UNIVERSITY OF CALIFORNIA
SANTA CRUZ

STEP TRUNCATION METHODS FOR NONLINEAR EVOLUTION EQUATIONS ON TENSOR MANIFOLDS

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

DOCTOR OF PHILOSOPHY

in

APPLIED MATHEMATICS

by

Abram Rodgers

December 2023

The Dissertation of Abram Rodgers
is approved:

Professor Daniele Venturi, Chair

Professor Qi Gong

Professor Hongyun Wang

Peter Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Step Truncation Methods for Nonlinear Evolution Equations on Tensor Manifolds

by

Abram Rodgers

We develop new adaptive algorithms for temporal integration of nonlinear evolution equations on tensor manifolds. These algorithms, which we call step-truncation methods, are based on performing one time step with a conventional time-stepping scheme, followed by a truncation operation onto a tensor manifold. In particular, we develop a mathematical framework for the analysis of these algorithms which encompasses both explicit and implicit time stepping. With this framework we prove convergence of a wide range of step-truncation methods, including one-step and multi-step methods. These methods rely only on arithmetic operations between tensors, which can be performed by efficient and scalable parallel algorithms. Adaptive step-truncation methods can be used to compute numerical solutions of high-dimensional PDEs, which, have become central to many new areas of application such optimal mass transport, random dynamical systems, and mean field optimal control. Numerical applications are presented and discussed for a linear advection problem, a class of Fokker-Planck equations, the Allen-Cahn equation, the nonlinear Schrödinger, and a Burgers' equation with uncertain initial condition.

For Grandma and Kevin.

Acknowledgments

The time I spent in graduate school was not only a profound period in my life, but a profound period in world history. Namely, my experience was heavily influenced by the COVID-19 pandemic. Reading the mathematics below does not tell the human side of this story. Though I do not know if the mathematics I've written will be referred to in the coming decades, I do know that (as in centuries past) history books will speak about the experiences of scientists during the pandemic in the same manner that we speak of Newton's seclusion during a 1665 pandemic. I have no doubt that in the past few years someone other than myself had a world-changing breakthrough while sorrowfully staring out the window at empty streets. I'd like to share what those moments of discovery were like from a first-hand perspective.

I finished my final graduate coursework and transitioned into dissertation research in the spring of 2020, as COVID-19 came to the United States. The majority of breakthroughs in this work were developed during the various stay-at-home periods. A number of key mathematical concepts were developed during evacuations that occurred due to the CZU Lightning Complex Fires with Alec Dektor. I clearly remember packing up many of my belongings and getting in a car with my housemate Zachary Potter to shelter at his parents' home in Visalia, California. Thanks Zach, being trapped with you while the world seemed to unravel was fun. I remember living in a tiny apartment with you fondly, even though the world was pretty screwed up basically the whole time we lived together.

Alec and I collaborated mostly by calling over the phone. I would pace in a guest room in Visalia while verbally going through convoluted algebraic expressions with Alec. I remember working out the formulation of a normal vector to a fixed rank manifold with him in one of those calls. Eventually the fires ended.

Zach and I returned to our small apartment in downtown Santa Cruz. My long phone calls with Alec did not stop though. Together we authored two papers on rank adaptivity for tensor integrators. I can't properly express how grateful I am for the experience of collaboration I had with Alec. For me, the highlight of the experience was during manuscript revision when one reviewer referred to our work by saying, "The proposed scheme is not very surprising (in fact, it adheres very closely to the definition of local error) but, as far as I know, it has not been published or presented before." I think that the reviewer meant it as a compliment, considering that they recommended publication. However, I'm still unsure. I laugh about it. It is almost as though the reviewer is saying "Anyone could do this, but you are the first. Good enough." That said, I don't disagree with the reviewer.

Alec and I were undergraduates in the UC Santa Cruz Mathematics department at the same time. We transitioned to graduate students in the Applied Mathematics department around the same time, under the advising of Daniele Venturi. Alec, Daniele, the support from the two of you, both academically and emotionally, made this doctoral dissertation possible. Though this time was intense, I can't find a way to thank the two of you enough for your influence on me, professional and personal. Once, when we were taking an advanced undergraduate course in linear algebra together, Alec asked me if I wanted to do the homework together with him. I told him I already finished it and needed to work on other things. Sorry Alec! At least we solved some linear algebra problems together later on. I'll say we're even.

I also have to thank Sarah Mitchell. People don't tend to have the patience to listen to so much mathematics they're not working on. Hopefully you will be willing to listen to considerably more mathematics you're not working on.

Chapter 1

Introduction

Computing the solution of high-dimensional Partial Differential Equations (PDEs) has become central to many new areas of application such as random media [102], optimal transport [114], random dynamical systems [112, 113], mean field games [20], machine learning, and functional-differential equations [109]. These equations take the form

$$\frac{\partial f(\boldsymbol{x},t)}{\partial t} = \mathcal{G}\left(f(\boldsymbol{x},t),\boldsymbol{x}\right), \qquad f(\boldsymbol{x},0) = f_0(\boldsymbol{x}), \tag{1.1}$$

where $f : \Omega \times [0,T] \to \mathbb{R}$ is a d-dimensional (time-dependent) scalar field defined on the domain $\Omega \subseteq \mathbb{R}^d$ ($d \geq 2$), and $\mathcal{G}$ is a nonlinear operator which may depend on the variables $\boldsymbol{x} = (x_1, \ldots, x_d)$ and may incorporate boundary conditions. By discretizing (1.1) in Ω, e.g., by finite differences, finite elements, or pseudo-spectral methods, we obtain the system of ordinary differential equations

$$\frac{d\boldsymbol{f}(t)}{dt} = \boldsymbol{G}(\boldsymbol{f}(t)), \qquad \boldsymbol{f}(0) = \boldsymbol{f}_0. \tag{1.2}$$

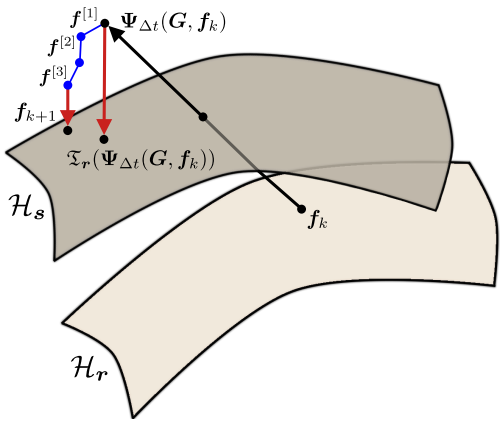


Figure 1.1: Sketch of implicit and explicit step-truncation integration methods. Given a tensor $\boldsymbol{f}_k$ with multilinear rank $\boldsymbol{r}$ on the tensor manifold $\mathcal{H}_{\boldsymbol{r}}$, we first perform an explicit time-step, e.g., with the conventional time-stepping scheme (1.6). The explicit step-truncation integrator then projects $\boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k)$ onto a new tensor manifold $\mathcal{H}_{\boldsymbol{s}}$ (solid red line). The multilinear rank $\boldsymbol{s}$ is chosen adaptively based on desired accuracy and stability constraints [92]. On the other hand, the implicit step-truncation method takes $\boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k)$ as input and generates a sequence of fixed-point iterates $\boldsymbol{f}^{[j]}$ shown as dots connected with blue lines. The last iterate is then projected onto a low rank tensor manifold, illustrated here also as a red line landing on $\mathcal{H}_{\boldsymbol{s}}$. This operation is equivalent to the compression step in the HT/TT-GMRES algorithm described in [40].

Here, $\boldsymbol{f} : [0, T] \to \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is a multi-dimensional array of real numbers (the solution tensor), and $\boldsymbol{G}$ is a tensor-valued nonlinear map (the discrete form of $\mathcal{G}$ corresponding to the chosen spatial discretization). The number of degrees of freedom associated with the solution $\boldsymbol{f}(t)$ to the Cauchy problem (1.2) is $N_{\text{dof}} = n_1 \cdot n_2 \cdots n_d$ at each time $t \geq 0$, which can be extremely large even for small d. For instance, the solution to the Boltzmann-BGK equation on a 6-dimensional flat torus [14] with 128 points in each variable x_i $(i = 1, \ldots, 6)$ yields $N_{\text{dof}} = 128^6 = 4398046511104$ degrees of freedom at each time t. As is well known, any straightforward spacial discretization of (1.1) in this manor inevitably leads to the so-called curse of dimensionality [11]. I.e. an exponential explosion in the computational storage cost or execution time to approximate the solution to a

given equation. However, not every particular solution to a given PDE requires that we store an exponentially large amount of data. For example, consider a linear advection equation with constant coefficients

$$\frac{\partial f(\boldsymbol{x},t)}{\partial t} = -\boldsymbol{a}\cdot\nabla f(\boldsymbol{x},t), \qquad f(\boldsymbol{x},0) = f_0(\boldsymbol{x}). \tag{1.3}$$

A standard exercise of introductory partial differential equations is to use the method of characteristics to show that the solution to the above problem is

$$f(\boldsymbol{x},t) = f_0(\boldsymbol{x}-\boldsymbol{a}t). \tag{1.4}$$

Therefore, if the initial condition admits a simple representation then so does the solution to the Initial Value Problem (IVP). E.g, if the solution factors as $f_0(x_1,x_2,\ldots,x_d) = f_0^1(x_1)f_0^2(x_2)\cdots f_0^d(x_d)$ (such as the ansatz of the separation of variables method) then

$$f(x_1,x_2,\ldots,x_d,t) = f_0^1(x_1-a_1t)f_0^2(x_2-a_2t)\cdots f_0^d(x_d-a_dt). \tag{1.5}$$

As seen in this example, if we are to store a numerical approximation to this solution, only $n_1+n_2+\cdots+n_d$ floating point numbers are required. Note the storage savings: exponential growth is reduced to polynomial growth. We essentially took a logarithm on the computational storage cost. And so we are motivated; To what extent is the special case of a separable solution generalizable? One successful generalization of this concept are the class of data compression algorithms known as low-rank tensor formats [63, 39, 24]. In a parallel research effort that has its roots in quantum field theory and quantum entanglement, researchers have recently developed a new generation of algorithms based on tensor networks and

low-rank tensor techniques to compute the solution of high-dimensional PDEs [57, 7, 13, 27, 58]. Tensor networks and low-rank tensor decompositions are essentially factorizations of entangled objects such as multivariate functions or operators, into networks of simpler objects which are amenable to efficient representation and computation. The process of building a tensor network relies on a hierarchical decomposition that can be visualized in terms of trees, and has its roots in the spectral theory for linear operators. Such rigorous mathematical foundations can be leveraged to construct high-order methods to compute the numerical solution of high-dimensional Cauchy problems of the form (1.2) at a cost that scales linearly with respect to the dimension d, and polynomially with respect to the tensor rank.

Upon applying a tensor decomposition approach to a given tensor or a given initial value problem, we are inherently introducing inaccuracies via the use of truncated singular value decompositions. In this dissertation we present algorithms to integrate (1.2) on classes of a low-rank tensor manifold which have rigorous consistency, stability, and convergence properties [93, 33, 92, 32, 58, 109]. These algorithms are known as step-truncation methods and they are based on integrating the solution $\boldsymbol{f}(t)$ off the tensor manifold for a short time using any conventional explicit time-stepping scheme, and then mapping it back onto the manifold using a tensor truncation operation (see Figure 1.1). To briefly describe these methods, let us discretize the ODE (1.2) in time with a one-step method on an evenly-spaced temporal grid as

$$\boldsymbol{f}_{k+1} = \boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k), \qquad \boldsymbol{f}_0 = \boldsymbol{f}(0), \tag{1.6}$$

where $\boldsymbol{f}_k$ denotes an approximation of $\boldsymbol{f}(k\Delta t)$ for $k = 0, 1, \ldots$, and $\boldsymbol{\Psi}_{\Delta t}$ is an increment function. To obtain a step-truncation integrator, we simply apply a

truncation operator $\mathfrak{T}_{\boldsymbol{r}}(\cdot)$, i.e., a nonlinear projection [45] onto a tensor manifold $\mathcal{H}_{\boldsymbol{r}}$ [107] with multilinear rank $\boldsymbol{r}$ to the scheme (1.6). This yields

$$\boldsymbol{f}_{k+1} = \mathfrak{T}_{\boldsymbol{r}}\left(\boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k)\right). \tag{1.7}$$

The need for tensor rank-reduction when iterating (1.6) can be easily understood by noting that tensor operations such as the application of an operator to a tensor and the addition between two tensors naturally increase tensor rank [64]. Hence, iterating (1.7) with no rank reduction can yield a fast increase in tensor rank, which, in turn, can tax computational resources heavily.

The dissertation is organized as follows. In Chapter 2, we provide an introduction to the fundamental concepts of low-rank tensor decompositions as well as the important operators required for the analysis of step-truncation methods. In Chapter 4 we develop a mathematical analysis of step-truncation methods. In particular, we provide rigorous results on stability and convergence for explicit and implicit time stepping algorithms. We also present a variant of Newton's iteration for the solutions to nonlinear equations on tensor manifolds. In Chapter 3 we provide a survey of particular step-truncation temporal integrators and show that each one satisfies the framework established in the prior chapter or a modification of it. We finish the chapter with an analysis of the various tunable parameters present in the step-truncation methods. In Chapter 5 we provide a wide array of applications. We demonstrate the theorems for fixed rank and rank-adaptive explicit integrators through the numerical solution of advection and diffusion problems. We also present numerical demonstrations for implicit step-truncation tensor integrators via the Allen-Cahn equation, the Fokker-Planck equation, and the nonlinear Schrödinger equation. Additionally, we shown an application to a Burgers' equation with an uncertain initial condition.

Chapter 2

Tensor Decompositions

In this chapter, we cover requisite background material to introduce step-truncation methods for solving partial differential equations. The topics of this chapter focus on introducing tensor decompositions and their mathematical properties. Informally, a tensor decomposition is a data compression algorithm. It transforms a multidimensional array of real or complex numbers into a list of arrays of smaller dimension. These decompositions are designed with the common operations of linear algebra in mind, allowing for computations to be done with these compressed objects without ever computing a decompressed, or full, array.

2.1 Tensor Decomposition Formats

In this work, when we say "tensor decomposition," or "low rank tensor format," we are referring to a method of writing a tensor $\boldsymbol{f} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ [1] as a sequence of Kronecker products of matrices with small computer memory storage size. Clearly, reducing storage cost of an array is extremely useful, effectively allowing us to use a

[1] We recall that $\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ and $\bigotimes_{j=1}^{d} \mathbb{R}^{n_j}$ are isomorphic and consider them identical without confusion.

cheaper (monetarily) computer for a more expensive (computationally) problem. Three well-known tensor decompositions are the Hierarchical Tucker, [45] (HT henceforth) Tensor Train, [73, 31] (TT henceforth) and the Tucker decomposition [105]. The HT format is based on using a binary tree data structure to preform a lossy compression of a tensor via matrix factorizations. TT is a specialization of HT (See [74], Remark 1) which simplifies the binary tree in use. The memory scaling is similar to HT, requiring at most $dn_{\max}r_{\max}^2$ real numbers stored. The Tucker decomposition follows a different setup, requiring up to $r_{\max}(dn_{\max} + r_{\max}^{d-1})$ stored real numbers. Due to the exponential memory growth in rank, we will not go into the detail of the Tucker format.

2.1.1 Singular Value Decomposition

The most straightforward example of what one may refer to as a tensor decomposition is simply the singular value decomposition (SVD). We recall from [103], Theorem 5.9, that we can approximate a matrix by its leading singular values and their vectors.

Theorem 1 (Low Rank Approximation of Matrices)**.** *Let $\boldsymbol{f} \in \mathbb{R}^{m\times n}$ be a rank r matrix. Let $\boldsymbol{f} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ be the reduced singular value decomposition* [2] *of $\boldsymbol{f}$, and let $1 \leq s < r$. Then*

$$\begin{aligned}\left\|\boldsymbol{f} - \sum_{k=1}^{s} \sigma_k \boldsymbol{u}_k \otimes \boldsymbol{v}_k\right\| &= \inf_{\hat{\boldsymbol{f}}\in\mathcal{H}_s} \left\|\boldsymbol{f} - \hat{\boldsymbol{f}}\right\| \\ &= \sqrt{\sum_{k=s+1}^{r} \sigma_k^2},\end{aligned}$$

where $\boldsymbol{u}_k, \boldsymbol{v}_k$ are the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ respectively and

[2] i.e. $\boldsymbol{U} \in \mathbb{R}^{m\times r}$, $\boldsymbol{V} \in \mathbb{R}^{n\times r}$, $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I} = \boldsymbol{V}^\top\boldsymbol{V}$ and, $\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$.

$\mathcal{H}_s = \{\ \boldsymbol{f} \in \mathbb{R}^{m\times n} \mid rank(\hat{\boldsymbol{f}}) = s\ \}$.

Clearly, this approximation is better whenever the singular values decay exceptionally fast. It is this property which encourages us to study tensor formats. In the SVD case, our tensor format is will be the list $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V})$. By explicitly storing these matrices without further compressions (e.g. Householder reflectors, storing only the diagonal of $\boldsymbol{\Sigma}$, etc.) we can see that the storage cost is exactly $s^2 + (m+n)s$. So if rank were to be fixed, then we have taken the quadratic-like cost of mn and reduced it to a linear cost of $O(m+n)$. Higher order tensor spaces will have decompositions which retain this property.

2.1.2 Hierarchical Tucker Decomposition

The hierarchical Tucker decomposition is a rewriting of a tensor in a recursive manner which eliminates exponential scaling of memory usage in the order of the tensor. This format is clearly defined and explained in [45, 46], in which it is described as an extension of the Tucker decomposition [105]. Since our work is an application of tensor formats rather than a study of their properties, we will give the case of an order 3 tensor in HT format. Then we state the general case memory scaling results and allow the reader to investigate the sources [45, 46, 105] for further technical detail.

We begin by introducing a few required notations. Let $\boldsymbol{f} \in \mathbb{R}^{n_1\times n_2\times n_3}$ and denote its entries by $\boldsymbol{f}[i,j,k]$. By grouping the indexes as $[i,[j,k]]$ and considering the tuple $[j,k]$ as a single index symbol $\kappa = [j,k]$, we see that we can reshape $\boldsymbol{f}$ into $\boldsymbol{f}^{(1)}$as

$$\boldsymbol{f}[i,j,k] = \boldsymbol{f}^{(1)}[i,\kappa]. \tag{2.1}$$

This is a called a matricization of $\boldsymbol{f}$ [3]. It is defined via a vector space isomorphism

[3]Also called an array flattening.

$\mathbb{R}^{n_1 \times n_2 \times n_3} \simeq \mathbb{R}^{n_1} \otimes \mathbb{R}^{n_2 \cdot n_3}$, which reshapes the high order array $\boldsymbol{f}$ into an order 2 array with the same number of entries. Specifically, this is the mode (1) matricization, hence the notation $\boldsymbol{f}^{(1)}$. Similarly, the mode (23) matricization comes from treating κ as a row index.

$$\boldsymbol{f}[i, j, k] = \boldsymbol{f}^{(23)}[\kappa, i] \tag{2.2}$$

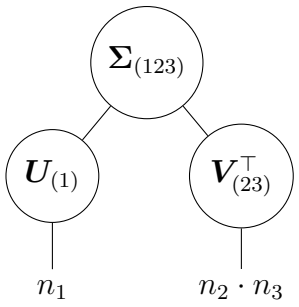
This corresponds to the isomorphism $\mathbb{R}^{n_1 \times n_2 \times n_3} \simeq \mathbb{R}^{n_2 \cdot n_3} \otimes \mathbb{R}^{n_1}$ and is a transposed version of the mode (1) matricization. The implicitly defined inverse isomorphism is called a dematricization in the corresponding mode. A particularly important matricization is the vectorization,

$$\mathrm{vec}(\boldsymbol{f}) = \boldsymbol{f}^{(123)}, \tag{2.3}$$

which is just a reinterpretation of $\boldsymbol{f}$ as an element of $\mathbb{R}^{n_1 \cdot n_2 \cdot n_3}$, i.e. a 1-dimensional array of reals. The HT format uses SVDs of these flattenings in order to construct an approximation of $\boldsymbol{f}$. Specifically, we see that $\boldsymbol{f}^{(1)}$ is a matrix and so we apply Theorem 1. This results in a matrix decomposition

$$\boldsymbol{f}^{(1)} \approx \boldsymbol{U}_{(1)} \boldsymbol{\Sigma}_{(123)} \boldsymbol{V}_{(23)}^{\top}, \tag{2.4}$$

where the approximation has rank $r_{(1)}$. Specifically, we have $\boldsymbol{U}_{(1)} \in \mathbb{R}^{n_1 \times r_{(1)}}$, $\boldsymbol{\Sigma}_{(123)} \in \mathbb{R}^{r_{(1)} \times r_{(1)}}$, $\boldsymbol{V}_{(23)} \in \mathbb{R}^{(n_2 \cdot n_3) \times r_{(1)}}$. There is a visual closely related to the factorization being performed. We can think of the process as the branching of a binary tree.



Linked nodes in the tree represent a summation along an index as a visual aid to the matrix multiplication in equation (2.4). Unlinked nodes are used to represent the size of the full tensor. We now count the number of stored matrix entries. From (2.4), it is immediate that $r_{(1)}(n_1 + r_{(1)} + n_2 n_3)$ is our memory cost. We now would like to compress the $\boldsymbol{f}$ further by projecting $\boldsymbol{V}_{(23)}$. By applying the following additional singular value decompositions,

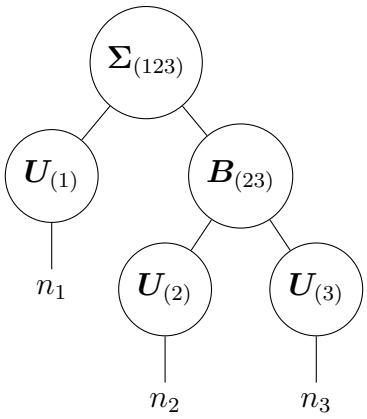
$$\boldsymbol{f}^{(2)} \approx \boldsymbol{U}_{(2)} \boldsymbol{\Sigma}_{(213)} \boldsymbol{V}_{(13)}^{\top}, \tag{2.5}$$

$$\boldsymbol{f}^{(3)} \approx \boldsymbol{U}_{(3)} \boldsymbol{\Sigma}_{(312)} \boldsymbol{V}_{(12)}^{\top}, \tag{2.6}$$

we are able to choose different ranks to approximate along each index of $\boldsymbol{f}[i, j, k]$. We only need to explicitly compute the left singular vectors, which are $\boldsymbol{U}_{(2)} \in \mathbb{R}^{n_2 \times r_{(2)}}$ and $\boldsymbol{U}_{(3)} \in \mathbb{R}^{n_3 \times r_{(3)}}$. To find our projected $\boldsymbol{V}_{(23)}$ we compute

$$\boldsymbol{B}_{(23)} = (\boldsymbol{U}_{(3)}^{\top} \otimes_{\mathcal{K}} \boldsymbol{U}_{(2)}^{\top}) \boldsymbol{V}_{23} \in \mathbb{R}^{r_{(2)} \cdot r_{(3)} \times r_{(1)}}, \tag{2.7}$$

where $\otimes_{\mathcal{K}}$ is the Kronecker product of matrices. $\boldsymbol{B}_{(23)}$ is called a transfer tensor. The binary tree diagram of this step is



One of the major theorems of [45] is that by keeping more singular vectors in each SVD, we can reconstruct an approximation of $\boldsymbol{f}$ as

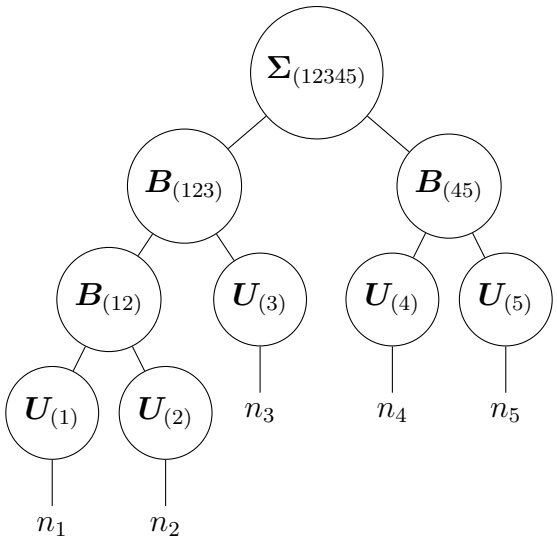
$$\text{vec}(\boldsymbol{f}) \approx (((\boldsymbol{U}_{(3)} \otimes_{\mathcal{K}} \boldsymbol{U}_{(2)})\boldsymbol{B}_{(23)}) \otimes_{\mathcal{K}} \boldsymbol{U}_{(1)}) \otimes_{\mathcal{K}} \text{vec}(\boldsymbol{\Sigma}_{(123)}) \tag{2.8}$$

We can now count the storage cost as $n_1 r_{(1)} + n_2 r_{(2)} + n_3 r_{(3)} + r_{(1)} r_{(2)} r_{(3)} + r_{(1)}^2$, rather than the uncompressed array cost of $n_1 n_2 n_3$. Following this process (called root-to-leaves truncation) for an order d tensor will result in an overall storage cost bound by

$$d n_{\max} r_{\max} + (d-2) r_{\max}^3 + r_{\max}^2 \tag{2.9}$$

where $n_{\max} = \max\{n_1, n_2, \ldots, n_d\}$ and $r_{\max}$ is the rank of the SVD with the most saved left singular vectors. If the ranks are near the size of $n_{\max}$, then clearly this polynomial beats the exponentially growing storage cost of full tensor storage, which is $n_{\min}^d \leq n_1 \cdot n_2 \cdot \ldots \cdot n_d \leq n_{\max}^d$.

The tree diagrams (called dimension trees) described for the example above also apply to the higher order tensor case. Here is a tree corresponding to an order 5 HT decomposition.



The reconstruction of this tensor is given by

$$\mathrm{vec}(\boldsymbol{f}) = (((\boldsymbol{U}_{(5)} \otimes_{\mathcal{K}} \boldsymbol{U}_{(4)})\boldsymbol{B}_{(45)})\otimes_{\mathcal{K}} ((\boldsymbol{U}_3\otimes_{\mathcal{K}}((\boldsymbol{U}_{(2)} \otimes_{\mathcal{K}} \boldsymbol{U}_{(1)})\boldsymbol{B}_{(12)}))\boldsymbol{B}_{(123)})) \otimes_{\mathcal{K}} \mathrm{vec}(\boldsymbol{\Sigma}_{(12345)}).$$

The general algorithms for approximating a tensor in the HT format corresponding to any desired binary tree are given in [45, 46, 64]. A key property of this decomposition and all tensor decompositions used in this work is that basic arithmetic operations between tensors do not require reconstructing the full array. Using multilinear formulas for operations involving the Kronecker product, one may create a new HT tensor with equal or higher ranks upon scalar multiplication, adding HT tensors, or multiplying HT tensors.

2.1.3 Tensor Train format

In this section, we overview the mathematical definition of the Tensor Train format. Tensor train is a direct simplification of the Hierarchical Tucker format, making the general case straightforward to present. We provide a number of the

arithmetic algorithms for the format as well.

Definition 1 (Tensor Train)**.** *A tensor in* $\boldsymbol{f} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ *is in Tensor Train (TT) format if there is an array of positive integers (called the TT rank)* $\boldsymbol{r} = (r_0, r_1, r_2, \ldots, r_{d-1}, r_d)$ *with* $r_0 = 1 = r_d$ *and a list of order 3 tensors (called the TT cores)* $\boldsymbol{C} = (\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_d)$ *where* $\boldsymbol{C}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ *such that the entries of* $\boldsymbol{f}$ *may be written as the iterated matrix product*

$$\boldsymbol{f}[i_1, i_2, \ldots, i_d] = \boldsymbol{C}_1[1, i_1, :]\boldsymbol{C}_2[:, i_2, :] \cdots \boldsymbol{C}_{d-1}[:, i_{d-1}, :]\boldsymbol{C}_d[:, i_d, 1]. \tag{2.10}$$

Though rather involved at first glance, the above expression may be derived by writing down a multivariate function series expansion by the method of separation of variables, rearranging the expression into a finite sequence of infinite matrix products, truncating the series so that the matrix products are finite, then discretizing in space so that the multivariate function is sampled on a tensor product grid. From this perspective, it becomes apparent that the tensor train cores represent a 2 dimensional array of functions of a single variable and the ranks are the number of functions present in a series expansion approximation. A more concrete exploration of this perspective is given in [32, 33].

Due to the iterated matrix product definition of the format, it becomes apparent that a number of arithmetic operations may be represented in the format by producing a new TT tensor with different ranks. In particular if $\boldsymbol{f}, \boldsymbol{g}$ have TT

Algorithm 1: Left Orthogonalization

Data: A tensor $\boldsymbol{f}$ in TT format with cores $(\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_d)$.
Result: A tensor $\boldsymbol{g} = \boldsymbol{f}$ in TT format with left orthogonal cores.

Reserve memory for each core $\boldsymbol{D}_c$, using the sizes of $\boldsymbol{C}_c$.
$\boldsymbol{D}_1 = \boldsymbol{C}_1$
for $c = 1, 2, \ldots, d-1$ **do**
  $[\boldsymbol{Q}_c, \boldsymbol{R}_c] = QR(\mathtt{V}(\boldsymbol{D}_c))$
  $\boldsymbol{H}_{c+1} = \boldsymbol{R}_c \mathtt{H}(\boldsymbol{C}_{c+1})$
  $\boldsymbol{D}_c[:] = \boldsymbol{Q}_c[:]$
  $\boldsymbol{D}_{c+1}[:] = \boldsymbol{H}_{c+1}[:]$
end
Set the cores of $\boldsymbol{g}$ as $(\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots, \boldsymbol{D}_d)$

core lists $\boldsymbol{C}, \boldsymbol{D}$ then by a simple inductive argument, we have

$$\begin{aligned}
&\boldsymbol{f}[i_1, i_2, \ldots, i_d] + \boldsymbol{g}[i_1, i_2, \ldots, i_d] \\
&\quad = \left[\boldsymbol{C}_1[1, i_1, :] \;\middle|\; \boldsymbol{D}_1[1, i_1, :]\right] \left[\begin{array}{c|c} \boldsymbol{C}_2[:, i_2, :] & \mathbf{0} \\ \hline \mathbf{0} & \boldsymbol{D}_2[:, i_2, :] \end{array}\right] \cdots \\
&\qquad\qquad \cdots \left[\begin{array}{c|c} \boldsymbol{C}_{d-1}[:, i_{d-1}, :] & \mathbf{0} \\ \hline \mathbf{0} & \boldsymbol{D}_{d-1}[:, i_{d-1}, :] \end{array}\right] \left[\begin{array}{c} \boldsymbol{C}_1[:, i_d, 1] \\ \hline \boldsymbol{D}_1[:, i_d, 1] \end{array}\right].
\end{aligned} \tag{2.11}$$

Thus the sum $\boldsymbol{w} = \boldsymbol{f} + \boldsymbol{g}$ is a TT tensor with cores obtained by concatenating those of $\boldsymbol{f}$ and $\boldsymbol{g}$. Scalar multiplication is straightforward, just scale any of the cores of the tensor.

$$\alpha \boldsymbol{f}[i_1, i_2, \ldots, i_d] = (\alpha \boldsymbol{C}_1[1, i_1, :])\, \boldsymbol{C}_2[:, i_2, :] \cdots \boldsymbol{C}_{d-1}[:, i_{d-1}, :] \boldsymbol{C}_d[:, i_d, 1]. \tag{2.12}$$

In order to perform more sophisticated operations such as tensor truncations, we must frequently reshape the tensor cores into lower dimensional arrays. Two

particularly useful reshapings are the horizontal flattening

$$\begin{aligned} \texttt{H} : \mathbb{R}^{a\times b\times c} &\longrightarrow \mathbb{R}^{a\times (bc)} \\ \boldsymbol{C}[i,j,k] &\longmapsto \boldsymbol{H}[i, j+bk], \end{aligned}$$

and the vertical flattening,

$$\begin{aligned} \texttt{V} : \mathbb{R}^{a\times b\times c} &\longrightarrow \mathbb{R}^{(ab)\times c} \\ \boldsymbol{C}[i,j,k] &\longmapsto \boldsymbol{V}[i+aj, k]. \end{aligned}$$

The above maps are two dimensional array analogs of the vectorization of a tensor, in which we have the coordinates are listed as $\boldsymbol{C}[i,j,k] = \boldsymbol{v}[i + aj + abk]$. `H` and `V` are also equivalent to the mode-1 matricization and a transposed mode-3 matricization respectively. Note that when the entries of a tensor are stored contiguously in computer memory, no memory movement or copying needs to be done to interpret a tensor as its flattening or vectorization. To undo the flattening in an algorithm, we denote copying all the entries as $\boldsymbol{C}[:] = \boldsymbol{H}[:]$ or $\boldsymbol{C}[:] = \boldsymbol{V}[:]$.

2.1.4 Orthogonalization and Truncation

The general procedure of truncating the ranks of a TT tensor comes in two phases. The first phase is to collect all the norm of the tensor into a single core through a sequence of matrix factorizations, leaving all other cores to have Frobenius norm 1. Then a sequence of SVDs are applied, multiplying a non-unitary factor from a tensor train core to its neighboring cores. The first phase is called orthogonalization of a TT tensor and we now present an algorithm for it.

There are two variants of orthogonalization we will discuss. One is Left-to-Right orthogonalization and the other Right-to-Left orthogonalization. They are

Algorithm 2: Right Orthogonalization

Data: A tensor $\boldsymbol{f}$ in TT format with cores $(\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_d)$.
Result: A tensor $\boldsymbol{g} = \boldsymbol{f}$ in TT format with right orthogonal cores.

Reserve memory for each core $\boldsymbol{D}_c$, using the sizes of $\boldsymbol{C}_c$.
$\boldsymbol{D}_d = \boldsymbol{C}_d$
for $c = d, d-1, \ldots, 2$ **do**
  $[\boldsymbol{L}_c, \boldsymbol{Q}_c] = LQ(\mathtt{H}(\boldsymbol{D}_c))$
  $\boldsymbol{H}_{c-1} = \mathtt{V}(\boldsymbol{C}_{c-1})\boldsymbol{L}_c$
  $\boldsymbol{D}_c[:] = \boldsymbol{Q}_c[:]$
  $\boldsymbol{D}_{c-1}[:] = \boldsymbol{H}_{c-1}[:]$
end
Set the cores of $\boldsymbol{g}$ as $(\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots, \boldsymbol{D}_d)$

the same algorithm mirrored across the middle core of the tensor train and each step in one is a transpose of its counterpart in the other. A TT tensor is left orthogonal if for $c = 1, 2, \ldots, d-1$, we have $\mathtt{V}(\boldsymbol{C}_c)^\top \mathtt{V}(\boldsymbol{C}_c) = \boldsymbol{I}$. Similarly, a TT tensor is right orthogonal if for $c = 2, 3, \ldots, d$, we have $\mathtt{H}(\boldsymbol{C}_c)\mathtt{H}(\boldsymbol{C}_c)^\top = \boldsymbol{I}$. One may transform the cores of a TT tensor to be left orthogonal while the full tensor is unchanged with the same observations seen via the group invariance introduced in Section 2.3. All we must do is sequentially flatten, orthogonalize by QR decompoosition, then unflatten each core, until the final one is reached. For the right orthogonal case, the procedure is largely identical, though we use LQ factorization, the lower-triangular orthogonalizing decomposition, instead. These processes are summarized in Algorithm 1 and Algorithm 2.

We now describe one algorithm for truncation of a tensor train given an orthogonalized TT tensor. The process is essentially the same as orthogonalization, we apply a sequence of orthogonal matrix factorizations to each core, multiplying one of the terms into that core's left or right neighbor. We then replace the current core with the entries of an orthogonal factor. The difference in this algorithm is that we apply an SVD, rather than QR or LQ factorization. The term

Algorithm 3: Left Truncation

Data: A tensor $\boldsymbol{f}$ in TT format with cores $(\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_d)$ and a desired accuracy ε.

Result: A tensor $\boldsymbol{g}$ so that $\|\boldsymbol{g} - \boldsymbol{f}\| \leq \varepsilon$ in TT format with left orthogonal cores.

Reserve memory for each core $\boldsymbol{D}_c$, using the sizes of $\boldsymbol{C}_c$.
Set $(\hat{\boldsymbol{C}}_1, \hat{\boldsymbol{C}}_2, \ldots, \hat{\boldsymbol{C}}_d)$ as a right orthogonalization of $\boldsymbol{f}$ by Algorithm 2.
$\hat{\varepsilon} = \varepsilon / \sqrt{d-1}$
$\boldsymbol{D}_1 = \hat{\boldsymbol{C}}_1$
for $c = 1, 2, \ldots, d-1$ **do**
  $[\boldsymbol{U}_c, \boldsymbol{\Sigma}_c, \boldsymbol{V}_c^\top] = SVD(\mathtt{V}(\boldsymbol{D}_c), \hat{\varepsilon})$ (Truncated SVD with tolerance $\hat{\varepsilon}$.)
  $\boldsymbol{H}_{c+1} = \boldsymbol{\Sigma}_c \boldsymbol{V}_c^\top \mathtt{H}(\hat{\boldsymbol{C}}_{c+1})$
  $\boldsymbol{D}_c[:] = \boldsymbol{U}_c[:]$
  $\boldsymbol{D}_{c+1}[:] = \boldsymbol{H}_{c+1}[:]$
end
Set the cores of $\boldsymbol{g}$ as $(\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots, \boldsymbol{D}_d)$

"truncation" comes from truncating the series expansion of a matrix discussed in Theorem 1 up to a desired tolerance. This truncation is described as the rounding algorithm in [82]. We present two variants of it here as Algorithms 3 and 4. There is no clear general rule for preferring one truncation algorithm to the other. They have identical computational costs and similar formulations. It is possible that differing runtime will occur based on row major versus column major layout of the matrices, though this is highly dependent on the specific memory mapping and workspaces used in the SVD, QR, and LQ calls.

2.2 Distributed Memory Parallel Tensor Decompositions

In this section, we provide a discussion of parallel memory layouts of Tensor decompositions and a presentation of a particular distributed memory layout for

Algorithm 4: Right Truncation

Data: A tensor $\boldsymbol{f}$ in TT format with cores $(\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_d)$ and a desired accuracy ε.

Result: A tensor $\boldsymbol{g}$ so that $\|\boldsymbol{g} - \boldsymbol{f}\| \leq \varepsilon$ in TT format with right orthogonal cores.

Reserve memory for each core $\boldsymbol{D}_c$, using the sizes of $\boldsymbol{C}_c$.
Set $(\hat{\boldsymbol{C}}_1, \hat{\boldsymbol{C}}_2, \ldots, \hat{\boldsymbol{C}}_d)$ as a left orthogonalization of $\boldsymbol{f}$ by Algorithm 1.
$\hat{\varepsilon} = \varepsilon/\sqrt{d-1}$
$\boldsymbol{D}_1 = \hat{\boldsymbol{C}}_1$
for $c = d, d-1, \ldots, 2$ **do**
  $[\boldsymbol{U}_c, \boldsymbol{\Sigma}_c, \boldsymbol{V}_c^\top] = SVD(\mathtt{H}(\boldsymbol{D}_c), \hat{\varepsilon})$ (Truncated SVD with tolerance $\hat{\varepsilon}$.)
  $\boldsymbol{H}_{c-1} = \mathtt{V}(\hat{\boldsymbol{C}}_{c-1})\boldsymbol{U}_c\boldsymbol{\Sigma}_c$
  $\boldsymbol{D}_c[:] = \boldsymbol{U}_c[:]$
  $\boldsymbol{D}_{c-1}[:] = \boldsymbol{H}_{c-1}[:]$
end
Set the cores of $\boldsymbol{g}$ as $(\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots, \boldsymbol{D}_d)$

the Tensor Train format.

When considering a scalable parallelization of the algorithms for tensor decomposition arithmetic and truncation, the choice of tensor decomposition itself becomes quite important. This is in contrast to the mathematical framework for step-truncation methods. The theory we develop in Chapter 4 is generally agnostic to the choice of dimension tree. On the other hand, operations such as orthogonalizing and truncating require sequences of QR factorizations and SVDs which easily cause computational bottlenecks if not properly treated.

Of the possible computer memory layouts, two options are the most apparent. One option is to note that the dimension tree is a graph. Parallel computing systems are themselves physical graphs of cables and compute nodes, and so it seems intuitive to program the computers to mimic the graph of the dimension tree. This is the approach taken to parallelize the HTucker format in [46]. In this approach, the truncation algorithms are implemented by having every compute node wait until it receives a factor from its parent or child as defined by the HTucker tree

(see Chapter 2.1.2). This choice does have a number of problems though. Chiefly, one must use exactly as many compute nodes as there are nodes in the HTucker tree. One may not over-allocate compute nodes if the dimension of the tensor being studied is fixed. Additionally, if any individual node has particularly large multilinear ranks, portions of the parallel computing system will need to wait on the factorizations (QR or SVD) of that node to complete before continuing execution. There are also other concerns on this memory layout, including that the QR, SVD, and matrix products are serial computations. A second approach is to pick a specific class of dimension trees and then optimize the parallel algorithms to perform well for that class of trees. This is the approach of [30]. They parallelize the Tensor Train format, which corresponds to an HTucker tree where tensor multi-indexes are always partitioned into a singleton index and a remainder multi-index [74]. The approach is to distribute every tensor in the factorization evenly across all compute nodes, resulting in perfect memory balancing. Then, all matrix factorization and product algorithms are reformulated with this distributed layout in mind. Due to the greatly improved computational balancing, we follow this approach.

2.2.1 Distributed Memory Tensor Train

Among the arithmetic operations performed on the tensor train during the evolution of the solution to a differential equation, the most costly is the truncation, due to the fact that it requires data access and editing of every core twice. Due to the sequential nature of the loops in Algorithms 1 to 4 we see that parallelizing the inner loop would be most effective. Since the rank of the tensor train is expected to change frequently during program execution, we must split the memory stored on a total of P compute nodes in a manner independent of

the tensor rank.

To this end, we introduce a $P \times d$ memory partition matrix $\boldsymbol{M}$ with positive integer values so that for each core $\boldsymbol{C}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$, we have $\sum_{p=1}^{P} \boldsymbol{M}[p,k] = n_k$. This matrix describes a block-tensor storage layout for the tensor train core list. More precisely, for each core $\boldsymbol{C}_k$, we define an array of core blocks $(\boldsymbol{C}_k^1, \boldsymbol{C}_k^2, \ldots, \boldsymbol{C}_k^P)$ with array sizes defined via $\boldsymbol{C}_k^p \in \mathbb{R}^{r_{k-1} \times \boldsymbol{M}[p,k] \times r_k}$ so that

$$\boldsymbol{C}_k[i,:,j] = \begin{bmatrix} \boldsymbol{C}_k^1[i,:,j] \\ \boldsymbol{C}_k^2[i,:,j] \\ \vdots \\ \boldsymbol{C}_k^P[i,:,j] \end{bmatrix}. \tag{2.13}$$

Each distributed memory compute node with index p stores the core list $(\boldsymbol{C}_1^p, \boldsymbol{C}_2^p, \ldots, \boldsymbol{C}_d^p)$. This memory layout is designed so that

$$\mathtt{V}(\boldsymbol{C}_k) = \begin{bmatrix} \mathtt{V}(\boldsymbol{C}_k^1) \\ \mathtt{V}(\boldsymbol{C}_k^2) \\ \vdots \\ \mathtt{V}(\boldsymbol{C}_k^P) \end{bmatrix} \quad \text{and} \quad \mathtt{H}(\boldsymbol{C}_k) = \begin{bmatrix} \mathtt{H}(\boldsymbol{C}_k^1) & | & \mathtt{H}(\boldsymbol{C}_k^2) & | & \cdots & | & \mathtt{H}(\boldsymbol{C}_k^P) \end{bmatrix}.$$

We may therefore compute a flattening of the cores without any memory movements or cross-node communications by reinterpreting the array storage offsets in column major layout. Additionally, sums and scalar multiplications of tensors may also be computed in parallel without communications.

This parallel layout also allows for straightforward applications of finite difference stencils. Consider a finite difference stencil which requires s many points where the number of 1 dimensional ghost cells required is $g = (s-1)/2$. Let $\boldsymbol{D}_{FD}$ be the linear operator for this stencil. To apply a partial derivative in variable k to

a TT tensor, we replace the entries of core $\boldsymbol{C}_k$ with the entries of $\boldsymbol{D}_{FD}$ applied to equation (2.13) for each i, j. To perform the parallel version of this stencil, we first perform a nonblocking communication to deliver the ghost cells of compute node p to their neighboring nodes $p \pm 1$. For nodes $p = 1$ or $p = P$, we instead rely on discrete boundary condition formulas, communicating again between nodes $p = 1$ and $p = P$ if periodic boundary conditions are required. After boundary condition communications, every processor has an appropriate local version of the finite difference stencil, which is identical save for the definition of the ghost cells.

Numerical integration is similar, though instead of sharing ghost cells of compute node p with compute node $p \pm 1$, we need only pass the data with compute node $p + 1$, due to the one-sided stencil of a cumulative summation formula.

2.2.2 Tall-Skinny QR Factorization

In order to truncate a TT tensor with the described block memory layout, we must perform a distributed memory QR factorization on a matrix of the form

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \vdots \\ \boldsymbol{A}_P \end{bmatrix},$$

or equivalently, perform an LQ factorization on the horizontally concatenated transpose of $\boldsymbol{A}$. For mathematical simplicity, we only present the QR variant, though our associated code has both variants. One such factorization amenable to this layout is the Tall-Skinny QR (TSQR) factorization [35]. This algorithm uses a binary tree structure to define a communication pattern for decomposing the QR factorization of $\boldsymbol{A}$ into a collection of smaller QR factorizations, avoiding

Algorithm 5: Parallel Left Orthogonalization

,

Data: A tensor $\boldsymbol{f}$ in distributed memory TT format with cores $([\boldsymbol{C}_1^p], [\boldsymbol{C}_2^p], \ldots, [\boldsymbol{C}_d^p])$.

Result: A tensor $\boldsymbol{g} = \boldsymbol{f}$ in distributed memory TT format with left orthogonal cores.

Reserve memory for each core block $\boldsymbol{D}_c^p$, using the sizes of $\boldsymbol{C}_c^p$.
$\boldsymbol{D}_1^p = \boldsymbol{C}_1^p$
for $c = 1, 2, \ldots, d-1$ **do**
  $[\boldsymbol{Q}_c^p, \boldsymbol{R}_c] = TSQR(\mathtt{V}(\boldsymbol{D}_c^p))$
  $\boldsymbol{H}_{c+1}^p = \boldsymbol{R}_c \mathtt{H}(\boldsymbol{C}_{c+1}^p)$
  $\boldsymbol{D}_c^p[:] = [\boldsymbol{Q}_c^p[:]$
  $\boldsymbol{D}_{c+1}^p[:] = \boldsymbol{H}_{c+1}^p[:]$
end
Set the cores of $\boldsymbol{g}$ as $([\boldsymbol{D}_1^p], [\boldsymbol{D}_2^p], \ldots, [\boldsymbol{D}_d^p])$

communications if possible. By factoring the blocks into their own QR factorizations, we see that the Q factor may be presented as a sequence of products of block diagonal matrices. This process is then applied to a recursive binary tree to track the nesting of the orthogonal Q factor. The final result is an orthogonal matrix $\boldsymbol{Q} = [\boldsymbol{Q}_1; \boldsymbol{Q}_2; \ldots; \boldsymbol{Q}_P]$ distributed in the same block layout as $\boldsymbol{A}$ and an upper triangular matrix $\boldsymbol{R}$ copied across all compute nodes. See Figure 2.1 for an algebraic representation of the case $P = 4$ and its corresponding binary tree storage structure. The transposed version of this algorithm is called the Wide-Fat LQ (WFLQ) factorization and has the same tree data storage structure, but every matrix factorization is transposed.

2.2.3 Parallel Orthogonalization and Truncation

We now introduce a parallelization of Algorithms 1 through 4. It can be seen that parallelizing the orthogonalization algorithms is as simple as replacing the QR factorizations with their TSQR variants. However, the truncation requires

$$
\begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \boldsymbol{A}_3 \\ \boldsymbol{A}_4 \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_1\boldsymbol{R}_1 \\ \boldsymbol{Q}_2\boldsymbol{R}_2 \\ \boldsymbol{Q}_3\boldsymbol{R}_3 \\ \boldsymbol{Q}_4\boldsymbol{R}_4 \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_1 & & & \\ & \boldsymbol{Q}_2 & & \\ & & \boldsymbol{Q}_3 & \\ & & & \boldsymbol{Q}_4 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \\ \boldsymbol{R}_3 \\ \boldsymbol{R}_4 \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_1 & & & \\ & \boldsymbol{Q}_2 & & \\ & & \boldsymbol{Q}_3 & \\ & & & \boldsymbol{Q}_4 \end{bmatrix} \begin{bmatrix} \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \end{bmatrix} \\ \begin{bmatrix} \boldsymbol{R}_3 \\ \boldsymbol{R}_4 \end{bmatrix} \end{bmatrix}
$$

$$
= \begin{bmatrix} \boldsymbol{Q}_1 & & & \\ & \boldsymbol{Q}_2 & & \\ & & \boldsymbol{Q}_3 & \\ & & & \boldsymbol{Q}_4 \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_{12}\boldsymbol{R}_{12} \\ \boldsymbol{Q}_{34}\boldsymbol{R}_{34} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_1 & & & \\ & \boldsymbol{Q}_2 & & \\ & & \boldsymbol{Q}_3 & \\ & & & \boldsymbol{Q}_4 \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_{12} & \\ & \boldsymbol{Q}_{34} \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_{12} \\ \boldsymbol{R}_{34} \end{bmatrix}
$$

$$
= \begin{bmatrix} \boldsymbol{Q}_1 & & & \\ & \boldsymbol{Q}_2 & & \\ & & \boldsymbol{Q}_3 & \\ & & & \boldsymbol{Q}_4 \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_{12} & \\ & \boldsymbol{Q}_{34} \end{bmatrix} \boldsymbol{Q}_{1234}\boldsymbol{R}_{1234}
$$

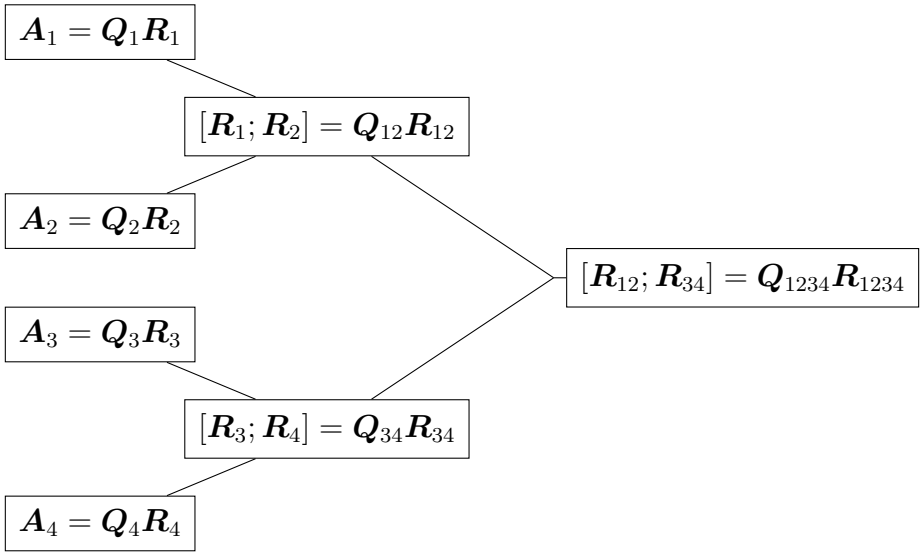


Figure 2.1: The case of 4 compute nodes for the parallel TSQR algorithm. A binary tree is formed which outlines the communication pattern for calculating the $\boldsymbol{R}$ factor. At each level, the node with larger ID sends its $\boldsymbol{R}$ factor to the tree sibling it connects to. The sibling then does a QR factorization with a concatenated pair of child $\boldsymbol{R}$ factors. This process repeats until the final $\boldsymbol{R}$ factor is found at the root of the tree stored on processor $p = 1$. We then broadcast this matrix to all other processors. To get the $\boldsymbol{Q}$ factor, we store the orthogonal $\boldsymbol{Q}_i$ for each tree node i and multiply the parent's $\boldsymbol{Q}_i$ on the right side of the child's $\boldsymbol{Q}$ factor, traversing the tree from the root to the leaves. This results in a distributed memory orthogonal matrix $\boldsymbol{Q} = [\boldsymbol{Q}_1; \boldsymbol{Q}_2; \boldsymbol{Q}_3; \boldsymbol{Q}_4]$.

Algorithm 6: Parallel Right Orthogonalization

Data: A tensor $\boldsymbol{f}$ in TT format with cores $([\boldsymbol{C}_1^p], [\boldsymbol{C}_2^p], \ldots, [\boldsymbol{C}_d^p])$.
Result: A tensor $\boldsymbol{g} = \boldsymbol{f}$ in distributed memory TT format with right orthogonal cores.

Reserve memory for each core block $\boldsymbol{D}_c^p$, using the sizes of $\boldsymbol{C}_c^p$.
$\boldsymbol{D}_d^p = \boldsymbol{C}_d^p$
for $c = d, d-1, \ldots, 2$ **do**
  $[\boldsymbol{L}_c, \boldsymbol{Q}_c^p] = WFLQ(\mathtt{H}(\boldsymbol{D}_c^p))$
  $\boldsymbol{H}_{c-1}^p = \mathtt{V}(\boldsymbol{C}_{c-1}^p)\boldsymbol{L}_c$
  $\boldsymbol{D}_c^p[:] = \boldsymbol{Q}_c^p[:]$
  $\boldsymbol{D}_{c-1}^p[:] = \boldsymbol{H}_{c-1}^p[:]$
end
Set the cores of $\boldsymbol{g}$ as $([\boldsymbol{D}_1^p], [\boldsymbol{D}_2^p], \ldots, [\boldsymbol{D}_d^p])$

a bit of manipulation. In order to parallelize this, we first note the following relationship of the SVD and QR factorizations,

$$\boldsymbol{Q}\boldsymbol{R} = \boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$$

$$\boldsymbol{R} = \boldsymbol{Q}^\top\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top.$$

Thus $\boldsymbol{R}$ has the same singular values as $\boldsymbol{A}$, and so truncating an SVD of $\boldsymbol{R}$ produces the same approximation error as truncating $\boldsymbol{A}$ directly. Since the TSQR algorithm is designed to produce a copy of $\boldsymbol{R}$ on each compute node, we apply a shared memory SVD redundantly on this copied $\boldsymbol{R}$ to determine the truncated tensor. This process is formalized in Algorithms 5 through 8.

These algorithms are designed so that the only communications required are those that are involved in the computation of the QR or LQ factorizations. Each algorithm is mathematically equivalent to its serial counterpart. The core difference is the use of a superscript $p = 1, 2, \ldots, P-1$, to denote which matrices take on different values within the different compute nodes. Matrices which are numerically identical across all nodes lack this superscript.

Algorithm 7: Parallel Left Truncation

Data: A tensor $\boldsymbol{f}$ in distributed memory TT format with cores $([\boldsymbol{C}_1^p], [\boldsymbol{C}_2^p], \ldots, [\boldsymbol{C}_d^p])$ and a desired accuracy ε.

Result: A tensor $\boldsymbol{g}$ so that $\|\boldsymbol{g} - \boldsymbol{f}\| \leq \varepsilon$ in TT format with left orthogonal cores.

Reserve memory for each core block $\boldsymbol{D}_c^p$, using the sizes of $\boldsymbol{C}_c^p$.
Set $([\hat{\boldsymbol{C}}_1^p], [\hat{\boldsymbol{C}}_2^p], \ldots, [\hat{\boldsymbol{C}}_d^p])$ as a right orthogonalization of $\boldsymbol{f}$ by Algorithm 6.
$\hat{\varepsilon} = \varepsilon/\sqrt{d-1}$
$\boldsymbol{D}_1^p = \hat{\boldsymbol{C}}_1^p$
for $c = 1, 2, \ldots, d-1$ **do**
  $[\boldsymbol{Q}_c^p, \boldsymbol{R}_c] = TSQR(\mathtt{V}(\boldsymbol{D}_c^p))$
  $[\boldsymbol{U}_c, \boldsymbol{\Sigma}_c, \boldsymbol{V}_c^\top] = SVD(\boldsymbol{R}_c, \hat{\varepsilon})$ (Truncated SVD with tolerance $\hat{\varepsilon}$.)
  $\boldsymbol{H}_{c+1}^p = \boldsymbol{\Sigma}_c \boldsymbol{V}_c^\top \mathtt{H}(\hat{\boldsymbol{C}}_{c+1}^p)$
  $\hat{\boldsymbol{Q}}_c^p = \boldsymbol{Q}_c^p \boldsymbol{U}_c$
  $\boldsymbol{D}_c^p[:] = \hat{\boldsymbol{Q}}_c^p[:]$
  $\boldsymbol{D}_{c+1}^p[:] = \boldsymbol{H}_{c+1}^p[:]$
end
Set the cores of $\boldsymbol{g}$ as $([\boldsymbol{D}_1]^p, [\boldsymbol{D}_2^p], \ldots, [\boldsymbol{D}_d^p])$

2.3 Fixed Rank Tensor Manifolds

We now discuss the geometric theory of tensor formats. It was proven in [107] that imposing rank constraints on the HT format generates a smooth manifold. An analogous theory for the TT format is presented in [108].

Rather than reproduce those results at length, we follow the case for matrices described in [2]. Define the sets

$$\mathrm{St}(m, r) = \{\boldsymbol{U} \in \mathbb{R}^{m \times r} \mid \det(\boldsymbol{U}^\top \boldsymbol{U}) \neq 0\} \tag{2.14}$$

$$\mathrm{GL}_r(\mathbb{R}) = \{\boldsymbol{A} \in \mathbb{R}^{r \times r} \mid \det(\boldsymbol{A}) \neq 0\} \tag{2.15}$$

which are the noncompact Stiefel manifold and general linear group respectively[4].

[4]These two manifolds are in fact dense open subsets of $\mathbb{R}^{m \times r}$ and $\mathbb{R}^{r \times r}$. Their complements are level sets of polynomials, i.e. a closed condition. Density comes from a straightforward application of SVD.

Algorithm 8: Parallel Right Truncation

Data: A tensor $\boldsymbol{f}$ in distributed memory TT format with cores $([\boldsymbol{C}_1^p], [\boldsymbol{C}_2^p], \ldots, [\boldsymbol{C}_d^p])$ and a desired accuracy ε.

Result: A tensor $\boldsymbol{g}$ so that $\|\boldsymbol{g} - \boldsymbol{f}\| \leq \varepsilon$ in TT format with right orthogonal cores.

Reserve memory for each core block $\boldsymbol{D}_c^p$, using the sizes of $\boldsymbol{C}_c^p$.
Set $([\hat{\boldsymbol{C}}_1^p], [\hat{\boldsymbol{C}}_2^p], \ldots, [\hat{\boldsymbol{C}}_d^p])$ as a right orthogonalization of $\boldsymbol{f}$ by Algorithm 5.
$\hat{\varepsilon} = \varepsilon/\sqrt{d-1}$
$\boldsymbol{D}_d^p = \hat{\boldsymbol{C}}_d^p$
for $c = d, d-1, \ldots, 2$ **do**
  $[\boldsymbol{L}_c, \boldsymbol{Q}_c^p] = WFLQ(\mathtt{H}(\boldsymbol{D}_c^p))$
  $[\boldsymbol{U}_c, \boldsymbol{\Sigma}_c, \boldsymbol{V}_c^\top] = SVD(\boldsymbol{L}_c, \hat{\varepsilon})$ (Truncated SVD with tolerance $\hat{\varepsilon}$.)
  $\boldsymbol{H}_{c-1}^p = \mathtt{V}(\hat{\boldsymbol{C}}_{c-1}^p)\boldsymbol{U}_c\boldsymbol{\Sigma}_c$
  $\hat{\boldsymbol{Q}}_c^p = \boldsymbol{V}_c^\top \boldsymbol{Q}_c^p$
  $\boldsymbol{D}_c^p[:] = \hat{\boldsymbol{Q}}_c^p[:]$
  $\boldsymbol{D}_{c-1}^p[:] = \boldsymbol{H}_{c-1}^p[:]$
end
Set the cores of $\boldsymbol{g}$ as $([\boldsymbol{D}_1^p], [\boldsymbol{D}_2^p], \ldots, [\boldsymbol{D}_d^p])$

Now define the set of fixed rank matrices

$$\mathcal{M}_r = \{\boldsymbol{X} \in \mathbb{R}^{m \times n} \mid \operatorname{rank}(\boldsymbol{X}) = r\} \tag{2.16}$$

Every matrix in this set admits a decomposition $\boldsymbol{X} = \boldsymbol{Q}\boldsymbol{A}\boldsymbol{Z}^\top$ (such as by SVD or Complete Orthogonal Decomposition) where $\boldsymbol{Q} \in \mathrm{St}(m, r)$, $\boldsymbol{A} \in \mathrm{GL}_r(\mathbb{R})$, and $\boldsymbol{Z} \in \mathrm{St}(n, r)$. Due to existence of such a decomposition, we find that there is a surjective map

$$\phi : \mathrm{St}(m, r) \times \mathrm{GL}_r(\mathbb{R}) \times \mathrm{St}(n, r) \to \mathcal{M}_r$$
$$\phi(\boldsymbol{Q}, \boldsymbol{A}, \boldsymbol{Z}) = \boldsymbol{Q}\boldsymbol{A}\boldsymbol{Z}^\top.$$

It can be immediately verified that $\phi(\boldsymbol{Q}, \boldsymbol{A}, \boldsymbol{Z}) = \phi(\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V})$ whenever

$$\exists \boldsymbol{W}, \boldsymbol{E} \in \mathrm{GL}_r(\mathbb{R}), \quad s.t. \quad \boldsymbol{Q} = \boldsymbol{U}\boldsymbol{W}^{-1},\ \boldsymbol{A} = \boldsymbol{W}\boldsymbol{S}\boldsymbol{E}^{\top},\ \boldsymbol{Z} = \boldsymbol{V}\boldsymbol{E}^{-1}. \tag{2.17}$$

I.e. the level sets of ϕ are orbits of a Lie group action by $\mathrm{GL}_r(\mathbb{R}) \times \mathrm{GL}_r(\mathbb{R})$. (Note the similarity to a change of basis formula.) It can be shown[5] that the differential of ϕ is a surjective linear map, and therefore ϕ is a submersion. This leads us to conclude

$$\mathcal{M}_r \simeq (\mathrm{St}(m, r) \times \mathrm{GL}_r(\mathbb{R}) \times \mathrm{St}(n, r))/(\mathrm{GL}_r(\mathbb{R}) \times \mathrm{GL}_r(\mathbb{R})) \tag{2.18}$$

is a manifold embedded in $\mathbb{R}^{m \times n}$. The equivalence classes formed by level sets of ϕ are the points on the fixed rank manifold $\mathcal{M}_r$.

It is also straightforward to find the closure of $\mathcal{M}_r$. This is found by noting that a sequence $\{\boldsymbol{A}_k\}_{k=1}^{\infty}$ of invertible matrices may converge to a rank deficient matrix. Thus, we see that

$$\overline{\mathcal{M}_r} = \mathcal{M}_{\leq r} = \{\boldsymbol{X} \in \mathbb{R}^{m \times n} \mid \mathrm{rank}(\boldsymbol{X}) \leq r\}. \tag{2.19}$$

Note that this closure is not a manifold. Using (2.18), we see that each fixed rank manifold has a different dimension, which depends on r. Thus, there is no way to make an atlas for $\mathcal{M}_{\leq r}$, as it is a finite union of manifolds of differing dimensions.

2.3.1 Tensor Manifold Projections

Denote by $\mathcal{H}_{\boldsymbol{r}} \subseteq \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ the manifold of hierarchical Tucker tensors with multilinear rank $\boldsymbol{r} = \{r_\alpha\}$ corresponding to a given dimension tree $\alpha \in \mathcal{T}_d$ [107].

[5] E.g. by writing a directional derivative by product rule and arguing linear independence along each basis vector.

Remark 1. *Every tensor* $\boldsymbol{f} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ *has an exact hierarchical Tucker (HT) decomposition [45]. Thus, if* $\boldsymbol{f}$ *is not the zero tensor, then* $\boldsymbol{f}$ *belongs to a manifold* $\mathcal{H}_{\boldsymbol{r}}$ *for some* $\boldsymbol{r}$*.*

We introduce three maps which are fundamental to the analysis of low-rank tensor integration. First, we define the nonlinear map

$$\begin{aligned} &\mathfrak{T}_{\boldsymbol{r}}^{\text{best}} : \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d} \to \overline{\mathcal{H}}_{\boldsymbol{r}}, \\ &\mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{f}) = \underset{\boldsymbol{h} \in \overline{\mathcal{H}}_{\boldsymbol{r}}}{\operatorname{argmin}} \ \|\boldsymbol{f} - \boldsymbol{h}\|_2 \,. \end{aligned} \tag{2.20}$$

Here, $\overline{\mathcal{H}}_{\boldsymbol{r}}$ denotes the closure of the tensor manifold $\mathcal{H}_{\boldsymbol{r}}$ and contains all tensors of multilinear rank smaller than or equal to $\boldsymbol{r}$ [6]. The map (2.20) provides the optimal rank-$\boldsymbol{r}$ approximation of a tensor $\boldsymbol{f} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$. The second map, known as high-order singular value decomposition (HOSVD) [46], is defined as a composition of linear maps obtained from a sequence of singular value decompositions of appropriate matricizations of the tensor $\boldsymbol{f}$. Such a map can be written explicitly as a composition of orthogonal projectors dependent on $\boldsymbol{f}$,

$$\begin{aligned} &\mathfrak{T}_{\boldsymbol{r}}^{\text{SVD}} : \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d} \to \overline{\mathcal{H}}_{\boldsymbol{r}}, \\ &\mathfrak{T}_{\boldsymbol{r}}^{\text{SVD}}(\boldsymbol{f}) = \prod_{\alpha \in \mathcal{T}_d^p} \boldsymbol{P}_\alpha \cdots \prod_{\alpha \in \mathcal{T}_d^1} \boldsymbol{P}_\alpha \boldsymbol{f}, \end{aligned} \tag{2.21}$$

where $\mathcal{T}_d^1 \ldots \mathcal{T}_d^p$ are the layers of the dimension tree $\mathcal{T}_d$. The orthogonal projectors $\boldsymbol{P}_\alpha$ are the same ones described in the algorithms of section 2.1. The map (2.21) provides a quasi-optimal[7] rank-$\boldsymbol{r}$ approximation of the tensor $\boldsymbol{f} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$,

[6] The inequality is taken component-wise for the array $\boldsymbol{r}$. See [107].

[7] A quasi-optimal truncation is a function $\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{f})$ satisfying $||\boldsymbol{f} - \mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{f})||_2 \leq C||\boldsymbol{f} - \mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{f})||_2$ for some C independent of $\boldsymbol{f}$.

and is related to the optimal rank-$\boldsymbol{r}$ truncation by the inequalities [45]

$$\left\|\boldsymbol{f} - \mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{f})\right\|_2 \leq \left\|\boldsymbol{f} - \mathfrak{T}_{\boldsymbol{r}}^{\text{SVD}}(\boldsymbol{f})\right\|_2 \leq \sqrt{2d-3}\left\|\boldsymbol{f} - \mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{f})\right\|_2. \tag{2.22}$$

In this work, we will regularly drop the superscript and write $\mathfrak{T}_{\boldsymbol{r}}$ referring to the more computationally practical $\mathfrak{T}_{\boldsymbol{r}}^{\text{SVD}}$. The third map we define is an orthogonal projection onto the tangent space $T_{\boldsymbol{f}}\mathcal{H}_{\boldsymbol{r}}$ of $\mathcal{H}_{\boldsymbol{r}}$ at the point $\boldsymbol{f}$. This projection is defined by the minimization problem

$$\begin{aligned} &\mathcal{P}_{\boldsymbol{f}} : \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d} \to T_{\boldsymbol{f}}\mathcal{H}_{\boldsymbol{r}}, \\ &\mathcal{P}_{\boldsymbol{f}}\boldsymbol{v} = \underset{\boldsymbol{h} \in T_{\boldsymbol{f}}\mathcal{H}_{\boldsymbol{r}}}{\operatorname{argmin}} \ \|\boldsymbol{v} - \boldsymbol{h}\|_2, \end{aligned} \tag{2.23}$$

which is a linear function of $\boldsymbol{v}$ ($\boldsymbol{v}$ is the solution to a linearly constrained least squares problem). An important fact which appears in later analysis is that the tangent plane projector is the derivative of $\mathfrak{T}_{\boldsymbol{r}}^{\text{best}}$. Consider the formal power series expansion of $\mathfrak{T}_{\boldsymbol{r}}^{\text{best}}$ around $\boldsymbol{f} \in \mathcal{H}_{\boldsymbol{r}}$

$$\mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{f} + \varepsilon\boldsymbol{v}) = \mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{f}) + \varepsilon\frac{\partial \mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{f})}{\partial \boldsymbol{f}}\boldsymbol{v} + \cdots, \tag{2.24}$$

where $\partial \mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{f})/\partial \boldsymbol{f}$ denotes the Jacobian of $\mathfrak{T}_{\boldsymbol{r}}^{\text{best}}$ at $\boldsymbol{f}$, $\boldsymbol{v} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, and $\varepsilon \in \mathbb{R}$ is small. Since $\boldsymbol{f} \in \mathcal{H}_{\boldsymbol{r}}$, we have that $\mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{f}) = \boldsymbol{f}$, which allows us to write (2.24) as

$$\mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{f} + \varepsilon\boldsymbol{v}) = \boldsymbol{f} + \varepsilon\frac{\partial \mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{f})}{\partial \boldsymbol{f}}\boldsymbol{v} + \cdots. \tag{2.25}$$

In the following Lemma we summarize that the Jacobian $\partial \mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{f})/\partial \boldsymbol{f}$ coincides with the orthogonal projection (2.23) onto the tangent space $T_{\boldsymbol{f}}\mathcal{H}_{\boldsymbol{r}}$.

Lemma 1 (Smoothness of the best truncation operator)**.** *The map $\mathfrak{T}_{\boldsymbol{r}}^{best}$ is con-*

tinuously differentiable on $\mathcal{H}_{\boldsymbol{r}}$. Moreover,

$$\frac{\partial \mathfrak{T}_{\boldsymbol{r}}^{best}(\boldsymbol{f})}{\partial \boldsymbol{f}} = \mathcal{P}_{\boldsymbol{f}}, \qquad \forall \boldsymbol{f} \in \mathcal{H}_{\boldsymbol{r}},$$

where $\mathcal{P}_{\boldsymbol{f}}$ is the orthogonal projection (2.23) *onto the tangent space of $\mathcal{H}_{\boldsymbol{r}}$ at $\boldsymbol{f}$.*

This result has been proven in [68] and [1] for finite-dimensional manifolds. A slightly different proof which holds for finite-dimensional manifolds without boundary is given in [76]. In Appendix A we provide an alternative proof which is based primarily on linear algebra rather than differential geometry. With Remark 1 in mind, we can apply Lemma 1 to every tensor except the zero tensor.

2.4 SVD Tensor Truncation as an Operator

In the interest of analyzing stability of temporal integrators, we now consider the SVD tensor truncation as an operator. The algorithms we present in Chapter 3 all consist of modifying a known method for solving differential equations by inserting tensor truncations at key steps. By computing the norm of the tensor truncation operator, we can guarantee that our temporal integrators are stable.

We begin our analysis by recalling the definition of semi-norm of a nonlinear function $\boldsymbol{T} : \mathbb{R}^N \to \mathbb{R}^M$. The semi-norm of $\boldsymbol{T}$ is the same as the norm of a linear map, but since $\boldsymbol{T}$ need not be continuous, we replace max with sup.

$$\|\boldsymbol{T}\| = \sup_{\boldsymbol{z} \neq \boldsymbol{0}} \frac{\|\boldsymbol{T}(\boldsymbol{z})\|}{\|\boldsymbol{z}\|}. \tag{2.26}$$

It can be verified that the above definition obeys triangle inequality and absolute

scalability. Moreover, for any $\boldsymbol{w} \neq \mathbf{0}$ we have

$$\frac{\|\boldsymbol{T}(\boldsymbol{w})\|}{\|\boldsymbol{w}\|} \leq \sup_{\boldsymbol{z}\neq\mathbf{0}} \frac{\|\boldsymbol{T}(\boldsymbol{z})\|}{\|\boldsymbol{z}\|}$$

by definition of supremum. Multiplying by $\|\boldsymbol{w}\|$ the denominator yields an inequality that is very similar to Cauchy-Schwartz

$$\|\boldsymbol{T}(\boldsymbol{w})\| \leq \|\boldsymbol{w}\| \sup_{\boldsymbol{z}\neq\mathbf{0}} \frac{\|\boldsymbol{T}(\boldsymbol{z})\|}{\|\boldsymbol{z}\|} = \|\boldsymbol{w}\| \, \|\boldsymbol{T}\| \,. \tag{2.27}$$

Now, suppose $\boldsymbol{T}$ satisfies the scalability property, i.e., $\boldsymbol{T}(\beta\boldsymbol{z}) = \beta\boldsymbol{T}(\boldsymbol{z})$, as with the SVD rank truncation operator. (Refer to definition (2.21). The orthogonal projections for a fixed rank $\boldsymbol{r}$ are independent of the scaling coefficient β.) Then we can pass the norm of $\boldsymbol{z}$ into the numerator.

$$\|\boldsymbol{T}\| = \sup_{\boldsymbol{z}\neq\mathbf{0}} \frac{\|\boldsymbol{T}(\boldsymbol{z})\|}{\|\boldsymbol{z}\|} = \sup_{\boldsymbol{z}\neq\mathbf{0}} \left\| \boldsymbol{T}\left(\frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}\right) \right\| = \sup_{\|\boldsymbol{u}\|=1} \|\boldsymbol{T}\,(\boldsymbol{u})\| \,.$$

In other words, for scalable functions, we can take maximization over the unit sphere in a given norm. Additionally, the norm here is arbitrary. Now we show that the operator semi-norm defines a norm on the vector space of scalable functions. Essentially, we need to show that for scalable $\boldsymbol{T}$, $||\boldsymbol{T}|| = 0$ implies $\boldsymbol{T}$ is zero everywhere. To this end, a proof by contradiction is sufficient. Suppose $\boldsymbol{T}(\boldsymbol{w}) \neq \mathbf{0}$. Then $||\boldsymbol{T}(\boldsymbol{w})|| > 0$. Since for any $\boldsymbol{v}$, we have $\boldsymbol{T}(\mathbf{0}) = \boldsymbol{T}(0\boldsymbol{v}) = 0\boldsymbol{T}(\boldsymbol{v}) = \mathbf{0}$, we must have $\boldsymbol{w} \neq \mathbf{0}$. So the ratio of $||\boldsymbol{T}(\boldsymbol{w})||$ and $||\boldsymbol{w}||$ is positive. This implies that

$$0 < \frac{\|\boldsymbol{T}(\boldsymbol{w})\|}{\|\boldsymbol{w}\|} = \left\| \boldsymbol{T}\left(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right) \right\| \leq \|\boldsymbol{T}\| = 0,$$

i.e., $0 < 0$, a contradiction. Therefore the operator semi-norm (2.26) induces a norm on the vector space of scalable functions. In other words, the operator norm

is well defined for scalable maps.

Remark 2. *Suppose the map* $\boldsymbol{T}$ *is scalable and Lipschitz. Then for any* $\boldsymbol{u} \neq 0$

$$\|\|\boldsymbol{T}(\boldsymbol{u}) - \boldsymbol{T}(\boldsymbol{0})\| \leq C\|\boldsymbol{u} - \boldsymbol{0}\| \quad \Rightarrow \quad \frac{\|\boldsymbol{T}(\boldsymbol{u})\|}{\|\boldsymbol{u}\|} \leq C \quad \Rightarrow \quad \sup_{\|\boldsymbol{u}\|=1} \|\boldsymbol{T}(\boldsymbol{u})\| \leq C$$

since $\boldsymbol{T}(\boldsymbol{0}) = \boldsymbol{0}$ *(*$\boldsymbol{T}$ *is scalable). This means that the Lipschitz constant of* $\boldsymbol{T}$ *is an upper bound for the operator norm.*

Up to this point, the discussion has been developed for arbitrary norms. We now restrict to the 2-norm in particular in order to evaluate the impact of dropping small singular values. This is the norm computed by squaring all entries of a tensor, summing, and then taking square root.

The following result characterizes the truncation operator $\mathfrak{T}_{\boldsymbol{r}}$ as a bounded nonlinear projection.

Lemma 2. *The operator 2-norm of the SVD rank-truncation operator is 1, i.e.,*

$$\sup_{\boldsymbol{x}\neq\boldsymbol{0}} \frac{\left\|\mathfrak{T}_{\boldsymbol{r}}^{SVD}(\boldsymbol{x})\right\|_2}{\|\boldsymbol{x}\|_2} = 1. \tag{2.28}$$

Lemma 2 states that the errors due to low-rank truncation must shrink the tensor being truncated or rotate it to another location equidistant from the origin.

Proof. In this proof, we adopt the shorthand $\mathfrak{T}_{\boldsymbol{r}}^{\mathrm{SVD}} = \mathfrak{T}_{\boldsymbol{r}}$. Recall from 2.3.1 that

$$\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{x}) = \prod_{\alpha\in\mathcal{T}_d^p} \boldsymbol{P}_\alpha \cdots \prod_{\alpha\in\mathcal{T}_d^1} \boldsymbol{P}_\alpha \boldsymbol{x},$$

where every $\boldsymbol{P}_\alpha$ is an orthogonal projection formed using α-mode matricizations of $\boldsymbol{x}$. The particular $\boldsymbol{P}_\alpha$ are dependent on a given $\boldsymbol{x}$. Recall that this implies the

truncation operator is nonlinear, but still scalable. Since all $\boldsymbol{P}_\alpha$ are orthogonal projections, they all have the property

$$||\boldsymbol{P}_\alpha \boldsymbol{v}||_2^2 = \langle \boldsymbol{P}_\alpha \boldsymbol{v}, \boldsymbol{P}_\alpha \boldsymbol{v} \rangle = \langle \boldsymbol{P}_\alpha^\top \boldsymbol{P}_\alpha \boldsymbol{v}, \boldsymbol{v} \rangle = \langle \boldsymbol{P}_\alpha \boldsymbol{P}_\alpha \boldsymbol{v}, \boldsymbol{v} \rangle = \langle \boldsymbol{P}_\alpha \boldsymbol{v}, \boldsymbol{v} \rangle \leq \|\boldsymbol{P}_\alpha \boldsymbol{v}\|_2 \|\boldsymbol{v}\|_2 . \tag{2.29}$$

Dividing by $\|\boldsymbol{P}_\alpha \boldsymbol{v}\|_2$, we have

$$\frac{\|\boldsymbol{P}_\alpha \boldsymbol{v}\|_2}{\|\boldsymbol{v}\|_2} \leq 1, \tag{2.30}$$

for arbitrary $\boldsymbol{v}$. Therefore, the operator norm is at most 1. Now apply this to the composition of operators which defines SVD based truncation.

$$\begin{aligned} \|\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{x})\|_2 = \left\| \prod_{\alpha \in \mathcal{T}_d^p} \boldsymbol{P}_\alpha \cdots \prod_{\alpha \in \mathcal{T}_d^1} \boldsymbol{P}_\alpha \boldsymbol{x} \right\|_2 &\leq \left\| \prod_{\alpha \in \mathcal{T}_d^p} \boldsymbol{P}_\alpha \cdots \prod_{\alpha \in \mathcal{T}_d^1} \boldsymbol{P}_\alpha \right\|_2 \|\boldsymbol{x}\|_2 \\ &\leq \prod_{\alpha \in \mathcal{T}_d^p} \|\boldsymbol{P}_\alpha\|_2 \cdots \prod_{\alpha \in \mathcal{T}_d^1} \|\boldsymbol{P}_\alpha\|_2 \|\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_2 \end{aligned}$$

Dividing by $\|\boldsymbol{x}\|_2$, we get

$$\frac{\|\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{x})\|_2}{\|\boldsymbol{x}\|_2} \leq 1. \tag{2.31}$$

Equality is achieved by noting that $\mathfrak{T}_{\boldsymbol{r}}(\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{x})) = \mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{x})$,

$$\begin{aligned} ||\mathfrak{T}_{\boldsymbol{r}}(\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{x}))||_2 &= ||\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{x})||_2, \\ \frac{||\mathfrak{T}_{\boldsymbol{r}}(\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{x}))||_2}{||\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{x})||_2} &= 1. \end{aligned}$$

□

Chapter 3

Step-Truncation Temporal Integrators

In this chapter, we provide a survey of several step-truncation algorithms. The core idea is to incorporate data compression algorithms into solutions for the ODE

$$\frac{d\boldsymbol{f}(t)}{dt} = \boldsymbol{G}(\boldsymbol{f}(t)), \qquad \boldsymbol{f}(0) = \boldsymbol{f}_0.$$

This is achieved by finding an approximate flow map $\boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}(0)) \approx \boldsymbol{f}(\Delta t)$ which incorporates the truncation operations from Chapter 2. For each algorithm or class of algorithms, we provide the conditions on the rank truncation error required in order to satisfy the step-truncation convergence theorem, i.e. Theorem 3 of Chapter 4. This convergence theorem states that when the approximate local flow map $\boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f})$ is consistent with the exact flow map $\boldsymbol{\varphi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f})$, i.e.

$$\boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}) = \boldsymbol{\varphi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}) + O(\Delta t^{p+1}),$$

and the zero-stability condition

$$\left\|\boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G},\hat{\boldsymbol{f}})-\boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G},\tilde{\boldsymbol{f}})\right\|\leq(1+C\Delta t)\left\|\hat{\boldsymbol{f}}-\tilde{\boldsymbol{f}}\right\|+E\Delta t^{m+1},$$

holds for small Δt, then the method's iterates are convergent to the exact solution of our ODE. We also provide a sufficient condition for zero-stability which allows for a simpler analysis of the explicit methods. For ease of notation, we do not explicitly write the dependence of the tensor ranks $\boldsymbol{r}$ on time step k as we are only analyzing one iteration of each scheme to prove satisfaction of the sufficient consistency and stability conditions. It should be noted, however, that all ranks are in fact time dependent, and may be alternatively denoted as $\boldsymbol{r}=\boldsymbol{r}_k$.

Zero Stability for Explicit Step-Truncation Methods

Our explicit methods all take the form

$$\boldsymbol{f}_{k+1}=\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{f}_k+\Delta t\mathcal{A}_{\Delta t}(\boldsymbol{G},\boldsymbol{f}_k)),\tag{3.1}$$

so that the local flow approximation is $\boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G},\boldsymbol{f})=\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{f}+\Delta t\mathcal{A}_{\Delta t}(\boldsymbol{G},\boldsymbol{f}))$. In this case, zero-stability is determined by the increment function $\mathcal{A}_{\Delta t}(\boldsymbol{G},\boldsymbol{f})$. We summarize this in the following Lemma.

Lemma 3. *If for all $\boldsymbol{f}$ iteration* (3.1) *satisfies*

$$\|\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{f}+\Delta t\mathcal{A}_{\Delta t}(\boldsymbol{G},\boldsymbol{f}))-(\boldsymbol{f}+\Delta t\mathcal{A}_{\Delta t}(\boldsymbol{G},\boldsymbol{f}))\|\leq E\Delta t^{m+1},$$

and $\mathcal{A}_{\Delta t}(\boldsymbol{G},\boldsymbol{f})$ satisfies the near-Lipschitz criterion as $\Delta t\to 0$

$$\left\|\mathcal{A}_{\Delta t}(\boldsymbol{G},\hat{\boldsymbol{f}})-\mathcal{A}_{\Delta t}(\boldsymbol{G},\tilde{\boldsymbol{f}})\right\|\leq(C+\Delta tD)\left\|\hat{\boldsymbol{f}}-\tilde{\boldsymbol{f}}\right\|+E\Delta t^{m},\tag{3.2}$$

then (3.1) *is zero-stable. Specifically,*

$$\begin{aligned}
&\left\|\boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G},\hat{\boldsymbol{f}}) - \boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G},\tilde{\boldsymbol{f}})\right\| \\
&\qquad = \left\|\mathfrak{T}_{\boldsymbol{r}}(\hat{\boldsymbol{f}} + \Delta t\mathcal{A}_{\Delta t}(\boldsymbol{G},\hat{\boldsymbol{f}})) - \mathfrak{T}_{\boldsymbol{r}}(\tilde{\boldsymbol{f}} + \Delta t\mathcal{A}_{\Delta t}(\boldsymbol{G},\tilde{\boldsymbol{f}}))\right\| \\
&\qquad \leq \left\|\hat{\boldsymbol{f}} + \Delta t\mathcal{A}_{\Delta t}(\boldsymbol{G},\hat{\boldsymbol{f}}) - (\tilde{\boldsymbol{f}} + \Delta t\mathcal{A}_{\Delta t}(\boldsymbol{G},\tilde{\boldsymbol{f}}))\right\| + 2E\Delta t^{m+1} \\
&\qquad \leq \left\|\hat{\boldsymbol{f}} - \tilde{\boldsymbol{f}}\right\| + \Delta t\left\|\mathcal{A}_{\Delta t}(\boldsymbol{G},\hat{\boldsymbol{f}}) - \mathcal{A}_{\Delta t}(\boldsymbol{G},\tilde{\boldsymbol{f}})\right\| + 2E\Delta t^{m+1} \\
&\qquad \leq \left\|\hat{\boldsymbol{f}} - \tilde{\boldsymbol{f}}\right\| + (C\Delta t + \Delta t^2 D)\left\|\hat{\boldsymbol{f}} - \tilde{\boldsymbol{f}}\right\| + 3E\Delta t^{m+1} \\
&\qquad \leq (1 + \max(C,D)2\Delta t)\left\|\hat{\boldsymbol{f}} - \tilde{\boldsymbol{f}}\right\| + 3E\Delta t^{m+1}.
\end{aligned}$$

Inequality (3.2) is an easier condition to check. Lemma 3 will save us the effort of repeating the inequalities above for each explicit method independently.

3.1 The Explicit Euler Method

Our first example is a first order method for solving (1.2) based on Euler forward. From Theorem 3, we know that the scheme

$$\boldsymbol{f}_{k+1} = \mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{f}_k + \Delta t\boldsymbol{G}(\boldsymbol{f}_k)) \tag{3.3}$$

is order one in Δt, provided the vector field $\boldsymbol{G}$ is Lipschitz and the truncation rank $\boldsymbol{r}$ at time step k satisfies

$$\|\boldsymbol{f}_k + \Delta t\boldsymbol{G}(\boldsymbol{f}_k) - \mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{f}_k + \Delta t\boldsymbol{G}(\boldsymbol{f}_k))\|_2 \leq M\Delta t^2,$$

for all $k = 1, 2, \ldots$. Applying the nonlinear vector field $\boldsymbol{G}$ to the solution tensor $\boldsymbol{f}$ can result in a tensor with large rank. Therefore, it may be desirable to apply a tensor truncation operator to $\boldsymbol{G}(\boldsymbol{f})$ at each time step. To implement this, we

build the additional truncation operator into the increment function

$$\mathcal{A}(\boldsymbol{G}, \boldsymbol{f}_k, \boldsymbol{s}, \Delta t) = \mathfrak{T}_{\boldsymbol{s}}(\boldsymbol{G}(\boldsymbol{f}_k)), \tag{3.4}$$

to obtain the new scheme

$$\boldsymbol{f}_{k+1} = \mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{f}_k + \Delta t \mathfrak{T}_{\boldsymbol{s}}(\boldsymbol{G}(\boldsymbol{f}_k))). \tag{3.5}$$

We now determine conditions for the time dependent ranks $\boldsymbol{s}$ and $\boldsymbol{r}$ which make the scheme (3.5) first order. To address consistency, suppose $\boldsymbol{f}(t)$ is the exact solution to (1.2) with initial condition $\boldsymbol{f}_0$ at time $t = 0$. Then, bound the local truncation error as

$$\begin{aligned}
||\boldsymbol{f}(\Delta t) - \boldsymbol{f}_1||_2 &\leq ||\boldsymbol{f}(\Delta t) - (\boldsymbol{f}_0 + \Delta t \boldsymbol{G}(\boldsymbol{f}_0))||_2 \\
&\quad + ||\boldsymbol{f}_0 + \Delta t \boldsymbol{G}(\boldsymbol{f}_0) - \mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{f}_0 + \Delta t \mathfrak{T}_{\boldsymbol{s}}(\boldsymbol{G}(\boldsymbol{f}_0)))||_2 \\
&\leq K\Delta t^2 + ||\boldsymbol{f}_0 + \Delta t \boldsymbol{G}(\boldsymbol{f}_0) - (\boldsymbol{f}_0 + \Delta t \mathfrak{T}_{\boldsymbol{s}}(\boldsymbol{G}(\boldsymbol{f}_0)))||_2 \\
&\quad + ||\boldsymbol{f}_0 + \Delta t \mathfrak{T}_{\boldsymbol{s}}(\boldsymbol{G}(\boldsymbol{f}_0)) - \mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{f}_0 + \Delta t \mathfrak{T}_{\boldsymbol{s}}(\boldsymbol{G}(\boldsymbol{f}_0)))||_2 \\
&= K\Delta t^2 + \Delta t \, ||\boldsymbol{G}(\boldsymbol{f}_0) - \mathfrak{T}_{\boldsymbol{s}}(\boldsymbol{G}(\boldsymbol{f}_0))||_2 \\
&\quad + ||\boldsymbol{f}_0 + \Delta t \mathfrak{T}_{\boldsymbol{s}}(\boldsymbol{G}(\boldsymbol{f}_0)) - \mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{f}_0 + \Delta t \mathfrak{T}_{\boldsymbol{s}}(\boldsymbol{G}(\boldsymbol{f}_0)))||_2 \,.
\end{aligned} \tag{3.6}$$

From this bound, we see that by selecting $\boldsymbol{s}$ and $\boldsymbol{r}$ at time $t_k = k\Delta t$ so that

$$||\boldsymbol{G}(\boldsymbol{f}_k) - \mathfrak{T}_{\boldsymbol{s}}(\boldsymbol{G}(\boldsymbol{f}_k))||_2 \leq M_1 \Delta t, \tag{3.7}$$

$$||\boldsymbol{f}_k + \Delta t \mathfrak{T}_{\boldsymbol{s}}(\boldsymbol{G}(\boldsymbol{f}_k)) - \mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{f}_k + \Delta t \mathfrak{T}_{\boldsymbol{s}}(\boldsymbol{G}(\boldsymbol{f}_k)))||_2 \leq M_2 \Delta t^2, \tag{3.8}$$

for all $k = 1, 2, \ldots$, yields an order one local truncation error for (3.5). To address stability, we show that (3.4) satisfies (3.2) with $m = 1$, assuming $\boldsymbol{G}$ is Lipschitz.

Indeed,

$$\begin{aligned}
&\left\|\mathfrak{T}_s(\boldsymbol{G}(\hat{\boldsymbol{f}})) - \mathfrak{T}_s(\boldsymbol{G}(\tilde{\boldsymbol{f}}))\right\|_2 \\
&\qquad \leq \left\|\mathfrak{T}_s(\boldsymbol{G}(\hat{\boldsymbol{f}})) - \boldsymbol{G}(\hat{\boldsymbol{f}})\right\|_2 + \left\|\boldsymbol{G}(\hat{\boldsymbol{f}}) - \boldsymbol{G}(\tilde{\boldsymbol{f}})\right\|_2 + \left\|\boldsymbol{G}(\tilde{\boldsymbol{f}}) - \mathfrak{T}_s(\boldsymbol{G}(\tilde{\boldsymbol{f}}))\right\|_2 \\
&\qquad \leq L\left\|\hat{\boldsymbol{f}} - \tilde{\boldsymbol{f}}\right\|_2 + 2M_1\Delta t,
\end{aligned}$$

where L is the Lipschitz constant of $\boldsymbol{G}$. Now applying Theorem 3 proves that the rank-adaptive Euler method (3.5) has $O(\Delta t)$ global error.

3.2 The Explicit Midpoint Method

Consider the following rank-adaptive step-truncation method based on the explicit midpoint rule (see [50, II.1])

$$\boldsymbol{f}_{k+1} = \mathfrak{T}_{\boldsymbol{\alpha}}\left(\boldsymbol{f}_k + \Delta t\left(\boldsymbol{G}\left(\boldsymbol{f}_k + \frac{\Delta t}{2}\boldsymbol{G}(\boldsymbol{f}_k)\right)\right)\right). \tag{3.9}$$

We have proven in Theorem 3 that (3.9) is order 2 in Δt, provided the vector field $\boldsymbol{G}$ is Lipschitz and the truncation rank $\boldsymbol{\alpha}$ at each discrete time t_k satisfies

$$\|\boldsymbol{a}_k - \mathfrak{T}_{\boldsymbol{\alpha}}(\boldsymbol{a}_k)\|_2 \leq M\Delta t^3,$$

for all $k = 1, 2, \ldots$. Here,

$$\boldsymbol{a}_k = \boldsymbol{f}_k + \Delta t\left(\boldsymbol{G}\left(\boldsymbol{f}_k + \frac{\Delta t}{2}\boldsymbol{G}(\boldsymbol{f}_k)\right)\right)$$

denotes the solution tensor at time t_{k+1} prior to truncation. For the same reasons we discussed in Section 3.1, it may be desirable to insert truncation operators inside the increment function. For our rank-adaptive explicit midpoint method

we consider the increment function

$$\mathcal{A}(\boldsymbol{G}, \boldsymbol{f}_k, \boldsymbol{\beta}, \boldsymbol{\gamma}, \Delta t) = \mathfrak{T}_{\boldsymbol{\beta}} \left(\boldsymbol{G} \left(\boldsymbol{f}_k + \frac{\Delta t}{2} \mathfrak{T}_{\boldsymbol{\gamma}}(\boldsymbol{G}(\boldsymbol{f}_k)) \right) \right), \tag{3.10}$$

which results in the step-truncation scheme

$$\boldsymbol{f}_{k+1} = \mathfrak{T}_{\boldsymbol{\alpha}} \left(\boldsymbol{f}_k + \Delta t \mathfrak{T}_{\boldsymbol{\beta}} \left(\boldsymbol{G} \left(\boldsymbol{f}_k + \frac{\Delta t}{2} \mathfrak{T}_{\boldsymbol{\gamma}}(\boldsymbol{G}(\boldsymbol{f}_k)) \right) \right) \right). \tag{3.11}$$

Following a similar approach as in Section 3.1, we aim to find conditions on the time dependent ranks $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ so that the local truncation error of the scheme (3.11) is order 2. For ease of notation, let us denote the truncation errors by $\varepsilon_{\boldsymbol{\kappa}} = \|\boldsymbol{g} - \mathfrak{T}_{\boldsymbol{\kappa}}(\boldsymbol{g})\|_2$, where $\boldsymbol{\kappa} = \boldsymbol{\alpha}$, $\boldsymbol{\beta}$, or $\boldsymbol{\gamma}$. To analyze consistency, again let $\boldsymbol{f}(t)$ be the exact solution to (1.2) with initial condition $\boldsymbol{f}_0$ at time $t = 0$. The local truncation error of the scheme (3.11) can be estimated as

$$\begin{aligned}
&||\boldsymbol{f}(\Delta t) - \boldsymbol{f}_1||_2 \\
&\qquad \leq \varepsilon_{\boldsymbol{\alpha}} + \left\| \boldsymbol{f}(\Delta t) - \left(\boldsymbol{f}_0 + \Delta t \mathfrak{T}_{\boldsymbol{\beta}} \left(\boldsymbol{G} \left(\boldsymbol{f}_0 + \frac{\Delta t}{2} \mathfrak{T}_{\boldsymbol{\gamma}}(\boldsymbol{G}(\boldsymbol{f}_0)) \right) \right) \right) \right\|_2 \\
&\qquad \leq \varepsilon_{\boldsymbol{\alpha}} + \Delta t \varepsilon_{\boldsymbol{\beta}} + \left\| \boldsymbol{f}(\Delta t) - \left(\boldsymbol{f}_0 + \Delta t \boldsymbol{G} \left(\boldsymbol{f}_0 + \frac{\Delta t}{2} \mathfrak{T}_{\boldsymbol{\gamma}}(\boldsymbol{G}(\boldsymbol{f}_0)) \right) \right) \right\|_2 \\
&\qquad \leq K \Delta t^3 + \varepsilon_{\boldsymbol{\alpha}} + \Delta t \varepsilon_{\boldsymbol{\beta}} + \frac{L \Delta t^2}{2} \varepsilon_{\boldsymbol{\gamma}},
\end{aligned}$$

where L is the Lipschitz constant of $\boldsymbol{G}$. From this bound, we see that if the truncation ranks $\boldsymbol{\alpha}, \boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are chosen such that

$$\varepsilon_{\boldsymbol{\alpha}} \leq A \Delta t^3, \qquad \varepsilon_{\boldsymbol{\beta}} \leq B \Delta t^2, \qquad \varepsilon_{\boldsymbol{\gamma}} \leq G \Delta t, \tag{3.12}$$

for some constants A, B, and G, then the local truncation error of the scheme (3.11) is order 2 in Δt. Also, if (3.12) is satisfied then the zero-stability require-

ment (3.2) is also satisfied. Indeed,

$$\left\|\mathfrak{T}_\beta\left(\boldsymbol{G}\left(\hat{\boldsymbol{f}}+\frac{\Delta t}{2}\mathfrak{T}_\gamma(\boldsymbol{G}(\hat{\boldsymbol{f}}))\right)\right)-\mathfrak{T}_\beta\left(\boldsymbol{G}\left(\tilde{\boldsymbol{f}}+\frac{\Delta t}{2}\mathfrak{T}_\gamma(\boldsymbol{G}(\tilde{\boldsymbol{f}}))\right)\right)\right\|_2 \\ \leq\left(L+\frac{L}{2}\Delta t\right)\left\|\hat{\boldsymbol{f}}-\tilde{\boldsymbol{f}}\right\|_2+2\varepsilon_\beta+L\Delta t\varepsilon_\gamma \tag{3.13}$$

holds for all tensors $\hat{\boldsymbol{f}},\tilde{\boldsymbol{f}}\in\mathbb{R}^{n_1\times\cdots\times n_d}$. To arrive at the above relationship, we applied triangle inequality several times to pull out the $\varepsilon_{\boldsymbol{\kappa}}$ terms and then used Lipschitz continuity of $\boldsymbol{G}$ multiple times. Thus, if the truncation ranks $\boldsymbol{\alpha},\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are chosen to satisfy (3.12) and the vector field $\boldsymbol{G}$ is Lipschitz, then Theorem 3 proves the method (3.11) has $O(\Delta t^2)$ global error.

3.3 Explicit Linear Multistep Methods

With some effort we can extend the rank-adaptive global error estimates to the well-known multi-step methods of Adams and Bashforth (see [50, III.1]). These methods are of the form

$$\boldsymbol{f}_{k+1}=\boldsymbol{f}_k+\Delta t\sum_{j=0}^{s-1}b_j\boldsymbol{G}(\boldsymbol{f}_{k-j}), \tag{3.14}$$

where s is the number of steps. A rank-adaptive step-truncation version of this method is

$$\boldsymbol{f}_{k+1}=\mathfrak{T}_{\boldsymbol{\alpha}}\left(\boldsymbol{f}_k+\Delta t\mathfrak{T}_{\boldsymbol{\beta}}\left(\sum_{j=0}^{s-1}b_j\mathfrak{T}_{\boldsymbol{\gamma}(j)}\left(\boldsymbol{G}(\boldsymbol{f}_{k-j})\right)\right)\right). \tag{3.15}$$

In order to obtain a global error estimate for (3.15), we follow the same steps as before. First we prove consistency, then we prove stability, and finally combine these results to obtain a global convergence result. For consistency, let $\boldsymbol{f}_0=\boldsymbol{f}(t_0)$, $\boldsymbol{f}_1=\boldsymbol{f}(t_1),\ldots,\boldsymbol{f}_{s-1}=\boldsymbol{f}(t_{s-1})$ be the exact solution to (1.2) given at the first s

time steps. Also, define the truncation errors $\varepsilon_{\boldsymbol{\kappa}} = \|\boldsymbol{g} - \mathfrak{T}_{\boldsymbol{\kappa}}(\boldsymbol{g})\|_2$, where $\boldsymbol{\kappa} = \boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}(j)$, $j = 0, \ldots, s-1$. The local error admits the bound

$$
\begin{aligned}
&||\boldsymbol{f}(t_s) - \boldsymbol{f}_s||_2 \\
&\quad \leq \varepsilon_{\boldsymbol{\alpha}} + \left\| \boldsymbol{f}(t_s) - \left(\boldsymbol{f}_{s-1} + \Delta t \mathfrak{T}_{\boldsymbol{\beta}} \left(\sum_{j=0}^{s-1} b_j \mathfrak{T}_{\boldsymbol{\gamma}(j)} \left(\boldsymbol{G}(\boldsymbol{f}_{s-1-j}) \right) \right) \right) \right\|_2 \\
&\quad \leq \varepsilon_{\boldsymbol{\alpha}} + \Delta t \varepsilon_{\boldsymbol{\beta}} + \left\| \boldsymbol{f}(t_s) - \left(\boldsymbol{f}_{s-1} + \Delta t \sum_{j=0}^{s-1} b_j \mathfrak{T}_{\boldsymbol{\gamma}(j)} \left(\boldsymbol{G}(\boldsymbol{f}_{s-1-j}) \right) \right) \right\|_2 \\
&\quad \leq \varepsilon_{\boldsymbol{\alpha}} + \Delta t \varepsilon_{\boldsymbol{\beta}} + \Delta t \sum_{j=0}^{s-1} |b_j| \varepsilon_{\boldsymbol{\gamma}(j)} + \left\| \boldsymbol{f}(t_s) - \left(\boldsymbol{f}_{s-1} + \Delta t \sum_{j=0}^{s-1} b_j \boldsymbol{G}(\boldsymbol{f}_{s-1-j}) \right) \right\|_2 .
\end{aligned}
$$

The last term is the local error for an order-s Adams-Bashforth method (3.14). Therefore, the local error of the step-truncation method (3.15) is also of order s if the truncation ranks $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}(j)$ are chosen such that

$$
\varepsilon_{\boldsymbol{\alpha}} \leq A \Delta t^{s+1}, \qquad \varepsilon_{\boldsymbol{\beta}} \leq B \Delta t^{s}, \qquad \varepsilon_{\boldsymbol{\gamma}(j)} \leq G_j \Delta t^{s}. \tag{3.16}
$$

To address stability, we first need to generalize the zero-stability condition (3.2) to the increment function

$$
\mathcal{A}(\boldsymbol{G}, \boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_s, \Delta t) = \mathfrak{T}_{\boldsymbol{\beta}_k} \left(\sum_{j=0}^{s-1} b_j \mathfrak{T}_{\boldsymbol{\gamma}_k(j)} \left(\boldsymbol{G}(\boldsymbol{f}_{k-j}) \right) \right) \tag{3.17}
$$

for the multi-step method (3.14). A natural choice is

$$
\begin{aligned}
\left\| \mathcal{A}(\boldsymbol{G}, \hat{\boldsymbol{f}}_1, \hat{\boldsymbol{f}}_2, \ldots, \hat{\boldsymbol{f}}_s, \Delta t) - \mathcal{A}(\boldsymbol{G}, \tilde{\boldsymbol{f}}_1, \tilde{\boldsymbol{f}}_2, \ldots, \tilde{\boldsymbol{f}}_s, \Delta t) \right\|_2 \leq \\
\sum_{j=1}^{s} C_j \left\| \hat{\boldsymbol{f}}_j - \tilde{\boldsymbol{f}}_j \right\|_2 + E \Delta t^m .
\end{aligned} \tag{3.18}
$$

Clearly, for $s = 1$ the criterion (3.18) specializes to the stability criterion given in (3.2). We have the bound

$$\begin{aligned}
&\left|\left|\mathcal{A}(\boldsymbol{G}, \hat{\boldsymbol{f}}_1, \hat{\boldsymbol{f}}_2, \ldots, \hat{\boldsymbol{f}}_s, \Delta t) - \mathcal{A}(\boldsymbol{G}, \tilde{\boldsymbol{f}}_1, \tilde{\boldsymbol{f}}_2, \ldots, \tilde{\boldsymbol{f}}_s, \Delta t)\right|\right|_2 \\
&\qquad \leq \left|\left| \mathfrak{T}_{\boldsymbol{\beta}} \left(\sum_{j=0}^{s-1} b_j \mathfrak{T}_{\boldsymbol{\gamma}(j)} \left(\boldsymbol{G}(\hat{\boldsymbol{f}}_{s-j}) \right) \right) - \mathfrak{T}_{\boldsymbol{\beta}} \left(\sum_{j=0}^{s-1} b_j \mathfrak{T}_{\boldsymbol{\gamma}(j)} \left(\boldsymbol{G}(\tilde{\boldsymbol{f}}_{s-j}) \right) \right) \right|\right|_2 \\
&\qquad \leq 2\varepsilon_{\boldsymbol{\beta}} + 2 \sum_{j=0}^{s-1} |b_j| \varepsilon_{\boldsymbol{\gamma}(j)} + \sum_{j=0}^{s-1} |b_j| \left|\left| \boldsymbol{G}(\hat{\boldsymbol{f}}_{s-j}) - \boldsymbol{G}(\tilde{\boldsymbol{f}}_{s-j}) \right|\right|_2 \\
&\qquad \leq 2\varepsilon_{\boldsymbol{\beta}} + 2 \sum_{j=0}^{s-1} |b_j| \varepsilon_{\boldsymbol{\gamma}(j)} + \sum_{j=0}^{s-1} L|b_j| \left|\left| \hat{\boldsymbol{f}}_{s-j} - \tilde{\boldsymbol{f}}_{s-j} \right|\right|_2,
\end{aligned}$$

where we used triangle inequality to set aside the $\varepsilon_{\boldsymbol{\kappa}}$ terms and subsequently applied Lipschitz continuity multiple times. From the above inequality, it is seen that if (3.16) is satisfied, then the stability condition (3.18) is also satisfied with $m = s$. With the consistency and stability results for the multistep step-truncation method (3.15) just obtained, it is straightforward to obtain the following global error estimate for (3.15).

Corollary 1 (Global error of rank-adaptive Adams-Bashforth scheme)**.** *Assume* $\boldsymbol{f}_0 = \boldsymbol{f}(t_0)$, $\boldsymbol{f}_1 = \boldsymbol{f}(t_1)$, $\ldots$, $\boldsymbol{f}_{s-1} = \boldsymbol{f}(t_{s-1})$ *are given initial steps for a convergent order-s method of the form* (3.14)*, and assume* $\boldsymbol{G}$ *is Lipschitz with constant* L*. If the rank-adaptive step-trunctation method* (3.15) *is order-s consistent with* (1.2)*, and the corresponding increment function* $\mathcal{A}$ *defined in* (3.17) *satisfies the stability condition* (3.18)*, then the global error satisfies*

$$||\boldsymbol{f}(T) - \boldsymbol{f}_N||_2 \leq Q\Delta t^s,$$

where Q *depends only on the local error constants of the Adams-Bashforth scheme* (3.14)*.*

Proof. The proof is based on an inductive argument on the number of steps taken (N in equation (4.1)), similar to the proof of Theorem 3. First, notice that by assuming the method (3.15) is order-s consistent, we immediately obtain (3.14) for the base case $N = s$. Now, suppose that

$$||\boldsymbol{f}(t_{N-k}) - \boldsymbol{f}_{N-k}||_2 \leq Q_{N-k}\Delta t^s \tag{3.19}$$

for all k, $1 \leq k \leq N-s$. Letting $\boldsymbol{a}_k = \boldsymbol{f}_k + \Delta t\mathcal{A}(\boldsymbol{G}, \boldsymbol{f}_k, \ldots, \boldsymbol{f}_{k-s+1}, \Delta t)$ denote one step prior to truncation, we expand the final step error in terms of penultimate step

$$\begin{aligned}
&||\boldsymbol{f}(T) - \boldsymbol{f}_N||_2 \\
&\quad = \left|\left|\boldsymbol{f}(t_N) - \mathfrak{T}_{\boldsymbol{r}_{N-1}}(\boldsymbol{f}_{N-1} + \Delta t\mathcal{A}(\boldsymbol{G}, \boldsymbol{f}_{N-1}, \ldots, \boldsymbol{f}_{N-s}, \Delta t))\right|\right|_2 \\
&\quad \leq ||\boldsymbol{f}(t_N) - \boldsymbol{a}_{N-1}||_2 + \\
&\qquad \left|\left|\boldsymbol{a}_{N-1} - \mathfrak{T}_{\boldsymbol{r}_{N-1}}(\boldsymbol{f}_{N-1} + \Delta t\mathcal{A}(\boldsymbol{G}, \boldsymbol{f}_{N-1}, \ldots, \boldsymbol{f}_{N-s}, \Delta t))\right|\right|_2 \\
&\quad = ||\boldsymbol{f}(t_N) - \boldsymbol{a}_{N-1}||_2 + \left|\left|\boldsymbol{a}_{N-1} - \mathfrak{T}_{\boldsymbol{r}_{N-1}}(\boldsymbol{a}_{N-1})\right|\right|_2 \\
&\quad \leq ||\boldsymbol{f}(t_N) - \boldsymbol{a}_{N-1}||_2 + M\Delta t^{s+1} \\
&\quad \leq ||\boldsymbol{f}(t_N) - (\boldsymbol{f}(t_{N-1}) + \Delta t\mathcal{A}(\boldsymbol{G}, \boldsymbol{f}(t_{N-1}), \ldots, \boldsymbol{f}(t_{N-s}), \Delta t))||_2 \\
&\qquad + ||\boldsymbol{f}(t_{N-1}) + \Delta t\mathcal{A}(\boldsymbol{G}, \boldsymbol{f}(t_{N-1}), \ldots, \boldsymbol{f}(t_{N-s}), \Delta t) - \boldsymbol{a}_{N-1}||_2 + M\Delta t^{p+1} \\
&\quad \leq ||\boldsymbol{f}(t_{N-1}) + \Delta t\mathcal{A}(\boldsymbol{G}, \boldsymbol{f}(t_{N-1}), \ldots, \boldsymbol{f}(t_{N-s}), \Delta t) - \boldsymbol{a}_{N-1}||_2 \\
&\qquad + K_{N-1}\Delta t^{s+1} + M\Delta t^{s+1},
\end{aligned}$$

where we explicitly denote $\boldsymbol{r}_{N-1}$ as the rank at time t_{N-1} and K_{N-1} as a local error constant for the untruncated scheme (3.14). Expanding $\boldsymbol{a}_{N-1}$ and using the

triangle inequality we find

$$\begin{aligned}||\boldsymbol{f}(T) - \boldsymbol{f}_N||_2 \leq\ & ||\boldsymbol{f}(t_{N-1}) - \boldsymbol{f}_{N-1}||_2 \\ & + \Delta t\, ||\mathcal{A}(\boldsymbol{G}, \boldsymbol{f}(t_{N-1}), \ldots, \boldsymbol{f}(t_{N-s}), \Delta t) - \mathcal{A}(\boldsymbol{G}, \boldsymbol{f}_{N-1}, \ldots, \boldsymbol{f}_{N-s}, \Delta t)||_2 \\ & + K_{N-1}\Delta t^{s+1} + A\Delta t^{s+1}.\end{aligned}$$

Applying the stability condition (3.18) yields

$$\begin{aligned}||\boldsymbol{f}(T) - \boldsymbol{f}_N||_2 \leq ||\boldsymbol{f}(t_{N-1}) - \boldsymbol{f}_{N-1}||_2 + \Delta t \sum_{j=1}^{s} C_j\, ||\boldsymbol{f}(t_{N-j}) - \boldsymbol{f}_{N-j}||_2 \\ + (K_{N-1} + A + E)\Delta t^{s+1}.\end{aligned}$$

The above inequality together with the inductive hypothesis (3.19) implies

$$||\boldsymbol{f}(T) - \boldsymbol{f}_N||_2 \leq Q_{N-1}\Delta t^s + \Delta t^{s+1} \sum_{j=1}^{s} C_j Q_{N-j} + (K_{N-1} + A + E)\Delta t^{s+1}, \tag{3.20}$$

concluding the proof. □

Similar to Theorem 3, only the constants K_j appearing in (3.20) depend on the time step index (note that Q_i also depends on K_j). Moreover, the constants K_j depend only on the multi-step method (3.14), and not on truncation.

3.4 The Implicit Euler Method

Consider the implicit Euler scheme (4.31), with the associated rootfinding problem

$$\boldsymbol{H}_k(\boldsymbol{f}_{k+1}) = \boldsymbol{f}_{k+1} - \boldsymbol{f}_k - \Delta t \boldsymbol{G}(\boldsymbol{f}_{k+1}) = \boldsymbol{0}$$

and suppose that the solution of the nonlinear equation $\boldsymbol{H}_k(\boldsymbol{f}_{k+1}) = \boldsymbol{0}$ is computed using the inexact Newton method with HT/TT-GMRES iterations as discussed in Section 4.4.1. Let $\boldsymbol{f}_{k+1}^{[j]}$ $(j = 1, 2, \ldots)$ be the sequence of tensors generated by the algorithm and approximating $\boldsymbol{f}_{k+1}$ (exact solution of $\boldsymbol{H}_k(\boldsymbol{f}_{k+1}) = \boldsymbol{0}$). We set a stopping criterion for terminating Newton's iterations based on the residual, i.e.,

$$\left\| \boldsymbol{H}_k \left(\boldsymbol{f}_{k+1}^{[j]} \right) \right\| \leq \varepsilon_{\text{tol}}. \tag{3.21}$$

This allows us to adjust the rank of $\boldsymbol{f}_{k+1}^{[j]}$ from one time step to the next, depending on the desired accuracy ε_{tol}. Our goal is to analyze convergence of such a rank-adaptive implicit method when a finite number of Newton steps are taken. To this end, we will fit the method into the framework of Theorem 3. We begin by noticing that

$$\begin{aligned} \left\| \boldsymbol{f}_{k+1} - \boldsymbol{f}_{k+1}^{[j]} \right\| &= \left\| \boldsymbol{H}_k^{-1} \left(\boldsymbol{H}_k \left(\boldsymbol{f}_{k+1} \right) \right) - \boldsymbol{H}_k^{-1} \left(\boldsymbol{H}_k \left(\boldsymbol{f}_{k+1}^{[j]} \right) \right) \right\| && (3.22) \\ &\leq L_{\boldsymbol{H}_k^{-1}} \left\| \boldsymbol{H} \left(\boldsymbol{f}_{k+1}^{[j]} \right) \right\| \leq L_{\boldsymbol{H}_k^{-1}} \varepsilon_{\text{tol}}, && (3.23) \end{aligned}$$

where $L_{\boldsymbol{H}_k^{-1}}$ is the local Lipschitz constant of the smooth inverse map $\boldsymbol{H}_k^{-1}$, the existence of which is granted by the inverse function theorem. This allows us to write the local truncation error as

$$\left\| \boldsymbol{f}(\Delta t) - \boldsymbol{f}_1^{[j]} \right\| \leq \| \boldsymbol{f}(\Delta t) - \boldsymbol{f}_1 \| + \left\| \boldsymbol{f}_1 - \boldsymbol{f}_1^{[j]} \right\| \leq K_1 \Delta t^2 + L_{\boldsymbol{H}^{-1}} \varepsilon_{\text{tol}}, \tag{3.24}$$

where K_1 is a local error coefficient. In order to maintain order one convergence, we require that $\varepsilon_{\text{tol}} \leq K \Delta t^2$ for some constant $K \geq 0$.

Next, we discuss the stability condition (4.22) in the context of the implicit step-truncation Euler scheme, assuming that $\boldsymbol{f}_{k+1}$ can be found exactly by the in-

exact Newton's method, eventually after an infinite number of iterations. Denote by $\hat{\boldsymbol{f}}_0$, $\tilde{\boldsymbol{f}}_0$ two different initial conditions. Performing one step of the standard implicit Euler's method yields the following bound

$$\begin{aligned}\left\|\hat{\boldsymbol{f}}_1 - \tilde{\boldsymbol{f}}_1\right\| &= \left\|\left(\hat{\boldsymbol{f}}_0 + \Delta t \boldsymbol{G}(\hat{\boldsymbol{f}}_1)\right) - \left(\tilde{\boldsymbol{f}}_0 + \Delta t \boldsymbol{G}(\tilde{\boldsymbol{f}}_1)\right)\right\| \\ &\leq \left\|\hat{\boldsymbol{f}}_0 - \tilde{\boldsymbol{f}}_0\right\| + \Delta t L_{\boldsymbol{G}} \left\|\hat{\boldsymbol{f}}_1 - \tilde{\boldsymbol{f}}_1\right\|.\end{aligned}$$

where $L_{\boldsymbol{G}}$ is the Lipschitz constant of $\boldsymbol{G}$. By collecting like terms, we obtain

$$\left\|\hat{\boldsymbol{f}}_1 - \tilde{\boldsymbol{f}}_1\right\| \leq \frac{1}{1 - \Delta t L_{\boldsymbol{G}}} \left\|\hat{\boldsymbol{f}}_0 - \tilde{\boldsymbol{f}}_0\right\| \leq (1 + 2\Delta t L_{\boldsymbol{G}}) \left\|\hat{\boldsymbol{f}}_0 - \tilde{\boldsymbol{f}}_0\right\|. \tag{3.25}$$

The last inequality comes by noting that $1/(1 - \Delta t L_{\boldsymbol{G}}) \leq 1 + 2\Delta t L_{\boldsymbol{G}}$ when Δt is sufficiently small, i.e., $\Delta t \leq 1/(2L_{\boldsymbol{G}})$. This is our zero-stability condition, i.e., a stability condition that holds for small Δt, which will be used for convergence analysis. Regarding the behavior of the scheme for finite Δt, see Section 4.5 for the proof that the implicit step-truncation Euler scheme is unconditionally stable. Next, we derive a stability condition of the form (3.25) when the root of $\boldsymbol{H}_k(\boldsymbol{f}_{k+1}) = \boldsymbol{0}$ is computed with the inexact Newton's method with HT/TT-GMRES iterations. In this case we have

$$\begin{aligned}\left\|\hat{\boldsymbol{f}}_1^{[j]} - \tilde{\boldsymbol{f}}_1^{[m]}\right\| &\leq \left\|\hat{\boldsymbol{f}}_1^{[j]} - \hat{\boldsymbol{f}}_1^{[\infty]}\right\| + \left\|\hat{\boldsymbol{f}}_1^{[\infty]} - \tilde{\boldsymbol{f}}_1^{[\infty]}\right\| + \left\|\tilde{\boldsymbol{f}}_1^{[m]} - \tilde{\boldsymbol{f}}_1^{[\infty]}\right\| \\ &= \left\|\hat{\boldsymbol{f}}_1^{[j]} - \hat{\boldsymbol{f}}_1\right\| + \left\|\hat{\boldsymbol{f}}_1 - \tilde{\boldsymbol{f}}_1\right\| + \left\|\tilde{\boldsymbol{f}}_1^{[m]} - \tilde{\boldsymbol{f}}_1\right\| \\ &\leq (1 + 2L_{\boldsymbol{G}}\Delta t) \left\|\hat{\boldsymbol{f}}_0 - \tilde{\boldsymbol{f}}_0\right\| + 2L_{\boldsymbol{H}^{-1}}\varepsilon_{\text{tol}}.\end{aligned} \tag{3.26}$$

Thus, the stability condition is satisfied by the same condition on ε_{tol} that satisfies the first order consistency condition (3.24), i.e., $\varepsilon_{\text{tol}} \leq K\Delta t^2$. At this point we apply Theorem 3 with consistency and stability conditions (4.21)-(4.22) replaced

by (3.24) and (3.26), respectively, and conclude that the implicit step-truncation Euler scheme is convergent with order one if $\varepsilon_{\text{tol}} \leq K\Delta t^2$.

Remark 3. *When $\boldsymbol{G}$ is linear, i.e., when the tensor ODE* (1.2) *is linear, then we may apply HT/TT-GMRES algorithm in Section 4.4.1 without invoking Newton's method. In this case, the local error coefficient $L_{\boldsymbol{H}_k^{-1}}$ can be exchanged for the coefficient of ε at the right side of inequality* (4.40)*.*

3.5 The Implicit Midpoint Method

The implicit midpoint rule [49],

$$\boldsymbol{f}_{k+1} = \boldsymbol{f}_k + \Delta t \boldsymbol{G}\left(\frac{1}{2}(\boldsymbol{f}_k + \boldsymbol{f}_{k+1})\right), \tag{3.27}$$

is a symmetric and symplectic method of order 2. By introducing

$$\boldsymbol{H}_k\left(\boldsymbol{f}_{k+1}\right) = \boldsymbol{f}_{k+1} - \boldsymbol{f}_k - \Delta t \boldsymbol{G}\left(\frac{1}{2}(\boldsymbol{f}_k + \boldsymbol{f}_{k+1})\right) = \boldsymbol{0} \tag{3.28}$$

we again see the implicit method as a root finding problem at each time step. To prove convergence of the implicit step-truncation midpoint method we follow the same steps described in the previous section. To this end, consider the sequence of tensors $\boldsymbol{f}_{k+1}^{[j]}$ generated by the inexact Newton method with HT/TT-GMRES iterations (see Section 4.4.1) applied to (3.28). The sequence of tensors $\boldsymbol{f}_{k+1}^{[j]}$ approximates $\boldsymbol{f}_{k+1}$ satisfying (3.28). As before, we terminate the inexact Newton's iterations as soon as condition (3.21) is satisfied. By repeating the same steps that led us to inequality (3.22), we have

$$\left\|\boldsymbol{f}(\Delta t) - \boldsymbol{f}_1^{[j]}\right\| \leq K_1 \Delta t^3 + L_{\boldsymbol{H}_k^{-1}} \varepsilon_{\text{tol}}. \tag{3.29}$$

Hence, setting the stopping tolerance as $\varepsilon_{\text{tol}} \leq K\Delta t^3$ we get second order consistency. We use this to determine the stability condition (4.22). As before, we derive the condition for when the zero of (3.28) is exact. Denote by $\hat{\boldsymbol{f}}_0$, $\tilde{\boldsymbol{f}}_0$ two different initial conditions. Performing one step of the standard implicit midpoint method yields the following bound

$$\left\|\hat{\boldsymbol{f}}_1 - \tilde{\boldsymbol{f}}_1\right\| \leq \left(1 + \Delta t\frac{L_{\boldsymbol{G}}}{2}\right)\left\|\hat{\boldsymbol{f}}_0 - \tilde{\boldsymbol{f}}_0\right\| + \Delta t\frac{L_{\boldsymbol{G}}}{2}\left\|\hat{\boldsymbol{f}}_1 - \tilde{\boldsymbol{f}}_1\right\|.$$

By collecting like terms we see that when $\Delta t \leq 1/L_{\boldsymbol{G}}$,

$$\begin{aligned}\left\|\hat{\boldsymbol{f}}_1 - \tilde{\boldsymbol{f}}_1\right\| &\leq \frac{2 + \Delta t L_{\boldsymbol{G}}}{2 - \Delta t L_{\boldsymbol{G}}}\left\|\hat{\boldsymbol{f}}_0 - \tilde{\boldsymbol{f}}_0\right\| \\ &\leq \left(1 + \Delta t\frac{3L_{\boldsymbol{G}}}{2}\right)\left\|\hat{\boldsymbol{f}}_0 - \tilde{\boldsymbol{f}}_0\right\|. \end{aligned} \tag{3.30}$$

The zero-stability condition for the implicit step-truncation midpoint method can now be found by repeating the arguments of inequality (3.26). This gives

$$\left\|\hat{\boldsymbol{f}}_1^{[j]} - \tilde{\boldsymbol{f}}_1^{[m]}\right\| \leq \left(1 + \Delta t\frac{3L_{\boldsymbol{G}}}{2}\right)\left\|\hat{\boldsymbol{f}}_0 - \tilde{\boldsymbol{f}}_0\right\| + 2L_{\boldsymbol{H}_k^{-1}}\varepsilon_{\text{tol}}. \tag{3.31}$$

Thus, the stability condition is satisfied by the same condition on ε_{tol} that satisfies the second order consistency condition (3.29), i.e., $\varepsilon_{\text{tol}} \leq K\Delta t^3$. At this point we apply Theorem 3 with consistency and stability conditions (4.21)-(4.22) replaced by (3.29) and (3.31), respectively, and conclude that the implicit step-truncation midpoint scheme is convergent with order two if $\varepsilon_{\text{tol}} \leq K\Delta t^3$.

3.6 Selection of Truncation Error Coefficients

In this section we provide a lower bound on the local error coefficients used throughout this chapter whenever a tensor rank truncation is applied. Specifically, we analyze the coefficients appearing in Proposition 1, the coefficients in inequalities (3.7,3.8), and their counterparts in the algorithms discussed throughout this chapter. We relate the error due to truncation to the singular value decay rate of a tensor. This rate may be used to tune the coefficients appearing in the local truncation error and zero-stability inequalities. To simplify the presentation we develop the bounds for the matrix case ($d = 2$) and note that similar results for $d > 2$ can be obtained by using the hierarchical approximability theorem discussed in [45].

With the local truncation error consistency in mind in particular, let $\{\sigma_i\}$ be the set of singular values of $\boldsymbol{a}_k$ and let $\boldsymbol{a}_k - \mathfrak{T}_r(\boldsymbol{a}_k) = \boldsymbol{E}_k \in \mathbb{R}^{n_1 \times n_2}$ be the error matrix due to SVD truncation. Then the local consistency condition of order p can be written as

$$\begin{aligned} \|\boldsymbol{a}_k - \mathfrak{T}_r(\boldsymbol{a}_k)\|_2^2 &= \|\boldsymbol{E}_k\|_2^2 \\ &= \sum_{i=r_k+1}^{\min(n_1,n_2)} \sigma_i^2 \leq M^2 \Delta t^{2p+2}. \end{aligned} \tag{3.32}$$

Equation (3.32) can be used to obtain the following lower bound for the coefficient M

$$M \geq \frac{1}{\Delta t^{p+1}} \sqrt{\sum_{i=r_k+1}^{\min(n_1,n_2)} \sigma_i^2}. \tag{3.33}$$

The lower bound can be explicitly computed if we have available the decay rate of the singular values $\{\sigma_i\}$. For instance, if the singular values decay exponentially fast (as in the case of singular values considered in [81]), i.e., $\sigma_i^2 \leq Cq^i$ for some

$C > 0$ and $q \in (0, 1)$, then by the geometric series formula we have that

$$\|\boldsymbol{a}_k\|_2^2 \leq C \sum_{i=1}^{\infty} q^i = C \frac{q}{(1-q)},$$

which yields $Cq \geq (1-q)\|\boldsymbol{a}_k\|_2^2$. In this case we may bound the local error as

$$\|\boldsymbol{E}_k\|_2^2 \leq C \sum_{i=r_k+1}^{\infty} q^i = C \sum_{i=1}^{\infty} q^i - C \sum_{i=1}^{r} q^i = C \frac{q - (q - q^{r+1})}{1-q} = C \frac{q^{r+1}}{1-q}.$$

Inserting this bound into (3.33) and recalling that $Cq \geq (1-q)\left\|\boldsymbol{a}_k\right\|_2^2$ and $\|\boldsymbol{a}_k\|_2 \geq \|\boldsymbol{E}_k\|_2$ we obtain

$$M \geq \frac{1}{\Delta t^{p+1}} \sqrt{\frac{C q^{r_k+1}}{(1-q)}} \geq \frac{1}{\Delta t^{p+1}} \sqrt{\frac{(1-q)\left\|\boldsymbol{a}_k\right\|_2^2 q^r}{(1-q)}} = \frac{\|\boldsymbol{a}_k\|_2 \sqrt{q^r}}{\Delta t^{p+1}}. \tag{3.34}$$

Equation (3.34) establishes a relationship between the local error coefficient M, the solution rank r at time step k, the time step Δt, and the 2-norm of the solution $\boldsymbol{a}_k$ at time step k. A similar relation can be derived for singular values $\{\sigma_i\}$ decaying algebraically, i.e., $\sigma_i^2 \leq C i^{-1-2s}$, where $s \in \mathbb{N}$. It was shown in [47] that this decay rate occurs when discretizing an s-times differentiable bivariate function. Moreover, it was also shown that

$$\|\boldsymbol{E}\|_2 \leq K \|\boldsymbol{a}_k\|_2 (r+1)^{-s},$$

where K is a constant related to the measure of the domain of the aforementioned s-times differentiable bivariate function. Therefore, if we choose the rank r to satisfy the inequality $M \Delta t^{p+1}/K \geq \|\boldsymbol{a}_k\|_2 (r_k+1)^{-s}$ then we have the local truncation error consistency condition is also satisfied. An expression for K may be found in Theorem 3.3 of [47].

Chapter 4

Analysis of Step-Truncation Methods

Here, we present a rigorous analysis of step-truncation methods, including proofs for convergence and absolute stability. A step-truncation temporal integration method consists of its two titular components. We begin by first discretizing the ODE (1.2)

$$\frac{d\boldsymbol{f}(t)}{dt} = \boldsymbol{G}(\boldsymbol{f}(t)), \qquad \boldsymbol{f}(0) = \boldsymbol{f}_0,$$

in time with any standard explicit one-step method on an evenly-spaced temporal grid

$$\boldsymbol{f}_{k+1} = \boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k). \tag{4.1}$$

Here, $\boldsymbol{f}_k$ denotes an approximation of the exact solution $\boldsymbol{f}(k\Delta t)$ for $k = 1, 2, ..., N$, and $\boldsymbol{\Psi}_{\Delta t}$ is a discrete time approximation of the flow map of an ODE, such as (1.2). For example, $\boldsymbol{\Psi}_{\Delta t}$ can be the approximate local flow corresponding to the

Euler forward method

$$\boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k) = \boldsymbol{f}_k + \Delta t \boldsymbol{G}(\boldsymbol{f}_k). \tag{4.2}$$

Secondly, in the interest of saving computational resources when iterating (4.1) we look for an approximation of $\boldsymbol{f}_k$ on a low-rank tensor manifold $\mathcal{H}_{\boldsymbol{r}}$ [107] with multilinear rank $\boldsymbol{r}$ by the use of the nonlinear SVD truncation operator [45]. This yields the explicit step-truncation scheme

$$\boldsymbol{f}_{k+1} = \mathfrak{T}_{\boldsymbol{r}} \left(\boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k) \right). \tag{4.3}$$

The rank $\boldsymbol{r}$ can vary with the time step based on appropriate error estimates as time integration proceeds [92]. Though some theorems still hold when using the optimally accurate truncation $\mathfrak{T}_{\boldsymbol{r}}^{\text{best}}$, such as the core theorem of Section 4.2, the stability properties may differ. For this reason, we assume the use of the SVD truncation operator throughout this chapter.

4.1 Linear Stability for Explicit Step-Truncation

The fully discrete method (4.3) is said to be space-stable if for some $T > 0$ the solution is bounded for $0 < t < T$, for arbitrary initial conditions in some Banach space, as the number of degrees of freedom (e.g., mesh points or modes) increases. The notion of bounded stability we are capturing in this section is a variation of absolute stability, rather than zero stability. Contrasting, zero stability places a continuity condition on the discrete flow map with respect to initial conditions. When the differential equation under study is linear and the temporal evolution scheme is a linear map, these notions' formulae coincide. However, the low-rank

truncation operators break linearity. We are actually using a nonlinear update step, even when the PDE we are solving is linear. Thus, we study the various notions of stability independently. In this section we address stability for explicit discretizations of linear problems. I.e., the (typically nonlinear) discrete time flow map may be replaced with a linear operator $\boldsymbol{\Psi}_{\Delta t}\left(\boldsymbol{G}, \boldsymbol{f}_k\right) = \boldsymbol{L}_{\Delta t}\boldsymbol{f}_k$. This method defines a stable method under norm $||\cdot||$ method if

$$\left\|\left(\boldsymbol{L}_{\Delta t}\right)^n \boldsymbol{v}_0 - \left(\boldsymbol{L}_{\Delta t}\right)^n \boldsymbol{w}_0\right\| \leq C_T \left\|\boldsymbol{v}_0 - \boldsymbol{w}_0\right\| \tag{4.4}$$

for any pair of initial states $\boldsymbol{v}_0, \boldsymbol{w}_0$. The real number $C_T \geq 0$ is the Lipschitz constant of the scheme for a fixed integration time T. C_T must be independent of $\boldsymbol{v}_0$, $\boldsymbol{w}_0$, the spacial resolution in $\boldsymbol{G}$, as well as Δt and n if $n\Delta t \leq T$. Reddy and Trefethen [88] showed that we may instead check if the scheme is power-bounded. It is easy see by the linearity of $\boldsymbol{L}_{\Delta t}$ that the above condition is equivalent to the inequality

$$\left\|\left(\boldsymbol{L}_{\Delta t}\right)^n \boldsymbol{z}_0\right\| \leq C_T \left\|\boldsymbol{z}_0\right\|, \tag{4.5}$$

where $\boldsymbol{z}_0 = \boldsymbol{v}_0 - \boldsymbol{w}_0$ is arbitrary. Clearly, if $\boldsymbol{z}_0 = \boldsymbol{0}$ then equality is achieved. So we lose no generality by imposing $\boldsymbol{z}_0 \neq \boldsymbol{0}$. Divide both sides by $||\boldsymbol{z}^0||$ to obtain

$$\frac{\left\|\left(\boldsymbol{L}_{\Delta t}\right)^n \boldsymbol{z}_0\right\|}{\left\|\boldsymbol{z}_0\right\|} \leq C_T. \tag{4.6}$$

It is immediate that the inequality holds for all $\boldsymbol{z}_0 \neq \boldsymbol{0}$ if and only if it holds for the maximizing $\boldsymbol{z}_0$. Therefore, stability for a linear update is equivalent to the time stepping operator being power bounded, i.e.,

$$\left\|\left(\boldsymbol{L}_{\Delta t}\right)^n\right\| = \max_{\boldsymbol{z}\neq 0} \frac{\left\|\left(\boldsymbol{L}_{\Delta t}\right)^n \boldsymbol{z}\right\|}{\left\|\boldsymbol{z}\right\|} \leq C_T. \tag{4.7}$$

If we let $||\cdot|| = ||\cdot||_2$, then the operator norm is the spectral radius. Reddy and Trefethen [88, 89] proved that power bounded property for $n \to \infty$ and fixed Δt is equivalent to a statement about the eigenvalues of "nearby" linear operators, the so-called ε-eigenvalues. Specifically, given any $\varepsilon > 0$, there exists $\boldsymbol{E}$ with $||\boldsymbol{E}||_2 \leq \varepsilon$ satisfying

$$\|\boldsymbol{L}_{\Delta t} + \boldsymbol{E}\|_2 \leq 1 + C_T \varepsilon. \tag{4.8}$$

They also show that if $n \to \infty$ is changed to the condition $n\Delta t \leq T$, then there is a direct generalization of the Lax stability inequality [66]. Specifically, given any $\varepsilon > 0$, there exists $\boldsymbol{E}$ with $||\boldsymbol{E}||_2 \leq \varepsilon$ satisfying

$$\|\boldsymbol{L}_{\Delta t} + \boldsymbol{E}\|_2 \leq 1 + K_T \varepsilon + Q_T \Delta t \qquad \text{for all } n \text{ such that} \quad n\Delta t \leq T. \tag{4.9}$$

The inequality above may be interpreted as "the operator $\boldsymbol{L}_{\Delta t}$ can be perturbed in a such a way that its eigenvalues grow linearly with time step away from a slightly enlarged unit disk in the complex plane." In their paper, the statement is made in terms of the spectral radius rather than operator norms. We now have all elements to prove stability of linear integrators on low-rank tensor manifolds.

Theorem 2 (Absolute Stability of for Step-Truncation)**.** *Suppose*

$$\boldsymbol{f}_{k+1} = \boldsymbol{L}_{\Delta t} \boldsymbol{f}_k \tag{4.10}$$

defines a Lax-stable linear method, i.e. $\|\boldsymbol{L}_{\Delta t}\|_2 \leq 1 + K\Delta t$*. Then the rank-truncated scheme*

$$\boldsymbol{f}_{k+1} = \mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{L}_{\Delta t} \boldsymbol{f}_k) \tag{4.11}$$

is bounded with the same stability constant as (4.10)*.*

Proof. Recall that the truncation operator under study is $\mathfrak{T}_{\boldsymbol{r}}^{\mathrm{SVD}} = \mathfrak{T}_{\boldsymbol{r}}$. Moreover, the theorem holds with an identical proof regardless of whether or not the rank $\boldsymbol{r}$ is time dependent. (The operator norm of the truncation is not dependent on the particular rank.) We proceed by induction. For $k = 1$ the theorem follows from inequality (2.27). For $k > 1$ we utilize Lemma 2, and write

$$\|\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{L}_{\Delta t}\boldsymbol{f}_k)\|_2 \leq \|\boldsymbol{L}_{\Delta t}\|_2^k \|\boldsymbol{f}_0\|_2 . \tag{4.12}$$

Then we recall that stability of a linear method in this form is equivalent[88] to

$$\left\|(\boldsymbol{L}_{\Delta t})^k\right\|_2 \leq C_T \quad \forall k\Delta t \leq T. \tag{4.13}$$

Assuming that the linear recurrence (4.10) is Lax-stable, we have

$$\left\|(\boldsymbol{L}_{\Delta t})^k\right\|_2 \leq \|(\boldsymbol{L}_{\Delta t})\|_2^k \leq (1 + K\Delta t)^k \leq e^{K\Delta t \cdot k} = e^{KT} = C_T.$$

Therefore,

$$\begin{aligned}
\|\boldsymbol{f}_{k+1}\|_2 = \|\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{L}_{\Delta t}\boldsymbol{f}_k)\|_2 &\leq \|\mathfrak{T}_{\boldsymbol{r}}\|_2 \|\boldsymbol{L}_{\Delta t}\boldsymbol{f}_k\|_2 \\
&\leq \|\boldsymbol{L}_{\Delta t}\boldsymbol{f}_k\|_2 \\
&= \|\boldsymbol{L}_{\Delta t}\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{L}_{\Delta t}\boldsymbol{f}_{k-1})\|_2 \\
&\leq \|\boldsymbol{L}_{\Delta t}\|_2 \|\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{L}_{\Delta t}\boldsymbol{f}_{k-1})\|_2 \\
&\leq \|\boldsymbol{L}_{\Delta t}\|_2^k \|\boldsymbol{f}_0\|_2 \qquad \text{(Inductive Hypothesis)} \\
&\leq C_T \|\boldsymbol{f}_0\|_2 .
\end{aligned}$$

Thus the iteration remains bounded. □

We note that Theorem 2 is a sufficiency condition. This does not exclude the

possibility that the truncation operator $\mathfrak{T}_{\boldsymbol{r}}^{\mathrm{SVD}}$ can stabilize an unstable scheme.

4.2 Consistency and Convergence for Ordinary Differential Equations

Up to this point, we have studied the effects of an individual truncation on the iterates of a conventional linear time-stepping scheme. However, a conventional time-stepping scheme may have stages which are amenable to low-rank tensor approximations. To illustrate this, a step-truncation explicit Euler method

$$\boldsymbol{f}_{k+1} = \mathfrak{T}_{\boldsymbol{r}} \left(\boldsymbol{f}_k + \Delta t \boldsymbol{G}(\boldsymbol{f}_k) \right), \tag{4.14}$$

may be further modified to incorporate additional compressions (truncations) for further memory savings. E.g., truncating $\boldsymbol{G}(\boldsymbol{f})$ before applying $\boldsymbol{\Psi}_{\Delta t}$ yields

$$\boldsymbol{f}_{k+1} = \mathfrak{T}_{\boldsymbol{r}_2} \left(\boldsymbol{f}_k + \Delta t \mathfrak{T}_{\boldsymbol{r}_1} \left(\boldsymbol{G}(\boldsymbol{f}_k) \right) \right). \tag{4.15}$$

Here $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are truncation ranks determined by the inequalities[1]

$$\| \boldsymbol{G}(\boldsymbol{f}_k) - \mathfrak{T}_{\boldsymbol{r}_1} \left(\boldsymbol{G}(\boldsymbol{f}_k) \right) \| \leq e_1, \qquad \left\| \widetilde{\boldsymbol{f}}_{k+1} - \mathfrak{T}_{\boldsymbol{r}_2} \left(\widetilde{\boldsymbol{f}}_{k+1} \right) \right\| \leq e_2, \tag{4.16}$$

where e_1 and e_2 are chosen error thresholds. As before, $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ can change with every time step. In particular, we are now interested in what error controls on e_1 and e_2 preserve consistency, stability, and provide convergence of the iterates $\boldsymbol{f}_k$

[1] Throughout the remainder of this chapter, $\|\cdot\|$ denotes the standard tensor 2-norm [45, 64], or a weighted version of it.

to differential equation (1.2). More generally, let

$$\boldsymbol{f}_{k+1} = \boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k, \boldsymbol{e}) \tag{4.17}$$

be an explicit step-truncation method in which we project all $\boldsymbol{G}(\boldsymbol{f}_k)$ and tensors appearing in the local flow approximation $\boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k)$ onto tensor manifolds $\overline{\mathcal{H}}_{\boldsymbol{r}_i}$ by setting suitable error thresholds $\boldsymbol{e} = (e_1, e_2, \ldots)$. For instance, if $\boldsymbol{\Psi}_{\Delta t}$ is defined by the explicit midpoint method, i.e.,

$$\boldsymbol{\Psi}_{\Delta t}\left(\boldsymbol{G}, \boldsymbol{f}_k\right) = \boldsymbol{f}_k + \boldsymbol{G}\left(\boldsymbol{f}_k + \frac{\Delta t}{2}\boldsymbol{G}(\boldsymbol{f}_k)\right) \tag{4.18}$$

then

$$\boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k, \boldsymbol{e}) = \mathfrak{T}_{\boldsymbol{r}_3}\left(\boldsymbol{f}_k + \mathfrak{T}_{\boldsymbol{r}_2}\left[\boldsymbol{G}\left(\boldsymbol{f}_k + \frac{\Delta t}{2}\mathfrak{T}_{\boldsymbol{r}_1}\left[\boldsymbol{G}(\boldsymbol{f}_k)\right]\right)\right]\right), \tag{4.19}$$

where $\boldsymbol{e} = (e_1, e_2, e_3)$ is a vector collecting the truncation error thresholds yielding the multilinear ranks $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{r}_3$. By construction, step-truncation methods of the form (4.17) satisfy

$$\|\boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}) - \boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}, \boldsymbol{e})\| \leq R(\boldsymbol{e}), \tag{4.20}$$

where $R(\boldsymbol{e})$ is the error due to tensor truncation. We close this section with a general convergence theorem for step-truncation temporal integrators. In Chapter 3, we provide several examples of step-truncation methods which satisfy this theorem and the conditions required on their truncation accuracy.

Theorem 3 (Convergence of step-truncation methods)**.** *Let* $\boldsymbol{\varphi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f})$ *be the one-step exact flow map defined by* (1.2)*, and* $\boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}, \boldsymbol{e})$ *be the approximate local flow defining a step-truncation method with local error of order* p*, i.e.,*

$$\|\boldsymbol{\varphi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}) - \boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}, \boldsymbol{e})\| \leq K\Delta t^{p+1} \quad as \quad \Delta t \to 0. \tag{4.21}$$

If there exist truncation errors $\boldsymbol{e} = \boldsymbol{e}(\Delta t)$ *(function of* Δt*) and constants* $C, E > 0$ *(dependent on* $\boldsymbol{G}$*) so that the zero-stability condition*

$$\left\|\boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \hat{\boldsymbol{f}}, \boldsymbol{e}) - \boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \tilde{\boldsymbol{f}}, \boldsymbol{e})\right\| \leq (1 + C\Delta t)\left\|\hat{\boldsymbol{f}} - \tilde{\boldsymbol{f}}\right\| + E\Delta t^{m+1} \tag{4.22}$$

holds as $\Delta t \to 0$*, then the step-truncation method is convergent with order* $z = min(m, p)$*.*

Proof. Under the assumption that Δt is small enough for our stability and consistency to hold, we proceed by induction on the number of steps. The one-step case coincides with the consistency condition. Next, we assume that

$$\left\|\boldsymbol{\varphi}_{\Delta t(N-1)}(\boldsymbol{G}, \boldsymbol{f}_0) - \boldsymbol{f}_{N-1}\right\| \leq Q_{N-1}\Delta t^z. \tag{4.23}$$

Let $T = N\Delta t$ be the final integration time. By applying triangle inequality and the semigroup property of the flow map,

$$\begin{aligned}\|\boldsymbol{\varphi}_T(\boldsymbol{G}, \boldsymbol{f}_0) - \boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_{N-1}, \boldsymbol{e})\| \leq \\ \left\|\boldsymbol{\varphi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{\varphi}_{\Delta t(N-1)}(\boldsymbol{G}, \boldsymbol{f}_0)) - \boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{\varphi}_{\Delta t(N-1)}(\boldsymbol{G}, \boldsymbol{f}_0), \boldsymbol{e})\right\| \\ + \left\|\boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{\varphi}_{\Delta t(N-1)}(\boldsymbol{G}, \boldsymbol{f}_0), \boldsymbol{e}) - \boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_{N-1}, \boldsymbol{e})\right\|.\end{aligned} \tag{4.24}$$

Using the consistency condition (4.21) we can bound the first term at the right

hand side of (4.24) by $K_{N-1}\Delta t^{p+1}$, where K_{N-1} represents a local error coefficient. On the other hand, using the stability condition (4.22) we can can bound the second term at the right hand side of (4.24) as

$$\left\|\boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{\varphi}_{\Delta t(N-1)}(\boldsymbol{G}, \boldsymbol{f}_0), \boldsymbol{e}) - \boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_{N-1}, \boldsymbol{e})\right\| \leq \\ (1 + C\Delta t)\left\|\boldsymbol{\varphi}_{\Delta t(N-1)}(\boldsymbol{G}, \boldsymbol{f}_0) - \boldsymbol{f}_{N-1}\right\| + E\Delta t^{m+1}. \tag{4.25}$$

A substitution of (4.25) and (4.23) into (4.24) yields

$$\|\boldsymbol{\varphi}_T(\boldsymbol{G}, \boldsymbol{f}_0) - \boldsymbol{\Phi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_{N-1}, \boldsymbol{e})\| \leq K_{N-1}\Delta t^{p+1} + (1 + C\Delta t)Q_{N-1}\Delta t^z + E\Delta t^{m+1}.$$

Recalling that $z = \min(m, p)$ completes the proof. □

We remark that the above proof may be modified to include explicit step-truncation linear multi-step methods, i.e., step-truncation Adams methods. To this end, it is sufficient to replace $\boldsymbol{f}_k$ with the vector $(\boldsymbol{f}_{k-s}, \boldsymbol{f}_{k-s+1}, \ldots, \boldsymbol{f}_k)$ and the stability condition (4.22) with a suitable multistep alternative. For a complete analysis of this for Adams-Bashforth methods, see Section 3.3.

4.3 Rank Adaptivity and Tensor Manifolds

One interpretation of Theorem 3 is "It is sufficient to adapt the rank of a step-truncation time update in such a way so as to guarantee zero stability and consistency in order to achieve long-time convergence." We now provide a partial converse. If one does not adapt the rank and instead holds it constant, then in the limit of small time step the velocity of the trajectory $\boldsymbol{f}(t)$ will point tangent to $\mathcal{H}_{\boldsymbol{r}}$.

Proposition 1 (Geometric interpretation of rank adaptivity)**.** *Let* $\boldsymbol{g} \in \mathcal{H}_{\boldsymbol{r}}$ *and* $\boldsymbol{v} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$*. The following are equivalent as* $\Delta t \to 0$*:*

$$\exists K \geq 0 \text{ so that } \|(\boldsymbol{I} - \mathcal{P}_{\boldsymbol{g}})\boldsymbol{v}\|_2 \leq K\Delta t, \tag{4.26}$$

$$\exists M \geq 0 \text{ so that } \left\|(\boldsymbol{g} + \Delta t\boldsymbol{v}) - \mathfrak{T}_{\boldsymbol{r}}^{best}(\boldsymbol{g} + \Delta t\boldsymbol{v})\right\|_2 \leq M\Delta t^2, \tag{4.27}$$

$$\exists N \geq 0 \text{ so that } \left\|(\boldsymbol{g} + \Delta t\boldsymbol{v}) - \mathfrak{T}_{\boldsymbol{r}}^{SVD}(\boldsymbol{g} + \Delta t\boldsymbol{v})\right\|_2 \leq N\Delta t^2. \tag{4.28}$$

Proof. The equivalence between (4.27) and (4.28) is an immediate consequence of (2.22). We now prove that (4.26) is equivalent to (4.27). For the forward implication, assume $\|(\boldsymbol{I} - \mathcal{P}_{\boldsymbol{g}})\boldsymbol{v}\|_2 \leq K\Delta t$. We have

$$\begin{aligned}\left\|(\boldsymbol{g} + \Delta t\boldsymbol{v}) - \mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{g} + \Delta t\boldsymbol{v})\right\|_2 &\leq \|(\boldsymbol{g} + \Delta t\boldsymbol{v}) - (\boldsymbol{g} + \Delta t\mathcal{P}_{\boldsymbol{g}}\boldsymbol{v})\|_2 + \Delta t^2 C \\ &= \Delta t \|\boldsymbol{v} - \mathcal{P}_{\boldsymbol{g}}\boldsymbol{v}\|_2 + \Delta t^2 C \\ &\leq \Delta t^2 K + \Delta t^2 C,\end{aligned}$$

where $C \geq 0$ denotes a constant obtained by a Taylor expansion of $\mathfrak{T}_{\boldsymbol{r}}^{\text{best}}$ (see (2.24)). Setting $M \geq K + C$, proves the forward implication. Conversely, if we assume $\left\|(\boldsymbol{g} + \Delta t\boldsymbol{v}) - \mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{g} + \Delta t\boldsymbol{v})\right\|_2 \leq M\Delta t^2$, then

$$\begin{aligned}\|(\boldsymbol{I} - \mathcal{P}_{\boldsymbol{g}})\boldsymbol{v}\|_2 &= \Delta t^{-1} \|\boldsymbol{g} + \Delta t\boldsymbol{v} - (\boldsymbol{g} + \Delta t\mathcal{P}_{\boldsymbol{g}}\boldsymbol{v})\|_2 \\ &\leq \Delta t^{-1} \left(\left\|(\boldsymbol{g} + \Delta t\boldsymbol{v}) - \mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{g} + \Delta t\boldsymbol{v})\right\|_2 + C\Delta t^2\right) \\ &\leq \Delta t M + \Delta t C.\end{aligned}$$

Setting $K \geq M + C$, we prove (4.27) implies (4.26). □

The rank increase criterion (4.26) for the tangent space projection offers geometric intuition which is not apparent from the step-truncation rank criteria

(4.27)-(4.28). That is, the solution rank should increase if the dynamics do not admit a sufficient approximation on the tensor manifold tangent space. Moreover, the truncation accuracy required for approximating the dynamics depends directly on the time step size Δt and the desired order of accuracy. By applying condition (4.26) to an ordinary differential equation projected tangent to the manifold $\mathcal{H}_{\boldsymbol{r}}$ at point $\boldsymbol{f}_k$,

$$\boldsymbol{f}_{k+1} = \boldsymbol{\Phi}_{\Delta t}\left(\mathcal{P}_{\boldsymbol{f}_k}\boldsymbol{G}, \boldsymbol{f}_k\right), \qquad \boldsymbol{f}(0) = \mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{f}_0). \tag{4.29}$$

it is possible to develop a rank-adaptive version of the step-truncation scheme proposed in [59]. Specifically, the solution rank $\boldsymbol{r}$ at each time step can be chosen to satisfy a bound on the component of (4.29) normal to the tensor manifold $\mathcal{H}_{\boldsymbol{r}}$.

4.4 Implicit Step-Truncation and Root Finding for Tensor Decompositions

To introduce implicit step-truncation tensor methods, let us begin with the standard Euler backward scheme

$$\boldsymbol{f}_{k+1} = \boldsymbol{f}_k + \Delta t \boldsymbol{G}(\boldsymbol{f}_{k+1}), \tag{4.30}$$

and the associated root-finding problem

$$\boldsymbol{H}_k(\boldsymbol{f}_{k+1}) = \boldsymbol{f}_{k+1} - \boldsymbol{f}_k - \Delta t \boldsymbol{G}(\boldsymbol{f}_{k+1}) = \boldsymbol{0}. \tag{4.31}$$

Equation (4.31) allows us to compute $\boldsymbol{f}_{k+1}$ as a zero of the function $\boldsymbol{H}_k$. This can be done, e.g., using the Netwon's method with initial guess $\boldsymbol{f}_{k+1}^{[0]} = \boldsymbol{f}_k$. As is well-known, if the Jacobian of $\boldsymbol{H}_k$ is invertible within a neighborhood of $\boldsymbol{f}_k$, then

the implicit function theorem guarantees the existence of a locally differentiable (in some neighborhood of $\boldsymbol{f}_k$) nonlinear map $\boldsymbol{\Theta}_{\Delta t}$ depending on $\boldsymbol{G}$ such that

$$\boldsymbol{f}_{k+1} = \boldsymbol{\Theta}_{\Delta t}\left(\boldsymbol{G}, \boldsymbol{f}_k\right). \tag{4.32}$$

In the setting of Newton's method described above, the map $\boldsymbol{\Theta}_{\Delta t}$ is computed iteratively. An implicit step-truncation scheme can be then formulated by applying the tensor truncation operator $\mathfrak{T}_{\boldsymbol{r}}$ to the right hand side of (4.32), i.e.,

$$\boldsymbol{f}_{k+1} = \mathfrak{T}_{\boldsymbol{r}}\left(\boldsymbol{\Theta}_{\Delta t}\left(\boldsymbol{G}, \boldsymbol{f}_k\right)\right). \tag{4.33}$$

The tensor truncation rank $\boldsymbol{r}$ can be selected based on the inequality

$$\|\mathfrak{T}_{\boldsymbol{r}}(\boldsymbol{\Theta}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k)) - \boldsymbol{\Theta}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k)\| \leq A\Delta t^2. \tag{4.34}$$

With some additional effort to show that this truncated iteration is zero-stable, it follows from Theorem 3 that this yields an order one (in time) integration scheme. Of course, if the Jacobian of $\boldsymbol{H}_k$ in equation (4.31) can be computed and stored in computer memory, then we can approximate $\boldsymbol{f}_{k+1} = \boldsymbol{\Theta}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k)$ for any given $\boldsymbol{f}_k$ and $\boldsymbol{G}$ using Newton's iterations. However, the exact Newton's method is not available to us in the high-dimensional tensor setting. Hence, we look for an approximation of $\boldsymbol{\Theta}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k)$ computed using the inexact Newton method. We briefly discuss the general approach of the inexact Newton iteration. The algorithm dates to a 1982 article [34] by Dembo et al. which allows one to apply approximate inverses of the Jacobian in exchange for weakening the convergence rate of the Cauchy sequence being produced. In our case, convergence is either superlinear or linear rather than quadratic. To illustrate this algorithm, consider

the iteration for solving a system of nonlinear equations written as $\boldsymbol{H}(\boldsymbol{f}) = \boldsymbol{0}$. The typical iteration for this would be written as

$$\boldsymbol{f}^{[j+1]} = \boldsymbol{f}^{[j]} - \left(\boldsymbol{J}_{\boldsymbol{H}}\left(\boldsymbol{f}^{[j]}\right)\right)^{-1} \boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right),$$

where $\boldsymbol{J}_{\boldsymbol{H}}\left(\boldsymbol{f}^{[j]}\right)$ is the Jacobian of $\boldsymbol{H}$ evaluated at $\boldsymbol{f}^{[j]}$. The inexact version of this iteration replaces the inverse Jacobian term with an approximate solution to a system of linear equations which must be satisfied up to a specified relative error. This algorithm was not described in the context of tensor truncations or low-rank methods originally. It only makes assumptions about the relative accuracy of the inverse Jacobian operator evaluation. In our work, we apply a matrix-free approach in which we evaluate the Jacobian by representing it as a function which gives a new HT tensor as its output. For the inexact matrix inverse step, we apply the relaxed TT-GMRES algorithm described in [40]. This is an iterative method for the solution of linear equations which makes use of an inexact matrix-vector product defined by low-rank truncation. Though the algorithm was developed for TT tensors, it may be also applied to HT tensors without significant changes.

4.4.1 The Inexact Newton Iteration

We describe the inexact Newton's method with HT/TT-GMRES iterations to solve an arbitrary algebraic equation of the form $\boldsymbol{H}_k(\boldsymbol{f}) = \boldsymbol{0}$ (e.g., equation (4.31)) on a tensor manifold with a given rank. The algorithm can be used to approximate the mapping $\boldsymbol{\Theta}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k)$ corresponding to any implicit integrator, and it returns a tensor which can be then truncated further as in (4.33). A large class of implicit methods can be equivalently formulated as a root-finding problem

for a nonlinear system of algebraic equations in the form

$$\boldsymbol{H}(\boldsymbol{f}) = \boldsymbol{0} \tag{4.35}$$

at each time step. In this section we develop numerical algorithms to compute an approximate solution of (4.35) on a tensor manifold $\mathcal{H}_{\boldsymbol{r}}$ with rank $\boldsymbol{r}$. In other words, we are interested in finding $\boldsymbol{r}$ so that $\boldsymbol{f} \in \mathcal{H}_{\boldsymbol{r}}$ that solves (4.35) with controlled accuracy. To this end, we combine the inexact inexact Newton method [34, Thm. 2.3 and Cor. 3.5] with the TT-GMRES linear solver proposed in [40].

Theorem 4 (Inexact Newton method [34])**.** *Let $\boldsymbol{H} : \mathbb{R}^N \to \mathbb{R}^N$ be continuously differentiable in a neighborhood of a zero $\boldsymbol{f}^*$, and suppose that the Jacobian of $\boldsymbol{H}$, i.e., $\boldsymbol{J}_{\boldsymbol{H}}(\boldsymbol{f}) = \partial \boldsymbol{H}(\boldsymbol{f})/\partial \boldsymbol{f}$, is invertible at $\boldsymbol{f}^*$. Given $\boldsymbol{f}^{[0]} \in \mathbb{R}^N$, consider the sequence*

$$\boldsymbol{f}^{[j+1]} = \boldsymbol{f}^{[j]} + \boldsymbol{s}^{[j]}, \qquad j = 0, 1, \ldots \tag{4.36}$$

where each $\boldsymbol{s}^{[j]}$ solves the Newton iteration up to relative error $\eta^{[j]}$, i.e., it satisfies

$$\left\| \boldsymbol{J}_{\boldsymbol{H}}\left(\boldsymbol{f}^{[j]}\right) \boldsymbol{s}^{[j]} + \boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right) \right\| \leq \left\| \boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right) \right\| \eta^{[j]}. \tag{4.37}$$

If $\eta^{[j]} < \tau < 1$ for all j, then there exists $\varepsilon > 0$ so that for any initial guess satisfying $\left\| \boldsymbol{f}^{[0]} - \boldsymbol{f}^* \right\| < \varepsilon$, the sequence $\{\boldsymbol{f}^{[j]}\}$ converges linearly to $\boldsymbol{f}^*$. If in addition, $\eta^{[j]} \to 0$ as $j \to \infty$, then the convergence speed is superlinear.

The proof is a nontrivial analysis of inserting a relative error of the approximate Jacobian inverse application into the formula for a Newton iteration. In Appendix B we present a proof of Theorem 4 by modifying the proof of [34, Thm. 2.3] to analyze the effect of a relative error term dependent on the iteration number. The

next question is how to compute an approximate solution of the linear system

$$\boldsymbol{J_H}\left(\boldsymbol{f}^{[j]}\right)\boldsymbol{s}^{[j]} = -\boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right) \tag{4.38}$$

satisfying the bound (4.37), without inverting the Jacobian $\boldsymbol{J_H}\left(\boldsymbol{f}^{[j]}\right)$ and assuming that $\boldsymbol{s}^{[j]} \in \mathcal{H}_{\boldsymbol{r}_j}$, i.e., that $\boldsymbol{s}^{[j]}$ is a tensor with rank $\boldsymbol{r}_j$. To this end we utilize the relaxed HT/TT-GMRES method discussed in [40]. HT/TT-GMRES is an adapted tensor-structured generalized minimal residual (GMRES) method to solve linear systems in a tensor format. The solver employs an indirect accuracy check and a stagnation restart check in its halting criterion which we summarize in the following Lemma.

Lemma 4 (Accuracy of HT/TT-GMRES [40])**.** *Let $\boldsymbol{J}\boldsymbol{f} = \boldsymbol{b}$ be a linear system where $\boldsymbol{f}$, $\boldsymbol{b}$ are tensors in HT or TT format, and $\boldsymbol{J}$ is a bounded, invertible linear operator on $\boldsymbol{f}$. Let $\{\boldsymbol{f}^{[0]}, \boldsymbol{f}^{[1]}, \ldots\}$ be the sequence of approximate solutions generated by HT/TT-GMRES algorithm in [40], and $\varepsilon > 0$ be the stopping tolerance for the iterations. Then*

$$\left\|\boldsymbol{J}\boldsymbol{f}^{[j]} - \boldsymbol{b}\right\| \leq m\|\boldsymbol{J}\|\|\boldsymbol{J}^{-1}\|\|\boldsymbol{b}\|\varepsilon, \tag{4.39}$$

where m is the number of Krylov iterations performed before restart. Similarly, the distance between $\boldsymbol{f}^{[j]}$ and the exact solution $\boldsymbol{f}$ can be bounded as

$$\left\|\boldsymbol{f}^{[j]} - \boldsymbol{f}\right\| \leq m\|\boldsymbol{J}\|\|\boldsymbol{J}^{-1}\|^2\|\boldsymbol{b}\|\varepsilon. \tag{4.40}$$

The proof of Lemma 4 is a rather trivial manipulation of the error terms present in GMRES with inexact operator application. In [40], the result is simply derived in a section of the work and not given its own theorem. We can now combine

the HT/TT-GMRES linear solver with the inexact Newton method, to obtain an algorithm that allows us to solve nonlinear algebraic equations of the form (4.35) on a tensor manifold.

Theorem 5 (HT/TT Newton method)**.** *Let* $\boldsymbol{H} : \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d} \to \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ *be a continuously differentiable nonlinear map which operates on HT or TT tensor formats, and let* $\boldsymbol{f}^*$ *be a zero of* $\boldsymbol{H}$*. Suppose that the Jacobian of* $\boldsymbol{H}$*, denoted as* $\boldsymbol{J_H}(\boldsymbol{f})$*, is invertible at* $\boldsymbol{f}^*$*. Given an initial guess* $\boldsymbol{f}^{[0]}$*, consider the iteration*

$$\boldsymbol{f}^{[j+1]} = \mathfrak{T}_{\boldsymbol{r}} \left(\boldsymbol{f}^{[j]} + \boldsymbol{s}^{[j]} \right) \tag{4.41}$$

where $\boldsymbol{s}^{[j]}$ *is the HT/TT-GMRES solution of* $\boldsymbol{J_H}\left(\boldsymbol{f}^{[j]}\right)\boldsymbol{s}^{[j]} = -\boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right)$ *satisfying*

$$\left\| \boldsymbol{J_H}\left(\boldsymbol{f}^{[j]}\right)\boldsymbol{s}^{[j]} + \boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right) \right\| \leq \frac{1}{2} \left\| \boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right) \right\| \eta^{[j]}, \tag{4.42}$$

where $\eta^{[j]}$ *is the relative error, which can be any value satisfying* $0 \leq \eta^{[j]} < \tau < 1$*. Then the Newton iteration converges linearly so long as for each* j *the rank* $\boldsymbol{r}$ *of the truncation operator* $\mathfrak{T}_{\boldsymbol{r}}$ *is chosen to satisfy*

$$\left\| \mathfrak{T}_{\boldsymbol{r}} \left(\boldsymbol{f}^{[j]} + \boldsymbol{s}^{[j]} \right) - \boldsymbol{f}^{[j]} - \boldsymbol{s}^{[j]} \right\| \leq \frac{\left\| \boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right) \right\|}{2 \left\| \boldsymbol{J_H}\left(\boldsymbol{f}^{[j]}\right) \right\|} \eta^{[j]}. \tag{4.43}$$

Proof. Let $\boldsymbol{f}^{[0]}$ be an initial guess. Consider the sequence

$$\tilde{\boldsymbol{f}}^{[j+1]} = \boldsymbol{f}^{[j]} + \boldsymbol{s}^{[j]} \qquad j = 0, 1, \ldots \tag{4.44}$$

where $\boldsymbol{s}^{[j]}$ is the HT/TT-GMRES solution of $\boldsymbol{J_H}\left(\boldsymbol{f}^{[j]}\right)\boldsymbol{s}^{[j]} = -\boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right)$ obtained with tolerance

$$\varepsilon_j < \frac{\eta^{[j]}}{2m \left\| \boldsymbol{J_H}\left(\boldsymbol{f}^{[j]}\right) \right\| \left\| \boldsymbol{J_H}^{-1}\left(\boldsymbol{f}^{[j]}\right) \right\|}, \tag{4.45}$$

and $0 \leq \eta^{[j]} < 1$. The next step is to truncate $\tilde{\boldsymbol{f}}^{[j+1]}$ to a tensor

$$\boldsymbol{f}^{[j+1]} = \mathfrak{T}_{\boldsymbol{r}}(\tilde{\boldsymbol{f}}^{[j+1]}) \tag{4.46}$$

with rank $\boldsymbol{r}$ chosen so that

$$\left\| \boldsymbol{f}^{[j+1]} - \tilde{\boldsymbol{f}}^{[j+1]} \right\| < \frac{\left\| \boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right) \right\|}{2\left\| \boldsymbol{J}_{\boldsymbol{H}}\left(\boldsymbol{f}^{[j]}\right) \right\|} \eta^{[j]}. \tag{4.47}$$

Then the error in solving the Newton iteration this step is

$$\tilde{\boldsymbol{r}}^{[j]} = \boldsymbol{J}_{\boldsymbol{H}}\left(\boldsymbol{f}^{[j]}\right)\left[\boldsymbol{f}^{[j+1]} - \tilde{\boldsymbol{f}}^{[j+1]} + \boldsymbol{s}^{j}\right] + \boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right), \tag{4.48}$$

which we constructed to satisfy the bound

$$\begin{aligned} \left\| \tilde{\boldsymbol{r}}^{[j]} \right\| &\leq \left\| \boldsymbol{J}_{\boldsymbol{H}}\left(\boldsymbol{f}^{[j]}\right)\boldsymbol{s}^{[j]} + \boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right) \right\| + \left\| \boldsymbol{J}_{\boldsymbol{H}}\left(\boldsymbol{f}^{[j]}\right)\left[\boldsymbol{f}^{[j+1]} - \tilde{\boldsymbol{f}}^{[j+1]}\right] \right\| \\ &< \left\| \boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right) \right\| \eta^{[j]}. \end{aligned}$$

Hence, the inexact HT/TT-GMRES Newton method converges linearly. This completes the proof. □

An Example

Let us provide a brief numerical demonstration of the inexact Newton method with HT/TT-GMRES iteration. To this end, consider the cubic function

$$\boldsymbol{H}(\boldsymbol{f}) = 1.5\boldsymbol{f} + 0.5\boldsymbol{f}_0 + 0.125(\boldsymbol{f} + \boldsymbol{f}_0)^3 \tag{4.49}$$

where all products are computed using the approximate element-wise Hadamard tensor product with accuracies set to 10^{-12}, and $\boldsymbol{f}_0$ is a given tensor which cor-

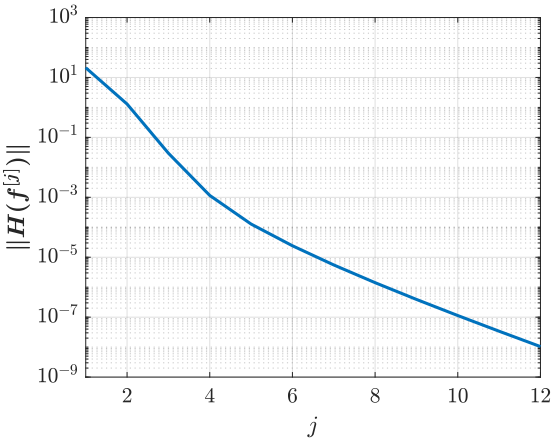


Figure 4.1: Error versus iteration count of inexact Newton's method in the HT format.

responds to the initial condition used in Section 5.3.1 truncated to an absolute tolerance of 10^{-4}. The Jacobian operator is easily obtained as

$$\boldsymbol{J}_{\boldsymbol{H}}(\boldsymbol{f})\boldsymbol{s} = 1.5\boldsymbol{s} + 0.375(\boldsymbol{f} + \boldsymbol{f}_0)^2\boldsymbol{s}. \tag{4.50}$$

We set the relative error of the matrix inverse to be $\eta = 10^{-3}$. In Figure 4.1 we plot the results of the proposed inexact Newton method with HT/TT-GMRES iterations. We see that the target tolerance of 2.2×10^{-8} is hit in just 12 iterations.

4.4.2 The Compression Step

While increasing the tensor rank may be necessary for convergence of the HT/TT-GMRES iterations, it is possible that we raise the rank by more than is required for the desired order of accuracy in a single time step. Therefore, it is convenient to apply an additional tensor truncation after computing, say, j steps of the HT inexact Newton's method which returns $\boldsymbol{f}_{k+1}^{[j]}$. This is the same as the "compression step" at the end of HT/TT-GMRES algorithm as presented in [40]. This

gives us our final estimate of $\boldsymbol{f}((k+1)\Delta t)$ as

$$\boldsymbol{f}_{k+1} = \mathfrak{T}_{\boldsymbol{r}}\left(\boldsymbol{f}_{k+1}^{[j]}\right), \qquad \left\|\mathfrak{T}_{\boldsymbol{r}}\left(\boldsymbol{f}_{k+1}^{[j]}\right) - \boldsymbol{f}_{k+1}^{[j]}\right\| \leq e_{\boldsymbol{r}}. \tag{4.51}$$

Regarding the selection of the truncation error $e_{\boldsymbol{r}}$ we proceed as follows. Suppose that $\boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k)$ is an explicit integration scheme of the same order (or higher) than the implicit scheme being considered. Then we can estimate the local error as

$$\begin{aligned}\left\|\boldsymbol{f}_{k+1}^{[j]} - \boldsymbol{f}((k+1)\Delta t)\right\| &\leq \left\|\boldsymbol{f}_{k+1}^{[j]} - \boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k)\right\| + \left\|\boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k) - \boldsymbol{f}((k+1)\Delta t)\right\| \\ &= \left\|\boldsymbol{f}_1^{[j]} - \boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k)\right\| + O(\Delta t^{p+1}),\end{aligned} \tag{4.52}$$

Thus, we may roughly estimate the local truncation error and set this as the chosen error for approximation to HT or TT rank $\boldsymbol{r}$ using

$$e_{\boldsymbol{r}} = \left\|\boldsymbol{f}_{k+1}^{[j]} - \boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k)\right\|. \tag{4.53}$$

We may drop more singular values than needed, especially if the choice of Δt is outside the region of stability of the explicit scheme $\boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k)$. However, this estimate guarantees that we do not change the overall convergence rate. Moreover, it cannot impact stability of the implicit step-truncation integrator since the compression step has operator norm equal to one, regardless of the rank chosen. (Recall Section 4.1.) In our numerical experiments we use the explicit step-truncation midpoint method to estimate local error, i.e., $\boldsymbol{\Psi}_{\Delta t}(\boldsymbol{G}, \boldsymbol{f}_k)$ in (4.53) is set as in (4.18).

4.5 Stability for Implicit Step-Truncation

We now address absolute stability of the proposed implicit step-truncation methods. The study on absolute stability allows us to determine whether the time stepping scheme is robust to perturbations due to finite Δt and finite tensor truncation tolerance. As is well-known, absolute stability analysis of classical time integration schemes allows us to claim stability over a large number of iterations for finite Δt and for specific linear prototype problems. For step-truncation integration on tensor manifolds, we have additional perturbations due to the tensor truncation induced by nonzero HT/TT-GMRES stopping tolerance. To study absolute stability of implicit step-truncation schemes for finite Δt and truncation tolerances, consider the following linear initial value problem

$$\frac{d\boldsymbol{f}}{dt} = \boldsymbol{L}\boldsymbol{f}, \quad \boldsymbol{f}(0) = \boldsymbol{f}_0 \tag{4.54}$$

where $\boldsymbol{L}$ is a linear operator with eigenvalues in the left half complex plane. After applying any standard implicit time stepping scheme, we end up with a system of linear equations of the form

$$\boldsymbol{A}\boldsymbol{f}_{k+1} = \boldsymbol{W}\boldsymbol{f}_k. \tag{4.55}$$

Specifically, for the implicit Euler we have $\boldsymbol{A} = \boldsymbol{I} - \Delta t\boldsymbol{L}$, $\boldsymbol{W} = \boldsymbol{I}$ while for the implicit midpoint method we have $\boldsymbol{A} = \boldsymbol{I} - 0.5\Delta t\boldsymbol{L}$, $\boldsymbol{W} = \boldsymbol{I} + 0.5\Delta t\boldsymbol{L}$. As is well known, both implicit Euler and implicit midpoint are unconditionally stable, in the sense that for any $\Delta t > 0$, $\|\boldsymbol{f}_k\| \to 0$ as $k \to \infty$. One way of proving this is by noting that whenever the eigenvalues of $\boldsymbol{L}$ have negative real part, we get

$$\|\boldsymbol{A}^{-1}\boldsymbol{W}\| < 1, \tag{4.56}$$

and therefore the mapping $\boldsymbol{A}^{-1}\boldsymbol{W}$ is contractive. This implies the sequence of $\boldsymbol{f}_k$ defined by (4.55) converges to zero. The following theorem characterizes what happens when we exchange exact matrix inverse $\boldsymbol{A}^{-1}$ with an inexact inverse computed by tensor HT/TT-GMRES iterations.

Theorem 6 (Absolute stability of implicit step-truncation methods)**.** *Consider an implicit scheme of the form* (4.55)*, and suppose that* $\|\boldsymbol{A}^{-1}\boldsymbol{W}\| < 1$*. Denote by* $\hat{\boldsymbol{f}}_k$ *the solution of* $\boldsymbol{A}\hat{\boldsymbol{f}}_k = \boldsymbol{W}\hat{\boldsymbol{f}}_{k-1}$ *(*$k = 1, 2, \ldots$*) obtained with the HT/TT-GMRES tensor solver described in Section 4.4.1, with* m *Krylov iterations and stopping tolerance* η*. If*

$$\left\|\boldsymbol{A}\hat{\boldsymbol{f}}_k - \boldsymbol{W}\hat{\boldsymbol{f}}_{k-1}\right\| \leq m\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|\eta, \tag{4.57}$$

then the distance between $\hat{\boldsymbol{f}}_k$ *and the exact solution* $\boldsymbol{f}_k = \boldsymbol{A}^{-1}\boldsymbol{W}\boldsymbol{f}_{k-1}$ *can be bounded as*

$$\left\|\hat{\boldsymbol{f}}_k - \boldsymbol{f}_k\right\| \leq \frac{m\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|^2}{1 - \|\boldsymbol{A}^{-1}\boldsymbol{W}\|}\eta. \tag{4.58}$$

Note that (4.58) implies that $\left\|\hat{\boldsymbol{f}}_k\right\| = O(\eta)$ as $k \to \infty$. In the context of HT/TT-GMRES iterations, the number η can be controlled by setting the stopping tolerance in Lemma 4 (Section 4.4.1) as

$$\varepsilon_k = \frac{\eta}{\left\|\boldsymbol{W}\hat{\boldsymbol{f}}_k\right\|} \tag{4.59}$$

at each time step k.

Proof. The proof follows from a straightforward inductive argument. For $k = 1$ we have

$$\begin{aligned}\hat{\boldsymbol{f}}_1 - \boldsymbol{f}_1 &= \boldsymbol{A}^{-1}\boldsymbol{A}\hat{\boldsymbol{f}}_1 - \boldsymbol{A}^{-1}\boldsymbol{W}\boldsymbol{f}_0 \\ &= \boldsymbol{A}^{-1}\left(\boldsymbol{A}\hat{\boldsymbol{f}}_1 - \boldsymbol{W}\boldsymbol{f}_0\right). \end{aligned} \tag{4.60}$$

By using (4.57), we can bound $\left\|\hat{\boldsymbol{f}}_1 - \boldsymbol{f}_1\right\|$ as

$$\left\|\hat{\boldsymbol{f}}_1 - \boldsymbol{f}_1\right\| \leq m\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|^2\eta. \tag{4.61}$$

For $k = 2$ we have

$$\begin{aligned}\left\|\hat{\boldsymbol{f}}_2 - \boldsymbol{f}_2\right\| &\leq \left\|\hat{\boldsymbol{f}}_2 - \boldsymbol{A}^{-1}\boldsymbol{W}\hat{\boldsymbol{f}}_1\right\| + \left\|\boldsymbol{A}^{-1}\boldsymbol{W}\hat{\boldsymbol{f}}_1 - \boldsymbol{f}_2\right\| \\ &= \left\|\hat{\boldsymbol{f}}_2 - \boldsymbol{A}^{-1}\boldsymbol{W}\hat{\boldsymbol{f}}_1\right\| + \left\|\boldsymbol{A}^{-1}\boldsymbol{W}\hat{\boldsymbol{f}}_1 - \boldsymbol{A}^{-1}\boldsymbol{W}\boldsymbol{f}_1\right\| \\ &\leq m\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|^2\eta + \|\boldsymbol{A}^{-1}\boldsymbol{W}\|\left\|\hat{\boldsymbol{f}}_1 - \boldsymbol{f}_1\right\| \\ &\leq \left(1 + \|\boldsymbol{A}^{-1}\boldsymbol{W}\|\right)m\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|^2\eta, \end{aligned}$$

where the last inequality follows from (4.61). More generally, let the inductive hypothesis be

$$\left\|\hat{\boldsymbol{f}}_{k-1} - \boldsymbol{f}_{k-1}\right\| \leq m\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|^2\eta\sum_{j=0}^{k-2}\|\boldsymbol{A}^{-1}\boldsymbol{W}\|^j. \tag{4.62}$$

Repeating the string of inequalities above and replacing the right sum with the inductive hypothesis, we obtain

$$\left\|\hat{\boldsymbol{f}}_k - \boldsymbol{f}_k\right\| \leq m\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|^2\eta\sum_{j=0}^{k-1}\|\boldsymbol{A}^{-1}\boldsymbol{W}\|^j \leq \frac{m\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|^2}{1 - \|\boldsymbol{A}^{-1}\boldsymbol{W}\|}\eta, \tag{4.63}$$

which completes the proof. □

The stability region of an explicit step-truncation method is the same as the corresponding method without truncation [93, 92]. Theorem 6 shows that the stability region of an implicit step-truncation method is identical to the corresponding method without truncation, though by relaxing accuracy we see that instead our iterates decay to within the solver tolerance of zero rather than converging to zero in an infinite time horizon. In other words, an implicit step-truncation method will be *unconditionally stable*. However, if the tolerance of HT/TT-GMRES is set too large, then we could see poor stability behavior akin to an explicit method.

Chapter 5

Numerical Applications

In this section, we showcase the algorithms outlined above. We exhibit the differing behavior of fixed rank and adaptive rank methods, as well as the differing performance of the implicit and explicit methods.

5.1 Fixed Rank Step-Truncation Methods

We present stability preservation for fixed rank explicit step truncation methods. These examples do not attempt to truncate to preserve consistency, but do satisfy Theorem 2.

5.1.1 Fixed Rank Adams-Bashforth

In this section we provide demonstrative examples of fixed rank truncated tensor methods applied to variable-coefficient advection-diffusion PDEs of the form

$$\frac{\partial}{\partial t}u(t,\boldsymbol{x}) = -\sum_{k=1}^{d}\frac{\partial}{\partial x_k}\Big(f_k(\boldsymbol{x})u(t,\boldsymbol{x})\Big) + \sum_{k=1}^{d}\sum_{q=1}^{d}\frac{\partial^2}{\partial x_k \partial x_q}(\Gamma_{kq}(\boldsymbol{x})u(t,\boldsymbol{x})), \qquad (5.1)$$

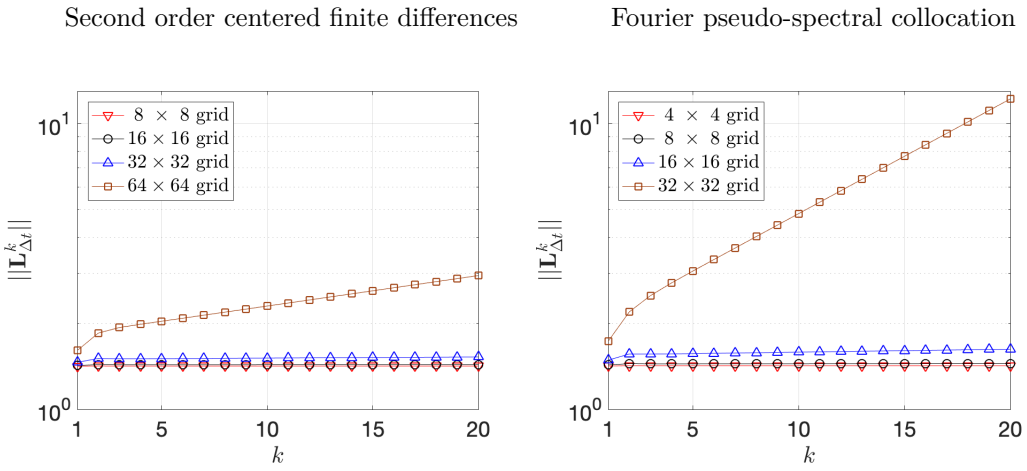


Figure 5.1: Two-dimensional PDE (5.2). Operator norm of $\boldsymbol{L}^k_{\Delta t}$ (see Eq. (5.3)) versus k for two conditionally stable schemes, namely the second order centered finite-differences and the Fourier pseudo-spectral collocation schemes on a grid with $n \times n$ evenly-spaced points in $[0, 2\pi]^2$, with $n = 4, 8, 16, 32, 64$. It is seen that that the Fourier pseudo-spectral method is less stable than the finite-difference method, in agreement with well-known results [51]. Here we set $\Delta t = 0.0025$.

where $\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x})\ f_2(\boldsymbol{x})\ \ldots\ f_d(\boldsymbol{x})]^T$ is the drift vector field and $\boldsymbol{\Gamma}(\boldsymbol{x}) = [\Gamma_{kq}(\boldsymbol{x})]$ is the symmetric positive definite diffusion matrix. As is well-known, the PDE (5.1) governs the evolution of the probability density function corresponding to an ODE driven by multiplicative white noise [90]. We approximate the solution of (5.1) in the spacial domain $\Omega = [0, 2\pi]^d$ with periodic boundary conditions. In particular, we look at the growth of matrix rank in the case of a 2D hyperbolic PDE. Additionally, we provide examples of Theorem 2 in the case of higher-dimensional advection-diffusion PDEs. The C++/MPI hierarchical Tucker code we developed to study these examples is available at [91].

Two Dimensional Hyperbolic PDE

Let us consider the two-dimensional hyperbolic PDE

$$\frac{\partial}{\partial t}u(t, \boldsymbol{x}) = -\frac{\partial}{\partial x_1}\Big(\sin(x_2) \cdot u(t, \boldsymbol{x})\Big) - \frac{\partial}{\partial x_2}\Big(\cos(x_1) \cdot u(t, \boldsymbol{x})\Big). \tag{5.2}$$

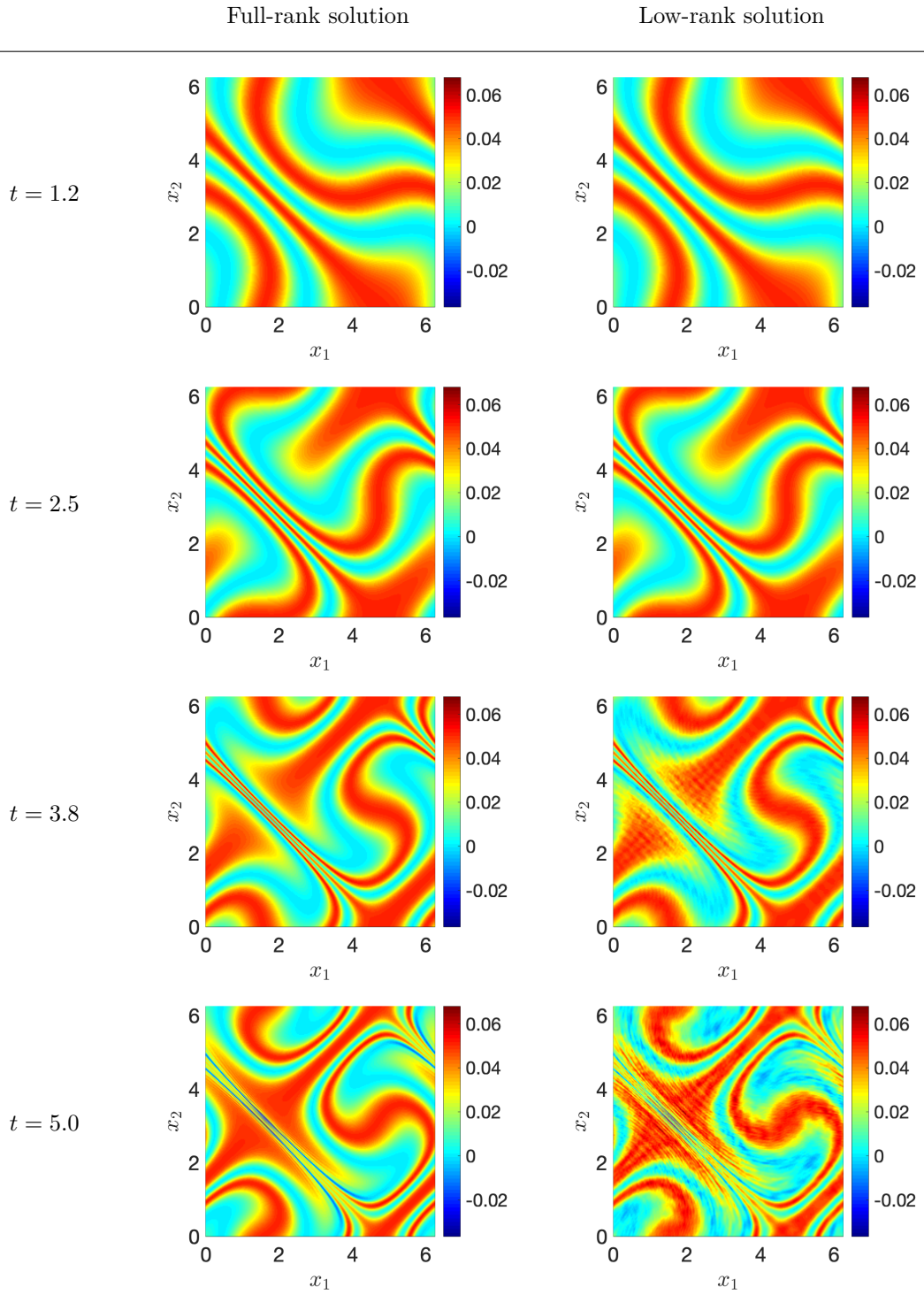


Figure 5.2: Numerical solution of the PDE (5.2) using a Fourier pseudo-spectral method on a grid with 256×256 nodes. The initial condition is chosen as $u_0(x_1, x_2) = \sin^2(x_1 + x_2)/(2\pi^2)$. Shown are the full-rank solution (left) and the fixed-rank tensor solution (right) we obtained by limiting the maximum rank to 64. It is seen that the two solutions diverge by $t = 3.8$, but stability is maintained as proven in Theorem 2.

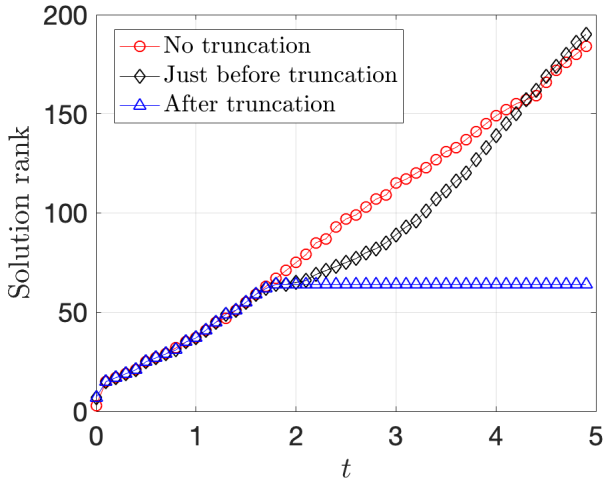


Figure 5.3: Tensor rank of the numerical solution to the PDE (5.2) versus time. The spatial derivatives are discretized using a Fourier pseudo-spectral method on a grid with 256×256 nodes. Hence the maximum rank of the solution tensor is 256. The rank-limited solution has maximum rank set to 64. Note that just before applying the truncation operator in a rank-limited scheme, the rank of the iterate appears to grow at a similar rate to the scheme with no truncation. The inaccuracies of the rank-truncated solutions shown in Figure 5.2 at $t = 3.8$ and $t = 5$ are due to the fact that the solution rank is much larger than 64 at such times (compare red and blue curves).

We discretize (5.2) in space on an evenly spaced grid with $n \times n$ points in $[0, 2\pi]^2$. Specifically we will consider both Fourier spectral methods and second order finite-differences discretization. In the two-dimensional setting we consider here, the semi-discrete form (1.2) involves two-dimensional arrays, i.e. matrices. Hence, hierarchical rank is the same as matrix rank in this case, since the rank of a matrix and its transpose coincide. Applying the two-step Adams-Bashforth method yields a linear recurrence relation of the form (4.10), with

$$\boldsymbol{L}_{\Delta t} = \begin{bmatrix} -\frac{\Delta t}{2}\boldsymbol{G} + \boldsymbol{I} & \frac{3}{2}\Delta t \boldsymbol{G} \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}, \tag{5.3}$$

and

$$\boldsymbol{G} = -\left(\boldsymbol{D} \otimes \boldsymbol{I}\right) \mathrm{diag}[\sin(\boldsymbol{x}_2)] + \left(\boldsymbol{I} \otimes \boldsymbol{D}\right) \mathrm{diag}[\cos(\boldsymbol{x}_1)]. \tag{5.4}$$

Here, $\sin(\boldsymbol{x}_2)$ and $\cos(\boldsymbol{x}_1)$ are vectorizations of $\sin(x_2)$ and $\cos(x_1)$ evaluated on an evenly-spaced 2D spacial grid, and $\boldsymbol{D}$ is the first order (one-dimensional) differentiation matrix.

In Figure 5.1 we plot the typical behavior of the operator norm $\left\|\boldsymbol{L}_{\Delta t}^{k}\right\|$ versus k for two conditionally stable schemes, namely the Fourier pseudo-spectral and the second order centered finite-difference schemes on a $n \times n$ grid, with $n = 4, 8, 16, 32, 64$. It is seen that $\left\|\boldsymbol{L}_{\Delta t}^{k}\right\|$ grows as k^2, in the case of the Fourier pseudo-spectral method, making it considerably less stable than the finite-difference method, in agreement with well-known results [51].

In Figure 5.2 we plot the numerical solution of (5.2) we obtained with an accurate Fourier spectral method. The initial condition is chosen as $u_0(x_1, x_2) = \sin^2(x_1 + x_2)/(2\pi^2)$. It is seen that the low-rank tensor solution obtained by capping the maximum rank to 64 slightly differs from the full rank solution at $t = 3.8$ and $t = 5$. However, stability is maintained as proven in Theorem 2. In Figure 5.3 we show that the solution rank grows in time. Such growth is determined by the fact that that solution to the hyperbolic problem (5.2) becomes harder to resolve as time increases (see Figure 5.2). In particular, in Figure 5.3 we see that just before applying the truncation operator, the rank of the iterate appears to grow at a similar rate to the tensor scheme with no truncation.

Six Dimensional Parabolic PDE

We now demonstrate a higher dimensional example which is both highly diffusive and very well approximated by a low-rank numerical solution tensor. To this end, we consider the Fokker-Planck equation (5.1) with $d = 6$. For our numerical

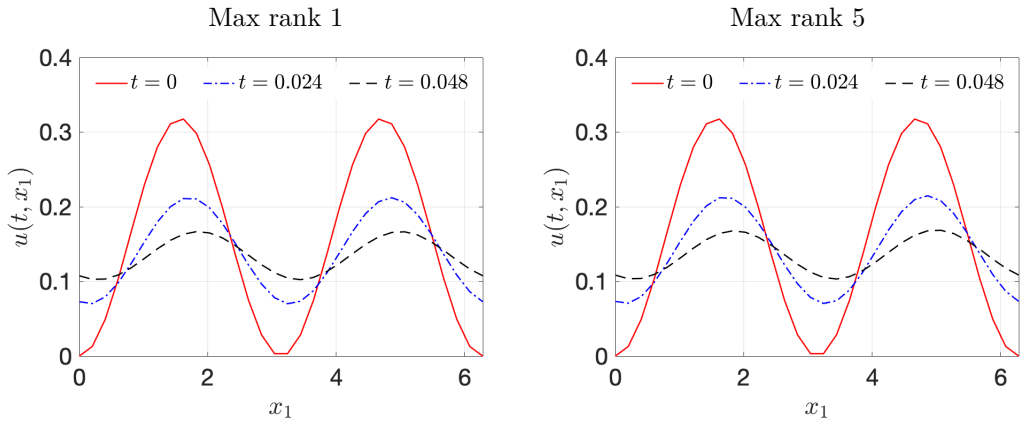


Figure 5.4: Six-dimensional Fokker-Plank equation (5.1). Temporal snapshots of the marginal PDF $u(t, x_1)$. It is seen that the system is highly diffusive and it yields a solution that can be well approximated by a low-rank hierarchical Tucker tensor format.

demonstration, we consider the following drift and diffusion coefficients

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} \cos(x_2) & \sin(x_3) & \cos(2x_4) & \sin(2x_5) & \cos(3x_6) & \sin(3x_1) \end{bmatrix}^\top + 6 \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T,$$

$$\boldsymbol{\Gamma}(\boldsymbol{x}) = \begin{bmatrix} 5\cos^2(x_6) & \sin(x_1) & \cos(x_2) & \sin(x_3) & \cos(x_4) & \sin(x_5) \\ \sin(x_1) & 5\cos^2(3x_3) & \sin(5x_2) & \cos(2x_1) & \sin(4x_6) & \cos(x_1) \\ \cos(x_2) & \sin(5x_2) & 5\cos^2(3x_5) & \sin(2x_3) & \cos(6x_2) & \sin(x_6) \\ \sin(x_3) & \cos(2x_1) & \sin(2x_3) & 5\cos^2(x_3) & \sin(x_1) & \cos(4x_4) \\ \cos(x_4) & \sin(4x_6) & \cos(6x_2) & \sin(x_1) & 5\cos^2(5x_5) & \sin(x_6) \\ \sin(x_5) & \cos(x_1) & \sin(x_6) & \cos(4x_4) & \sin(x_6) & 5\cos^2(7x_6) \end{bmatrix} + 6\boldsymbol{I}.$$

The matrix of drift coefficients was chosen to encourage mixing between different variables and the diffusion coefficients were chosen so that the symmetric matrix

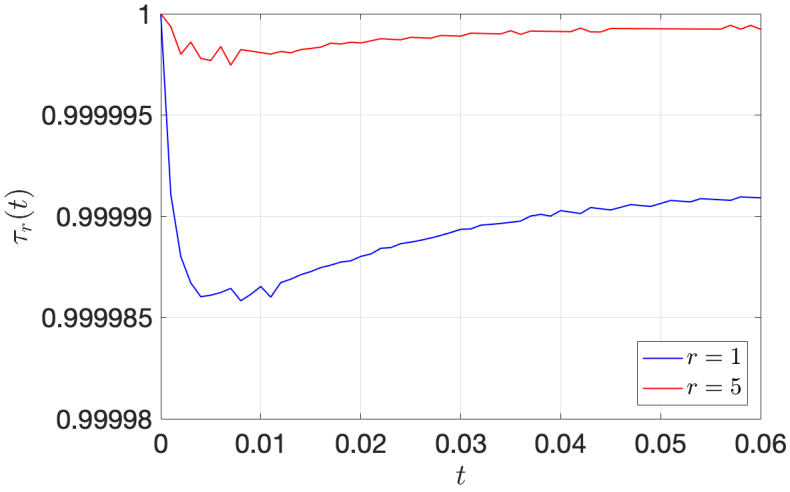


Figure 5.5: In Lemma 2, we proved that the truncation operator $\mathfrak{T}_r$ satisfies the inequality $||\mathfrak{T}_r(\boldsymbol{u})||_2 \leq ||\boldsymbol{u}||_2$. In this figure we plot of the ratio $\tau_r(t) = ||\mathfrak{T}_r(\boldsymbol{u})||_2/||\boldsymbol{u}||_2$ versus time. We see that setting max rank equal to 5 does in fact give us an extra single digit of accuracy in the nonlinear rank projection. However,$\tau_1(t)$ and $\tau_5(t)$ are very close to 1, which explains why the two plots in Figure 5.4 are visually identical.

is diagonally dominant. Therefore, $\boldsymbol{\Gamma}$ will always be positive definite ensuring that (5.1) is a bounded diffusion problem. We discretize (5.1) in space using the Fourier pseudo-spectral collocation method as in 5.1.1. Specifically, we construct an evenly-spaced grid in $[0, 2\pi]^6$ with 31 points in each variable. In principle this yields 31^6 degrees of freedom, which require 110 MB (Mega Bytes) of storage if a tensor product representation in double precision floating point arithmetic is utilized. However, if we employ a rank $\boldsymbol{r}$ hierarchical Tucker tensor format the memory footprint is reduced to $[r(31 \times 6) + 4r^3 + r^2]/8$ Bytes. For instance, a rank 40 hierarchical Tucker tensor format in 6 dimensions on a grid with 31 points in each variable requires only 33 kB (kilo Bytes) of storage. By using the identity $\partial^2/\partial x_k \partial x_q = \partial^2/\partial x_q \partial x_k$, we see that we can just apply the strict upper triangle part of $\boldsymbol{\Gamma}$ once and then double the result. The matrix $\boldsymbol{G}$ at right-hand side of

(1.2) in this case has a rather involved expression and therefore it is not reported here. We supplement (5.1) with the initial condition

$$u(0, \boldsymbol{x}) = \frac{1}{\pi^6} \prod_{j=1}^{6} \sin(x_j)^2. \tag{5.5}$$

Note that (5.5) is positive and it integrates to one over the hyper-cube $[0, 2\pi]^6$, i.e., it is a probability density function. In Figure 5.4 we plot the temporal evolution of the marginal PDF

$$u(t, x_1) = \int_{[0,2\pi]^5} u(t, \boldsymbol{x}) dx_2 \cdots dx_6 \tag{5.6}$$

It is seen that the PDE (5.1) is highly diffusive and it yields a solution that can be well approximated by a low-rank hierarchical Tucker tensor format. In Figure 5.5 we provide a numerical verification of our Lemma 2. To this end, we plot the ratio $\tau_r(t) = ||\mathfrak{T}_r(\boldsymbol{u})||_2/||\boldsymbol{u}||_2$ versus time, and verify that is always smaller than one for any choice of rank. Note that both $\tau_1(t)$ and $\tau_5(t)$ are very close to 1, which explains why the two plots in Figure 5.4 are visually identical. In our simulations we found that the hierarchical ranks of the HT tensor solution do not grow monotonically as in the two-dimensional advection problem (see Figure 5.3). Instead, they reach a peak very quickly in time. This is because the solution is well approximated by a rank one tensor (see Figure 5.5).

5.2 Explicit Rank-Adaptive Methods

In this section we present and discuss numerical applications of the proposed explicit rank-adaptive step-truncation methods. We have seen that these methods are defined by parameters summarized in Table 5.1. To choose such parameters in

Integration Method	Free Parameters	Dependent Parameters
Rank-adaptive Euler (Sec. 3.1)	$\Delta t, M_1, M_2$	$\varepsilon_{\boldsymbol{r}} = M_1 \Delta t^2$, $\varepsilon_{\boldsymbol{s}} = M_2 \Delta t$
Rank-adaptive midpoint (Sec. 3.2)	$\Delta t, A, B, G$	$\varepsilon_{\boldsymbol{\alpha}} = A \Delta t^3$, $\varepsilon_{\boldsymbol{\beta}} = B \Delta t^2$, $\varepsilon_{\boldsymbol{\gamma}} = G \Delta t$
Two-step rank-adaptive Adams-Bashforth (Sec. 3.3)	$\Delta t, A, B, G_0, G_1$	$\varepsilon_{\boldsymbol{\alpha}} = A \Delta t^3$, $\varepsilon_{\boldsymbol{\beta}} = B \Delta t^2$, $\varepsilon_{\boldsymbol{\gamma}(0)} = G_0 \Delta t^2$, $\varepsilon_{\boldsymbol{\gamma}(1)} = G_1 \Delta t^2$

Table 5.1: Free and dependent parameters of the explicit rank-adaptive step-truncation integrators presented in Chapter 3.

each numerical example we proceed as follows: We first choose the time step Δt so that the scheme without tensor truncation is stable. Theorem 3 then guarantees convergence of the rank-adaptive step-truncation scheme for any selection of the other parameters, e.g., M_1 and M_2 in the rank-adaptive Euler scheme listed in Table 5.1. For guidance on how to select the remaining parameters one may apply the results of Section 3.6, which are based on the knowledge of the singular values of the solution. An alternative heuristic criterion is to select the free parameters roughly inverse to the time step so that the local error parameters, e.g., $\varepsilon_{\boldsymbol{r}}$ and $\varepsilon_{\boldsymbol{s}}$ in Table 5.1, do not exceed a specified threshold ε^*, i.e., $\varepsilon_{\boldsymbol{r}} \leq \varepsilon^*$ and $\varepsilon_{\boldsymbol{s}} \leq \varepsilon^*$.

5.2.1 Rank shock problem

In this section we test ability of the proposed rank-adaptive schemes to track accuracy and rank for a problem where the rank of the vector field suddenly jumps to a higher value. To this end, consider the following matrix-valued ordinary

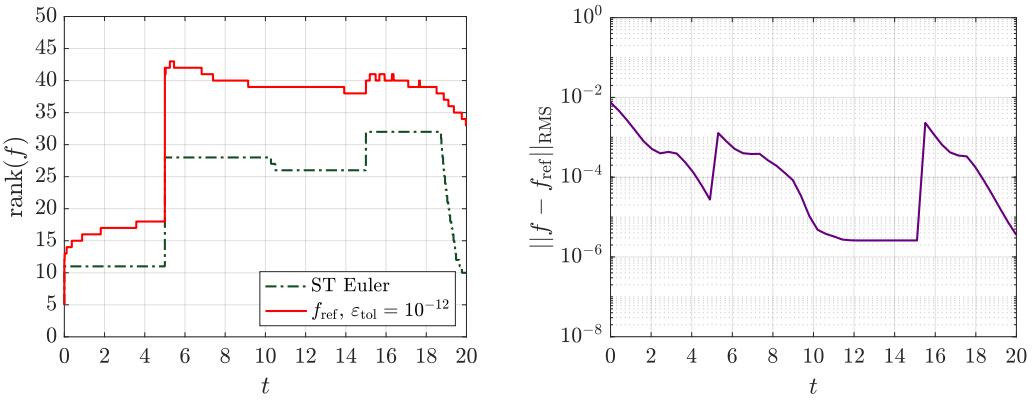


Figure 5.6: Rank shock problem. Numerical performance of rank-adaptive Euler method applied to the ODE (5.7)-(5.8). It is seen that the method accurately tracks the overall shape of the reference solution rank, which was computed to a singular value threshold of 10^{-12}. Moreover, the numerical error behaves as expected, decreasing as steady state is approached.

differential equation

$$\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}t} = \boldsymbol{A}\boldsymbol{f} + \boldsymbol{f}\boldsymbol{A}^\top + \boldsymbol{v}(t), \quad t \in [0, 20], \quad \boldsymbol{f}(0) \in \mathbb{R}^{N\times N}, \tag{5.7}$$

where $\boldsymbol{A}$ is a symmetric negative definite matrix and $\boldsymbol{v}(t)$ a forcing term that switches between a low rank and high rank matrix

$$\boldsymbol{v}(t) = \begin{cases} \boldsymbol{v}_{\text{high}}, & t \in (5, 15) \\ \boldsymbol{v}_{\text{low}}, & t \notin (5, 15) \end{cases}. \tag{5.8}$$

In equation (5.7) $\boldsymbol{A}\boldsymbol{f}+\boldsymbol{f}\boldsymbol{A}^\top$ is a stabilizing term which is tangent to the fixed rank manifold at all time while $\boldsymbol{v}(t)$ steers the solution off of the fixed rank manifold.

Integration Method	Free Parameters	Dependent Parameters
Rank-adaptive Euler (Sec. 3.1)	$\Delta t = 2 \times 10^{-3}$, $M_1 = M_2 = 10^2$	$\varepsilon_{\boldsymbol{r}} = 4 \times 10^{-4}$, $\varepsilon_{\boldsymbol{s}} = 2 \times 10$

Table 5.2: Integration parameters for the rank shock problem (5.7).

For our numerical experiment we let $\boldsymbol{A}$ take the form

$$\boldsymbol{A} = \begin{bmatrix} -b & a & 0 & 0 & \dots & 0 \\ a & -b & a & 0 & \dots & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & \dots & 0 & a & -b & a \\ 0 & \dots & 0 & 0 & a & -b \end{bmatrix} \qquad a, b \in \mathbb{R}, \tag{5.9}$$

which is a finite difference stencil with shifted eigenvalues. We set $a = 1$ and $b = 3$ to ensure the matrix $\boldsymbol{A}$ is diagonally dominant with negative eigenvalues. This guarantees that the initial value problem (5.7) will be stable regardless of how large the $N \times N$ matrix size is; for our demonstration we set $N = 100$. For the forcing term $\boldsymbol{v}(t)$ we set

$$\boldsymbol{v}_{\text{low}} = \sum_{j=1}^{r_{\text{low}}} \boldsymbol{\phi}_j \boldsymbol{\psi}_j^\top, \qquad \boldsymbol{v}_{\text{high}} = \sum_{j=1}^{r_{\text{high}}} \sigma^j \boldsymbol{\psi}_j \boldsymbol{\phi}_j^\top, \tag{5.10}$$

with ranks $r_{\text{low}} = 6$ and $r_{\text{high}} = 25$. Here, $\boldsymbol{\psi}_j[i] = \sin(2\pi ij/N)$, $\boldsymbol{\phi}_j[i] = \cos(2\pi ij/N)$ and $\sigma^j = (3/4)^j$. Since the vector field is discontinuous in time, we apply the order 1 rank-adaptive Euler method with parameters summarized in Table 5.2. For quantification of the numerical error, we use the root mean square error of matrices

(Frobenious norm)

$$\|\boldsymbol{g} - \boldsymbol{f}\|_{\mathrm{RMS}} = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N} \frac{(\boldsymbol{g}[i,j] - \boldsymbol{f}[i,j])^2}{N^2}}. \tag{5.11}$$

To obtain a reference solution f_{ref} we simply integrate (5.7) using RK4. As seen in Figure 5.6, the numerical solution successfully tracks the overall shape of the reference solution's rank over time. The numerical error also behaves as expected, decreasing as a steady state is approached.

5.2.2 Fokker-Planck equation

In this section we apply the proposed rank-adaptive step-truncation algorithms to a Fokker-Planck equation with space-dependent drift and constant diffusion, and demonstrate their accuracy in predicting relaxation to statistical equilibrium. As is well-known, the Fokker-Planck equation describes the evolution of the probability density function (PDF) of the state vector solving the Itô stochastic differential equation (SDE)

$$d\boldsymbol{X}_t = \boldsymbol{\mu}(\boldsymbol{X}_t)dt + \sigma d\boldsymbol{W}_t. \tag{5.12}$$

Here, $\boldsymbol{X}_t$ is the d-dimensional state vector, $\boldsymbol{\mu}(\boldsymbol{X}_t)$ is the d-dimensional drift, σ is a constant drift coefficient and $\boldsymbol{W}_t$ is a d-dimensional standard Wiener process. The Fokker-Planck equation that corresponds to (5.12) has the form

$$\frac{\partial f(\boldsymbol{x},t)}{\partial t} = -\sum_{i=1}^{d} \frac{\partial}{\partial x_i}\left(\mu_i(\boldsymbol{x}) f(\boldsymbol{x},t)\right) + \frac{\sigma^2}{2}\sum_{i=1}^{d} \frac{\partial^2 f(\boldsymbol{x},t)}{\partial x_i^2}, \qquad f(\boldsymbol{x},0) = f_0(\boldsymbol{x}), \tag{5.13}$$

Integration Method	Free Parameters	Dependent Parameters
Adaptive Euler (Sec. 3.1)	$\Delta t = 6.25 \times 10^{-4}$, $M_1 = M_2 = 10^2$	$\varepsilon_{\boldsymbol{r}} = 3.90625 \times 10^{-5}$, $\varepsilon_{\boldsymbol{s}} = 6.25 \times 10^{-2}$
Adaptive Midpoint (Sec. 3.2)	$\Delta t = 6.25 \times 10^{-4}$, $A = B = 10^3$, $G = 10^2$	$\varepsilon_{\boldsymbol{\alpha}} = 2.44140625 \times 10^{-7}$, $\varepsilon_{\boldsymbol{\beta}} = 3.90625 \times 10^{-4}$, $\varepsilon_{\boldsymbol{\gamma}} = 6.25 \times 10^{-2}$
Two-step rank-adaptive Adams-Bashforth (HT Format) (Sec. 3.3)	$\Delta t = 6.25 \times 10^{-4}$, $A = B = 10^3$, $G_0 = G_1 = 10^2$	$\varepsilon_{\boldsymbol{\alpha}} = 2.44140625 \times 10^{-7}$, $\varepsilon_{\boldsymbol{\beta}} = 3.90625 \times 10^{-4}$, $\varepsilon_{\boldsymbol{\gamma}(0)} = 3.90625 \times 10^{-5}$, $\varepsilon_{\boldsymbol{\gamma}(1)} = 3.90625 \times 10^{-5}$,

Table 5.3: Table of parameters for the rank-adaptive step-truncation integrators of the Fokker-Planck equation (5.13) in dimension $d = 2$ with initial condition (5.15). The only free parameters are the local error coefficients. These were heuristically chosen so that the truncation at each step (to rank $\boldsymbol{r}$ or $\boldsymbol{\alpha}$) would be considerably smaller than the time step.

where $f_0(\boldsymbol{x})$ is the PDF of the initial state $\boldsymbol{X}_0$. In our numerical demonstrations, we set $\sigma = 2$,

$$\mu_i(\boldsymbol{x}) = (\gamma(x_{i+1}) - \gamma(x_{i-2}))\xi(x_{i-1}) - \phi(x_i), \qquad i = 1, \ldots, d, \tag{5.14}$$

where the functions $\gamma(x)$, $\xi(x)$, and $\phi(x)$ are 2π-periodic. Also, in (5.14) $x_{i+d} = x_i$. We solve (5.13) on the flat torus $\Omega = [0, 2\pi]^d$ with dimension $d = 2$ and $d = 4$.

5.2.3 Two-dimensional Fokker-Planck equation

Set $d = 2$ in (5.13) and consider the initial condition

$$f_0(x_1, x_2) = \frac{1}{m_0}\left[e^{\sin(x_1 - x_2)^2} + \sin(x_1 + x_2)^2\right], \tag{5.15}$$

where m_0 is a normalization factor. Discretize (5.15) on a two-dimensional grid of evenly-spaced points and then truncate the initial tensor (matrix) within machine accuracy into HT format. Also, set $\gamma(x) = \sin(x)$, $\xi(x) = \cos(x)$, and $\phi(x) =$

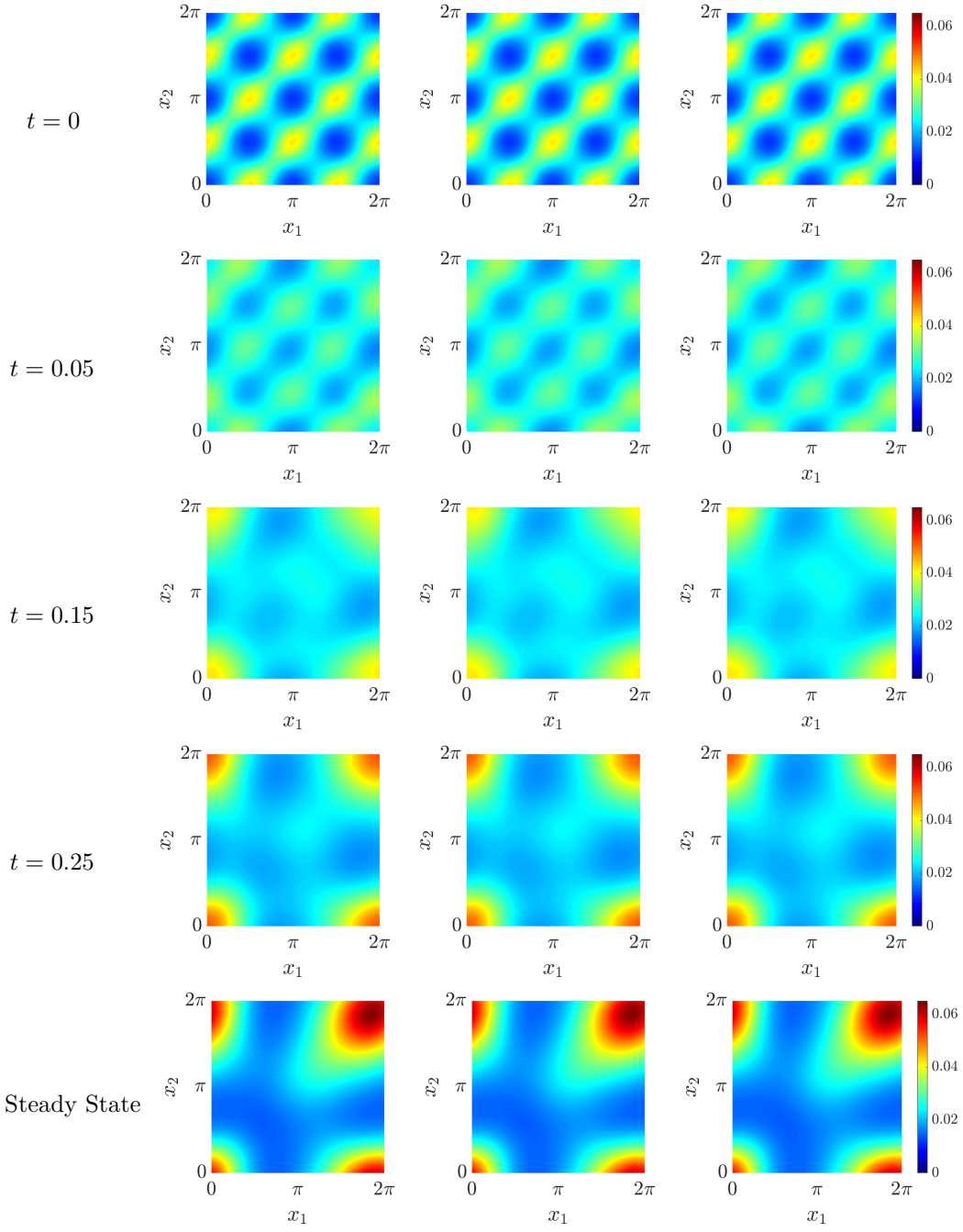


Figure 5.7: Numerical solution to the Fokker-Planck equation (5.13) in dimension $d = 2$ with initial condition (5.15) obtained using three distinct methods: rank-adaptive explicit Euler (3.5), two-step rank-adaptive Adams-Bashforth (3.15), and a reliable reference solution obtained by solving the ODE (1.2) corresponding to (5.13). The numerical results are obtained on a 50×50 spatial grid. The parameters for the step-truncation integrators we used in this example are detailed in Table 5.3.

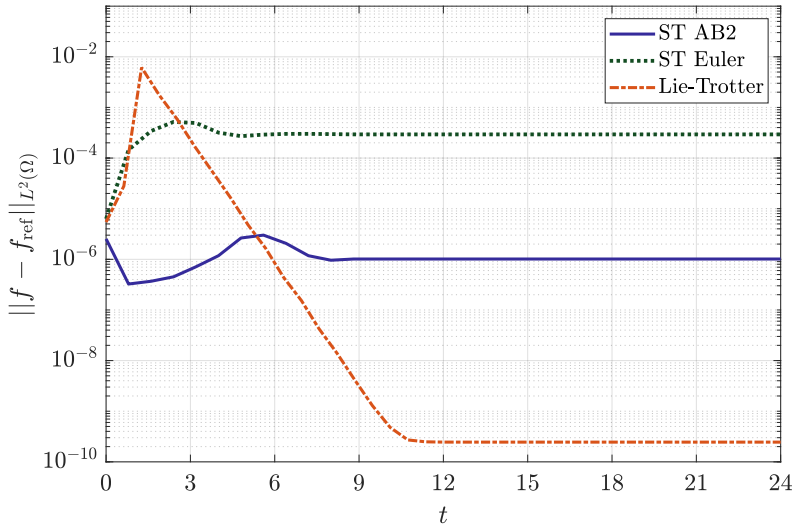


Figure 5.8: Fokker-Planck equation (5.13) in dimension $d = 2$ with initial condition (5.15). $L^2(\Omega)$ error of rank-adaptive Euler forward, rank-adaptive AB2, and rank-adaptive Lie-Trotter [31] (with normal vector threshold 10^{-4}) solutions with respect to the reference solution. The numerical results are obtained on a 50×50 spatial grid.

$\exp(\sin(x))+1$ for the drift functions in (5.13). In Figure 5.7, we plot the numerical solution of the Fokker-Planck equation (5.13) in dimension $d = 2$ corresponding to the initial condition (5.15). We computed our solutions with four different methods:

1. Rank-adaptive explicit Euler (3.5);
2. Two-step rank-adaptive Adams-Bashforth (AB) method (3.15);
3. Rank-adaptive tensor method with Lie-Trotter operator splitting[31];
4. RK4 method applied to the ODE (1.2) corresponding to a full tensor product discretization of (5.13). We denote this reference solution as f_{ref}.

The parameters we used for the rank-adaptive step-truncation methods 1. and 2. are summarized in Table 5.3. The steady state was determined for this compu-

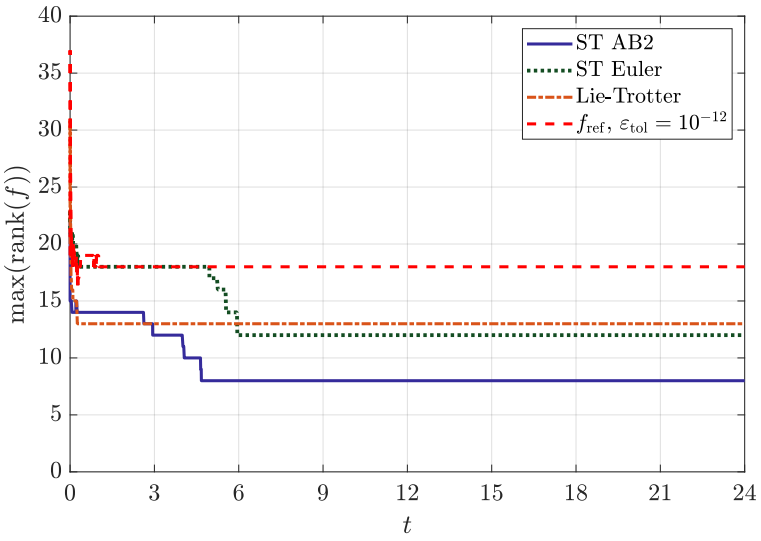


Figure 5.9: Fokker-Planck equation (5.13) in dimension $d = 2$ with initial condition (5.15). Rank versus time for rank-adaptive step-truncation Euler forward, AB2, rank-adaptive Lie-Trotter with normal vector threshold 10^{-4} [31], and reference numerical solutions. The numerical results are obtained on a 50×50 spatial grid. The reference solution rank was computed with a singular value tolerance of $\varepsilon_{\text{tol}}^{-12}$.

tation by halting execution when $\|\partial f_{\text{ref}}/\partial t\|_{L^2(\Omega)}$ was below the numerical threshold 10^{-13}. This occurs at approximately $t \approx 24$ for the initial condition (5.15). The numerical results in Figure 5.7 show that the step-truncation methods listed above match all visual behavior of the reference solution. Observing Figures 5.8 and 5.9, we note that while the rank-adaptive AB2 methods nearly double the digits of accuracy (in the $L^2(\Omega)$ norm), only a modest increase in rank is required to achieve this gain in accuracy. This is because the rank in each adaptive step-truncation scheme is determined by the increment function $\mathcal{A}$ (which defines the scheme), the nonlinear operator $\boldsymbol{G}$, and the truncation error threshold (which depends on Δt). More precisely, the closer $\mathcal{A}(\boldsymbol{G}, \boldsymbol{f}, \Delta t)$ is to the tangent space of the manifold $\mathcal{H}_{\boldsymbol{r}}$ at $\boldsymbol{f}_k$, the less the rank will increase in the next time step. In our demonstration, this occurs as the solution $\boldsymbol{f}_k$ approaches steady state, since, as

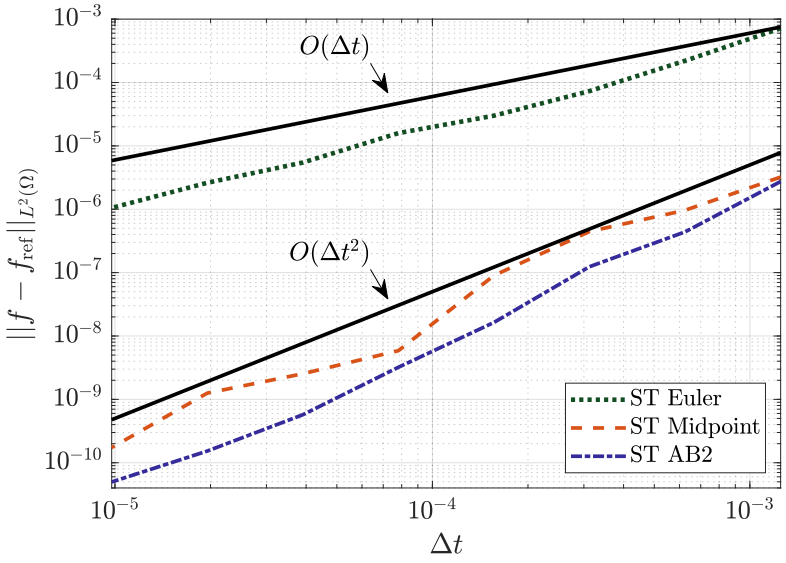


Figure 5.10: Fokker-Planck equation (5.13) in dimension $d = 2$ with initial condition (5.15). $L^2(\Omega)$ error at $T = 1$ for the rank-adaptive step-truncation methods summarized in Table 5.3. The numerical results are obtained on a 40×40 spatial grid.

the rate at which the probability density evolves in time slows down, the quantity $\|\mathcal{A}(\boldsymbol{G}, \boldsymbol{f}, \Delta t)\|_2$ tends to zero. Consequently, $\|(\boldsymbol{I} - \mathcal{P}_{\boldsymbol{g}})\mathcal{A}(\boldsymbol{G}, \boldsymbol{f}, \Delta t)\|_2$ will also tend towards zero since $\boldsymbol{I} - \mathcal{P}_{\boldsymbol{g}}$ is a bounded linear operator. For fixed Δt, this means that the rank increase conditions (4.26)-(4.28) will have a smaller likelihood of being triggered. As we shrink Δt, the truncation error requirements for consistency (4.26)-(4.28) become more demanding, and thus a higher solution rank is expected. In Figures 5.8 and 5.9 we also see that the rank-adaptive tensor method with Lie-Trotter integrator proposed in [31] performs better on this problem than rank-adaptive step-truncation methods, especially when the solution approaches the steady state. However, it should be noted that the rank-adaptive method with operator splitting and normal vector control is considerably more involved to implement than the step-truncation methods, which are essentially slight modifications of a standard single-step or multi-step method. In Figure 5.10 we

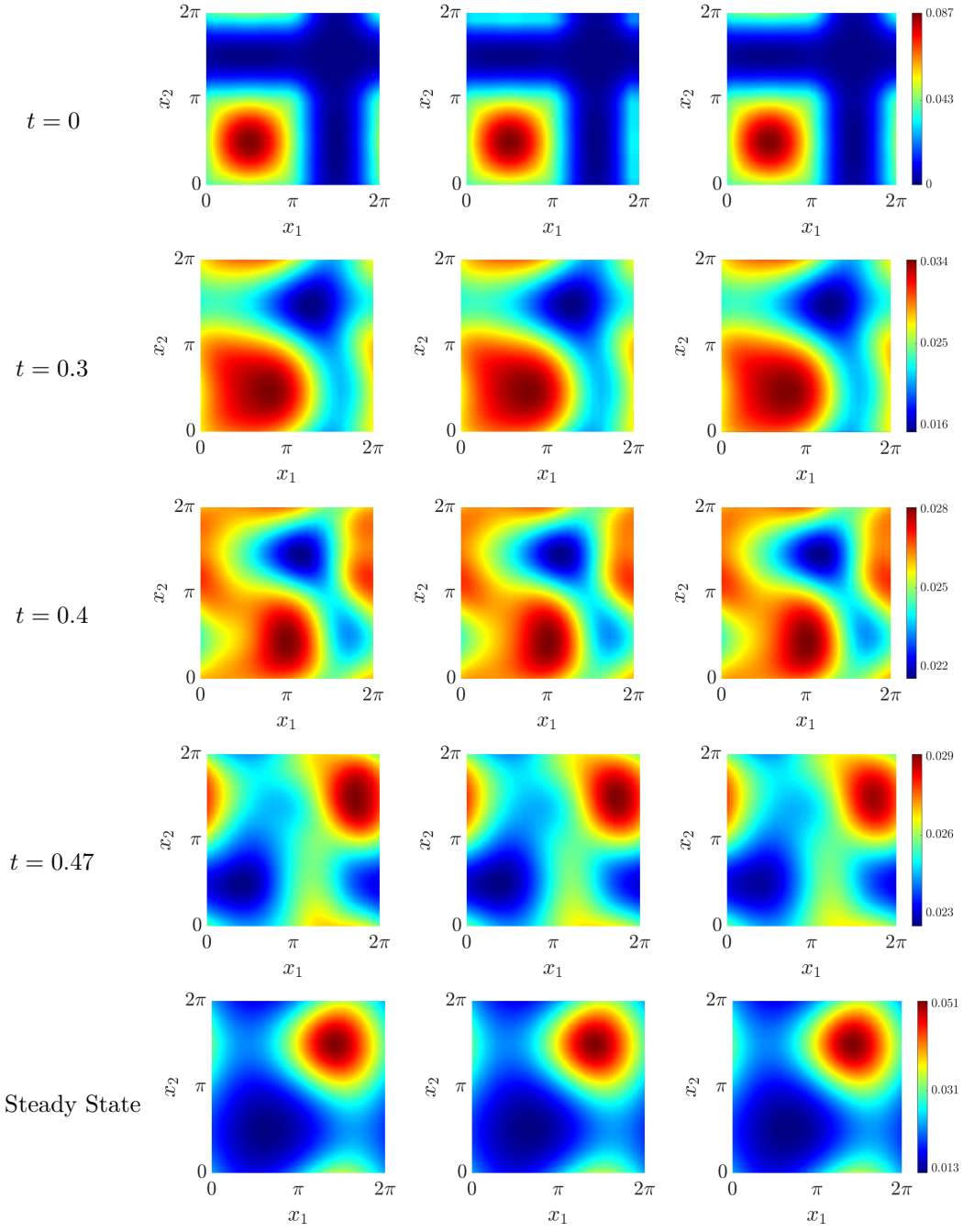


Figure 5.11: Marginal probability density function (5.17) obtained by integrating numerically the Fokker-Planck equation (5.13) in dimension $d = 4$ with initial condition (5.16) using two methods: rank-adaptive Euler forward and rank-adaptive AB2. The reference solution computed with a variable time step size RK4 method with absolute tolerance of 10^{-14} computed on a grid with $20^4 = 160000$ evenly-spaced points.

Integration Method	Free Parameters	Dependent Parameters
Adaptive Euler (HT & TT Formats) (Sec. 3.1)	$\Delta t = 10^{-3}$, $M_1 = M_2 = 10^2$	$\varepsilon_{\boldsymbol{r}} = 10^{-4}$, $\varepsilon_{\boldsymbol{s}} = 10^{-1}$
Adaptive Midpoint (HT Format) (Small Threshold) (Sec. 3.2)	$\Delta t = 10^{-3}$, $A = B = 10^3$, $G = 10^2$	$\varepsilon_{\boldsymbol{\alpha}} = 10^{-6}$, $\varepsilon_{\boldsymbol{\beta}} = 10^{-3}$, $\varepsilon_{\boldsymbol{\gamma}} = 10^{-1}$
Two-step rank-adaptive Adams-Bashforth (HT Format) (Small Threshold) (Sec. 3.3)	$\Delta t = 10^{-3}$, $A = B = 10^3$, $G_0 = G_1 = 10^2$	$\varepsilon_{\boldsymbol{\alpha}} = 10^{-6}$, $\varepsilon_{\boldsymbol{\beta}} = 10^{-3}$, $\varepsilon_{\boldsymbol{\gamma}(0)} = 10^{-4}$, $\varepsilon_{\boldsymbol{\gamma}(1)} = 10^{-4}$
Two-step rank-adaptive Adams-Bashforth (HT Format) (Large Threshold) (Sec. 3.3)	$\Delta t = 10^{-3}$, $A = B = 4 \times 10^4$, $G_0 = G_1 = 4 \times 10^2$	$\varepsilon_{\boldsymbol{\alpha}} = 4 \times 10^{-5}$, $\varepsilon_{\boldsymbol{\beta}} = 4 \times 10^{-2}$, $\varepsilon_{\boldsymbol{\gamma}(0)} = 4 \times 10^{-3}$, $\varepsilon_{\boldsymbol{\gamma}(1)} = 4 \times 10^{-3}$
Adaptive Midpoint (HT Format) (Large Threshold) (Sec. 3.2)	$\Delta t = 10^{-3}$, $A = B = 5 \times 10^4$, $G = 5 \times 10^3$	$\varepsilon_{\boldsymbol{\alpha}} = 5 \times 10^{-5}$, $\varepsilon_{\boldsymbol{\beta}} = 5 \times 10^{-2}$, $\varepsilon_{\boldsymbol{\gamma}} = 5$

Table 5.4: Table of parameters for the rank-adaptive step-truncation integrators of the Fokker-Planck equation (5.13) in dimension $d = 4$ with initial condition (5.16). These were heuristically chosen so that the truncation at each step (to rank $\boldsymbol{r}$ or $\boldsymbol{\alpha}$) would be considerably smaller than the time step. The first step of AB2 uses midpoint with the coefficients listed above.

demonstrate numerically the global error bound we proved in Theorem 3. The error scaling constant Q turns out to be $Q = 2$ for rank-adaptive AB2, $Q = 5$ for rank-adaptive midpoint, and $Q = 0.6$ for rank-adaptive Euler forward.

5.2.4 Four-dimensional Fokker-Planck equation

Next, we present numerical results for the Fokker-Planck equation (5.13) in dimension $d = 4$. In this case, the best truncation operator (2.20) is not explicitly known. Instead, we use the explicit step-truncation methods of Chapter 3, with truncation operator $\mathfrak{T}_r^{\mathrm{SVD}}$ defined in (2.21) (see [45, 64] for more details). We set

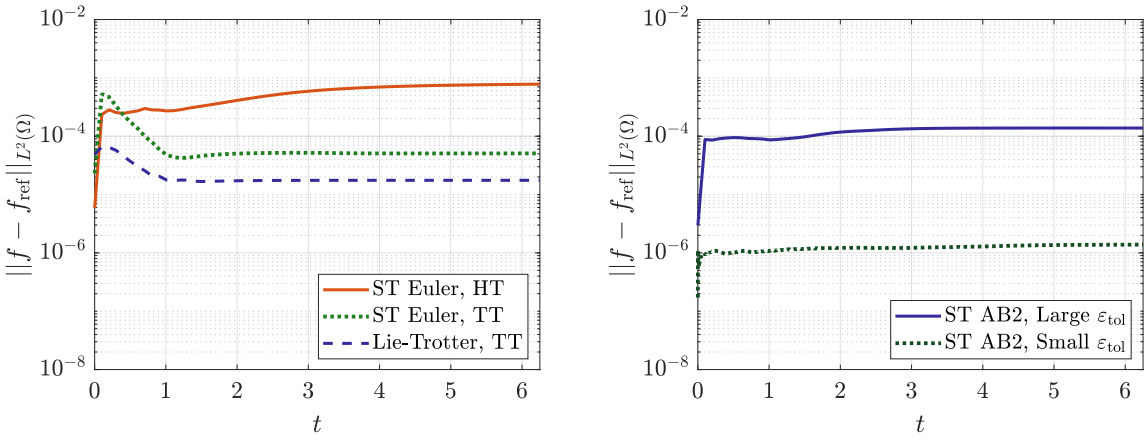


Figure 5.12: $L^2(\Omega)$ error of numerical solutions to the Fokker-Planck equation (5.13) in dimension $d = 4$ with initial condition (5.16). The parameters we used for all rank-adaptive step-truncation methods are summarized in Table 5.4. The rank-adaptive Lie-Trotter method uses a threshold of 10^{-2} for the PDE component normal to the tensor manifold (see [31]).

the initial condition as

$$f_0(x_1, x_2, x_3, x_4) = \frac{1}{m_0} \sum_{j=1}^{L} \left(\prod_{i=1}^{4} \frac{\sin((2j-1)x_i) + 1}{2^{2(j-1)}} + \prod_{i=1}^{4} \frac{\exp(\cos(2jx_i))}{2^{2j-1}} \right), \tag{5.16}$$

where m_0 is a normalization constant. Clearly, (5.16) can be represented exactly in a hierarchical Tucker tensor format provided we use an overall maximal tree rank of $r_0 = 2L$. For our numerical simulations we choose $L = 10$. We change the drift functions slightly from the two-dimensional example we discussed in the previous section. Specifically, here we set $\gamma(x) = \sin(x)$, $\xi(x) = \exp(\sin(x)) + 1$, and $\phi(x) = \cos(x)$ and repeat all numerical tests presented in Section 5.2.3, i.e., we run three rank-adaptive step-truncation simulations with different increment functions: one based on Euler forward (3.5) and one based AB2 (3.15). The parameters we used for these methods are summarized in Table 5.4.

For spatial discretization, we use the Fourier pseudo-spectral method with $20^4 = 160000$ points. We emphasize that a matrix representing the discretized

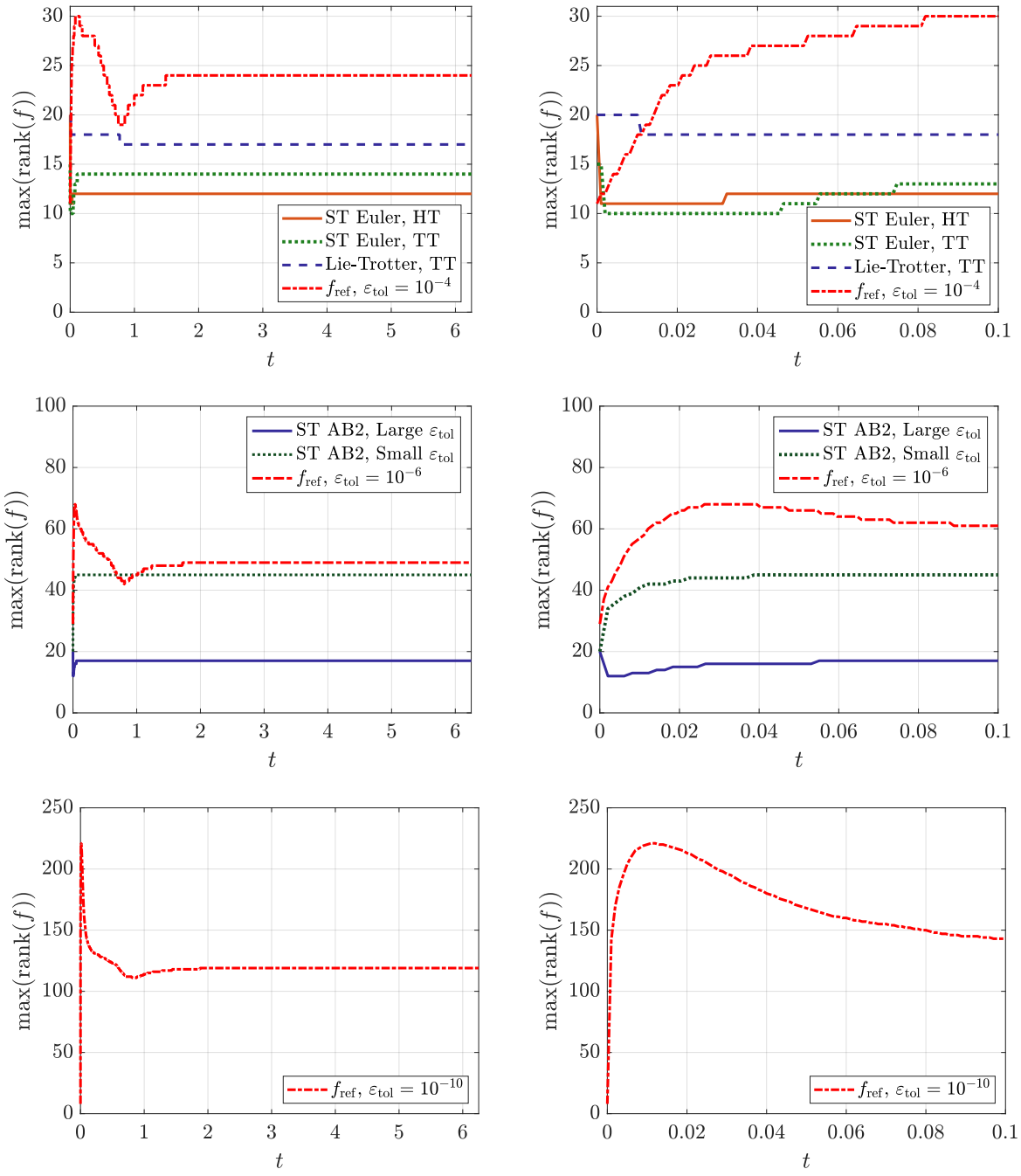


Figure 5.13: Rank versus time for the numerical solutions of Fokker-Planck equation (5.13) in dimension $d = 4$ with initial condition (5.16) (left column: $0 \leq t \leq 6.25$, right column: $0 \leq t \leq 0.1$). We truncate the reference solution to ε_{tol} in HT format. The rank-adaptive Lie-Trotter method uses a threshold of 10^{-2} for the PDE component normal to the tensor manifold (see [31]).

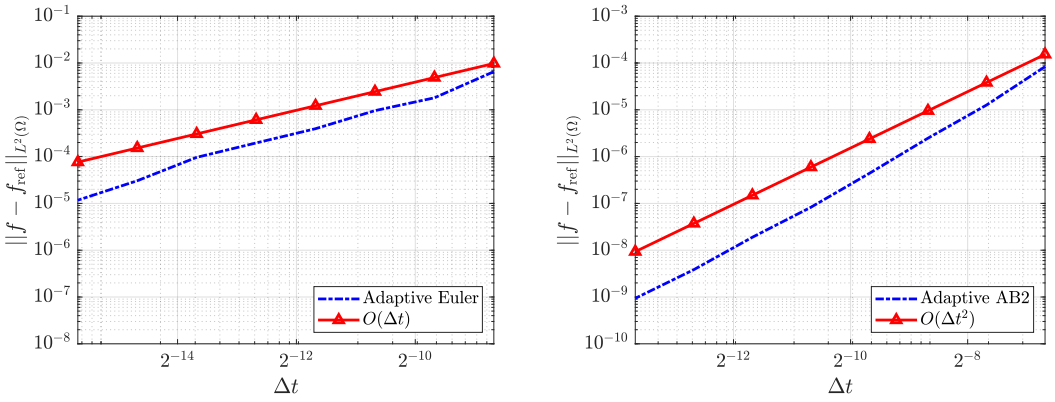


Figure 5.14: Fokker-Planck equation (5.13) in dimension $d = 4$ with initial condition (5.16). $L^2(\Omega)$ errors at $T = 0.1$ versus Δt for different rank-adaptive step-truncation methods. All tests used the HTucker tensor format.

Fokker- Planck operator at the right hand side of (5.13) would be very sparse and require approximately 205 gigabytes in double precision floating point format. The solution vector requires 1.28 megabytes of memory (160000 floating point numbers in double precision). The HTucker format reduces these memory costs considerably. The large threshold solution of Figure 5.13 is only 25 kilobytes when stored to disk using the HTucker Matlab software package [64]. The spatial differential operator for the Fokker-Planck equation can also be represented in HTucker format, and costs only 21 kilobytes. The storage savings are massive, so long as the rank is kept low. In Figure 5.11, we plot a few time snapshots of the marginal PDF

$$f_{12}(x_1, x_2, t) = \int_0^{2\pi} \int_0^{2\pi} f(x_1, x_2, x_3, x_4, t) dx_3 dx_4 \tag{5.17}$$

we obtained by integrating (5.13) in time with rank-adaptive Euler forward and rank-adaptive AB2. In Figure 5.13 we plot the solution rank versus time for all rank-adaptive step-truncation integrators summarized in Table 5.4. The results largely reflect those of the two dimensional domain. However, a notable difference

is the abrupt change in rank. This is because the density function in this case relaxes to steady state fairly quickly. Numerically, the steady state is determined by halting execution when $\|\partial f_{\text{ref}}/\partial t\|_2$ is below the numerical threshold 10^{-8}. This happens at approximately $t \approx 6.25$ for the initial condition (5.16). As the rate of change in the density function becomes very small, we see that the rank no longer changes. This happens near time $t = 0.1$ (see Figure 5.13).

The proposed rank-adaptive step-truncation methods can provide solutions with varying accuracy depending the threshold, i.e., the parameters summarized in Table 5.4. To show this, in Figure 5.13 we compare the rank dynamics in the adaptive AB2 simulations obtained with small or large thresholds. Note that the solution computed with a large error threshold is rather low rank (see Figure 5.13). We also see that the rank can be kept near the rank of the initial condition, if desired (again see Figure 5.13). Finally, in Figure 5.14 we plot the $L^2(\Omega)$ error at $T = 0.1$ versus Δt for two different rank-adaptive step-truncation methods, i.e., Euler and AB2. It is seen that the order of AB2 is slightly larger than 2. This can be explained by noting that the error due to rank truncation is essentially a sum of singular values. Such singular values can be smaller than the truncation thresholds $\varepsilon_{\boldsymbol{\kappa}}$ ($\boldsymbol{\kappa} = \boldsymbol{r}, \boldsymbol{s}, \boldsymbol{\alpha}$, ...), suggesting the theoretical bounds may not be tight.

5.3 Implicit Rank-Adaptive Methods

In this section we study the performance of the proposed implicit step-truncation methods in three numerical applications involving time-dependent PDEs. Specifically, we study the Allen-Cahn equation [78] in two spatial dimensions, the Fokker-Planck equation [90] in four dimensions, and the nonlinear Schrödinger equation [104] in six dimensions. Our code was built on the backbone of the HTucker Mat-

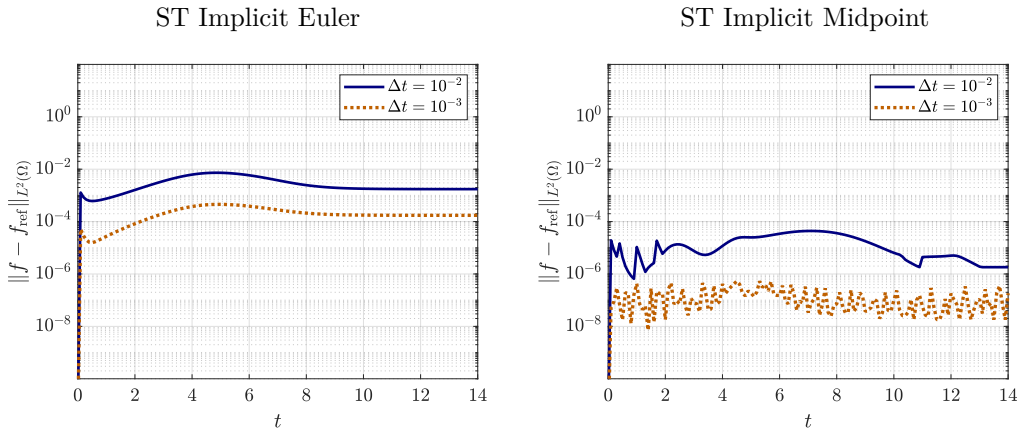


Figure 5.15: Error versus time for step-truncation numerical solutions of Allen-Cahn equation (5.13) in dimension $d = 2$ with initial condition (5.19).

lab package [64]. All tensor operations for the step-truncation schemes described in Chapter 3 use function calls to the HTucker Matlab library.

5.3.1 Allen-Cahn equation

The Allen-Cahn equation is a reaction-diffusion equation that describes the process of phase separation in multi-component alloy systems [3, 4]. In its simplest form, the equation has a cubic polynomial non-linearity (reaction term) and a diffusion term [56], i.e.,

$$\frac{\partial f}{\partial t} = \varepsilon \Delta f + f - f^3. \tag{5.18}$$

In our application, we set $\varepsilon = 0.1$, and solve (5.18) on the two-dimensional flat torus $\Omega = [0, 2\pi]^2$. We employ a second order splitting [42] method to solve the Laplacian Δf as a fixed rank temporal integration and cubic $f - f^3$ term using our rank adaptive integration. The initial condition is set as

$$f_0(x, y) = u(x, y) - u(x, 2y) + u(3x + \pi, 3y + \pi) - 2u(4x, 4y) + 2u(5x, 5y), \tag{5.19}$$

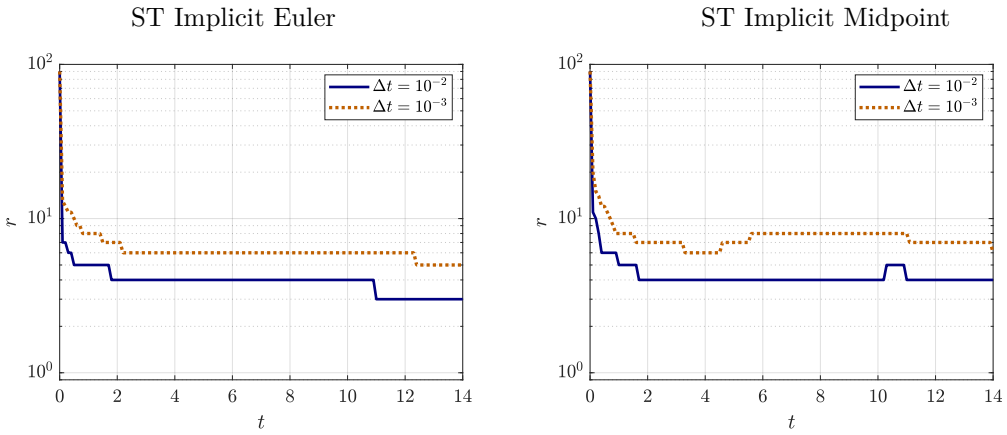


Figure 5.16: Rank versus time for step-truncation numerical solutions of Allen-Cahn equation (5.13) in dimension $d = 2$ with initial condition (5.19).

where

$$u(x,y) = \frac{\left[e^{-\tan^2(x)} + e^{-\tan^2(y)}\right] \sin(x)\sin(y)}{1 + e^{|\csc(-x/2)|} + e^{|\csc(-y/2)|}}. \tag{5.20}$$

We discretize (5.18) in space using the two-dimensional Fourier pseudospectral collocation method [51] with 257×257 points in $\Omega = [0, 2\pi]^2$. This results in a matrix ODE in the form of (1.2). We truncate the initial condition to absolute and relative SVD tolerances of 10^{-9}, which yields an initial condition represented by a 257×257 matrix of rank 90. We also computed a benchmark solution of the matrix ODE using a variable step RK4 method with absolute tolerance set to 10^{-14}. We denote the benchmark solution as f_{ref}. In Figure 5.15 we observe the transient accuracy of our order one and order two implicit methods. The stopping tolerance for inexact Newton's iterations is set to $\varepsilon_{\text{tol}} = 2.2 \times 10^{-8}$, while the HT/TT-GMRES relative error is chosen as $\eta = 10^{-3}$. Time integration was halted at $t = 14$. After this time, the system is close to steady state and the errors stay bounded near the final values plotted in Figure 5.15. Similarly, the rank also levels out around $t = 14$. In Figure 5.16, we plot temporal evolution of the rank for both the implicit step-truncation Euler and midpoint methods.

Due to the smoothing properties of the Laplacian, the high frequencies in the initial condition quickly decay and, correspondingly, the rank drops significantly within the first few time steps. Due to the rapidly decaying rank for this problem, we have plotted it in log scale. In Figure 5.17, we provide a comparison between the rank-adaptive implicit step-truncation midpoint method we propose here and the rank-adaptive explicit step-truncation midpoint method

$$\boldsymbol{f}_{k+1} = \mathfrak{T}_{\boldsymbol{r}_3}\left(\boldsymbol{f}_k + \Delta t \mathfrak{T}_{\boldsymbol{r}_2}\left(\boldsymbol{G}\left(\boldsymbol{f}_k + \frac{\Delta t}{2}\mathfrak{T}_{\boldsymbol{r}_1}(\boldsymbol{G}(\boldsymbol{f}_k))\right)\right)\right), \tag{5.21}$$

of Section 3.2. Figure 5.17 shows that the explicit step-truncation midpoint method undergoes a numerical instability for $\Delta t = 10^{-3}$. Indeed it is a conditionally stable method. The explicit step-truncation midpoint method also has other issues. In particular, in the rank-adaptive setting we consider here, we have that in the limit $\Delta t \to 0$ the parameters $\varepsilon_{\boldsymbol{r}_1}$, $\varepsilon_{\boldsymbol{r}_2}$ and $\varepsilon_{\boldsymbol{r}_3}$ all go to zero (see equation (3.12)). This implies that the truncation operators may retain all singular values, henceforth maxing out the rank and thereby giving up all computational gains of low-rank tensor compression. On the other hand, if Δt is too large, then we have stability issues as discussed above. Indeed, we see both these problems with the explicit step-truncation midpoint method, giving only a relatively narrow region of acceptable time step sizes in which the method is effective.

In Table 5.5 we provide a comparison between explicit and implicit step-truncation midpoint methods in terms of computational cost (CPU-time on an Intel Core I9-7980XE workstation) and accuracy at time $t = 1$. It is seen that the implicit step-truncation midpoint method is roughly 20 to 30 times faster than the explicit step-truncation midpoint method for a comparable error. Moreover, solutions with a large time step ($\Delta t > 10^{-3}$) are impossible to achieve with the explicit step-truncation method due to time step restrictions associated with

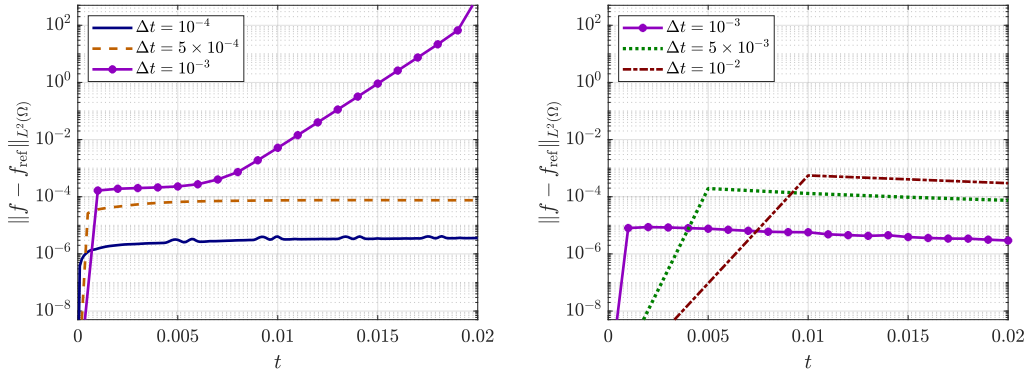


Figure 5.17: Allen-Cahn equation (5.18). Comparison between the $L^2(\Omega)$ errors of explicit and implicit step-truncation midpoint methods for different Δt.

conditional stability.

5.3.2 Fokker-Planck equation

Consider the Fokker-Planck equation

$$\frac{\partial f(\boldsymbol{x},t)}{\partial t} = -\sum_{i=1}^{d} \frac{\partial}{\partial x_i}\left(\mu_i(\boldsymbol{x}) f(\boldsymbol{x},t)\right) + \frac{\sigma^2}{2}\sum_{i=1}^{d} \frac{\partial^2 f(\boldsymbol{x},t)}{\partial x_i^2} \tag{5.22}$$

on a four-dimensional ($d = 4$) flat torus $\Omega = [0, 2\pi]^4$. The components of the drift are chosen as

$$\mu_i(\boldsymbol{x}) = (\gamma(x_{i+1}) - \gamma(x_{i-2}))\xi(x_{i-1}) - \phi(x_i), \qquad i = 1, \ldots, d, \tag{5.23}$$

where $\gamma(x) = \sin(x)$, $\xi(x) = \exp(\sin(x)) + 1$, and $\phi(x) = \cos(x)$ are 2π-periodic functions. In (5.23) we set $x_{i+d} = x_i$. For this particular drift field, the right side of (5.22) can be split into a component tangential to the tensor manifold $\mathcal{H}_{\boldsymbol{r}}$ and

ST Explicit Midpoint		
Δt	Runtime (seconds)	$\|\boldsymbol{f} - \boldsymbol{f}_{\text{ref}}\|$
1.0×10^{-3}	Did not finish	Unstable
5.0×10^{-4}	2.2946×10^{2}	6.0713×10^{-3}
2.5×10^{-4}	4.7828×10^{2}	5.2628×10^{-4}
1.0×10^{-4}	1.2619×10^{3}	5.3648×10^{-5}
5.0×10^{-5}	2.7354×10^{3}	1.1723×10^{-5}
ST Implicit Midpoint		
Δt	Runtime (seconds)	$\|\boldsymbol{f} - \boldsymbol{f}_{\text{ref}}\|$
1.0×10^{-1}	5.3097	2.5652×10^{-2}
5.0×10^{-2}	1.0495×10^{1}	7.7248×10^{-3}
2.5×10^{-2}	1.8987×10^{1}	2.0977×10^{-3}
1.0×10^{-2}	3.8025×10^{1}	6.7477×10^{-5}
5.0×10^{-3}	7.6183×10^{1}	3.6012×10^{-6}

Table 5.5: Allen-Cahn equation (5.18). Comparison between explicit and implicit step-truncation (ST) midpoint methods in terms of computational cost (CPU-time on an Intel Core I9-7980XE workstation) and accuracy at final time $t = 1$. It is seen that the implicit step-truncation midpoint method is roughly 20 to 30 times faster than the explicit step-truncation midpoint method for a comparable error.

a component that is non-tangential as

$$\frac{\partial f(\boldsymbol{x},t)}{\partial t} = \sum_{i=1}^{d} \underbrace{\left(-\gamma(x_{i+1})\xi(x_{i-1})\frac{\partial f(\boldsymbol{x},t)}{\partial x_i} + \gamma(x_{i-2})\xi(x_{i-1})\frac{\partial f(\boldsymbol{x},t)}{\partial x_i} \right)}_{\text{Not tangential}} + \underbrace{\left(\frac{\partial}{\partial x_i}\phi(x_i) f(\boldsymbol{x},t) + \frac{\sigma^2}{2}\frac{\partial^2 f(\boldsymbol{x},t)}{\partial x_i^2} \right)}_{\text{tangential}}. \quad (5.24)$$

We solve (5.24) using an operator splitting method. To this end, we notice that there are $3d$ many terms in the summation above, and therefore we first solve the first d time dependent PDEs which are tangential to the tensor manifold $\mathcal{H}_{\boldsymbol{r}}$, i.e.,

$$\frac{\partial g_i}{\partial t} = \frac{\partial}{\partial x_i}\phi(x_i) g_i + \frac{\sigma^2}{2}\frac{\partial^2 g_i}{\partial x_i^2}, \qquad i = 1, \ldots, d. \quad (5.25)$$

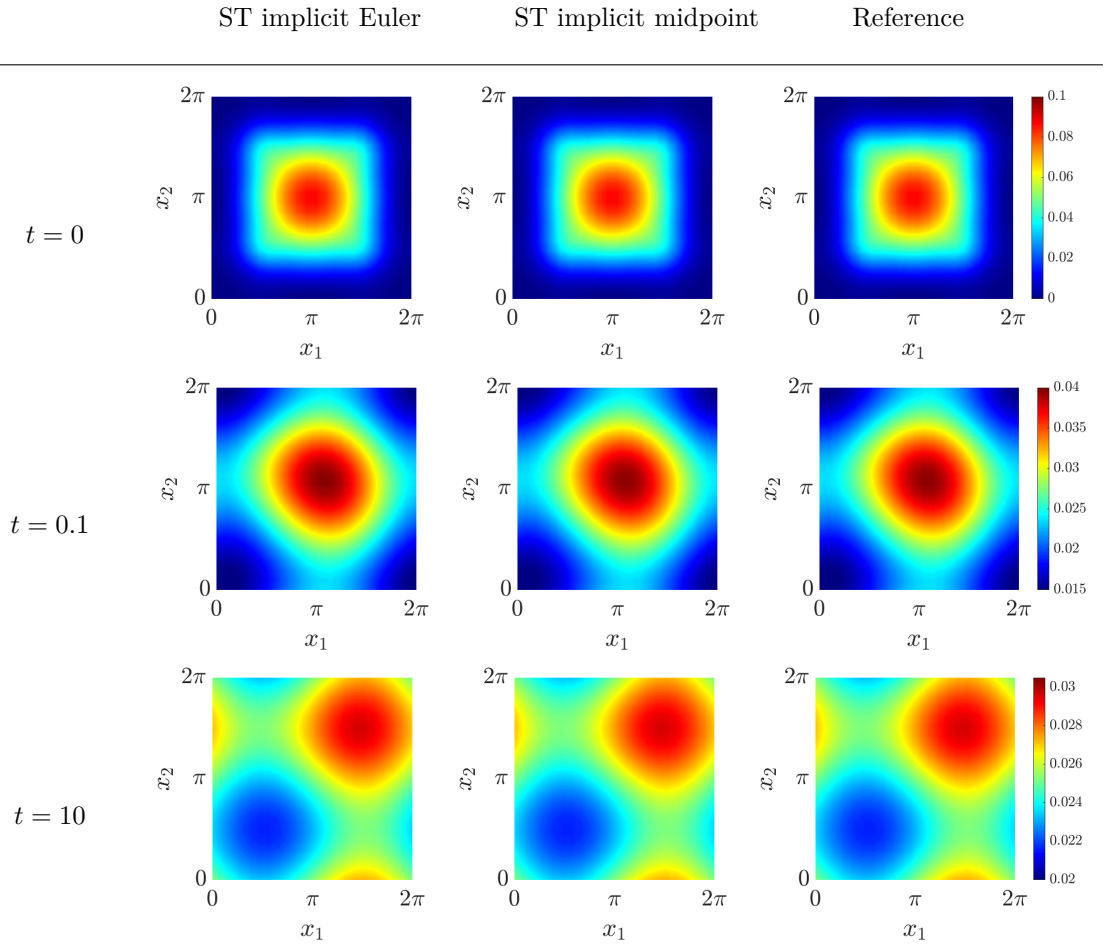


Figure 5.18: Marginal probability density function (5.17) obtained by integrating numerically the Fokker–Planck equation (5.13) in dimension $d = 4$ with $\sigma = 5$ and initial condition (5.16) with two methods: i) rank-adaptive implicit step-truncation Euler and ii) rank-adaptive implicit step-truncation midpoint. The reference solution is a variable time step RK4 method with absolute tolerance of 10^{-14}. These solutions are computed on a grid with $20 \times 20 \times 20 \times 20$ interior points (evenly spaced). The steady state is determined for this computation by halting execution when $\|\partial f_{\text{ref}}/\partial t\|_2$ is below a numerical threshold of 10^{-8}. This happens at approximately $t \approx 10$ for the initial condition (5.16).

Then we solve the non-tangential equations in two batches,

$$\frac{\partial u_j}{\partial t} = \gamma(x_{i+1})\xi(x_{i-1})\frac{\partial u_j}{\partial x_i}, \qquad j = 1, \ldots, d \tag{5.26}$$

$$\frac{\partial u_k}{\partial t} = \gamma(x_{i-2})\xi(x_{i-1})\frac{\partial u_k}{\partial x_i}, \qquad k = 2, \ldots, 2d. \tag{5.27}$$

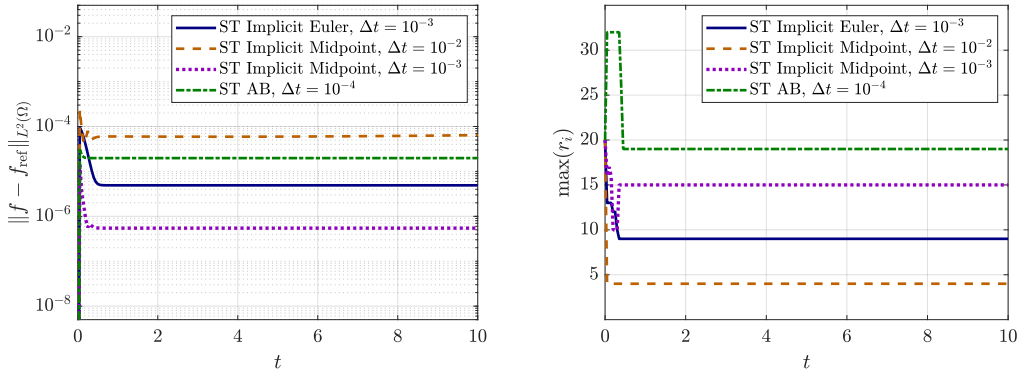


Figure 5.19: $L^2(\Omega)$ error and rank versus time for numerical solutions of Fokker–Planck equation (5.13) in dimension $d = 4$ with initial condition (5.16). The rank plotted here is the largest rank for all tensors being used to represent the solution in HT format. Rank of the reference solution is in HT format.

This yields the first order (Lie-Trotter) approximation $f(\boldsymbol{x}, \Delta t) = u_{2d}(\boldsymbol{x}, \Delta t) + O(\Delta t^2)$. We also use these same lists of PDEs for the second order (Strang) splitting integrator. For each time step in both first- and second order splitting methods, we terminate the HT/TT-GMRES iterations by setting the stopping tolerance $\varepsilon_{\text{tol}} = 10^{-9}$. We set the initial probability density function (PDF) as

$$f_0(x_1, x_2, x_3, x_4) = \frac{1}{F_0} \sum_{j=1}^{M} \left(\prod_{i=1}^{4} \frac{\sin((2j-1)x_i - \pi/2) + 1}{2^{2(j-1)}} + \prod_{i=1}^{4} \frac{\exp(\cos(2jx_i + \pi))}{2^{2j-1}} \right), \tag{5.28}$$

where F_0 is a normalization constant. This gives an HTucker tensor with rank bounded by $2M$. We set $M = 10$ to give ranks bounded by 20. We discretize (5.22)-(5.28) in Ω with the Fourier pseudospectral collocation method [51] on a tensor product grid with $N = 20$ evenly-spaced points along each coordinate x_i, giving the total number of points $(N+1)^4 = 194481$. This number corresponds to the number of entries in the tensor $\boldsymbol{f}(t)$ appearing in equation (1.2). Also, we set $\sigma = 5$ in (5.22).

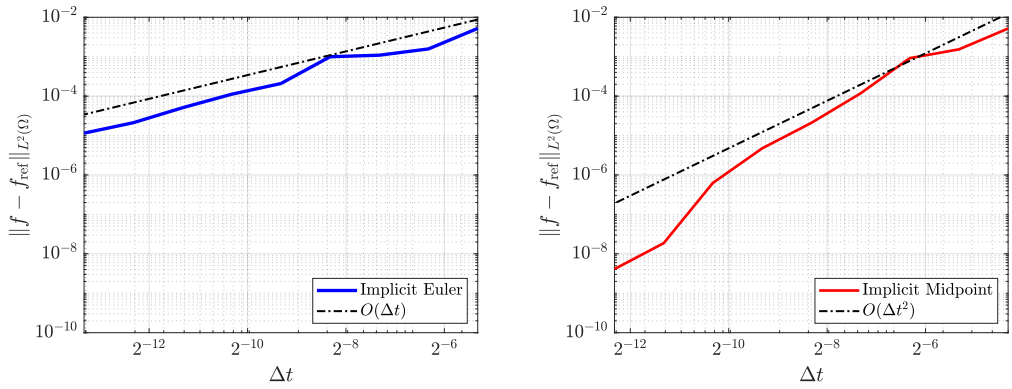


Figure 5.20: $L^2(\Omega)$ errors at $t = 0.1$ for the implicit rank-adaptive step-truncation implicit Euler and midpoint methods versus Δt. The reference solution of (5.13) was computed using a variable time step RK4 method with absolute tolerance of 10^{-14}.

In Figure 5.18 we compare a few time snapshots of the marginal PDF

$$f_{12}(x_1, x_2) = \int_0^{2\pi} \int_0^{2\pi} f(x_1, x_2, x_3, x_4) dx_3 dx_4, \tag{5.29}$$

we obtained with the rank-adaptive implicit step-truncation Euler and midpoint methods, as well as the reference marginal PDF. The solution very quickly relaxes to nearly uniform by $t = 0.1$, then slowly rises to its steady state distribution by $t = 10$.

In Figure 5.19 we study accuracy and rank of the proposed implicit step-truncation methods in comparison with the rank-adaptive explicit Adams-Bashforth (AB) method of order 2 (see [92]). The implicit step-truncation methods use a time step size of $\Delta t = 10^{-3}$, while step-truncation AB2 uses a step size of $\Delta t = 10^{-4}$. The highest error and lowest rank come from the implicit step-truncation midpoint method with $\Delta t = 10^{-2}$. The highest rank and second highest error go to the step-truncation AB2 method, which runs with time step size 10^{-4} for stability. This causes a penalty in the rank, whence time step is made small, hence the rank must increase to maintain convergence order. (See Chapter 3 for detailed rank increase

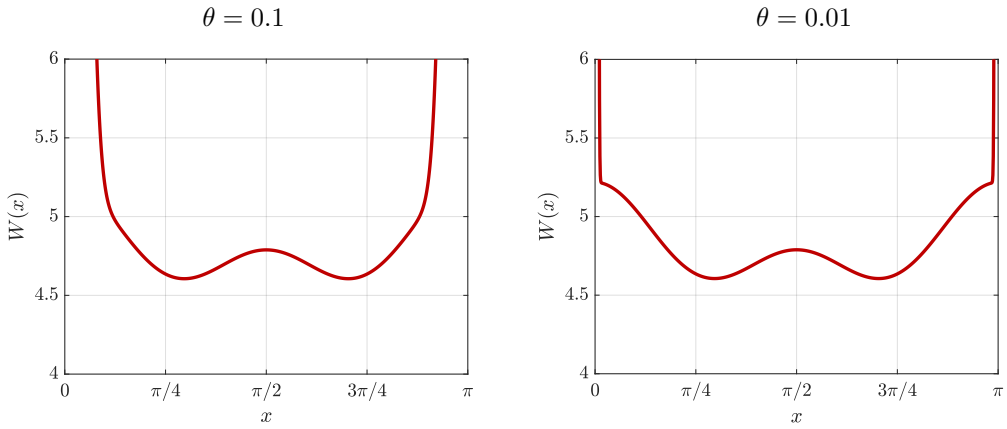


Figure 5.21: Double-well potential (5.32)-(5.33) for different values of θ. It is seen that as $\theta \to 0$, the potential barrier at $x = 0$ and $x = \pi$ becomes infinitely high. This is identical to the well-known homogeneous boundary conditions for particles trapped in a box.

requirements.) The implicit step-truncation midpoint method performs the best, with error of approximately 10^{-6} and rank lower than the step-truncation AB2 method at steady state. Overall, the proposed implicit step-truncation methods perform extremely well on linear problems of this form, especially when the right hand side is explicitly written as a sum of tensor products of one dimensional operators. In Figure (5.20) we show a plot of the convergence rate of implicit step-truncation Euler and midpoint methods. For this figure, we set $\sigma = 2$. The convergence rates are order one and order two respectively, verifying Theorem 3.

5.3.3 Nonlinear Schrödinger equation

The nonlinear Schrödinger equation is a complex-valued PDE whose main applications are wave propagation in nonlinear optical fibers and Bose-Einstein con-

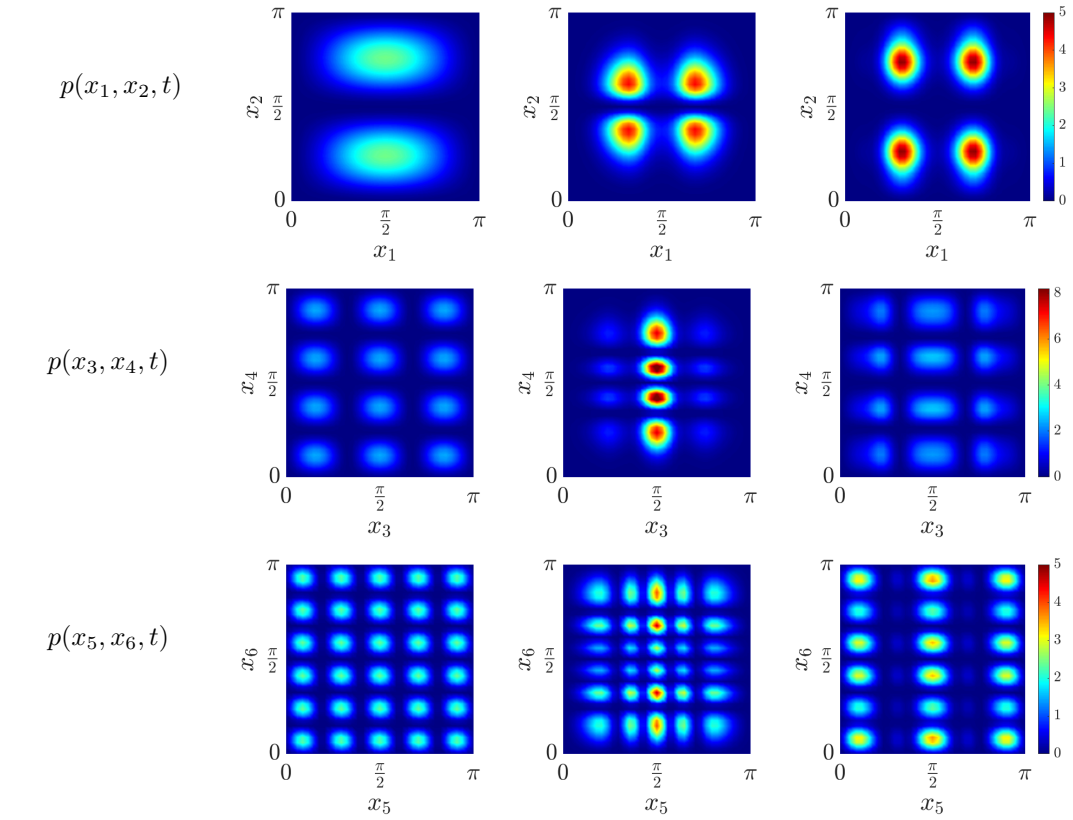


Figure 5.22: Marginal probability density functions representing particle positions generated by the nonlinear Schrödinger equation (5.31) with $\varepsilon = 10^{-4}$, interaction potential (5.32) and initial condition (5.34).

densates [95, 104]. The equation can be written as[1]

$$\frac{\partial \phi(\boldsymbol{x}, t)}{\partial t} = \frac{i}{2}\Delta\phi(\boldsymbol{x}, t) - iV(\boldsymbol{x})\phi(\boldsymbol{x}, t) - i\varepsilon|\phi(\boldsymbol{x}, t)|^2\phi(\boldsymbol{x}, t). \tag{5.31}$$

where $V(\boldsymbol{x})$ is the particle interaction potential. In our example, we consider 6 particles trapped on a line segment in the presence of a double-well potential

[1]As is well-known, the nonlinear Schrödinger equation (5.31) is a Hamiltonian PDE which can be derived as a stationary point of the energy density (Hamilton's functional)

$$H(\phi) = \int_\Omega \left(\frac{1}{4}\|\nabla\phi\|^2 + \frac{1}{2}V(\boldsymbol{x})|\phi|^2 + \frac{\varepsilon}{4}|\phi|^4 \right) d\boldsymbol{x}. \tag{5.30}$$

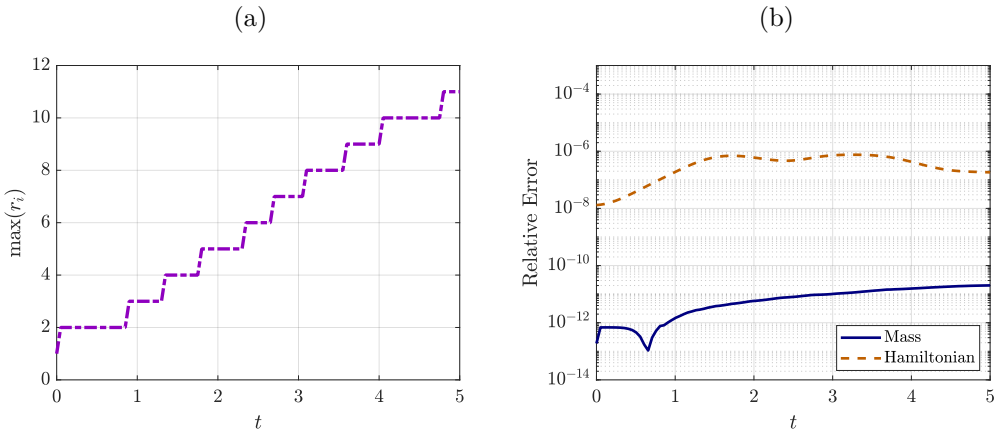


Figure 5.23: (a) Maximum tensor rank versus time, and (b) relative error in the solution mass and Hamiltonian (5.30) for nonlinear Schrödinger equation (5.31) in dimension $d = 6$, with $\varepsilon = 10^{-4}$, interaction potential (5.32) and initial condition (5.34).

defined as

$$V(\boldsymbol{x}) = 50 \sum_{k=1}^{6} W(x_k), \qquad W(x_k) = \left[1 + e^{\cos(x_k)^2} + \frac{3}{4}\left(1 + e^{\sin(x_k)^2}\right)\right] \eta_\theta(x_k). \tag{5.32}$$

Here, $W(x_k)$ is a potential with barriers at $x_k = 0$ and $x_k = \pi$ (see Figure 5.21). The function $\eta_\theta(x_i)$ is a mollifier which converges weakly to $1 + \delta(x_i) + \delta(x_i - \pi)$ as $\theta \to 0$. One such mollifier is

$$\eta_\theta(x) = 1 + \frac{1}{\sqrt{2\pi}\theta}\left(\exp\left[-\frac{x^2}{2\theta^2}\right] + \exp\left[-\frac{(x-\pi)^2}{2\theta^2}\right]\right). \tag{5.33}$$

As $\theta \to 0$, the weak limit of η_θ translates to zero Dirichlet boundary conditions on the domain $\Omega = [0, \pi]^6$. The Dirichlet conditions naturally allow us to use a discrete sine transform to compute the Laplacian's differentiation matrices. We discretize the domain Ω on a uniform grid with 35 points per dimension. This gives us a tensor with $35^6 = 838265625$ entries, or 14.7 Gigabytes per temporal solution snapshot if we store the uncompressed tensor in a double precision IEEE

754 floating point format. We choose a product of pure states for our initial condition, i.e.,

$$\phi(\boldsymbol{x}, 0) = \prod_{k=1}^{6} \frac{6^{1/6} 4k}{2k\pi - \sin(2\pi k)} \sin(kx_k), \tag{5.34}$$

The normalizing constant $6^{1/6}4k/(2k\pi - \sin(2\pi k))$ guarantees that the wavefunction has an initial mass of 6 particles. We now apply an operator splitting method to solve (5.31). The linear components are all tangential to the tensor manifold $\mathcal{H}_{\boldsymbol{r}}$, and have a physical interpretation. Specifically,

$$\frac{\partial g_k}{\partial t} = \frac{i}{2} \frac{\partial^2 g_k}{\partial x_k^2} - 50iW(x_k)g_k, \qquad k = 1, \ldots, 6, \tag{5.35}$$

is a sequence of one-dimensional linear Schrödinger equations. The non-tangential part reduces to

$$\frac{du}{dt} = i\varepsilon |u|^2 u, \tag{5.36}$$

which may be interpreted as an ODE describing all pointwise interactions of the particles. Here we set $\varepsilon = 10^{-4}$ to model weak iteractions. Clearly, the linear terms in (5.31) have purely imaginary eigenvalues. Therefore to integrate the semi-discrete form of (5.31) in time we need a numerical scheme that has the imaginary axis within its stability region. Since implicit step-truncation Euler method introduces a significant damping, thereby exacerbating inaccuracy due to discrete time stepping, we apply the implicit step-truncation midpoint method. For this problem, we set $\Delta t = 5 \times 10^{-2}$ and the tensor truncation error to be constant in time at $100\Delta t^3$ to maintain second order consistency. Tolerance of the inexact Newton method was set to 5×10^{-5} and the HT/TT-GMRES relative error to $5 \times \eta = 10^{-4}$. In Figure 5.22 we plot the time-dependent marginal probability density functions for the joint position variables (x_k, x_{k+1}), $k = 1, 3, 5$.

Such probability densities are defined as

$$p(x_1, x_2, t) = \frac{1}{6} \int_{[0,1]^4} \phi^*(\boldsymbol{x}, t)\phi(\boldsymbol{x}, t) dx_3 dx_4 dx_5 dx_6, \tag{5.37}$$

and analogously for $p(x_3, x_4, t)$ and $p(x_5, x_6, t)$. It is seen that the lower energy pure states (position variables (x_1, x_2)) quickly get trapped in the two wells, oscillating at their bottoms. Interestingly, at $t = 2.5$ it appears that particle x_3 is most likely to be observed in between the two wells whenever the particle x_4 is in a well bottom.

In Figure 5.23(a), we plot the rank over time for this problem. The rank also has physical meaning. A higher rank HT tensor is equivalent to a wavefunction with many entangled states, regardless of which $L^2(\Omega)$ basis we choose. In the example discussed in this section, the particles interacting over time monotonically increase the rank. We emphasize that the nonlinearity in (5.31) poses a significant challenge to tensor methods. In fact, in a single application of the function $i\varepsilon|u|^2u$, we may end up cubing the rank. This can be mitigated somewhat by using the approximate element-wise tensor multiplication routine. Even so, if the inexact Newton Method requires many dozens of iterations to halt, the rank may grow very rapidly in a single time step, causing a slowing due to large array storage. This problem is particularly apparent when $\varepsilon \approx 1$ or if Δt is made significantly smaller, e.g. $\Delta t = 10^{-4}$. A more effective way of evaluating nonlinear functions on tensors decompositions would certainly mitigate this issue. In Figure 5.23(b), we plot the relative error of the solution mass and the Hamiltonian (5.30) over time. The relative errors hover around 10^{-11} and 10^{-6}, respectively. It is remarkable that even though the additional tensor truncation done after the inner loop of the implicit solver in principle destroys the symplectic properties of the midpoint method, the mass and Hamiltonian are still preserved with high accuracy.

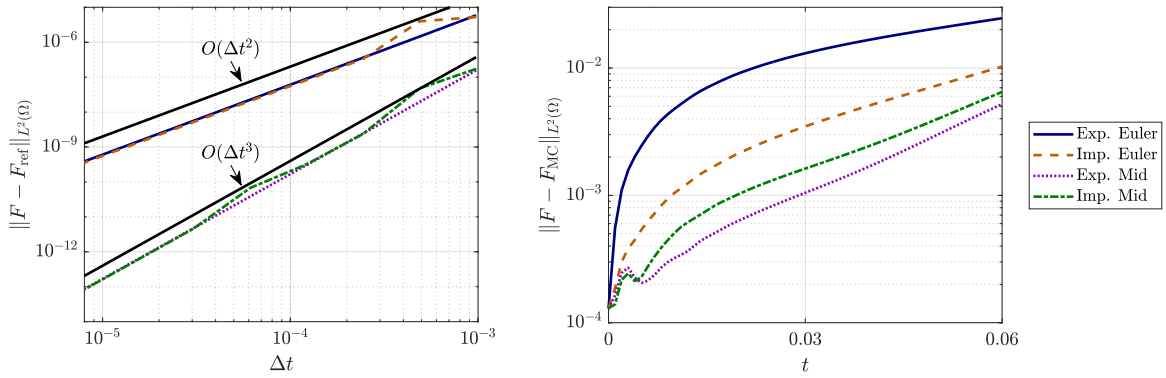


Figure 5.24: Left: Local truncation error found by comparing each time stepping scheme with its Richardson extrapolation. Right: Numerical difference of the $u(0,t)$ one-point marginal CDF with Monte-Carlo estimate with finite difference discretization (5.38) with 8 points and Step-Truncation methods of dimension $N = 8$.

5.4 Burgers' Equation with Uncertain Initial State

In this section, we present an application to the Burgers' equation with reaction term

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \gamma \frac{\partial^2 u}{\partial x^2} + R(u), \tag{5.38}$$

and an uncertain initial condition $u(x,0)$, $R(u) = \alpha \sin(\pi u)$, and 2π periodic boundary conditions. When u is bound between 0 and 1, i.e. the roots of R, the reaction term pulls the extreme points of u near the roots of R. Meanwhile, advection-diffusion terms of the Burger's equation smooth and flatten out the solutions. We solve the PDF equation corresponding to this problem. What

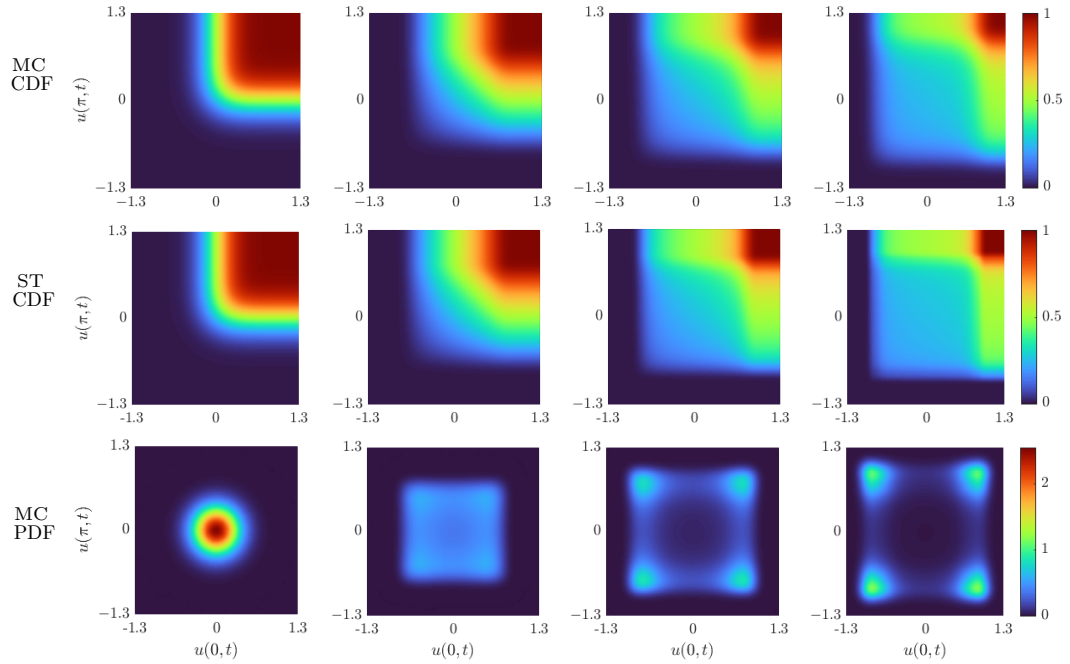


Figure 5.25: Two-point joint CDF of $u(0,t)$ and $u(\pi,t)$. The CDF is computed by generating numerical solutions to (5.40) for $N = 20$ and marginalizing the solution in the remaining 18 variables. We also show a Monte-Carlo estimate of the joint CDF obtained by sampling 5×10^6 solutions to (5.38).

follows is the Liouville-type linear hyperbolic conservation law

$$\frac{\partial}{\partial t} p(t, u_1, u_2, \dots u_N) = \\ -\sum_{i=1}^{N} \frac{\partial}{\partial u_j} \left(\sum_{j=1}^{N} \left(-u_i D_{ij}^{(1)} u_j + \delta_{ij} R(u_j) \gamma D_{ij}^{(2)} u_j \right) p(t, u_1, u_2, \dots u_N) \right), \tag{5.39}$$

where δ_{ij} is the Kronecker delta. In the above PDF equation, $D_{ij}^{(1)}$ and $D_{ij}^{(2)}$ turn out as differention matrix coefficients. In our numerical experiments, we use the second order finite difference matrix on a periodic domain. From here, it is possible to write down a numerical tensor method to solve the parital differential equation. However, we assumed through the analysis of tensor methods is that the

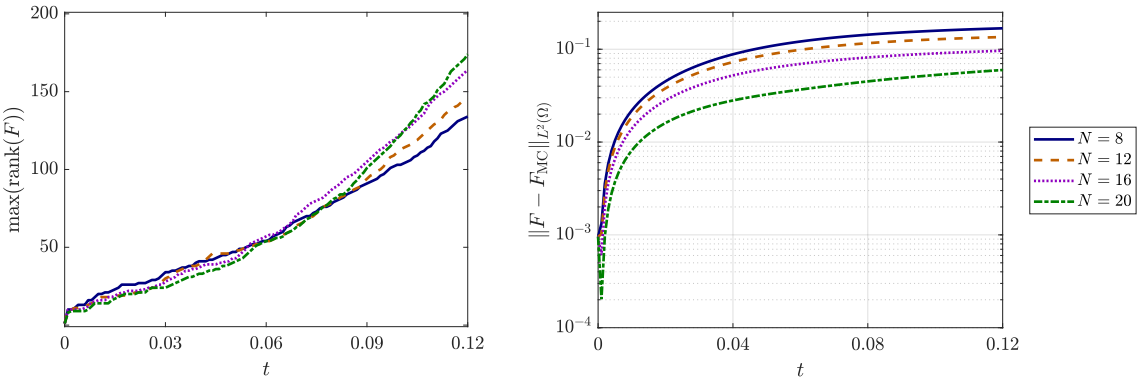


Figure 5.26: Left: Highest rank of the TT cores with varying dimension for the explicit method solutions to equation (5.40). Right: Decreasing error with increasing dimension for the two-point CDFs shown in Figure 5.25.

solution to the PDE is in $L^2(\Omega)$. It is clear that need not be the case in (5.39). For example, if we ignore the reaction term and select homogeneous boundary conditions, the solution to (5.38) decays to zero. Thus p converges to a Dirac delta distribution and our tensor truncation algorithms fail inside of the SVD evalution of Algorithm 8, where the $L^2(\Omega)$ norm is checked.

To remedy this, we integrate p to obtain the Cumulative Distribution Function (CDF). Doing so results in an integro-partial differential equation. Specifically,

$$\begin{aligned}\frac{\partial F}{\partial t} = &\sum_{i,j=1}^{N} (u_i D_{ij}^{(1)} u_j - \gamma D_{ij}^{(2)} u_j - \delta_{ij} R(u_j)) \frac{\partial F}{\partial u_i} \\ &- \sum_{\substack{i,j=1 \\ i \neq j}}^{N} (D_{ij}^{(1)} u_i - \gamma D_{ij}^{(2)}) \int_{-\infty}^{u_j} \frac{\partial F}{\partial u_i} d\mu_j \end{aligned} \tag{5.40}$$

may be derived by integrating the right side of (5.39) and applying several integration-by-parts formulae with the integration dummy-variables μ_1, μ_2, ..., μ_N. In order to apply a discretized tensor-train method to equation (5.40), we seek a discretization of the domain. We choose a square high-dimensional cube centered at the

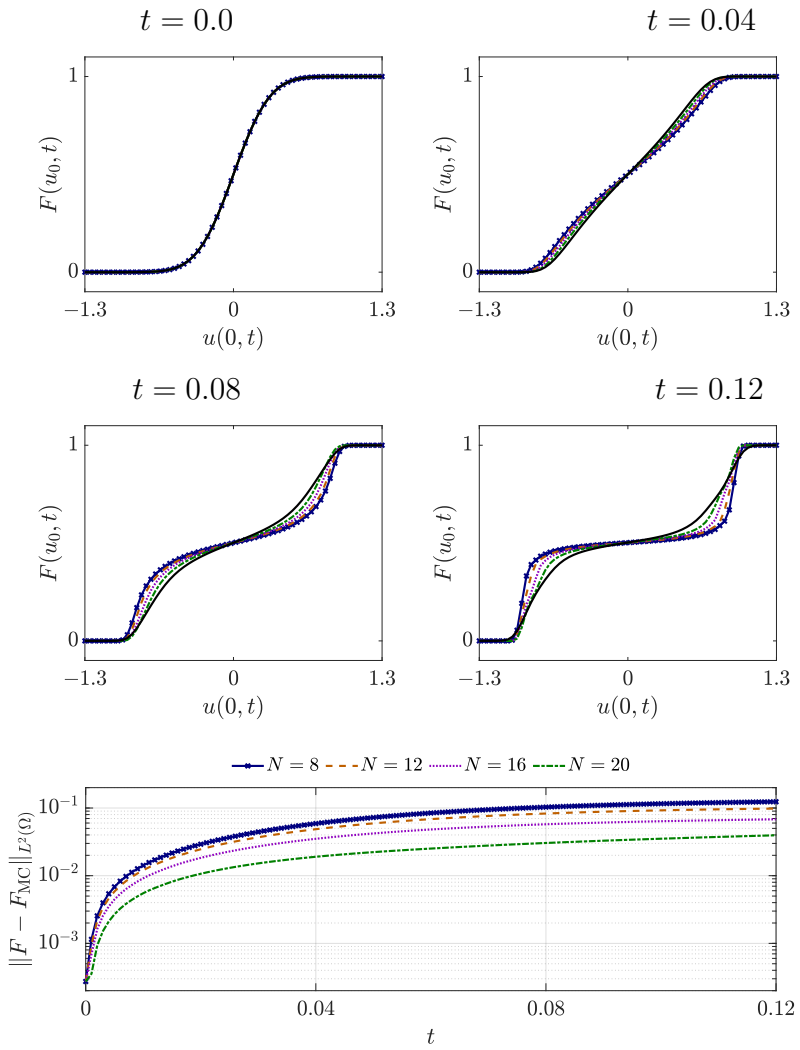


Figure 5.27: One-point marginal CDF of numerical solutions to (5.40) with varying dimensionality accompanied with time dependent numerical error of one-point marginal CDF with varying dimensionality aligned with each time snapshot.

origin which covers the support of p. On the boundary of this cube, it is straightforward to show that the derivative of F normal to the cube boundary is zero. Therefore we may apply outflow boundary conditions to the space-like derivatives of our tensor integrator.

We discretize the u_i space-like variables of (5.40) on a uniform box $[-1.3, 1.3]^N$ with $n_i = 64$ points per axis. We replace the integral with a trapezoidal rule

cumulative summation formula and the partial derivatives with centered second order finite difference matrices. For the initial condition, we set the CDF to be a multivariate Gaussian with standard deviation $\sigma = 0.25$. In our numerical examples, the value of the PDF on the boundary of the computational domain is less than 10^{-48}, and so we treat it to be numerically zero. We then compute the CDF using numerical integration of the TT tensor and normalize by the value in the entry $\boldsymbol{F}_0[n_1, n_2, \ldots, n_N]$. This normalization is also performed after each iteration of the temporal integrators as well.

With this prototype equation fully discretized, we perform a comparison of time-stepping methods to verify temporal accuracy of order 1 and order 2. To stabilize the explicit methods, we add in numerical diffusion proportional to the roundoff error in Δt to the right hand side in a similar manner to the Lax-Wendroff method [67]. In order to guarantee that we are accurately capturing second and first order errors in time, we must ensure that the semidiscrete PDE we are solving for our reference solution matches that of our step-truncation integrators. To this end, we compute the local truncation errors of each scheme by computing its difference with a Richardson extraplolation (See [50], II.4) of it to 1 order higher. I.e. we take two steps of size $\Delta t/2$ and combine it with a step of size Δt to create a more accurate estimate for the purposes of comparing it to a single step of size Δt. To quantify transient error, we compare our methods with an empirically obtained CDF Monte-Carlo simulation and a Kernel Density Estimator (KDE)[17]. To generate the data for Monte-Carlo, we run 5×10^6 simluations of equation (5.38) with second order accuracy in space and 8 points. The initial conditions are sampled from the same distribution described above. The results of this numerical experiment are captured in the right side of Figure 5.24. We compare the error of the marginal CDF of $u(x, t)$ at $x = 0$. Time steps

were chosen as 10^{-3}. The low rank truncation error coefficients were chosen as 10^{-1} for all methods.

The norm chosen for low-rank truncation error control was the root-mean-square (RMS) tensor norm. This is achieved by weighting every TT core $\boldsymbol{C}_i$ by $\sqrt{1/64} = 1/8$ before computing any inner products or truncations. TT-GMRES stopping tolerance for all simulations was set as 5×10^{-6} in first order estimates and 5×10^{-7} for order 2 estimates. The second order methods perform better than either first order method, but similarly to one another in terms of accuracy and so we opt to use the explicit midpoint method in higher dimensional simulations.

For higher dimensional step-truncation simulation we study the convergence of the solutions to our CDF equations as the physical space finite difference stencil becomes finer. For step-truncation temporal evolution, we chose the explicit midpoint method due to its favorable performance in Figure 5.24. As a comparison benchmark in the higher dimensional case, we create a Monte-Carlo estimation using Fourier pseudospectral collocation to discretize the Burgers' equation in space. For the higher dimensional reference MC data, we use a periodic grid with $N = 20$ points. We again produce 5×10^6 trajectories from the same initial distribution discussed above. Storing a time snapshot of the Burgers' equation solution every temporal grid point $t_k = k\Delta t = k10^{-3}$ results in over 100 gigabytes of reference data. In contrast, our stored Tensor Train solution snapshots for all numerical experiments total less than 44 gigabytes. From this MC data we create approximations of the two-point CDFs at $(x_0, x_{N/2}) = (0, \pi)$ and one-point at $x_0 = 0$. These may be found in Figures 5.25 and 5.27 respectively. We see that all two dimensional results are in visual agreement with the Monte-Carlo estimate.

In Figure 5.27 we plot the one-point marginal CDF of a simulation with $N = 8, 12, 16, 20$ at $x = 0$. Here we can see the effects of increasing N on the numerical

solution, with the most qualitative matching with the KDE found in the higher order tensors. In Figure 5.26 we display the numerical difference between the Monte-Carlo benchmark simulation and the various step-truncation methods. We see considerable improvement of accuracy as the dimension is increased. Runtime increases considerably with dimension though. Once the solution rank surpasses 100, the $N = 20$ case requires over 45 minutes per time step with $\Delta t = 10^{-3}$ on our Intel i9-7980xe workstation. This indicates that high dimensionality continues to be a major barrier to accurate simulations. An alternative would be to discretize the operators $D_{ij}^{(k)}$ acting on physical space with high accuracy approximation, such as the pseudospectral method used in the KDE benchmark data. However, this raises the number of nonzero entries in the operator and increases computation time by considerably raising the rank of the operator on the RHS of (5.40). In the case of a psedudospectral approximation, we go from linearly many nonzero terms to over N^2 nonzero terms in the right side of (5.40).

We now make a remark on memory savings in the largest dimensional cases. The TT numerical solutions require at most 690 megabytes, or 6.9×10^8 bytes, per time step in the highest rank cases we have studied. This constitutes compressing a multidimensional double precision floating point array of size 64^{20}, which would normally contain approximately 10^{22} petabytes, or 10^{37} bytes. That is to say that we capture $O(10^{-2})$ accuracy of the data while achieving a data compression ratio of $1.44 \times 10^{26}\%$ and data space savings of approximately 100%. Of course, 690 megabytes is not negligible on practical computing systems. It is essentially nothing compared to the decompressed data though.

We now discuss structure preservation of the solutions. Though our schemes are qualitatively accurate to an empirically obtained solution and convergence is supported by the above theoretical and numerical investigations, close inspec-

tion of Figure 5.27 and the TT simulation data shows that the methods are not monotonicity preserving.

Implementation of numerical tensor methods for very high dimensional problems also have a number of technical complexities. In particular, the number of entries of the decompressed tensor train grow exponentially. While this is no problem for the storage of the TT cores or the representation of the tensors, it does pose numerical problems when computing the floating point arithmetic. The RMS norm, for instance, is both required to avoid floating point overflow (since the unweighted norm is typically larger than the inverse of floating point precision) and also needs careful formulation to avoid underflow (since the weight n^{-N} is typically smaller than machine precision). Similar consideration is also required when formulating the TT-GMRES solver.

Chapter 6

Conclusion

We have presented step-truncation methods for the solutions to initial value problems on tensor manifolds. These methods are centered around the concept of stepping off of a tensor manifold with a conventional time-stepping scheme, then truncating the solution tensor back to a lower rank. The mathematical framework developed here is centered around analyzing semi-discrete PDEs. As is common in numerical analysis, we rely heavily on concepts from finite dimensional analysis to prove statements about our algorithms. We are able to provide guarantees of types of both stability and convergence. In particular, we show that it is possible to produce step-truncation methods of any order in time. The analysis is carried out for both explicit and implicit methods. To this end we developed an algorithm based on Newton's method for the solution to nonlinear equations on tensor manifolds.

A number of numerical experiments are provided as well, verifying the proven convergence rates. We provide applications to advection problems, Fokker-Planck equations, the Nonlinear Schrödinger equation, and a Burgers' equation with uncertain initial condition. All applications indicate that these methods have favorable accuracy and stability. However, no schemes are proven to be structure

preserving. Moreover, we are essentially following a method-of-lines style of analysis. In essence, we prove that if one has a tensor valued ODE, then that ODE may be approximated via step-truncation method with whatever accuracy is desired.

Further analysis of step-truncation integrators should include the combination of low-rank truncation with the space-discretization as well as the temporal discretization. Such an analysis would quantify how the roundoff error in space would relate to the low-rank truncation error. This would possibly shine light on the ability to construct structure preserving integrators fitting into the convergence theorems presented in this thesis.

Appendix A

Proof of Lemma 1

In this section, we present a proof of Lemma 1 which is specific to $\mathcal{H}_{\boldsymbol{r}}$. First, we start by constructing an open set centered about a point with known rank.

Lemma 5. *Let $\boldsymbol{f} \in \mathcal{H}_{\boldsymbol{r}}$ be a point on the hierarchical Tucker manifold of constant rank. Let $\boldsymbol{v} \in T_{\boldsymbol{f}}\mathcal{H}_{\boldsymbol{r}}$ be an arbitrary vector in the tangent plane of $\mathcal{H}_{\boldsymbol{r}}$ at $\boldsymbol{f}$. Then there exists $\eta > 0$ such that for all ε satisfying $0 \leq \varepsilon \leq \eta$, we have $\boldsymbol{f} + \varepsilon\boldsymbol{v} = \boldsymbol{g} \in \mathcal{H}_{\boldsymbol{r}}$. As a consequence, if $U_{\boldsymbol{f}} \subseteq T_{\boldsymbol{f}}\mathcal{H}_{\boldsymbol{r}}$ is a closed and bounded set containing the origin, then there exists an open subset $V_{\boldsymbol{f}} \subseteq U_{\boldsymbol{f}}$ such that $\boldsymbol{f} + \boldsymbol{h} \in \mathcal{H}_{\boldsymbol{r}}$, for all $\boldsymbol{h} \in V_{\boldsymbol{f}}$.*

Proof. First, consider a simpler problem, in which we have two matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times m}$, where $\boldsymbol{A}$ is full column rank. Consider the function

$$p(\eta) = \det\left((\boldsymbol{A} + \eta\boldsymbol{B})^{\top}(\boldsymbol{A} + \eta\boldsymbol{B})\right). \tag{A.1}$$

Clearly, $p(\eta)$ is a polynomial and thus smooth in η. Moreover, $p(0) \neq 0$ since $\boldsymbol{A}$ is full column rank. Since p is smooth, there exists some $\eta > 0$ such that $p(\varepsilon) \neq 0$ for all $\varepsilon \in [0, \eta]$. Since the full-rank hierarchical Tucker manifold is defined via the full column rank constraints on an array of matrices corresponding to matricizations of

the tensor [107], we can apply the principle above to every full column rank matrix associated with the tree, using addition of a point and a tangent as referenced in Proposition 3 of [29]. We have now proved the part one of the lemma where η is taken to be the minimum over the tree nodes. As for existence of an open set, suppose $U_{\boldsymbol{f}}$ is open and bounded. Now we apply the above matrix case to the boundary $\partial U_{\boldsymbol{f}}$, giving us a star shaped set $S_{\boldsymbol{f}} \subseteq U_{\boldsymbol{f}}$. Letting $V_{\boldsymbol{f}} = S_{\boldsymbol{f}} \setminus \partial S_{\boldsymbol{f}}$ be the interior, completes the proof of the lemma. □

We use the open set constructed above to prove smoothness using the same techniques as [76].

Proof. (Lemma 1) Let $\boldsymbol{f} \in \mathcal{H}_{\boldsymbol{r}} \subseteq \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$. By Lemma 5, there exists an open norm-ball $B(\boldsymbol{f}, \kappa)$ located at $\boldsymbol{f}$ with radius $\kappa > 0$ so that

$$\boldsymbol{f} + \boldsymbol{v} \in \mathcal{H}_{\boldsymbol{r}} \qquad \forall \boldsymbol{v} \in \mathcal{P}_{\boldsymbol{f}} \left(B(\boldsymbol{f}, \kappa) \right). \tag{A.2}$$

Let $\mathcal{U}_{\boldsymbol{f}} = \mathcal{H}_{\boldsymbol{r}} \cap B(\boldsymbol{f}, \kappa)$ be a set which is open in the topology of $\mathcal{H}_{\boldsymbol{r}}$. Also, let $(\boldsymbol{q}_{\boldsymbol{f}}, \boldsymbol{q}_{\boldsymbol{f}}^{-1}(\mathcal{U}_{\boldsymbol{f}}))$ be a local parametrization at $\boldsymbol{f}$. For the parametrizing coordinates, we take an open subset $\boldsymbol{q}_{\boldsymbol{f}}^{-1}(\mathcal{U}_{\boldsymbol{f}}) = V_{\boldsymbol{f}} \subseteq T_{\boldsymbol{f}}\mathcal{H}_{\boldsymbol{r}}$ of the tangent space embedded in $\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$. This means that the parametrization $\boldsymbol{q}_{\boldsymbol{f}}$ takes tangent vectors as inputs and maps them into tensors in $\mathcal{H}_{\boldsymbol{r}}$, i.e.

$$\boldsymbol{q}_{\boldsymbol{f}} : T_{\boldsymbol{f}}\mathcal{H}_{\boldsymbol{r}} \to \mathcal{H}_{\boldsymbol{r}}. \tag{A.3}$$

Moreover, we assume that the coordinates are arranged in column major ordering as a vector. This allows for the Jacobian $\partial \boldsymbol{q}_{\boldsymbol{f}} / \partial \boldsymbol{v}$ to be a basis for the tangent space $T_{\boldsymbol{f}}\mathcal{H}_{\boldsymbol{r}}$. Note that $\partial \boldsymbol{q}_{\boldsymbol{f}} / \partial \boldsymbol{v}$ is a $(n_1 n_2 \cdots n_d) \times \dim(T_{\boldsymbol{f}}\mathcal{H}_{\boldsymbol{r}})$ matrix with real coefficients. Now, let $\boldsymbol{M}(\boldsymbol{f})$ be a matrix of column vectors spanning the space orthogonal to $T_{\boldsymbol{f}}\mathcal{H}_{\boldsymbol{r}}$ in $\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$. Since the two linear spaces are disjoint, we

have a local coordinate map for the ball $B(\boldsymbol{f}, \kappa)$, given by

$$\boldsymbol{C}(\boldsymbol{v}, \boldsymbol{g}) = \boldsymbol{q_f}(\boldsymbol{v}) + [\boldsymbol{M}(\boldsymbol{q_f}(\boldsymbol{v}))]\boldsymbol{g}, \tag{A.4}$$

where $\boldsymbol{v}$ is tangent and $\boldsymbol{g}$ is normal (both column vectors). By construction,

$$\mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{C}(\boldsymbol{v}, \boldsymbol{g})) = \boldsymbol{q_f}(\boldsymbol{v}) \tag{A.5}$$

is smooth in both $\boldsymbol{v}$ and $\boldsymbol{g}$. Therefore, we can take the total derivative on the embedded space and apply the chain rule to obtain the Jacobian of $\mathfrak{T}_{\boldsymbol{r}}^{\text{best}}(\boldsymbol{f})$. Doing so, we have

$$\begin{aligned}\frac{\partial}{\partial(\boldsymbol{v}, \boldsymbol{g})}\mathfrak{T}_{\boldsymbol{r}}^{best}(\boldsymbol{C}(\boldsymbol{v}, \boldsymbol{g})) &= \frac{\partial \mathfrak{T}_{\boldsymbol{r}}^{\text{best}}}{\partial \boldsymbol{C}}\frac{\partial \boldsymbol{C}}{\partial(\boldsymbol{v}, \boldsymbol{g})} \\ &= \frac{\partial \mathfrak{T}_{\boldsymbol{r}}^{\text{best}}}{\partial \boldsymbol{C}}\left[\frac{\partial \boldsymbol{q_f}}{\partial \boldsymbol{v}} + \sum_{i=1}^{n^{\perp}} \frac{\partial \boldsymbol{M}_i(\boldsymbol{q_f}(\boldsymbol{v}))}{\partial \boldsymbol{v}} g_i \middle| \boldsymbol{M}(\boldsymbol{q_f}(\boldsymbol{v}))\right], \end{aligned} \tag{A.6}$$

where the symbol $[\cdot|\cdot]$ denotes column concatenation of matrices, $n^{\perp}$ is the dimension of the normal space $(T_{\boldsymbol{f}}\mathcal{H}_{\boldsymbol{r}})^{\perp}$, $\boldsymbol{M}_i$ is the i-th column of $\boldsymbol{M}$, and g_i is the i-th component of $\boldsymbol{g}$. We can take $\boldsymbol{g} = \boldsymbol{0}$ since the above expression extends smoothly from the embedding space onto $\mathcal{H}_{\boldsymbol{r}}$. Hence, the Jacobian of $\mathfrak{T}^{\text{best}}$ is the solution to the linear equation

$$\frac{\partial \mathfrak{T}_{\boldsymbol{r}}^{\text{best}}}{\partial \boldsymbol{C}}\left[\frac{\partial \boldsymbol{q_f}}{\partial \boldsymbol{v}} \middle| \boldsymbol{M}(\boldsymbol{q_f}(\boldsymbol{v}))\right] = \left[\frac{\partial \boldsymbol{q_f}}{\partial \boldsymbol{v}} \middle| \boldsymbol{0}\right]. \tag{A.7}$$

Since the right factor of the left hand side has a pair of orthogonal blocks, we can

write the inverse using the pseudo-inverse of the blocks, i.e.,

$$\left[\frac{\partial \boldsymbol{q_f}}{\partial \boldsymbol{v}}\middle| \boldsymbol{M}(\boldsymbol{q_f}(\boldsymbol{v}))\right]^{-1} = \begin{bmatrix} \left[\frac{\partial \boldsymbol{q_f}}{\partial \boldsymbol{v}}\right]^{+} \\ \\ [\boldsymbol{M}(\boldsymbol{q_f}(\boldsymbol{v}))]^{+} \end{bmatrix}. \tag{A.8}$$

The right hand side is the block concatenation of the rows of each pseudo-inverse. Plugging the above expression into (A.7), we find

$$\frac{\partial \mathfrak{T}_r^{\text{best}}}{\partial \boldsymbol{C}} = \frac{\partial \boldsymbol{q_f}}{\partial \boldsymbol{v}} \left[\frac{\partial \boldsymbol{q_f}}{\partial \boldsymbol{v}}\right]^{+}, \tag{A.9}$$

which is exactly the expression for the orthogonal projection onto the tangent space [74]. This completes the proof. □

Appendix B

Proof of Theorem 4

Before proceeding, we state the following Lemma for use in the proof of the theorem.

Lemma 6. *Let* $0 < h \leq t < 1$. *Then there exists* $0 < e \leq h$ *so that*

$$(1+e)[h(1+e)+2e] \leq t.$$

This follows from

$$(1+e)[h(1+e)+2e] = h + (1+h)2e + (1+h)2e^2 \leq h + 4e + 4e^2,$$

which is bound by t *if* $4e + 4e^2 \leq M = \min(h, t-h)$. *I.e., whenever*

$$e \leq \frac{-1+\sqrt{1+M}}{2}.$$

We also recall the definition of a superlinearly convergent sequence. This definition is useful in the proof of Theorem 4.

Definition 2. *A sequence* $\boldsymbol{f}^{[j]}$ *converges superlinearly to* $\boldsymbol{f}^*$ *if there exists a se-*

quence of real numbers $\tau_j \to 0$ *as* $j \to \infty$ *such that*

$$\|\boldsymbol{f}^{[j+1]} - \boldsymbol{f}^*\| \leq \tau_j \|\boldsymbol{f}^{[j]} - \boldsymbol{f}^*\|. \tag{B.1}$$

Clearly, such a sequence is convergent since there exists an iteration number k so that $\tau_j < \alpha = 1/2$ for any $j > k$, so the distance from the point $\boldsymbol{f}^*$ shrinks at least as fast as $O(\alpha^j)$ as $j \to \infty$. The same claim can be made for any $0 < \alpha < 1$, and so the sequence converges faster than any linear recurrence relation. We now restate the theorem before proving it.

Theorem (Inexact Newton method [34])**.** *Let* $\boldsymbol{H} : \mathbb{R}^N \to \mathbb{R}^N$ *be continuously differentiable in a neighborhood of a zero* $\boldsymbol{f}^*$*, and suppose that the Jacobian of* $\boldsymbol{H}$*, i.e.,* $\boldsymbol{J}_{\boldsymbol{H}}(\boldsymbol{f}) = \partial \boldsymbol{H}(\boldsymbol{f})/\partial \boldsymbol{f}$*, is invertible at* $\boldsymbol{f}^*$*. Given* $\boldsymbol{f}^{[0]} \in \mathbb{R}^N$*, consider the sequence*

$$\boldsymbol{f}^{[j+1]} = \boldsymbol{f}^{[j]} + \boldsymbol{s}^{[j]}, \qquad j = 0, 1, \ldots \tag{B.2}$$

where each $\boldsymbol{s}^{[j]}$ *solves the Newton iteration up to relative error* $\eta^{[j]}$*, i.e., it satisfies*

$$\left\| \boldsymbol{J}_{\boldsymbol{H}}\left(\boldsymbol{f}^{[j]}\right) \boldsymbol{s}^{[j]} + \boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right) \right\| \leq \left\| \boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right) \right\| \eta^{[j]}. \tag{B.3}$$

If $\eta^{[j]} \leq \tau < 1$ *for all* j*, then there exists* $\varepsilon > 0$ *so that for any initial guess satisfying* $\left\| \boldsymbol{f}^{[0]} - \boldsymbol{f}^* \right\| < \varepsilon$*, the sequence* $\{\boldsymbol{f}^{[j]}\}$ *converges linearly to* $\boldsymbol{f}^*$*. If in addition,* $\eta^{[j]} \to 0$ *as* $j \to \infty$*, then the convergence speed is superlinear.*

Proof. The proof will come in two segments. We first analyze the effect of a single step of the iteration, proving that the convergence speed is at least linear. In the second segment, we show superlinear convergence.

Let $\boldsymbol{J}^* = \boldsymbol{J}_{\boldsymbol{H}}(\boldsymbol{f}^*)$ and $\boldsymbol{J}^{[k]} = \boldsymbol{J}_{\boldsymbol{H}}(\boldsymbol{f}^{[k]})$ Since $\boldsymbol{J}^*$ is invertible, we have the

following from equivalence-of-norms

$$\mu^{-1}\|\boldsymbol{g}\| \leq \|\boldsymbol{J}^*\boldsymbol{g}\| \leq \mu\|\boldsymbol{g}\| \tag{B.4}$$

with $\mu = \max(\|\boldsymbol{J}^*\|, \|(\boldsymbol{J}^*)^{-1}\|)$. Since $\eta^{[0]} \leq \tau_0 = \tau$, by Lemma 6 there exists $\gamma_0 \leq \mu^{-1}\eta^{[0]}$ so that

$$(1+\mu\gamma_0)[\eta^{[0]}(1+\mu\gamma_0) + 2\mu\gamma_0] \leq \tau_0$$

We now begin the base case of an inductive argument. We constrain the tolerance of the initial guess $\boldsymbol{f}^{[0]}$. Let $\varepsilon > 0$ be sufficiently small so that for all $\boldsymbol{g}$, satisfying $\|\boldsymbol{g} - \boldsymbol{f}^*\| \leq \mu^2\varepsilon$ we have

$$\|\boldsymbol{J}_{\boldsymbol{H}}(\boldsymbol{g}) - \boldsymbol{J}_{\boldsymbol{H}}(\boldsymbol{f}^*)\| \leq \gamma_0, \tag{B.5}$$

$$\left\|\boldsymbol{J}_{\boldsymbol{H}}(\boldsymbol{g})^{-1} - \boldsymbol{J}_{\boldsymbol{H}}(\boldsymbol{f}^*)^{-1}\right\| \leq \gamma_0, \tag{B.6}$$

$$\|\boldsymbol{H}(\boldsymbol{g}) - \boldsymbol{J}_{\boldsymbol{H}}(\boldsymbol{f}^*)(\boldsymbol{g} - \boldsymbol{f}^*)\| \leq \gamma_0 \|\boldsymbol{g} - \boldsymbol{f}^*\|. \tag{B.7}$$

Inequality (B.5) may be satisfied by assumption of continuous differentiability. Inequality (B.6) follows from continuity of the inverse Jacobian guaranteed via the inverse function theorem. Inequality (B.7) is an appeal to the definition of the operator derivative at the root $\boldsymbol{f}^*$ where $\boldsymbol{H}(\boldsymbol{f}^*) = \boldsymbol{0}$. Assume the initial guess satisfies $\|\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\| \leq \varepsilon \cdot \min(1, \mu^2)$. We now analyze the effect of the first step of the iteration. We can expand one iteration of the scheme's residual in terms of (B.5) through (B.7). Doing so yields

$$\begin{aligned}\boldsymbol{J}^*\left(\boldsymbol{f}^{[1]} - \boldsymbol{f}^*\right) = \Big(\boldsymbol{I} + \boldsymbol{J}^*\left(\left(\boldsymbol{J}^{[0]}\right)^{-1} - \left(\boldsymbol{J}^*\right)^{-1}\right)\Big)\Big(\boldsymbol{r}^{[0]} + \left(\boldsymbol{J}^{[0]} - \boldsymbol{J}^*\right)\left(\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\right) \\ - \boldsymbol{H}\left(\boldsymbol{f}^{[0]}\right) + \boldsymbol{J}^*(\boldsymbol{f}^{[0]} - \boldsymbol{f}^*)\Big),\end{aligned} \tag{B.8}$$

where $\boldsymbol{r}^{[j]} = \boldsymbol{J}_{\boldsymbol{H}}\left(\boldsymbol{f}^{[j]}\right)\boldsymbol{s}^{[j]} + \boldsymbol{H}\left(\boldsymbol{f}^{[j]}\right)$ is the residual in the linear solve step of the Newton iteration. Note that a similar relationship holds between step $\boldsymbol{f}^{[j+1]}$ and $\boldsymbol{f}^{[j]}$. In the $\boldsymbol{J}^*$ norm, denoted $\|\cdot\|_*$, we have

$$\left\|\boldsymbol{f}^{[1]} - \boldsymbol{f}^*\right\|_* \leq \left(1 + \|\boldsymbol{J}^*\| \left\|\left(\boldsymbol{J}^{[0]}\right)^{-1} - \left(\boldsymbol{J}^*\right)^{-1}\right\|\right)\left(\left\|\boldsymbol{r}^{[0]}\right\| + \left\|\boldsymbol{J}^{[0]} - \boldsymbol{J}^*\right\|\left\|\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\right\| + \left\|\boldsymbol{H}\left(\boldsymbol{f}^{[0]}\right) - \boldsymbol{J}^*(\boldsymbol{f}^{[0]} - \boldsymbol{f}^*)\right\|\right). \tag{B.9}$$

By substituting in the relative accuracy bound $\|\boldsymbol{r}^{[0]}\| \leq \eta^{[0]}\left\|\boldsymbol{H}\left(\boldsymbol{f}^{[0]}\right)\right\|$ and appealing to the three continuity conditions (B.5) through (B.7) as well as the operator norm μ, we have

$$\left\|\boldsymbol{f}^{[1]} - \boldsymbol{f}^*\right\|_* \leq (1 + \mu\gamma_0)\left(\eta^{[0]}\left\|\boldsymbol{H}\left(\boldsymbol{f}^{[0]}\right)\right\| + 2\gamma_0\left\|\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\right\|\right) \tag{B.10}$$

The residual $\left\|\boldsymbol{H}\left(\boldsymbol{f}^{[0]}\right)\right\|$ may be bound in terms of condition (B.7), through an appeal to the triangle inequality. Specifically,

$$\begin{aligned}\left\|\boldsymbol{H}\left(\boldsymbol{f}^{[0]}\right)\right\| &\leq \left\|\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\right\|_* + \left\|\boldsymbol{H}(\boldsymbol{f}^{[0]}) - \boldsymbol{J}^*(\boldsymbol{f}^{[0]} - \boldsymbol{f}^*)\right\| \\ &\leq \left\|\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\right\|_* + \gamma_0\left\|\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\right\|.\end{aligned} \tag{B.11}$$

Combining (B.10) and (B.11) then appealing to the equivalence principle (B.4) we have the bound

$$\left\|\boldsymbol{f}^{[1]} - \boldsymbol{f}^*\right\|_* \leq (1 + \mu\gamma_0)\left(\eta^{[0]}(1 + \gamma_0\mu) + 2\gamma_0\mu\right)\left\|\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\right\|_*. \tag{B.12}$$

The coefficient of the norm on the right side is bound by $\tau < 1$. Therefore

$$\left\|\boldsymbol{f}^{[1]} - \boldsymbol{f}^*\right\| \leq \tau\mu\left\|\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\right\|_* \leq \mu^2\tau\left\|\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\right\| < \mu^2\left\|\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\right\| \leq \mu^2\varepsilon.$$

Applying the inductive hypothesis $\left\|\boldsymbol{f}^{[j]} - \boldsymbol{f}^*\right\|_* \leq \tau^j \left\|\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\right\|_*$ we find

$$\left\|\boldsymbol{f}^{[j+1]} - \boldsymbol{f}^*\right\| \leq \tau\mu \left\|\boldsymbol{f}^{[j]} - \boldsymbol{f}^*\right\|_* \leq \mu\tau^{j+1} \left\|\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\right\|_* \leq \mu^2\tau^{j+1} \left\|\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\right\|.$$

Thus the iteration converges to the root at rate $O(\tau^j)$ as $j \to \infty$.

We now begin the second segment, in which we show that the rate is superlinear so long as $\eta^{[j]} \to 0$. We define the following quantities

$$\left\|\boldsymbol{J_H}(\boldsymbol{f}^{[j]}) - \boldsymbol{J_H}(\boldsymbol{f}^*)\right\| = \alpha_j, \tag{B.13}$$

$$\left\|\boldsymbol{J_H}(\boldsymbol{f}^{[j]})^{-1} - \boldsymbol{J_H}(\boldsymbol{f}^*)^{-1}\right\| = \beta_j, \tag{B.14}$$

$$\frac{\left\|\boldsymbol{H}(\boldsymbol{f}^{[j]}) - \boldsymbol{J_H}(\boldsymbol{f}^*)(\boldsymbol{f}^{[j]} - \boldsymbol{f}^*)\right\|}{\|\boldsymbol{f}^{[j]} - \boldsymbol{f}^*\|} = \kappa_j. \tag{B.15}$$

Since the sequence of $\boldsymbol{f}^{[j]}$ is convergent, continuity implies $\alpha_j \to 0$ and $\beta_j \to 0$ as $j \to \infty$. Additionally, the definition of the derivative requires that $\kappa_j \to 0$, as the higher order error terms must be $o\left(\left\|\boldsymbol{f}^{[j]} - \boldsymbol{f}^*\right\|\right)$ as $j \to \infty$. By inserting these quantities into the inequality (B.9) for step j to $j+1$ we have

$$\left\|\boldsymbol{f}^{[j+1]} - \boldsymbol{f}^*\right\|_* \leq (1 + \mu\beta_j)\left(\eta^{[j]} \left\|\boldsymbol{H}(\boldsymbol{f}^{[j]})\right\| + (\alpha_j + \kappa_j) \left\|\boldsymbol{f}^{[j]} - \boldsymbol{f}^*\right\|\right).$$

Now we appeal to (B.11) to arrive at $\left\|\boldsymbol{H}(\boldsymbol{f}^{[j]})\right\| \leq \left\|\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\right\|_* + \kappa_j \left\|\boldsymbol{f}^{[0]} - \boldsymbol{f}^*\right\|$. The error bound becomes

$$\begin{aligned}\left\|\boldsymbol{f}^{[j+1]} - \boldsymbol{f}^*\right\|_* &\leq (1 + \mu\beta_j)\left(\eta^{[j]}(1 + \mu\kappa_j) + \mu(\alpha_j + \kappa_j)\right) \left\|\boldsymbol{f}^{[j]} - \boldsymbol{f}^*\right\|_* \\ &= \tau_j \left\|\boldsymbol{f}^{[j]} - \boldsymbol{f}^*\right\|_*. \end{aligned} \tag{B.16}$$

When $\eta^{[j]} \to 0$, $\tau_j = (1 + \mu\beta_j)\left(\eta^{[j]}(1 + \mu\kappa_j) + \mu(\alpha_j + \kappa_j)\right)$ converges to zero as $j \to \infty$, thereby satisfying Definition 2 in the $\boldsymbol{J}^*$ norm. □

Appendix C

Step-truncation methods for matrix-valued ODEs on matrix manifolds with fixed rank

To make Lemma 1 concrete, in this Appendix we write down $\mathfrak{T}_{\boldsymbol{r}}^{\text{best}}$ and its Jacobian $\mathcal{P}_{\boldsymbol{f}}$ for problems where $\boldsymbol{f} \in \mathbb{R}^{n_1 \times n_2}$ is a matrix. In this situation, the tree rank $\boldsymbol{r}$ is just a single integer r. One can see from the accuracy inequalities for best truncation proven in [45] that the $\mathfrak{T}_r^{\text{best}}$ is obtained from truncating the smallest $\min(n_1, n_2) - r$ singular values and singular vectors. For simplicity, we will write down the best truncation scheme for (1.2) using the Euler forward method. This gives

$$\boldsymbol{f}_{k+1} = \mathfrak{T}_r^{\text{best}}(\boldsymbol{f}_k + \Delta t \boldsymbol{G}(\boldsymbol{f}_k)). \tag{C.1}$$

Assuming that we are fixing rank to be the same as the initial condition for all k, we have that $\mathfrak{T}_r^{\text{best}}(\boldsymbol{f}_k) = \boldsymbol{f}_k$. Now we can apply SVD perturbation theory [71, 100] to express the best truncation operator in terms of a power series expansion in Δt. Representing our decomposition as a tuple of matrices $(\boldsymbol{\Sigma}_k, \boldsymbol{Q}_k, \boldsymbol{V}_k)$, where

$\boldsymbol{f}_k = \boldsymbol{Q}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^\top$ is the reduced singular value decomposition, we have that

$$\boldsymbol{\Sigma}_{k+1} = \boldsymbol{\Sigma}_k + \Delta t \mathrm{diag}(\boldsymbol{Q}_k^\top \boldsymbol{G}(\boldsymbol{f}_k)\boldsymbol{V}_k) + O(\Delta t^2), \tag{C.2}$$

$$\begin{aligned}\boldsymbol{Q}_{k+1} = \boldsymbol{Q}_k &+ \Delta t \boldsymbol{Q}_k(\boldsymbol{H}_k \odot (\boldsymbol{Q}_k^\top \boldsymbol{G}(\boldsymbol{f}_k)\boldsymbol{V}_k\boldsymbol{\Sigma}_k + \boldsymbol{\Sigma}_k\boldsymbol{V}_k^\top \boldsymbol{G}(\boldsymbol{f}_k)^\top \boldsymbol{Q}_k)) \\ &+ \Delta t(\boldsymbol{I} - \boldsymbol{Q}_k\boldsymbol{Q}_k^\top)\boldsymbol{G}(\boldsymbol{f}_k)\boldsymbol{V}_k\boldsymbol{\Sigma}_k^{-1} + O(\Delta t^2),\end{aligned} \tag{C.3}$$

$$\begin{aligned}\boldsymbol{V}_{k+1} = \boldsymbol{V}_k &+ \Delta t \boldsymbol{V}_k(\boldsymbol{H}_k \odot (\boldsymbol{\Sigma}_k\boldsymbol{Q}_k^\top \boldsymbol{G}(\boldsymbol{f}_k)\boldsymbol{V}_k + \boldsymbol{V}_k^\top \boldsymbol{G}(\boldsymbol{f}_k)^\top \boldsymbol{Q}_k\boldsymbol{\Sigma}_k)) \\ &+ \Delta t(\boldsymbol{I} - \boldsymbol{V}_k\boldsymbol{V}_k^\top)\boldsymbol{G}(\boldsymbol{f}_k)^\top\boldsymbol{Q}_k\boldsymbol{\Sigma}_k^{-1} + O(\Delta t^2).\end{aligned} \tag{C.4}$$

Here, $\odot$ denotes the element-wise (Hadamard) product of matrices, and the matrix

$$\begin{cases}\boldsymbol{H}_k[i,j] = 1/(\boldsymbol{\Sigma}_k[j,j]^2 - \boldsymbol{\Sigma}_k[i,i]^2), & i \neq j, \\ \boldsymbol{H}_k[i,j] = 0, & i = j,\end{cases} \tag{C.5}$$

is skew-symmetric and stores information about the differences of the singular values. The diag$(\cdot)$ operation zeros out all elements off of the diagonal. The tangent space projection operator is the coefficient of the Δt terms. From here, we can see that the evolution equation corresponding to (C.1) is

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\Sigma} = \mathrm{diag}(\boldsymbol{Q}^\top \boldsymbol{G}(\boldsymbol{f})\boldsymbol{V}), \tag{C.6}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{Q} = \boldsymbol{Q}(\boldsymbol{H} \odot (\boldsymbol{Q}^\top \boldsymbol{G}(\boldsymbol{f})\boldsymbol{V}\boldsymbol{\Sigma} + \boldsymbol{\Sigma}\boldsymbol{V}^\top \boldsymbol{G}(\boldsymbol{f})^\top \boldsymbol{Q})) + (\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^\top)\boldsymbol{G}(\boldsymbol{f})\boldsymbol{V}\boldsymbol{\Sigma}^{-1}, \tag{C.7}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{V} = \boldsymbol{V}(\boldsymbol{H} \odot (\boldsymbol{\Sigma}\boldsymbol{Q}^\top \boldsymbol{G}(\boldsymbol{f})\boldsymbol{V} + \boldsymbol{V}^\top \boldsymbol{G}(\boldsymbol{f})^\top \boldsymbol{Q}\boldsymbol{\Sigma})) + (\boldsymbol{I} - \boldsymbol{V}\boldsymbol{V}^\top)\boldsymbol{G}(\boldsymbol{f})^\top\boldsymbol{Q}\boldsymbol{\Sigma}^{-1}. \tag{C.8}$$

By setting $\boldsymbol{U} = \boldsymbol{Q}\boldsymbol{\Sigma}$ it can be verified that the pair $(\boldsymbol{U}, \boldsymbol{V})$ satisfy the dynamically bi-orthogonal equations of [23]. It should be noted that this is not the only parametrization of the fixed-rank solution $\boldsymbol{f} = \boldsymbol{Q}\boldsymbol{\Sigma}\boldsymbol{V}^\top$. Of particular interest is

the closely related projection method given by the DDO approximation

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{A} = \boldsymbol{W}^\top \boldsymbol{G}(\boldsymbol{f})\boldsymbol{B}, \tag{C.9}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{W} = (\boldsymbol{I} - \boldsymbol{W}\boldsymbol{W}^\top)\boldsymbol{G}(\boldsymbol{f})\boldsymbol{B}\boldsymbol{A}^{-1}, \tag{C.10}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{B} = (\boldsymbol{I} - \boldsymbol{B}\boldsymbol{B}^\top)\boldsymbol{G}(\boldsymbol{f})^\top \boldsymbol{W}\boldsymbol{A}^{-\top}. \tag{C.11}$$

The above equations are equivalent to the SVD equations in the sense that

$$\boldsymbol{W}(t)\boldsymbol{A}(t)\boldsymbol{B}^\top(t) = \boldsymbol{f}(t) = \boldsymbol{Q}(t)\boldsymbol{\Sigma}(t)\boldsymbol{V}^\top(t) \tag{C.12}$$

as long as the singular values are distinct and the equation holds at $t = 0$. A comparison of methods for fixed rank initial value problems is given in [79].

Bibliography

[1] P-A Absil and Jérôme Malick. Projection-like retractions on matrix manifolds. *SIAM J. Optim.*, 22(1):135–158, 2012.

[2] P-A Absil and Ivan V Oseledets. Low-rank retractions: a survey and new results. *Computational Optimization and Applications*, 62(1):5–29, 2015.

[3] S. M. Allen and J. W. Cahn. Ground state structures in ordered binary alloys with second neighbor interactions. *Acta Metall.*, 20(3):423–433, 1972.

[4] S. M. Allen and J. W. Cahn. A correction to the ground state of FCC binary ordered alloys with first and second neighbor pairwise interactions. *Scripta Metallurgica*, 7(12):1261–1264, 1973.

[5] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[6] W. Austin, G. Ballard, and T. G. Kolda. Parallel tensor compression for large-scale scientific data. In *IPDPS'16: Proceedings of the 30th IEEE International Parallel and Distributed Processing Symposium*, pages 912–922, May 2016.

[7] M. Bachmayr, R. Schneider, and A. Uschmajew. Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. *Found. of 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] C. Beck, W. E, and A. Jentzen. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *J. Nonlinear Sci.*, 2019.

[11] R. E. Bellman. *Dynamic programming*. Princeton University Press, 1957.

[12] M. J. Beran. *Statistical continuum theories*. New York: Interscience Publishers, 1968.

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

[14] A. M. P. Boelens, D. Venturi, and D. M. Tartakovsky. Tensor methods for the Boltzmann-BGK equation. *J. Comput. Phys.*, 421:109744, 2020.

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

[16] R. Borrelli and S. Dolgov. Expanding the range of hierarchical equations of motion by tensor-train implementation. *The Journal of Physical Chemistry B*, 2021.

[17] ZI Botev, JF Grotowski, and DP Kroese. Kernel density estimation via diffusion. *The Annals of Statistics*, pages 2916–2957, 2010.

[18] H. J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

[19] 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.

[20] R. Carmona and F. Delarue. *Probabilistic theory of mean field games with applications II*. Springer, 2018.

[21] G. Ceruti and C. Lubich. An unconventional robust integrator for dynamical low-rank approximation. *ArXiv*, (2010.02022):1–19, 2020.

[22] Y. Chen, L. Zhang, H. Wang, and W. E. Ground state energy functional with Hartree-Fock efficiency and chemical accuracy. *J. Phys. Chem. A*, 124(35), 2020.

[23] M. Cheng, T. Y. Hou, 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.

[24] A. Chertkov and I. Oseledets. Solution of the Fokker-Planck equation by cross approximation method in the tensor train format. *arXiv preprint arXiv:2102.08143*, 2021.

[25] F. Chinesta, R. Keunings, and A. Leygue. *The Proper generalized decomposition for advanced numerical simulations*. Springer, 2014.

[26] 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.

[27] H. Cho, D. Venturi, and G.E. Karniadakis. Numerical methods for high-dimensional probability density function equations. *J. Comput. Phys.*, 305:817–837, January 2016.

[28] H. Cho D. Venturi, T. P. Sapsis and G. E. Karniadakis. A computable evolution equation for the joint response-excitation probability density function of stochastic dynamical systems. *Proc. R. Soc. A*, 468:759–783, 2011.

[29] Curt Da Silva and Felix J Herrmann. Optimization on the hierarchical Tucker manifold-applications to tensor completion. *Linear Algebra and its Appl.*, 481:131–173, 2015.

[30] H. Al Daas, G. Ballard, and P. Benner. Parallel algorithms for tensor train arithmetic. *SIAM J. Sci. Comput.*, 44(1):C25–C53, 2022.

[31] A. Dektor, A. Rodgers, and D. Venturi. Rank-adaptive tensor methods for high-dimensional nonlinear PDEs. *Journal of Scientific Computing*, 88(36):1–27, 2021.

[32] A. Dektor and D. Venturi. Dynamically orthogonal tensor methods for high-dimensional nonlinear PDEs. *J. Comput. Phys.*, 404:109125, 2020.

[33] A. Dektor and D. Venturi. Dynamic tensor approximation of high-dimensional nonlinear PDEs. *J. Comput. Phys.*, 437:110295, 2021.

[34] Ron S. Dembo, Stanley C. Eisenstat, and Trond Steihaug. Inexact newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982.

[35] James Demmel, Laura Grigori, Mark Hoemmen, and Julien Langou. Communication-optimal parallel and sequential qr and lu factorizations. *SIAM Journal on Scientific Computing*, 34(1):A206–A239, 2012.

[36] G. Di Marco and L. Pareschi. Numerical methods for kinetic equations. *Acta Numerica*, 23:369–520, May 2014.

[37] S. Dolgov. A tensor decomposition algorithm for large odes with conservation laws. *Computational Methods in Applied Mathematics*, 19(1):23–38, 2019.

[38] S. Dolgov and B. Khoromskij. Tensor-product approach to global time-space-parametric discretization of chemical master equation. 2012.

[39] S. Dolgov, B. Khoromskij, and I. Oseledets. Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the Fokker–Planck equation. *SIAM J. Sci. Comput.*, 34(6):A3016–A3038, 2012.

[40] S. V. Dolgov. TT-GMRES: solution to a linear system in the structured tensor format. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 28(2):149–172, 2013.

[41] W. E, J. Han, and Q. Li. A mean-field optimal control formulation of deep learning. *Res. Math. Sci.*, 6(10):1–41, 2019.

[42] István Faragó and Agnes Havasiy. *Operator splittings and their applications*. Nova Science Publishers, 2009.

[43] J. Foo and G. E. Karniadakis. Multi-element probabilistic collocation method in high dimensions. *J. Comput. Phys.*, 229:1536–1557, 2010.

[44] W. Gangbo, W. Li, S. Osher, and M. Puthawala. Unnormalized optimal transport. *J. Comput. Phys.*, 399:108940, 2019.

[45] L. Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM J. Matrix Anal. Appl.*, 31(4):2029–2054, 2010.

[46] L. Grasedyck and C. Löbbert. Distributed hierarchical SVD in the hierarchical Tucker format. *Numer. Linear Algebra Appl.*, 25(6):e2174, 2018.

[47] Michael Griebel and Guanglian Li. On the decay rate of the singular values of bivariate functions. *SIAM J. Numer. Anal.*, 56(2):974–993, 2018.

[48] W. Hackbusch. *Tensor spaces and numerical tensor calculus*. Springer, 2012.

[49] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration*, volume 31 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 2006. Structure-preserving algorithms for ordinary differential equations.

[50] E. Hairer, G. Wanner, and S. P. Nørsett. *Solving ordinary differential equations I: Nonstiff problems*, volume 8 of *Springer Series in Computational Mathematics,*. Springer Berlin Heidelberg, second edition, 1993.

[51] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral methods for time-dependent problems*. Cambridge University Press, 2007.

[52] S. Holtz, T. Rohwedder, and R. Schneider. On manifolds of tensors of fixed TT-rank. *Numer. Math.*, 120(4):701–731, 2012.

[53] E. Hopf. Statistical hydromechanics and functional calculus. *J. Rat. Mech. Anal.*, 1(1):87–123, 1952.

[54] A. Karimi and M. R. Paul. Extensive chaos in the Lorenz-96 model. *Chaos*, 20(4):043105(1–11), 2010.

[55] L. Karlsson, D. Kressner, and A. Uschmajew. Parallel algorithms for tensor completion in the CP format. *Parallel compting*, 57:222–234, 2016.

[56] A.-K. Kassam and L. N. Trefethen. Fourth-order time stepping for stiff PDEs. *SIAM J. Sci. Comput.*, 26(4):1214–1233, 2005.

[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. EDP Sci., Les Ulis, 2015.

[58] 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.

[59] E. Kieri and B. Vandereycken. Projection methods for dynamical low-rank approximation of high-dimensional problems. *Comput. Methods in Appl. Math.*, 19(1):73–92, 2019.

[60] V. I. Klyatskin. *Dynamics of stochastic systems*. Elsevier Publishing Company, 2005.

[61] O. Koch and C. Lubich. Dynamical tensor approximation. *SIAM J. Matrix Anal. Appl.*, 31(5):2360–2375, 2010.

[62] Othmar Koch and Christian Lubich. Dynamical low-rank approximation. *SIAM J. Matrix Anal. Appl.*, 29(2):434–454, 2007.

[63] T. Kolda and B. W. Bader. Tensor decompositions and applications. *SIREV*, 51:455–500, 2009.

[64] D. Kressner and C. Tobler. Algorithm 941: htucker – a Matlab toolbox for tensors in hierarchical Tucker format. *ACM Transactions on Mathematical Software*, 40(3):1–22, 2014.

[65] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.

[66] P. Lax and R. D. Richtmyer. Survery of the stability of linear finite difference equations. *Communications on pure and applied mathematics*, 9:267–293, 1956.

[67] Peter Lax and Burton Wendroff. Systems of conservation laws. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 1958.

[68] Adrian S Lewis and Jérôme Malick. Alternating projections on manifolds. *Math. of Operations Res.*, 33(1):216–234, 2008.

[69] G. Li and H. Rabitz. Regularized random-sampling high dimensional model representation (RS-HDMR). *J. Math. Chem.*, 43(3):1207–1232, 2008.

[70] L. Li, J. Qiu, and G. Russo. A high order semi-Lagrangian finite difference method for nonlinear Vlasov and BGK models. *Comm. Appl. Math. Comput.*, 5:170–198, 2023.

[71] Jun Liu, Xiangqian Liu, and Xiaoli Ma. First-order perturbation analysis of singular vectors in singular value decomposition. *IEEE Trans. on Signal Process.*, 56(7):3044–3049, 2008.

[72] Edward N Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on predictability*, volume 1, 1996.

[73] C. Lubich, I. V. Oseledets, and B. Vandereycken. Time integration of tensor trains. *SIAM J. Numer. Anal.*, 53(2):917–941, 2015.

[74] C. Lubich, T. Rohwedder, R. Schneider, and B. Vandereycken. Dynamical approximation by hierarchical Tucker and tensor-train tensors. *SIAM J. Matrix Anal. Appl.*, 34(2):470–494, 2013.

[75] Christian Lubich and Ivan V. Oseledets. A projector-splitting integrator for dynamical low-rank approximation. *BIT Numer. Math.*, 54(1):171–188, 2014.

[76] Thomas Marz and Colin B Macdonald. Calculus on surfaces with general closest point functions. *SIAM J. Numer. Anal.*, 50(6):3303–3328, 2012.

[77] X. Meng, Z. Li, D. Zhang, and G. Karniadakis. Ppinn: Parareal physics-informed neural network for time-dependent PDEs. *Computer Methods in Applied Mechanics and Engineering*, 370:113250, 10 2020.

[78] H. Montanelli and Y. Nakatsukasa. Fourth-order time-stepping for stiff PDEs on the sphere. *SIAM J. Sci. Comput.*, 40(1):A421–A451, 2018.

[79] Eleonora Musharbash, Fabio Nobile, and Tao Zhou. Error analysis of the dynamically orthogonal approximation of time dependent random PDEs. *SIAM J. Sci. Comput.*, 37(2):A776–A810, 2015.

[80] 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.

[81] Mark R. Opmeer. Decay of singular values of the gramians of infinite-dimensional systems. In *2015 European Control Conference (ECC)*, pages 1183–1188, 2015.

[82] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[83] I. V. Oseledets. TT-Toolbox 2.3. *GitHub Repository*, https://github.com/oseledets/TT-Toolbox, 2014.

[84] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical mathematics*. Springer, second edition, 2007.

[85] M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *J. Comput. Phys.*, 357:125–141, 2018.

[86] 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.

[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. *Journal of Computational Physics*, 378:686–707, 2019.

[88] S. C. Reddy and L. N. Trefethen. Lax stability of fully discrete spectral methods via stability region and pseudo-eigenvalues. *Computer methods in applied mechanics and engineering*, 80:147–164, 1990.

[89] S. C. Reddy and L. N. Trefethen. Stability of the method of lines. *Numer. Math.*, 62:235–267, 1992.

[90] H. Risken. *The Fokker-Planck equation: methods of solution and applications*. Springer-Verlag, second edition, 1989. Mathematics in science and engineering, vol. 60.

[91] A. Rodgers. htucker-mpi. `https://github.com/akrodger/htucker-mpi`, 2019.

[92] A. Rodgers, A. Dektor, and D. Venturi. Adaptive integration of nonlinear evolution equations on tensor manifolds. *J. Sci. Comput.*, 92(39):1–31, 2022.

[93] A. Rodgers and D. Venturi. Stability analysis of hierarchical tensor methods for time-dependent PDEs. *J. Comput. Phys.*, 409:109341, 2020.

[94] T. Rohwedder and A. Uschmajew. On local convergence of alternating schemes for optimization of convex problems in the tensor train format. *SIAM J. Numer. Anal.*, 51(2):1134–1162, 2013.

[95] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Data-driven discovery of partial differential equations. *Science Adv.*, 3(4):e1602614, 2017.

[96] L. Ruthotto, S. Osher, W. Li, L. Nurbekyan, and S. W. Fung. A machine learning framework for solving high-dimensional mean field game and mean field control problems. *PNAS*, 117(17):9183–9193, 2020.

[97] Themistoklis P Sapsis and Pierre FJ Lermusiaux. Dynamically orthogonal field equations for continuous stochastic dynamical systems. *Physica D: Nonlinear Phenomena*, 238(23-24):2347–2360, 2009.

[98] R. Schneider and A. Uschmajew. Approximation rates for the hierarchical tensor format in periodic sobolev spaces. *Journal of complexity*, 30(2):56–71, 2014.

[99] T. Shi, M. Ruth, and A. Townsend. Parallel algorithms for computing the tensor-train decomposition. *ArXiv*, (2111.10448):1–23, 2021.

[100] Gilbert W Stewart. Perturbation theory for the singular value decomposition. Technical report, 1998.

[101] J. C. Strikwerda. *Finite difference schemes and partial differential equations*. SIAM, second edition, 2004.

[102] S. Torquato. *Random heterogeneous materials, microstructure and macroscopic properties*. Springer, 2002.

[103] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[104] A. Trombettoni and A. Smerzi. Discrete solitons and breathers with dilute Bose-Einstein condensates. *Phys. Rev. Lett.*, 86:2353–2356, 2001.

[105] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[106] A. Uschmajew. Local convergence of the alternating least squares algorithm for canonical tensor approximation. *SIAM J. Matrix Anal. Appl.*, 33(2):639–652, 2012.

[107] A. Uschmajew and B. Vandereycken. The geometry of algorithms using hierarchical tensors. *Linear Algebra Appl.*, 439(1):133–166, 2013.

[108] André Uschmajew and Bart Vandereycken. Geometric methods on low-rank matrix and tensor manifolds. In *Handbook of Variational Methods for Nonlinear Geometric Data*, pages 261–313. Springer, 2020.

[109] D. Venturi. The numerical approximation of nonlinear functionals and functional differential equations. *Phys. Reports*, 732:1–102, 2018.

[110] D. Venturi. The numerical approximation of nonlinear functionals and functional differential equations. *Physics Reports*, 732:1–102, 2018.

[111] D. Venturi and A. Dektor. Spectral methods for nonlinear functionals and functional differential equations. *Research in the Mathematical Sciences*, 8(27):1–39, 2021.

[112] D. Venturi and G. E. Karniadakis. Convolutionless Nakajima-Zwanzig equations for stochastic analysis in nonlinear dynamical systems. *Proc. R. Soc. A*, 470(2166):1–20, 2014.

[113] 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.

[114] C. Villani. *Optimal transport: old and new*. Springer, 2009.

[115] Gerhard Wanner and Ernst Hairer. *Solving ordinary differential equations II*. Springer Berlin Heidelberg, 1996.

[116] S. Weinberg. *The quantum theory of fields: volume I*. Cambridge University Press, 2002.

[117] 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.

[118] J. Zinn-Justin. *Quantum field theory and critical phenomena*. Oxford Univ. Press, fourth edition, 2002.