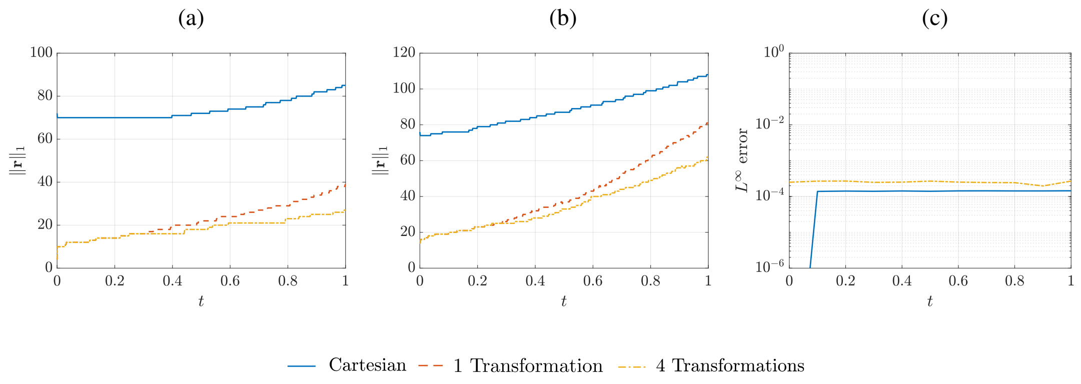


Figure 4.14: 3D advection equation (4.82): Multilinear rank of the PDE solution (a) and PDE right-hand-side (b) in Cartesian coordinates and transformed coordinates versus time. In (c) we plot the L^∞ error of the FTT solutions relative to a benchmark solution.

fixed Cartesian coordinates. We then tested the FTT integrator with rank-reducing coordinate transformation in two different simulation settings. In the first one, we performed a coordinate transformation only at time $t = 0$. Such a coordinate transformation is not done using the Riemannian gradient descent algorithm, since, in this case we have the optimal coordinate transformation available analytically and we can simply evaluate the FTT tensor on the low rank coordinates. In the second simulation we also performed a coordinate transformation at time $t = 0$ (once again the coordinate transformation at time $t = 0$ is not done using the Riemannian gradient descent algorithm) and then used the coordinate-adaptive integration Algorithm 4 with $\max \text{rank} = 15$ and $k_r = 5$. With these parameters the coordinate-adaptive algorithm triggers three additional coordinate transformations at times $t \in \{0.25, 0.59, 0.9\}$ that are computed using the Riemannian gradient descent algorithm 2 with step size $\Delta\epsilon = 10^{-4}$ and stopping tolerance $\eta = 10^{-1}$. In Figure 4.15(c) we plot the absolute value of the derivative of the cost function $C(\mathbf{\Gamma}(\epsilon))$ versus ϵ for the instances of gradient descent at times $t > 0$. We observe that the rate of change of the cost function becomes smaller as we iterate the gradient descent routine, i.e., the cost function is decreasing less per iteration after several iterations. This indicates that the cost function is approaching a flatter region and our gradient descent method is becoming less effective for reducing the cost function. In Figure 4.14(a) we plot the 1-norm of the FTT solution rank vector versus time and in Figure 4.14(b) we plot the 1-norm of the FTT solution velocity (i.e., the PDE right hand side) for each FTT simulation. We observe that the FTT-ridge solutions and right hand side of the PDE have rank that is smaller than the corresponding ranks of the FTT solution in Cartesian coordinates. We also observe that the adaptive coordinate transformations performed at times $t > 0$ do not reduce the solution rank at the time of application, but they do slow the rank increase as time integration proceeds. In Figure 4.15(a)-(b) we plot the singular values of the FTT solutions at final time and note that both FTT-ridge solutions have singular values that decay significantly faster than the FTT solution in Cartesian coordinates. Moreover, the additional coordinate transformations performed by the coordinate-adaptive integrator causes the singular values of the FTT-ridge solution to decay faster than the other FTT-ridge solution that used only one coordinate transformation at $t = 0$. In Figure 4.13 we provide volumetric plots of the PDE solution in Cartesian coordinates and in the transformed coordinate system computed with the coordinate-adaptive algorithm 4 at time $t = 1$. We observe that the reduced rank tensor ridge solution appears to be more symmetrical with respect to the underlying coordinate axes than the solution in Cartesian coordinates.

It is important to note that the operator $G(\cdot, \boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) \cdot \nabla$ in (3.82) is a separable operator of rank $\boldsymbol{g} = \begin{bmatrix} 1 & 3 & 3 & 1 \end{bmatrix}$. For a general linear coordinate transformation $\boldsymbol{\Gamma}$ the operator $G_{\boldsymbol{\Gamma}}$ (see (4.67)) acting in the transformed coordinate system can be obtained using trigonometric identities with (non-optimal) separation ranks. A more efficient representation

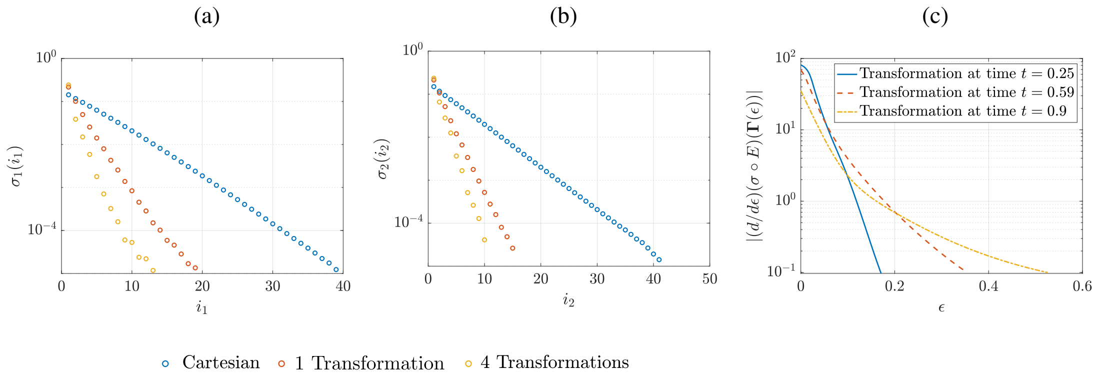


Figure 4.15: (a)-(b) Multilinear spectra of the FTT solutions to the 3D advection equation (4.82) at time $t = 1$ in Cartesian coordinates and in low-rank coordinates. (c) Absolute value of the derivative of the cost function in (4.20) during gradient descent of the 3D advection equation (4.82) solutions at times $t > 0$.

of $G_{\boldsymbol{\Gamma}}$ can be obtained by FTT compression and then splitting $G_{\boldsymbol{\Gamma}}$ (3.2) into a sum of three operators

$$G_{\boldsymbol{\Gamma}} = G_{\boldsymbol{\Gamma}}^{(1)} + G_{\boldsymbol{\Gamma}}^{(2)} + G_{\boldsymbol{\Gamma}}^{(3)}, \tag{4.88}$$

where $G_{\boldsymbol{\Gamma}}^{(i)}$ have ranks $\boldsymbol{g}_{\boldsymbol{\Gamma}}^{(i)} = \begin{bmatrix} 1 & 4 & 4 & 1 \end{bmatrix}$ for each $i = 1, 2, 3$. Then we apply the operator $G_{\boldsymbol{\Gamma}}$ to the FTT-ridge solution at each time using the procedure summarized in (3.3). In this case, since the PDE operator G in Cartesian coordinates is separable and low-rank, it is not surprising that a linear coordinate transformation increases the PDE operator rank. However by splitting the operator such as (4.88) and performing a FTT truncation operation after applying each lower rank operator $G_{\boldsymbol{\Gamma}}^{(i)}$ we can mitigate the computational cost.

In Figure 4.14(c) we plot the L^∞ error of the transformed solutions and the FTT solution in Cartesian coordinates relative to a benchmark solution. We observe that the L^∞ error of our coordinate-adaptive solution is very close to the L^∞ error of the FTT solution computed in Cartesian coordinates. This implies that the error incurred by transforming coordinates, integrating the PDE in the new coordinate system, and then transforming coordinates back is not significant.

Computational cost The CPU-time of integrating the three-dimensional advection equation (3.79) from $t = 0$ to $t = 1$ is 413 seconds when computed using FTT in Cartesian coordinates, 173 seconds when computed using FTT-ridge in low-rank coordinates with one coordinate transformation at $t = 0$, and 299 seconds when computed using the coordinate-adaptive FTT Algorithm 4. These timings do not include the coordinate transformations at time $t = 0$ since they were not performed using Riemannian gradient descent. The timings do include the computation of the new coordinate systems at times $t > 0$.

4.3.4.2 Five-dimensional simulation results

Finally, we consider the advection equation (4.82) in dimension five ($d = 5$) with coefficients

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} -x_2 \\ x_3 \\ x_5 \\ -x_2 \\ -x_3 \end{bmatrix}. \tag{4.89}$$

This allows us to test our coordinate adaptive algorithm for a case in which we know the optimal ridge matrix. In the initial condition (3.85) we set the following parameters: $N_g = 2$,

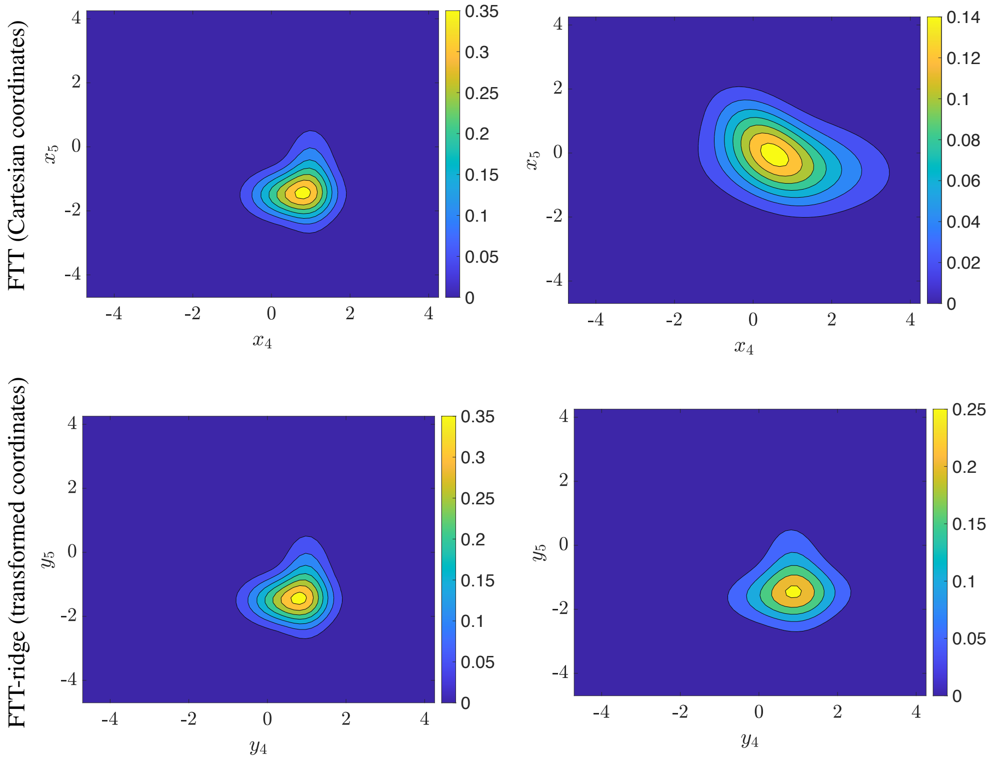


Figure 4.16: Marginal PDFs of the solution to the 5D advection equation (4.82) computed in Cartesian coordinates and low-rank adaptive coordinates at time $t = 0$ and time $t = 1$.

$$\boldsymbol{R}^{(1)} = \boldsymbol{R}^{(2)} = \boldsymbol{I}_{5\times 5},$$
$$\boldsymbol{\beta} = \begin{bmatrix} 1/2 & 2 & 1/2 & 3 & 1/2 \\ 1 & 1/3 & 2 & 1 & 1/2 \end{bmatrix}, \tag{4.90}$$

and

$$\boldsymbol{t} = \begin{bmatrix} 1 & 1 & 1 & -1 & 1 \\ 0 & 0 & 3/2 & -1/2 & 1/2 \end{bmatrix}, \tag{4.91}$$

which results in an initial condition with FTT rank $\begin{bmatrix} 1 & 2 & 2 & 2 & 2 & 1 \end{bmatrix}$. Due to the choice of coefficients (4.89), the analytical solution to the PDE (3.79) can be written as a ridge function

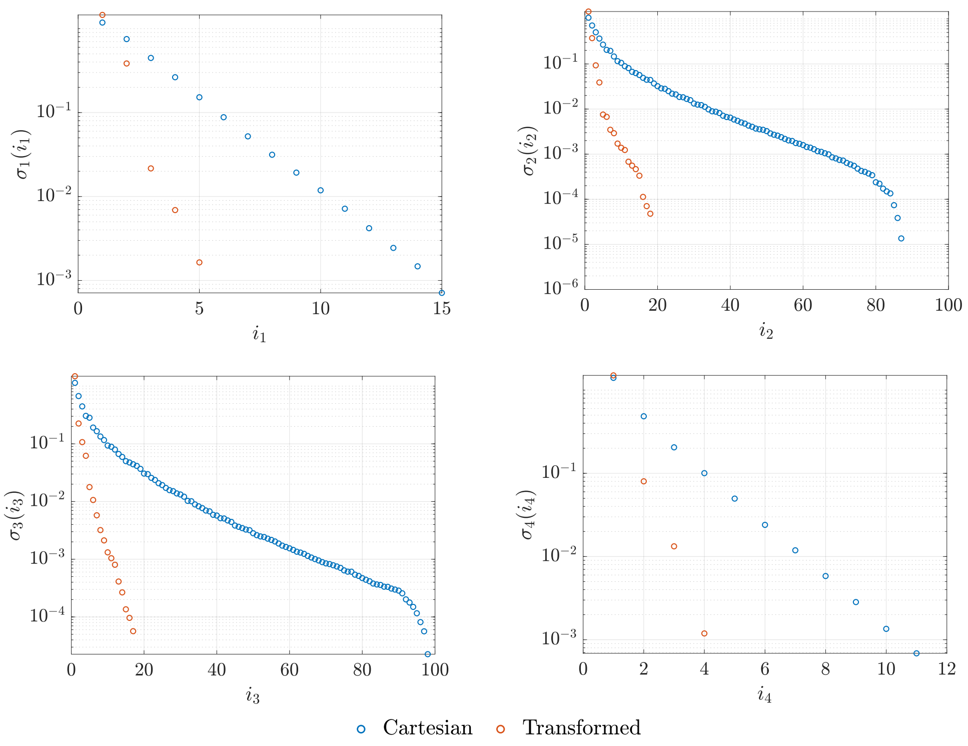


Figure 4.17: Multilinear spectra of the solution to the advection 5D PDE (4.82) in Cartesian coordinates and transformed coordinates at time $t = 1$.

in terms of the PDE initial condition

$$u(\boldsymbol{x}, t) = u_0\left(e^{t\boldsymbol{B}}\boldsymbol{x}\right), \tag{4.92}$$

where

$$\boldsymbol{B} = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}. \tag{4.93}$$

Thus (4.92) is a tensor ridge solution to the 5D advection equation (4.82) with the same rank as the initial condition, i.e., in this case there exists a tensor ridge solution at each time with FTT rank equal to $\begin{bmatrix} 1 & 2 & 2 & 2 & 2 & 1 \end{bmatrix}$.

We ran two simulations of the PDE (3.79) up to $t = 1$. The first simulation is computed with a step-truncation method in fixed Cartesian coordinates. The second simulation is computed with the coordinate-adaptive FTT-ridge tensor method that performs coordinate corrections at each time step (Algorithm 5) with $\Delta\epsilon = 5 \times 10^{-4}$ and $M_{\text{iter}} = 1$. In Figure 4.16 we plot marginal PDFs of the FTT solution and the FTT-ridge solution at initial time $t = 0$ and final time $t = 1$. We observe that the reduced rank FTT-ridge solution appears to be more symmetrical with respect to the coordinate axes than the corresponding function on Cartesian coordinates. In Figure 4.18(a) we plot the 1-norm of the FTT solution rank versus time and in Figure 4.18(b) we plot the FTT rank of the PDE velocity vector (right hand side of the PDE) versus time for both FTT solutions. We observe that the FTT solution rank in Cartesian coordinates grows quickly compared to the FTT-ridge solution in Cartesian coordinates. This is expected due to the existence of a low-rank FTT-ridge solution (4.92). Note that the coordinate-adaptive algorithm 5 produces a FTT-ridge solution with ridge matrix that is different than $e^{t\boldsymbol{B}}$ in (4.92). The reason can be traced back to the cost function we are minimizing, i.e., the Schauder norm (see section 4.2.1), and the fact that we do not fully determine the minimizer at each step, but rather perform only one ϵ-step in the direction of the Riemannian gradient. Although the rank of the FTT-ridge solution computed with algorithm 5 is larger than the rank of the analytical solution (4.92), the algorithm still controls the FTT solution rank during time integration. In Figure 4.17 we plot the multilinear spectra of the two FTT solutions at time $t = 1$. We observe

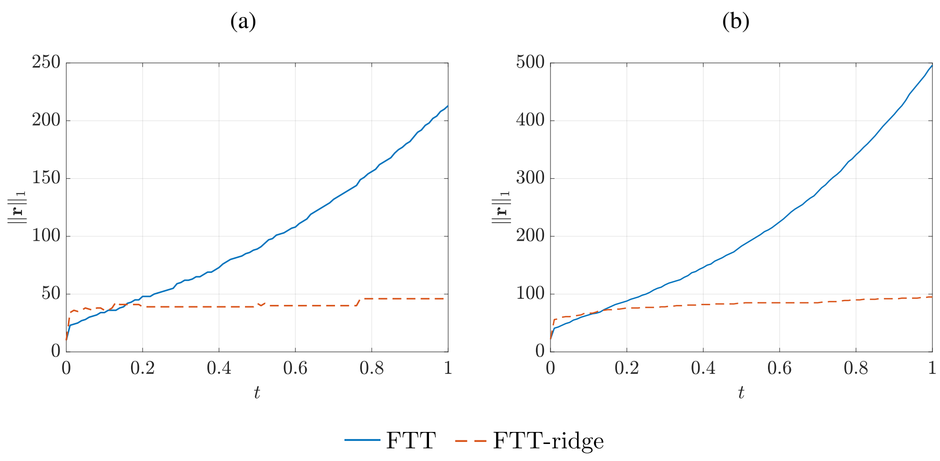


Figure 4.18: (a) 1-norm of the solution rank vectors versus time. (b) 1-norm of the solution velocity (PDE right hand side) rank vectors versus time.

that the multilinear spectra of the FTT-ridge function decay significantly faster than the spectra of the FTT solution in Cartesian coordinates.

For this problem it is not straightforward to compute a benchmark solution on a full tensor product grid. If we were to use the same resolution as the FTT solutions, i.e., 200 points in each dimension, then each time snapshot of the benchmark solution would be an array containing $200^5 \approx 3.2 \times 10^{11}$ double precision floating point numbers. This requires 2.56 terabytes of memory storage per time snapshot. In lieu of comparing our FTT solutions with a benchmark solution, we compared the two FTT solutions with each other. To do so we mapped the FTT-ridge solution back to Cartesian coordinates every 250 time steps by solving a PDE of the form (4.13) numerically. We compared the $(x4, x5)$-marginal PDFs of the two solutions and found that the global L^∞ norm of the difference of the two solution PDFs is bounded by 6×10^{-4}.

Computational cost The CPU-time of integrating the five-dimensional advection equation (4.82) from $t = 0$ to $t = 1$ is 2046 seconds when computed using FTT in Cartesian coordinates and 2035 seconds when computed using FTT-ridge in low-rank coordinates with coordinate corrections at each time step.

Appendix 4.A The Riemannian manifold of coordinate transformations

We endow the search space $\mathrm{SL}_d(\mathbb{R})$ in (4.31) with a Riemannian manifold structure. To this end, we first notice that $\mathrm{SL}_d(\mathbb{R})$ is a matrix Lie group over $\mathbb{R}$ and in particular is a smooth manifold. A point $\boldsymbol{A} \in \mathrm{SL}_d(\mathbb{R})$ corresponds to a linear coordinate transformation of $\mathbb{R}^d$ with determinant equal to 1. A smooth path $\boldsymbol{\Theta}(\epsilon)$ on the manifold $\mathrm{SL}_d(\mathbb{R})$ paramaterized by $\epsilon \in (-\delta, \delta)$ is a collection of smoothly varying linear coordinate transformations with determinant equal to 1 for all $\epsilon \in (-\delta, \delta)$. Denote by $\mathcal{C}^1\left((-\delta, \delta), \mathrm{SL}_d(\mathbb{R})\right)$ the collection of all continuously differentiable paths $\boldsymbol{\Theta}(\epsilon)$ on the manifold $\mathrm{SL}_d(\mathbb{R})$ parameterized by $\epsilon \in (-\delta, \delta)$. The tangent space of $\mathrm{SL}_d(\mathbb{R})$ at the point $\boldsymbol{A} \in \mathrm{GL}_d(\mathbb{R})$ is defined to be the collection of equivalence classes of velocities associated to all possible curves on $\mathrm{SL}_d(\mathbb{R})$ passing through the point $\boldsymbol{A}$

$$T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R}) = \left\{ \left.\frac{d\boldsymbol{\Theta}(\epsilon)}{d\epsilon}\right|_{\epsilon=0} : \quad \boldsymbol{\Theta} \in \mathcal{C}^1\left((-\delta, \delta), \mathrm{SL}_d(\mathbb{R})\right), \quad \boldsymbol{\Theta}(0) = \boldsymbol{A} \right\}. \tag{4.94}$$

It is well-known (e.g., [101]) that the tangent space of $\mathrm{SL}_d(\mathbb{R})$ at the point $\boldsymbol{A}$ is given by

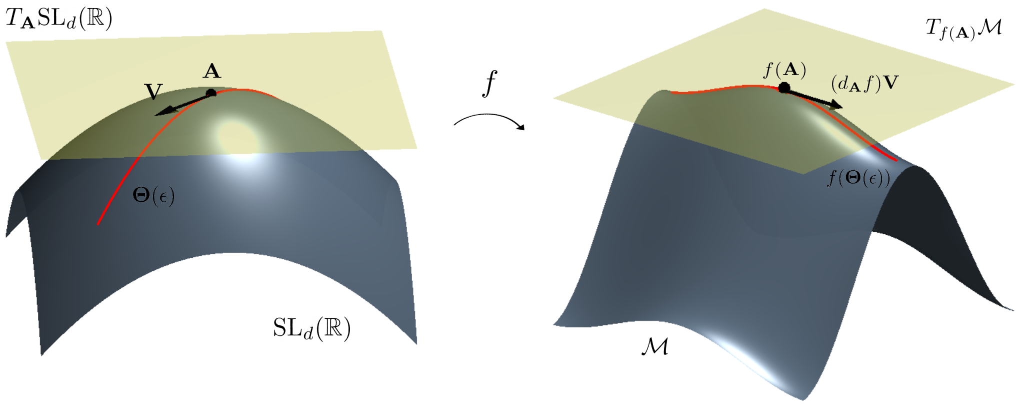
$$T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R}) = \mathfrak{sl}_d(\mathbb{R})\boldsymbol{A} = \{\boldsymbol{N}\boldsymbol{A} : \boldsymbol{N} \in \mathfrak{sl}_d(\mathbb{R})\}, \tag{4.95}$$

where $\mathfrak{sl}_d(\mathbb{R})$ denotes the collection of all $d \times d$ real matrices with vanishing trace. We can easily verify that if $\mathbf{\Theta}(\epsilon)$ is a smooth collection of matrices parameterized by $\epsilon \in (-\delta, \delta)$ with $\mathbf{\Theta}(0) \in \mathrm{SL}_d(\mathbb{R}) = \mathfrak{sl}_d(\mathbb{R})$ and $d\mathbf{\Theta}(\epsilon)/d\epsilon \in T_{\mathbf{\Theta}(\epsilon)}\mathrm{SL}_d(\mathbb{R})$ for all ϵ, then $\det(\mathbf{\Theta}(\epsilon)) = 1$ for all ϵ, i.e., $\mathbf{\Theta}(\epsilon) \in \mathrm{SL}_d(\mathbb{R})$ for all ϵ. The proof of this result is a direct application of Jacobi's formula [73] which states

$$\begin{aligned}\frac{d}{d\epsilon}\det(\mathbf{\Theta}(\epsilon)) &= \det(\mathbf{\Theta}(\epsilon))\,\mathrm{trace}\left(\frac{d\mathbf{\Theta}(\epsilon)}{d\epsilon}\mathbf{\Theta}^{-1}(\epsilon)\right)\\ &= \det(\mathbf{\Theta}(\epsilon))\,\mathrm{trace}\left(\boldsymbol{N}\mathbf{\Theta}(\epsilon)\mathbf{\Theta}^{-1}(\epsilon)\right)\\ &= \det(\mathbf{\Theta}(\epsilon))\,\mathrm{trace}(\boldsymbol{N})\\ &= 0,\end{aligned} \tag{4.96}$$

since $\boldsymbol{N} \in \mathfrak{sl}_d(\mathbb{R})$. Thus the determinant of $\mathbf{\Theta}(\epsilon)$ is constant and since $\det(\mathbf{\Theta}(0)) = 1$ it follows that $\det(\mathbf{\Theta}(\epsilon)) = 1$ for all $\epsilon \in (-\delta, \delta)$. In the language of abstract differential equations, $\mathfrak{sl}_d(\mathbb{R})$ is referred to as the Lie algebra associated with the Lie group $\mathrm{SL}_d(\mathbb{R})$. In Figure 4.19(a) we provide an illustration of a path $\mathbf{\Theta}(\epsilon)$ on the manifold $\mathrm{SL}_d(\mathbb{R})$ passing through the point $\boldsymbol{A}$ and the tangent space $T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R})$ of $\mathrm{SL}_d(\mathbb{R})$ at $\boldsymbol{A}$. Also depicted in Figure 4.19 is a smooth function f from $\mathrm{SL}_d(\mathbb{R})$ to another smooth manifold $\mathcal{M}$. The image of the path $\mathbf{\Theta}(\epsilon)$ on $\mathrm{SL}_\mathrm{d}(\mathbb{R})$ under f is a path $f(\mathbf{\Theta}(\epsilon))$ on $\mathcal{M}$. Under this mapping of curves, we can associate the tangent vector $d\mathbf{\Theta}(\epsilon)/d\epsilon|_{\epsilon=0}$ in $T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R})$ with a tangent vector $df(\mathbf{\Theta}(\epsilon))/d\epsilon|_{\epsilon=0}$ in $T_{f(\boldsymbol{A})}\mathcal{M}$. This association gives rise to the notion of the directional derivative of a function $f : \mathrm{SL}_d(\mathbb{R}) \to \mathcal{M}$ which we now define.

Definition A1. *Let f be a smooth function from the Riemaniann manifold $\mathrm{SL}_d(\mathbb{R})$ to a smooth manifold $\mathcal{M}$. The directional derivative of f at the point $\boldsymbol{A} \in \mathrm{SL}_d(\mathbb{R})$ in the direction $\boldsymbol{V} \in$*



$T_{\mathbf{A}}\mathrm{SL}_d(\mathbb{R})$
$\mathbf{V}$
$\mathbf{A}$
$\boldsymbol{\Theta}(\epsilon)$
$\mathrm{SL}_d(\mathbb{R})$
f
$T_{f(\mathbf{A})}\mathcal{M}$
$f(\mathbf{A})$
$(d_{\mathbf{A}}f)\mathbf{V}$
$f(\boldsymbol{\Theta}(\epsilon))$
$\mathcal{M}$

Figure 4.19: An illustration of the directional derivative of the function $f : \mathrm{SL}_d(\mathbb{R}) \to \mathcal{M}$. The curve $\boldsymbol{\Theta}(\epsilon)$ on $\mathrm{SL}_d(\mathbb{R})$ is mapped to the curve $f(\boldsymbol{\Theta}(\epsilon))$ on $\mathcal{M}$ and the directional derivative of f at $\boldsymbol{A} \in \mathrm{SL}_d(\mathbb{R})$ in the direction $\boldsymbol{V} \in T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R})$ is the velocity of the curve $f(\boldsymbol{\Theta}(\epsilon))$ at $f(\boldsymbol{A})$.

$T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R})$ *is defined as*

$$(d_{\boldsymbol{A}}f)\boldsymbol{V} = \left.\frac{\partial f(\boldsymbol{\Theta}(\epsilon))}{\partial \epsilon}\right|_{\epsilon=0}, \tag{4.97}$$

where $\boldsymbol{\Theta}(\epsilon)$ *is a smooth curve on* $\mathrm{SL}_d(\mathbb{R})$ *passing through the point* $\boldsymbol{A}$ *at* $\epsilon = 0$ *with velocity* $\boldsymbol{V}$.

It is a standard exercise of differential geometry to verify that the directional derivative (4.97) is independent of the choice of curve $\boldsymbol{\Theta}(\epsilon)$. The map $d_{\boldsymbol{A}}f$ appearing in (4.97) is a linear map from $T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R})$ to $T_{f(\boldsymbol{A})}\mathcal{M}$ known as the differential of f.

In Figure 4.19 we provide an illustration of the mapping f and its differential. The differential and directional derivative allow us to understand the change in f when moving from the point $\boldsymbol{A} \in \mathrm{SL}_d(\mathbb{R})$ in the direction of the tangent vector $\boldsymbol{V} \in T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R})$. Next, we compute the Riemannian gradient of f, i.e., a specific tangent vector on $\mathrm{SL}_d(\mathbb{R})$ that points in a direction which makes the function f vary the most.

For functions defined on Euclidean space, the connection between directional derivative and gradient is understood by using the standard inner product defined for Euclidean spaces. A generalization of the Euclidean inner product for an abstract manifold such as $\mathrm{SL}_d(\mathbb{R})$ is the Riemannian metric $(\cdot,\cdot)_{\boldsymbol{A}}$, which is a collection of smoothly varying inner products on each tangent space $T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R})$. In particular, we define

$$(\boldsymbol{V},\boldsymbol{W})_{\boldsymbol{A}} = \mathrm{trace}\left[\left(\boldsymbol{V}\boldsymbol{A}^{-1}\right)\left(\boldsymbol{W}\boldsymbol{A}^{-1}\right)^{\top}\right], \qquad \forall \boldsymbol{A}\in \mathrm{SL}_d(\mathbb{R}), \quad \forall \boldsymbol{V},\boldsymbol{W}\in T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R}) \tag{4.98}$$

on $\mathrm{SL}_d(\mathbb{R})$. With this Riemannian metric, we can define the Riemannian gradient.

Definition A2. *Let f be a smooth function from $\mathrm{SL}_d(\mathbb{R})$ to a smooth manifold $\mathcal{M}$. The Riemannian gradient of f at the point $\boldsymbol{A}\in \mathrm{SL}_d(\mathbb{R})$ is the unique vector field $\mathrm{grad}f(\boldsymbol{A})$ satisfying*

$$(d_{\boldsymbol{A}}f)\boldsymbol{V} = (\mathrm{grad}f(\boldsymbol{A}),\boldsymbol{V})_{\boldsymbol{A}}, \qquad \forall \boldsymbol{V}\in T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R}). \tag{4.99}$$

Analogous to the Euclidean case, the Riemannian gradient points in the direction which f increases the most, and, the negative gradient points the the direction which f decreases most. With this Riemannian geometric structure, we proceed by constructing a path on $\mathrm{SL}_d(\mathbb{R})$, known as a descent path, which converges to a local minimum of the cost function $S\circ E$ in (4.31).

Appendix 4.B Theorems and Proofs

First, we show that the tensor rank is invariant under translation of functions with compact support. This allows us to disregard translations when looking for rank reducing coordinate flows.

Proposition B1. *Let $u_{TT} \in L^2_\mu(\Omega)$ ($\Omega \subseteq \mathbb{R}^d$) be a rank-$\boldsymbol{r}$ FTT with* $\operatorname{supp}(u_{TT}) \subseteq \Omega$ *and $C_{\boldsymbol{t}} : \mathbb{R}^d \to \mathbb{R}^d$ a coordinate translation, i.e.,*

$$C_{\boldsymbol{t}}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{t}, \tag{4.100}$$

where $\boldsymbol{t} \in \mathbb{R}^d$, such that $\operatorname{supp}(u_{TT}) \subseteq C_{\boldsymbol{t}}(\Omega)$. *Then $u_{TT}(C_{\boldsymbol{t}}(\boldsymbol{x}))$ is also a rank-$\boldsymbol{r}$ FTT tensor.*

Proof. Let

$$u_{\text{TT}}(\boldsymbol{x}) = \boldsymbol{Q}_{\leq i}(\boldsymbol{x}_{\leq i})\boldsymbol{\Sigma}_i\boldsymbol{Q}_{>i}(\boldsymbol{x}_{>i}) \tag{4.101}$$

be an orthogonalized expansion of the the rank-$\boldsymbol{r}$ FTT u_{TT} as in (4.21). Then by a simple change of variables in the integrals it is easy to verify that the translated cores $\boldsymbol{Q}^\top_{\leq i}(\boldsymbol{x}_{\leq i} + \boldsymbol{t}_{\leq i}), \boldsymbol{Q}_{>i}(\boldsymbol{x}_{>i} + \boldsymbol{t}_{>i})$ also satisfy the orthogonality conditions

$$\begin{aligned}\left\langle \boldsymbol{Q}^\top_{\leq i}(\boldsymbol{x}_{\leq i} + \boldsymbol{t}_{\leq i})\boldsymbol{Q}_{\leq i}(\boldsymbol{x}_{\leq i} + \boldsymbol{t}_{\leq i})\right\rangle_{\leq i} &= \boldsymbol{I}_{\boldsymbol{r}_i \times r_i},\\ \left\langle \boldsymbol{Q}_{>i}(\boldsymbol{x}_{>i} + \boldsymbol{t}_{>i})\boldsymbol{Q}^\top_{>i}(\boldsymbol{x}_{>i} + \boldsymbol{t}_{>i})\right\rangle_{>i} &= \boldsymbol{I}_{\boldsymbol{r}_i \times r_i},\end{aligned} \tag{4.102}$$

and thus

$$u_{\text{TT}}(C_{\boldsymbol{t}}(\boldsymbol{x})) = \boldsymbol{Q}_{\leq i}(\boldsymbol{x}_{\leq i} + \boldsymbol{t}_{\leq i})\boldsymbol{\Sigma}_i\boldsymbol{Q}_{>i}(\boldsymbol{x}_{>i} + \boldsymbol{t}_{>i}) \tag{4.103}$$

is an orthogonalized FTT tensor. Hence, $u_{\text{TT}} \circ C_{\boldsymbol{t}}$ has the same multilinear rank $\boldsymbol{r}$ as u_{TT}.

□

Next, we provide a proof of Proposition 4.2.1. To do so we first provide the differentials of the maps E, S and $C = S \circ E$ in the following lemmas.

Lemma B1. *The differential of the evaluation map E corresponding to u_{TT} (see eqn.* (4.29)*) at the point $\boldsymbol{A} \in \mathrm{GL}_d(\mathbb{R})$ in the direction $\boldsymbol{V} \in T_{\boldsymbol{A}}\mathrm{GL}_d(\mathbb{R})$ is*

$$(d_{\boldsymbol{A}}E)\boldsymbol{V} = \nabla v(\boldsymbol{x}) \cdot (\boldsymbol{V}\boldsymbol{x}). \tag{4.104}$$

Proof. This result is proven directly from the definitions. Let $\boldsymbol{\Theta}(\epsilon)$ be a smooth curve on $\mathrm{GL}_d(\mathbb{R})$ passing through $\boldsymbol{A}$ with velocity $\boldsymbol{V}$ at $\epsilon = 0$. Then

$$\begin{aligned}(d_{\boldsymbol{A}}E)\boldsymbol{V} &= \frac{\partial}{\partial \epsilon} E(\boldsymbol{\Theta}(\epsilon))\bigg|_{\epsilon=0} \\ &= \frac{\partial}{\partial \epsilon} u_{\mathrm{TT}}(\boldsymbol{\Theta}(\epsilon)\boldsymbol{x})\bigg|_{\epsilon=0} \\ &= \nabla u_{\mathrm{TT}}(\boldsymbol{A}\boldsymbol{x}) \cdot (\boldsymbol{V}\boldsymbol{x}) \\ &= \nabla v(\boldsymbol{x}) \cdot (\boldsymbol{V}\boldsymbol{x}) \,. \end{aligned} \tag{4.105}$$

□

Lemma B2. *The differential of S (see* (4.27)*) at the point $v \in L^2_\mu(\Omega)$ in the direction $w \in T_v L^2_\mu(\Omega)$ is*

$$(d_v S)w = \sum_{i=1}^{d-1} \int_\Omega \boldsymbol{Q}_{\leq i} \boldsymbol{Q}_{>i} w(\boldsymbol{x}) d\mu(\boldsymbol{x}) \tag{4.106}$$

where $\boldsymbol{Q}_{\leq i}, \boldsymbol{Q}_{>i}$ are FTT cores of v as in eqn. (4.21) *satisfying the orthogonality conditions* (4.24)*.*

Proof. Let $\gamma(\epsilon)$ be a smooth curve on $L^2_\mu(\Omega)$ passing through v at $\epsilon = 0$ with velocity w. At each ϵ the function $\gamma(\epsilon)$ admits orthogonalizations of the form

$$\gamma(\epsilon) = \boldsymbol{Q}_{\leq i}(\epsilon)\boldsymbol{\Sigma}_i(\epsilon)\boldsymbol{Q}_{>i}(\epsilon) \tag{4.107}$$

for each $i = 1, 2, \ldots, d-1$. Differentiating (4.107) with respect to ϵ we obtain

$$\frac{\partial \gamma}{\partial \epsilon} = \frac{\partial \boldsymbol{Q}_{\leq i}}{\partial \epsilon} \boldsymbol{\Sigma}_i \boldsymbol{Q}_{>i} + \boldsymbol{Q}_{\leq i} \frac{\partial \boldsymbol{\Sigma}_i}{\partial \epsilon} \boldsymbol{Q}_{>i} + \boldsymbol{Q}_{\leq i} \boldsymbol{\Sigma}_i \frac{\partial \boldsymbol{Q}_{>i}}{\partial \epsilon}. \tag{4.108}$$

Multiplying on the left by $\boldsymbol{Q}_{\leq i}^{\top}$ and on the right by $\boldsymbol{Q}_{>i}^{\top}$, applying the operator $\langle \cdot \rangle_{\leq i, >i}$, and evaluating at $\epsilon = 0$ we obtain

$$\frac{\partial \boldsymbol{\Sigma}_i}{\partial \epsilon} = \left\langle \boldsymbol{Q}_{\leq i}^{\top} w(\boldsymbol{x}) \boldsymbol{Q}_{>i}^{\top} \right\rangle_{\leq i, >i} - \left\langle \boldsymbol{Q}_{\leq i}^{\top} \frac{\partial \boldsymbol{Q}_{\leq i}}{\partial \epsilon} \right\rangle_{\leq i} \boldsymbol{\Sigma}_i - \boldsymbol{\Sigma}_i \left\langle \frac{\partial \boldsymbol{Q}_{>i}}{\partial \epsilon} \boldsymbol{Q}_{>i}^{\top} \right\rangle_{>i}, \tag{4.109}$$

where we used orthogonality of the FTT cores $\boldsymbol{Q}_{\leq i}$ and $\boldsymbol{Q}_{>i}$. Differentiating the orthogonality constraints (4.24) with respect to ϵ we obtain

$$\left\langle \frac{\partial \boldsymbol{Q}_{\leq i}^{\top}}{\partial \epsilon} \boldsymbol{Q}_{\leq i} \right\rangle_{\leq i} = -\left\langle \boldsymbol{Q}_{\leq i}^{\top} \frac{\partial \boldsymbol{Q}_{\leq i}}{\partial \epsilon} \right\rangle_{\leq i}, \qquad \left\langle \frac{\partial \boldsymbol{Q}_{>i}}{\partial \epsilon} \boldsymbol{Q}_{>i}^{\top} \right\rangle_{>i} = -\left\langle \boldsymbol{Q}_{>i} \frac{\partial \boldsymbol{Q}_{>i}^{\top}}{\partial \epsilon} \right\rangle_{>i}, \quad \forall \epsilon, \tag{4.110}$$

which implies that the second two terms on the right hand side of (4.109) side are skew-symmetric and thus have zeros on the diagonal. Hence the diagonal entries of $\dfrac{\partial \boldsymbol{\Sigma}_i}{\partial \epsilon}$ are the diagonal entries of the matrix $\left\langle \boldsymbol{Q}_{\leq i}^{\top} w(\boldsymbol{x}) \boldsymbol{Q}_{>i}^{\top} \right\rangle_{\leq i, >i}$ or written element-wise

$$\frac{\partial S_i(\alpha_i)}{\partial \epsilon} = \int q_{\leq i}(\alpha_i) w(\boldsymbol{x}) q_{>i}(\alpha_i) d\mu(\boldsymbol{x}). \tag{4.111}$$

Finally summing (4.111) over $i = 1, 2, \ldots, d-1$ and $\alpha_i = 1, 2, \ldots, r_i$ and using matrix product notation for the latter summation we obtain

$$\sum_{i=1}^{d-1} \sum_{\alpha_i=1}^{r_i} \frac{\partial S_i(\alpha_i)}{\partial \epsilon} = \int_{\Omega} \boldsymbol{Q}_{\leq i} \boldsymbol{Q}_{>i} w(\boldsymbol{x}) d\mu(\boldsymbol{x}), \tag{4.112}$$

proving the result.

$\square$

Combining the results of Lemma B1 and Lemma B2 with a simple application of the chain rule for differentials we prove the following Lemma.

Lemma B3. *The differential of the function $C = S \circ E$ at the point $\boldsymbol{A} \in \mathrm{GL}_d(\mathbb{R})$ in the direction $\boldsymbol{V} \in T_{\boldsymbol{A}}\mathrm{GL}_d(\mathbb{R})$ is*

$$d_{\boldsymbol{A}}(S \circ E)\boldsymbol{V} = \sum_{i=1}^{d-1} \int_\Omega \boldsymbol{Q}_{\leq i}\boldsymbol{Q}_{>i}\nabla v(\boldsymbol{x}) \cdot (\boldsymbol{V}\boldsymbol{x})\, d\mu(\boldsymbol{x}), \tag{4.113}$$

where $\boldsymbol{Q}_{\leq i}, \boldsymbol{Q}_{>i}$ are orthogonal FTT cores of $v(\boldsymbol{x})$.

Next we provide the Riemannian gradient of $C = S \circ E$ when its domain is $\mathrm{GL}_d(\mathbb{R})$.

Proposition B2. *The Riemannian gradient of $(S \circ E) : \mathrm{GL}_d(\mathbb{R}) \to \mathbb{R}$ at the point $\boldsymbol{A} \in \mathrm{GL}_d(\mathbb{R})$ is given by*

$$\mathrm{grad}(S \circ E)(\boldsymbol{A}) = \widehat{\boldsymbol{D}}\boldsymbol{A}, \tag{4.114}$$

where

$$\widehat{\boldsymbol{D}} = \sum_{i=1}^{d-1} \int \boldsymbol{Q}_{\leq i}\boldsymbol{Q}_{>i}\nabla v(\boldsymbol{x})\, (\boldsymbol{A}\boldsymbol{x})^\top\, d\mu(\boldsymbol{x}) \tag{4.115}$$

Proof. To prove this result we check directly using the definition of Riemannian gradient. For any $\boldsymbol{A} \in \mathrm{GL}_d(\mathbb{R})$ and $\boldsymbol{V} \in T_{\boldsymbol{A}}\mathrm{GL}_d(\mathbb{R})$ we have

$$\begin{aligned}
\left(\widehat{\boldsymbol{D}}\boldsymbol{A}, \boldsymbol{V}\right)_{\boldsymbol{A}} &= \left(\left[\sum_{i=1}^{d-1} \int \boldsymbol{Q}_{\leq i}\boldsymbol{Q}_{>i}\nabla v(\boldsymbol{x})\, (\boldsymbol{A}\boldsymbol{x})^\top\, d\mu(\boldsymbol{x})\right] \boldsymbol{A}, \boldsymbol{V}\right)_{\boldsymbol{A}} \\
&= \mathrm{trace}\left(\sum_{i=1}^{d-1} \int \boldsymbol{Q}_{\leq i}\boldsymbol{Q}_{>i}\nabla v(\boldsymbol{x})\boldsymbol{x}^\top \boldsymbol{A}^\top d\mu(\boldsymbol{x})\boldsymbol{A}^{-\top}\boldsymbol{V}^\top\right) \\
&= \sum_{i=1}^{d-1} \int \boldsymbol{Q}_{\leq i}\boldsymbol{Q}_{>i}\mathrm{trace}\left(\nabla v(\boldsymbol{x})\boldsymbol{x}^\top \boldsymbol{V}^\top\right) d\mu(\boldsymbol{x}) \\
&= \sum_{i=1}^{d-1} \int \boldsymbol{Q}_{\leq i}\boldsymbol{Q}_{>i}\nabla v(\boldsymbol{x}) \cdot (\boldsymbol{V}\boldsymbol{x})\, d\mu(\boldsymbol{x}),
\end{aligned} \tag{4.116}$$

where in the last equality we used the fact that $\mathrm{trace}(\boldsymbol{v}\boldsymbol{w}^\top) = \boldsymbol{v} \cdot \boldsymbol{w}$ for any column vectors $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^d$.

$\square$

In general, the trace of $\widehat{\boldsymbol{D}}$ is not equal to zero and thus $\widehat{\boldsymbol{D}}\boldsymbol{A}$ is not an element of the tangent space $T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R})$ (see eqn. (4.95)). In order to obtain the Riemannian gradient $\boldsymbol{D}\boldsymbol{A}$ of $S \circ E$: $\mathrm{SL}_d(\mathbb{R}) \to \mathbb{R}$, we modify the diagonal entries of $\widehat{\boldsymbol{D}}$ to ensure that $\boldsymbol{D}\boldsymbol{A}$ satisfies the properties of Riemannian gradient and also belongs to the tangent space $T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R})$.

Proof.[Proposition 4.2.1] First we prove that $\boldsymbol{D}\boldsymbol{A}$ with $\boldsymbol{D}$ defined in (4.34) is an element of $T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R})$, i.e., we prove that $\mathrm{trace}(\boldsymbol{D}) = 0$:

$$\begin{aligned}\mathrm{trace}(\boldsymbol{D}) &= \sum_{i=1}^{d-1} \int \boldsymbol{Q}_{\leq i}\boldsymbol{Q}_{>i}\mathrm{trace}\left(\nabla v(\boldsymbol{x})\,(\boldsymbol{A}\boldsymbol{x})^\top - \frac{\nabla v(\boldsymbol{x})^\top \boldsymbol{A}\boldsymbol{x}}{d}\boldsymbol{I}_{d\times d}\right) d\mu(\boldsymbol{x}) \\ &= \sum_{i=1}^{d-1} \int \boldsymbol{Q}_{\leq i}\boldsymbol{Q}_{>i}\left[\mathrm{trace}\left(\nabla v(\boldsymbol{x})\,(\boldsymbol{A}\boldsymbol{x})^\top\right) - \mathrm{trace}\left(\frac{\nabla v(\boldsymbol{x})^\top \boldsymbol{A}\boldsymbol{x}}{d}\boldsymbol{I}_{d\times d}\right)\right] d\mu(\boldsymbol{x}).\end{aligned} \tag{4.117}$$

It is easy to verify that $\mathrm{trace}\left(\nabla v(\boldsymbol{x})\,(\boldsymbol{A}\boldsymbol{x})^\top\right) = \mathrm{trace}\left(\frac{\nabla v(\boldsymbol{x})^\top \boldsymbol{A}\boldsymbol{x}}{d}\boldsymbol{I}_{d\times d}\right)$ and hence $\mathrm{trace}(\boldsymbol{D}) = 0$. Next we show that $(\boldsymbol{D}\boldsymbol{A}, \boldsymbol{V})_{\boldsymbol{A}} = d_{\boldsymbol{A}}(S \circ E)\boldsymbol{V}$ for all $\boldsymbol{A} \in \mathrm{SL}_d(\mathbb{R})$ and $\boldsymbol{V} \in T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R})$. Indeed, for any $\boldsymbol{A} \in \mathrm{SL}_d(\mathbb{R})$ and $\boldsymbol{V} \in T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R})$ we have

$$\begin{aligned}(\boldsymbol{D}\boldsymbol{A}, \boldsymbol{V})_{\boldsymbol{A}} &= \sum_{i=1}^{d-1} \int \boldsymbol{Q}_{\leq i}\boldsymbol{Q}_{>i}\mathrm{trace}\left[\left(\nabla v(\boldsymbol{x})\,(\boldsymbol{A}\boldsymbol{x})^\top - \frac{\nabla v(\boldsymbol{x})^\top \boldsymbol{A}\boldsymbol{x}}{d}\boldsymbol{I}_{d\times d}\right)\left(\boldsymbol{V}\boldsymbol{A}^{-1}\right)^\top\right] d\mu(\boldsymbol{x}) \\ &= \sum_{i=1}^{d-1} \int \boldsymbol{Q}_{\leq i}\boldsymbol{Q}_{>i}\Bigg[\mathrm{trace}\left(\nabla v(\boldsymbol{x})\,(\boldsymbol{A}\boldsymbol{x})^\top \left(\boldsymbol{V}\boldsymbol{A}^{-1}\right)^\top\right) \\ &\qquad\qquad - \frac{\nabla v(\boldsymbol{x})^\top \boldsymbol{A}\boldsymbol{x}}{d}\mathrm{trace}\left(\left(\boldsymbol{V}\boldsymbol{A}^{-1}\right)^\top\right)\Bigg] d\mu(\boldsymbol{x}).\end{aligned} \tag{4.118}$$

Since $\boldsymbol{V} \in T_{\boldsymbol{A}}\mathrm{SL}_d(\mathbb{R})$ we have that $\boldsymbol{V} = \boldsymbol{W}\boldsymbol{A}$ for some real matrix $\boldsymbol{W}$ with $\mathrm{trace}(\boldsymbol{W}) = 0$.

Using this in the preceding equation we have

$$\begin{aligned}(\boldsymbol{D}(\boldsymbol{A})\boldsymbol{A},\boldsymbol{V})_{\boldsymbol{A}} &= \sum_{i=1}^{d-1}\int \boldsymbol{Q}_{\leq i}\boldsymbol{Q}_{>i}\Bigg[\text{trace}\left(\nabla v(\boldsymbol{x})\boldsymbol{x}^\top\boldsymbol{A}^\top\boldsymbol{A}^{-\top}\boldsymbol{V}^\top\right) \\ &\qquad - \frac{\nabla v(\boldsymbol{x})^\top\boldsymbol{A}\boldsymbol{x}}{d}\text{trace}\left(\boldsymbol{W}^\top\right)\Bigg]d\mu(\boldsymbol{x}) \\ &= \sum_{i=1}^{d-1}\int \boldsymbol{Q}_{\leq i}\boldsymbol{Q}_{>i}\text{trace}\left(\nabla v(\boldsymbol{x})\boldsymbol{x}^\top\boldsymbol{V}^\top\right)d\mu(\boldsymbol{x}) \\ &= \sum_{i=1}^{d-1}\int \boldsymbol{Q}_{\leq i}\boldsymbol{Q}_{>i}\nabla v(\boldsymbol{x})\cdot(\boldsymbol{V}\boldsymbol{x})\,d\mu(\boldsymbol{x}) \\ &= d_{\boldsymbol{A}}(S\circ E)\boldsymbol{V},\end{aligned} \tag{4.119}$$

completing the proof.

□

Lemma B4. *Let $\sigma_i(\alpha_i)$ ($i = 1,2,\ldots,d$, $\alpha_i = 1,2,\ldots,r_i$) be the multilinear spectrum of the FTT $v_{TT}(\boldsymbol{x}) \approx u_{TT}(\boldsymbol{A}\boldsymbol{x})$ and assume that for each $i = 1,2,\ldots,d$ the real numbers $\sigma_i(\alpha_i)$ are distinct for all $\alpha_i = 1,2,\ldots,r_i$. Then the cost function $(S\circ E)$ is a differentiable at the point $\boldsymbol{A}$.*

Proof. Let $\boldsymbol{\Theta}(\epsilon) \in \mathcal{C}^1((-\delta,\delta),\text{SL}_d(\mathbb{R}))$ with $\boldsymbol{\Theta}(0) = \boldsymbol{A}$. The ϵ-dependent multilinear spectrum $\sigma_i(\alpha_i;\epsilon)$ ($i = 1,2,\ldots,d$, $\alpha_i = 1,2,\ldots,r_i$) of $u_{\text{TT}}(\boldsymbol{\Theta}(\epsilon)\boldsymbol{x})$ are given by the eigenvalues of an ϵ-dependent self-adjoint compact Hermitian operator. In a neighborhood of $\epsilon = 0$ the eigenvalue $\sigma_i(\alpha_i;\epsilon)$ admits a series expansion [56, p. xx]

$$\sigma_i(\alpha_i;\epsilon) = \sigma_i(\alpha_i;0) + \epsilon\hat{\sigma}_i(\alpha_i), \tag{4.120}$$

and thus $\sigma_i(\alpha_i;\epsilon)$ is differentiable with respect to ϵ at $\epsilon = 0$. Hence the sum of all eigenvalues

$$\sum_{i=1}^{d-1}\sum_{\alpha_i=1}^{r_i}\sigma_i(\alpha_i;\epsilon)$$

is differentiable at $\epsilon = 0$ and thus the cost function C is differentiable at $\boldsymbol{A} \in \mathrm{SL}_d(\mathbb{R})$.

$\square$

Algorithm 3: Modified FTT truncation algorithm.

```
Input:
$v \to$ FTT tensor with cores $\boldsymbol{\Psi}_1, \boldsymbol{\Psi}_2, \dots, \boldsymbol{\Psi}_d$,
$\delta \to$ desired accuracy.
Output:
$v_{\text{TT}} \to$ truncated FTT tensor satisfying $\|v - v_{\text{TT}}\|_{L^2_\mu(\Omega)} \leq \delta \|v\|_{L^2_\mu(\Omega)}$,
$S(v_{\text{TT}}) \to$ sum of all multilinear singular values of $v_{\text{TT}}$,
$\boldsymbol{D} \to$ left factor of the Riemannian gradient (4.33).
Runtime:
    $\hat{\delta} = \frac{\delta}{\sqrt{d-1}} \|v\|_{L^2}$        (Set truncation parameter)
    $S(v) = 0$        (Initialize $S(v)$)

    for $i = 1$ to $d-1$        (Left-to-right orthogonalization)

        $[\boldsymbol{Q}_i, \boldsymbol{R}_i] = \text{QR}(\boldsymbol{\Psi}_i)$
        $\boldsymbol{\Psi}_{i+1} = \boldsymbol{R}_i \boldsymbol{\Psi}_{i+1}$

    end

    for $i = d$ to $2$        (Right-to-left truncation and gradient computation)

        $[\boldsymbol{L}_i, \boldsymbol{Q}_i] = \text{LQ}(\boldsymbol{\Psi}_i)$
        $[\boldsymbol{U}_i, \boldsymbol{\Sigma}_i, \boldsymbol{V}_i] = \text{SVD}_{\hat{\delta}}(\boldsymbol{L}_i)$
        $\boldsymbol{Q}_i = \boldsymbol{V}_i^\top \boldsymbol{Q}_i$
        $\boldsymbol{Q}_{i-1} = \boldsymbol{Q}_{i-1} \boldsymbol{U}_i$,
        $S(v) = S(v) + \text{sum}(\Sigma_i)$,
        $\boldsymbol{D}^{(i-1)} = \int \boldsymbol{Q}_{\leq i-1} \boldsymbol{Q}_{>i-1} \left(\nabla v(\boldsymbol{x}) \boldsymbol{x}^\top\right) d\mu(\boldsymbol{x})$

    end
    $\boldsymbol{D} = \sum_{i=1}^{d-1} \boldsymbol{D}^{(i)}$
    $v_{\text{TT}} = \boldsymbol{Q}_1 \boldsymbol{\Sigma}_2 \boldsymbol{Q}_2 \dots \boldsymbol{Q}_d$
```

Algorithm 4: PDE integrator with adaptive rank-reducing coordinate flows.

Input:

$u_0 \to$ initial condition in FTT tensor format,

$\Delta t \to$ temporal step size,

$N_t \to$ total number of time steps,

$\text{max rank} \to$ maximum rank during FTT integration before attempting rank reduction,

$\text{max time} \to$ maximum CPU-time for one time step before attempting rank reduction,

$k_r \to$ increase for maximum rank after performing coordinate transformation,

$k_t \to$ increase for maximum time after performing coordinate transformation,

$\Delta\epsilon \to$ gradient descent step-size,

$\eta \to$ tolerance for coordinate gradient descent,

$M_{\text{iter}} \to$ maximum number of iterations for gradient descent routine.

Output:

$\mathbf{\Gamma} \to$ rank-reducing linear coordinate transformation for PDE solution,

$v_{\text{TT}}(\boldsymbol{x}, t_f) = u_{\text{TT}}(\mathbf{\Gamma}\boldsymbol{x}, t_f) \to$ FTT-ridge solution tensor at time t_f.

Runtime:

```
    $\mathbf{\Gamma} = \boldsymbol{I}$,
    $v_0 = u_{\text{TT}}$,
    **for** $k = 0$ **to** $N_t$
        **if** $\text{time} > \text{max time}$ **or** $\text{rank} > \text{max rank}$
            $[v_k, \mathbf{\Gamma}_{\text{new}}] = \text{gradient descent}(v_k, \Delta\epsilon, \eta, M_{\text{iter}})$
            $\mathbf{\Gamma} = \mathbf{\Gamma}_{\text{new}}\mathbf{\Gamma}$
            $\text{max rank} = \text{max rank} + k_r$
            $\text{max time} = \text{max time} + k_t$
        **end**
        $[v_{k+1}, \text{time}, \text{rank}] = \mathfrak{T}_\delta\left(v_k + \Delta t \Phi(v_k, G_{\mathbf{TT},\mathbf{\Gamma}}, \Delta t)\right)$
    **end**
```

Algorithm 5: PDE integrator with coordinate corrections at each time step.

```
Input:
$u_0 \to$ initial condition in FTT tensor format,
$\Delta t \to$ temporal step size,
$N_t \to$ total number of time steps,
$\Delta\epsilon \to$ gradient descent step-size,
$\eta \to$ tolerance for coordinate gradient descent,
$M_{\text{iter}} \to$ maximum number of iterations for gradient descent routine.
Output:
$\mathbf{\Gamma} \to$ rank-reducing linear coordinate transformation for PDE solution,
$v_{\text{TT}}(\boldsymbol{x}, t_f) = u_{\text{TT}}(\mathbf{\Gamma}\boldsymbol{x}, t_f) \to$ FTT-ridge solution at time $t_f$.
Runtime:
    $\mathbf{\Gamma} = \boldsymbol{I}$,
    $v_0 = u_{\text{TT}}$,
    for $k = 0$ to $N_t$
        $[v_k, \mathbf{\Gamma}_{\text{new}}] = \text{gradient descent}(v_k, \Delta\epsilon, \eta, M_{\text{iter}})$
        $\mathbf{\Gamma} = \mathbf{\Gamma}_{\text{new}}\mathbf{\Gamma}$
        $v_{k+1} = \mathfrak{T}_\delta\left(v_k + \Delta t \Phi(v_k, G_{\text{TT},\mathbf{\Gamma}}, \Delta t)\right)$
    end
```

Chapter 5

Conclusion

We presented methods for computing the solution to high-dimensional nonlinear PDEs with low-rank tensors at a storage and computational cost that grows linearly with the problem dimension. In Chapter 2 we introduced low-rank tensor expansions, their associated geometric features, and provided a numerical application to a prototype three-dimensional function. In Chapter 3 we developed numerical methods for computing the solution to high-dimensional PDEs using low-rank tensor expansions. These methods are based on deriving equations of motion for PDEs on a low-rank tensor manifold and then constructing an approximate PDE solution as a path on the tensor manifold. Two schemes for computing approximate PDE solutions on tensor manifolds were presented. The first is based on dynamic approximation in which the PDE velocity vector is projected onto the tensor manifold tangent space and the second is based on step-truncation in which the solution is projected onto the tensor manifold at each time step. It was shown that these two schemes are consistent with each other as the temporal step-size is sent to zero. We also presented a rank-adaptive algorithm to adaptively add and remove tensor modes from the PDE solution during time integration. The adaptive algorithm is based on a thresholding criterion that limits the component of the PDE velocity vector normal to the tensor manifold. We demonstrated the rank-adaptive tensor integrator on a two-dimensional advection equation, a two-dimensional Kuramoto-Sivashinsky equation, and on a four-dimensional Fokker-Planck equation.

Building upon these low-rank tensor integration schemes, in Chapter 4 we presented a tensor rank reduction method based on coordinate transformations that can greatly increase the efficiency of high-dimensional tensor approximation algorithms. The idea is to determine a coordinate transformation of a given functions domain so that the function in the new coordinate system has smaller tensor rank. A theoretical framework based on coordinate flows was described for linear transformations giving rise to a class of functions referred to as tensor ridge functions. We developed an algorithm based on coordinate flows and Riemannian optimization on matrix manifolds for computing rank reducing linear coordinate transformations. The effectiveness of the proposed rank reduction algorithm was demonstrated on a three-dimensional prototype function. We also described methods for integrating nonlinear PDEs in low-rank coordinate systems based on step-truncation tensor integrators. Two coordinate-adaptive time integration schemes were proposed in order to control the solution rank and computational cost during temporal integration. We provided a demonstration and comparison of the two coordinate-adaptive algorithms by applying them to various PDEs in dimensions two, three, and five. Specifically we studied linear advection equations and a nonlinear reaction-diffusion equation.

In this dissertation we have demonstrated that low-rank tensor methods are a promising tool for the numerical treatment of high-dimensional problems. At the present time there are many limitations of tensor methods and opportunities to improve their efficiency which provide interesting directions for future research. First, currently tensor methods are limited to relatively simple domains. Some recent work has been done to use tensor methods more complicated geometries (e.g., [106]), more research in this direction is expected. Second, using a coordinate transformation to reduce the rank of a tensor is an idea that we recently introduced [31] which opens a new line of research for tensor compression. More exploration in this direction is warranted in the following ways. Rank reduction via coordinate transformations can potentially be generalized to larger classes of nonlinear coordinate transformations. Also, the use of coordinate transformations to control the rank of an operator between Hilbert spaces $G : H \to H$ should be addressed in order to find coordinate transformations that reduce the overall computational cost of solving PDEs with tensors. In addition, alternative algorithms for computing rank reducing coordinate transformations and for changing the coordinate system of a tensor should be explored in future research. Finally high-performance implementations of the methods presented in this dissertation, e.g., based on parallel arithmetic for tensors [26] or parallel in time integration schemes [38], should developed.

Bibliography

[1] D. Venturi A. M. P. Boelens and D. M. Tartakovsky. Parallel tensor methods for high-dimensional linear PDEs. *J. Comput. Phys.*, 375:519–539, 2018.

[2] R. Aris. *Vectors, tensors and the basic equations of fluid mechanics*. Dover publications, 1989.

[3] N. Aubry. On the hidden beauty of the proper orthogonal decomposition. *Theoretical and Computational Fluid Dynamics*, 2(5):339–352, Aug 1991.

[4] N. Aubry, R. Guyonnet, and R. Lima. Spatiotemporal analysis of complex signals: theory and applications. *J. Statist. Phys.*, 64(3-4):683–739, 1991.

[5] N. Aubry and R. Lima. Spatiotemporal and statistical symmetries. *J. Statist. Phys.*, 81(3-4):793–828, 1995.

[6] H. Babaee, M. Choi, T.P. Sapsis, and G. E. Karniadakis. A robust bi-orthogonal/dynamically-orthogonal method using the covariance pseudo-inverse with application to stochastic flow problems. *J. Comput. Phys.*, 344:303–319, 2017.

[7] M. Bachmayr, R. Schneider, and A. Uschmajew. Tensor networks and hierarchical ten-

sors for the solution of high-dimensional partial differential equations. *Found. Comput. Math.*, 16(6), 2016.

[8] J. Baldeaux and M. Gnewuch. Optimal randomized multilevel algorithms for infinite-dimensional integration on function spaces with ANOVA-type decomposition. *SIAM J. Numer. Anal.*, 52(3):1128–1155, 2014.

[9] V. Barthelmann, E. Novak, and K. Ritter. High dimensional polynomial interpolation on sparse grids. *Advances in Computational Mechanics*, 12:273–288, 2000.

[10] J. A. Bengua, H. N. Phien, H. D. Tua, and N. M. Do. Efficient tensor completion for color image and video recovery: low-rank tensor train. *IEEE Trans. Image Process.*, 26(5):2466–2479, 2017.

[11] R. Bertram and J. E. Rubin. Multi-timescale systems and fast-slow analysis. *Math. Biosci.*, 287:105–121, 2017.

[12] G. Beylkin and M. J. Mohlenkamp. Numerical operator calculus in higher dimensions. *Proc. Natl. Acad. Sci. USA*, 99(16):10246–10251, 2002.

[13] D. Bigoni, A. P. Engsig-Karup, and Y. M. Marzouk. Spectral tensor-train decomposition. *SIAM J. Sci. Comput.*, 38(4):A2405–A2439, 2016.

[14] S. Blanes, F. Casas, J. A. Oteo, and J. Ros. The Magnus expansion and some of its applications. *Phys. Rep.*, 470(5-6):151–238, 2009.

[15] H. J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

[16] Y. Cao, Z. Chen, and M. Gunzbuger. ANOVA expansions and efficient sampling methods for parameter dependent nonlinear PDEs. *Int. J. Numer. Anal. Model.*, 6:256–273, 2009.

[17] C. Cercignani. *The Boltzmann equation and its applications*. Springer, 1988.

[18] G. Ceruti and C. Lubich. An unconventional robust integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 62:23–44, 2022.

[19] M. Cheng, T. Y. Hou, and Z. Zhang. A dynamically bi-orthogonal method for time-dependent stochastic partial differential equations II: adaptivity and generalizations. *J. Comput. Phys.*, 242:753–776, 2013.

[20] F. Chinesta, R. Keunings, and A. Leygue. *The Proper generalized decomposition for advanced numerical simulations*. Springer, 2014.

[21] E. Chiumiento and M. Melgaard. Stiefel and Grassmann manifolds in quantum chemistry. *J. Geom. Phys.*, 62(8):1866–1881, 2012.

[22] A. Chkifa, A. Cohen, and C. Schwab. High-dimensional adaptive sparse polynomial interpolation and applications to parametric PDEs. *Found. Comput. Math.*, 14:601–633, 2014.

[23] H. Cho, D. Venturi, and G. E. Karniadakis. Numerical methods for high-dimensional probability density function equation. *J. Comput. Phys*, 315:817–837, 2016.

[24] M. Choi, T. P. Sapsis, and G. E. Karniadakis. On the equivalence of dynamically orthogonal and bi-orthogonal methods: theory and numerical simulations. *J. Comput. Phys.*, 270:1–20, 2014.

[25] P. G. Constantine, E. Dow, and Q. Wang. Active subspace methods in theory and practice: applications to kriging surfaces. *SIAM J. Sci. Comput.*, 36(4):A1500–A1524, 2014.

[26] H. A. Daas, G. Ballard, and P. Benner. Parallel algorithms for tensor train arithmetic. *SIAM Journal on Scientific Computing*, 44(1):C25–C53, 2022.

[27] H. A. Daas, G. Ballard, P. Cazeaux, E. Hallman, A. Międlar, M. Pasha, T.W. Reid, and A. K. Saibaba. Randomized algorithms for rounding in the tensor-train format. *SIAM Journal on Scientific Computing*, 45(1):A74–A95, 2023.

[28] A. Dektor, A. Rodgers, and D. Venturi. Rank-adaptive tensor methods for high-dimensional nonlinear PDEs. *J. Sci. Comput.*, 88(36):1–27, 2021.

[29] A. Dektor and D. Venturi. Dynamically orthogonal tensor methods for high-dimensional nonlinear PDEs. *J. Comput. Phys.*, 404:109125, 2020.

[30] A. Dektor and D. Venturi. Dynamic tensor approximation of high-dimensional nonlinear PDEs. *J. Comput. Phys.*, 437:110295, 2021.

[31] A. Dektor and D. Venturi. Tensor rank reduction via coordinate flows. *arXiv:2207.11955*, pages 1–33, 2023.

[32] J. Dick, F. Y. Kuo, and I. H. Sloan. High-dimensional integration: the quasi-Monte Carlo way. *Acta Numer.*, 22:133–288, 2013.

[33] G. Dimarco and L. Pareschi. Numerical methods for kinetic equations. *Acta Numerica*, 23:369–520, 2014.

[34] A. Falcó, W. Hackbusch, and A. Nouy. Geometric structures in tensor representations. *arXiv:1505.03027*, pages 1–50, 2015.

[35] F. Feppon and P. F. J. Lermusiaux. A geometric approach to dynamical model order reduction. *SIAM J. Matrix Anal. Appl.*, 39(1):510–538, 2018.

[36] F. Feppon and P. F. J. Lermusiaux. The extrinsic geometry of dynamical systems tracking nonlinear matrix projections. *SIAM J. Matrix Anal. Appl.*, 40(2):814–844, 2019.

[37] J. Foo and G. E. Karniadakis. Multi-element probabilistic collocation method in high dimensions. *J. Comput. Phys.*, 229:1536–1557, 2010.

[38] J. Kusch G. Ceruti and C. Lubich. A parallel rank-adaptive integrator for dynamical low-rank approximation. *arXiv:2304.05660*, pages 1–22, 2023.

[39] L Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM J. Matrix Anal. Appl.*, 31(4):2029–2054, 2009/10.

[40] L. Grasedyck, D. Kressner, and C. Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitt.*, 36(1):53–78, 2013.

[41] L. Grasedyck and C. Löbbert. Distributed hierarchical SVD in the hierarchical Tucker format. *Numer. Linear Algebra Appl.*, 25(6):e2174, 2018.

[42] M. Griebel and G. Li. On the decay rate of the singular values of bivariate functions. *SIAM J. Numer. Anal.*, 56(2):974–993, 2019.

[43] W. Hackbusch. *Tensor spaces and numerical tensor calculus*. Springer, 2012.

[44] J. Heng, A. Doucet, and Y. Pokern. Gibbs flow for approximate transport with applications to Bayesian computation. *J. R. Stat. Soc. Series B*, 83:156–187, 2021.

[45] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral methods for time-dependent problems*, volume 21 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2007.

[46] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral methods for time-dependent problems*, volume 21 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2007.

[47] S. Holtz, T. Rohwedder, and R. Schneider. On manifolds of tensors of fixed TT-rank. *Numer. Math.*, 120(4):701–731, 2012.

[48] E. Hopf. Statistical hydromechanics and functional calculus. *J. Rat. Mech. Anal.*, 1(1):87–123, 1952.

[49] E. Hopf and E. W. Titt. On certain special solutions of the ϕ-equation of statistical hydrodynamics. *J. Rat. Mech. Anal.*, 2(3):587–592, 1953.

[50] C. Itzykson and J. B. Zuber. *Quantum field theory*. Dover, 2005. Republication of the work originally published by McGraw-Hill, Inc., NY, 1980.

[51] R. V. Jensen. Functional integral approach to classical statistical dynamics. *J. Stat. Phys.*, 25(2):183–210, 1981.

[52] B. Jouvet and R. Phythian. Quantum aspects of classical and statistical fields. *Phys. Rev. A*, 19:1350–1355, 1979.

[53] A. Kalogirou, E. E. Keaveny, and D.T. Papageorgiou. An in-depth numerical study of the two-dimensional Kuramoto-Sivashinsky equation. *Proc. A.*, 471(2179):20140932, 20, 2015.

[54] G. E. Karniadakis and S. Sherwin. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press, second edition, 2005.

[55] A.-K Kassam and L. N. Trefethen. Fourth-order time-stepping for stiff pdes. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005.

[56] T. Kato. *Perturbation theory for linear operators*. Classics in Mathematics. Springer-Verlag, Berlin, 1995. Reprint of the 1980 edition.

[57] B. N. Khoromskij. Tensor numerical methods for multidimensional PDEs: theoretical analysis and initial applications. In *CEMRACS 2013—modelling and simulation of complex systems: stochastic and deterministic approaches*, volume 48 of *ESAIM Proc. Surveys*, pages 1–28.

[58] B. N. Khoromskij. Tensor numerical methods for multidimensional PDEs: theoretical analysis and initial applications. In *CEMRACS 2013—modelling and simulation of complex systems: stochastic and deterministic approaches*, volume 48 of *ESAIM Proc. Surveys*, pages 1–28.

[59] E. Kieri, C. Lubich, and H. Walach. Discretized dynamical low-rank approximation in the presence of small singular values. *SIAM J. Numer. Anal.*, 54(2):1020–1038, 2016.

[60] E. Kieri and B. Vandereycken. Projection methods for dynamical low-rank approximation of high-dimensional problems. *Comput. Methods Appl. Math.*, 19(1):73–92, 2019.

[61] O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM J. Matrix Anal. Appl.*, 29(2):434–454, 2007.

[62] O. Koch and C. Lubich. Dynamical tensor approximation. *SIAM J. Matrix Anal. Appl.*, 31(5):2360–2375, 2010.

[63] O. Koch, C. Neuhauser, and M. Thalhammer. Error analysis of high-order splitting methods for nonlinear evolutionary Schrödinger equations and application to the MCTDHF equations in electron dynamics. *ESAIM Math. Model. Numer. Anal.*, 47(5):1265–1286, 2013.

[64] T. Kolda and B. W. Bader. Tensor decompositions and applications. *SIREV*, 51:455–500, 2009.

[65] C. Krumnow, L. Veis, Ö. Legeza, and J. Eisert. Fermionic orbital optimization in tensor network states. *Phys. Rev. Lett.*, 117:210402, 2016.

[66] L.D. Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.

[67] G. Li and H. Rabitz. Regularized random-sampling high dimensional model representation (RS-HDMR). *Journal of Mathematical Chemistry*, 43(3):1207–1232, 2008.

[68] C. Lu, J. Tang, S. Yan, and Z. Lin. Nonconvex nonsmooth low rank minimization via iteratively reweighted nuclear norm. *IEEE Trans. Image Process.*, 25(2):829–839, 2016.

[69] C. Lubich, I. V. Oseledets, and B. Vandereycken. Time integration of tensor trains. *SIAM J. Numer. Anal.*, 53(2):917–941, 2015.

[70] C. Lubich, B. Vandereycken, and H. Walach. Time integration of rank-constrained Tucker tensors. *SIAM J. Numer. Anal.*, 56(3):1273–1290, 2018.

[71] H. Luo and T. R. Bewley. On the contravariant form of the Navier-Stokes equations in time-dependent curvilinear coordinate systems. *J. Comput. Phys.*, 199:355–375, 2004.

[72] T. Y. Hou M. Cheng and Z. Zhang. A dynamically bi-orthogonal method for time-dependent stochastic partial differential equations I: derivation and algorithms. *J. Comput. Phys.*, 242:843–868, 2013.

[73] J. R. Magnus and H. Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons, 2019.

[74] P. C. Martin, E. D. Siggia, and H. A. Rose. Statistical dynamics of classical systems. *Phys. Rev. A*, 8:423–437, 1973.

[75] C. Hubig I. P. McCulloch and U. Schollwöck. Generic construction of efficient matrix product operators. *Phys. Rev. B*, 95:035129, 2017.

[76] A. S. Monin and A. M. Yaglom. *Statistical Fluid Mechanics, Volume II: Mechanics of Turbulence*. Dover, 2007.

[77] E. Musharbash and F. Nobile. Dual dynamically orthogonal approximation of incompressible Navier Stokes equations with random boundary conditions. *J. Comput. Phys.*, 354:135–162, 2018.

[78] E. Musharbash, F. Nobile, and T. Zhou. Error analysis of the dynamically orthogonal approximation of time dependent random PDEs. *SIAM J. Sci. Comput.*, 37(2):A776–A810, 2015.

[79] A. Narayan and J. Jakeman. Adaptive Leja sparse grid constructions for stochastic collocation and high-dimensional approximation. *SIAM J. Sci. Comput.*, 36(6):A2952–A2983, 2014.

[80] M. Z. Nashed. Differentiability and related properties of nonlinear operators: Some aspects of the role of differentials in nonlinear functional analysis. In *Nonlinear Functional Anal. and Appl. (Proc. Advanced Sem., Math. Res. Center, Univ. of Wisconsin, Madison, Wis., 1970)*, pages 103–309. Academic Press, New York, 1971.

[81] I. Oseledets and E. Tyrtyshnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra Appl.*, 432(1):70–88, 2010.

[82] I. V. Oseledets. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33(5):2295–2317, 2011.

[83] R. Mahony P.-A. Absil and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, Princeton, NJ, 2008.

[84] R. Phythian. The functional formalism of classical statistical dynamics. *J. Phys A: Math. Gen.*, 10(5):777–788, 1977.

[85] A. Pinkus. *Ridge Functions*. Cambridge University Press, 2015.

[86] M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *J. Comput. Phys.*, 357:125–141, 2018.

[87] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:606–707, 2019.

[88] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

[89] M. Reed and B. Simon. *Methods of modern mathematical physics. I.* Academic Press, Inc. [Harcourt Brace Jovanovich, Publishers], New York, second edition, 1980. Functional analysis.

[90] H.-K. Rhee, R. Aris, and N. R. Amundson. *First-order partial differential equations, volume 1: theory and applications of single equations*. Dover, 2001.

[91] H. Risken. *The Fokker-Planck equation: methods of solution and applications*. Springer-Verlag, second edition, 1989. Mathematics in science and engineering, vol. 60.

[92] A. Rodgers, A. Dektor, and D. Venturi. Adaptive integration of nonlinear evolution equations on tensor manifolds. *J. Sci. Comput.*, 92(39), 2022.

[93] A. Rodgers and D. Venturi. Stability analysis of hierarchical tensor methods for time-dependent pdes. *J. Comput. Phys.*, 409:109341, 2020.

[94] A. Rodgers and D. Venturi. Implicit step-truncation integration of nonlinear PDEs on low-rank tensor manifolds. *arXiv:2207.01962*, pages 1–30, 2022.

[95] M. Rosenblatt. Remarks on a multivariate transformation. *Ann. Math. Stat.*, 23:470–472, 1952.

[96] G. Rudolph and M. Schmidt. *Differential geometry and mathematical physics. Part II. Fibre bundles, topology and gauge fields*. Springer, 2017.

[97] D. Salas and L. Thibault. On characterizations of submanifolds via smoothness of the distance function in Hilbert spaces. *J. Optim. Theory Appl.*, 182(1):189–210, 2019.

[98] F. Santambrogio. *Optimal transport for applied mathematicians*, volume 87 of *Progress in Nonlinear Differential Equations and their Applications*. Birkhäuser/Springer, Cham, 2015. Calculus of variations, PDEs, and modeling.

[99] T. P. Sapsis and P. F. Lermusiaux. Dynamically orthogonal field equations for continuous stochastic dynamical systems. *Phys. D*, 238(23-24):2347–2360, 2009.

[100] R. Schneider and A. Uschmajew. Approximation rates for the hierarchical tensor format in periodic Sobolev spaces. *J. Complexity*, 30(2):56–71, 2014.

[101] T. Schulte-Herbrüggen, S. J. Glaser, G. Dirr, and U. Helmke. Gradient flows for optimization in quantum information and quantum dynamics: foundations and applications. *Rev. Math. Phys.*, 22(6):597–667, 2010.

[102] T. Shi, M. Ruth, and A. Townsend. Parallel algorithms for computing the tensor-train decomposition. *arXiv:2111.10448*, pages 1–23, 2021.

[103] S. Sirisup and G.E. Karniadakis. A spectral viscosity method for correcting the long-term behavior of pod models. *Journal of Computational Physics*, 194(1):92–116, 2004.

[104] A. Spantini, D. Bigoni, and Y. Marzouk. Inference via low-dimensional couplings. *J. Mach. Learn. Res.*, 19:1–71, 2018.

[105] A. Uschmajew and B. Vandereycken. The geometry of algorithms using hierarchical tensors. *Linear Algebra Appl.*, 439(1):133–166, 2013.

[106] A. Uschmajew and A. Zeiser. Dynamical low-rank approximation of the vlasov-poisson equation with piecewise linear spatial boundary. *arXiv:2303.01856*, pages 1–19, 2023.

[107] D. Venturi. A fully symmetric nonlinear biorthogonal decomposition theory for random fields. *Phys. D*, 240(4-5):415–425, 2011.

[108] D. Venturi. Conjugate flow action functionals. *J. Math. Phys.*, (54):113502(1–19), 2013.

[109] D. Venturi. The numerical approximation of nonlinear functionals and functional differential equations. *Physics Reports*, 732:1–102, 2018.

[110] D. Venturi, M. Choi, and G.E. Karniadakis. Supercritical quasi-conduction states in stochastic rayleigh–bénard convection. *International Journal of Heat and Mass Transfer*, 55(13):3732–3743, 2012.

[111] D. Venturi and A. Dektor. Spectral methods for nonlinear functionals and functional differential equations. *Res. Math. Sci.*, 8(27):1–39, 2021.

[112] D. Venturi, T. P. Sapsis, H. Cho, and G. E. Karniadakis. A computable evolution equation for the joint response-excitation probability density function of stochastic dynamical systems. *Proc. R. Soc. A*, 468(2139):759–783, 2012.

[113] M. Lindsey E.M. Stoudenmire Y. Khoo YH. Hur, J. G. Hoskins. Generative modeling via tensor train sketching. *arXiv e-prints*, page arXiv:2202.11788v5, March 2023.

[114] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *J. Comput. Phys.*, 394:56–81, 2019.

[115] J. Zinn-Justin. *Quantum field theory and critical phenomena*. Oxford Univ. Press, fourth edition, 2002.