Unconventional contributions to dynamical low-rank approximation of tree tensor networks

Dissertation
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
M. Sc. Gianluca Ceruti
aus Rom (Italien)

Tübingen
2021

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	29.07.2021
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Christian Lubich
2. Berichterstatter:	Prof. Dr. Andreas Prohl

Dedicato a Gulia ed alla mia famiglia.

Summary

The present thesis deals with the numerical time-integration of high-dimensional time-dependent ordinary differential equations arising from, e.g. the discretization of high-dimensional time-dependent partial differential equations.

Because the size of such problems is assumed to be extremely large, standard discretization techniques are not feasible. The number of quantities needed to be stored and treated exceeds standard capacities of common computational devices; a circumstance usually referred to as *curse of dimensionality*. A different ansatz is required to be used, i.e. the dynamical low-rank approximation.

Dynamical low-rank approximation consists of projecting the time-derivative of the time-dependent solution of the given problem onto the tangent space of a search space of lower complexity, e.g. the manifold of low-rank matrices of fixed rank r.

In the manifold of low-rank matrices, an efficient numerical integrator based on a projector-splitting approach has been recently proposed. This numerical integrator has been proven to retain a low-memory footprint and to be robust with respect to the presence of small singular values, desirable property which is needed for achieving sufficient precision in the final approximation.

Different low-rank manifolds exist in the high-dimensional setting, and analogous projector-splitting integrators have been there proposed. In the literature, exhaustive and complete research has been carried out for dynamical low-rank approximation of high-dimensional low-rank loop-free formats, such as tensors in Tucker format and tensors in hierarchical Tucker format.

We present here the extension of the projector-splitting integrator for matrices to the most general class of tree tensor networks, which by construction includes all mentioned low-rank loop-free formats.

Furthermore, a non-trivial modification of the projector-splitting integrator, preserving (skew-)symmetry for matrices and tensors in Tucker format, is presented.

The latter results are generalized to low-rank matrices and tensors in Tucker format. Due to its unusual derivation and construction, the new derived numerical integrator is referred to as the unconventional integrator.

In contrast to the original projector-splitting integrator for matrices and tensors in Tucker format, the new unconventional integrator introduces more parallelism. It preserves the original excellent properties of the projector-splitting integrator for matrices and tensors in Tucker format and provides more stability when strong dissipative problems are considered.

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der numerischen Zeitintegration von hoch-dimensionalen, zeit-abhängigen, gewöhnlichen Differentialgleichungen, welche beispielsweise bei der Diskretisierung von partiellen Differentialgleichungen auftreten. Treten in den Problemen sehr hohe Dimensionen auf, sind Standardtechniken nicht mehr durchführbar. Die Anzahl und Größe der Objekte, welche gespeichert und verarbeitet werden müssen, übersteigt bei Weitem die Kapazitäten eines Standard-Computers. Dieser Umstand wird häufig als der Fluch der Dimension bezeichnet. Deshalb müssen andere Ansätze zur Behandlung solcher Probleme herangezogen werden, wie zum Beispiel dynamische Approximationen von niedrigem Rang.
Dynamische Approximationen von niedrigem Rang beruhen auf der Projektion der Zeitableitung einer zeitabhängigen Lösung eines gegebenen Problems auf den Tangentialraum eines Lösungsraums, welcher eine niedrigere Komplexität aufweist. Die Mannigfaltigkeit der Matrizen von festem Rang r sind ein Beispiel für einen solchen Lösungsraums. Auf dieser speziellen Mannigfaltigkeit wurde mit Hilfe einer Zerlegung der Projektion (=Projektions-Splitting) ein Integrator vorgestellt. Dieser erhält die spezielle Struktur der Objekte bei, was ein effizientes Speichern ermöglicht. Weiter weist der Integrator Robustheit bezüglich kleiner Singulärwerte auf, was essentiell ist um eine ausreichende Genauigkeit der Approximation zu gewährleisten.
Für hoch-dimensionale Probleme existieren diverse Mannigfaltigkeiten von niedrigem Rang, auf welchen analog Integratoren vorgestellt wurden, die ebenfalls auf einer Zerlegung der Projektion beruhen. Die Literatur ist reich an Arbeiten zu dynamischen Approximationen von niedrigem Rang für hoch-dimensionale, niedrig-rangige und schleifen-freie Darstellungen, wie zum Beispiel Tucker-Tensoren oder Tensoren in hierarchischer Tucker Darstellung.
Wir präsentieren hier die Erweiterung des Projektions-Splitting Integrators für Matrizen auf die Klasse der Tensor-Netzwerke, welche nach Konstruktion alle oben genannten Darstellungen verallgemeinert.
Weiter wird eine nicht-triviale Modifikation des Projektions-Splitting Integrators vorgestellt, welche die (Schief-)Symmetrie von Matrizen und Tucker-Tensoren erhält. Diese Modifikation wurde auf Matrizen von niedrigem Rang und Tensoren in Tucker Darstellung verallgemeinert. Aufgrund der ungewöhnlichen Herleitung und Konstruktion des neuen Integrators, bezeichnen wir diesen als unkonventionellen Integrator. Im Unterschied zum Projektions-Splitting Integrator für Matrizen und Tucker-Tensoren, ermöglicht der unkonventionelle Integrator paralleles Rechnen. Er erhält die ursprünglichen vorteilhaften Eigenschaften des Projektions-Splitting Integrators für Matrizen und Tucker-Tensoren und bietet mehr Stabilität bei stark dissipativen Problemen.

Objectives, list of publications and contributions

The objective of the present work is to extend the matrix projector-splitting integrator of [33] to tree tensor networks [7] and to investigate non-trivial (skew-)symmetry preserving modifications of the matrix projector-splitting integrator [5]. The latter results are generalized to low-rank matrices and tensors in Tucker format [6].

The present work is based on the following list of publications:

[7] G. Ceruti, C. Lubich, and H. Walach. Time integration of tree tensor networks. *SIAM J. Numer. Anal.* 59 (2021), 289-313.

[5] G. Ceruti and C. Lubich. Time integration of symmetric and anti-symmetric low-rank matrices and Tucker tensors. *BIT Numer. Math.*, 60:591–614, 2020.

[6] G. Ceruti and C. Lubich. An unconventional robust integrator for dynamical low-rank approximation. *BIT Numer. Math.*, 2021.

For each manuscript, a printable version is available in the appendix.

For each manuscript, the theoretical results are equal contributions of the authors. The implementation of the numerical experiments in [7, 5, 6] is due to the author of the present thesis.

For each manuscript, the source code used for the numerical experiments can be found at `https://bitbucket.org/dlrtree/`.

Acknowledgements

Tale compito, sinceramente il più importante, richiede di essere svolto nella lingua più congeniale allo scrittore stesso. Pertanto, mi appropinquo a tale scopo per mezzo dell'uso della lingua che ebbe origine in quel di Firenze, c.a. 700 anni or sono.

Desidero cominciare con il ringraziamento più importante di tutti: desidero ringraziare Christian (Prof. Lubich). Giunto a Tubinga, sei stato immediatamente ben preciso e mi hai chiesto di evitare, nella vita di tutti i giorni, di darti del "Lei". Per molti mesi e per motivi oramai oscuri, non sono riuscito a farlo e dunque, alla fine di questo percorso, mi sembra giusto invertire le due cose e darti qui del "Tu".

Tolto questo lungo preambolo, spiegazione dovuta più che altro al lettore ignaro di tali vicende, desidero ringraziarti per un motivo estremamente importante.

Nei limiti delle mie possibilità, adoro fare matematica. Poter completare la mia educazione accademica ed aspirare al titolo di dottorato era il mio sogno più grande. Sei stato un mentore eccezionale, paziente quando necessario ed esigente quando sapevi che avrei potuto raggiungere risultati più importanti. Arrivato qualche anno fa a Tubinga, pieno di dubbi e paure, guardo ora avanti pieno di fiducia e ricordi bellissimi. Per tutti questi motivi, non credo vi siano parole a sufficienza per esprimere la mia gratitudine nei tuoi confronti.

Il secondo ringraziamento va al Prof. Teufel e al Prof. Guglielmi. Entrambi, in maniera indiretta, sono stati artefici di questo mio percorso. Senza il Prof. Guglielmi, probabilmente, non vi sarebbe stato un viaggio a Roma e dunque una piacevole intervista di lavoro a piazza Navona. Senza il supporto del Prof. Teufel e del GRK 1838, questa mia avventura non avrebbe potuto avere inizio.

Ringrazio ulteriormente il Prof. Frank e la Prof.ssa Hochbruck dell'istituto tecnologico di Karlsruhe. Senza il loro supporto, manifestatosi attraverso l'operato dell'SFB 1173, questo lavoro non sarebbe potuto giungere al termine. Nei successi e negli insuccessi, mi avete insegnato moltissimo. Ho imparato innumerevoli lezioni di vita che porterò meco per tutto il resto della mia carriera, accademica e non.

Un ringraziamento speciale va al Prof. Tyrtyshnikov (INM). Fin dal principio, ha supportato attivamente questa mia scelta e per primo mi ha trasmesso la passione per il calcolo numerico sui tensori e la ricerca scientifica.

Ringrazio Frau Eberspächer e Frau Kabagema-Bilan. Senza il loro costante aiuto, non sarei mai riuscito a galleggiare nel mare della burocrazia universitaria.

Sebbene una solida conoscenza scientifica fosse un presupposto importante per raggiungere lo scopo che mi ero prefissato, alla fine di questo percorso, mi accorgo che una quantità innumerevole di ulteriori capacità erano richieste. Personalmente, tuttora, ritengo di possederne ben poche. Dunque, se sono riuscito a completare tutto quanto è perché vicino a me ho sempre avuto una persona speciale che con amore e pazienza sempre riesce a tirare fuori il meglio di me: Gulia. A lei, va il mio amore ed il ringraziamento non accademico più grande.

Ringrazio i miei genitori, mia sorella e mio fratello per essere sempre stati vicini e presenti in questa e nelle precedenti avventure. Senza i 30 anni e più di amorevoli discussioni familiari all'italiana, su temi assolutamente diversi tra di loro ma rigorosamente in contemporanea (così vuole l'etichetta nostrana), non avrei mai imparato il valore della resilienza e non sarei forte come lo sono oggi.

Ringrazio tutta la mia famiglia nella tundra moscovita: Olga, Sasha, Vania, Masha, Jenia, Roma, Xenia, Misha. Neanche nel più freddo degli inverni russi mi sono mai sentito solo. Credo che se dovessi decidere dove collocare il centro del mondo, sarebbe senz'altro una certa dacia a nord della città di Mosca.

Ringrazio i miei due angeli custodi: Pina e Tiziana. In un momento di grande difficoltà personale mi avete dato tantissimo senza mai chiedere nulla in cambio.

Ringrazio tutto il team array: Alberto (detto il re d'Anagni – per ovvie ragioni), Alessandro (detto er Giappo – per altrettante ovvie ragioni) e Francesco (detto l'ingegnere – perché è un ingegnere). Infingardo fu quel giorno che ci sedemmo tutti allo stesso tavolo; se oggi sono in grado di scrivere due righe di codice, lo devo certamente ai commenti esageratamente "onesti" di Alberto.

Ringrazio Francesco, Stefanone ed il Prof. Di Fiore. Amici e colleghi da una vita, da sempre ci accompagna la passione per la matematica e la voglia di fare ricerca. Trovandoci adesso in ambito frascatano, non si può non aggiungere un ringraziamento agli amici di sempre: Francesca, Andrea, Nicolò, Isabella, Daniele, Roberto e Lorenzo. Aggiungo anche Tania e tutti i nuovi piccoli arrivati che con gioia e felicità riempiono la nostra vita di sorrisi e domande affascinanti.

Ringrazio tutta la mia famiglia brasileira: Ananda, Ro, Rogerio, Fra, Ka, Tia Maria, Tia Sandra, Vo' Antonia, Caro, Bea. Il tempo speso insieme in Brasile è un ricordo che mi accompagna sempre nelle mie giornate. Un pensiero speciale va a Marika, che dall'alto ci guarda e veglia su di noi.

Ringrazio i miei amici e colleghi tedeschi, a cominciare dalle due persone che per prime mi hanno accolto a Tubinga: Hanna e Balázs. Non è la prima volta che mi avventuro in una realtà del tutto nuova e diversa, ma ciò non vuol dire che sia facile. Senza il vostro supporto e consiglio, tutto sarebbe stato assai più complicato. Ringrazio inoltre tutti i membri, passati e presenti, del gruppo di analisi numerica di Tubinga: i due Dominik, Jörg, Lukas, Bin e Yanyan. Aggiungo in questo gruppo anche Jonas con il quale abbiamo lavorato enormemente e piacevolmente nel corso di questo ultimo anno, ben fuori dal comune.

Ringrazio oltremodo Jochen che con enorme pazienza, nel corso di questi ultimi anni, mi ha sentito più e più volte camminare avanti e indietro nel mio appartamento, nel mentre di sofisticate (forse anche troppo) elucubrazioni matematiche.

Per concludere, il ringraziamento finale e per me uno dei più importanti va alla categoria giovanissimi: Nanni, Giorgio e Chicco. Ho visto crescere ognuno di voi tre e siete per me una parte fondamentale ed una forza trainante della mia vita. A Chicco, ormai ti sei trasformato in un grande gigante gentile: mi riempie d'orgoglio vedere la persona matura che sei diventato; se so parlare un po' di slang parigino, te lo devo. A Giorgio, ti ringrazio per tutte le volte che con la tua intelligenza e curiosità, mi hai fatto capire che tutte le cose, in fondo, son molto (ma molto) più semplici di quello che penso. A Nanni, per me rimarrai sempre la piccola Rubona ed anche se non siamo sempre vicini, sappi che ti voglio un mondo di bene e sempre sarai nei miei pensieri. Ti ringrazio per il tuo sorriso che, come una panacea, sempre mi fa passare qualsiasi malumore.

Contents

Introduction

Galileo Galilei, considered among the founders of modern science, in his book *Il Saggiatore* (1623) wrote: "Philosophy is written in this all-encompassing book that is constantly open to our eyes, that is the universe; but it cannot be understood unless one first learns to understand the language and knows the characters in which it is written. It is written in mathematical language, and its characters are triangles, circles, and other geometrical figures; without these it is humanly impossible to understand a word of it, and one wanders in a dark labyrinth."

Poetry could have been chosen as a language to describe scientific discoveries, and we leave to other branches of knowledge to speculate how our modern world would have been forged. Mathematics has been selected and starting from so on his lead has been followed. The exploration of the macro (stars and galaxies) – micro (quantum reality) world has started and numerous people, who were first labelled as philosophers, have put their foots on this path. It is a vast journey, which covers centuries of developments and brings us to this work, which humbly asks to be part of this huge collection.

In this approach, time evolution of a natural phenomenon is translated into a mathematical model. In a precise mathematical formulation, time evolution of a certain physical process is described via an evolution equation

$$\partial_t u = \mathcal{F}(t, u), \quad u(t_0) = u_0.$$

This short-hand, compact and abstract formulation includes a variety of linear and non-linear examples, see e.g. [12]: Schrödinger equation, kinetic equations, Hamilton–Jacobi equation, Fokker–Plank equation, etc.

Since an analytical solution does not exist or is unknown, a numerical approximation is desirable. We discretize the continuous problem and provide algorithms to approximate such a time-dependent solution. Then, our numerical procedures are implemented on modern calculators and need to adapt to their limitations, such as the limited amount of storage. If the size of the discretized problem is too large to be stored in a physical device, new approaches are required and we are intended to investigate one of them: dynamical low-rank approximation. We refer to [45] for a comprehensive overview on the topic. The present work completes and extends results illustrated there.

It is convenient to start our exposition from a different and reduced perspective. Consider an ordinary differential equation of a finite-dimensional problem

$$\dot{\mathbf{A}}(t) = \mathbf{F}(t, \mathbf{A}(t)), \quad \mathbf{A}(t_0) = \mathbf{A}_0 \in \mathbb{R}^{m \times n}.$$

We assume that such a formulation arises, for example, from a suitable discretization of the original problem. Here, we assume that the dimensions $m, n \in \mathbb{N}$ are extremely

large. Storing and manipulating these quantities can reveal itself as problematic. As already mentioned, an alternative approach is needed.

We intend to present an approach that consists of approximating the time-dependent solution $\mathbf{A}(t)$ via a quantity of lower complexity, such as a time-dependent low-rank approximation of rank $r \ll \{m, n\}$ obtained via a singular value decomposition [18, 10]

$$\mathbf{A}(t) \approx \mathbf{Y}(t) = \mathbf{U}(t)\mathbf{S}(t)\,\mathbf{V}(t)^\top$$

where $\mathbf{U}(t) \in \mathbb{R}^{m\times r}$, $\mathbf{V}(t) \in \mathbb{R}^{n\times r}$ have orthonormal columns and $\mathbf{S}(t) \in \mathbb{R}^{r\times r}$ is of full rank r. Equivalently, for each time $t \in [t_0, T]$, we aim to find

$$\mathbf{Y}(t) \in \mathcal{M}_r \quad \text{such that} \quad \|\,\mathbf{Y}(t)\text{–}\mathbf{A}(t)\|_{\mathrm{F}} = \text{min!}\ ,$$

where $\mathcal{M}_r$ denotes the manifold of matrices of size $m \times n$ and of fixed rank r. This ansatz allows to treat a larger numbers of quantities, but it introduces a novel difficulty. At each time, this approximation must be computed and the computation cost is again proportional to the size, m times n, of the problem [18].

A different approach, the dynamical low-rank approximation for matrices is first presented in [28]. The time-dependent optimization problem on the low-rank manifold $\mathcal{M}_r$ is transferred onto its tangent space. We aim to find

$$\dot{\mathbf{Y}}(t) \in \mathcal{T}_{\mathbf{Y}(t)}\mathcal{M}_r \quad \text{such that} \quad \|\dot{\mathbf{Y}}(t)\text{–}\dot{\mathbf{A}}(t)\|_{\mathrm{F}} = \text{min!}\ ,$$

or equivalently

$$\langle \dot{\mathbf{Y}}(t) - \dot{\mathbf{A}}(t),\ \delta\,\mathbf{Y}\rangle_{\mathrm{F}} = 0 \quad \text{for all } \delta\,\mathbf{Y} \in \mathcal{T}_{\mathbf{Y}(t)}\mathcal{M}_r\ .$$

In the framework of quantum dynamics, this approach was already known as the Dirac–Frenkel variational principle [31]. When evolution equations for the single factors $\mathbf{U}(t)$, $\mathbf{V}(t)$ and $\mathbf{S}(t)$ were derived in [28], an immediate undesired effect was observed: the appearance of the inverse of the matrix $\mathbf{S}(t)$. Difficulties with small singular values might arise when standard numerical integrators are employed. A regularization technique, such as adding small perturbations to avoid singularities can be used, but its effect needs to be analysed and properly controlled to avoid artefacts in the final approximation [38].

We proceed further and recast the minimization problem as

$$\dot{\mathbf{Y}}(t) = P(\mathbf{Y}(t))\,\mathbf{F}(t, \mathbf{Y}(t)), \quad \mathbf{Y}(t_0) = \mathbf{Y}_0 \in \mathcal{M}_r\ ,$$

where $P(\mathbf{Y}(t))$ is the orthogonal projection onto the tangent space of the low-rank manifold $\mathcal{M}_r$ at $\mathbf{Y}(t)$. Observing that the orthogonal projection is given as an alternating sum of three different terms, a numerical procedure known as the matrix projector-splitting integrator is derived in [33]. An incredible theoretical result is immediately proved: if the original dynamic $\mathbf{A}(t)$ occurs in the low-rank manifold for $t_0 \le t \le T$, the matrix projector-splitting integrator is exact. The numerical approximation obtained with the matrix projector-splitting integrator is identical to the full continuous solution. The exactness property is then used to prove a second remarkable property: the matrix projector-splitting integrator is robust with respect to the presence of small singular values [24]. Combined with its low-memory footprint, this numerical integrator is of particular interest.

The first area of application is given by many-body quantum dynamics [32]. The main interest is to study and analyse quantum dynamics of systems with more than two particles. The discretized dynamic is now represented by a tensor, a multi-dimensional array, $A(t) \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, given via a discrete tensor differential equation.

There exist different low-rank formats for tensors, we refer to [30, 17] for a complete survey. A direct generalization of the SVD factorization for matrices is the Tucker tensor decomposition [43] or HOSVD [8, 9]. A tensor $Y \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ has multilinear rank $(r_1, \ldots, r_d)$ if and only if it can be factorized as a Tucker tensor

$$Y = C \bigtimes_{i=1}^{d} \mathbf{U}_i, \quad \text{i.e.,} \quad y_{k_1,\ldots,k_d} = \sum_{l_1=1}^{r_1} \cdots \sum_{l_d=1}^{r_d} c_{l_1,\ldots,l_d} u_{k_1,l_1} \ldots u_{k_d,l_d},$$

where the basis matrices $\mathbf{U}_i \in \mathbb{R}^{n_i \times r_i}$ have orthonormal columns and the core tensor $C \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ has full multilinear rank $(r_1, \ldots, r_d)$. Dynamical low-rank approximation for matrices have been successfully extended to tensors in Tucker format [29]. In the framework of quantum dynamics, an extension of the matrix projector-splitting integrator has been first presented in [32] and efficiently implemented in [25]. This integrator has then been reformulated as a recursive application of the matrix projector-splitting integrator, and an alternative derivation has been found [36]. It has been proven that the two formulations are equivalent [36]. Using the last derivation as a recursive iteration of the matrix case, it has been proven that the Tucker numerical integrators continue to satisfy the exactness property and the robustness with respect to small singular values.

The Tucker tensor format is a fundamental tool to treat mildly high-dimensional problems. However, an insurmountable limit is hidden inside of this decomposition. The size of the core tensor $C \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ scales exponentially with the ranks and the dimension of the problem. To deal with large dimensional problems ($d \geq 10$), it is required to break this growth of dimensionality in the core tensor and along recent years, recursive formats such as Hierarchical Tucker (HT) [20, 16] have been introduced. A relevant category of interest belonging to the HT set is given by tensors in tensor train format [39]. Already known in physics as matrix product-states [40], they have been recently re-presented to the mathematical community.

Dynamical low-rank approximation has been extended to tensors in HT format in [35] and a remarkable extension of the matrix projector-splitting integrator to tensors in tensor train format has been derived in [34], and reformulated in physical idiom in [21]. An exactness proof is given in [34], and the robust error bound has been then proved in [24].

Tensors in hierarchical format such as tensor trains allow to deal with large dimensional problems, but in different physical applications a more diverse and richer ansatz is required. Examples are given by kinetic equations where space and velocity are split equally [11], or quantum dynamics on two-dimensional quantum lattice [41, 42]. Therefore, we are interested in tensors where their configuration is described by a general tree, not only a binary one as in the HT setting.

A tensor in a recursive format where its configuration is described by a general tree, is called tree tensor network, shortly denoted by the acronym TTN. First successful attempts to extend the matrix projector-splitting integrator to TTNs in specific and application-related frameworks can be found in [11, 3, 26]. There, important experimental results are shown but no general derivation or theoretical analysis is presented.

A first contribution of the present work consists of the extension of the matrix projector-splitting integrator together with its remarkable properties to TTNs [7].

Various new elements are introduced for the first time. A novel, useful and convenient notation for TTNs is given from scratch. The Tucker integrator is reformulated as a composition of subflows, and a more compact representation than the one introduced in [36] is obtained.

Then, the derivation of the numerical integrator for TTNs is not obtained via a projector-splitting approach but rather follows a recursive algorithmic derivation such as in [36, 11]. The novel integrator recursively applies the Tucker integrator at each level. If the folded differential problem appearing in the course of the application of the Tucker integrator has too large size, it is not solved directly, but the structure of the new TTN format is exploited. The large and untreatable problem is unfolded and the Tucker integrator is recursively applied again.

Since such TTN structures are extremely general, nested and complicated, the theoretical analysis and an effective implementation of the algorithm might prove to be considerably difficult. This issue is overcome with the introduction of two useful, practical and theoretical tools: prolongations and restrictions. Their efficient implementation allows the construction of an extension of the matrix projector-splitting integrator for TTNs. Their theoretical properties make the new algorithm inherit the remarkable exactness and robustness property of the matrix projector-splitting integrator and of the Tucker integrators.

Because of its recursive structure, the novel and more general algorithm is referred to as "recursive tree tensor network integrator".

The recursive TTN integrator represents the most abstract formulation of the original matrix projector-splitting integrator. Therefore, it includes all the previous mentioned cases: matrices, Tucker tensors, tensors in HT format. Its derivation successfully concludes a path started and first promoted in [33].

A second contribution [5] of the present work consists in a non-trivial modification of the matrix projector-splitting integrator preserving (skew-)symmetry of matrices and Tucker tensors. In this case, we remind that the low-rank SVD factorization for matrices is such that the (skew-)symmetry property is preserved in the core part of the decomposition and all basis matrices are identical. Analogously this decomposition holds true for tensors in Tucker format [19].

Modifications of the matrix projector-splitting integrator for preservation of the symmetric or skew-symmetric property have been investigated first in [37]. There, applications of interest are Lyapunov matrix differential equations which have large symmetric matrices as solutions. Another field of interest is given by quantum dynamics of systems of bosonic or fermionic particles, where the symmetric or anti-symmetric wave function is approximated by low-rank symmetric or anti-symmetric Tucker tensors [4, 27, 1].

In contrast to [37], the new algorithm presented in [5] consists of only two steps, both for the matrix case and for the multi-dimensional Tucker case, respectively. Furthermore, it is proved that the proposed symmetry and anti-symmetry preserving numerical integrator is first order in time, continues to satisfy the exactness property, and is robust with respect to the presence of small singular values.

The last contribution of the present work is given in [6]. The numerical integrator for (skew-)symmetric matrices and (anti-)symmetric Tucker tensors is further generalized.

It is proved that the new unconventional integrator continues to satisfy the remarkable exactness and robustness property. Furthermore, the new algorithm removes a source of instability introduced by the backward middle step of the matrix projector-splitting integrator and adds more parallelism.

The present manuscript is structured as follows.

In Chapter 1, we briefly recall dynamical low-rank approximation for matrices. We present the matrix projector-splitting integrator together with its remarkable properties. Then, we describe tensors in Tucker format and the Tucker numerical integrators.

In Chapter 2, we introduce the notion of tree tensor network and we present the recursive tree tensor network integrator.

In Chapter 3, the new unconventional low-rank integrator for matrices and Tucker tensors is presented.

Throughout the manuscript matrices are denoted by bold letters. Where not explicit or stated otherwise, the norm is intended as the Frobenius norm.

Throughout the present work we will focus on the real case. The complex case is obtained by carefully replacing the transpose with the conjugate transpose.

For each algorithm only one time-step evolution will be presented. Successive approximations in time can be obtained by re-iterating the same procedure.

For each contribution only the main elements of interest, such as ideas, definitions, lemmas and theorems are provided. We always refer to the respective papers for a more detailed description.

Chapter 1

Dynamical low-rank approximation

In the present chapter dynamical low-rank approximation for matrices is described. Then, we present the matrix projector-splitting integrator, together with its remarkable properties. The latter results are extended to tensors in Tucker format, a high-order generalization of the singular value decomposition for matrices.

1.1 Reduced singular value decomposition

We start recalling a matrix factorization of interest to the present work: the reduced singular value decomposition. Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ be a matrix of rank r, it admits a factorization of type

$$\mathbf{A} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^\top\,, \tag{1.1}$$

where $\mathbf{U} \in \mathbb{R}^{m\times r}$, $\mathbf{V} \in \mathbb{R}^{n\times r}$ have orthonormal columns and $\mathbf{S} \in \mathbb{R}^{r\times r}$ is of full rank.

Such a decomposition is obtained, e.g. by performing a singular value decomposition of the matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ and discarding all singular values after the r largest ones. We remind that if the factor $\mathbf{S}$ is not assumed to be diagonal, with singular values sorted in decreasing order, such a decomposition is not unique [18].

The set $\mathcal{M}_r$ of matrices with given dimensions m and n and fixed rank r forms a smooth manifold embedded in $\mathbb{R}^{m\times n}$ [23, 28].

1.2 Dynamical low-rank approximation for matrices

Let $\mathbf{A}(t) \in \mathbb{R}^{m\times n}$ be a continuously differentiable time-dependent matrix. For each time $t \in [t_0, T]$, we aim to compute the best rank-r approximation $\mathbf{X}(t)$ to $\mathbf{A}(t)$, i.e.

$$\mathbf{X}(t) \in \mathcal{M}_r \quad \text{such that} \quad \|\mathbf{X}(t) - \mathbf{A}(t)\| = \min!\ . \tag{1.2}$$

In the matrix setting, it is known that the best rank-r approximation $\mathbf{X}(t) \in \mathcal{M}_r$ is obtained by computing a rank-r reduced singular value factorization of $\mathbf{A}(t)$ [18]. However, if the dimensions m and n are extremely large, such an approach is not feasible. The computational cost of a single decomposition, proportional to the product m times n, becomes immediately prohibitive.

A different point of view, the dynamical low-rank approximation, was first presented in [28]. There, the authors minimize (1.2) with respect to the tangent space. Find

$$\dot{\mathbf{Y}}(t) \in \mathcal{T}_{\mathbf{Y}(t)}\mathcal{M}_r \quad \text{such that} \quad \|\dot{\mathbf{Y}}(t) - \dot{\mathbf{A}}(t)\| = \min!\ . \tag{1.3}$$

If the time-dependent matrix $\mathbf{A}(t) \in \mathbb{R}^{m\times n}$ is not given explicitly but as a solution of a matrix differential equation

$$\dot{\mathbf{A}}(t) = \mathbf{F}(t, \mathbf{A}(t)), \quad \mathbf{A}(t_0) = \mathbf{A}_0, \tag{1.4}$$

the dynamical low-rank approximation is

$$\dot{\mathbf{Y}}(t) \in \mathcal{T}_{\mathbf{Y}(t)}\mathcal{M}_r \quad \text{such that} \quad \|\dot{\mathbf{Y}}(t) - F(t, \mathbf{Y}(t))\| = \min! \,. \tag{1.5}$$

Dynamical low-rank approximation offers different advantages. Contrary to (1.2), it makes direct use of the differential nature of the problem (1.4). The approximation space $\mathcal{T}_{\mathbf{Y}(t)}\mathcal{M}_r$ evolves and adapts in time with the dynamic of (1.4) itself. The optimization problem (1.2) transforms in a linear projection. Let

$$\mathbf{Y}(t) = \mathbf{U}(t)\mathbf{S}(t)\,\mathbf{V}(t)^\top \in \mathcal{M}_r \,, \tag{1.6}$$

be a time-dependent reduced SVD-like factorization [10] with factor $\mathbf{S}(t) \in \mathbb{R}^{r\times r}$ only assumed to be full rank. The orthogonal projection $\mathrm{P}(\mathbf{Y}(t))$ onto the tangent space of the low-rank manifold of rank-r matrices at $\mathbf{Y}(t)$ is given by [28]

$$\mathrm{P}(\mathbf{Y})\,\mathbf{Z} = \mathbf{Z}\,\mathbf{V}\,\mathbf{V}^\top - \mathbf{U}\,\mathbf{U}^\top\,\mathbf{Z}\,\mathbf{V}\,\mathbf{V}^\top + \mathbf{U}\,\mathbf{U}^\top\,\mathbf{Z} \qquad \forall\,\mathbf{Z} \in \mathbb{R}^{m\times n}. \tag{1.7}$$

It has been shown in [28, 33] that (1.5) is equivalent to

$$\dot{\mathbf{Y}}(t) = \mathrm{P}(\mathbf{Y}(t))\,\mathbf{F}(t, \mathbf{Y}(t)), \quad \mathbf{Y}(t_0) = \mathbf{Y}_0, \tag{1.8}$$

where $\mathbf{Y}_0 \in \mathcal{M}_r$ denotes the best rank-r approximation to $\mathbf{A}_0$.

Solving the differential equation (1.8) directly is a source of numerical difficulties: an unbearable stiffness is hidden inside of the orthogonal projection. In fact, the local Lipschitz constant of the orthogonal projection $\mathrm{P}(\cdot)$ onto the tangent space depends on the inverse of the smallest nonzero singular value of its argument, as shown in the following lemma.

Lemma 1 ([28, Lemma 4.2])**.** *Let the rank-r matrix $\mathbf{X} \in \mathcal{M}_r$ be such that its smallest nonzero singular value satisfies $\sigma_r(\mathbf{X}) \geq \rho > 0$, and let $Y \in \mathcal{M}_r$ with $\|\,\mathbf{Y} - \mathbf{X}\,\| \leq \frac{1}{8}\rho$. Then, the following bounds hold: for all $\mathbf{Z} \in \mathbb{R}^{m\times n}$,*

$$\|\big(\mathrm{P}(\mathbf{Y}) - \mathrm{P}(\mathbf{X})\big)\,\mathbf{Z}\,\| \leq 8\rho^{-1}\|\,\mathbf{Y} - \mathbf{X}\,\|\|\,\mathbf{Z}\,\|_2, \tag{1.9}$$

$$\|\big(\mathbf{I} - \mathrm{P}(\mathbf{Y})\big)(\mathbf{Y} - \mathbf{X})\| \leq 4\rho^{-1}\|\,\mathbf{Y} - \mathbf{X}\,\|^2. \tag{1.10}$$

Being part of the geometric structure of the low-rank manifold $\mathcal{M}_r$ [15], such a dependence on small singular values is omnipresent. It becomes more transparent when evolution equations for the factors $\mathbf{U}(t), \mathbf{V}(t)$ and $\mathbf{S}(t)$ of $\mathbf{Y}(t)$ are derived. Imposing gauge conditions

$$\mathbf{U}^\top\,\dot{\mathbf{U}} = 0 \quad \text{and} \quad \mathbf{V}^\top\,\dot{\mathbf{V}} = 0,$$

it has been shown in [28, Prop. 2.1] that

$$\begin{aligned}
\dot{\mathbf{S}} &= \mathbf{U}^\top\,\mathbf{F}(t, \mathbf{Y})\,\mathbf{V}, & \mathbf{S}(t_0) &= \mathbf{S}_0, \\
\dot{\mathbf{U}} &= (\mathbf{I} - \mathbf{U}\,\mathbf{U}^\top)\,\mathbf{F}(t, \mathbf{Y})\,\mathbf{V}\,\mathbf{S}^{-1}, & \mathbf{U}(t_0) &= \mathbf{U}_0, \\
\dot{\mathbf{V}} &= (\mathbf{I} - \mathbf{V}\,\mathbf{V}^\top)\,\mathbf{F}(t, \mathbf{Y})^\top\,\mathbf{U}\,\mathbf{S}^{-\top}, & \mathbf{V}(t_0) &= \mathbf{V}_0 \,.
\end{aligned} \tag{1.11}$$

We observe that due to the presence of the inverse of the matrix $\mathbf{S}$ in the second and third equation, difficulties with small singular values naturally arise.

The objective of the present work is to derive numerical integrators for dynamical low-rank approximation and to study time-evolution of high-dimensional low-rank phenomena, therefore the appearance of small singular values in the numerical approximation cannot be neglected. The lowest singular value maintained in the approximation should not expected to be much larger than the biggest dropped singular value of the solution, which has to be small enough in order to achieve sufficient accuracy. A different approach has to be investigated.

1.3 Matrix projector-splitting integrator

In [33] a Lie–Trotter splitting technique is applied to (1.8) and an efficient numerical integrator, the matrix projector-splitting integrator, is derived. There it has been observed that the orthogonal projection $\mathrm{P}(\mathbf{Y})$ of (1.7) is given as an alternating sum of projections onto the range $\mathcal{R}(\mathbf{Y})$ and co-range $\mathcal{R}(\mathbf{Y}^\top)$

$$\mathrm{P}(\mathbf{Y})\,\mathbf{Z} = \mathbf{Z}\,\mathrm{P}_{\mathcal{R}(\mathbf{Y}^\top)} - \mathrm{P}_{\mathcal{R}(\mathbf{Y})}\,\mathbf{Z}\,\mathrm{P}_{\mathcal{R}(\mathbf{Y}^\top)} + \mathrm{P}_{\mathcal{R}(\mathbf{Y})}\,\mathbf{Z}\ . \tag{1.12}$$

Rather than solving (1.8), which would lead to difficulties with small singular values as previously discussed, the differential equations

$$\begin{aligned}
\dot{\mathbf{Y}}_I &= F(t,\mathbf{Y}_I)\mathrm{P}_{\mathcal{R}(\mathbf{Y}_I^\top)}, & \mathbf{Y}_I(t_0) &= \mathbf{Y}_0,\\
\dot{\mathbf{Y}}_{II} &= -\mathrm{P}_{\mathcal{R}(\mathbf{Y}_{II})}F(t,\mathbf{Y}_{II})\mathrm{P}_{\mathcal{R}(\mathbf{Y}_{II}^\top)}, & \mathbf{Y}_{II}(t_0) &= \mathbf{Y}_I(t_1),\\
\dot{\mathbf{Y}}_{III} &= \mathrm{P}_{\mathcal{R}(\mathbf{Y}_{III})}(\mathbf{Y}_{III})F(t,\mathbf{Y}_{III}), & \mathbf{Y}_{III}(t_0) &= \mathbf{Y}_{II}(t_1).
\end{aligned} \tag{1.13}$$

are solved in a sequential order for $t \in [t_0, t_1]$.

It has been shown in [33] that the solutions of the differential equations of (1.13) combine well with the time-dependent SVD-like factorization $\mathbf{Y} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^\top$ of (1.6).

In the first equation, the factor $\mathbf{V}$ is fixed. We evolve in time the product $\mathbf{K} = \mathbf{U}\,\mathbf{S}$ followed by a QR decomposition to recover the single factors.

In the middle equation, the factors $\mathbf{U}$ and $\mathbf{V}$ are fixed and the factor $\mathbf{S}$ is updated.

The third equation is equal to the first one for the transposed problem.

This leads to the construction of an efficient first order in time numerical algorithm, which we refer to as the matrix projector-splitting integrator.

One time step from t_0 to $t_1 = t_0 + h$ starting from a factored rank-r matrix $\mathbf{Y}_0 = \mathbf{U}_0\,\mathbf{S}_0\,\mathbf{V}_0^\top$ proceeds as follows.

Algorithm 1: One time step of the matrix projector-splitting integrator

---

Data: $\mathbf{Y}_0 = \mathbf{U}_0\,\mathbf{S}_0\,\mathbf{V}_0^\top$ in factorized form, function $\mathbf{F}(t,\mathbf{Y})$, t_0, t_1
Result: $\mathbf{Y}_1 = \mathbf{U}_1\,\mathbf{S}_1\,\mathbf{V}_1^\top$ in factorized form
begin

Integrate from $t = t_0$ to t_1 the $m \times r$ matrix differential equation

$$\dot{\mathbf{K}}(t) = \mathbf{F}(t, \mathbf{K}(t)\,\mathbf{V}_0^\top)\mathbf{V}_0, \qquad \mathbf{K}(t_0) = \mathbf{U}_0\,\mathbf{S}_0.$$

Compute a QR factorization $\mathbf{K}(t_1) = \mathbf{U}_1\,\widehat{\mathbf{S}}_1$.

Integrate from $t = t_0$ to t_1 the $r \times r$ differential equation

$$\dot{\mathbf{S}}(t) = -\,\mathbf{U}_1^\top\,\mathbf{F}(t, \mathbf{U}_1\,\mathbf{S}(t)\,\mathbf{V}_0^\top)\mathbf{V}_0, \quad \mathbf{S}(t_0) = \widehat{\mathbf{S}}_1.$$

Set $\widetilde{\mathbf{S}}_0 = \mathbf{S}(t_1)$.

Integrate from $t = t_0$ to t_1 the $n \times r$ matrix differential equation

$$\dot{\mathbf{L}}(t) = \mathbf{F}(t, \mathbf{U}_1\,\mathbf{L}(t)^\top)^\top\,\mathbf{U}_1, \qquad \mathbf{L}(t_0) = \mathbf{V}_0\,\widetilde{\mathbf{S}}_0^\top.$$

Compute a QR factorization $\mathbf{L}(t_1) = \mathbf{V}_1\,\mathbf{S}_1^\top$.

The approximation after one time step is given by

$$\mathbf{Y}_1 = \mathbf{U}_1 \, \mathbf{S}_1 \, \mathbf{V}_1^\top .$$

To proceed further, $\mathbf{Y}_1$ is taken as the starting value for the next step, and so on.

We observe that in each step of the matrix projector-splitting integrator a differential problem, whose size is smaller compared to (1.4), has to be solved. The possibly large matrices $\mathbf{Y}_0$ and $\mathbf{Y}_1$ are never formed. Instead, at each step of algorithm 1 only the factors of the decomposition are updated.

The matrix projector-splitting integrator has remarkable properties. It reproduces rank-r matrices exactly, as shown in the following theorem.

Theorem 1 (Exactness property, [33, Theorem 4.1])**.** *Let $\mathbf{A}(t) \in \mathbb{R}^{m\times n}$ be of rank r for $t_0 \le t \le t_1$, so that $\mathbf{A}(t)$ has a factorization* (1.6)*, $\mathbf{A}(t) = \mathbf{U}(t)\mathbf{S}(t)\,\mathbf{V}(t)^\top$. Moreover, assume that the $r \times r$ matrices $\mathbf{U}(t_1)^\top \mathbf{U}(t_0)$ and $\mathbf{V}(t_1)^\top \mathbf{V}(t_0)$ are invertible. With $\mathbf{Y}_0 = \mathbf{A}(t_0)$, the matrix projector-splitting integrator for $\dot{\mathbf{Y}}(t) = \mathrm{P}(\mathbf{Y}(t))\dot{\mathbf{A}}(t)$ is then exact: $\mathbf{Y}_1 = \mathbf{A}(t_1)$.*

In principle, the three terms in the alternating sum of the orthogonal projection $\mathrm{P}(\mathbf{Y})$ of (1.7) onto the tangent space can be permuted, which leads to the construction of alternative algorithms. The chosen order is the only one that leads to the derivation of a numerical integrator preserving the exactness property [33].

The exactness property is required to prove a second remarkable property which justifies the existence, the analysis and the extension to the multi-dimensional case of the matrix projector-splitting integrator.

The matrix projector-splitting integrator is robust with respect to the presence of small singular values of the solution or its approximation, in contrast with standard integrators applied to (1.8) or the equivalent differential equations for the factors $\mathbf{S}(t)$, $\mathbf{U}(t)$, $\mathbf{V}(t)$, containing a factor $\mathbf{S}(t)^{-1}$ on the right-hand sides of (1.11).

Theorem 2 (Robust error bound, [24, Theorem 2.1])**.** *Let $\mathbf{A}(t)$ denote the solution of the matrix differential equation* (1.4)*. Assume that the following conditions:*

1. *$\mathbf{F}$ is Lipschitz-continuous and bounded: for all $\mathbf{Y}, \widetilde{\mathbf{Y}} \in \mathbb{R}^{m\times n}$ and $0 \le t \le T$,*

$$\| \mathbf{F}(t,\mathbf{Y}) - \mathbf{F}(t,\widetilde{\mathbf{Y}})\| \le L \| \mathbf{Y} - \widetilde{\mathbf{Y}}\|, \qquad \| \mathbf{F}(t,\mathbf{Y})\| \le B \ .$$

2. *The non-tangential part of $\mathbf{F}(t,\mathbf{Y})$ is ε-small:*

$$\|(\mathbf{I} - \mathrm{P}(\mathbf{Y}))\,\mathbf{F}(t,\mathbf{Y})\| \le \varepsilon$$

   for all $\mathbf{Y} \in \mathcal{M}_r$ in a neighbourhood of $\mathbf{A}(t)$ and $0 \le t \le T$.

3. *The error in the initial value is δ-small:*

$$\| \mathbf{Y}_0 - \mathbf{A}_0\| \le \delta.$$

Let $\mathbf{Y}_n$ denote the rank-r approximation to $\mathbf{A}(t_n)$ at $t_n = nh$ obtained after n steps of the matrix projector-splitting integrator with step-size $h > 0$. Then, the error satisfies for all n with $t_n = nh \le T$

$$\| \mathbf{Y}_n - \mathbf{A}(t_n)\| \le c_0\delta + c_1\varepsilon + c_2 h,$$

where the constants c_i only depend on L, B, and T. In particular, the constants are independent of singular values of the exact or approximate solution.

In [24, Section 2.6.3] it is proved that the error bound obtained via an inexact solution of the matrix differential equations, as appearing in the three substeps of the matrix projector-splitting integrator, consists of additional terms. These new terms are proportional to the local errors of the inexact solutions with constants *independent* of small singular values of the solution or its approximation.

1.4 Tensors in Tucker format

The matrix projector-splitting integrator and its properties have been further extended to the multi-dimensional case. For $t_0 \leq t \leq T$, let

$$A(t) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$$

be a continuously differentiable time-dependent tensor. We assume that it is either given explicitly or as a solution of a tensor differential equation

$$\dot{A}(t) = F(t, A(t)), \quad A(t_0) = A^0. \tag{1.14}$$

There exist different low-rank factorizations for tensors [17]. We start presenting the Tucker decomposition, a high-order generalization of the reduced singular value decomposition for matrices. Meanwhile, it is rather straightforward to work and do computations with matrices in SVD-like format, a new set of tools needs to be introduced for tensors in Tucker format.

Let $Y \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a tensor. We present two fundamental operations: matricization and its inverse, tensorization.

Matricization in the ith mode is the realization of a tensor Y in a matrix format, and is indicated by the function $\mathbf{Mat}_i$. It is attained by inserting in co-lexicographic order in the lth row (for $l = 1, \ldots, n_i$), all elements of Y that have the index l in the ith position. The reciprocal operation, the tensorization in the ith mode of a matrix, is indicated by the function Ten_i:

$$\mathbf{Y}_{(i)} = \mathbf{Mat}_i(Y) \in \mathbb{R}^{n_i \times n_{\neg i}} \quad \text{if and only if} \quad Y = \mathrm{Ten}_i(\mathbf{Y}_{(i)}) \in \mathbb{R}^{n_1 \times \cdots \times n_d}.$$

where $n_{\neg i} = \prod_{j \neq i} n_j$.

For a tensor $Y \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, the multilinear rank $\mathbf{r} = (r_1, \ldots, r_d)$ is defined as the d-tuple of the ranks r_i of the matricizations $\mathbf{Mat}_i(Y) \in \mathbb{R}^{n_i \times n_{\neg i}}$ for $i = 1, \ldots, d$.

It follows from [9, 30] that a tensor $Y \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ of multilinear rank $\mathbf{r} = (r_1, \ldots, r_d)$ can be decomposed as a tensor in Tucker format

$$Y = C \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \cdots \times_d \mathbf{U}_d = C \bigtimes_{i=1}^{d} \mathbf{U}_i, \tag{1.15}$$

$$\text{i.e.,} \quad y_{k_1,\ldots,k_d} = \sum_{l_1=1}^{r_1} \cdots \sum_{l_d=1}^{r_d} c_{l_1,\ldots,l_d} u_{k_1,l_1} \ldots u_{k_d,l_d},$$

where the basis matrices $\mathbf{U}_i \in \mathbb{R}^{n_i \times r_i}$ have orthonormal columns and the core tensor $C \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ has full multilinear rank $\mathbf{r} = (r_1, \ldots, r_d)$.

The tensors of dimensions $(n_1, \ldots, n_d)$ and fixed multilinear rank $\mathbf{r} = (r_1, \ldots, r_d)$ form a smooth embedded manifold in $\mathbb{R}^{n_1 \times \cdots \times n_d}$ denoted by $\mathcal{M}_{\mathbf{r}}$ [29].

1.5 Tucker numerical integrators

Dynamical low-rank approximation has been extended to tensors in Tucker format in [29]. The orthogonal projection $\mathrm{P}(Y)$ onto the tangential space of the manifold $\mathcal{M}_{\mathbf{r}}$ is given as an alternating sum of $(2d+1)$ terms

$$\mathrm{P}(Y) = \sum_{i=1}^{d} \Big(\mathrm{P}_i^+(Y) - \mathrm{P}_i^-(Y) \Big) + \mathrm{P}_0(Y) \ . \tag{1.16}$$

We refer to [29, 32] for a detailed derivation and description of each term of the sum. Like in the matrix case, dynamical low-rank approximation of (1.14) for tensors in Tucker format can be recast as a differential equation

$$\dot{Y}(t) = \mathrm{P}(Y(t))F(t, Y(t)), \quad Y(t_0) = Y^0 \in \mathcal{M}_{\mathbf{r}}, \tag{1.17}$$

where $Y^0 \in \mathcal{M}_{\mathbf{r}}$ is an approximation to the initial data A^0 of (1.14).

Then, following the same strategy as in the matrix case, a Lie-Trotter splitting is applied to (1.17). In the framework of quantum dynamics, a projector-splitting numerical integrator for tensors in Tucker format has been first obtained in [32]. Such an integrator has then been reformulated in [36], which is called the nested Tucker integrator. It has been proven in [36, Section 6] that these two formulations are equivalent. Let

$$Y^0 = C^0 \bigtimes_{i=1}^{d} \mathbf{U}_i^0 \in \mathcal{M}_{\mathbf{r}}$$

be an approximation to the initial data $A^0 \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. Both numerical integrators share an analogous approach, which can be summarized in the following general scheme:

Set $C_0^0 = C^0$.
For $i = 1, \ldots, d$, update $\mathbf{U}_i^0 \to \mathbf{U}_i^1$ and modify $C_{i-1}^0 \to C_i^0$.
Update $C_d^0 \to C^1$.

For any intermediate substep of the proposed scheme, just one factor of the decomposition is updated. Except for the core tensor, the others remain untouched.

After one time step, this schematic numerical procedure gives an approximation $Y^1 = C^1 \bigtimes_{i=1}^{d} \mathbf{U}_i^1 \in \mathcal{M}_{\mathbf{r}}$ to $A(t_1)$. The proposed scheme is further iterated in time to obtain successive factorized approximations $Y^n \in \mathcal{M}_{\mathbf{r}}$ to $A(t_n)$.

The details hidden inside the update processes of the Tucker numerical integrators are not trivial and must be carefully organized. Since the two numerical integrators are equivalents to each other, we focus our attention on the nested Tucker integrator. To reduce the complexity of explanation and simplify the description of the nested Tucker integrator, we present the subflows $\Phi^{(i)}$ and Ψ.

For $i = 1, \dots, d$, the subflow $\Phi^{(i)}$ corresponds to the update process of the i-th basis matrix and respective modification of the core tensor.

Algorithm 2: Subflow $\Phi^{(i)}$

Data: $Y^0 = C^0 \bigtimes_{j=1}^{d} \mathbf{U}_j^0$ in factorized form, $F(t, Y), t_0, t_1$
Result: $Y^1 = C^1 \bigtimes_{j=1}^{d} \mathbf{U}_j^1$ in factorized form
begin
set $\mathbf{U}_j^1 = \mathbf{U}_j^0 \quad \forall j \neq i$
compute the QR decomposition $\mathbf{Mat}_i(C^0)^\top = \mathbf{Q}_i^0 \, \mathbf{S}_i^{0,\top} \in \mathbb{R}^{r_{\neg i} \times r_i}$
set $\mathbf{K}_i^0 = \mathbf{U}_i^0 \, \mathbf{S}_i^0 \in \mathbb{R}^{n_i \times r_i}$
solve the $n_i \times r_i$ matrix differential equation
$\dot{\mathbf{K}}_i(t) = \mathbf{F}_i(t, \mathbf{K}_i(t))$ with initial value $\mathbf{K}_i(t_0) = \mathbf{K}_i^0$
and return $\mathbf{K}_i^1 = \mathbf{K}_i(t_1)$; here
$\mathbf{F}_i(t, \mathbf{K}_i) = \mathbf{Mat}_i(F(t, \mathrm{Ten}_i(\mathbf{K}_i(t)\mathbf{V}_i^{0,\top})))\mathbf{V}_i^0$ with
$\mathbf{V}_i^{0,\top} = \mathbf{Mat}_i(\mathrm{Ten}_i(\mathbf{Q}_i^{0,\top}) \bigtimes_{j \neq i} \mathbf{U}_j^0)$
compute the QR decomposition $\mathbf{K}_i^1 = \mathbf{U}_i^1 \, \widehat{\mathbf{S}}_i^1$
solve the $r_i \times r_i$ matrix differential equation
$\dot{\mathbf{S}}_i(t) = -\widehat{\mathbf{F}}_i(t, \mathbf{S}_i(t))$ with initial value $\mathbf{S}_i(t_0) = \widehat{\mathbf{S}}_i^1$
and return $\widetilde{\mathbf{S}}_i^0 = \mathbf{S}_i(t_1)$; here
$\widehat{\mathbf{F}}_i(t, \mathbf{S}_i) = \mathbf{U}_i^{1,\top} \mathbf{F}_i(t, \mathbf{U}_i^1 \, \mathbf{S}_i)$
set $C^1 = \mathrm{Ten}_i(\widetilde{\mathbf{S}}_i^0 \, \mathbf{Q}_i^{0,\top})$
end

The subflow Ψ describes how to update the core tensor. This can be interpreted as a Galerkin projection of the given core on the new updated basis set.

Algorithm 3: Subflow Ψ

Data: $Y^0 = C^0 \bigtimes_{j=1}^{d} \mathbf{U}_j^0$ in factorized form, $F(t, Y), t_0, t_1$
Result: $Y^1 = C^1 \bigtimes_{j=1}^{d} \mathbf{U}_j^1$ in factorized form
begin
set $\mathbf{U}_j^1 = \mathbf{U}_j^0 \quad \forall j = 1, \dots, d.$
solve the $r_1 \times \cdots \times r_d$ tensor differential equation
$\dot{C}(t) = \widetilde{F}(t, C(t))$ with initial value $C(t_0) = C^0$
and return $C^1 = C(t_1)$; here
$\widetilde{F}(t, C) = F(t, C \bigtimes_{j=1}^{d} \mathbf{U}_j^1) \bigtimes_{j=1}^{d} \mathbf{U}_j^{1,\top}$
end

Finally, the approximation after one time step is given by

$$Y^1 = \Psi \circ \Phi^{(d)} \circ \cdots \circ \Phi^{(1)}(Y^0) \, . \tag{1.18}$$

To proceed further in time, $Y^1 \in \mathcal{M}_{\mathbf{r}}$ is taken as the starting value for the next step. We refer to [36, 7] for a detailed description of these subflows.

Because the construction of the nested Tucker integrator is based on the matrix projector-splitting integrator [36, Section 3], the nested Tucker integrator inherits the exactness property.

Theorem 3 (Exactness property, [36, Theorem 4.1]). *Let $A(t) = C(t) \bigtimes_{i=1}^{d} \mathbf{U}_i(t)$ be of multilinear rank $(r_1, \ldots, r_d)$ for $t_0 \le t \le t_1$. Moreover, assume that the $r_i \times r_i$ matrix $\mathbf{U}_i(t_1)^\top \mathbf{U}_i(t_0)$ is invertible for each $i = 1, \ldots, d$. With $Y^0 = A(t_0)$, the nested Tucker integrator with rank $(r_1, \ldots, r_d)$ for $\dot{Y}(t) = \mathrm{P}(Y(t))\dot{A}(t)$ with starting value $Y^0 = A(t_0)$ is then exact: $Y^1 = A(t_1)$.*

Analogously, the nested Tucker integrator inherits from the matrix projector-splitting integrator the robustness with respect to small singular values in the matricizations of the core tensor.

Theorem 4 (Robust error bound, [36, Theorem 5.1]). *Let $A(t)$ denote the solution of the tensor differential equation* (1.14)*. Assume the following conditions:*

1. *The function F is Lipschitz-continuous and bounded.*

$$\begin{aligned} &\|F(t,Y) - F(t,\widetilde{Y})\| \le L\|Y - \widetilde{Y}\| && \text{for all } Y, \widetilde{Y} \in \mathcal{M}_{\boldsymbol{r}}, \\ &\|F(t,Y)\| \le B && \text{for all } Y \in \mathcal{M}_{\boldsymbol{r}}. \end{aligned}$$

2. *The non-tangential part of $F(t,Y)$ is ε-small:*

$$\|(I - \mathrm{P}(Y))F(t,Y)\| \le \varepsilon$$

   for all $Y \in \mathcal{M}_{\boldsymbol{r}}$ in a neighbourhood of $A(t)$ and $0 \le t \le T$.

3. *The error in the initial value is δ-small:*

$$\|Y^0 - A^0\| \le \delta.$$

Let Y^n denote the approximation of multilinear rank $(r_1, \ldots, r_d)$ to $A(t_n)$ at $t_n = nh$ obtained after n steps of the nested Tucker integrator with step-size $h > 0$. Then, the error satisfies for all n with $t_n = nh \le T$

$$\|Y^n - A(t_n)\| \le c_0\delta + c_1\varepsilon + c_2 h,$$

where the constants c_i only depend on the Lipschitz constant L and bound B of F, on T, and on the dimension d. In particular, the constants are independent of singular values of matricizations of the exact or approximate solution.

Chapter 2

Tree tensor network integrator

Despite their interesting properties, tensors in Tucker format cannot be used to deal with high-dimensional problems. In fact, they suffer of an undesired effect known as *curse of dimensionality*, the uncontrollable growth in the amount of data to be stored and treated. The size of a core tensor in a Tucker decomposition is of magnitude $\mathcal{O}(r^d)$, where r is the maximum value of the multilinear rank in each dimension and d is the order of the tensor. Even for moderate ranks, an exponential growth in the amount of data for large d is encountered.

In order to deal with this difficulty, recursive formats have been proposed in [20, 16]: tensors in hierarchical Tucker format, shortly denoted HT.

The HT format consists of a multi-level and recursive Tucker decomposition where its configuration is described by a *binary* tree. A peculiar subclass is given by the so-called tensor trains; already known in physics as matrix product states [40], they have been recently re-proposed to the mathematical community in [39]. Its linear structure allows for an efficient implementation and made it a viable option for treating large high-dimensional problems.

Dynamical low-rank approximation has been extended to tensors in HT format in [35] and a remarkable extension of the matrix projector-splitting integrator to tensor trains has been derived in [34].

However, in various applications such as quantum dynamics and kinetic equations, the tensor train format is not sufficient. The ansatz of the solution or the structure of the problem needs to be described by a general tree representation [41, 42, 11]. Therefore, it is of interest to introduce a general class of *tree* tensor networks, shortly denoted by the acronym TTNs.

In the framework of kinetic equations, a first extension of the matrix projector-splitting integrator to TTNs with specific tree structure and an ad hoc formulation can be found in [11].

In the current chapter, based on contribution [7], the matrix and Tucker projector-splitting integrator together with their properties are extended to tree tensor networks. This derivation has been not obtained via a projector-splitting ansatz but rather follows a recursive algorithmic approach already presented in [36, 11].

Dealing with nested and recursive structures such as TTNs demands a convenient notation. A new one, useful for this purposes, is going to be presented from scratch.

2.1 Tree Tensor networks

A tensor network can be informally described as a weighted graph where to each node, a certain value, represented by a tensor or a matrix, is associated.

A tree tensor network is a peculiar instance of tensor network, where its configuration is described by a tree: each node in the network admits one and only one parent node.

A formal definition of TTN as multi-level Tucker tensor has been provided in [7, Section 2]. For convenience of the reader, we recall verbatim the definition. We start providing the definition of the set of trees $\mathcal{T}$.

Definition 1 (Ordered trees with unequal leaves)**.** *Let $\mathcal{L}$ be a given finite set, the elements of which are referred to as leaves. We define the set $\mathcal{T}$ of trees τ with the corresponding set of leaves $L(\tau) \subseteq \mathcal{L}$ recursively as follows:*

(i) Leaves are trees: $\mathcal{L} \subset \mathcal{T}$, *and* $L(\ell) := \{\ell\}$ *for each* $\ell \in \mathcal{L}$.

(ii) Ordered m-tuples of trees with different leaves are again trees: *If, for some* $m \geq 2$,

$$\tau_1, \ldots, \tau_m \in \mathcal{T} \quad \text{with} \quad L(\tau_i) \cap L(\tau_j) = \emptyset \quad \forall i \neq j,$$

then their ordered m-tuple is in $\mathcal{T}$:

$$\tau := (\tau_1, \ldots, \tau_m) \in \mathcal{T}, \quad \text{and} \quad L(\tau) := \dot{\bigcup}_{i=1}^{m} L(\tau_i) \ .$$

We denote by $T(\tau)$ the set of all subtrees of a tree $\tau \in \mathcal{T}$, including τ. On the set of trees $\mathcal{T}$ we define a partial ordering by writing, for $\sigma, \tau \in \mathcal{T}$,

$$\begin{aligned} \sigma \leq \tau \quad &\text{if and only if} \quad \sigma \in T(\tau), \\ \sigma < \tau \quad &\text{if and only if} \quad \sigma \leq \tau \ \text{ and } \ \sigma \neq \tau. \end{aligned}$$

We set a maximal tree $\bar{\tau} \in \mathcal{T}$ (with set of leaves $L(\bar{\tau}) = \mathcal{L}$) and associate the following elements to any of its leaves and subtrees:

1. To each leaf $\ell \in \mathcal{L}$ we associate a dimension n_ℓ, a rank $r_\ell \leq n_\ell$ and a *basis matrix* $\mathbf{U}_\ell \in \mathbb{R}^{n_\ell \times r_\ell}$ of full rank r_ℓ.

2. To every subtree $\tau = (\tau_1, \ldots, \tau_m) \leq \bar{\tau}$ we associate a rank r_τ and a *connection tensor* $C_\tau \in \mathbb{R}^{r_\tau \times r_{\tau_1} \times \cdots \times r_{\tau_m}}$ of full multilinear rank $(r_\tau, r_{\tau_1}, \ldots, r_{\tau_m})$. We set $r_{\bar{\tau}} = 1$.

Using these quantities, a tree tensor network is constructed via a recursion process that starts from the leaves up to the root of the maximal tree.

Definition 2 (Tree tensor network)**.** *For a given tree $\bar{\tau} \in \mathcal{T}$ and basis matrices $\mathbf{U}_\ell$ and connection tensors C_τ as described in 1. and 2. above, we recursively define a tensor $X_{\bar{\tau}}$ with a tree tensor network representation as follows:*

(i) For each leaf $\tau = \ell \in \mathcal{L}$, we set

$$X_\ell := \mathbf{U}_\ell^\top \in \mathbb{R}^{r_\ell \times n_\ell} \ .$$

(ii) If, for some $m \geq 2$, the tree $\tau = (\tau_1, \ldots, \tau_m)$ is a subtree of $\bar{\tau}$, then we set $n_\tau = \prod_{i=1}^m n_{\tau_i}$ and $\mathbf{I}_\tau$ the identity matrix of dimension r_τ, and

$$\begin{aligned} X_\tau &:= C_\tau \times_0 \mathbf{I}_\tau \bigtimes_{i=1}^m \mathbf{U}_{\tau_i} \in \mathbb{R}^{r_\tau \times n_{\tau_1} \times \cdots \times n_{\tau_m}}, \\ \mathbf{U}_\tau &:= \mathbf{Mat}_0(X_\tau)^\top \in \mathbb{R}^{n_\tau \times r_\tau} \ . \end{aligned}$$

Throughout this work, we fix the convention to count as 0-mode the mode of the dimension r_τ of $X_\tau \in \mathbb{R}^{r_\tau \times n_{\tau_1} \times \cdots \times n_{\tau_m}}$. The 0-mode symbolically represents the connection among the node τ and its parent node inside of the tree tensor network.

For numerical stability, it is convenient to work with orthonormal matrices. Therefore, we introduce the following definition of orthonormal TTN.

Definition 3 (Orthonormal tree tensor network)**.** *A tree tensor network $X_{\bar\tau}$ (more precisely, its representation in terms of the matrices $\mathbf{U}_\tau$) is called* orthonormal *if for each subtree $\tau < \bar\tau$, the matrix $\mathbf{U}_\tau$ has orthonormal columns.*

Let τ be a tree, the set $\mathcal{M}_\tau$ of tensors with an orthonormal tree tensor network representation of given dimensions $(n_\ell)_{\ell \in L(\tau)}$ and fixed ranks $(r_\sigma)_{\sigma \leq \tau}$ form a smooth embedded manifold in the tensor space $\mathbb{R}^{\times_{\ell \in L(\tau)} n_\ell}$ [44, 13, 14].

2.2 Recursive tree tensor network integrator

We now extend the matrix projector-splitting integrator, and the Tucker integrators, to tree tensor networks.

The derivation of the recursive TTN integrator proceeds sequentially from the maximal tree to the leaves. In the first part, for a prescribed initial TTN value, new functions and new initial datas are recursively defined. Then, starting from the leaves and jumping up and down across the network, the values associated to the nodes in the tree tensor network are updated via a recursive application of the nested Tucker integrator. At each step, we deal only with the factors of the decomposition and only small dimensional problems are solved.

We observe that such a derivation is obtained following an analogous technique of [36]; there, the large and untreatable full-dimensional problem (1.14) is reduced to a sequential application of the matrix projector-splitting integrator to the matricization in each mode of a given tensor differential equation. If the dimension of the differential problem appearing in the last L-step of the matrix projector-splitting integrator is too large, the problem is not solved directly but re-formulated ad hoc in a smaller dimensional space, where the matrix projector-splitting integrator is applied again.

In the recursive TTN integrator, we encounter an analogous difficulty in the time-evolution of the factor $\mathbf{U}_\tau \in \mathbb{R}^{n_\tau \times r_\tau}$, appearing in the K-step. Instead of solving this directly, we take advantage of the definition of tree tensor network. We unfold and re-formulate the differential equation appearing in the K-step, solving smaller differential equations for the factors of the network.

We start illustrating the first part of the derivation of the algorithm. For each subtree $\tau = (\tau_1, \ldots, \tau_m) \leq \bar\tau$ we introduce the space

$$\mathcal{V}_\tau := \mathbb{R}^{r_\tau \times n_{\tau_1} \times \cdots \times n_{\tau_m}} \ . \tag{2.1}$$

Let us assume that a prescribed initial value $Y_\tau^0 \in \mathcal{M}_\tau$ and a tensor-valued function $F_\tau : [t_0, t^*] \times \mathcal{V}_\tau \to \mathcal{V}_\tau$ are given. The construction of the new initial value $Y_{\tau_i}^0 \in \mathcal{M}_{\tau_i}$ and the new function $F_{\tau_i} : [t_0, t^*] \times \mathcal{V}_{\tau_i} \to \mathcal{V}_{\tau_i}$ proceeds as follows. Let

$$Y_\tau^0 = C_\tau^0 \times_0 \mathbf{I}_\tau \bigtimes_{i=1}^m \mathbf{U}_{\tau_i}^0 \in \mathcal{M}_\tau \ .$$

Matricize in the i-mode the connecting tensor C_τ^0 and consider its QR decomposition

$$\mathbf{Mat}_i (C_\tau^0)^\top = \mathbf{Q}_{\tau_i}^0 \, \mathbf{S}_{\tau_i}^0 \ .$$

Define the matrix

$$\mathbf{V}^0_{\tau_i} := \mathbf{Mat}_i(\mathrm{Ten}_i(\mathbf{Q}^{0,\top}_{\tau_i}) \times_0 \mathbf{I}_\tau \bigtimes_{j\neq i} \mathbf{U}^0_{\tau_j})^\top \in \mathbb{R}^{r_\tau n_{\neg\tau_i} \times r_{\tau_i}} \ .$$

The formulation of the new tensor-valued function F_{τ_i} and the new initial value $Y^0_{\tau_i}$ can be obtained by defining the *prolongation*

$$\pi_{\tau,i}(Y_{\tau_i}) := \mathrm{Ten}_i\big((\mathbf{V}^0_{\tau_i}\, \mathbf{Mat}_0(Y_{\tau_i}))^\top\big) \in \mathcal{V}_\tau \quad \text{for } \ Y_{\tau_i} \in \mathcal{V}_{\tau_i} \tag{2.2}$$

and the *restriction*

$$\pi^\dagger_{\tau,i}(Z_\tau) := \mathrm{Ten}_0\big((\mathbf{Mat}_i(Z_\tau)\, \mathbf{V}^0_{\tau_i})^\top\big) \in \mathcal{V}_{\tau_i}, \quad \text{for } \ Z_\tau \in \mathcal{V}_\tau. \tag{2.3}$$

The prolongation and the restriction are essential tools in the derivation and implementation of the recursive tree tensor network integrator. They satisfy two central properties.

Lemma 2 ([7, Lemma 4.1])**.** *Let $\tau = (\tau_1, \ldots, \tau_m)$ and $i = 1, \ldots, m$. The restriction $\pi^\dagger_{\tau,i} : \mathcal{V}_\tau \to \mathcal{V}_{\tau_i}$ is both a left inverse and the adjoint (with respect to the tensor Euclidean inner product) of the prolongation $\pi_{\tau,i} : \mathcal{V}_{\tau_i} \to \mathcal{V}_\tau$, that is,*

$$\pi^\dagger_{\tau,i}(\pi_{\tau,i}(Y_{\tau_i})) = Y_{\tau_i} \qquad \text{for all } \ Y_{\tau_i} \in \mathcal{V}_{\tau_i} \tag{2.4}$$

$$\langle \pi_{\tau,i}(Y_{\tau_i}), Z_\tau \rangle_{\mathcal{V}_\tau} = \langle Y_{\tau_i}, \pi^\dagger_{\tau,i}(Z_\tau) \rangle_{\mathcal{V}_{\tau_i}} \quad \text{for all } \ Y_{\tau_i} \in \mathcal{V}_{\tau_i},\ Z_\tau \in \mathcal{V}_\tau. \tag{2.5}$$

Moreover, $\|\pi_{\tau,i}(Y_{\tau_i})\|_{\mathcal{V}_\tau} = \|Y_{\tau_i}\|_{\mathcal{V}_{\tau_i}}$ and $\|\pi^\dagger_{\tau,i}(Z_\tau)\|_{\mathcal{V}_{\tau_i}} \leq \|Z_\tau\|_{\mathcal{V}_\tau}$, where the norms are the tensor Euclidean norms.

These properties represent a fundamental instrument for the theoretical analysis of the proposed algorithm. A detailed derivation of the restriction and prolongation operators can be found in [7, Section 4]. We omit a full description because it will simply obscure the fundamental structure behind the derivation of the recursive TTN integrator. Having introduced these new elements we present the following definition.

Definition 4. *Let $\tau = (\tau_1, \ldots, \tau_m)$ and $i = 1, \ldots, m$. For a given tensor-valued function $F_\tau : [t_0, t^*] \times \mathcal{V}_\tau \to \mathcal{V}_\tau$ and a tree tensor network $Y^0_\tau \in \mathcal{M}_\tau$, we recursively define*

$$F_{\tau_i} = \pi^\dagger_{\tau,i} \circ F_\tau \circ \pi_{\tau,i}$$
$$Y^0_{\tau_i} = \pi^\dagger_{\tau,i}(Y^0_\tau).$$

We have now completed the presentation of the first part of the recursive TTN integrator, the recursive construction of new initial values and functions associated to the nodes of the network. We now present the second part, the time evolution of the values, matrices or connecting tensors, associated with the network.

The integrator is now formulated as a recursive application of the nested Tucker integrator. The recursion process starts from the maximal tree. If we are not on a leaf, we recursively apply the nested Tucker integrator with respect to the given initial value. A new initial value and a new function, as previously defined, are used.

We take advantage of the compact formulation introduced in [7] for the nested Tucker integrator , and we introduce analogous subflows $\Phi_\tau^{(i)}$ and Ψ_τ. The subscript τ indicates the position inside of the tree according where we are located on. Via those subflows, one time step of the *recursive TTN integrator* can be compactly defined as

$$Y_\tau^1 = \Psi_\tau \circ \Phi_\tau^{(m)} \circ \cdots \circ \Phi_\tau^{(1)}(Y_\tau^0) \ . \tag{2.6}$$

The recursion starts from the maximal tree $\bar\tau$, with given $Y_{\bar\tau}^0 = Y^0 \in \mathcal{M}_{\bar\tau}$ and given function $F_{\bar\tau} = F : [t_0, t^*] \times \mathcal{V}_{\bar\tau} \to \mathcal{V}_{\bar\tau}$. The action of the subflow $\Phi_\tau^{(0)}$ is omitted, it is proved in [7, Lemma 3.1] to be trivial. The identity matrix present in the 0-mode does not change.

Similarly to the nested Tucker integrator, for $i = 1, \dots, m$ the subflows $\Phi_\tau^{(i)}$ recursively update the basis matrix $\mathbf{U}_{\tau_i}^0$ and modify the connecting tensor C_τ^0.

Algorithm 4: Subflow $\Phi_\tau^{(i)}$

Data: tree $\tau = (\tau_1, \dots, \tau_m)$, TTN in factorized form
$Y_\tau^0 = C_\tau^0 \times_0 \mathbf{I}_\tau \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^0$ with $\mathbf{U}_{\tau_j}^0 = \mathbf{Mat}_0(X_{\tau_j}^0)^\top$,
function $F_\tau(t, Y_\tau), t_0, t_1$

Result: TTN $Y_\tau^1 = C_\tau^1 \times_0 \mathbf{I}_r \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^1$ with $\mathbf{U}_{\tau_j}^1 = \mathbf{Mat}_0(X_{\tau_j}^1)^\top$
in factorized form

begin

set $\mathbf{U}_{\tau_j}^1 = \mathbf{U}_{\tau_j}^0 \quad \forall j \neq i$

compute the QR factorization $\mathbf{Mat}_i(C_\tau^0)^\top = \mathbf{Q}_{\tau_i}^0 \, \mathbf{S}_{\tau_i}^{0,\top}$

set $Y_{\tau_i}^0 = \pi_{\tau,i}^\dagger(Y_\tau^0) = X_{\tau_i}^0 \times_0 \mathbf{S}_{\tau_i}^{0,\top}$

if $\tau_i = \ell$ is a leaf, **then** solve the $n_\ell \times r_\ell$ matrix differential equation
$\dot Y_{\tau_i}(t) = F_{\tau_i}(t, Y_{\tau_i}(t))$ with initial value $Y_{\tau_i}(t_0) = Y_{\tau_i}^0$
and return $Y_{\tau_i}^1 = Y_{\tau_i}(t_1)$

else
compute $Y_{\tau_i}^1 =$ *Recursive TTN Integrator* $(\tau_i, Y_{\tau_i}^0, F_{\tau_i}, t_0, t_1)$

compute the QR decomposition $\mathbf{Mat}_0(C_{\tau_i}^1)^\top = \widehat{\mathbf{Q}}_{\tau_i}^1 \widehat{\mathbf{S}}_{\tau_i}^1$, where
$C_{\tau_i}^1$ is the connecting tensor of $Y_{\tau_i}^1$

set $\mathbf{U}_{\tau_i}^1 = \mathbf{Mat}_0(X_{\tau_i}^1)^\top$, where the TTN $X_{\tau_i}^1$ is obtained from $Y_{\tau_i}^1$ by
replacing the connecting tensor with $\widehat C_{\tau_i}^1 = \mathrm{Ten}_0(\widehat{\mathbf{Q}}_{\tau_i}^{1,T})$

solve the $r_{\tau_i} \times r_{\tau_i}$ matrix differential equation
$\dot{\mathbf{S}}_{\tau_i}(t) = -\widehat{\mathbf{F}}_{\tau_i}(t, \mathbf{S}_{\tau_i}(t))$ with initial value $\mathbf{S}_{\tau_i}(t_0) = \widehat{\mathbf{S}}_{\tau_i}^1$
and return $\widetilde{\mathbf{S}}_{\tau_i}^0 = \mathbf{S}_{\tau_i}(t_1)$; here
$\widehat{\mathbf{F}}_{\tau_i}(t, \mathbf{S}_{\tau_i}) = \mathbf{U}_{\tau_i}^{1,\top} \mathbf{Mat}_0\big(F_{\tau_i}(t, X_{\tau_i}^1 \times_0 \mathbf{S}_{\tau_i}^\top)\big)^\top$

set $C_\tau^1 = \mathrm{Ten}_i(\widetilde{\mathbf{S}}_{\tau_i}^0 \, \mathbf{Q}_{\tau_i}^{0,\top})$

end

The subflow Ψ_τ is similar to the one defined for the nested Tucker integrator. It suffices to replace the function F with F_τ, the input and output data are now given by TTNs.

Algorithm 5: Subflow Ψ_τ

Data: tree $\tau = (\tau_1, \dots, \tau_m)$, TTN in factorized form
$Y_\tau^0 = C_\tau^0 \times_0 \mathbf{I}_\tau \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^0$ with $\mathbf{U}_{\tau_j}^0 = \mathbf{Mat}_0(X_{\tau_j}^0)^\top$,
function $F_\tau(t, Y_\tau), t_0, t_1$

Result: TTN $Y_\tau^1 = C_\tau^1 \times_0 \mathbf{I}_\tau \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^1$ with $\mathbf{U}_{\tau_j}^1 = \mathbf{Mat}_0(X_{\tau_j}^1)^\top$
in factorized form

begin

set $\mathbf{U}_{\tau_j}^1 = \mathbf{U}_{\tau_j}^0 \quad \forall j = 1, \dots, m.$

solve the $r_\tau \times r_{\tau_1} \times \cdots \times r_{\tau_m}$ tensor differential equation

$\dot{C}_\tau(t) = \widetilde{F}_\tau(t, C_\tau(t))$ with initial value $C_\tau(t_0) = C_\tau^0$

and return $C_\tau^1 = C_\tau(t_1)$; here

$\widetilde{F}_\tau(t, C_\tau) = F_\tau(t, C_\tau \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^1) \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^{1,\top}$

end

We refer to [7, Section 4] for a detailed description of these subflows.

2.3 Exactness and Robustness

The recursive TTN integrator inherits the exactness property and the robustness with respect to small singular values of the matrix projector-splitting integrator.

For $t_0 \le t \le t_1$, let $A_{\bar\tau}(t) = A(t) \in \mathcal{V}_{\bar\tau}$ be a continuously differentiable time-dependent family of tree tensor networks of full tree rank $(r_\tau)_{\tau \le \bar\tau}$. For each subtree $\tau = (\tau_1, \dots, \tau_m) \le \bar\tau$ we recursively define

$$A_{\tau_i}(t) := \pi_{\tau,i}^\dagger\big(A_\tau(t)\big).$$

The restriction $\pi_{\tau,i}^\dagger$ computed with respect to $Y_\tau^0 = A_\tau(t_0)$ has been proven to retain the full tree rank of the initial data [7, Lemma 4.3], i.e.

$$A_\tau(t_0) \text{ has full tree rank } (r_\sigma)_{\sigma \le \tau} \text{ for every subtree } \tau \le \bar\tau. \tag{2.7}$$

We impose analogous condition at time $t_1 > t_0$:

$$A_\tau(t_1) \text{ has full tree rank } (r_\sigma)_{\sigma \le \tau} \text{ for every subtree } \tau \le \bar\tau. \tag{2.8}$$

We thus arrive at the following theorem.

Theorem 5 (Exactness, [7, Theorem 5.1])**.** *Let $A(t)$ be a continuously differentiable time-dependent family of tree tensor networks $A(t)$ of full tree rank $(r_\tau)_{\tau \le \bar\tau}$ for $t_0 \le t \le t_1$, and suppose that the non-degeneracy condition (2.8) is satisfied. Then the recursive TTN integrator used with the same tree rank $(r_\tau)_{\tau \in T(\bar\tau)}$ for $F(t, Y) = \dot{A}(t)$ is exact: starting from $Y^0 = A(t_0)$ we obtain $Y^1 = A(t_1)$.*

The proof of the exactness property for the recursive TTN integrator is based on the exactness result for the nested Tucker integrator [36], which relies on the exactness result of the matrix projector-splitting integrator [33].

The result is obtained via an induction argument on the height of the tree. A tree of height one, corresponds to a tensor in Tucker format. There, the exactness property holds and is then transferred to the remaining parts of the tree tensor network. We refer to [7, Section 5] for more details.

Analogously, because the construction of the recursive TTN integrator is based on a recursive application of the nested Tucker integrator, the recursive TTN integrator preserves the robustness with respect to the presence of small singular values in the matricization of the connecting tensors, as stated in the following theorem.

Theorem 6 (Robust error bound, [7, Theorem 6.1])**.** *Let $A(t)$ denote the solution of the tensor differential equation* (1.14)*. Assume the following conditions:*

1. *The function $F : [t_0, t^*] \times \mathcal{M}_{\bar{\tau}} \to \mathcal{V}_{\bar{\tau}}$ is Lipschitz-continuous and bounded.*

$$\begin{aligned} &\|F(t,Y) - F(t,\widetilde{Y})\| \leq L\|Y - \widetilde{Y}\| && \text{for all } Y, \widetilde{Y} \in \mathcal{M}_{\bar{\tau}}, \\ &\|F(t,Y)\| \leq B && \text{for all } Y \in \mathcal{M}_{\bar{\tau}}. \end{aligned}$$

2. *The non-tangential part of $F(t,Y)$ is ε-small:*

$$\|(I - \mathrm{P}(Y))F(t,Y)\| \leq \varepsilon$$

   for all $Y \in \mathcal{M}_{\bar{\tau}}$ in a neighbourhood of $A(t)$ and $0 \leq t \leq T$.

3. *The error in the initial value is δ-small:*

$$\|Y^0 - A^0\| \leq \delta.$$

Under the above assumptions, the error of the numerical approximation Y^n at $t_n = nh$, obtained with n time steps of the recursive TTN integrator with step size $h > 0$, is bounded by

$$\|Y^n - A(t_n)\| \leq c_0\delta + c_1\varepsilon + c_2 h \qquad \text{for } \; t_n \leq t^*,$$

where c_i depend only on L, B, t^*, and the tree $\bar{\tau}$.

The robustness proof is also obtained via an induction argument on the height of the tree, in combination with the robustness result of the nested Tucker integrator. We refer to [7, Section 6] for more details.

Chapter 3

An unconventional robust integrator

The matrix projector-splitting integrator of Section 1.3 encounters certain inevitable limitations, arising from its derivation and construction. In the present chapter, based on contributions [5] and [6], three areas of improvement are investigated:

- We observe that each step of the matrix projector-splitting integrator has to be performed in sequential order. There is no parallelism present at the level of the numerical algorithm.
- The differential equation in the middle S-step is solved backward in time. When considering strong dissipative problems, this leads to instabilities [2].
- The matrix projector-splitting integrator does not preserve symmetry or anti-symmetry if the differential equation (1.4) does. Cases of interest when the underlying description of the considered problem requires this property to be preserved, e.g. Lyapunov matrix differential equations [37] or quantum dynamics of bosonic or fermionic systems [1, 4], where the wave function is approximated by a symmetric or anti-symmetric Tucker tensor.

In [6] a non-trivial modification of the matrix projector-splitting integrator, with the same favourable exactness and robustness property, is presented. The new integrator tackles the mentioned issues and offers the following advantages:

1. Time-integration of the matrix differential equations appearing in the K-step and L-step, and the two QR decompositions are performed in parallel.
2. The differential equation for the S-step is solved forward in time.
3. The integrator preserves (skew-)symmetry if the differential equation does.

Contrary to the matrix projector-splitting integrator, no time-reversible numerical integrator is yet known for this new version.

3.1 The unconventional matrix integrator

The derivation of the unconventional matrix integrator finds its origin in [5]. There, a non-trivial modification of the matrix projector-splitting integrator preserving symmetry or skew-symmetry for matrices and Tucker tensors was presented.

The proposed (skew-)symmetry preserving numerical integrator has been proven to be robust as the matrix projector-splitting integrator. The unconventional matrix integrator has been obtained as a non-immediate extension of these results.

One time step from time t_0 to $t_1 = t_0 + h$ starting from a rank-r matrix $\mathbf{Y}_0 = \mathbf{U}_0 \mathbf{S}_0 \mathbf{V}_0^\top$ computes an updated rank-r factorization $\mathbf{Y}_1 = \mathbf{U}_1 \mathbf{S}_1 \mathbf{V}_1^\top$ as follows.

Algorithm 6: One time step of the unconventional matrix integrator

Data: $\mathbf{Y}_0 = \mathbf{U}_0\,\mathbf{S}_0\,\mathbf{V}_0^\top$ in factorized form, function $\mathbf{F}(t,\mathbf{Y})$, t_0, t_1
Result: $\mathbf{Y}_1 = \mathbf{U}_1\,\mathbf{S}_1\,\mathbf{V}_1^\top$ in factorized form
begin

Integrate from $t = t_0$ to t_1 the $m \times r$ matrix differential equation

$$\dot{\mathbf{K}}(t) = \mathbf{F}(t, \mathbf{K}(t)\,\mathbf{V}_0^\top)\mathbf{V}_0, \qquad \mathbf{K}(t_0) = \mathbf{U}_0\,\mathbf{S}_0.$$

Compute a QR factorization $\mathbf{K}(t_1) = \mathbf{U}_1\,\mathbf{R}_1$.

Integrate from $t = t_0$ to t_1 the $n \times r$ matrix differential equation

$$\dot{\mathbf{L}}(t) = \mathbf{F}(t, \mathbf{U}_0\,\mathbf{L}(t)^\top)^\top\,\mathbf{U}_0, \qquad \mathbf{L}(t_0) = \mathbf{V}_0\,\mathbf{S}_0^\top.$$

Compute a QR factorization $\mathbf{L}(t_1) = \mathbf{V}_1\,\widetilde{\mathbf{R}}_1$.

Integrate from $t = t_0$ to t_1 the $r \times r$ differential equation

$$\begin{aligned}\dot{\mathbf{S}}(t) &= \mathbf{U}_1^\top\,\mathbf{F}(t, \mathbf{U}_1\,\mathbf{S}(t)\,\mathbf{V}_1^\top)\mathbf{V}_1,\\ \mathbf{S}(t_0) &= \mathbf{U}_1^\top\,\mathbf{Y}_0\,\mathbf{V}_1 = (\mathbf{U}_1^\top\,\mathbf{U}_0)\mathbf{S}_0(\mathbf{V}_1^\top\,\mathbf{V}_0)^\top.\end{aligned}$$

Set $\mathbf{S}_1 = \mathbf{S}(t_1)$.

The approximation after one time step is given by

$$\mathbf{Y}_1 = \mathbf{U}_1\,\mathbf{S}_1\,\mathbf{V}_1^\top\,.$$

To proceed further, $\mathbf{Y}_1$ is taken as the starting value for the next step, and so on.

We observe that the K-step as appearing in the unconventional matrix integrator is identical to the K-step of the matrix projector-splitting integrator: the matrix $\mathbf{U}_1$ computed after one time step is the same.

The differential problem appearing in the L-step is equal to the problem appearing in the K-step via the transposed function $\mathbf{G}(t,\mathbf{Y}) = \mathbf{F}(t,\mathbf{Y}^\top)^\top$, with transposed starting value $\mathbf{Y}_0^\top$. We note that the matrix $\mathbf{V}_1$ computed at the end of the L-step in the unconventional matrix integrator generally differs from the one computed in the L-step of the matrix projector-splitting integrator.

Contrary to the matrix projector-splitting integrator, we observe that the K-step and the L-step are independent to each other and can be solved in parallel. The factors $\mathbf{R}_1$ and $\widetilde{\mathbf{R}}_1$ arising from the QR decompositions are discarded.

The S-step of the unconventional matrix integrator can be interpreted as a Galerkin projection for the differential equation (1.4) into the space of matrices generated by the updated basis $\mathbf{U}_1$ and $\mathbf{V}_1$. Contrary to the matrix projector-splitting integrator, there is no minus sign on the right-hand side of the differential equation appearing in the S-step, which indicates a forward in time integration. The initial value of the S-step in the unconventional matrix integrator differs from the initial value of the S-step in the matrix projector-splitting integrator.

Finally, the tiny factors $\mathbf{U}_1^\top\,\mathbf{U}_0 \in \mathbb{R}^{r\times r}$ and $\mathbf{V}_1^\top\,\mathbf{V}_0 \in \mathbb{R}^{r\times r}$, needed for the initial value in the S-step, can be computed at the end of the K-step and L-step, in parallel.

3.2 Exactness property and robust error bound

In [33, Section 5] it has been observed via numerical experiments that alternative formulations of the matrix projector-splitting integrator do not lead to robustness. Surprisingly, the unconventional matrix integrator shares in a non-trivial manner the same desired properties of the matrix projector-splitting integrator of Section 1.3: the exactness property and the robustness with respect to small singular values.

Theorem 7 (Exactness, [6, Theorem 3])**.** *The exactness property of Theorem* 1 *holds verbatim also for the unconventional matrix integrator.*

Theorem 8 (Robust error bound, [6, Theorem 4])**.** *The robust error bound of Theorem* 2 *holds verbatim also for the unconventional matrix integrator.*

Let $\mathbf{U}_1 \in \mathbb{R}^{m\times r}$ be the matrix computed after one time step of the K-step in the unconventional matrix integrator. The proofs of Theorem 7 and Theorem 8 are, respectively, based on the following auxiliary lemmas [33, 6].

Lemma 3 ([6, Lemma 1])**.** *Let* $\mathbf{A}(t) \in \mathbb{R}^{m\times n}$ *be of rank* r *for* $t_0 \le t \le t_1$*, so that* $\mathbf{A}(t)$ *has a factorization* (1.6)*,* $\mathbf{A}(t) = \mathbf{U}(t)\mathbf{S}(t)\,\mathbf{V}(t)^\top$*. Moreover, assume that the* $r \times r$ *matrix* $\mathbf{V}(t_1)^\top\,\mathbf{V}(t_0)$ *is invertible. Then,*

$$\mathbf{U}_1\,\mathbf{U}_1^\top\,\mathbf{A}(t_1) = \mathbf{A}(t_1).$$

The proof of the robust error bound of the unconventional matrix integrator is based on the following result. Under the assumptions of Theorem 2, let

$$\vartheta := (4e^{Lh}BL + 9BL)h^2 + (3e^{Lh} + 4)\varepsilon h + e^{Lh}\delta \tag{3.1}$$

be the local error bound obtained after one time step of the matrix projector-splitting integrator, as derived in [24, Theorem 2.1].

Lemma 4 ([6, Lemma 2])**.** *Let* $\mathbf{A}_1$ *be the solution at time* $t_1 = t_0 + h$ *of the full problem (1.4) with initial condition* $\mathbf{A}_0$*. Assume that conditions* 1.-3. *of Theorem* 2 *are fulfilled. Then, the following estimate holds:*

$$\|\,\mathbf{U}_1\,\mathbf{U}_1^\top\,\mathbf{A}_1 - \mathbf{A}_1\| \le \vartheta.$$

We observe that the matrix $\mathbf{V}_1$ computed in the L-step is obtained as a solution of the transposed problem appearing in the K-step, with transposed initial value. The results of Lemma 3 and Lemma 4 apply to $\mathbf{V}_1$ with respect to $\mathbf{A}_1^\top$.

The exactness property of the unconventional matrix integrator is obtained as a consequence of the previous observation and the result of Lemma 3. Applying Lemma 4 and the last observation, the following local error bound holds.

Lemma 5 (Local error, [6, Lemma 4])**.** *If* $\mathbf{A}_0 = \mathbf{Y}_0$ *in* (1.4)*, the following local error bound holds:*

$$\|\,\mathbf{Y}_1 - \mathbf{A}_1\| \le h(\hat{c}_1\varepsilon + \hat{c}_2 h),$$

where the constants only depend on L *and* B *and a bound of the step size. In particular, the constants are independent of singular values of the exact or approximate solution.*

Relying on the Lipschitz continuity of the function $\mathbf{F}$ and using a standard argument of Lady Windermere's fan [22, Section II.3], we conclude the proof of Theorem 8 and move from the local error to the global error bound.

3.3 The unconventional Tucker integrator

The unconventional matrix integrator generalizes to tensors in Tucker format.

Given a tensor of multilinear rank $(r_1, \dots, r_d)$ in Tucker format $Y^0 = C^0 \bigtimes_{i=1}^d \mathbf{U}_i^0$, one time step of integration from time t_0 to $t_1 = t_0 + h$ computes $Y^1 = C^1 \bigtimes_{i=1}^d \mathbf{U}_i^1$, in factorized form via

Algorithm 7: One time step of the unconventional Tucker integrator

Data: Tucker tensor $Y^0 = C^0 \bigtimes_{i=1}^d \mathbf{U}_i^0$, $F(t, Y)$, t_0, t_1
Result: Tucker tensor $Y^1 = C^1 \bigtimes_{i=1}^d \mathbf{U}_i^1$
begin
  for $i = 1$ **to** d **do in parallel**
    Compute the QR factorization

$$\mathbf{Mat}_i(C^0)^\top = \mathbf{Q}_i \mathbf{S}_i^{0,\top}.$$

    Define the matrix

$$\mathbf{V}_i^{0,\top} = \mathbf{Mat}_i(\mathrm{Ten}_i(\mathbf{Q}_i^\top) \bigtimes_{j \neq i} \mathbf{U}_j^0) \in \mathbb{R}^{r_i \times n_{\neg i}} \ .$$

    Define the function

$$\mathbf{F}_i(t, \cdot) := \mathbf{Mat}_i \circ F(t, \cdot) \circ \mathit{Ten}_i \ .$$

    Integrate from $t = t_0$ to t_1 the $n_i \times r_i$ matrix differential equation

$$\dot{\mathbf{K}}_i(t) = \mathbf{F}_i(t, \mathbf{K}_i(t)\, \mathbf{V}_i^{0,\top})\, \mathbf{V}_i^0, \qquad \mathbf{K}_i(t_0) = \mathbf{U}_i^0\, \mathbf{S}_i^0.$$

    Perform a QR factorization $\mathbf{K}_i(t_1) = \mathbf{U}_i^1\, \mathbf{R}_i^1$.

    Set $\mathbf{M}_i = \mathbf{U}_i^{1,\top}\, \mathbf{U}_i^0$.

  Integrate from $t = t_0$ to t_1 the $r_1 \times \cdots \times r_d$ tensor differential equation

$$\dot{C}(t) = F\left(t, C(t) \bigtimes_{i=1}^d \mathbf{U}_i^1\right) \bigtimes_{i=1}^d \mathbf{U}_i^{1,\top},$$
$$C(t_0) = Y^0 \bigtimes_{i=1}^d \mathbf{U}_i^{1,\top} = C^0 \bigtimes_{i=1}^d \mathbf{M}_i.$$

  Set $C^1 = C(t_1)$.

To continue in time, we take $Y^1 \in \mathcal{M}_{\mathbf{r}}$ as starting value for the next step.

The unconventional Tucker integrator inherits from the unconventional matrix integrator the exactness property and the robustness with respect to small singular values in the matricization of the core tensors.

Theorem 9 (Exactness, [6, Theorem 6])**.** *The exactness property of Theorem* 3 *holds verbatim also for the unconventional Tucker integrator.*

Theorem 10 (Robust error bound, [6, Theorem 7])**.** *The robust error bound of Theorem* 4 *holds verbatim also for the unconventional Tucker integrator.*

The proof of Theorem 9 and Theorem 10, respectively, follows as a consequence of the matrix-valued results of Lemma 3 and Lemma 4. It suffices to prove that the matrix-valued functions $\mathbf{F}_i(t,\cdot)$ as defined in the course of the unconventional Tucker integrator satisfy conditions 1. - 2. of Theorem 2, see [6, Lemma 5].

3.4 Symmetric and anti-symmetric low-rank Tucker tensors

To conclude, we will briefly discuss the last property of interest of the unconventional Tucker integrator: the unconventional Tucker integrator preserves (anti-)symmetry if the differential equation (1.14) does. Being the matrix setting included in the multi-dimensional case, only the latter one will be presented.

Let $Y = (y_{i_1,\dots,i_d}) \in \mathbb{R}^{n\times\cdots\times n}$ be a tensor of order d. We denote the element-wise action of an element σ in the permutation group S_d on Y as

$$\sigma(Y) := \left(y_{i_{\sigma(1)},\dots,i_{\sigma(d)}}\right) .$$

We thus introduce the following definitions.

Definition 5. *Let $Y \in \mathbb{R}^{n\times\cdots\times n}$ be a tensor of order d. A tensor Y is defined symmetric if $\sigma(Y) = Y$ for all $\sigma \in S_d$.*

Definition 6. *Let $Y \in \mathbb{R}^{n\times\cdots\times n}$ be a tensor of order d. A tensor Y is defined anti-symmetric if $\sigma(Y) = (-1)^{\mathrm{sign}(\sigma)}\, Y$ for all $\sigma \in S_d$.*

We assume that the right-hand side function in (1.14) is such that for all $\sigma \in S_d$ one of the following conditions holds

$$\sigma\big(F(t,\sigma(Y))\big) = F(t,Y) \tag{3.2}$$
$$\sigma\big(F(t,\sigma(Y))\big) = (-1)^{\mathrm{sign}(\sigma)}\, F(t,Y) \tag{3.3}$$

Under the above assumptions, we observe that the solution of (1.14) with (anti-)symmetric initial value remain (anti-)symmetric. The following result holds.

Theorem 11 ([6, Theorem 8])**.** *Let Y^0 be symmetric or anti-symmetric and assume that the function F satisfies property* (3.2) *or* (3.3)*, respectively. Then, the approximation Y^1 obtained after one time step of the unconventional Tucker integrator is symmetric or anti-symmetric, respectively.*

We underline the fact that under condition (3.2) or (3.3), the unconventional matrix and Tucker integrator reduces to the (anti)-symmetry preserving low-rank matrix and Tucker integrator of [5].

Bibliography

[1] O. E. Alon, A. I. Streltsov, and L. S. Cederbaum. Multiconfigurational time-dependent Hartree method for bosons: many-body dynamics of bosonic systems. *Phys. Rev. A*, 77:033613.

[2] M. Bachmayr, H. Eisenmann, E. Kieri, and A. Uschmajew. Existence of dynamical low-rank approximations to parabolic problems. *Math. Comp.*, 2021.

[3] D. Bauernfeind and M. Aichhorn. Time Dependent Variational Principle for Tree Tensor Networks. *SciPost Phys.*, 8:24, 2020.

[4] J. Caillat, J. Zanghellini, M. Kitzler, O. Koch, W. Kreuzer, and A. Scrinzi. Correlated multielectron systems in strong laser fields: a multiconfiguration time-dependent Hartree-Fock approach. *Phys. Rev. A*, 71:012712, Jan 2005.

[5] G. Ceruti and C. Lubich. Time integration of symmetric and anti-symmetric low-rank matrices and Tucker tensors. *BIT Numer. Math.*, 60:591–614, 2020.

[6] G. Ceruti and C. Lubich. An unconventional robust integrator for dynamical low-rank approximation. *BIT Numer. Math.*, 2021.

[7] G. Ceruti, C. Lubich, and H. Walach. Time integration of tree tensor networks. *SIAM J. Numer. Anal.*, 59(1):289–313, 2021.

[8] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21:1253–1278, 2000.

[9] L. De Lathauwer, B. De Moor, and J. Vandewalle. On the best rank-1 and rank-$(R_1, R_2, \cdots, R_N)$ approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000.

[10] L. Dieci and T. Eirola. On smooth decompositions of matrices. *SIAM J.Matrix Anal. Appl.*, 20(3):800–819, 1999.

[11] L. Einkemmer and C. Lubich. A low-rank projector-splitting integrator for the Vlasov-Poisson equation. *SIAM J. Sci. Comput.*, 40(5):B1330–B1360, 2018.

[12] L. C. Evans. *Partial differential equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, second edition, 2010.

[13] A. Falcó, W. Hackbusch, and A. Nouy. Geometric structures in tensor representations (final release). *arXiv:1505.03027*, 2015.

[14] A. Falcó, W. Hackbusch, and A. Nouy. Tree-based tensor formats. *SeMA J.*, pages 1–15, 2018.

[15] F. Feppon and P. F. J. Lermusiaux. A geometric approach to dynamical model order reduction. *SIAM J. Matrix Anal. Appl.*, 39(1):510–538, 2018.

[16] L. Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM J. Matrix Anal. Appl.*, 31:2029–2054, 2010.

[17] L. Grasedyck, D. Kressner, and C. Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

[18] W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*. Springer, 2012.

[19] W. Hackbusch. On the representation of symmetric and antisymmetric tensors. In *Contemporary computational mathematics—a celebration of the 80th birthday of Ian Sloan. Vol. 1, 2*, pages 483–515. Springer, Cham, 2018.

[20] W. Hackbusch and S. Kühn. A new scheme for the tensor representation. *Journal of Fourier analysis and applications*, 15:706–722, 2009.

[21] J. Haegeman, C. Lubich, I. Oseledets, B. Vandereycken, and F. Verstraete. Unifying time evolution and optimization with matrix product states. *Phys. Rev. B*, 94(16):165116, 2016.

[22] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I. Nonstiff problems*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1993.

[23] U. Helmke and J. B. Moore. *Optimization and dynamical systems*. Communications and Control Engineering Series. Springer-Verlag, London, 1994.

[24] E. Kieri, C. Lubich, and H. Walach. Discretized dynamical low-rank approximation in the presence of small singular values. *SIAM J. Numer. Anal.*, 54:1020–1038, 2016.

[25] B. Kloss, I. Burghardt, and C. Lubich. Implementation of a novel projector-splitting integrator for the multi-configurational time-dependent Hartree approach. *J. Chem. Phys.*, 146(17):174107, 2017.

[26] B. Kloss, D. R. Reichman, and Y. B. Lev. Studying dynamics in two-dimensional quantum lattices using tree tensor network states. *SciPost Phys.*, 9:70, 2020.

[27] O. Koch. Efficient computation of the MCTDHF approximation to the time-dependent Schrödinger equation. *Opuscula Math.*, 26(3):483–497, 2006.

[28] O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM J. Matrix Anal. Appl.*, 29(2):434–454, 2007.

[29] O. Koch and C. Lubich. Dynamical tensor approximation. *SIAM J. Matrix Anal. Appl.*, 31:2360–2375, 2010.

[30] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51:455–500, 2009.

[31] C. Lubich. *From quantum to classical molecular dynamics: reduced models and numerical analysis*. Zurich Lectures in Advanced Mathematics. European Mathematical Society (EMS), Zürich, 2008.

[32] C. Lubich. Time integration in the multiconfiguration time-dependent Hartree method of molecular quantum dynamics. *Appl. Math. Res. Express*, 2015:311–328, 2015.

[33] C. Lubich and I. V. Oseledets. A projector-splitting integrator for dynamical low-rank approximation. *BIT Numer. Math.*, 54:171–188, 2014.

[34] C. Lubich, I. V. Oseledets, and B. Vandereycken. Time integration of tensor trains. *SIAM J. Numer. Anal.*, 53:917–941, 2015.

[35] 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.

[36] C. Lubich, B. Vandereycken, and H. Walach. Time integration of rank-constrained Tucker tensors. *SIAM J. Numer. Anal.*, 56:1273–1290, 2018.

[37] H. Mena, A. Ostermann, L. M. Pfurtscheller, and C. Piazzola. Numerical low-rank approximation of matrix differential equations. *J. Comput. Appl. Math.*, 340:602–614, 2018.

[38] H.-D. Meyer, F. Gatti, and G. A. Worth. *Multidimensional quantum dynamics: MCTDH theory and applications*. John Wiley & Sons, 2009.

[39] I. V. Oseledets. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33(5):2295–2317, 2011.

[40] D. Perez-García, F. Verstraete, M. M. Wolf, and J. I. Cirac. Matrix product state representations. *Quantum Information and Computation*, 7(5-6):401–430, 2007.

[41] Y.-Y. Shi, L.-M. Duan, and G. Vidal. Classical simulation of quantum many-body systems with a tree tensor network. *Phys. Rev. A*, 74(2):022320, 2006.

[42] L. Tagliacozzo, G. Evenbly, and G. Vidal. Simulation of two-dimensional quantum systems using a tree tensor network that exploits the entropic area law. *Phys. Rev. B*, 80(23):235127, 2009.

[43] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

[44] A. Uschmajew and B. Vandereycken. The geometry of algorithms using hierarchical tensors. *Linear Algebra Appl.*, 439(1):133–166, 2013.

[45] H. M. Walach. *Time integration for the dynamical low-rank approximation of matrices and tensors*. PhD thesis, Eberhard Karls Universität Tübingen, 2019.

Appendix A

Time integration of tree tensor networks

TIME INTEGRATION OF TREE TENSOR NETWORKS

GIANLUCA CERUTI*, CHRISTIAN LUBICH*, AND HANNA WALACH*

Abstract. Dynamical low-rank approximation by tree tensor networks is studied for the data-sparse approximation of large time-dependent data tensors and unknown solutions to tensor differential equations. A time integration method for tree tensor networks of prescribed tree rank is presented and analyzed. It extends the known projector-splitting integrators for dynamical low-rank approximation by matrices and rank-constrained Tucker tensors and is shown to inherit their favorable properties. The integrator is based on recursively applying the low-rank Tucker tensor integrator. In every time step, the integrator climbs up and down the tree: it uses a recursion that passes from the root to the leaves of the tree for the construction of initial value problems on subtree tensor networks using appropriate restrictions and prolongations, and another recursion that passes from the leaves to the root for the update of the factors in the tree tensor network. The integrator reproduces given time-dependent tree tensor networks of the specified tree rank exactly and is robust to the typical presence of small singular values in matricizations of the connection tensors, in contrast to standard integrators applied to the differential equations for the factors in the dynamical low-rank approximation by tree tensor networks.

Key words. Tree tensor network, tensor differential equation, dynamical low-rank approximation, time integrator

AMS subject classifications. 15A69, 65L05, 65L20, 65L70

1. Introduction. For the approximate solution of the initial value problem for a (huge) system of differential equations for the tensor $A(t) \in \mathbb{R}^{n_1 \times \ldots \times n_d}$,

$$\dot{A}(t) = F(t, A(t)), \tag{1.1}$$

we aim to construct $Y(t) \approx A(t)$ in an approximation manifold $\mathcal{M}$ of much smaller dimension, which in the present work will be chosen as a manifold of tree tensor networks of fixed tree rank. This shall provide a data-sparse computational approach to high-dimensional problems that cannot be treated by direct time integration because of both excessive memory requirements and computational cost.

A differential equation for $Y(t) \in \mathcal{M}$ is obtained by choosing the time derivative $\dot{Y}(t)$ as that element in the tangent space $T_{Y(t)}\mathcal{M}$ for which

$$\|\dot{Y}(t) - F(t, Y(t))\| \quad \text{is minimal,}$$

where the norm is chosen as the Euclidean norm of the vector of the tensor entries. In the quantum physics and chemistry literature, this approach is known as the Dirac–Frenkel time-dependent variational principle, named after work by Dirac in 1930 who used the approach in the context of what is nowadays known as the time-dependent Hartree–Fock method for the multi-particle time-dependent Schrödinger equation; see, e.g., [16, 17]. Equivalently, this minimum-defect condition can be stated as a Galerkin condition on the state-dependent approximation space $T_{Y(t)}\mathcal{M}$,

$$\dot{Y}(t) \in T_{Y(t)}\mathcal{M} \quad \text{such that} \quad \langle \dot{Y}(t) - F(t, Y(t)), Z \rangle = 0 \quad \forall\, Z \in T_{Y(t)}\mathcal{M}.$$

Using the orthogonal projection $P(Y)$ onto the tangent space $T_Y\mathcal{M}$, this can be reformulated as the (abstract) ordinary differential equation on $\mathcal{M}$,

$$\dot{Y}(t) = P(Y(t))F(t, Y(t)). \tag{1.2}$$

*Mathematisches Institut, Universität Tübingen, Auf der Morgenstelle 10, D–72076 Tübingen, Germany. Email: {ceruti,lubich,walach}@na.uni-tuebingen.de

This equation needs to be solved numerically in an efficient and robust way. For fixed-rank matrix and tensor manifolds, the orthogonal projection $P(Y)$ turns out to be an alternating sum of subprojections, which reflects the multilinear structure of the problem. The explicit form of the tangent space projection in the low-rank matrix case as an alternating sum of three subprojections was derived in [14] and was used in [19] to derive a projector-splitting integrator for low-rank matrices, which efficiently updates an SVD-like low-rank factorization in every time step and which is robust to the typically arising small singular values that cause severe difficulties with standard integrators applied to the system of differential equations for the factors of the SVD-like decomposition of the low-rank matrices; see [12]. The projector-splitting integrator was extended to tensor trains / matrix product states in [20]; see also [10] for a description of the algorithm in a physical idiom. The projector-splitting integrator was extended to Tucker tensors of fixed multilinear rank in [18]. A reinterpretation was given in [21], in which the Tucker tensor integrator was rederived by recursively performing inexact substeps in the matrix projector-splitting integrator applied to matricizations of the tensor differential equation followed by retensorization. This interpretation made it possible to show that the Tucker integrator inherits the favorable robustness properties of the low-rank matrix projector-splitting integrator.

In the present paper we take up such a recursive approach to derive an integrator for general tree tensor networks, which is shown to be efficiently implementable (provided that the righthand side function F can be efficiently evaluated on tree tensor networks in factorized form) and to inherit the robust convergence properties of the low-rank matrix, Tucker tensor and tensor train / matrix product state integrators shown previously in [12, 21]. The proposed integrator for tree tensor networks reduces to the well-proven projector-splitting integrators in the particular cases of Tucker tensors and tensor trains / matrix product states. We expect (but will not prove) that it can itself be interpreted as a projector-splitting integrator based on splitting the tangent space projection of the fixed-rank tree tensor network manifold.

For a special tree, this integrator (given in an *ad hoc* formulation) was already used in [5] for the Vlasov–Poisson equation of plasma physics. In that case, the tree is given by the separation $((x_1, x_2, x_3), (v_1, v_2, v_3))$ of the position and velocity variables, which are further separated into their Cartesian coordinates. Very recently, this tree tensor network integrator (or very similar versions) was applied to problems of current interest in quantum physics in [3] and [13], studying multi-orbital Anderson impurity models and the dynamics in two-dimensional quantum lattices, respectively. None of these papers gives a systematic construction and numerical analysis of the integrator. This is the objective of the present paper.

In Section 2 we introduce notation and the formulation of tree tensor networks as multilevel-structured Tucker tensors and give basic properties, emphasizing orthonormal factorizations. The tree tensor network (TTN) is constructed from the basis matrices at the leaves and the connection tensors at the inner vertices of the tree in a multilinear recursion that passes from the leaves to the root of the tree.

In Section 3 we recall the algorithm of the Tucker tensor integrator of [21] and extend it to the case of several Tucker tensors with the same basis matrices. This extended Tucker integrator, which is nothing but the Tucker integrator for an extended Tucker tensor, is a basic building block of the integrator for tree tensor networks.

In Section 4, the main algorithmic section of this paper, we derive the recursive TTN integrator and discuss the basic algorithmic aspects. In every time step, the integrator uses a recursion that passes from the root to the leaves of the tree for the

construction of initial value problems on subtree tensor networks using appropriate restrictions and prolongations, and another recursion that passes from the leaves to the root for the update of the factors in the tree tensor network. The integrator only solves low-dimensional matrix differential equations (of the dimension of the basis matrices at the leaves) and low-dimensional tensor differential equations (of the dimension of the connection tensors at the inner vertices of the tree), alternating with orthogonal matrix decompositions of such small matrices and of matricizations of the connection tensors.

In Section 5 we prove a remarkable exactness property: if $F(t, Y) = \dot{A}(t)$ for a given tree tensor network $A(t)$ of the specified tree rank, then the recursive TTN integrator for this tree rank reproduces $A(t)$ exactly. This exactness property is proved using the analogous exactness property of the Tucker tensor integrator proved in [21], which in turn was proved using the exactness property for the matrix projector-splitting integrator that was discovered and proved in [19].

In Section 6 we prove first-order error bounds that are independent of small singular values of matricizations of the connecting tensors. The proof relies on the similar error bound for the Tucker integrator [21], which in turn relies on such an error bound for the fixed-rank matrix projector-splitting integrator proved in [12], in a proof that uses in an essential way the exactness property. The robustness to small singular values distinguishes the proposed integrator substantially from standard integrators applied to the differential equations for the basis matrices and connection tensors derived in [26]. We note that the proposed TTN integrator foregoes the formulation of these differential equations for the factors. The ill-conditioned density matrices whose inverses appear in these differential equations are never formed, let alone inverted, in the TTN integrator.

The present paper thus completes a path to extend the low-rank matrix projector-splitting integrator of [19], together with its favorable properties, from the dynamical low-rank approximation by matrices of a prescribed rank to Tucker tensors of prescribed multilinear rank to general tree tensor networks of prescribed tree rank.

In Section 7 we present a numerical experiment which shows the error behaviour of the proposed integrator in accordance with the theory. We choose the example of retraction of the sum of a tree tensor network and a tangent network, which is an operation needed in many optimization algorithms for tree tensor networks; cf. [1] for the low-rank matrix case. The corresponding example was already chosen in numerical experiments for the low-rank matrix, tensor train and Tucker tensor cases in [19, 20, 21], respectively. It is beyond the scope of this paper to present the results of numerical experiments with the recursive TTN integrator in actual applications of tree tensor networks in physics, chemistry or other sciences. We note, however, that striking numerical experiments with this integrator have already been reported for the Vlasov–Poisson equation of plasma physics in [5] and for problems in quantum physics in [3, 13].

While we describe the TTN integrator for real tensors, the algorithm and its properties extend in a straightforward way to complex tensors as arise in quantum physics. Only some additional care in using transposes $\mathbf{U}^\top$ versus adjoints $\mathbf{U}^* = \overline{\mathbf{U}}^\top$ is required for this extension.

Throughout the paper, we denote tensors by roman capitals and matrices by boldface capitals.

2. Preparation: Matrices, Tucker tensors, tree tensor networks, and their ranks.

2.1. Matrices of rank r. The singular value decomposition (SVD) shows that a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is of rank r if and only if it can be factorized as

$$\mathbf{A} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^\top,$$

where $\mathbf{U} \in \mathbb{R}^{m\times r}$ and $\mathbf{V} \in \mathbb{R}^{n\times r}$ have orthonormal columns, and $\mathbf{S} \in \mathbb{R}^{r\times r}$ has full rank r. The decomposition is not unique. (The SVD is a particular choice with diagonal $\mathbf{S}$.) The real $m \times n$ matrices of rank (exactly) r are known to form a smooth embedded manifold in $\mathbb{R}^{m\times n}$ [11].

2.2. Tucker tensors of multilinear rank (r_i). For a tensor $A \in \mathbb{R}^{n_1\times\cdots\times n_d}$, the multilinear rank $(r_1, \dots, r_d)$ is defined as the d-tuple of the ranks r_i of the matricizations $\mathbf{Mat}_i(A) \in \mathbb{R}^{n_i\times n_{\neg i}}$ for $i = 1, \dots, d$, where $n_{\neg i} = \prod_{j\neq i} n_j$. We recall that the ith matricization aligns in the kth row (for $k = 1, \dots, n_i$) all entries of A that have the index k in the ith position, usually ordered co-lexicographically. The inverse operation is tensorization of the matrix, denoted by Ten_i:

$$\mathbf{A}_{(i)} = \mathbf{Mat}_i(A) \in \mathbb{R}^{n_i\times n_{\neg i}} \quad \text{if and only if} \quad A = \mathrm{Ten}_i(\mathbf{A}_{(i)}) \in \mathbb{R}^{n_1\times\cdots\times n_d}.$$

It is known from [4] that the tensor A has multilinear rank $(r_1, \dots, r_d)$ if and only if it can be factorized as a *Tucker tensor* (we adopt the shorthand notation from [15])

$$A = C \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \cdots \times_d \mathbf{U}_d = C \bigtimes_{i=1}^{d} \mathbf{U}_i, \tag{2.1}$$
$$\text{i.e.,}\quad a_{k_1,\dots,k_d} = \sum_{l_1=1}^{r_1} \cdots \sum_{l_d=1}^{r_d} c_{l_1,\dots,l_d} u_{k_1,l_1} \dots u_{k_d,l_d},$$

where the *basis matrices* $\mathbf{U}_i \in \mathbb{R}^{n_i\times r_i}$ have orthonormal columns and the *core tensor* $C \in \mathbb{R}^{r_1\times\cdots\times r_d}$ has full multilinear rank $(r_1, \dots, r_d)$. (This requires a compatibility condition among the ranks: $r_i \le \prod_{j\neq i} r_j$. In particular, this condition is satisfied if all ranks r_i are equal.) As in the matrix case ($d = 2$), this decomposition is not unique.

A useful formula for the matricization of Tucker tensors is

$$\mathbf{Mat}_i\big(C \bigtimes_{j=1}^{d} \mathbf{U}_j\big) = \mathbf{U}_i\, \mathbf{Mat}_i(C) \Big(\bigotimes_{j\neq i} \mathbf{U}_j^\top \Big), \tag{2.2}$$

where $\otimes$ denotes the Kronecker product of matrices.

The tensors of given dimensions $(n_1, \dots, n_d)$ and fixed multilinear rank $(r_1, \dots, r_d)$ are known to form a smooth embedded manifold in $\mathbb{R}^{n_1\times\cdots\times n_d}$.

2.3. Orthonormal tree tensor networks of tree rank (r_τ). A tree tensor network is a multilevel-structured Tucker tensor where the configuration is described by a tree. The notion of a 'tree tensor network' was coined in the quantum physics literature [24], but tree tensor networks were actually already used a few years earlier in the multilayer MCTDH method of chemical physics [26]. In the mathematical literature, tree tensor networks with binary trees have been studied as 'hierarchical tensors' [9] and with general trees as 'tensors in tree-based tensor format' [6, 7]. We remark that Tucker tensors and matrix product states / tensor trains [23, 22] are

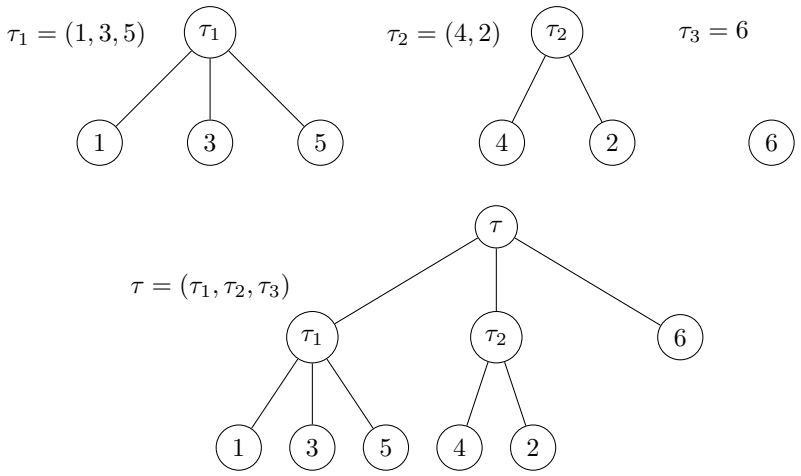


FIG. 2.1. *Graphical representation of a tree and three subtrees with the set of leaves* $\mathcal{L} = \{1, 2, 3, 4, 5, 6\}$.

particular instances of tree tensor networks, whose trees are trees of minimal height (bushes) and binary trees of maximal height, respectively.

In an informal way, one arrives at a tree tensor network by first considering a tensor in Tucker format, in which then the basis matrices are tensorized and approximated by tensors in a low-rank Tucker format. Their basis matrices are again tensorized and approximated by tensors in low-rank Tucker format, and so on over multiple levels. A different viewpoint common in quantum physics is to describe a tree tensor network as resulting from a collection of tensors that have two types of indices: those corresponding to physical degrees of freedom and further auxiliary indices that always appear on two tensors. The graph with the tensors as vertices and the indices as edges is assumed to be a tree, i.e., to have no loops. The tree tensor network is then obtained by contracting over the auxiliary indices.

As there does not appear to exist a firmly established mathematical notation for tree tensor networks, we give a formulation from scratch that turns out useful for the formulation, implementation and analysis of the numerical methods.

DEFINITION 2.1 (Ordered trees with unequal leaves). *Let $\mathcal{L}$ be a given finite set, the elements of which are referred to as leaves. We define the set $\mathcal{T}$ of trees τ with the corresponding set of leaves $L(\tau) \subseteq \mathcal{L}$ recursively as follows:*

(i) Leaves are trees: $\mathcal{L} \subset \mathcal{T}$, *and* $L(\ell) := \{\ell\}$ *for each* $\ell \in \mathcal{L}$.

(ii) Ordered m-tuples of trees with different leaves are again trees: *If, for some* $m \geq 2$,

$$\tau_1, \ldots, \tau_m \in \mathcal{T} \quad \text{with} \quad L(\tau_i) \cap L(\tau_j) = \emptyset \quad \forall i \neq j,$$

then their ordered m-tuple is in $\mathcal{T}$:

$$\tau := (\tau_1, \ldots, \tau_m) \in \mathcal{T}, \quad \text{and} \quad L(\tau) := \dot{\bigcup}_{i=1}^{m} L(\tau_i) \ .$$

The graphical interpretation is that (i) leaves are trees and (ii) every other tree $\tau \in \mathcal{T}$ is formed by connecting a root to several trees with different leaves. (Note that $m = 1$ is excluded: the 1-tuple $\tau = (\tau_1)$ is considered to be identical to τ_1.) $L(\tau)$ is the set of leaves of the tree τ.

The trees $\tau_1, \ldots, \tau_m$ are called direct subtrees of the tree $\tau = (\tau_1, \ldots, \tau_m)$, which together with direct subtrees of direct subtrees of τ etc. are called the *subtrees* of τ. We let $T(\tau)$ be the set of subtrees of a tree $\tau \in \mathcal{T}$, including τ. More formally, we set

$$T(\ell) := \{\ell\} \text{ for } \ell \in \mathcal{L}, \text{ and } T(\tau) := \{\tau\} \,\dot\cup \bigdot_{i=1}^{m} T(\tau_i) \text{ for } \tau = (\tau_1, \ldots, \tau_m).$$

In the graphical interpretation, the subtrees are in a bijective correspondence with the vertices of the tree, by assigning to each subtree its root; see Figure 2.1.

On the set of trees $\mathcal{T}$ we define a partial ordering by writing, for $\sigma, \tau \in \mathcal{T}$,

$$\begin{aligned} \sigma \le \tau \quad &\text{if and only if} \quad \sigma \in T(\tau), \\ \sigma < \tau \quad &\text{if and only if} \quad \sigma \le \tau \ \text{ and } \ \sigma \ne \tau. \end{aligned}$$

We now fix a maximal tree $\bar\tau \in \mathcal{T}$ (with $L(\bar\tau) = \mathcal{L}$). On this tree we work with the following quantities:

1. To each leaf $\ell \in \mathcal{L}$ we associate a dimension n_ℓ, a rank $r_\ell \le n_\ell$ and a *basis matrix* $\mathbf{U}_\ell \in \mathbb{R}^{n_\ell \times r_\ell}$ of full rank r_ℓ.
2. To every subtree $\tau = (\tau_1, \ldots, \tau_m) \le \bar\tau$ we associate a rank r_τ and a *connection tensor* $C_\tau \in \mathbb{R}^{r_\tau \times r_{\tau_1} \times \cdots \times r_{\tau_m}}$ of full multilinear rank $(r_\tau, r_{\tau_1}, \ldots, r_{\tau_m})$. We set $r_{\bar\tau} = 1$.

This can be interpreted as associating a tensor to the *root* of each subtree. In this way every vertex of the given tree carries either a matrix — if it is a leaf — or else a tensor whose order equals the number of edges leaving the vertex. In applications, the basis matrices correspond to the physical degrees of freedom and the connection tensors determine the correlation network.

With these data, a tree tensor network (TTN) is constructed in a recurrence relation that passes from the leaves to the root of the tree.

DEFINITION 2.2 (Tree tensor network). *For a given tree $\bar\tau \in \mathcal{T}$ and basis matrices $\mathbf{U}_\ell$ and connection tensors C_τ as described in 1. and 2. above, we recursively define a tensor $X_{\bar\tau}$ with a tree tensor network representation as follows:*

(i) For each leaf $\tau = \ell \in \mathcal{L}$, we set

$$X_\ell := \mathbf{U}_\ell^\top \in \mathbb{R}^{r_\ell \times n_\ell} \ .$$

(ii) If, for some $m \ge 2$, the tree $\tau = (\tau_1, \ldots, \tau_m)$ is a subtree of $\bar\tau$, then we set $n_\tau = \prod_{i=1}^m n_{\tau_i}$ and $\mathbf{I}_\tau$ the identity matrix of dimension r_τ, and

$$\begin{aligned} X_\tau &:= C_\tau \times_0 \mathbf{I}_\tau \bigtimes_{i=1}^m \mathbf{U}_{\tau_i} \in \mathbb{R}^{r_\tau \times n_{\tau_1} \times \cdots \times n_{\tau_m}}, \\ \mathbf{U}_\tau &:= \mathbf{Mat}_0(X_\tau)^\top \in \mathbb{R}^{n_\tau \times r_\tau} \ . \end{aligned}$$

For short, we refer to the tensor $X_{\bar\tau}$ as a *tree tensor network* on the tree $\bar\tau$.

The expression on the righthand side of the definition of X_τ in (ii) can be viewed as an r_τ-tuple of m-tensors with the same basis matrices $\mathbf{U}_{\tau_i}$ but different core tensors $C_\tau(k,:) \in \mathbb{R}^{n_{\tau_1} \times \cdots \times n_{\tau_m}}$ for $k = 1, \ldots, r_\tau$. The vectorizations of these r_τ m-tensors, which are of dimension n_τ, form the columns of the matrix $\mathbf{U}_\tau$. The index 0 in $\times_0$ and $\mathbf{Mat}_0$ refers to the mode of dimension r_τ of $X_\tau \in \mathbb{R}^{r_\tau \times n_{\tau_1} \times \cdots \times n_{\tau_m}}$, which we count as mode 0. The product $\times_0 \mathbf{I}_\tau$ is redundant in the definition of X_τ, but we include it to emphasize the fact that X_τ is a tensor of order $m+1$. In the graphical representation, the edge 0 is directed upward to the parent vertex and the edges $1, \ldots, m$ are directed downward to the subtrees; see Figure 2.1.

This construction gives a data-sparse representation of a tensor with $\prod_{\ell\in\mathcal{L}} n_\ell$ entries. For a rough bound of the memory requirements, let d be the number of leaves and note that the number of vertices of the tree that are not leaves, is less than d. We set $n = \max_\ell n_\ell$ and $r = \max_\tau r_\tau$ and let $m+1$ be the maximal order of the connection tensors C_τ. The basis matrices and connection tensors then have less than

$$dnr + dr^{m+1} \ll n^d \quad \text{entries.}$$

We note that the representation of $X_{\bar\tau}$ in terms of the basis matrices $\mathbf{U}_\ell$ and connection tensors C_τ of full multilinear rank is not unique; cf. [25]. It is favorable to work with orthonormal matrices, so that each tensor X_τ for $\tau \le \bar\tau$ is in the Tucker format.

DEFINITION 2.3 (Orthonormal tree tensor network). *A tree tensor network $X_{\bar\tau}$ (more precisely, its representation in terms of the matrices $\mathbf{U}_\tau$) is called* orthonormal *if for each subtree $\tau < \bar\tau$, the matrix $\mathbf{U}_\tau$ has orthonormal columns.*

The following is a key lemma.

LEMMA 2.4. *For a tree $\tau = (\tau_1, \dots, \tau_m) \in \mathcal{T}$, let the matrices $\mathbf{U}_{\tau_1}, \dots, \mathbf{U}_{\tau_m}$ have orthonormal columns. Then, the matrix $\mathbf{U}_\tau$ has orthonormal columns if and only if the matricization $\mathbf{Mat}_0(C_\tau)^\top \in \mathbb{R}^{r_{\tau_1}\cdots r_{\tau_m}\times r_\tau}$ has orthonormal columns.*

Proof. We have, by the definition of $\mathbf{U}_\tau$ and X_τ and the unfolding formula (2.2),

$$\mathbf{U}_\tau^\top = \mathbf{Mat}_0(X_\tau) = \mathbf{I}_\tau\, \mathbf{Mat}_0(C_\tau) \bigotimes_{i=1}^m \mathbf{U}_{\tau_i}^\top = \mathbf{Mat}_0(C_\tau) \bigotimes_{i=1}^m \mathbf{U}_{\tau_i}^\top .$$

It follows that

$$\mathbf{U}_\tau^\top\, \mathbf{U}_\tau = \mathbf{Mat}_0(C_\tau)\Big(\bigotimes_{i=1}^m \mathbf{U}_{\tau_i}^\top\, \mathbf{U}_{\tau_i} \Big)\, \mathbf{Mat}_0(C_\tau)^\top = \big(\mathbf{Mat}_0(C_\tau)^\top\big)^\top\, \mathbf{Mat}_0(C_\tau)^\top,$$

which proves the result. ∎

We observe that due to the recursive definition of a tree tensor network it thus suffices to require that for each leaf the matrix $\mathbf{U}_\ell$ and for every other subtree $\tau < \bar\tau$ the matrix $\mathbf{Mat}_0(C_\tau)^\top$ have orthonormal columns. *Unless stated otherwise, we intend the tree tensor networks to be orthonormal in this paper.*

The orthonormality condition of Lemma 2.4 is very useful, because it reduces the orthonormality condition for the large, recursively constructed and computationally inaccessible matrix $\mathbf{U}_\tau \in \mathbb{R}^{n_{\tau_1}\cdots n_{\tau_m}\times r_\tau}$ to the orthonormality condition for the smaller, given matrix $\mathbf{Mat}_0(C_\tau)^\top \in \mathbb{R}^{r_{\tau_1}\cdots r_{\tau_m}\times r_\tau}$. To our knowledge, the first use of this important property was made in the chemical physics literature in the context of the multilayer MCTDH method [26].

A further consequence of Lemma 2.4 is that *every (non-orthonormal) tree tensor network has an orthonormal representation.* This is shown using a QR decomposition of non-orthonormal matrices $\mathbf{Mat}_0(C_{\tau_i})^\top = \mathbf{Q}_{\tau_i}\, \mathbf{R}_{\tau_i}$ and including the factor $\mathbf{R}_{\tau_i}$ in the tensor C_τ of the parent tree $\tau = (\tau_1, \dots, \tau_m)$: C_{τ_i} is changed to $\mathrm{Ten}_0(\mathbf{Q}_{\tau_i}^\top)$ and C_τ is changed to $C_\tau \bigtimes_{i=1}^m \mathbf{R}_{\tau_i}$. This is done recursively from the leaves to the root. We note that the property of C_τ to be of full multilinear rank $(r_\tau, r_{\tau_1}, \dots, r_{\tau_m})$ does not change under these transformations.

We rely on the following property, which can be proved as in [25], where binary trees are considered (this corresponds to the case $m = 2$ above); see also [6].

LEMMA 2.5. *Let a tree $\bar\tau \in \mathcal{T}$ be given together with dimensions $(n_\ell)_{\ell\in L(\bar\tau)}$ and ranks $(r_\tau)_{\tau\in T(\bar\tau)}$. The set $\mathcal{M}_{\bar\tau} = \mathcal{M}(\bar\tau, (n_\ell)_{\ell\in L(\bar\tau)}, (r_\tau)_{\tau\in T(\bar\tau)})$ of tensors with an*

orthonormal tree tensor network representation of the given dimensions and ranks is a smooth embedded manifold in the tensor space $\mathbb{R}^{\times_{\ell \in L(\bar{\tau})} n_\ell}$.

3. Extended Tucker Integrator. In Subsection 3.1 we recapitulate the algorithm of the Tucker tensor integrator of [18, 21], and in Subsection 3.2 we extend the algorithm to r-tuples of Tucker tensors with the same basis matrices. This will be a basic building block for the tree tensor network integrator derived in the next section.

3.1. Tucker tensor integrator. Let $\mathcal{M}_{\mathbf{r}}$ be the manifold of tensors of order d with fixed multi-linear rank $\mathbf{r}=(r_1, \ldots, r_d)$. Let us consider an approximation $Y^0 \in \mathcal{M}_{\mathbf{r}}$ to the initial data A^0,

$$Y^0 = C^0 \bigtimes_{i=1}^{d} \mathbf{U}_i^0 \in \mathbb{R}^{n_1 \times \cdots \times n_d},$$

where the basis matrices $\mathbf{U}_i^0 \in \mathbb{R}^{n_i \times r_i}$ have orthonormal columns and the core tensor $C^0 \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ has full multilinear rank $\mathbf{r}=(r_1, \ldots, r_d)$. The Tucker integrator is a numerical procedure that gives, after $(d+1)$ substeps, an approximation in Tucker tensor format, $Y^1 \in \mathcal{M}_{\mathbf{r}}$, to the full solution $A(t_1)$ after a time step $t_1 = t_0 + h$. The procedure is repeated over further time steps to yield approximations $Y^n \in \mathcal{M}_{\mathbf{r}}$ to $A(t_n)$. At each substep of the algorithm, only one factor of the Tucker representation is updated while the others, with the exception of the core tensor, remain fixed. The essential structure of the algorithm can be summarized as follows: Starting from $Y^0 = C^0 \bigtimes_{i=1}^{d} \mathbf{U}_i^0$ in factorized form, initialize $C_0^0 = C^0$.

For $i = 1, \ldots, d$, update $\mathbf{U}_i^0 \to \mathbf{U}_i^1$ and modify $C_{i-1}^0 \to C_i^0$.
Update $C_d^0 \to C^1$.

This yields the result after one time-step, $Y^1 = C^1 \bigtimes_{i=1}^{d} \mathbf{U}_i^1$ in factorized form.

To simplify the description, we introduce subflows $\Phi^{(i)}$ and Ψ, corresponding to the update of the basis matrices and of the core tensor, respectively. We set $r_{\neg i} = \prod_{j \neq i} r_i$ and use the notation $\mathbf{S}_i^{0,\top} = \left(\mathbf{S}_i^0\right)^\top$, $\mathbf{V}_i^{0,\top} = \left(\mathbf{V}_i^0\right)^\top$ etc.

Algorithm 1: Subflow $\Phi^{(i)}$

Data: $Y^0 = C^0 \bigtimes_{j=1}^{d} \mathbf{U}_j^0$ in factorized form, $F(t,Y), t_0, t_1$
Result: $Y^1 = C^1 \bigtimes_{j=1}^{d} \mathbf{U}_j^1$ in factorized form
begin
  set $\mathbf{U}_j^1 = \mathbf{U}_j^0 \quad \forall j \neq i$
  compute the QR decomposition $\mathbf{Mat}_i(C^0)^\top = \mathbf{Q}_i^0 \, \mathbf{S}_i^{0,\top} \in \mathbb{R}^{r_{\neg i} \times r_i}$
  set $\mathbf{K}_i^0 = \mathbf{U}_i^0 \, \mathbf{S}_i^0 \in \mathbb{R}^{n_i \times r_i}$
  solve the $n_i \times r_i$ matrix differential equation
    $\dot{\mathbf{K}}_i(t) = \mathbf{F}_i(t, \mathbf{K}_i(t))$ with initial value $\mathbf{K}_i(t_0) = \mathbf{K}_i^0$
    and return $\mathbf{K}_i^1 = \mathbf{K}_i(t_1)$; here
      $\mathbf{F}_i(t, \mathbf{K}_i) = \mathbf{Mat}_i(F(t, \mathrm{Ten}_i(\mathbf{K}_i(t)\mathbf{V}_i^{0,\top}))\mathbf{V}_i^0$ with
      $\mathbf{V}_i^{0,\top} = \mathbf{Mat}_i(\mathrm{Ten}_i(\mathbf{Q}_i^{0,\top}) \bigtimes_{j \neq i} \mathbf{U}_j^0)$
  compute the QR decomposition $\mathbf{K}_i^1 = \mathbf{U}_i^1 \, \widehat{\mathbf{S}}_i^1$
  solve the $r_i \times r_i$ matrix differential equation
    $\dot{\mathbf{S}}_i(t) = -\widehat{\mathbf{F}}_i(t, \mathbf{S}_i(t))$ with initial value $\mathbf{S}_i(t_0) = \widehat{\mathbf{S}}_i^1$
    and return $\widetilde{\mathbf{S}}_i^0 = \mathbf{S}_i(t_1)$; here
      $\widehat{\mathbf{F}}_i(t, \mathbf{S}_i) = \mathbf{U}_i^{1,\top} \, \mathbf{F}_i(t, \mathbf{U}_i^1 \, \mathbf{S}_i)$
  set $C^1 = \mathrm{Ten}_i(\widetilde{\mathbf{S}}_i^0 \, \mathbf{Q}_i^{0,\top})$
end

The following remarks will be used in the next subsection:

• Instead of the QR decomposition, any orthogonal decomposition (e.g., SVD) can be used that yields $\mathbf{Q}_i^0$ and $\mathbf{U}_i^1$ with orthonormal columns. This yields the same tensor Y^1, albeit in a different factorization.

• In the case where $F(t,Y)$ does not depend on either t or Y, we obtain the same result $\widetilde{\mathbf{S}}_i^0$ if we replace the differential equation for $\mathbf{S}$ with the negative sign from $t_0 \to t_1$, with initial value $\mathbf{S}_i(t_0) = \widehat{\mathbf{S}}_i^1$, by the differential equation with the positive sign for $\mathbf{S}$ backward in time from $t_1 \to t_0$ with final value $\mathbf{S}_i(t_1) = \widehat{\mathbf{S}}_i^1$:

solve $\dot{\mathbf{S}}_i(t) = \widehat{\mathbf{F}}_i(t, \mathbf{S}_i(t))$ with final value $\mathbf{S}_i(t_1) = \widehat{\mathbf{S}}_i^1$ and return $\widetilde{\mathbf{S}}_i^0 = \mathbf{S}_i(t_0)$.

For a general non-autonomous function $F(t,Y)$, the forward and backward formulations are no longer equivalent, but the robust convergence result of [21] holds equally true for both formulations.

We further remark that the differential equations for $\mathbf{K}_i(t)$ and $\mathbf{S}_i(t)$ need to be solved numerically by a standard integrator, unless $F(t,Y) = \dot{A}(t)$ is independent of Y.

The remaining subflow Ψ describes the final update process of the core.

Algorithm 2: Subflow Ψ

Data: $Y^0 = C^0 \bigtimes_{j=1}^d \mathbf{U}_j^0$ in factorized form, $F(t, Y), t_0, t_1$
Result: $Y^1 = C^1 \bigtimes_{j=1}^d \mathbf{U}_j^1$ in factorized form
begin
  set $\mathbf{U}_j^1 = \mathbf{U}_j^0 \quad \forall j = 1, \ldots, d.$
  solve the $r_1 \times \cdots \times r_d$ tensor differential equation
    $\dot{C}(t) = \widetilde{F}(t, C(t))$ with initial value $C(t_0) = C^0$
    and return $C^1 = C(t_1)$; here
      $\widetilde{F}(t, C) = F(t, C \bigtimes_{j=1}^d \mathbf{U}_j^1) \bigtimes_{j=1}^d \mathbf{U}_j^{1,\top}$
end

Finally, the result of the Tucker tensor integrator after one time step can be expressed in a compact way as

$$Y^1 = \Psi \circ \Phi^{(d)} \circ \cdots \circ \Phi^{(1)}(Y^0) \ . \tag{3.1}$$

We refer the reader to [18, 21] for a detailed derivation and major properties of this Tucker tensor integrator.

The efficiency of the implementation of this algorithm depends on the possibility to evaluate the functions $\mathbf{F}_i$ without explicitly forming the large slim matrix $\mathbf{V}_i^0 \in \mathbb{R}^{n_{\neg i} \times r_i}$ and the tensor $\mathrm{Ten}_i(\mathbf{K}_i(t)\mathbf{V}_i^{0,\top}) \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. This is the case if $F(t, C \bigtimes_{j=1}^d \mathbf{U}_j)$ is a linear combination of Tucker tensors of moderate rank whose factors can be computed directly from the factors C and $\mathbf{U}_j$ without actually computing the entries of the Tucker tensor.

3.2. Extended Tucker Integrator. We consider the case of a Tucker tensor of order $1 + d$ where the first basis matrix in the decomposition of the initial data is the identity matrix of dimension $r \times r$,

$$Y^0 = C^0 \times_0 \mathbf{I}_r \bigtimes_{i=1}^d \mathbf{U}_i^0 \in \mathbb{R}^{r \times n_1 \times \cdots \times n_d},$$

as appears in the recursive construction of orthonormal tree tensor networks. This can be viewed as a collection of r Tucker tensors in $\mathbb{R}^{n_1 \times \cdots \times n_d}$ with the same basis matrices $\mathbf{U}_i^0$. Recalling (3.1), the action of the Tucker integrator after one time step can be represented as

$$Y^1 = \Psi \circ \Phi^{(d)} \circ \cdots \circ \Phi^{(1)} \circ \Phi^{(0)}(Y^0) \ .$$

The following result simplifies the computation.

LEMMA 3.1. *With an appropriate choice of orthogonalization, the action of the subflow $\Phi^{(0)}$ on Y^0 becomes trivial , i.e.,*

$$\Phi^{(0)}(Y^0) = Y^0.$$

Proof. In the first step of the subflow $\Phi^{(0)}$, we matricize the core tensor C^0 in the zero mode and we perform a QR decomposition,

$$\mathbf{Mat}_0(C^0)^\top = \mathbf{Q}_0^0 \, \mathbf{S}_0^{0,\top}.$$

We define $\mathbf{V}_0^{0,\top} = \mathbf{Mat}_0(\mathrm{Ten}_0(\mathbf{Q}_0^{0,\top}) \bigtimes_{l=1}^{d} \mathbf{U}_l^{0,\top})$ and we set $\mathbf{K}_0^0 = \mathbf{I}_r \, \mathbf{S}_0^0 \in \mathbb{R}^{r\times r}$. The next step consists of solving the differential equation

$$\begin{aligned}\dot{\mathbf{K}}_0(t) &= \mathbf{Mat}_0(F(t, \mathrm{Ten}_0(\mathbf{K}_0(t)\mathbf{V}_0^{0,\top}))\mathbf{V}_0^0,\\ \mathbf{K}_0(t_0) &= \mathbf{K}_0^0\,.\end{aligned}$$

We define $\mathbf{K}_0^1 = \mathbf{K}_0(t_1) \in \mathbb{R}^{r\times r}$ and we use the trivial orthogonal decomposition (it is irrelevant that this is not the QR decomposition),

$$\mathbf{K}_0^1 = \mathbf{I}_r \, \mathbf{K}_0^1\,.$$

Finally, we solve the differential equation

$$\begin{aligned}\dot{\mathbf{S}}_0(t) &= \mathbf{I}_r^\top \, \mathbf{Mat}_0(F(t, \mathrm{Ten}_0(\mathbf{I}_r \, \mathbf{S}_0(t)\mathbf{V}_0^{0,\top}))\mathbf{V}_0^0,\\ \mathbf{S}_0(t_1) &= \mathbf{K}_0^1\,.\end{aligned}$$

This is the same differential equation as for $\mathbf{K}_0$, now solved backwards in time, and so we have

$$\widetilde{\mathbf{S}}_0^0 = \mathbf{S}_0(t_0) = \mathbf{K}_0(t_0) = \mathbf{S}_0^0.$$

Therefore, $C^1 = C^0$ and we conclude that $\Phi^{(0)}(Y^0) = Y^0$. □

We can now introduce the extended Tucker integrator as given by Algorithm 3 below. The adjective 'extended' refers to the fact that we now approximate the solution to a differential equation for r Tucker tensors with the same basis matrices rather than just one Tucker tensor. This extension is nontrivial but turns out to be remarkably simple.

Algorithm 3: Extended Tucker Integrator

Data: $Y^0 = C^0 \times_0 \mathbf{I}_r \bigtimes_{j=1}^{d} \mathbf{U}_j^0$ in factorized form, $F(t, Y), t_0, t_1$
Result: $Y^1 = C^1 \times_0 \mathbf{I}_r \bigtimes_{j=1}^{d} \mathbf{U}_j^1$ in factorized form
begin
  Set $Y^{[0]} = Y^0$
  for $i = 1 \dots d$ **do**
    Compute $Y^{[i]} = \Phi^{(i)}(Y^{[i-1]})$ in factorized form by Algorithm 1
  end
  Compute $Y^1 = \Psi(Y^{[d]})$ in factorized form by Algorithm 2
end

4. Recursive tree tensor network integrator. We now come to the central algorithmic section of this paper. We derive an integrator for orthonormal tree tensor networks, which in every time step updates the orthonormal-basis matrices $\mathbf{U}_\ell(t)$ of the leaves and the orthonormality-constrained connection tensors $C_\tau(t)$ of the other vertices of the tree. This is done without ever computing the entries of the high-dimensional tensor $Y(t)$ that has the tree tensor network representation given by the factors $\mathbf{U}_\ell(t)$ and $C_\tau(t)$ and (approximately) solves the projected differential equation (1.2) — provided that the function F can be evaluated at the tree tensor network $Y(t)$ using only its factors.

4.1. Derivation. Let $\bar{\tau} \in \mathcal{T}$ be a given tree with the set of leaves $L(\bar{\tau}) = \{1, \dots, d\}$ and $(r_\tau)_{\tau \in T(\bar{\tau})}$ a specified family of tree ranks, where we assume $r_{\bar{\tau}} = 1$. For each subtree $\tau = (\tau_1, \dots, \tau_m) \in T(\bar{\tau})$ we introduce the space

$$\mathcal{V}_\tau := \mathbb{R}^{r_\tau \times n_{\tau_1} \times \cdots \times n_{\tau_m}} . \tag{4.1}$$

In the following, we associate to each subtree τ of the given tree $\bar{\tau}$ a tensor-valued function $F_\tau : [0, t^*] \times \mathcal{V}_\tau \to \mathcal{V}_\tau$. Its actual recursive construction, starting from the root with $F_{\bar{\tau}} = F$ and passing to the leaves, will be given in the next subsection.

Consider a tree $\tau = (\tau_1, \dots, \tau_m)$ and an extended Tucker tensor Y_τ^0 associated to the tree τ,

$$Y_\tau^0 = C_\tau^0 \times_0 \mathbf{I}_\tau \bigtimes_{i=1}^m \mathbf{U}_{\tau_i}^0 .$$

Applying the extended Tucker integrator with the function F_τ we have that

$$Y_\tau^1 = \Psi_\tau \circ \Phi_\tau^{(m)} \circ \cdots \circ \Phi_\tau^{(1)}(Y_\tau^0) .$$

We recall that the subflow $\Phi_\tau^{(i)}$ gives the update process of the basis matrix $\mathbf{U}_{\tau_i}^0 \in \mathbb{R}^{n_{\tau_i} \times r_{\tau_i}}$. The extra subscript τ indicates that the subflow is computed for the function F_τ.

We have two cases:

(i) If τ_i is a leaf, we directly apply the subflow $\Phi_\tau^{(i)}$ and update the basis matrix.

(ii) Else, we apply $\Phi_\tau^{(i)}$ only approximately (but call the procedure still $\Phi_\tau^{(i)}$). We tensorize the basis matrix and we construct new initial data $Y_{\tau_i}^0$ and a function F_{τ_i}. We iterate the procedure in a recursive way, reducing the dimensionality of the problem at each recursion.

We are now in a position to formulate the recursive tree tensor network (TTN) integrator. It has the same general structure as the extended Tucker integrator as given by Algorithm 3.

The difference to the extended Tucker integrator is that now the subflow $\Phi_\tau^{(i)}$ is no longer the same, but it recursively uses the TTN integrator for the subtrees. This approximate subflow is defined in close analogy to the subflow $\Phi^{(i)}$ for Tucker tensors, but the first differential equation is solved only approximately unless τ_i is a leaf. We remark that in the following Algorithm 5 the tensors Y_{τ_i} correspond to a tensorization of the matrices $\mathbf{K}_i$ in Algorithm 1; see the next two subsections for details of this correspondence.

Algorithm 4: Recursive TTN Integrator

Data: tree $\tau = (\tau_1, \dots, \tau_m)$, TTN in factorized form
$Y_\tau^0 = C_\tau^0 \times_0 \mathbf{I}_\tau \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^0$ with $\mathbf{U}_{\tau_j}^0 = \mathbf{Mat}_0(X_{\tau_j}^0)^\top$,
function $F_\tau(t, Y_\tau), t_0, t_1$

Result: TTN $Y_\tau^1 = C_\tau^1 \times_0 \mathbf{I}_\tau \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^1$ with $\mathbf{U}_{\tau_j}^1 = \mathbf{Mat}_0(X_{\tau_j}^1)^\top$
in factorized form

begin
  set $Y_\tau^{[0]} = Y_\tau^0$
  for $i = 1 \dots m$ **do**
    compute $Y_\tau^{[i]} = \Phi_\tau^{(i)}(Y_\tau^{[i-1]})$ in factorized form by Algorithm 5
  end
  compute $Y_\tau^1 = \Psi_\tau(Y_\tau^{[m]})$ in factorized form by Algorithm 6
end

Algorithm 5: Subflow $\Phi_\tau^{(i)}$

Data: tree $\tau = (\tau_1, \ldots, \tau_m)$, TTN in factorized form
$Y_\tau^0 = C_\tau^0 \times_0 \mathbf{I}_\tau \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^0$ with $\mathbf{U}_{\tau_j}^0 = \mathbf{Mat}_0(X_{\tau_j}^0)^\top$,
function $F_\tau(t, Y_\tau), t_0, t_1$

Result: TTN $Y_\tau^1 = C_\tau^1 \times_0 \mathbf{I}_r \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^1$ with $\mathbf{U}_{\tau_j}^1 = \mathbf{Mat}_0(X_{\tau_j}^1)^\top$
in factorized form

begin

set $\mathbf{U}_{\tau_j}^1 = \mathbf{U}_{\tau_j}^0 \quad \forall j \neq i$

compute the QR factorization $\mathbf{Mat}_i(C_\tau^0)^\top = \mathbf{Q}_{\tau_i}^0 \mathbf{S}_{\tau_i}^{0,\top}$

set $Y_{\tau_i}^0 = X_{\tau_i}^0 \times_0 \mathbf{S}_{\tau_i}^{0,\top}$

if $\tau_i = \ell$ is a leaf, **then** solve the $n_\ell \times r_\ell$ matrix differential equation
$\dot{Y}_{\tau_i}(t) = F_{\tau_i}(t, Y_{\tau_i}(t))$ with initial value $Y_{\tau_i}(t_0) = Y_{\tau_i}^0$
and return $Y_{\tau_i}^1 = Y_{\tau_i}(t_1)$

else
compute $Y_{\tau_i}^1 =$ *Recursive TTN Integrator* $(\tau_i, Y_{\tau_i}^0, F_{\tau_i}, t_0, t_1)$

compute the QR decomposition $\mathbf{Mat}_0(C_{\tau_i}^1)^\top = \widehat{\mathbf{Q}}_{\tau_i}^1 \widehat{\mathbf{S}}_{\tau_i}^1$, where
$C_{\tau_i}^1$ is the connecting tensor of $Y_{\tau_i}^1$

set $\mathbf{U}_{\tau_i}^1 = \mathbf{Mat}_0(X_{\tau_i}^1)^\top$, where the TTN $X_{\tau_i}^1$ is obtained from $Y_{\tau_i}^1$ by
replacing the connecting tensor with $\widehat{C}_{\tau_i}^1 = \mathrm{Ten}_0(\widehat{\mathbf{Q}}_{\tau_i}^{1,T})$

solve the $r_{\tau_i} \times r_{\tau_i}$ matrix differential equation
$\dot{\mathbf{S}}_{\tau_i}(t) = -\widehat{\mathbf{F}}_{\tau_i}(t, \mathbf{S}_{\tau_i}(t))$ with initial value $\mathbf{S}_{\tau_i}(t_0) = \widehat{\mathbf{S}}_{\tau_i}^1$
and return $\widetilde{\mathbf{S}}_{\tau_i}^0 = \mathbf{S}_{\tau_i}(t_1)$; here
$\widehat{\mathbf{F}}_{\tau_i}(t, \mathbf{S}_{\tau_i}) = \mathbf{U}_{\tau_i}^{1,\top} \mathbf{Mat}_0\big(F_{\tau_i}(t, X_{\tau_i}^1 \times_0 \mathbf{S}_{\tau_i}^\top)\big)^\top$

set $C_\tau^1 = \mathrm{Ten}_i(\widetilde{\mathbf{S}}_{\tau_i}^0 \mathbf{Q}_{\tau_i}^{0,\top})$

end

The subflow Ψ_τ is the same as for the Tucker integrator, for the function F_τ instead of F.

Algorithm 6: Subflow Ψ_τ

Data: tree $\tau = (\tau_1, \ldots, \tau_m)$, TTN in factorized form
$Y_\tau^0 = C_\tau^0 \times_0 \mathbf{I}_\tau \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^0$ with $\mathbf{U}_{\tau_j}^0 = \mathbf{Mat}_0(X_{\tau_j}^0)^\top$,
function $F_\tau(t, Y_\tau), t_0, t_1$

Result: TTN $Y_\tau^1 = C_\tau^1 \times_0 \mathbf{I}_\tau \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^1$ with $\mathbf{U}_{\tau_j}^1 = \mathbf{Mat}_0(X_{\tau_j}^1)^\top$
in factorized form

begin

set $\mathbf{U}_{\tau_j}^1 = \mathbf{U}_{\tau_j}^0 \quad \forall j = 1, \ldots, m.$

solve the $r_\tau \times r_{\tau_1} \times \cdots \times r_{\tau_m}$ tensor differential equation
$\dot{C}_\tau(t) = \widetilde{F}_\tau(t, C_\tau(t))$ with initial value $C_\tau(t_0) = C_\tau^0$
and return $C_\tau^1 = C_\tau(t_1)$; here
$\widetilde{F}_\tau(t, C_\tau) = F_\tau(t, C_\tau \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^1) \bigtimes_{j=1}^m \mathbf{U}_{\tau_j}^{1,\top}$

end

The differential equations for $Y_{\tau_i}(t)$, $\mathbf{S}_{\tau_i}(t)$ in Algorithm 5 and $C_\tau(t)$ in Algorithm 6 need to be solved approximately by a standard numerical integrator, unless

$F(t,Y)$ is independent of Y (which then implies that also the functions $F_\tau(t,Y)$ are independent of Y).

The efficiency of the implementation of this algorithm depends on the possibility to evaluate the functions F_τ, $\widehat{\mathbf{F}}_\tau$ and $\widetilde{F}_\tau$ efficiently for all subtrees τ of $\bar\tau$, without explicitly forming large matrices or tensors whose dimension exceeds by far that of the basis matrices and connecting tensors. This is the case if F maps TTNs into linear combinations of TTNs of moderate tree rank whose factors can be computed directly from the basis matrices and connecting tensors without actually computing the entries of the TTN.

4.2. Constructing F_τ and Y^0_τ via restrictions/prolongations. In a recursion that passes from the root to the leaves of $\bar\tau$, we construct for each subtree τ of $\bar\tau$ the tensor-valued function F_τ that is used in the recursive TTN integrator. We note that $\mathcal{V}_{\bar\tau}$ is isomorphic to $\mathbb{R}^{n_1\times\cdots\times n_d}$ (since $r_{\bar\tau}=1$) and we start the construction by setting $F_{\bar\tau}=F:[0,t^*]\times\mathcal{V}_{\bar\tau}\to\mathcal{V}_{\bar\tau}$. Given a subtree $\tau=(\tau_1,\dots,\tau_m)\in\mathcal{T}$, we now assume by induction that

$$F_\tau:[0,t^*]\times\mathcal{V}_\tau\to\mathcal{V}_\tau$$

is already defined. For each $i=1,\dots,m$, we need to determine the tensor-valued function F_{τ_i} that appears in the subflow $\Phi^{(i)}_\tau$ of the recursive TTN integrator. For the initial data $Y^0_\tau=C^0_\tau\times_0\mathbf{I}_\tau\bigtimes_{i=1}^m\mathbf{U}^0_{\tau_i}$ the subflow $\Phi^{(i)}_\tau$ given by Algorithm 5 first computes the QR decomposition

$$\mathbf{Mat}_i(C^0_\tau)^\top=\mathbf{Q}^0_{\tau_i}\,\mathbf{S}^{0,\top}_{\tau_i},$$

where $\mathbf{Q}^0_{\tau_i}\in\mathbb{R}^{r_\tau r_{\neg\tau_i}\times r_{\tau_i}}$ with $r_{\neg\tau_i}=\prod_{j\neq i}r_{\tau_j}$ has orthonormal columns and $\mathbf{S}^0_{\tau_i}\in\mathbb{R}^{r_{\tau_i}\times r_{\tau_i}}$. Since

$$\begin{aligned}Y^0_\tau=C^0_\tau\times_0\mathbf{I}_\tau\bigtimes_{i=1}^m\mathbf{U}^0_{\tau_i}&=\mathrm{Ten}_i(\mathbf{S}^0_{\tau_i}\,\mathbf{Q}^{0,\top}_{\tau_i})\times_0\mathbf{I}_\tau\bigtimes_{i=1}^m\mathbf{U}^0_{\tau_i}\\&=\mathrm{Ten}_i(\mathbf{Q}^{0,\top}_{\tau_i})\times_0\mathbf{I}_\tau\bigtimes_{j\neq i}\mathbf{U}^0_{\tau_j}\times_i(\mathbf{U}^0_{\tau_i}\,\mathbf{S}^0_{\tau_i}),\end{aligned}$$

we then have the SVD-like decomposition

$$\mathbf{Mat}_i(Y^0_\tau)=\mathbf{U}^0_{\tau_i}\,\mathbf{S}^0_{\tau_i}\,\mathbf{V}^{0,\top}_{\tau_i}\tag{4.2}$$

with the (computationally inaccessible) matrix

$$\mathbf{V}^0_{\tau_i}=\mathbf{Mat}_i(\mathrm{Ten}_i(\mathbf{Q}^{0,\top}_{\tau_i})\times_0\mathbf{I}_\tau\bigtimes_{j\neq i}\mathbf{U}^0_{\tau_j})^\top\in\mathbb{R}^{r_\tau n_{\neg\tau_i}\times r_{\tau_i}}$$

for $n_{\neg\tau_i}=\prod_{j\neq i}n_{\tau_j}=n_\tau/n_{\tau_i}$. We note that both $\mathbf{U}^0_{\tau_i}$ and $\mathbf{V}^0_{\tau_i}$ have orthonormal columns.

Like in Algorithm 1 for the subflow $\Phi^{(i)}$ of the Tucker integrator, we consider the differential equation for $\mathbf{K}_{\tau_i}(t)\in\mathbb{R}^{n_{\tau_i}\times r_{\tau_i}}$,

$$\begin{aligned}\dot{\mathbf{K}}_{\tau_i}(t)&=\mathbf{F}_{\tau_i}(t,\mathbf{K}_{\tau_i}(t)),\\\mathbf{K}_{\tau_i}(t_0)&=\mathbf{U}^0_{\tau_i}\,\mathbf{S}^0_{\tau_i}\ ,\end{aligned}\tag{4.3}$$

where

$$\mathbf{F}_{\tau_i}(\mathbf{K}_{\tau_i})=\mathbf{Mat}_i\big(F_\tau(t,\mathrm{Ten}_i(\mathbf{K}_{\tau_i}\,\mathbf{V}^{0,\top}_{\tau_i}))\big)\mathbf{V}^0_{\tau_i}.$$

Algorithm 5 uses this differential equation in tensorized form and solves it approximately by recurrence down to the leaves. This relation is seen as follows: By definition of the tree tensor network, there exists $X^0_{\tau_i} \in \mathcal{V}_{\tau_i}$ such that

$$\mathbf{U}^0_{\tau_i} = \mathbf{Mat}_0(X^0_{\tau_i})^\top, \qquad \text{i.e.,} \quad X^0_{\tau_i} = \mathrm{Ten}_0(\mathbf{U}^{0,\top}_{\tau_i}).$$

This implies that the initial condition in (4.3) can be rewritten as

$$\mathbf{K}_{\tau_i}(t_0) = \mathbf{U}^0_{\tau_i}\,\mathbf{S}^0_{\tau_i} = \mathbf{Mat}_0(X^0_{\tau_i})^\top \mathbf{S}^0_{\tau_i} = \mathbf{Mat}_0(X^0_{\tau_i} \times_0 \mathbf{S}^{0,\top}_{\tau_i})^\top,$$

which is the initial value chosen in Algorithm 5. We introduce

$$Y_{\tau_i}(t) = \mathrm{Ten}_0(\mathbf{K}_{\tau_i}(t)^\top), \qquad \text{i.e.,} \quad \mathbf{K}_{\tau_i}(t) = \mathbf{Mat}_0(Y_{\tau_i}(t))^\top. \tag{4.4}$$

By substitution, (4.3) can be rewritten as the differential equation that appears in Algorithm 5,

$$\begin{aligned} \dot{Y}_{\tau_i}(t) &= F_{\tau_i}(t, Y_{\tau_i}(t)) \\ Y_{\tau_i}(t_0) &= Y^0_{\tau_i} := X^0_{\tau_i} \times_0 \mathbf{S}^{0,\top}_{\tau_i}\ , \end{aligned}$$

where now

$$\begin{aligned} F_{\tau_i}(t, Y_{\tau_i}) &= \mathrm{Ten}_0\big(\mathbf{F}_{\tau_i}(t, \mathbf{Mat}_0(Y_{\tau_i})^\top\big) \\ &= \mathrm{Ten}_0\Big(\big(\mathbf{Mat}_i(\ F_\tau(t, \mathrm{Ten}_i(\mathbf{Mat}_0(Y_{\tau_i})^\top\, \mathbf{V}^{0,\top}_{\tau_i})))\mathbf{V}^0_{\tau_i}\big)^\top\Big)\ . \end{aligned} \tag{4.5}$$

The construction of the tensor-valued function F_{τ_i} becomes more transparent by introducing the *prolongation*

$$\pi_{\tau,i}(Y_{\tau_i}) := \mathrm{Ten}_i\big((\mathbf{V}^0_{\tau_i}\,\mathbf{Mat}_0(Y_{\tau_i}))^\top\big) \in \mathcal{V}_\tau \quad \text{ for } \ Y_{\tau_i} \in \mathcal{V}_{\tau_i} \tag{4.6}$$

and the *restriction*

$$\pi^\dagger_{\tau,i}(Z_\tau) := \mathrm{Ten}_0\big((\mathbf{Mat}_i(Z_\tau)\,\mathbf{V}^0_{\tau_i})^\top\big) \in \mathcal{V}_{\tau_i}, \quad \text{ for } \ Z_\tau \in \mathcal{V}_\tau, \tag{4.7}$$

where the tensorization Ten_0 is for a matrix in $\mathbb{R}^{r_{\tau_i} \times n_{\tau_i}}$ according to the dimensions of the subtrees of τ_i.

We note the following properties.

LEMMA 4.1. *Let $\tau = (\tau_1, \ldots, \tau_m)$ and $i = 1, \ldots, m$. The restriction $\pi^\dagger_{\tau,i} : \mathcal{V}_\tau \to \mathcal{V}_{\tau_i}$ is both a left inverse and the adjoint (with respect to the tensor Euclidean inner product) of the prolongation $\pi_{\tau,i} : \mathcal{V}_{\tau_i} \to \mathcal{V}_\tau$, that is,*

$$\pi^\dagger_{\tau,i}(\pi_{\tau,i}(Y_{\tau_i})) = Y_{\tau_i} \qquad \textit{for all } \ Y_{\tau_i} \in \mathcal{V}_{\tau_i} \tag{4.8}$$

$$\langle \pi_{\tau,i}(Y_{\tau_i}), Z_\tau\rangle_{\mathcal{V}_\tau} = \langle Y_{\tau_i}, \pi^\dagger_{\tau,i}(Z_\tau)\rangle_{\mathcal{V}_{\tau_i}} \quad \textit{for all } \ Y_{\tau_i} \in \mathcal{V}_{\tau_i},\ Z_\tau \in \mathcal{V}_\tau. \tag{4.9}$$

Moreover, $\|\pi_{\tau,i}(Y_{\tau_i})\|_{\mathcal{V}_\tau} = \|Y_{\tau_i}\|_{\mathcal{V}_{\tau_i}}$ and $\|\pi^\dagger_{\tau,i}(Z_\tau)\|_{\mathcal{V}_{\tau_i}} \le \|Z_\tau\|_{\mathcal{V}_\tau}$, where the norms are the tensor Euclidean norms.

Proof. Since $\mathbf{V}^{0,\top}_{\tau_i}\,\mathbf{V}^0_{\tau_i} = \mathbf{I}$, we obtain (4.8). Using that the tensorization Ten_i is the adjoint of the matricization $\mathbf{Mat}_i$ for the Frobenius inner product and that taking transposes in both matrices of a Frobenius inner product does not change the inner product, we arrive at (4.9). The norm equality follows from the definition (4.6) and the fact that the matrix $\mathbf{V}^0_{\tau_i}$ has orthonormal columns. The norm bound follows

from (4.7) on noting the general matrix norm inequality $\|\mathbf{AB}\|_F \leq \|\mathbf{A}\|_2 \, \|\mathbf{B}\|_F$ and the fact that $\| \mathbf{V}_{\tau_i}^{0,\top} \|_2 = 1$. ☐

We emphasize that the mappings $\pi_{\tau,i}$ and $\pi_{\tau,i}^\dagger$ depend on the initial data Y_τ^0. We observe that we can write (4.5) more compactly as $F_{\tau_i} = \pi_{\tau,i}^\dagger \circ F_\tau \circ \pi_{\tau,i}$. For the initial data we find from (4.2) and (4.3) that

$$Y_{\tau_i}^0 = \mathrm{Ten}_0(\mathbf{K}_{\tau_i}(t_0)^\top) = \mathrm{Ten}_0\big((\mathbf{U}_{\tau_i}^0 \, \mathbf{S}_{\tau_i}^0)^\top\big) = \mathrm{Ten}_0\big((\mathbf{Mat}_i(Y_\tau^0) \, \mathbf{V}_{\tau_i}^0)^\top\big) = \pi_{\tau,i}^\dagger(Y_\tau^0).$$

We thus arrive at the following.

Definition 4.2. *For the given tensor-valued function $F_{\bar\tau} = F : [t_0, t^*] \times \mathcal{V}_{\bar\tau} \to \mathcal{V}_{\bar\tau}$ and a tree tensor network $Y_{\bar\tau}^0 \in \mathcal{M}_{\bar\tau}$, we define recursively for each tree $\tau = (\tau_1, \ldots, \tau_m) \leq \bar\tau$ and for $i = 1, \ldots, m$*

$$F_{\tau_i} = \pi_{\tau,i}^\dagger \circ F_\tau \circ \pi_{\tau,i}$$
$$Y_{\tau_i}^0 = \pi_{\tau,i}^\dagger(Y_\tau^0).$$

These are the nonlinear operators and initial data that are used in the recursive TTN integrator. We remark that their construction has a formal similarity to that of operators and functions in multilevel methods; cf. [8].

An important observation is the following.

Lemma 4.3. *If the initial tree tensor network $Y_{\bar\tau}^0$ has full tree rank $(r_\sigma)_{\sigma \leq \bar\tau}$, then Y_τ^0 has full tree rank $(r_\sigma)_{\sigma \leq \tau}$ for every subtree $\tau \leq \bar\tau$.*

Proof. Let $\tau = (\tau_1, \ldots, \tau_m) \in T(\bar\tau)$ and $i = 1, \ldots, m$. From the above derivation we have, with the tensor network $X_{\tau_i} = \mathrm{Ten}_0(\mathbf{U}_{\tau_i}^{0,\top})$ of full tree rank,

$$Y_{\tau_i}^0 = \mathrm{Ten}_0((\mathbf{U}_{\tau_i}^0 \, \mathbf{S}_{\tau_i}^0)^\top) = X_{\tau_i}^0 \times_0 \mathbf{S}_{\tau_i}^{0,\top}.$$

If Y_τ^0 has full tree rank, then $\mathbf{S}_{\tau_i}^0$ is invertible, and hence also $Y_{\tau_i}^0$ has full tree rank. By induction we find that for every subtree $\tau \leq \bar\tau$, the restricted initial tensor Y_τ^0 has full tree rank. ☐

In terms of the manifold (see Lemma 2.5)

$$\mathcal{M}_\tau = \mathcal{M}(\tau, (n_\ell)_{\ell \in L(\tau)}, (r_\sigma)_{\sigma \leq \tau}) \tag{4.10}$$

of tree tensor networks for the tree $\tau \in \mathcal{T}$ of given dimensions $(n_\ell)_{\ell \in L(\tau)}$ and full tree rank $(r_\sigma)_{\sigma \leq \tau}$, Lemma 4.3 can be restated as saying that for $\tau = (\tau_1, \ldots, \tau_m)$ and $Y_\tau^0 \in \mathcal{M}_\tau$, the restriction $\pi_{\tau,i}^\dagger(Y_\tau^0)$ is in $\mathcal{M}_{\tau_i}$. This statement is not true for an arbitrary $Y_\tau \in \mathcal{M}_\tau$ that is different from Y_τ^0 (recall that the chosen restriction operator $\pi_{\tau,i}^\dagger$ depends on Y_τ^0). In particular, a loss of rank occurs if for some j, the basis matrices are such that $\mathbf{U}_{\tau_j}^{0,\top} \, \mathbf{U}_{\tau_j}$ is a singular $r_{\tau_j} \times r_{\tau_j}$ matrix. However, for the prolongation we have the following.

Lemma 4.4. *Let $\tau = (\tau_1, \ldots, \tau_m) \in \mathcal{T}$ and $i = 1, \ldots, m$. If $Y_{\tau_i} \in \mathcal{M}_{\tau_i}$, then the prolongation $\pi_{\tau,i}(Y_{\tau_i})$ is in $\mathcal{M}_\tau$.*

Proof. We have, using the definition of $\mathbf{V}_{\tau_i}^0$ and writing $\mathbf{U}_{\tau_i} := \mathbf{Mat}_0(Y_{\tau_i})^\top$,

$$\begin{aligned}
\pi_{\tau,i}(Y_{\tau_i}) &= \mathrm{Ten}_i\big((\mathbf{V}_{\tau_i}^0 \, \mathbf{Mat}_0(Y_{\tau_i}))^\top\big) \\
&= \mathrm{Ten}_i\big((\mathbf{Mat}_i(\mathrm{Ten}_i(\mathbf{Q}_{\tau_i}^{0,\top}) \times_0 \mathbf{I}_\tau \bigtimes_{j \neq i} \mathbf{U}_{\tau_j}^0)^\top \, \mathbf{Mat}_0(Y_{\tau_i}))^\top\big) \\
&= \mathrm{Ten}_i\big(\mathbf{U}_{\tau_i} \, \mathbf{Mat}_i(\mathrm{Ten}_i(\mathbf{Q}_{\tau_i}^{0,\top}) \times_0 \mathbf{I}_\tau \bigtimes_{j \neq i} \mathbf{U}_{\tau_j}^0)\big) \\
&= \mathrm{Ten}_i(\mathbf{Q}_{\tau_i}^{0,\top}) \times_0 \mathbf{I}_\tau \bigtimes_{j \neq i} \mathbf{U}_{\tau_j}^0 \times_i \mathbf{U}_{\tau_i},
\end{aligned}$$

which is of full tree rank. ☐

4.3. QR decomposition. We now explain how the first QR decomposition in Algorithm 1 is related to that of Algorithm 5. We consider a tree $\tau = (\tau_1, \dots, \tau_m) \in \mathcal{T}$ and let τ take the role of τ_i in Algorithm 5 for ease of notation. In the extension of Algorithm 1 via (4.3), we would need the QR-decomposition of $\mathbf{K}_\tau^1 \in \mathbb{R}^{n_\tau \times r_\tau}$, where we recall that $n_\tau = \prod_{j=1}^m n_{\tau_j}$ can get prohibitively large. This difficulty is overcome using that the tree tensor network is orthonormal: the QR-decomposition of the full matrix $\mathbf{K}_\tau^1$ is equivalent to the QR-decomposition of the matricization of a small core tensor. In fact, by construction we have that

$$\mathbf{K}_\tau^1 = \mathbf{Mat}_0(Y_\tau^1)^\top,$$

where

$$Y_\tau^1 = C_\tau^1 \times_0 \mathbf{I}_\tau \bigtimes_{i=1}^m \mathbf{U}_{\tau_i}^1 \ .$$

This implies that

$$\mathbf{K}_\tau^1 = \Big(\bigotimes_{i=1}^m \mathbf{U}_{\tau_i}^1 \Big) \mathbf{Mat}_0(C_\tau^1)^\top.$$

Since the Kronecker product of orthonormal matrices is orthonormal, it suffices to do a QR-decomposition of the comparatively small matrix $\mathbf{Mat}_0(C_\tau^1)^\top \in \mathbb{R}^{r_{\tau_1} \cdots r_{\tau_m} \times r_\tau}$, as is done in Algorithm 5.

4.4. Efficient computation of the prolongation $\pi_{\tau,i}$. The construction of the extremely large matrix $\mathbf{V}_{\tau_i}^{0,\top}$ appearing in the integrator must be avoided. In fact, recalling that

$$\mathbf{V}_{\tau_i}^{0,\top} = \mathbf{Mat}_i(\mathrm{Ten}_i(\mathbf{Q}_{\tau_i}^{0,\top}) \bigtimes_{j \neq i} \mathbf{U}_{\tau_j}^0) \in \mathbb{R}^{r_{\tau_i} \times r_\tau n_{\neg \tau_i}},$$

the mapping $\pi_{\tau,i}$ can be easily computed,

$$\begin{aligned}
\pi_{\tau,i}(Y) &= \mathrm{Ten}_i(\mathbf{Mat}_0(Y)^\top \mathbf{V}_{\tau_i}^{0,\top}) \\
&= \mathrm{Ten}_i(\mathbf{Mat}_0(Y)^\top \mathbf{Mat}_i(\mathrm{Ten}_i(\mathbf{Q}_{\tau_i}^{0,\top}) \bigtimes_{j \neq i} \mathbf{U}_{\tau_j}^0)) \\
&= \mathrm{Ten}_i\big(\mathbf{Mat}_i(\mathrm{Ten}_i(\mathbf{Q}_{\tau_i}^{0,\top}) \times_i \mathbf{Mat}_0(Y)^\top \bigtimes_{j \neq i} \mathbf{U}_{\tau_j}^0)\big) \\
&= \mathrm{Ten}_i(\mathbf{Q}_{\tau_i}^{0,\top}) \times_i \mathbf{Mat}_0(Y)^\top \bigtimes_{j \neq i} \mathbf{U}_{\tau_j}^0 \ .
\end{aligned}$$

The action of the prolongation $\pi_{\tau,i}$ on the tree tensor $Y \in \mathcal{V}_{\tau_i}$ yields a new larger tree tensor in $\mathcal{V}_\tau$ with the core tensor $\mathrm{Ten}_i(\mathbf{Q}_{\tau_i}^{0,\top})$.

4.5. Efficient computation of the restriction $\pi_{\tau,i}^\dagger$. The computation of the mapping $\pi_{\tau_i}^\dagger(Z_\tau)$ can be done efficiently by contraction if Z_τ is itself a tree tensor network on the tree $\tau = (\tau_1, \dots, \tau_m)$ with the same dimensions n_τ and n_{τ_i} but possibly different ranks s_τ and s_{τ_i} instead of r_τ and r_{τ_i}, respectively:

$$Z_\tau = G_\tau \times_0 \mathbf{I}_\tau \bigtimes_{i=1}^m \mathbf{W}_{\tau_i} \in \mathcal{V}_\tau$$

with the core tensor $G_\tau \in \mathbb{R}^{s_\tau \times s_{\tau_1} \times \cdots \times s_{\tau_m}}$ (not necessarily of full multilinear rank) and matrices $\mathbf{W}_{\tau_i} \in \mathbb{R}^{n_{\tau_i} \times s_{\tau_i}}$ (not necessarily of full rank). We have that

$$\pi_{\tau_i}^\dagger(Z_\tau) = \mathrm{Ten}_0(\mathbf{V}_{\tau_i}^{0,\top} \mathbf{Mat}_i(Z_\tau)^\top)$$

$$
\begin{aligned}
&= \mathrm{Ten}_0(\mathbf{Q}_{\tau_i}^{0,\top} \big(\mathbf{I}_\tau \bigotimes_{j\neq i} \mathbf{U}_{\tau_j}^{0,\top} \big) \mathbf{Mat}_i(Z_\tau)^\top) \\
&= \mathrm{Ten}_0(\mathbf{Q}_{\tau_i}^{0,\top} \mathbf{Mat}_i(G_\tau \times_0 \mathbf{I}_\tau \bigtimes_{j\neq i} (\mathbf{U}_{\tau_j}^{0,\top} \mathbf{W}_{\tau_j}) \times_i \mathbf{W}_{\tau_i})^\top) \\
&= \mathrm{Ten}_0(\mathbf{Q}_{\tau_i}^{0,\top} \mathbf{Mat}_i(G_\tau \times_0 \mathbf{I}_\tau \bigtimes_{j\neq i} (\mathbf{U}_{\tau_j}^{0,\top} \mathbf{W}_{\tau_j}))^\top \mathbf{W}_{\tau_i}^\top)
\end{aligned}
$$

We define the small matrix

$$
\mathbf{R}_{\tau_i} := \mathbf{Q}_{\tau_i}^{0,\top} \mathbf{Mat}_i(G_\tau \times_0 \mathbf{I}_\tau \bigtimes_{j\neq i} (\mathbf{U}_{\tau_j}^{0,\top} \mathbf{W}_{\tau_j}))^\top \in \mathbb{R}^{r_{\tau_i} \times s_{\tau_i}},
$$

where we note that the product of the large matrices $\mathbf{U}_{\tau_j}^0$ and $\mathbf{W}_{\tau_j}$ (with n_{τ_j} rows, which is prohibitive unless τ_j is a leaf) can be computed recursively from small matrices: For a tree $\tau = (\tau_1, \dots, \tau_m)$, let

$$
\begin{aligned}
\mathbf{U}_\tau &= \mathbf{Mat}_0(C_\tau \times_0 \mathbf{I}_\tau \bigtimes_{i=1}^m \mathbf{U}_{\tau_i})^\top \in \mathbb{R}^{n_\tau \times r_\tau}, \\
\mathbf{W}_\tau &= \mathbf{Mat}_0(G_\tau \times_0 \mathbf{I}_\tau \bigtimes_{i=1}^m \mathbf{W}_{\tau_i})^\top \in \mathbb{R}^{n_\tau \times s_\tau}.
\end{aligned}
$$

Then, (2.2) shows that the small matrix $\mathbf{U}_\tau^\top \mathbf{W}_\tau \in \mathbb{R}^{r_\tau \times s_\tau}$ equals

$$
\mathbf{U}_\tau^\top \mathbf{W}_\tau = \mathbf{Mat}_0\big(C_\tau \bigtimes_{i=1}^m (\mathbf{U}_{\tau_i}^\top \mathbf{W}_{\tau_i})^\top\big) \, \mathbf{Mat}_0(G_\tau)^\top, \tag{4.11}
$$

and hence this matrix can be computed recursively, passing from the leaves to the root τ. In this way we compute only products $\mathbf{U}_\ell^\top \mathbf{W}_\ell$ for the leaves $\ell \in L(\tau)$ and products of matrices whose dimensions depend only on the ranks and orders of the connection tensors.

We return to the above expression for $\pi_{\tau_i}^\dagger(Z_\tau)$ and recall that by definition of the tree tensor network, there exists a tree tensor network Z_{τ_i} on the tree τ_i such that

$$
\mathbf{W}_{\tau_i} = \mathbf{Mat}_0(Z_{\tau_i})^\top.
$$

We then have

$$
\begin{aligned}
\pi_{\tau_i}^\dagger(Z_\tau) &= \mathrm{Ten}_0(\mathbf{R}_{\tau_i} \mathbf{W}_{\tau_i}^\top) = \mathrm{Ten}_0(\mathbf{R}_{\tau_i} \mathbf{Mat}_0(Z_{\tau_i})) = \mathrm{Ten}_0(\mathbf{Mat}_0(Z_{\tau_i} \times_0 \mathbf{R}_{\tau_i})) \\
&= Z_{\tau_i} \times_0 \mathbf{R}_{\tau_i} \, .
\end{aligned}
$$

This implies that $\pi_{\tau_i}^\dagger(Z_\tau)$ differs from Z_{τ_i} only in that the (small) core tensor G_{τ_i} of Z_{τ_i} is replaced by $\widehat{G}_{\tau_i} = G_{\tau_i} \times_0 \mathbf{R}_{\tau_i}$, or equivalently, $\mathbf{Mat}_0(\widehat{G}_{\tau_i}) = \mathbf{R}_{\tau_i} \mathbf{Mat}_0(G_{\tau_i})$.

4.6. Computational complexity. In the implementation of Algorithm 5 and Algorithm 6, the matrices $\mathbf{U}_{\tau_i} = \mathbf{Mat}_0(X_{\tau_i})^T$ (with superscripts 0 and 1), the tensors Y_{τ_i} and the values of the function F_{τ_i} are not stored and computed entrywise but only through their TTN factorization into basis matrices for the leaves and connection tensors. Furthermore, products of prohibitively large matrices such as $\mathbf{U}_{\tau_i}^{1,\top} \mathbf{Mat}_0\big(F_{\tau_i}(t, X_{\tau_i}^1 \times_0 \mathbf{S}_{\tau_i}^\top)\big)^\top$ are reduced to matrix products of small matrices using (4.11) recursively. A count of the required operations and the required memory yields the following result.

LEMMA 4.5. *Let d be the order of the tensor $A(t)$ (i.e., the number of leaves of the tree $\bar\tau$), $l < d$ the number of levels (i.e., the height of the tree $\bar\tau$), and let $n = \max_\ell n_\ell$ be the maximal dimension, $r = \max_\tau r_\tau$ the maximal rank and m the maximal order of the connection tensors. We assume that for every tree tensor network $X_{\bar\tau}$ we have that $Z_{\bar\tau} = F(t, X_{\bar\tau})$ is again a tree tensor network, with ranks $s_\tau \leq cr$ for all subtrees $\tau \leq \bar\tau$ with a moderate constant c. Then, one time step of the tree tensor integrator given by Algorithms 4–6 can be implemented such that it requires*

- $O(dr(n + r^m))$ *storage,*
- $O(ld)$ *tensorizations/matricizations of matrices/tensors with* $\leq r^{m+1}$ *entries,*
- $O(ld^2r^2(n + r^m))$ *arithmetical operations and*
- $O(d)$ *evaluations of the function* F*,*

provided the differential equations in Algorithms 5 and 6 are solved approximately using a fixed number of function evaluations per time step.

Proof. The only nontrivial count regards the arithmetical operations. In Algorithm 5, there are two QR decompositions of $r^m \times r$ matrices, which requires $O(r^m \cdot r^2) = O(r^{m+2})$ operations. In total over all subtrees of $\bar{\tau}$, the total computational cost for the QR decompositions is thus $O(dr^{m+2})$.

The computation of $X^0_{\tau_i} \times_0 \mathbf{S}^{0,\top}_{\tau_i}$ in factorized form requires only computing the smaller product $C^0_{\tau_i} \times_0 \mathbf{S}^{0,\top}_{\tau_i}$, where $C^0_{\tau_i}$ is the core tensor of $X^0_{\tau_i}$, as at the end of the previous subsection. The computational cost for this product is $O(r^{m+2})$ operations, and in total over all subtrees of $\bar{\tau}$, the total computational cost is thus $O(dr^{m+2})$ operations.

The matrix product $\mathbf{U}^{1,\top}_{\tau_i} \mathbf{Mat}_0\big(F_{\tau_i}(t, X^1_{\tau_i} \times_0 \mathbf{S}^\top_{\tau_i})\big)^\top$ is computed recursively using (4.11). This requires $O(|L(\tau_i)|(nr^2 + r^{m+2}))$ operations, where $|L(\tau_i)|$ is the number of leaves of τ_i. In total over all subtrees of $\bar{\tau}$, the computational cost for these products is therefore $O(ld(nr^2 + r^{m+2}))$ operations.

To evaluate the function F_{τ_i}, we need prolongations (which do not require any arithmetical operations) and restrictions. The computational cost for computing a restriction $\pi^\dagger_{\tau,i}$ in the way described in the previous subsection, is $O(|L(\tau_i)|(nr^2 + r^{m+2}))$ operations for the recursive computation of the matrix $\mathbf{R}_{\tau_i}$ via (4.11). The mode-0 multiplication of $\mathbf{R}_{\tau_i}$ with the core tensor requires another $O(r^{m+2})$ operations. One evaluation of F_{τ_i} requires several restrictions from the root down to τ_i and then costs $O(d(nr^2 + r^{m+2}))$ operations in addition to the evaluation of F. In total over all subtrees of $\bar{\tau}$, the computational cost then becomes $O(ld^2(nr^2 + r^{m+2}))$ operations.

In Algorithm 6, we note that the tree tensor network $Z_\tau = F_\tau(t, Y_\tau)$, when written in factorized form as $Z_\tau = G_\tau \times_0 \mathbf{I}_\tau \bigtimes_{i=1}^m \mathbf{W}_{\tau_i}$ for a tree $\tau = (\tau_1, \dots \tau_m)$, has

$$Z_\tau \bigtimes_{i=1}^m \mathbf{U}^\top_{\tau_i} = G_\tau \times_0 \mathbf{I}_\tau \bigtimes_{i=1}^m \mathbf{U}^\top_{\tau_i} \mathbf{W}_{\tau_i} .$$

The products $\mathbf{U}^\top_{\tau_i} \mathbf{W}_{\tau_i}$ are computed recursively via (4.11) in $O(|L(\tau)|(nr^2 + r^{m+2}))$ operations. In total over all subtrees of $\bar{\tau}$, the computational cost then again becomes $O(dl(nr^2 + r^{m+2}))$ operations. ∎

5. Exactness property of the TTN integrator. We will show that under a non-degeneracy condition, the TTN integrator with the tree rank (r_τ) reproduces time-dependent tree tensor networks $A(t)$ with the same tree rank *exactly* at every time step when the integrator is applied with $F(t, Y) = \dot{A}(t)$ and exact initial value $Y^0 = A(t_0)$. Such an exactness result is already known for the special cases of projector-splitting integrators for low-rank matrices [19], tensor trains / matrix product states [20], and Tucker tensors [21]. The latter result will now be used in a recursive way to prove the exactness property of the TTN integrator.

We first formulate the non-degeneracy condition. Consider a time-dependent family of tree tensor networks $A(t)$ of full tree rank $(r_\tau)_{\tau \leq \bar{\tau}}$, and set $Y^0_{\bar{\tau}} = A(t_0)$, for which we consider the restricted tensor networks $A_\tau(t) := \big(A(t)\big)_\tau$ defined by the restrictions (4.7) associated with $Y^0_{\bar{\tau}}$ for the subtrees $\tau \leq \bar{\tau}$. By Lemma 4.3, we then have for every subtree $\tau \leq \bar{\tau}$ that

$$A_\tau(t_0) \text{ has full tree rank } (r_\sigma)_{\sigma \leq \tau} \text{ for every subtree } \tau \leq \bar{\tau}. \tag{5.1}$$

We impose the condition that the same full-rank property still holds at $t_1 > t_0$:

$$A_\tau(t_1) \text{ has full tree rank } (r_\sigma)_{\sigma \le \tau} \text{ for every subtree } \tau \le \bar{\tau}. \tag{5.2}$$

THEOREM 5.1 (Exactness). *Let $A(t)$ be a continuously differentiable time-dependent family of tree tensor networks $A(t)$ of full tree rank $(r_\tau)_{\tau \le \bar{\tau}}$ for $t_0 \le t \le t_1$, and suppose that the non-degeneracy condition* (5.2) *is satisfied. Then the recursive TTN integrator used with the same tree rank $(r_\tau)_{\tau \in T(\bar{\tau})}$ for $F(t, Y) = \dot{A}(t)$ is exact: starting from $Y^0 = A(t_0)$ we obtain $Y^1 = A(t_1)$.*

Proof. The result is obtained from the exactness result of the Tucker integrator that was proved in [21] and an induction argument over the height of the trees. The height is defined in a formal way as follows:

(i) If $\tau = \ell \in \mathcal{L}$, then we set $h(\tau) = 0$; i.e., leaves have height 0.

(ii) If $\tau = (\tau_1, \dots, \tau_m) \in \mathcal{T}$, then we set $h(\tau) = 1 + \max\{h(\tau_1), \dots, h(\tau_m)\}$.

We note that, since the restricitions $\pi_\tau^\dagger$ for $\tau \le \bar{\tau}$ do not depend on time t, time differentiation commutes with these linear maps and we have

$$\dot{A}_\tau(t) := \frac{d}{dt} A_\tau(t) = \big(\dot{A}(t)\big)_\tau.$$

(i) Consider first trees $\tau = (\tau_1, \dots, \tau_m)$ of height 1. The tree tensor network $A_\tau(t)$ is then a Tucker tensor, which by (5.1) and (5.2) has full multilinear rank $(r_\tau, r_{\tau_1}, \dots, r_{\tau_m})$ at both $t = t_0$ and $t = t_1$. The TTN integrator with $F_\tau(t, Y_\tau) = \dot{A}_\tau(t)$ is in this case the same as the Tucker integrator of [21] and hence reproduces $A_\tau(t_1)$ exactly by [21, Theorem 4.1]. We note that condition (5.2) for the leaves τ_i corresponds to the invertibility condition in [21, Theorem 4.1].

(ii) For trees of height $k \ge 2$ we work with the induction hypothesis that the recursive TTN integrator with $F_\tau(t, Y_\tau) = \dot{A}_\tau(t)$ is exact for all trees $\tau < \bar{\tau}$ of height strictly smaller than k. For a tree $\tau = (\tau_1, \dots, \tau_m)$ of height k the TTN integrator is therefore exact for the subtrees τ_i, and hence the recursive steps in the TTN integrator are solved exactly. This reduces the recursive TTN integrator to the Tucker integrator for $F_\tau(t, Y_\tau) = \dot{A}_\tau(t)$, where by (5.1) and (5.2), the tensor $A_\tau(t)$, viewed as a Tucker tensor $A_\tau(t) = C_\tau(t) \bigtimes_{i=1}^m \mathbf{U}_{\tau_i}(t)$, has full multilinear rank $(r_\tau, r_{\tau_1}, \dots, r_{\tau_m})$ at both $t = t_0$ and $t = t_1$. From the exactness result of [21, Theorem 4.1] it then follows that the integrator reproduces $A_\tau(t_1)$ exactly. This completes the induction argument. Finally, we thus obtain the exactness result for the maximal tree $\bar{\tau}$, which is the stated result. ∎

6. Error bound. We derive an error bound for the integrator that is independent of singular values of matricizations of the connecting tensors, based on the corresponding result for Tucker tensors proved in [21], which in turn was based on the corresponding result for matrices proved in [12]. We recall the notation $\mathcal{V}_\tau$ for the tensor space (4.1) and $\mathcal{M}_\tau$ for the tree tensor network manifold (4.10). We set $\mathcal{V} = \mathcal{V}_{\bar{\tau}}$ and $\mathcal{M} = \mathcal{M}_{\bar{\tau}}$ for the full tree $\bar{\tau}$.

We assume that $F : [0, t^*] \times \mathcal{M} \to \mathcal{V}$ is Lipschitz continuous and bounded,

$$\|F(t, Y) - F(t, \widetilde{Y})\| \le L \|Y - \widetilde{Y}\| \qquad \text{for all } Y, \widetilde{Y} \in \mathcal{M}, \tag{6.1}$$

$$\|F(t, Y)\| \le B \qquad \text{for all } Y \in \mathcal{M}. \tag{6.2}$$

Here and in the following, the chosen norm $\|\cdot\|$ is the tensor Euclidean norm. As usual in the numerical analysis of ordinary differential equations, this could be weakened to

a local Lipschitz condition and local bound in a neighborhood of the exact solution $A(t)$ of the tensor differential equation (1.1) to the initial data $A(t_0) = A^0 \in \mathcal{V}$, but for convenience we will work with the global Lipschitz condition and bound.

We further assume that $F(t, Y)$ is in the tangent space $\mathcal{T}_Y\mathcal{M}$ up to a small remainder: with $P(Y)$ denoting the orthogonal projection onto $\mathcal{T}_Y\mathcal{M}$, we assume that for some $\varepsilon > 0$,

$$\|F(t, Y) - P(Y)F(t, Y)\| \le \varepsilon \tag{6.3}$$

for all $(t, Y) \in [0, t^*] \times \mathcal{M}$ in some ball $\|Y\| \le \rho$, where it is assumed that the exact solution $A(t)$, $0 \le t \le t^*$, has a bound that is strictly smaller than ρ.

Finally, we assume that the initial value A^0 and the starting value $Y^0 \in \mathcal{M}$ of the numerical method are δ-close:

$$\|Y^0 - A^0\| \le \delta. \tag{6.4}$$

THEOREM 6.1 (Error bound). *Under the above assumptions, the error of the numerical approximation Y^n at $t_n = nh$, obtained with n time steps of the TTN integrator with step size $h > 0$, is bounded by*

$$\|Y^n - A(t_n)\| \le c_0\delta + c_1\varepsilon + c_2 h \qquad \text{for } \ t_n \le t^*,$$

where c_i depend only on L, B, t^*, and the tree $\bar\tau$. This holds true provided that δ, ε and h are so small that the above error bound guarantees that $\|Y^n\| \le \rho$.

We remark that the error bound is still valid for sufficiently small δ, ε and h if the bound (6.3) is satisfied only in some tubular neighborhood $\{(t, Y) \in [0, t^*] \times \mathcal{M} : \|Y - A(t)\| \le \vartheta\}$ for an arbitrary fixed $\vartheta > 0$. We do not include the proof of this more general result, because the higher technical intricacies would obscure the basic argument of the proof.

The proof of Theorem 6.1 works recursively, based on the corresponding result for Tucker tensors given in [21] and using a similar induction argument to the proof of Theorem 5.1. To make this feasible, we need that the conditions on $F = F_{\bar\tau}$ are also satisfied for the reduced functions F_τ for every subtree $\tau \le \bar\tau$, which are constructed recursively in Definition 4.2. For $Y_\tau \in \mathcal{M}_\tau$, let $P_\tau(Y_\tau)$ be the orthogonal projection onto the tangent space $T_{Y_\tau}\mathcal{M}_\tau$. We have the following remarkable property.

LEMMA 6.2. *If $F_{\bar\tau} = F$ satisfies conditions* (6.1)–(6.3)*, then we have for every subtree $\tau \le \bar\tau$, with the same L, B, and ε,*

$$\|F_\tau(t, Y_\tau) - F_\tau(t, \widetilde{Y_\tau})\| \le L\|Y_\tau - \widetilde{Y_\tau}\| \qquad \text{for all } Y_\tau, \widetilde{Y_\tau} \in \mathcal{M}_\tau, \tag{6.5}$$

$$\|F_\tau(t, Y_\tau)\| \le B \qquad \text{for all } Y_\tau \in \mathcal{M}_\tau \tag{6.6}$$

and

$$\|F_\tau(t, Y_\tau) - P_\tau(Y_\tau)F_\tau(t, Y_\tau)\| \le \varepsilon \tag{6.7}$$

for all $(t, Y_\tau) \in [0, t^*] \times \mathcal{M}_\tau$ with $\|Y_\tau\| \le \rho$.

The proof of the ε-bound (6.7) is based on the following lemma.

LEMMA 6.3. *Let $\tau = (\tau_1, \dots, \tau_m) \in \mathcal{T}$ and $i = 1, \dots, m$. Let $M_\tau : \mathcal{M}_\tau \to \mathcal{V}_\tau$ be such that it maps into the tangent space:*

$$M_\tau(Y_\tau) \in T_{Y_\tau}\mathcal{M}_\tau \quad \text{for all } \ Y_\tau \in \mathcal{M}_\tau.$$

Let $\pi^\dagger_{\tau,i}$ and $\pi_{\tau,i}$ be the restrictions and prolongations corresponding to some $Y^0_\tau \in \mathcal{M}_\tau$, and define

$$M_{\tau_i} = \pi^\dagger_{\tau,i} \circ M_\tau \circ \pi_{\tau,i}.$$

Then, $M_{\tau_i} : \mathcal{M}_{\tau_i} \to \mathcal{V}_{\tau_i}$ also maps into the tangent space:

$$M_{\tau_i}(Y_{\tau_i}) \in T_{Y_{\tau_i}}\mathcal{M}_{\tau_i} \quad \textit{for all} \;\; Y_{\tau_i} \in \mathcal{M}_{\tau_i}.$$

Proof. Let $Y_{\tau_i} \in \mathcal{M}_{\tau_i}$ and define $Y_\tau = \pi_{\tau,i}(Y_{\tau_i})$, which by Lemma 4.4 is in $\mathcal{M}_\tau$. By assumption, $M_\tau(Y_\tau) \in T_{Y_\tau}\mathcal{M}_\tau$, and hence there exists a path $X_\tau(\theta) \in \mathcal{M}_\tau$, for θ near 0, with $X_\tau(0) = Y_\tau$ and $\frac{d}{d\theta}|_{\theta=0} X_\tau = M_\tau(Y_\tau)$. Then, the restricted path $X_{\tau_i}(\theta) = \pi^\dagger_{\tau,i} X_\tau(\theta)$ has

$$X_{\tau_i}(0) = \pi^\dagger_{\tau,i}(Y_\tau) = \pi^\dagger_{\tau,i}(\pi_{\tau,i}(Y_{\tau_i})) = Y_{\tau_i},$$

because $\pi^\dagger_{\tau,i}$ is a left inverse of $\pi_{\tau,i}$ by Lemma 4.1. By local continuity of the rank, we then have $X_{\tau_i}(\theta) \in \mathcal{M}_{\tau_i}$. Moreover,

$$\left.\frac{d}{d\theta}\right|_{\theta=0} X_{\tau_i} = \pi^\dagger_{\tau,i} \left.\frac{d}{d\theta}\right|_{\theta=0} X_\tau = \pi^\dagger_{\tau,i} M_\tau(Y_\tau) = M_{\tau_i}(Y_{\tau_i}).$$

Hence, $M_{\tau_i}(Y_{\tau_i}) \in T_{Y_{\tau_i}}\mathcal{M}_{\tau_i}$. □

Proof. (of Lemma 6.2) The Lipschitz bound and the norm bound of F_τ follow directly from the corresponding bounds of F, using that restriction and prolongation are operators of norm 1 by Lemma 4.1. It then remains to show (6.7). For $Y \in \mathcal{M}$ we write

$$F(t,Y) = P(Y)F(t,Y) + (I - P(Y))F(t,Y) \equiv M(t,Y) + R(t,Y)$$

with $M(t,Y) \in T_Y\mathcal{M}$ and $\|R(t,Y)\| \le \varepsilon$ by (6.3). We let $M_{\bar\tau} = M$ and $R_{\bar\tau} = R$ and define recursively, for $\tau = (\tau_1, \ldots, \tau_m) \le \bar\tau$,

$$M_{\tau_i} = \pi^\dagger_{\tau,i} \circ M_\tau \circ \pi_{\tau,i},$$
$$R_{\tau_i} = \pi^\dagger_{\tau,i} \circ R_\tau \circ \pi_{\tau,i}.$$

By Lemma 6.3, for every subtree $\tau \le \bar\tau$, M_τ maps into the tangent space:

$$M_\tau(Y_\tau) \in T_{Y_\tau}\mathcal{M}_\tau \quad \text{for all} \;\; Y_\tau \in \mathcal{M}_\tau.$$

Hence,

$$(I - P_\tau(Y_\tau))F_\tau(t,Y_\tau) = (I - P_\tau(Y_\tau))R_\tau(t,Y_\tau),$$

and once again, since restriction and prolongation are operators of norm 1, it follows from (6.3) that

$$\|(I - P_\tau(Y_\tau))F_\tau(t,Y_\tau)\| \le \|R_\tau(t,Y_\tau)\| \le \varepsilon.$$

This proves Lemma 6.2. □

Proof. (of Theorem 6.1) It suffices to assume that $Y^0 = A(t_0) \in \mathcal{M}$, since the difference of exact solutions of the differential equation (1.1) corresponding to initial

values that differ at most by δ, is bounded by $c_0\delta$ for $t_0 \le t \le t^*$ under the imposed Lipschitz condition on F. Moreover, it then suffices to show that the local error after one time step is of magnitude $O(h(\varepsilon + h))$. The result for the global error is then obtained with the familiar Lady Windermere's fan argument, as in [12] and [21].

As in the proof of Theorem 5.1, we proceed by induction on the height of the tree.

For trees of height 1, the recursive TTN integrator coincides with the Tucker integrator of [21], for which the error estimate has been proved in [21].

For trees $\tau = (\tau_1, \dots, \tau_m)$ of higher height, we observe that in the recursive TTN integrator, the differential equations for Y_{τ_i} are solved approximately by intermediate tree tensor networks with lower height, for which the $O(h(\varepsilon + h))$ error bound holds by the induction hypothesis. If instead the differential equations for Y_{τ_i} were solved exactly, then the integrator would again reduce to the Tucker integrator and error in this idealized Y_τ after one step would be $O(h(\varepsilon+h))$. By studying the influence of the inexact solution of the differential equation for Y_{τ_i} on the error (as in [12, Subsection 2.6.3]), we find that the error of the actual Y_τ is still of magnitude $O(h(\varepsilon + h))$. We omit the details of this perturbation argument, since it is cumbersome to write down explicitly and requires no ideas beyond using the triangle inequality.

This completes the induction argument. Finally, we thus obtain the error bound for $\bar\tau$, which yields the result of Theorem 6.1. □

7. Numerical experiments. The recursive TTN integrator has already been applied to problems from plasma physics [5] and quantum physics [13], where numerical results are reported. In the following we therefore give just two illustrative numerical examples. We choose the tree $\bar\tau$ of Figure 2.1. The dimensions n_ℓ and the ranks r_τ are taken the same for all the nodes and are fixed to $n_\ell = n = 16$ for all leaves ℓ and $r_\tau = r = 5$ for all subtrees $\tau < \bar\tau$. We have chosen such small dimensions n to be able to easily compute the reference solution, which is a full tensor with n^6 entries. In contrast, the storage for the tree tensor network for this tree is $6nr + r^4 + 2r^3$ entries.

The computations were done using Matlab R2017a software with Tensor Toolbox package v2.6 [2]. The implementation of the TTN is done using an Object Oriented Paradigm; we define in Matlab a class `Node` with two properties: `Value` and `Children`. The `Value` can be either a connection tensor or an orthonormal matrix. The `Children` is an ordered list of objects of type `Node`. If the `Children` list is empty, we are on one of the leaves of the tree; the `Value` of the `Node` is by construction an orthonormal matrix. The TTN is defined as an object of type `Node`.

The recursive TTN integrator is then implemented by recursively applying the Extended Tucker Integrator to an object of type `Node`. The recursion process in the recursive TTN algorithm is controlled by counting the elements of the `Children` list associated with the `Node`: if empty, we are on one of the leaves of the tree.

7.1. Tree tensor network addition and retraction. Let $\bar\tau$ be the given tree and let $\mathcal{M}_{\bar\tau}$ be the manifold of tree tensor networks of given dimensions (n_ℓ) and tree rank $(r_\tau)_{\tau\le\bar\tau}$; see Section 2.3. We consider the addition of two given tensors $A \in \mathcal{M}_{\bar\tau}$ and $B \in \mathcal{T}_A\mathcal{M}_{\bar\tau}$ (a tangent tensor),

$$C = A + B.$$

Then, C is a tree tensor network on the same tree but of larger rank. We want to compute a tree tensor network retraction to the manifold $\mathcal{M}_{\bar\tau}$, i.e., to the original

tree rank $(r_\tau)_{\tau \leq \bar{\tau}}$. Such a retraction is typically required in optimization problems on low-rank manifolds and needs to be computed in each iterative step. The approach considered here consists of reformulating the addition problem as the solution of the following differential equation at time $t = 1$:

$$\dot{C}(t) = B, \quad C(0) = A.$$

We compare the approximation $Y^1 \in \mathcal{M}_{\bar{\tau}}$, computed with one time step of the recursive TTN integrator with step size $h = 1$, with a different retraction, denoted by X, obtained by computing the full addition C and recursively retracting to the manifold $\mathcal{M}_\tau$ for each $\tau \leq \bar{\tau}$. For the latter, we use the built-in function `tucker_als` of the Tensor Toolbox Package [2]; we recursively apply the function to the full tensor C and its retensorized basis matrices.

These comparisons are illustrated in Figure 7.1 where the norm of B is varied. We observe both retractions Y^1 and X have very similar error, and their difference is considerably smaller than their errors. Decreasing the norm of the tensor B reduces the approximation error as expected, proportional to $\|B\|^2$. This behaviour of the TTN integrator used for retraction is the same as observed for the Tucker integrator in [21] for the analogous problem of the addition of a Tucker tensor of given multilinear rank and a tangent tensor.

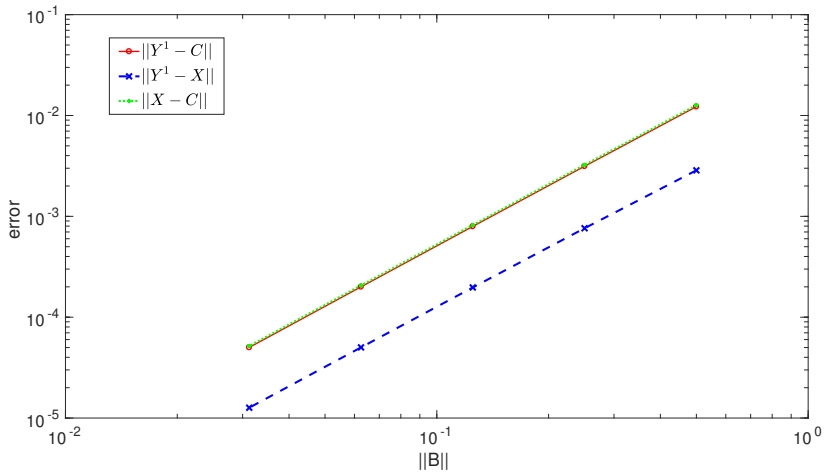


FIG. 7.1. *Error of retracted tree tensor network sum.*

The advantage of the retraction via the TTN integrator is that the result is completely built within the tree tensor network manifold. No further retraction is needed, which is favorable for storage and computational complexity.

7.2. Verification of the exactness property. We consider a tree tensor network $A^0 \in \mathcal{M}_{\bar{\tau}}$. For each subtree $\tau \leq \bar{\tau}$, let $\mathbf{W}_\tau \in \mathbb{R}^{r_\tau \times r_\tau}$ be a skew-symmetric matrix which we choose of Frobenius norm 1. We consider a time-dependent tree tensor network $A(t) \in \mathcal{M}_{\bar{\tau}}$ such that $A(t_0) = A^0$ with basis matrices propagated in time through

$$\mathbf{U}_\ell(t) = e^{t\mathbf{W}_\ell}\, \mathbf{U}_\ell^0, \qquad \ell \in L(\bar{\tau})$$

and the connection tensors changed according to

$$C_\tau(t) = C^0_\tau \times_0 e^{t\mathbf{W}_\tau}, \qquad \tau \leq \bar{\tau},\ \tau \notin L(\bar{\tau}).$$

The time-dependent tree tensor network does not change rank and as predicted by Theorem 5.1, it is reproduced exactly by the recursive TTN integrator, up to round-off errors. The absolute errors $\|Y_n - A(t_n)\|$ calculated at time $t_n = nh$ with step sizes $h = 0.1,\ 0.01,\ 0.001$ until time $t^* = 1$ are shown in Figure 7.2.

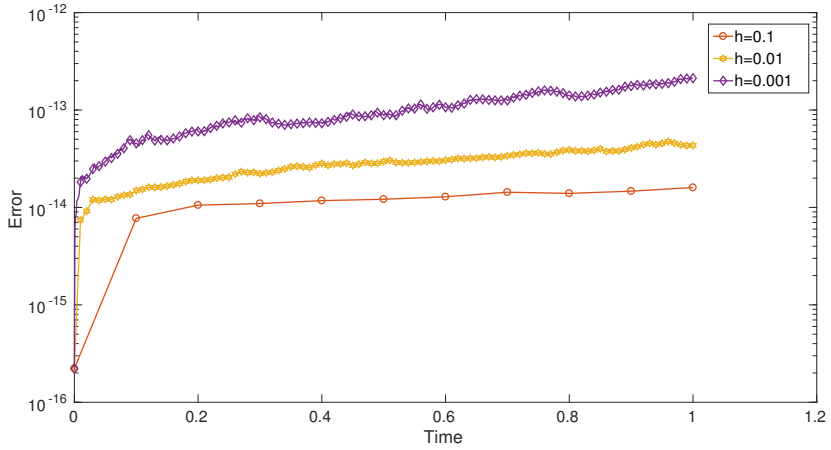


FIG. 7.2. *Error vs. time in a case of exactness up to round-off errors.*

Acknowledgements. We thank two anonymous referees for their helpful comments on a previous version.

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) — Project-ID 258734477 — SFB 1173 and DFG GRK 1838.

REFERENCES

[1] P.-A. Absil and I. V. Oseledets. Low-rank retractions: a survey and new results. *Comput. Optim. Appl.*, 62(1):5–29, 2015.
[2] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.6. Available online, February 2015.
[3] D. Bauernfeind and M. Aichhorn. Time dependent variational principle for tree tensor networks. *arXiv preprint arXiv:1908.03090*, 2019.
[4] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21:1253–1278, 2000.
[5] L. Einkemmer and C. Lubich. A low-rank projector-splitting integrator for the Vlasov-Poisson equation. *SIAM J. Sci. Comput.*, 40(5):B1330–B1360, 2018.
[6] A. Falcó, W. Hackbusch, and A. Nouy. Geometric structures in tensor representations (final release). *arXiv preprint arXiv:1505.03027*, 2015.
[7] A. Falcó, W. Hackbusch, and A. Nouy. Tree-based tensor formats. *SeMA J.*, pages 1–15, 2018.
[8] W. Hackbusch. *Multigrid methods and applications*, volume 4 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1985.
[9] W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*. Springer, 2012.
[10] J. Haegeman, C. Lubich, I. Oseledets, B. Vandereycken, and F. Verstraete. Unifying time evolution and optimization with matrix product states. *Physical Review B*, 94(16):165116, 2016.

[11] U. Helmke and J. B. Moore. *Optimization and dynamical systems*. Communications and Control Engineering Series. Springer-Verlag, London, 1994.
[12] E. Kieri, C. Lubich, and H. Walach. Discretized dynamical low-rank approximation in the presence of small singular values. *SIAM J. Numer. Anal.*, 54:1020–1038, 2016.
[13] B. Kloss, Y. B. Lev, and D. R. Reichman. Studying dynamics in two-dimensional quantum lattices using tree tensor network states. *arXiv preprint arXiv:2003.08944*, 2020.
[14] O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM J. Matrix Anal. Appl.*, 29(2):434–454, 2007.
[15] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51:455–500, 2009.
[16] P. Kramer and M. Saraceno. *Geometry of the time-dependent variational principle in quantum mechanics*, volume 140 of *Lecture Notes in Physics*. Springer-Verlag, Berlin-New York, 1981.
[17] C. Lubich. *From quantum to classical molecular dynamics: reduced models and numerical analysis*. Zurich Lectures in Advanced Mathematics. European Mathematical Society (EMS), Zürich, 2008.
[18] C. Lubich. Time integration in the multiconfiguration time-dependent Hartree method of molecular quantum dynamics. *Appl. Math. Res. Express*, 2015:311–328, 2015.
[19] C. Lubich and I. V. Oseledets. A projector-splitting integrator for dynamical low-rank approximation. *BIT*, 54:171–188, 2014.
[20] C. Lubich, I. V. Oseledets, and B. Vandereycken. Time integration of tensor trains. *SIAM J. Numer. Anal.*, 53:917–941, 2015.
[21] C. Lubich, B. Vandereycken, and H. Walach. Time integration of rank-constrained Tucker tensors. *SIAM J. Numer. Anal.*, 56:1273–1290, 2018.
[22] I. V. Oseledets. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33(5):2295–2317, 2011.
[23] D. Perez-García, F. Verstraete, M. M. Wolf, and J. I. Cirac. Matrix product state representations. *Quantum Information and Computation*, 7(5-6):401–430, 2007.
[24] Y.-Y. Shi, L.-M. Duan, and G. Vidal. Classical simulation of quantum many-body systems with a tree tensor network. *Physical Review A*, 74(2):022320, 2006.
[25] A. Uschmajew and B. Vandereycken. The geometry of algorithms using hierarchical tensors. *Linear Algebra Appl.*, 439(1):133–166, 2013.
[26] H. Wang and M. Thoss. Multilayer formulation of the multiconfiguration time-dependent Hartree theory. *J. Chem. Phys.*, 119(3):1289–1299, 2003.

Appendix B

Time integration of symmetric and anti-symmetric low-rank matrices and Tucker tensors

Time integration of symmetric and anti-symmetric low-rank matrices and Tucker tensors

Gianluca Ceruti, Christian Lubich

Abstract A numerical integrator is presented that computes a symmetric or skew-symmetric low-rank approximation to large symmetric or skew-symmetric time-dependent matrices that are either given explicitly or are the unknown solution to a matrix differential equation. A related algorithm is given for the approximation of symmetric or anti-symmetric time-dependent tensors by symmetric or anti-symmetric Tucker tensors of low multilinear rank. The proposed symmetric or anti-symmetric low-rank integrator is different from recently proposed projector-splitting integrators for dynamical low-rank approximation, which do not preserve symmetry or anti-symmetry. However, it is shown that the (anti-)symmetric low-rank integrators retain favourable properties of the projector-splitting integrators: low-rank time-dependent matrices and tensors are reproduced exactly, and the error behaviour is robust to the presence of small singular values, in contrast to standard integration methods applied to the differential equations of dynamical low-rank approximation. Numerical experiments illustrate the behaviour of the proposed integrators.

1 Introduction

In this paper we propose and analyse an algorithm that computes a symmetric or skew-symmetric low-rank approximation to large symmetric or skew-symmetric time-dependent matrices that are either given explicitly or are the unknown solution to a matrix differential equation. A related algorithm is given for the approximation of symmetric or anti-symmetric time-dependent tensors by symmetric or anti-symmetric Tucker tensors of low multilinear rank.

In the matrix case, motivation for this work comes from Lyapunov and Riccati differential equations, which have large symmetric matrices as solutions, which can often be well approximated by low-rank matrices [19]. For tensors, our main motivation comes from the quantum dynamics of bosonic or fermionic systems, where the symmetric or anti-symmetric wave function is approximated by low-rank symmetric or anti-symmetric Tucker tensors in the MCTDHB and MCTDHF methods for bosons

G. Ceruti and Ch. Lubich
Mathematisches Institut, Universität Tübingen, Auf der Morgenstelle 10, D-72076 Tübingen, Germany. E-mail: {ceruti, lubich}@na.uni-tuebingen.de

and fermions, respectively [1,4]. An efficient integrator that preserves symmetry and anti-symmetry and uses them to reduce the computational complexity, is needed in these and other applications, such as using a step of the integrator as a computationally efficient retraction in optimization algorithms for (anti-)symmetric low-rank matrices and tensors.

The algorithms proposed in this paper are non-trivial modifications of the projector-splitting integrators for the dynamical low-rank approximation of matrices and Tucker tensors that were proposed in [17] and [16,18], respectively. The projector-splitting integrators have been shown to possess remarkable robustness to the typical presence of small singular values [10,18], as opposed to applying standard integrators to the differential equations of dynamical low-rank approximation that are given in [12,13]. However, the projector-splitting integrators do not preserve symmetry or anti-symmetry.

We will show that the (anti-)symmetry-preserving integrators proposed here retain the robustness with respect to small singular values of the projector-splitting algorithms. This relies on an exactness property, namely that time-dependent matrices and tensors of the approximation rank are reproduced exactly by the integrator. This exactness property will also be shown to be retained from the projector-splitting integrators. We note, however, that the integrators proposed here can no longer be interpreted as splitting integrators.

The new (anti-)symmetry-preserving integrators are favourable also from the computational viewpoint: compared with the projector-splitting integrator, the computational cost is halved in the (skew-)symmetric matrix case; in the case of d-dimensional (anti-)symmetric tensors, the computational cost for the core tensor is reduced by the factor $1/d!$, and that for the basis matrices by $1/d$.

A first attempt to modify the projector-splitting integrator and preserve the symmetry in the matrix setting, can be found in [19]: numerical examples show the correct behaviour of the approximate solution but no convergence analysis or extension to multi-dimensional arrays is provided, and no use of the symmetry is made to reduce the computational effort.

The outline of the paper is the following: in Section 2, we briefly restate the idea of dynamical low-rank approximation for matrices and we present the matrix projector-splitting integrator with some of its properties. In Section 3, we consider the case of (skew-)symmetric matrices; we present the (skew-)symmetry-preserving low-rank integrator and study its properties. In Section 4, we recapitulate the projector-splitting integrator for low-rank Tucker tensors. In Section 5, we present the integrator for (anti)-symmetric tensors of low multilinear rank and study its properties. In the final section, we present numerical experiments that illustrate the approximation properties and the robustness to small singular values.

Throughout the paper, we use the convention to denote matrices by boldface capital letters and tensors by italic capital letters.

2 General matrices: recap of the projector-splitting integrator for dynamical low-rank approximation

The objective is to approximate large time-dependent matrices $\mathbf{A}(t) \in \mathbb{R}^{m\times n}$ for $0 \leq t \leq T$ by rank-r matrices $\mathbf{Y}(t)$ with comparatively low rank $r \ll m, n$, which require much less storage than $\mathbf{A}(t)$ when they are available in a factorized, SVD-like form.

The large, or often too large matrices $\mathbf{A}(t)$ may be given explicitly or they are the unknown solution to a matrix differential equation (with right-hand side function $\mathbf{F} : \mathbb{R} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$)

$$\dot{\mathbf{A}}(t) = \mathbf{F}(t, \mathbf{A}(t)), \qquad \mathbf{A}(t_0) = \mathbf{A}_0. \tag{1}$$

Dynamical low-rank approximation as presented in [12] determines $\mathbf{Y}(t)$ as the solution of a projected matrix differential equation, with a projection $\mathrm{P}(\mathbf{Y})$ onto the tangent space $T_{\mathbf{Y}}\mathcal{M}_r$ of the manifold of rank-r matrices at $\mathbf{Y} \in \mathcal{M}_r$,

$$\dot{\mathbf{Y}}(t) = \mathrm{P}(\mathbf{Y}(t))\mathbf{F}(t, \mathbf{Y}(t)), \qquad \mathbf{Y}(t_0) = \mathbf{Y}_0, \tag{2}$$

where $\mathbf{Y}_0$ is a rank-r approximation to $\mathbf{A}_0$, typically obtained by a truncated singular value decomposition. (Here, $\mathbf{F}(t, \mathbf{Y}) = \dot{\mathbf{A}}(t)$ if $\mathbf{A}(t)$ is given explicitly.) The solution $\mathbf{Y}(t)$ to this projected matrix differential equation then stays in the rank-r manifold $\mathcal{M}_r$.

To make this abstract formulation practically useful, rank-r matrices $\mathbf{Y}(t)$ are written (non-uniquely) in factored form

$$\mathbf{Y}(t) = \mathbf{U}(t)\mathbf{S}(t)\mathbf{V}(t)^\top, \tag{3}$$

where the slim matrices $\mathbf{U}(t) \in \mathbb{R}^{m\times r}$ and $\mathbf{V}(t) \in \mathbb{R}^{n\times r}$ each have r orthonormal columns, and the small square matrix $\mathbf{S}(t) \in \mathbb{R}^{r\times r}$ is invertible. We choose the tangent space projection $\mathrm{P}(\mathbf{Y})$ as the orthogonal projection onto $T_{\mathbf{Y}}(\mathcal{M}_r)$ with respect to the Euclidean or Frobenius inner product $\langle \mathbf{A}, \mathbf{B}\rangle = \mathbf{vec}(\mathbf{A})^\top \mathbf{vec}(\mathbf{B})$, where $\mathbf{vec}(\mathbf{A}) \in \mathbb{R}^{mn}$ is a vectorization of $\mathbf{A}$. Then, $\mathrm{P}(\mathbf{Y})$ is given as an alternating sum of three subprojections [12],

$$\mathrm{P}(\mathbf{Y})\mathbf{Z} = \mathbf{Z}\mathbf{V}\mathbf{V}^\top - \mathbf{U}\mathbf{U}^\top\mathbf{Z}\,\mathbf{V}\mathbf{V}^\top + \mathbf{U}\mathbf{U}^\top\mathbf{Z}. \tag{4}$$

The projector-splitting integrator of [17] is a Lie–Trotter or Strang splitting method that splits the right-hand side of (2) according to the three terms in (4). It turned out that such a splitting combines very well with the factorization (3). In the first substep of a Lie–Trotter splitting, $\mathbf{K} := \mathbf{U}\mathbf{S}$ is updated, in the second substep $\mathbf{S}$ is updated, and in the third substep $\mathbf{L} := \mathbf{V}\mathbf{S}^\top$. The algorithm alternates between the numerical solution of matrix differential equations (of dimensions $m \times r$, $r \times r$, $n \times r$) and orthogonal decompositions of slim matrices (of dimensions $m \times r$ and $n \times r$). One time step of integration from time t_0 to $t_1 = t_0 + h$ starting from a factored rank-r matrix $\mathbf{Y}_0 = \mathbf{U}_0\mathbf{S}_0\mathbf{V}_0^\top$ proceeds as follows:

1. **K-step** : Update $\mathbf{U}_0 \to \mathbf{U}_1, \mathbf{S}_0 \to \hat{\mathbf{S}}_1$
   Integrate from $t = t_0$ to t_1 the $m \times r$ matrix differential equation

   $$\dot{\mathbf{K}}(t) = \mathbf{F}(t, \mathbf{K}(t)\mathbf{V}_0^\top)\mathbf{V}_0, \qquad \mathbf{K}(t_0) = \mathbf{U}_0\mathbf{S}_0.$$

   Perform a QR factorization $\mathbf{K}(t_1) = \mathbf{U}_1\hat{\mathbf{S}}_1$.
2. **S-step** : Update $\hat{\mathbf{S}}_1 \to \tilde{\mathbf{S}}_0$
   Integrate from $t = t_0$ to t_1 the $r \times r$ matrix differential equation

   $$\dot{\mathbf{S}}(t) = -\mathbf{U}_1^\top \mathbf{F}(t, \mathbf{U}_1\mathbf{S}(t)\mathbf{V}_0^\top)\mathbf{V}_0, \qquad \mathbf{S}(t_0) = \hat{\mathbf{S}}_1,$$

   and set $\tilde{\mathbf{S}}_0 = \mathbf{S}(t_1)$.

3. **L-step** : Update $\mathbf{V}_0 \to \mathbf{V}_1, \tilde{\mathbf{S}}_0 \to \mathbf{S}_1$
   Integrate from $t = t_0$ to t_1 the $n \times r$ matrix differential equation

$$\dot{\mathbf{L}}(t) = \mathbf{F}(t, \mathbf{U}_1 \mathbf{L}(t)^\top)^\top \mathbf{U}_1, \qquad \mathbf{L}(t_0) = \mathbf{V}_0 \tilde{\mathbf{S}}_0^\top .$$

   Perform a QR factorization $\mathbf{L}(t_1) = \mathbf{V}_1 \mathbf{S}_1^\top$.

   Then, the approximation after one time step is given by

$$\mathbf{Y}_1 = \mathbf{U}_1 \mathbf{S}_1 \mathbf{V}_1^\top .$$

To proceed, we iterate the procedure taking $\mathbf{Y}_1$ as starting point for the next step.

The above algorithm describes the first-order Lie–Trotter splitting. The algorithm for the second-order Strang splitting is obtained by concatenating the above algorithm with the same algorithm in reversed order, each for half the step-size; see [17] for the detailed description.

The projector-splitting integrator has remarkable properties. First, it reproduces rank-r matrices without error.

Theorem 1 (Exactness property, [17, Theorem 4.1]) *Let $\mathbf{A}(t) \in \mathbb{R}^{m \times n}$ be of rank r for $t_0 \le t \le t_1$, so that $\mathbf{A}(t)$ has a factorization (3), $\mathbf{A}(t) = \mathbf{U}(t)\mathbf{S}(t)\mathbf{V}(t)^\top$. Moreover, assume that the $r \times r$ matrix $\mathbf{V}(t_1)^\top \mathbf{V}(t_0)$ is invertible. With $\mathbf{Y}_0 = \mathbf{A}(t_0)$, the projector-splitting integrator for $\dot{\mathbf{Y}}(t) = \mathrm{P}(\mathbf{Y}(t))\dot{\mathbf{A}}(t)$ is then exact: $\mathbf{Y}_1 = \mathbf{A}(t_1)$.*

The second remarkable property is the robustness of the algorithm to the presence of small singular values of the solution or its approximation. This is in contrast to standard integrators applied to (2) or the equivalent differential equations for the factors $\mathbf{U}(t)$, $\mathbf{S}(t)$, $\mathbf{V}(t)$, which contain a factor $\mathbf{S}(t)^{-1}$ on the right-hand sides [12, Prop. 2.1]. Moreover, the local Lipschitz constant of the tangent space projection $\mathrm{P}(\cdot)$ is proportional to the inverse of the smallest nonzero singular value [12, Lemma 4.2]. The appearance of small singular values is typical in applications, because the smallest singular value retained in the approximation cannot be expected to be much larger than the largest discarded singular value of the solution, which needs to be small to obtain good accuracy of the low-rank approximation.

Theorem 2 (Robust error bound, [10, Theorem 2.1]) *Let $\mathbf{A}(t)$ denote the solution of the matrix differential equation (1). Assume that the following conditions hold in the Frobenius norm $\|\cdot\| = \|\cdot\|_F$:*

1. *$\mathbf{F}$ is Lipschitz-continuous and bounded: for all $\mathbf{Y}, \widetilde{\mathbf{Y}} \in \mathbb{R}^{m \times n}$ and $0 \le t \le T$,*

$$\|\mathbf{F}(t, \mathbf{Y}) - \mathbf{F}(t, \widetilde{\mathbf{Y}})\| \le L \|\mathbf{Y} - \widetilde{\mathbf{Y}}\|, \qquad \|\mathbf{F}(t, \mathbf{Y})\| \le B \ .$$

2. *The non-tangential part of $\mathbf{F}(t, \mathbf{Y})$ is ε-small:*

$$\|(\mathbf{I} - \mathrm{P}(\mathbf{Y}))\mathbf{F}(t, \mathbf{Y})\| \le \varepsilon$$

   for all $\mathbf{Y} \in \mathcal{M}$ in a neighbourhood of $\mathbf{A}(t)$ and $0 \le t \le T$.
3. *The error in the initial value is δ-small:*

$$\|\mathbf{Y}_0 - \mathbf{A}_0\| \le \delta.$$

Let $\mathbf{Y}_n$ denote the rank-r approximation to $\mathbf{A}(t_n)$ at $t_n = nh$ obtained after n steps of the projector-splitting integrator with step-size $h > 0$. Then, the error satisfies for all n with $t_n = nh \leq T$

$$\|\mathbf{Y}_n - \mathbf{A}(t_n)\| \leq c_0\delta + c_1\varepsilon + c_2 h,$$

where the constants c_i only depend on L, B, and T. In particular, the constants are independent of singular values of the exact or approximate solution.

It is further shown in [10, Section 2.6.3] that an inexact solution of the matrix differential equations in the projector-splitting integrator leads to an additional error that is bounded in terms of the local errors in the inexact substeps, again with constants that do not depend on small singular values.

Numerical experiments with the matrix projector-splitting integrator and comparisons with standard numerical integrators are reported in [17, 10]. These experiments show good behaviour also for spatially discretized partial differential equations where the Lipschitz constant becomes large, a case that as of now is not covered by the theory.

3 Symmetric and skew-symmetric matrices: a structure-preserving integrator for dynamical low-rank approximation

We now assume that the right-hand side function in (1) is such that

$$\mathbf{F}(t, \mathbf{Y}) \text{ is (skew-)symmetric whenever } \mathbf{Y} \text{ is (skew-)symmetric.} \tag{5}$$

This condition ensures that the solutions to the matrix differential equation (1) and the projected differential equation (2) are (skew-)symmetric provided the initial values are (skew-)symmetric. For (2), this is seen from formula (4) for the tangent space projection with equal left and right factors $\mathbf{V} = \mathbf{U}$ in the decomposition (3) of the (skew-)symmetric rank-r matrix $\mathbf{Y} = \mathbf{U}\mathbf{S}\mathbf{U}^\top$.

While the projector-splitting integrator for dynamical low-rank approximation described in the previous section has favourable properties, it does *not* preserve symmetry or skew-symmetry of the solution $\mathbf{A}(t)$ to (1).

3.1 (Skew-)symmetry preserving integrator

We now propose a modified integrator that preserves symmetry and skew-symmetry and still retains the exactness and robustness properties of the projector-splitting integrator. A step with this integrator consists of two substeps. The first substep is identical to the first substep (**K**-step) of the projector-splitting integrator: it updates $\mathbf{K} = \mathbf{U}\mathbf{S}$ in the decomposition $\mathbf{Y} = \mathbf{U}\mathbf{S}\mathbf{U}^\top$. The second substep is a substantially modified update of $\mathbf{S}$, which can be viewed as a Galerkin approximation in the basis provided by the first substep.

Given $\mathbf{Y}_0 = \mathbf{U}_0\mathbf{S}_0\mathbf{U}_0^\top$ with a (skew-)symmetric $r \times r$-matrix $\mathbf{S}_0$ at time t_0, we compute the factorization $\mathbf{Y}_1 = \mathbf{U}_1\mathbf{S}_1\mathbf{U}_1^\top$ with a (skew-)symmetric $r \times r$-matrix $\mathbf{S}_1$ at time $t_1 = t_0 + h$ by the following algorithm:

Algorithm 1: One time step of the (skew-)symmetry preserving integrator

Data: $\mathbf{Y}^0 = \mathbf{U}_0\mathbf{S}_0\mathbf{U}_0^\top$ in factorized form, function $\mathbf{F}(t, \mathbf{Y})$, t_0, t_1
Result: $\mathbf{Y}_1 = \mathbf{U}_1\mathbf{S}_1\mathbf{U}_1^\top$ in factorized form

begin

Integrate from $t = t_0$ to t_1 the $n \times r$ matrix differential equation

$$\dot{\mathbf{K}}(t) = \mathbf{F}(t, \mathbf{K}(t)\mathbf{U}_0^\top)\mathbf{U}_0, \qquad \mathbf{K}(t_0) = \mathbf{U}_0\mathbf{S}_0.$$

Compute a QR-factorization $\mathbf{K}(t_1) = \mathbf{U}_1\mathbf{R}$.

Integrate from $t = t_0$ to t_1 the $r \times r$ differential equation

$$\dot{\mathbf{S}}(t) = \mathbf{U}_1^\top \mathbf{F}(t, \mathbf{U}_1\mathbf{S}(t)\mathbf{U}_1^\top)\mathbf{U}_1,$$
$$\mathbf{S}(t_0) = \mathbf{U}_1^\top \mathbf{Y}_0\mathbf{U}_1 = (\mathbf{U}_1^\top \mathbf{U}_0)\mathbf{S}_0(\mathbf{U}_1^\top \mathbf{U}_0)^\top.$$

Set $\mathbf{S}_1 = \mathbf{S}(t_1)$.

To continue in time, we take $\mathbf{Y}_1$ as starting value for the next step and perform another step of the integrator.

Note that in this integrator the factor $\mathbf{R}$ in the QR-decomposition of the first substep is not reused in the second substep, in contrast to the projector-splitting integrator. The computational cost is approximately halved, since the $\mathbf{L}$-step is not needed here.

We will now show that the (skew-)symmetric integrator retains the exactness and robustness properties of the projector-splitting integrator, using these known results in the proof.

3.2 Exactness property of the (skew-)symmetric integrator

The exactness result Theorem 1 extends in the following way.

Theorem 3 (Exactness property) *Let $\mathbf{A}(t) \in \mathbb{R}^{n\times n}$ be (skew-)symmetric and of rank r for $t_0 \le t \le t_1$, so that $\mathbf{A}(t)$ has a factorization (3) with equal left and right factors, $\mathbf{A}(t) = \mathbf{U}(t)\mathbf{S}(t)\mathbf{U}(t)^\top$. Moreover, assume that the $r\times r$ matrix $\mathbf{U}(t_1)^\top\mathbf{U}(t_0)$ is invertible. With $\mathbf{Y}_0 = \mathbf{A}(t_0)$, the (skew-)symmetric integrator for $\dot{\mathbf{Y}}(t) = \mathrm{P}(\mathbf{Y}(t))\dot{\mathbf{A}}(t)$ is then exact: $\mathbf{Y}_1 = \mathbf{A}(t_1)$.*

Proof We note that the projector-splitting integrator and the (skew-)symmetric integrator have the same first step. Let $\mathbf{U}_1 \in \mathbb{R}^{n\times r}$ be the matrix with orthonormal columns computed in the first substep. Due to the exactness of the matrix projector-splitting integrator as given by Theorem 1 we know that $\mathbf{U}_1$ and $\mathbf{A}(t_1)$ have the same range and therefore

$$\mathbf{U}_1\mathbf{U}_1^\top \mathbf{A}(t_1) = \mathbf{A}(t_1). \tag{6}$$

Denoting $\Delta\mathbf{A} = \mathbf{A}(t_1) - \mathbf{A}(t_0)$, the (skew-)symmetric integrator provides for the second substep the solution

$$\mathbf{S}_1 = \mathbf{U}_1^\top \mathbf{Y}_0\mathbf{U}_1 + \mathbf{U}_1^\top(\mathbf{A}(t_1) - \mathbf{A}(t_0))\mathbf{U}_1 = \mathbf{U}_1^\top \mathbf{A}(t_1)\mathbf{U}_1,$$

since $\mathbf{Y}_0 = \mathbf{A}(t_0)$. The result after a time step of the (skew-)symmetric integrator is

$$\mathbf{Y}_1 = \mathbf{U}_1\mathbf{S}_1\mathbf{U}_1^\top = \mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}(t_1)(\mathbf{U}_1\mathbf{U}_1^\top) = \mathbf{A}(t_1),$$

where the last equality holds because of (6) and the (skew-)symmetry of $\mathbf{A}(t_1)$. □

3.3 Robustness to small singular values

The error bound of Theorem 2 extends in the following way.

Theorem 4 (Robust error bound) *Let $\mathbf{A}(t)$ denote the (skew-)symmetric solution of the matrix differential equation (1) with $\mathbf{F}$ satisfying (5). Assume that conditions 1.-3. of Theorem 2 are fulfilled.*

Let $\mathbf{Y}_n$ denote the rank-r approximation to $\mathbf{A}(t_n)$ at $t_n = nh$ obtained after n steps of the (skew-)symmetric integrator of Algorithm 1 with step-size $h > 0$. Then, the error satisfies for all n with $t_n = nh \leq T$

$$\|\mathbf{Y}_n - \mathbf{A}(t_n)\| \leq c_0\delta + c_1\varepsilon + c_2 h,$$

where the constants c_i only depend on L, B, and T. In particular, the constants are independent of singular values of the exact or approximate solution.

As in [10, Section 2.6.3], it can be further shown that an inexact solution of the matrix differential equations in the projector-splitting integrator leads to an additional error that is bounded in terms of the local errors in the inexact substeps, again with constants that do not depend on small singular values.

Remark 1 The method of Algorithm 1 is of order 1, and higher order can be obtained simply by composition as, e.g., in [8, Section II.4]. However, like for the projector-splitting integrator of [17], it is not known if an error bound of higher order in the step-size h can be obtained with constants that are independent of small singular values. Numerical experiments with the Strang version of the projector-splitting integrator, which is of order 2, indicate an order reduction in some examples with very small singular values [21].

We now prepare for the proof of Theorem 4, which views the (skew-)symmetric integrator as a perturbed variant of the projector-splitting integrator.

Let us introduce the quantity

$$\vartheta(h,\varepsilon) := (4e^{Lh}BL + 9BL)h^2 + (3e^{Lh} + 4)\varepsilon h\ ,$$

which represents the local error bound after one time step of the projector-splitting integrator, as proved in [10, Theorem 2.1].

In the following, we denote by $\mathbf{U}_1 \in \mathbb{R}^{n\times r}$ the matrix with orthonormal columns obtained in the first substep of the integrator. We recall that the matrix projector-splitting and the (skew-)symmetric integrator have the first substep in common.

We denote by $\mathbf{A}_1$ the (skew-)symmetric solution at time t_1 of the full problem (1), where we consider the initial data to coincide with the (skew-)symmetric rank-r matrix $\mathbf{Y}_0$. For the local error analysis, the following lemma is needed.

Lemma 1 *Let $\mathbf{U}_1, \mathbf{A}_1$ be defined as above. The following estimate holds:*

$$\|\mathbf{U}_1\mathbf{U}_1^\top \mathbf{A}_1\mathbf{U}_1\mathbf{U}_1^\top - \mathbf{A}_1\| \le 2\vartheta(h,\varepsilon).$$

Proof The local error analysis in [10] shows that the $r \times n$ matrix $\mathbf{Z} = \mathbf{S}_1^{\mathrm{ps}}\mathbf{V}_1^{\mathrm{ps},\top}$, where $\mathbf{S}_1^{\mathrm{ps}}$ and $\mathbf{V}_1^{\mathrm{ps}}$ are the matrices computed in the third substep of the projector-splitting algorithm, satisfies

$$\|\mathbf{U}_1\mathbf{Z} - \mathbf{A}_1\| \le \vartheta := \vartheta(h,\varepsilon).$$

The square of the left-hand side can be split into two terms:

$$\begin{aligned}\|\mathbf{U}_1\mathbf{Z} - \mathbf{A}_1\|^2 &= \|\mathbf{U}_1\mathbf{Z} - \mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1 + \mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1 - \mathbf{A}_1\|^2 \\ &= \|\mathbf{U}_1\mathbf{U}_1^\top(\mathbf{U}_1\mathbf{Z} - \mathbf{A}_1) + (\mathbf{I} - \mathbf{U}_1\mathbf{U}_1^\top)\mathbf{A}_1\|^2 \\ &= \|\mathbf{U}_1\mathbf{U}_1^\top(\mathbf{U}_1\mathbf{Z} - \mathbf{A}_1)\|^2 + \|(\mathbf{I} - \mathbf{U}_1\mathbf{U}_1^\top)\mathbf{A}_1\|^2.\end{aligned}$$

Hence,

$$\|\mathbf{U}_1\mathbf{U}_1^\top(\mathbf{U}_1\mathbf{Z} - \mathbf{A}_1)\|^2 + \|(\mathbf{I} - \mathbf{U}_1\mathbf{U}_1^\top)\mathbf{A}_1\|^2 \le \vartheta^2.$$

From the second term it follows that

$$\|\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1 - \mathbf{A}_1\| \le \vartheta.$$

By the (skew-)symmetry of $\mathbf{A}_1$, this implies

$$\begin{aligned}\|\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1\mathbf{U}_1\mathbf{U}_1^\top - \mathbf{A}_1\| &= \|\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1\mathbf{U}_1\mathbf{U}_1^\top - \mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1 + \mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1 - \mathbf{A}_1\| \\ &\le \|\mathbf{U}_1\mathbf{U}_1^\top(\mathbf{A}_1\mathbf{U}_1\mathbf{U}_1^\top - \mathbf{A}_1)\| + \|\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1 - \mathbf{A}_1\| \\ &= \|\mathbf{U}_1\mathbf{U}_1^\top(\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1 - \mathbf{A}_1)^\top\| + \|\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1 - \mathbf{A}_1\| \\ &\le 2\|\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1 - \mathbf{A}_1\| \\ &\le 2\vartheta,\end{aligned}$$

which yields the result. □

In the following lemma, we show that the approximation given after one time step is $O(h(h+\varepsilon))$ close to the solution of system (1) when the starting values coincide.

Lemma 2 (Local Error) *The following local error bound holds:*

$$\|\mathbf{Y}_1 - \mathbf{A}_1\| \le h(\hat{c}_1\varepsilon + \hat{c}_2 h),$$

where the constants only depend on L and B and a bound of the step size. In particular, the constants are independent of singular values of the exact or approximate solution.

Proof By the identity $\mathbf{Y}_1 = \mathbf{U}_1\mathbf{S}_1\mathbf{U}_1^\top$ and Lemma 1 we have that

$$\begin{aligned}\|\mathbf{Y}_1 - \mathbf{A}_1\| &\le \|\mathbf{Y}_1 - \mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1\mathbf{U}_1\mathbf{U}_1^\top\| + \|\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1\mathbf{U}_1\mathbf{U}_1^\top - \mathbf{A}_1\| \\ &\le \|\mathbf{U}_1(\mathbf{S}_1 - \mathbf{U}_1^\top\mathbf{A}_1\mathbf{U}_1)\mathbf{U}_1^\top\| + 2\vartheta \\ &= \|\mathbf{S}_1 - \mathbf{U}_1^\top\mathbf{A}_1\mathbf{U}_1\| + 2\vartheta.\end{aligned}$$

The analysis of the local error thus reduces to estimating $\|\mathbf{S}_1 - \mathbf{U}_1^\top\mathbf{A}_1\mathbf{U}_1\|$. To this end, we introduce the following quantity: for $t_0 \le t \le t_1$,

$$\widetilde{\mathbf{S}}(t) := \mathbf{U}_1^\top\mathbf{A}(t)\mathbf{U}_1.$$

We observe that

$$\mathbf{A}(t) = \mathbf{U}_1\mathbf{U}_1^\top \mathbf{A}(t)\mathbf{U}_1\mathbf{U}_1^\top + \Big(\mathbf{A}(t) - \mathbf{U}_1\mathbf{U}_1^\top \mathbf{A}(t)\mathbf{U}_1\mathbf{U}_1^\top\Big) = \mathbf{U}_1\widetilde{\mathbf{S}}(t)\mathbf{U}_1^\top + \mathbf{R}(t),$$

where $\mathbf{R}(t)$ is defined as the term in big brackets. From Lemma 1 and from the bound B of $\mathbf{F}$, which yields for $t_0 \le t \le t_1$

$$\|\mathbf{A}(t) - \mathbf{A}(t_1)\| \le \int_{t_0}^{t_1} \|\dot{\mathbf{A}}(s)\|\, ds = \int_{t_0}^{t_1} \|\mathbf{F}(s, \mathbf{A}(s))\|\, ds \le Bh,$$

we conclude that the remainder term is bounded by

$$\|\mathbf{R}(t)\| \le \|\mathbf{R}(t) - \mathbf{R}(t_1)\| + \|\mathbf{R}(t_1)\| \le 2Bh + 2\vartheta.$$

This yields that $\mathbf{F}(t, \mathbf{A}(t))$ can be written as

$$\begin{aligned}\mathbf{F}(t, \mathbf{A}(t)) &= \mathbf{F}(t, \mathbf{U}_1\widetilde{\mathbf{S}}(t)\mathbf{U}_1^\top + \mathbf{R}(t)) \\ &= \mathbf{F}(t, \mathbf{U}_1\widetilde{\mathbf{S}}(t)\mathbf{U}_1^\top) + \mathbf{D}(t),\end{aligned}$$

where the defect

$$\mathbf{D}(t) := \mathbf{F}(t, \mathbf{U}_1\widetilde{\mathbf{S}}(t)\mathbf{U}_1^\top + \mathbf{R}(t)) - \mathbf{F}(t, \mathbf{U}_1\widetilde{\mathbf{S}}(t)\mathbf{U}_1^\top)$$

is bounded via the Lipschitz continuity of $\mathbf{F}$ as

$$\|\mathbf{D}(t)\| \le L\|\mathbf{R}(t)\| \le 2L(Bh + \vartheta).$$

We now compare the two differential equations

$$\begin{aligned}\dot{\widetilde{\mathbf{S}}}(t) &= \mathbf{U}_1^\top \mathbf{F}(t, \mathbf{U}_1\widetilde{\mathbf{S}}(t)\mathbf{U}_1^\top)\mathbf{U}_1 + \mathbf{U}_1^\top \mathbf{D}(t)\mathbf{U}_1, & \widetilde{\mathbf{S}}(t_0) &= \mathbf{U}_1^\top \mathbf{Y}_0\mathbf{U}_1, \\ \dot{\mathbf{S}}(t) &= \mathbf{U}_1^\top \mathbf{F}(t, \mathbf{U}_1\mathbf{S}(t)\mathbf{U}_1^\top)\mathbf{U}_1, & \mathbf{S}(t_0) &= \mathbf{U}_1^\top \mathbf{Y}_0\mathbf{U}_1.\end{aligned}$$

By construction, the solution of the first differential equation at time t_1 is $\widetilde{\mathbf{S}}(t_1) = \mathbf{U}_1^\top \mathbf{A}_1\mathbf{U}_1$. The solution of the second differential equation is $\mathbf{S}_1$ as given by the second substep of the (skew-)symmetric integrator. We now apply the Gronwall inequality to the previous system and obtain

$$\|\mathbf{S}_1 - \mathbf{U}_1^\top \mathbf{A}_1\mathbf{U}_1\| \le \int_{t_0}^{t_1} e^{L(t_1 - s)}\, \|\mathbf{D}(s)\|\, ds \le e^{Lh}\, 2L(Bh + \vartheta)h.$$

The result now follows using the definition of ϑ. □

Thanks to the Lipschitz continuity of the function $\mathbf{F}$, we conclude the proof of Theorem 4 from the local to the global errors by the standard argument of Lady Windermere's fan [9, Section II.3].

4 General tensors: recap of the projector-splitting integrator for the dynamical low-rank approximation by Tucker tensors

The objective is to approximate time-dependent tensors[1] $A(t) \in \mathbb{C}^{n_1 \times \cdots \times n_d}$ for $0 \leq t \leq T$ by tensors $Y(t)$ of multilinear rank $\mathbf{r} = (r_1, \ldots, r_d)$, with $r_i \ll n_i$. (We recall that r_i is the rank of the ith matricization $\mathbf{Mat}_i(Y) \in \mathbb{C}^{n_i \times n_i'}$ with $n_i' = \prod_{j \neq i} n_j$, which aligns all entries of Y with ith index k in the kth row. The retensorization is denoted by $Ten_i(\cdot)$, such that $Ten_i(\mathbf{Mat}_i(Y)) = Y$.)

The tensors $A(t)$ may be given explicitly or they are the unknown solution to a tensor differential equation (with right-hand side function $F : \mathbb{R} \times \mathbb{C}^{n_1 \times \cdots \times n_d} \to \mathbb{C}^{n_1 \times \cdots \times n_d}$)

$$\dot{A}(t) = F(t, A(t)), \qquad A(0) = A_0. \tag{7}$$

Dynamical low-rank approximation as presented in [13] determines $Y(t)$ as the solution of the projected matrix differential equation, with a projection $\mathrm{P}(Y)$ onto the tangent space $T_Y \mathcal{M}_{\mathbf{r}}$ of the manifold of tensors of multilinear rank $\mathbf{r}$ at $Y \in \mathcal{M}_{\mathbf{r}}$,

$$\dot{Y}(t) = \mathrm{P}(Y(t))F(t, Y(t)), \qquad Y(t_0) = Y_0, \tag{8}$$

where Y_0 is a rank-$\mathbf{r}$ approximation to A_0. (Here, $F(t, Y) = \dot{A}(t)$ if $A(t)$ is given explicitly.) Tensors $Y(t)$ of multilinear rank $\mathbf{r}$ are represented non-uniquely in the Tucker form [5] (using here the multilinear notation of [14])

$$Y(t) = C(t) \bigtimes_{i=1}^{d} \mathbf{U}_i(t), \tag{9}$$

where the core tensor $C(t) \in \mathbb{C}^{r_1 \times \cdots \times r_d}$ is of full multi-linear rank and the basis matrices $\mathbf{U}_i \in \mathbb{C}^{n \times r_i}$ have orthonormal columns. We choose the tangent space projection $\mathrm{P}(Y)$ as the orthogonal projection onto $T_Y(\mathcal{M}_r)$ with respect to the Euclidean inner product $\langle A, B \rangle = \mathbf{vec}(A)^* \mathbf{vec}(B)$, where $\mathbf{vec}(A)$ is a vectorization of A. Then, $\mathrm{P}(\mathbf{Y})$ is given as an alternating sum of $2d - 1$ subprojections [16], and like in the matrix case, a projector-splitting integrator with favourable properties can be formulated and efficiently implemented. The matrix projector-splitting integrator proposed in Section 2.1 has been successfully extended to the Tucker tensor format in different algorithmic versions in [16] and [18]. It is shown in [18, Section 6] that the proposed Tucker integrators are mathematically equivalent. The algorithm runs through the modes $i = 1, \ldots, d$ and solves differential equations for matrices of the dimension of the slim basis matrices and for the core tensor in alternation with orthogonal decompositions of slim matrices. We refer also to [11,3] for the formulation and implementation of this algorithm in the context of the MCTDH method [20] of molecular quantum dynamics in the chemical physics literature.

Moreover, the Tucker integrator has been proved in [18] to satisfy analogous properties to the matrix projector-splitting integrator: the exactness property and the robust convergence in the presence of small singular values of matricizations of the core tensor. We refer to [18, Theorems 4.1 and 5.1] for the precise formulation, which is very similar to the matrix case.

[1] In view of the applications in quantum dynamics, we here consider tensors with complex entries.

5 Symmetric and anti-symmetric tensors: a structure-preserving integrator for dynamical low-rank approximation

A tensor $A = (a_{i_1,\dots,i_d}) \in \mathbb{C}^{n\times\cdots\times n}$ is *symmetric* if for every permutation $\sigma \in S(d)$,

$$a_{i_{\sigma(1)},\dots,i_{\sigma(d)}} = a_{i_1,\dots,i_d},$$

and A is *anti-symmetric* if for every permutation $\sigma \in S(d)$,

$$a_{i_{\sigma(1)},\dots,i_{\sigma(d)}} = (-1)^{\mathrm{sign}(\sigma)} a_{i_1,\dots,i_d}.$$

It follows from [5] and [7] that a symmetric/anti-symmetric tensor $Y \in \mathbb{C}^{n\times\cdots\times n}$ of multi-linear rank $\mathbf{r} = (r,\dots,r)$ admits a Tucker decomposition

$$Y = C \bigtimes_{i=1}^{d} \mathbf{U},$$

where the core tensor $C \in \mathbb{C}^{r\times\cdots\times r}$ is symmetric/anti-symmetric of full rank $\mathbf{r}$ and the basis matrix $\mathbf{U} \in \mathbb{C}^{n\times r}$ is the same for all indices.

We assume that the right-hand side function in (7) is such that

$$F(t,Y) \text{ is (anti-)symmetric whenever } Y \text{ is (anti-)symmetric.} \tag{10}$$

Like (5) in the matrix case, this ensures that the solutions to the tensor differential equation (7) and the projected differential equation (8) are (anti-)symmetric provided the initial tensors are (anti-)symmetric. As we noted already in the matrix case, the projector-splitting Tucker integrator does not preserve (anti-)symmetry.

5.1 (Anti-)symmetry preserving Tucker integrator

The numerical integrator defined in Section 3 for the matrix case extends in a natural way to the Tucker tensor format. The first substep, which updates the basis matrix $\mathbf{U}$, is identical to the first substep of the general Tucker integrator in [18,16]. The second substep is a Galerkin method with the updated basis and determines the updated (anti-)symmetric core tensor.

Given the (anti-)symmetric tensor $Y_0 = C_0 \bigtimes_{i=1}^{d} \mathbf{U}_0$, we compute the (anti-)symmetric approximation $Y_1 = C_1 \bigtimes_{i=1}^{d} \mathbf{U}_1$ at time $t_1 = t_0 + h$ as follows:

Algorithm 2: One time step of the (anti-)symmetry preserving Tucker integrator

Data: Tucker tensor $Y_0 = C_0 \bigtimes_{i=1}^{d} \mathbf{U}_0$, $F(t, Y)$, t_0, t_1
Result: Tucker tensor $Y_1 = C_1 \bigtimes_{i=1}^{d} \mathbf{U}_1$

begin

Matricize the core tensor C_0 in the first mode.

Perform a QR-factorization:

$$\mathbf{Mat}_1(C_0)^\top = \mathbf{Q}_0 \mathbf{S}_0^\top,$$

where $\mathbf{Q}_0 \in \mathbb{C}^{r^{d-1} \times r}$ has orthonormal columns. Define

$$\mathbf{V}_0^\top = \mathbf{Q}_0^\top \bigotimes_{i=2}^{d} \mathbf{U}_0^\top .$$

Integrate from $t = t_0$ to t_1 the $n \times r$ matrix differential equation

$$\dot{\mathbf{K}}(t) = \mathbf{Mat}_1(F(t, Ten_1(\mathbf{K}(t)\mathbf{V}_0^\top)))\overline{\mathbf{V}}_0, \qquad \mathbf{K}(t_0) = \mathbf{U}_0 \mathbf{S}_0.$$

Compute the QR-factorization $\mathbf{K}(t_1) = \mathbf{U}_1 \mathbf{R}$.

Integrate from $t = t_0$ to t_1 the $r \times \cdots \times r$ (d times) tensor equation

$$\dot{C}(t) = F\left(t, C(t) \bigtimes_{i=1}^{d} \mathbf{U}_1\right) \bigtimes_{i=1}^{d} \mathbf{U}_1^*,$$
$$C(t_0) = Y_0 \bigtimes_{i=1}^{d} \mathbf{U}_1^* = C_0 \bigtimes_{i=1}^{d} (\mathbf{U}_1^* \mathbf{U}_0).$$

Set $C_1 = C(t_1)$.

To continue, we take Y_1 as the starting value for the next step.

5.2 Exactness property of the (anti-)symmetric Tucker integrator

The following result extends the exactness results of Theorem 3 and [18, Theorem 4.1] to (anti-)symmetric tensors.

Theorem 5 (Exactness property) *Let $A(t) \in \mathbb{C}^{n \times \cdots \times n}$ be (anti-)symmetric and of multilinear rank $(r, \ldots, r)$ for $t_0 \le t \le t_1$, so that $A(t) = C(t) \bigtimes_{i=1}^{d} \mathbf{U}(t)$, where the $n \times r$ basis matrix $\mathbf{U}$ has orthonormal columns. Moreover, assume that the $r \times r$ matrix $\mathbf{U}(t_1)^* \mathbf{U}(t_0)$ is invertible. With $Y_0 = A(t_0)$, the (anti-)symmetric Tucker integrator for $\dot{Y}(t) = \mathrm{P}(Y(t))\dot{A}(t)$ is then exact: $Y_1 = A(t_1)$.*

Proof The projector-splitting Tucker integrator and the (anti-)symmetric integrator have the same first substep. Let $\mathbf{U}_1 \in \mathbb{R}^{n \times r}$ be the basis matrix with orthonormal columns computed in the first substep. Due to the exactness of the projector-splitting

Tucker integrator as shown by [18, Theorem 4.1] we have that $A(t_1)$ has the (anti)-symmetric Tucker representation

$$A(t_1) = \widehat{C}_1 \bigtimes_{i=1}^{d} \mathbf{U}_1$$

for some (anti-)symmetric core tensor $\widehat{C}_1 \in \mathbb{C}^{r\times\cdots\times r}$. Using the rule $A \times_i \mathbf{V} \times_i \mathbf{W} = A \times_i (\mathbf{WV})$, this implies that

$$A(t_1) \times_i (\mathbf{U}_1\mathbf{U}_1^*) = A(t_1), \qquad i = 1, \dots, d\ .$$

With $Y_0 = A(t_0)$ we obtain from the second substep of the algorithm

$$\begin{aligned} Y_1 = C_1 \bigtimes_{i=1}^{d} \mathbf{U}_1 &= \Big(Y_0 \bigtimes_{i=1}^{d} \mathbf{U}_1^* + (A(t_1) - A(t_0)) \bigtimes_{i=1}^{d} \mathbf{U}_1^*\Big) \bigtimes_{i=1}^{d} \mathbf{U}_1 \\ &= \Big(A(t_1) \bigtimes_{i=1}^{d} \mathbf{U}_1^*\Big) \bigtimes_{i=1}^{d} \mathbf{U}_1 = A(t_1) \bigtimes_{i=1}^{d} (\mathbf{U}_1\mathbf{U}_1^*) = A(t_1), \end{aligned}$$

which proves the exactness. □

5.3 Robustness to small singular values

The robust error bounds from Theorem 4 and [18, Theorem 5.1] extend to the (anti)-symmetric Tucker integrator as follows. The norm $\|B\|$ of a tensor B used here is the Euclidean norm of the entries of B.

Theorem 6 (Robust error bound) *Let $A(t)$ denote the (anti-)symmetric solution of the tensor differential equation (7) with F satisfying (10). Assume the following:*

1. *F is Lipschitz-continuous and bounded.*
2. *The non-tangential part of $F(t, Y)$ is ε-small:*

$$\|(I - \mathrm{P}(Y))F(t, Y)\| \le \varepsilon$$

   for all Y of multilinear rank $(r, \dots, r)$ in a neighbourhood of $A(t)$ and $0 \le t \le T$.
3. *The error in the initial value is δ-small:*

$$\|Y_0 - A_0\| \le \delta.$$

Let Y_n denote the (anti-)symmetric approximation of multinear rank $(r, \dots, r)$ to $A(t_n)$ at $t_n = nh$ obtained after n steps of the (anti-)symmetric Tucker integrator with step-size $h > 0$. Then, the error satisfies for all n with $t_n = nh \le T$

$$\|Y_n - A(t_n)\| \le c_0\delta + c_1\varepsilon + c_2h,$$

where the constants c_i only depend on the Lipschitz constant L and bound B of F, on T, and on the dimension d. In particular, the constants are independent of singular values of matricizations of the exact or approximate solution.

It can be further shown that an inexact solution of the matrix differential equations in the integrator leads to an additional error that is bounded in terms of the local errors in the inexact substeps, again with constants that do not depend on small singular values.

The proof of Theorem 6 proceeds similar to the proof of Theorem 4 for the (skew)-symmetric matrix case. We begin with a key lemma and then analyze the local error produced after one time step, comparing the numerical solution with the exact solution that starts from the same initial value $A_0 = Y_0$. We denote the value of this solution at t_1 by A_1. The basis matrix computed in the first substep of the integrator is denoted by $\mathbf{U}_1$.

Lemma 3 *The following estimate holds:*

$$||A_1 \bigtimes_{i=1}^{d} \mathbf{U}_1\mathbf{U}_1^* - A_1|| \leq c\, h(BLh + \varepsilon),$$

where c only depends on d and a bound for hL.

Proof The error bound of [18, Theorem 5.1] shows that there exists $Z \in \mathbb{C}^{r\times n\times\cdots\times n}$ such that

$$\|Z \times_1 \mathbf{U}_1 - A_1\| \leq c_* h(BLh + \varepsilon) =: \vartheta.$$

We observe that

$$\|Z \times_1 \mathbf{U}_1 - A_1\| = \|\mathbf{Mat}_1(Z \times_1 \mathbf{U}_1 - A_1)\| = \|\mathbf{U}_1\mathbf{Mat}_1(Z) - \mathbf{Mat}_1(A_1)\|.$$

As in the matrix case we obtain

$$\|\mathbf{U}_1\mathbf{U}_1^*\mathbf{Mat}_1(A_1) - \mathbf{Mat}_1(A_1)\| \leq \vartheta.$$

Thanks to (anti-)symmetry we have

$$\|\mathbf{U}_1\mathbf{U}_1^*\mathbf{Mat}_1(A_1) - \mathbf{Mat}_1(A_1)\| = \|\mathbf{U}_1\mathbf{U}_1^*\mathbf{Mat}_i(A_1) - \mathbf{Mat}_i(A_1)\|, \quad i = 1,\dots,d,$$

which yields

$$\|A_1 \times_i \mathbf{U}_1\mathbf{U}_1^* - A_1\| \leq \vartheta, \qquad i = 1,\dots,d.$$

To conclude, we observe

$$\begin{aligned}
&\|A_1 \bigtimes_{i=1}^{d} \mathbf{U}_1\mathbf{U}_1^* - A_1\| \\
&\leq \|A_1 \bigtimes_{i=1}^{d} \mathbf{U}_1\mathbf{U}_1^* - A_1 \bigtimes_{i=1}^{d-1} \mathbf{U}_1\mathbf{U}_1^* + A_1 \bigtimes_{i=1}^{d-1} \mathbf{U}_1\mathbf{U}_1^* - A_1\| \\
&\leq \|A_1 \bigtimes_{i=1}^{d} \mathbf{U}_1\mathbf{U}_1^* - A_1 \bigtimes_{i=1}^{d-1} \mathbf{U}_1\mathbf{U}_1^*\| + \|A_1 \bigtimes_{i=1}^{d-1} \mathbf{U}_1\mathbf{U}_1^* - A_1\| \\
&\leq \|(A_1 \times_d \mathbf{U}_1\mathbf{U}_1^* - A_1) \bigtimes_{i=1}^{d-1} \mathbf{U}_1\mathbf{U}_1^*\| + \|A_1 \bigtimes_{i=1}^{d-1} \mathbf{U}_1\mathbf{U}_1^* - A_1\| \\
&\leq \|A_1 \times_d \mathbf{U}_1\mathbf{U}_1^* - A_1\| + \|A_1 \bigtimes_{i=1}^{d-1} \mathbf{U}_1\mathbf{U}_1^* - A_1\| \\
&\leq \vartheta + \|A_1 \bigtimes_{i=1}^{d-1} \mathbf{U}_1\mathbf{U}_1^* - A_1\|,
\end{aligned}$$

and the result follows by an iteration of this argument. □

We are now in the position to analyse the local error produced after one time step of the integrator.

Lemma 4 (Local error) *The error of the (anti-)symmetric Tucker integrator after one time step satisfies*

$$\|Y_1 - A_1\| \leq \hat{c}\, h(BLh + \varepsilon),$$

where $\hat{c}$ only depends on d and a bound of hL. In particular, the constant is independent of singular values of the exact or approximate solution.

Proof By construction of the algorithm and by Lemma 3 we have

$$\begin{aligned}
\|Y_1 - A_1\| &\leq \|Y_1 - A_1 \mathsf{X}_{i=1}^{d} \mathbf{U}_1\mathbf{U}_1^*\| + \|A_1 \mathsf{X}_{i=1}^{d} \mathbf{U}_1\mathbf{U}_1^* - A_1\| \\
&\leq \|C_1 \mathsf{X}_{i=1}^{d} \mathbf{U}_1 - A_1 \mathsf{X}_{i=1}^{d} \mathbf{U}_1\mathbf{U}_1^*\| + c\vartheta \\
&\leq \|C_1 \mathsf{X}_{i=1}^{d} \mathbf{U}_1 - (A_1 \mathsf{X}_{i=1}^{d} \mathbf{U}_1^*) \mathsf{X}_{i=1}^{d} \mathbf{U}_1\| + c\vartheta \\
&\leq \|(C_1 - A_1 \mathsf{X}_{i=1}^{d} \mathbf{U}_1^*) \mathsf{X}_{i=1}^{d} \mathbf{U}_1\| + c\vartheta \\
&\leq \|C_1 - A_1 \mathsf{X}_{i=1}^{d} \mathbf{U}_1^*\| + c\vartheta.
\end{aligned}$$

The problem reduces to estimating $\|C_1 - A_1 \mathsf{X}_{i=1}^{d} \mathbf{U}_1^*\|$. We introduce the tensor

$$\widetilde{C}(t) := A(t) \mathsf{X}_{i=1}^{d} \mathbf{U}_1^*,$$

which satisfies

$$\dot{\widetilde{C}}(t) = F(t, A(t)) \mathsf{X}_{i=1}^{d} \mathbf{U}_1^*, \qquad \widetilde{C}(t_0) = Y_0 \mathsf{X}_{i=1}^{d} \mathbf{U}_1^*.$$

In the same way as in the proof of Lemma 2 (replacing $\mathbf{S}$ by C) this is compared with the differential equation for $C(t)$ in the second substep of the integrator. This yields the stated result. □

Using the Lipschitz continuity of the function F, we conclude the proof of Theorem 6 from the local to the global errors by the standard argument of Lady Windermere's fan [9, Section II.3].

6 Numerical Experiments

In this section we show results of various numerical experiments. The computations were done using Matlab R2017a software with Tensor Toolbox package v2.6 [2] and TensorLab package v3.0 [23].

6.1 Addition of symmetric tensors: a computationally inexpensive retraction

Let $A \in \mathbb{C}^{n\times\cdots\times n}$ be a symmetric tensor of multi-linear rank $\mathbf{r} = (r, \ldots, r)$ and let $B \in \mathbb{C}^{n\times\cdots\times n}$. We consider the addition of two given tensors,

$$C = A + B,$$

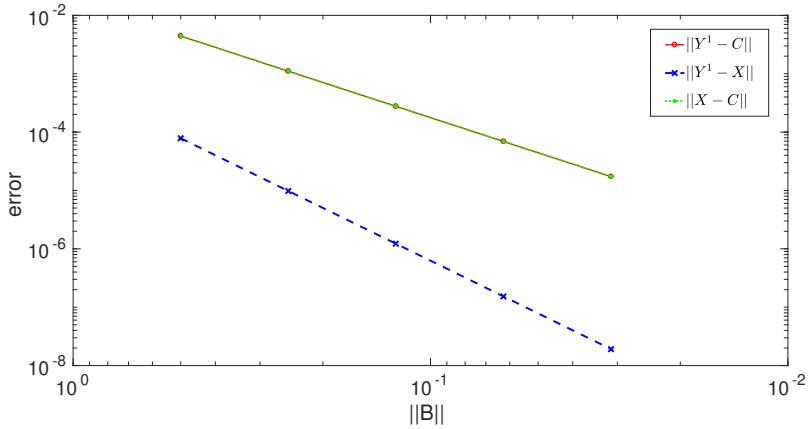
where $C \in \mathbb{C}^{n\times\cdots\times n}$ is not necessarily of low rank and we want to compute a symmetric rank-$(r, \ldots, r)$ approximation. Such a retraction is typically required in optimization problems on low-rank manifolds and needs to be computed in each iterative step of a

descent algorithm. The approach considered here consists of reformulating the addition problem as the solution of the following differential equation at time $t = 1$:

$$\dot{C}(t) = B, \quad C(0) = A.$$

We will compare the solution obtained by computing the full addition and retracting to the manifold of symmetric tensors of multilinear rank $(r, \dots, r)$ with the one obtained from the application of the symmetric low-rank tensor integrator. The advantage of the latter method is that the approximation is built inside the manifold, so that no truncation to rank r is needed.

For our numerical example, we initialize A as a symmetric random Tucker tensor of size $100 \times 100 \times 100$ and multi-linear rank $\mathbf{r} = (10, 10, 10)$; we take B as an element in the tangent space of the symmetric rank-$\mathbf{r}$ tensor manifold at A. We compare the dynamical low rank approximation Y_1 generated by the algorithm introduced in Section 5 with a low rank symmetric retraction [6,22] of the full solution denoted by X. For the last part, we use the built-in `tucker_als` and `tucker_sym` functions of the Tensor Toolbox Package.



We observe that the approximation Y_1 shows the correct behavior, at reduced computational cost. Decreasing the norm of the tensor B decreases the approximation error as expected, proportional to $\|B\|^2$.

6.2 Robustness with respect to small singular values

We present two numerical examples and show robustness of the proposed symmetric integrator in the presence of small singular values. For the sake of presentation we consider the matrix case. Analogous examples can be implemented for Tucker tensors, and similar results are obtained.

In the first example, the time-dependent matrix is given explicitly as

$$\mathbf{A}(t) = \left(e^{t\mathbf{W}}\right) e^{t} \mathbf{D} \left(e^{t\mathbf{W}}\right)^{\top}, \quad 0 \le t \le 1 \ .$$

The matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ is diagonal with elements $d_j = 2^{-j}$ and the matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ is skew-symmetric and randomly generated. We choose $N = 100$ and final time $T = 1$.

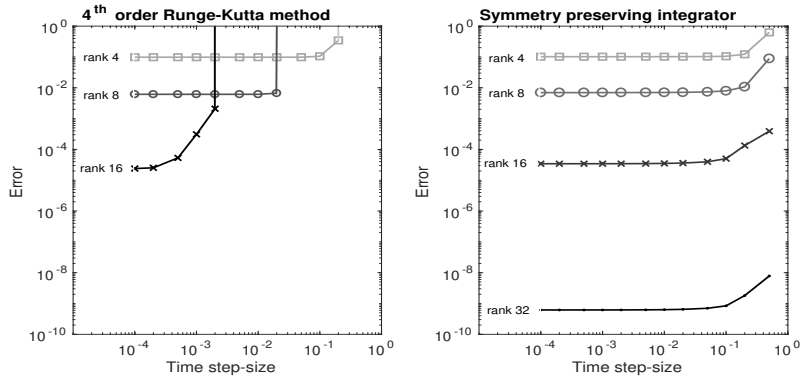


Fig. 1 Comparison of the explicit Runge Kutta method (left) and the proposed symmetry-preserving integrator (right) for different approximation ranks and step sizes in the case of a given time-dependent symmetric matrix.

We compare the symmetric low-rank integrator presented in Section 3 with a numerical solution obtained with a 4-th order explicit Runge-Kutta method applied to the system of differential equations for dynamical low-rank approximation as derived in [12].

The numerical results for different ranks are shown in Figure 1. In contrast to the Runge–Kutta method, the proposed symmetric low-rank integrator does not suffer of a step-size restriction in the presence of small singular values.

In the second example, we integrate the Lyapunov matrix differential equation (cf. [19])

$$\dot{\mathbf{X}}(t) = \mathbf{A}\mathbf{X}(t) + \mathbf{X}(t)\mathbf{A}^\top + \mathbf{Q}, \quad \mathbf{X}(0) = \mathbf{U}_0\mathbf{S}_0\mathbf{U}_0^\top \ .$$

Here, we choose $\mathbf{A} = \texttt{tridiag}(-1, 2, -1) \otimes \mathbf{I} + \mathbf{I} \otimes \texttt{tridiag}(-1, 2, -1) \in \mathbb{R}^{N\times N}$ as a discrete Laplacian. The positive definite matrix $\mathbf{Q} \in \mathbb{R}^{N\times N}$ has rank 5 and is randomly generated. The orthonormal matrix $\mathbf{U}_0 \in \mathbb{R}^{N\times N}$ is randomly generated and $\mathbf{S}_0 \in \mathbb{R}^{N\times N}$ is of rank 1 with only one non-zero element, $s_{11} = 1$.

The reference solution and the linear subproblems appearing in the definition of the symmetric low-rank integrator have been solved with the Matlab solver `ode45` and stringent tolerance parameters {'RelTol', 1e-10, 'AbsTol', 1e-14} . We choose $N = 100$ and final time $T = 0.1$. The singular values of the reference solution and the absolute errors $\|Y_n - A(t_n)\|_F$ at time $t_n = T$ of the approximate solutions for different ranks calculated with different step-sizes are shown in Figure 2.

6.3 Ground state of a fermionic multi-particle system

A natural field of application of the (anti-)symmetric low-rank algorithms is in quantum dynamics of systems of fermions or bosons, which is described by anti-symmetric or symmetric multivariate wave functions, respectively. In the MCTDHF and MCTDHB methods [1,4], the approximate wave function is sought for in the form of a low-rank Tucker tensor that is anti-symmetric and symmetric, respectively. It approximates the huge tensor of coefficients with respect to some fixed spatial basis, such as a Fourier basis. The (anti-)symmetric low-rank time integrator proposed in this paper appears ideally suited as a numerical integrator for such problems.

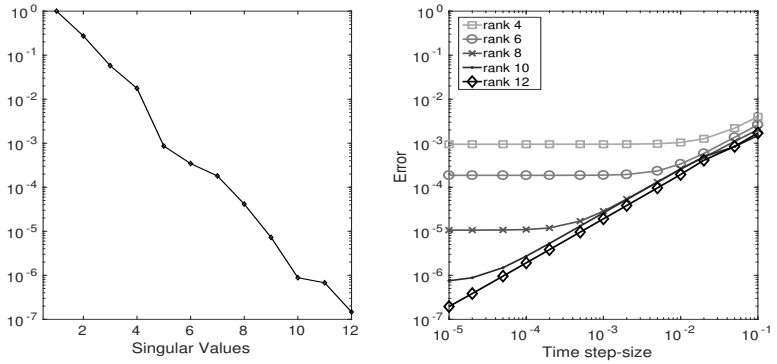


Fig. 2 First twelve singular values of the reference solution at time $T = 0.1$ and approximation errors for different ranks at different step-sizes for the Lyapunov matrix differential equation.

As a first illustration of the approach, we apply the method for the calculation of the ground state of a system of d fermions in 1 space dimension. We calculate the ground state as the solution for large time of the imaginary-time Schrödinger equation

$$\partial_t \psi = -\mathcal{H}\psi, \quad \psi(t_0) = \psi_0. \tag{11}$$

We consider the Hamiltonian given by

$$\mathcal{H} := \sum_{l=1}^{d} \Big(-\tfrac{1}{2}\partial_l^2 + V(x_l) - \sum_{k=l+1}^{d} V(x_l - x_k) \Big),$$

where $x_l \in \mathbb{R}$ represents the position of the l-th particle and we choose the torsion potential

$$V(x) = 1 - \cos(x) \ .$$

Choosing a collocation method with a tensor Fourier basis set (with K basis functions per particle) for approximating the anti-symmetric wave function leads to a huge tensor differential equation (7), where a low-rank approximation to the anti-symmetric d-dimensional tensor $Y(t) \in \mathbb{C}^{K\times\cdots\times K}$ is to be computed. This is done with a variational splitting method. The stiffness introduced by the Laplacian will be handled with a split-step Fourier method [15] while the two-particle interaction is treated with the anti-symmetric low-rank integrator of Section 5.
We introduce the space discretization

$$x_j = \frac{2\pi j}{K}, \quad j = -K/2, \ldots, K/2 - 1 \ .$$

Let $Y(t)$ be a time-dependent tensor defined element-wise by

$$Y_{j_1,j_2,\ldots,j_d}(t) = \psi(t, x_{j_1}, \ldots, x_{j_d}) \ .$$

The fermionic property of the system implies that the tensor $Y(t)$ is anti-symmetric. Denoting $\mathcal{F}_K$ the Fourier matrix, we define

$$\begin{aligned} \mathbf{D} &:= \mathcal{F}_K^{-1} \mathrm{diag}\{\tfrac{1}{2} j^2\} \mathcal{F}_K, \\ \mathbf{V}_{\cos} &:= \mathrm{diag}\{\cos(x_j)\}, \\ \mathbf{V}_{\sin} &:= \mathrm{diag}\{\sin(x_j)\} \ . \end{aligned}$$

The Fourier collocation space discretization of (11) is equivalent to the system

$$\dot{Y} = -H[Y], \qquad Y(t_0) = C_0 \bigtimes_{i=1}^{d} \mathbf{U}_0. \tag{12}$$

Using the trigonometric equality $\cos(x-y) = \cos(x)\cos(y) + \sin(x)\sin(y)$, the linear operator H can be written in a multi-linear product form as

$$H[Y] = \frac{3d-d^2}{2} Y + \sum_{l=1}^{d} Y \times_l \mathbf{D} - Y \times_l \mathbf{V}_{\cos} + \sum_{k=l+1}^{d} Y \times_l \mathbf{V}_{\cos} \times_k \mathbf{V}_{\cos} + Y \times_l \mathbf{V}_{\sin} \times_k \mathbf{V}_{\sin} \, .$$

In order to remove the stiffness introduced by the Laplacian, we split (12) in $(d+2)$ sub-problems. The solution of the first sub-problem at time $t_1 = t_0 + h$ is obtained updating the core tensor C_0 in the initial data,

$$\widetilde{C}_0 = \exp\Big(-h \frac{3d-d^2}{2} \Big) C_0 \, .$$

Afterwards, we start considering the equation

$$\dot{Y}_I = -Y_I \times_1 \mathbf{D} + Y_I \times_1 \mathbf{V}_{\cos}, \quad Y_I(t_0) = \widetilde{C}_0 \times \mathbf{U}_0 \, .$$

We matricize in the first mode,

$$\mathbf{Mat}_1(\dot{Y}_I) = -\mathbf{D}\, \mathbf{Mat}_1(Y_I) + \mathbf{V}_{\cos} \mathbf{Mat}_1(Y_I)$$

with initial data,

$$\mathbf{Mat}_1(Y_I(t_0)) = \mathbf{Mat}_1(Y_0) = \mathbf{U}_0 \mathbf{Mat}_1(\widetilde{C}_0 \bigtimes_{i=2}^{d} \mathbf{U}_0) \, .$$

The solution can now be computed with the 1-dimensional split-step Fourier method. Denoting

$$\widetilde{\mathbf{U}}_0 = e^{+\frac{h}{2}\mathbf{V}_{\cos}} \mathcal{F}_K^{-1} e^{-h\mathbf{T}} \mathcal{F}_K e^{+\frac{h}{2}\mathbf{V}_{\cos}} \mathbf{U}_0 \, , \quad \mathbf{T} = \mathcal{F}_K \mathbf{D} \mathcal{F}_K^{-1}$$

and tensorizing back in the first mode we have that

$$Y_I(t_1) = \widetilde{C}_0 \times_1 \widetilde{\mathbf{U}}_0 \bigtimes_{i=2}^{d} \mathbf{U}_0 \, .$$

Taking this as initial condition and iterating the same process for all the successive modes we obtain the updated anti-symmetric tensor

$$X_0 = \widetilde{C}_0 \bigtimes_{i=1}^{d} \widetilde{\mathbf{U}}_0 \, .$$

We now apply the anti-symmetric low-rank integrator to the multi-particle interaction,

$$\dot{X} = -W[X], \qquad X(t_0) = X_0,$$

where

$$W[Y] = \sum_{l=1}^{d} \sum_{k=l+1}^{d} Y \times_l \mathbf{V}_{\cos} \times_k \mathbf{V}_{\cos} + Y \times_l \mathbf{V}_{\sin} \times_k \mathbf{V}_{\sin} \, .$$

The core C_0 is renormalized after each step, since the absolute size of the tensor is irrelevant. We emphasize the fact that all along the implementation, it is crucial to use the structure of the Tucker tensor and avoid to build the huge matrix $\mathbf{V}_0$ appearing

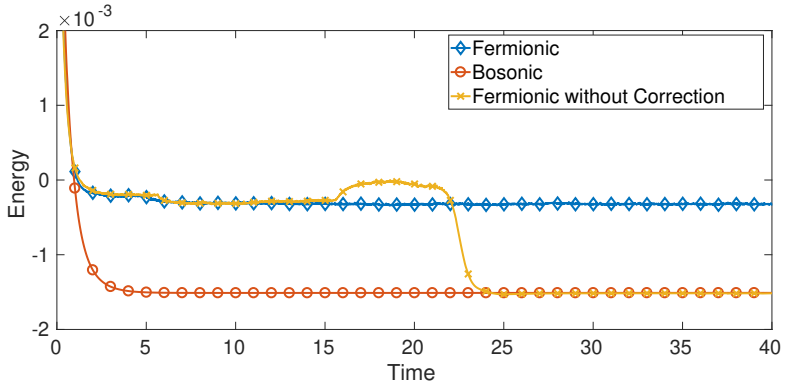


Fig. 3 Evolution to the fermionic ground state energy computed with rank 5.

in the definition of the integrator. The $\mathbf{K}$ and C problems are linear and can be solved with few iterations of the Arnoldi process.

In our numerical experiment we choose $d = 3$ particles in 1 space dimension and fix the number of Fourier basis functions per particle at $K = 128$, the step-size at $h = 0.01$ and we propagate the system until $T = 40$.

We introduce the discrete energy

$$E(Y) = \left(\frac{2\pi}{K}\right)^d \langle Y, H[Y]\rangle_F \,.$$

Although the integrator preserves the anti-symmetry in theory, in a straightforward implementation round-off errors will destroy the anti-symmetry and take the system to the lowest state of energy: the bosonic ground state - the one achieved starting from a symmetric initial value. This behavior can be corrected in the integrator by enforcing the anti-symmetry of the small core tensor (which is violated only by round-off errors) at each step or ever after a few steps. In this way the computation tends to the fermionic ground state. The energy levels generated by the approximation of rank 5 are shown in Figure 3 for the bosonic system and the fermionic system with and without enforced anti-symmetrization.

6.4 MCTDHF - Ultra fast laser dynamics

In the second example we consider the situation where an external pulsing laser field is introduced in the system. We refer to [4] for the physical description of the problem and its MCTDHF formulation. We consider the time-dependent Schrödinger equation

$$i\partial_t \psi = \mathcal{H}(t)\psi, \quad \psi(t_0) = \psi_0,$$

with the Hamiltonian

$$\mathcal{H}(t) := \sum_{l=1}^{d} \frac{1}{2}\left[\frac{1}{i}\partial_l - \omega(t)\right]^2 + V(x_l) - \sum_{k=l+1}^{d} V(x_l - x_k).$$

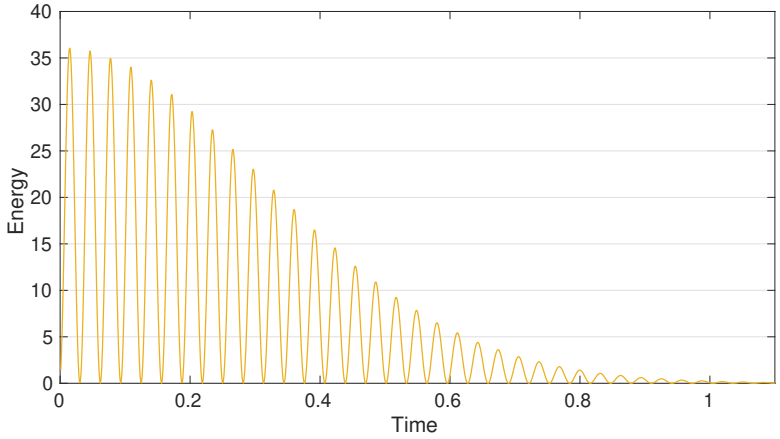


Fig. 4 Energy evolution computed with rank 5.

with the torsion potential V as before and with parameters

$$\omega(t) := A_0 e^{-t^2/\tau^2} \sin(\Omega t), \quad A_0 = 100, \quad \Omega = 100, \quad \tau = 0.2\pi \ .$$

As initial value, we choose the ground-state calculated at the previous step. We fix the number of Fourier basis functions per particle at $K = 128$, the step-size at $h = 0.005$ and we propagate the system until $T = 1$ by the same algorithm as in the previous subsection, but this time for the real-time evolution instead of the imaginary-time evolution.

The time evolution of the energy obtained by the approximation of rank 5 is shown in Figure 4.

Acknowledgements We thank Balázs Kovács and Hanna Walach for their constructive comments and suggestions. This work was supported by Deutsche Forschungsgemeinschaft, Graduiertenkolleg 1838 "Spectral Theory and Dynamics of Quantum Systems".

References

1. O. E. Alon, A. I. Streltsov, and L. S. Cederbaum. Multiconfigurational time-dependent Hartree method for bosons: many-body dynamics of bosonic systems. *Phys. Rev. A*, 77:033613, Mar 2008.
2. B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.6. Available online, February 2015.
3. M. Bonfanti and I. Burghardt. Tangent space formulation of the multi-configuration time-dependent Hartree equations of motion: the projector-splitting algorithm revisited. *Chemical Physics*, 515:252 – 261, 2018.
4. J. Caillat, J. Zanghellini, M. Kitzler, O. Koch, W. Kreuzer, and A. Scrinzi. Correlated multielectron systems in strong laser fields: a multiconfiguration time-dependent Hartree-Fock approach. *Phys. Rev. A*, 71:012712, Jan 2005.
5. L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.
6. L. De Lathauwer, B. De Moor, and J. Vandewalle. On the best rank-1 and rank-$(R_1, R_2, \cdots, R_N)$ approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000.

7. W. Hackbusch. On the representation of symmetric and antisymmetric tensors. In *Contemporary computational mathematics—a celebration of the 80th birthday of Ian Sloan. Vol. 1, 2*, pages 483–515. Springer, Cham, 2018.
8. E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration. Structure-preserving algorithms for ordinary differential equations*, volume 31 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 2006.
9. E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I. Nonstiff problems*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1993.
10. E. Kieri, C. Lubich, and H. Walach. Discretized dynamical low-rank approximation in the presence of small singular values. *SIAM J. Numer. Anal.*, 54(2):1020–1038, 2016.
11. B. Kloss, I. Burghardt, and C. Lubich. Implementation of a novel projector-splitting integrator for the multi-configurational time-dependent Hartree approach. *J. Chem. Phys.*, 146(17):174107, 2017.
12. O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM J. Matrix Anal. Appl.*, 29(2):434–454, 2007.
13. O. Koch and C. Lubich. Dynamical tensor approximation. *SIAM J. Matrix Anal. Appl.*, 31(5):2360–2375, 2010.
14. T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, 2009.
15. C. Lubich. *From quantum to classical molecular dynamics: reduced models and numerical analysis*. Zurich Lectures in Advanced Mathematics. European Mathematical Society (EMS), Zürich, 2008.
16. C. Lubich. Time integration in the multiconfiguration time-dependent Hartree method of molecular quantum dynamics. *Appl. Math. Res. Express. AMRX*, (2):311–328, 2015.
17. C. Lubich and I. V. Oseledets. A projector-splitting integrator for dynamical low-rank approximation. *BIT*, 54(1):171–188, 2014.
18. C. Lubich, B. Vandereycken, and H. Walach. Time integration of rank-constrained Tucker tensors. *SIAM J. Numer. Anal.*, 56(3):1273–1290, 2018.
19. H. Mena, A. Ostermann, L. M. Pfurtscheller, and C. Piazzola. Numerical low-rank approximation of matrix differential equations. *J. Comput. Appl. Math.*, 340:602–614, 2018.
20. H.-D. Meyer, F. Gatti, and G. A. Worth. *Multidimensional quantum dynamics: MCTDH theory and applications*. John Wiley & Sons, 2009.
21. A. Ostermann, C. Piazzola, and H. Walach. Convergence of a low-rank lie–trotter splitting for stiff matrix differential equations. *arXiv:1803.10473*, 2018.
22. P. A. Regalia. Monotonically convergent algorithms for symmetric tensor approximation. *Linear Algebra Appl.*, 438(2):875–890, 2013.
23. N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer. Tensorlab 3.0. *available online, URL: www. tensorlab. net*, 2016.

Appendix C

An unconventional robust integrator for dynamical low-rank approximation

An unconventional robust integrator for dynamical low-rank approximation

Gianluca Ceruti, Christian Lubich

Abstract We propose and analyse a numerical integrator that computes a low-rank approximation to large time-dependent matrices that are either given explicitly via their increments or are the unknown solution to a matrix differential equation. Furthermore, the integrator is extended to the approximation of time-dependent tensors by Tucker tensors of fixed multilinear rank. The proposed low-rank integrator is different from the known projector-splitting integrator for dynamical low-rank approximation, but it retains the important robustness to small singular values that has so far been known only for the projector-splitting integrator. The new integrator also offers some potential advantages over the projector-splitting integrator: It avoids the backward time integration substep of the projector-splitting integrator, which is a potentially unstable substep for dissipative problems. It offers more parallelism, and it preserves symmetry or anti-symmetry of the matrix or tensor when the differential equation does. Numerical experiments illustrate the behaviour of the proposed integrator.

Keywords dynamical low-rank approximation · structure-preserving integrator · matrix and tensor differential equations · Tucker tensor format

Mathematics Subject Classification (2000) 65L05 · 65L20 · 65L70 · 15A69

1 Introduction

For the approximation of huge time-dependent matrices (or tensors) that are the solution to a matrix differential equation, dynamical low-rank approximation [12, 13] projects the right-hand side function of the differential equation to the tangent space of matrices (or tensors) of a fixed rank at the current approximation. This yields differential equations for the factors of an SVD-like decomposition of the time-dependent low-rank approximation. The direct numerical integration of these differential equations by standard methods such as explicit or implicit Runge–Kutta

G. Ceruti and Ch. Lubich
Mathematisches Institut, Universität Tübingen, Auf der Morgenstelle 10, D-72076 Tübingen, Germany. E-mail: {ceruti, lubich}@na.uni-tuebingen.de

methods is highly problematic because in the typical presence of small singular values in the approximation, it leads to severe step size restrictions proportional to the smallest nonzero singular value. This difficulty does not arise with the projector-splitting integrator proposed in [16], which foregoes a direct time discretization of the differential equations for the factors and instead splits the orthogonal projection onto the tangent space, which is an alternating sum of subprojections. This approach leads to an efficiently implementable integrator that is robust to small singular values [11,16]. It has been extended, together with its robustness properties, from the matrix case to Tucker tensors in [15,18], to tensor trains / matrix product states in [17,8], and to general tree tensor networks in [6].

In the present paper we propose and analyse a different integrator that is shown to have the same robust error behaviour as the projector-splitting integrator. This new integrator can apparently not be interpreted as a splitting integrator or be included in another familiar class of integrators. Its substeps look formally similar to those of the projector-splitting integrator but are arranged in a different, less sequential way. Like in the projector-splitting integrator, the differential equations in the substeps are linear if the original differential equation is linear, even though the projected differential equation becomes nonlinear. The new integrator bears some similarity also to the constant-mean-field integrator of [3] and the splitting integrator of [10].

Beyond the robustness to small singular values, the new integrator has some favourable further properties that are not shared with the projector-splitting integrator. Maybe most importantly, it has no backward time integration substep as in the projector-splitting integrator. This appears advantageous in strongly dissipative problems, where the backward time integration step represents an unstable substep. Moreover, the new integrator has enhanced parallelism in its substeps, and in the Tucker tensor case even a reduced serial computational cost. It preserves symmetry or anti-symmetry of the matrix or tensor when the differential equation does. It reduces to the (anti-)symmetry-preserving low-rank integrator of [5] in this case.

On the other hand, unlike the projector-splitting integrator it cannot be efficiently extended to a time-reversible integrator. When applied to the time-dependent Schrödinger equation (as an integrator for the MCTDH method of quantum molecular dynamics; cf. [2,3,15]), the new integrator preserves the norm, but it has no energy conservation as shown in [15] for the projector-splitting integrator.

In Section 2 we recapitulate dynamical low-rank approximation and the projector-splitting integrator for the matrix case. We restate its exactness property and its robust error bound.

In Section 3 we present the new low-rank matrix integrator and show that it has the same exactness property and robust error bound as the matrix projector-splitting integrator.

In Section 4 we recapitulate dynamical low-rank approximation by Tucker tensors of fixed multilinear rank and the extension of the projector-splitting integrator to the Tucker tensor case.

In Section 5 we present the new low-rank Tucker tensor integrator and show that it has the same exactness property and robust error bound as the Tucker tensor projector-splitting integrator.

In Section 6 we illustrate the behaviour of the new low-rank matrix and Tucker tensor integrators by numerical experiments.

While we describe the integrator for real matrices and tensors, the algorithm and its properties extend in a straightforward way to complex matrices and tensors, requiring only some care in using transposes $\mathbf{U}^\top$ versus adjoints $\mathbf{U}^* = \overline{\mathbf{U}}^\top$.

Throughout the paper, we use the convention to denote matrices by boldface capital letters and tensors by italic capital letters.

2 Recap: the matrix projector-splitting integrator

Dynamical low-rank approximation of time-dependent matrices [12] replaces the exact solution $\mathbf{A}(t) \in \mathbb{R}^{m\times n}$ of a (too large) matrix differential equation

$$\dot{\mathbf{A}}(t) = \mathbf{F}(t, \mathbf{A}(t)), \qquad \mathbf{A}(t_0) = \mathbf{A}_0 \tag{1}$$

by the solution $\mathbf{Y}(t) \in \mathbb{R}^{m\times n}$ of rank r of the differential equation projected to the tangent space of the manifold of rank-r matrices at the current approximation,

$$\dot{\mathbf{Y}}(t) = \mathrm{P}(\mathbf{Y}(t))\mathbf{F}(t, \mathbf{Y}(t)), \qquad \mathbf{Y}(t_0) = \mathbf{Y}_0, \tag{2}$$

where the initial rank-r matrix $\mathbf{Y}_0$ is typically obtained from a truncated singular value decomposition (SVD) of $\mathbf{A}_0$. (We note that $\mathbf{F}(t, \mathbf{Y}) = \dot{\mathbf{A}}(t)$ if $\mathbf{A}(t)$ is given explicitly.) For the actual computation with rank-r matrices, they are represented in a non-unique factorized SVD-like form

$$\mathbf{Y}(t) = \mathbf{U}(t)\mathbf{S}(t)\mathbf{V}(t)^\top, \tag{3}$$

where the slim matrices $\mathbf{U}(t) \in \mathbb{R}^{m\times r}$ and $\mathbf{V}(t) \in \mathbb{R}^{n\times r}$ each have r orthonormal columns, and the small matrix $\mathbf{S}(t) \in \mathbb{R}^{r\times r}$ is invertible.

The orthogonal tangent space projection $\mathrm{P}(\mathbf{Y})$ can be written explicitly as an alternating sum of three subprojections onto the co-range, the intersection of co-range and range, and the range of the rank-r matrix $\mathbf{Y} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$ [12, Lemma 4.1]:

$$\mathrm{P}(\mathbf{Y})\mathbf{Z} = \mathbf{Z}\mathbf{V}\mathbf{V}^\top - \mathbf{U}\mathbf{U}^\top\mathbf{Z}\mathbf{V}\mathbf{V}^\top + \mathbf{U}\mathbf{U}^\top\mathbf{Z}, \qquad \mathbf{Z} \in \mathbb{R}^{m\times n}. \tag{4}$$

The projector-splitting integrator of [16] splits the right-hand side of (2) according to the three subprojections in the stated ordering and solves the subproblems consecutively in the usual way of a Lie–Trotter or Strang splitting. This approach yields an efficient time-stepping algorithm that updates the factors in the SVD-like decomposition of the rank-r matrices in every time step, alternating between solving differential equations for matrices of the dimension of the factor matrices and orthogonal decompositions of slim matrices.

One time step from t_0 to $t_1 = t_0 + h$ starting from a factored rank-r matrix $\mathbf{Y}_0 = \mathbf{U}_0\mathbf{S}_0\mathbf{V}_0^\top$ proceeds as follows:

1. **K-step** : Update $\mathbf{U}_0 \to \mathbf{U}_1, \mathbf{S}_0 \to \hat{\mathbf{S}}_1$
   Integrate from $t = t_0$ to t_1 the $m \times r$ matrix differential equation

   $$\dot{\mathbf{K}}(t) = \mathbf{F}(t, \mathbf{K}(t)\mathbf{V}_0^\top)\mathbf{V}_0, \qquad \mathbf{K}(t_0) = \mathbf{U}_0\mathbf{S}_0.$$

   Perform a QR factorization $\mathbf{K}(t_1) = \mathbf{U}_1\hat{\mathbf{S}}_1$.

2. **S-step** : Update $\hat{\mathbf{S}}_1 \to \tilde{\mathbf{S}}_0$
   Integrate from $t = t_0$ to t_1 the $r \times r$ matrix differential equation

$$\dot{\mathbf{S}}(t) = -\mathbf{U}_1^\top \mathbf{F}(t, \mathbf{U}_1 \mathbf{S}(t) \mathbf{V}_0^\top) \mathbf{V}_0, \qquad \mathbf{S}(t_0) = \hat{\mathbf{S}}_1,$$

   and set $\tilde{\mathbf{S}}_0 = \mathbf{S}(t_1)$.
3. **L-step** : Update $\mathbf{V}_0 \to \mathbf{V}_1, \tilde{\mathbf{S}}_0 \to \mathbf{S}_1$
   Integrate from $t = t_0$ to t_1 the $n \times r$ matrix differential equation

$$\dot{\mathbf{L}}(t) = \mathbf{F}(t, \mathbf{U}_1 \mathbf{L}(t)^\top)^\top \mathbf{U}_1, \qquad \mathbf{L}(t_0) = \mathbf{V}_0 \tilde{\mathbf{S}}_0^\top.$$

   Perform a QR factorization $\mathbf{L}(t_1) = \mathbf{V}_1 \mathbf{S}_1^\top$.

   Then, the approximation after one time step is given by

$$\mathbf{Y}_1 = \mathbf{U}_1 \mathbf{S}_1 \mathbf{V}_1^\top.$$

To proceed further, $\mathbf{Y}_1$ is taken as the starting value for the next step, and so on.

The projector-splitting integrator has very favourable properties. First, it reproduces rank-r matrices exactly.

Theorem 1 (Exactness property, [16, Theorem 4.1]) *Let $\mathbf{A}(t) \in \mathbb{R}^{m \times n}$ be of rank r for $t_0 \le t \le t_1$, so that $\mathbf{A}(t)$ has a factorization (3), $\mathbf{A}(t) = \mathbf{U}(t)\mathbf{S}(t)\mathbf{V}(t)^\top$. Moreover, assume that the $r \times r$ matrices $\mathbf{U}(t_1)^\top \mathbf{U}(t_0)$ and $\mathbf{V}(t_1)^\top \mathbf{V}(t_0)$ are invertible. With $\mathbf{Y}_0 = \mathbf{A}(t_0)$, the projector-splitting integrator for $\dot{\mathbf{Y}}(t) = \mathrm{P}(\mathbf{Y}(t))\dot{\mathbf{A}}(t)$ is then exact: $\mathbf{Y}_1 = \mathbf{A}(t_1)$.*

Even more remarkable, the algorithm is robust to the presence of small singular values of the solution or its approximation, as opposed to standard integrators applied to (2) or the equivalent differential equations for the factors $\mathbf{U}(t)$, $\mathbf{S}(t)$, $\mathbf{V}(t)$, which contain a factor $\mathbf{S}(t)^{-1}$ on the right-hand sides [12, Prop. 2.1]. The appearance of small singular values is ubiquitous in low-rank approximation, because the smallest singular value retained in the approximation cannot be expected to be much larger than the largest discarded singular value of the solution, which is required to be small for good accuracy of the low-rank approximation.

Theorem 2 (Robust error bound, [11, Theorem 2.1]) *Let $\mathbf{A}(t)$ denote the solution of the matrix differential equation (1). Assume that the following conditions hold in the Frobenius norm $\|\cdot\| = \|\cdot\|_F$:*

1. *$\mathbf{F}$ is Lipschitz-continuous and bounded: for all $\mathbf{Y}, \widetilde{\mathbf{Y}} \in \mathbb{R}^{m \times n}$ and $0 \le t \le T$,*

$$\|\mathbf{F}(t, \mathbf{Y}) - \mathbf{F}(t, \widetilde{\mathbf{Y}})\| \le L \|\mathbf{Y} - \widetilde{\mathbf{Y}}\|, \qquad \|\mathbf{F}(t, \mathbf{Y})\| \le B \ .$$

2. *The non-tangential part of $\mathbf{F}(t, \mathbf{Y})$ is ε-small:*

$$\|(\mathbf{I} - \mathrm{P}(\mathbf{Y}))\mathbf{F}(t, \mathbf{Y})\| \le \varepsilon$$

   for all $\mathbf{Y} \in \mathcal{M}$ in a neighbourhood of $\mathbf{A}(t)$ and $0 \le t \le T$.
3. *The error in the initial value is δ-small:*

$$\|\mathbf{Y}_0 - \mathbf{A}_0\| \le \delta.$$

Let $\mathbf{Y}_n$ denote the rank-r approximation to $\mathbf{A}(t_n)$ at $t_n = nh$ obtained after n steps of the projector-splitting integrator with step-size $h > 0$. Then, the error satisfies for all n with $t_n = nh \leq T$

$$\|\mathbf{Y}_n - \mathbf{A}(t_n)\| \leq c_0\delta + c_1\varepsilon + c_2 h,$$

where the constants c_i only depend on L, B, and T. In particular, the constants are independent of singular values of the exact or approximate solution.

In [11, Section 2.6.3] it is shown that an inexact solution of the matrix differential equations in the projector-splitting integrator leads to an additional error that is bounded in terms of the local errors in the inexact substeps, again with constants that do not depend on small singular values.

Numerical experiments with the matrix projector-splitting integrator and comparisons with standard numerical integrators are reported in [16,11].

3 A new robust low-rank matrix integrator

We now present a different integrator that has the same exactness and robustness properties as the projector-splitting integrator but which differs in the following favourable properties:

1. The solution of the differential equations for the $m \times r$ and $n \times r$ matrices can be done in parallel, and also the two QR decompositions can be done in parallel.
2. The differential equation for the small $r \times r$ matrix is solved forward in time, not backwards.
3. The integrator preserves (skew-)symmetry if the differential equation does.

While item 1. can clearly speed up the computation, item 2. is of interest for strongly dissipative problems, for which the S-step in the projector-splitting algorithm with the minus sign in the differential equations is an unstable substep of the algorithm. This does not appear in the new algorithm. We mention that in [1], the problem of the backward substep for parabolic problems has recently been addressed in a different way. In the alternative variant of the projector-splitting integrator proposed in [4], which is based on a rearrangement of the terms in (4), the S-step is integrated forward in time. However, contrary to the algorithm proposed here, no robust convergence with respect to small singular values has been proved.

On the other hand, contrary to the projector-splitting integrator, there is apparently no efficient way to construct a time-reversible integrator from this new integrator.

3.1 Formulation of the algorithm

One time step of integration from time t_0 to $t_1 = t_0 + h$ starting from a factored rank-r matrix $\mathbf{Y}_0 = \mathbf{U}_0\mathbf{S}_0\mathbf{V}_0^\top$ computes an updated rank-r factorization $\mathbf{Y}_1 = \mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^\top$ as follows.

1. Update $\mathbf{U}_0 \to \mathbf{U}_1$ and $\mathbf{V}_0 \to \mathbf{V}_1$ in parallel:

   K-step: Integrate from $t = t_0$ to t_1 the $m \times r$ matrix differential equation

   $$\dot{\mathbf{K}}(t) = \mathbf{F}(t, \mathbf{K}(t)\mathbf{V}_0^\top)\mathbf{V}_0, \qquad \mathbf{K}(t_0) = \mathbf{U}_0\mathbf{S}_0.$$

   Perform a QR factorization $\mathbf{K}(t_1) = \mathbf{U}_1\mathbf{R}_1$ and compute the $r \times r$ matrix $\mathbf{M} = \mathbf{U}_1^\top\mathbf{U}_0$.

   L-step : Integrate from $t = t_0$ to t_1 the $n \times r$ matrix differential equation

   $$\dot{\mathbf{L}}(t) = \mathbf{F}(t, \mathbf{U}_0\mathbf{L}(t)^\top)^\top\mathbf{U}_0, \qquad \mathbf{L}(t_0) = \mathbf{V}_0\mathbf{S}_0^\top.$$

   Perform a QR factorization $\mathbf{L}(t_1) = \mathbf{V}_1\widetilde{\mathbf{R}}_1$ and compute the $r \times r$ matrix $\mathbf{N} = \mathbf{V}_1^\top\mathbf{V}_0$.
2. Update $\mathbf{S}_0 \to \mathbf{S}_1$:

   S-step : Integrate from $t = t_0$ to t_1 the $r \times r$ matrix differential equation

   $$\dot{\mathbf{S}}(t) = \mathbf{U}_1^\top\mathbf{F}(t, \mathbf{U}_1\mathbf{S}(t)\mathbf{V}_1^\top)\mathbf{V}_1, \qquad \mathbf{S}(t_0) = \mathbf{M}\mathbf{S}_0\mathbf{N}^\top,$$

   and set $\mathbf{S}_1 = \mathbf{S}(t_1)$.

The $m \times r$, $n \times r$ and $r \times r$ matrix differential equations in the substeps are solved approximately using a standard integrator, e.g., an explicit or implicit Runge–Kutta method or an exponential integrator when $\mathbf{F}$ is dominantly linear.

We note that the L-step equals the K-step for the transposed function $\mathbf{G}(t, \mathbf{Y}) = \mathbf{F}(t, \mathbf{Y}^\top)^\top$ and transposed starting values. Unlike the projector-splitting algorithm, the triangular factors of the QR-decompositions are not reused. The S-step can be viewed as a Galerkin method for the differential equation (1) in the space of matrices $\mathbf{U}_1\mathbf{S}\mathbf{V}_1^\top$ generated by the updated basis matrices. In contrast to the projector-splitting integrator, there is no minus sign on the right-hand side of the differential equation for $\mathbf{S}(t)$. We further note that $\mathbf{U}_1$ of the new integrator is identical to $\mathbf{U}_1$ of the projector-splitting integrator, but $\mathbf{V}_1$ is in general different.

Remark 1 There exists a modification where all three differential equations for $\mathbf{K}$, $\mathbf{L}$ and $\mathbf{S}$ can be solved in parallel. That variant solves the K- and L-steps as above, but in the S-step it solves instead the $r \times r$ matrix differential equation

$$\dot{\mathbf{S}}(t) = \mathbf{U}_0^\top\mathbf{F}(t, \mathbf{U}_0\mathbf{S}(t)\mathbf{V}_0^\top)\mathbf{V}_0, \qquad \mathbf{S}(t_0) = \mathbf{S}_0$$

and finally sets

$$\mathbf{S}_1 = \mathbf{M}^{-\top}\,\mathbf{S}(t_1)\,\mathbf{N}^{-1}.$$

This modified integrator can be shown to have the same exactness property as proved below for the integrator formulated above, and also a similar robust error bound *under the condition that* the inverses of the matrices $\mathbf{M}$ and $\mathbf{N}$ are bounded by a constant. This condition can, however, be guaranteed only for step sizes that are small in comparison to the smallest nonzero singular value. In our numerical experiments this method did not behave as reliably as the method proposed above, and despite its interesting properties it will therefore not be further discussed in the following.

3.2 Exactness property and robust error bound

We will prove the following remarkable results for the integrator of Section 3.1.

Theorem 3 *The exactness property of Theorem* 1 *holds verbatim also for the new integrator.*

Theorem 4 *The robust error bound of Theorem* 2 *holds verbatim also for the new integrator.*

As in [11, Section 2.6.3], it can be further shown that an inexact solution of the matrix differential equations in the projector-splitting integrator leads to an additional error that is bounded in terms of the local errors in the inexact substeps, again with constants that do not depend on small singular values.

3.3 Proof of Theorem 3

For the proof of Theorem 3 we need the following auxiliary result, which extends an analogous result in [5] for symmetric matrices.

Lemma 1 *Let* $\mathbf{A}(t) \in \mathbb{R}^{m\times n}$ *be of rank* r *for* $t_0 \leq t \leq t_1$*, so that* $\mathbf{A}(t)$ *has a factorization* (3)*,* $\mathbf{A}(t) = \mathbf{U}(t)\mathbf{S}(t)\mathbf{V}(t)^\top$*. Moreover, assume that the* $r \times r$ *matrix* $\mathbf{V}(t_1)^\top\mathbf{V}(t_0)$ *is invertible. Then,*

$$\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}(t_1) = \mathbf{A}(t_1).$$

Proof The solution of the K-step at time t_1 is

$$\mathbf{K}(t_1) = \mathbf{A}(t_0)\mathbf{V}_0 + \big(\mathbf{A}(t_1) - \mathbf{A}(t_0)\big)\mathbf{V}_0 = \mathbf{A}(t_1)\mathbf{V}_0.$$

Hence,

$$\mathbf{K}(t_1) = \mathbf{U}(t_1)\big[\mathbf{S}(t_1)(\mathbf{V}(t_1)^\top\mathbf{V}(t_0))\big] \ .$$

By assumption, the factor in big square brackets is invertible. Computing a QR-decomposition of this term, we have

$$\mathbf{K}(t_1) = \mathbf{U}(t_1)\mathbf{Q}\mathbf{R} \ ,$$

where $\mathbf{Q} \in \mathbb{R}^{r\times r}$ is an orthogonal matrix and $\mathbf{R} \in \mathbb{R}^{r\times r}$ is invertible and upper triangular. The QR-factorization of $\mathbf{K}(t_1)$ thus yields

$$\mathbf{U}_1 = \mathbf{U}(t_1)\mathbf{Q} \in \mathbb{R}^{m\times r}.$$

To conclude,

$$\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}(t_1) = \mathbf{U}(t_1)\mathbf{Q}\mathbf{Q}^\top\mathbf{U}(t_1)^\top\mathbf{A}(t_1) = \mathbf{U}(t_1)\mathbf{U}(t_1)^\top\mathbf{A}(t_1) = \mathbf{A}(t_1),$$

which is the stated result. □

Proof (of Theorem 3) Since the L-step is the K-step for the transposed matrix $\mathbf{A}(t)^\top$, which has the same rank as $\mathbf{A}(t)$, it follows from Lemma 1 that

$$\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}(t_1) = \mathbf{A}(t_1), \qquad \mathbf{V}_1\mathbf{V}_1^\top\mathbf{A}(t_1)^\top = \mathbf{A}(t_1)^\top. \tag{5}$$

The integrator yields in the S-step

$$\mathbf{S}_1 = \mathbf{U}_1^\top\mathbf{Y}_0\mathbf{V}_1 + \mathbf{U}_1^\top(\mathbf{A}(t_1) - \mathbf{A}(t_0))\mathbf{V}_1 = \mathbf{U}_1^\top\mathbf{A}(t_1)\mathbf{V}_1,$$

since $\mathbf{Y}_0 = \mathbf{A}(t_0)$. The result after a time step of the new integrator is

$$\mathbf{Y}_1 = \mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^\top = \mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}(t_1)(\mathbf{V}_1\mathbf{V}_1^\top) = \mathbf{A}(t_1),$$

where the last equality holds because of (5). □

3.4 Proof of Theorem 4

Under the assumptions of Theorem 2, we introduce the quantity

$$\vartheta := (4e^{Lh}BL + 9BL)h^2 + (3e^{Lh} + 4)\varepsilon h + e^{Lh}\delta\,, \tag{6}$$

which is the local error bound of the projector-splitting integrator after one time step, as proved in [11, Theorem 2.1].

Lemma 2 *Let $\mathbf{A}_1$ be the solution at time $t_1 = t_0 + h$ of the full problem (1) with initial condition $\mathbf{A}_0$. Assume that conditions* 1.-3. *of Theorem* 2 *are fulfilled. Then,*

$$\|\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1 - \mathbf{A}_1\| \le \vartheta.$$

Proof The result is proved in the course of the proof of Lemma 1 in [5]. We give the proof here for the convenience of the reader. The local error analysis in [11] shows that the $r \times n$ matrix $\mathbf{Z} = \mathbf{S}_1^{\mathrm{ps}}\mathbf{V}_1^{\mathrm{ps},\top}$, where $\mathbf{S}_1^{\mathrm{ps}}$ and $\mathbf{V}_1^{\mathrm{ps}}$ are the matrices computed in the third substep of the projector-splitting algorithm, satisfies

$$\|\mathbf{U}_1\mathbf{Z} - \mathbf{A}_1\| \le \vartheta.$$

The square of the left-hand side can be split into two terms:

$$\begin{aligned}\|\mathbf{U}_1\mathbf{Z} - \mathbf{A}_1\|^2 &= \|\mathbf{U}_1\mathbf{Z} - \mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1 + \mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1 - \mathbf{A}_1\|^2\\ &= \|\mathbf{U}_1\mathbf{U}_1^\top(\mathbf{U}_1\mathbf{Z} - \mathbf{A}_1) + (\mathbf{I} - \mathbf{U}_1\mathbf{U}_1^\top)\mathbf{A}_1\|^2\\ &= \|\mathbf{U}_1\mathbf{U}_1^\top(\mathbf{U}_1\mathbf{Z} - \mathbf{A}_1)\|^2 + \|(\mathbf{I} - \mathbf{U}_1\mathbf{U}_1^\top)\mathbf{A}_1\|^2.\end{aligned}$$

Hence,

$$\|\mathbf{U}_1\mathbf{U}_1^\top(\mathbf{U}_1\mathbf{Z} - \mathbf{A}_1)\|^2 + \|(\mathbf{I} - \mathbf{U}_1\mathbf{U}_1^\top)\mathbf{A}_1\|^2 \le \vartheta^2.$$

This yields the stated result for the second term. □

Lemma 3 *Let $\mathbf{A}_1$, $\mathbf{U}_1$ and $\mathbf{V}_1$ be defined as above. The following estimate holds:*

$$\|\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1\mathbf{V}_1\mathbf{V}_1^\top - \mathbf{A}_1\| \le 2\vartheta.$$

Proof The L-step is the K-step for the transposed function $\mathbf{G}(t,\mathbf{Y}) = \mathbf{F}(t,\mathbf{Y}^\top)^\top$, which again fulfills conditions 1.- 3. of Theorem 2. Conditions 1. and 3. hold because of the invariance of the Frobenius norm under transposition. Condition 2. holds because

$$\|(\mathbf{I}-\mathrm{P}(\mathbf{Y}))\mathbf{G}(t,\mathbf{Y})\| = \|(\mathbf{I}-\mathrm{P}(\mathbf{Y}^\top))\mathbf{F}(t,\mathbf{Y}^\top)\| \le \varepsilon,$$

where we used the identity $\mathrm{P}(\mathbf{Y})\mathbf{Z}^\top = \left[\mathrm{P}(\mathbf{Y}^\top)\mathbf{Z}\right]^\top$. From Lemma 2 we thus have

$$\begin{aligned} \|\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1 - \mathbf{A}_1\| &\le \vartheta, \\ \|\mathbf{V}_1\mathbf{V}_1^\top\mathbf{A}_1^\top - \mathbf{A}_1^\top\| &\le \vartheta. \end{aligned} \tag{7}$$

This implies that

$$\begin{aligned} \|\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1\mathbf{V}_1\mathbf{V}_1^\top - \mathbf{A}_1\| &\le \|\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1\mathbf{V}_1\mathbf{V}_1^\top - \mathbf{A}_1\mathbf{V}_1\mathbf{V}_1^\top + \mathbf{A}_1\mathbf{V}_1\mathbf{V}_1^\top - \mathbf{A}_1\| \\ &\le \|\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1\mathbf{V}_1\mathbf{V}_1^\top - \mathbf{A}_1\mathbf{V}_1\mathbf{V}_1^\top\| + \|\mathbf{A}_1\mathbf{V}_1\mathbf{V}_1^\top - \mathbf{A}_1\| \\ &\le \|\big(\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1 - \mathbf{A}_1\big)\mathbf{V}_1\mathbf{V}_1^\top\| + \|\mathbf{V}_1\mathbf{V}_1^\top\mathbf{A}_1^\top - \mathbf{A}_1^\top\| \\ &\le \|\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1 - \mathbf{A}_1\| \cdot \|\mathbf{V}_1\mathbf{V}_1^\top\|_2 + \|\mathbf{V}_1\mathbf{V}_1^\top\mathbf{A}_1^\top - \mathbf{A}_1^\top\|. \end{aligned}$$

Since $\|\mathbf{V}_1\mathbf{V}_1^\top\|_2 = 1$, the result follows from (7). □

In the following lemma, we show that the approximation given after one time step is $O(h(h+\varepsilon))$ close to the solution of system (1) when the starting values coincide.

Lemma 4 (Local Error) *If $\mathbf{A}_0 = \mathbf{Y}_0$, the following local error bound holds:*

$$\|\mathbf{Y}_1 - \mathbf{A}_1\| \le h(\hat{c}_1\varepsilon + \hat{c}_2 h),$$

where the constants only depend on L and B and a bound of the step size. In particular, the constants are independent of singular values of the exact or approximate solution.

Proof The proof has similarities to that of the local error bound for the (skew)-symmetry preserving integrator of [5, Lemma 2] but requires a few crucial modifications. By the identity $\mathbf{Y}_1 = \mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^\top$ and Lemma 3 we have that

$$\begin{aligned} \|\mathbf{Y}_1 - \mathbf{A}_1\| &\le \|\mathbf{Y}_1 - \mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1\mathbf{V}_1\mathbf{V}_1^\top\| + \|\mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}_1\mathbf{V}_1\mathbf{V}_1^\top - \mathbf{A}_1\| \\ &\le \|\mathbf{U}_1(\mathbf{S}_1 - \mathbf{U}_1^\top\mathbf{A}_1\mathbf{V}_1)\mathbf{V}_1^\top\| + 2\vartheta \\ &= \|\mathbf{S}_1 - \mathbf{U}_1^\top\mathbf{A}_1\mathbf{V}_1\| + 2\vartheta. \end{aligned}$$

The analysis of the local error thus reduces to estimating $\|\mathbf{S}_1 - \mathbf{U}_1^\top\mathbf{A}_1\mathbf{V}_1\|$. To this end, we introduce the following quantity: for $t_0 \le t \le t_1$,

$$\widetilde{\mathbf{S}}(t) := \mathbf{U}_1^\top\mathbf{A}(t)\mathbf{V}_1.$$

We write

$$\mathbf{A}(t) = \mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}(t)\mathbf{V}_1\mathbf{V}_1^\top + \Big(\mathbf{A}(t) - \mathbf{U}_1\mathbf{U}_1^\top\mathbf{A}(t)\mathbf{V}_1\mathbf{V}_1^\top\Big) = \mathbf{U}_1\widetilde{\mathbf{S}}(t)\mathbf{V}_1^\top + \mathbf{R}(t),$$

where $\mathbf{R}(t)$ denotes the term in big brackets. Lemma 3 and the bound B of $\mathbf{F}$ yield, for $t_0 \le t \le t_1$,

$$\|\mathbf{A}(t) - \mathbf{A}(t_1)\| \le \int_{t_0}^{t_1} \|\dot{\mathbf{A}}(s)\|\, ds = \int_{t_0}^{t_1} \|\mathbf{F}(s, \mathbf{A}(s))\|\, ds \le Bh.$$

Hence the remainder term is bounded by

$$\|\mathbf{R}(t)\| \le \|\mathbf{R}(t) - \mathbf{R}(t_1)\| + \|\mathbf{R}(t_1)\| \le 2Bh + 2\vartheta.$$

It follows that $\mathbf{F}(t, \mathbf{A}(t))$ can be written as

$$\begin{aligned}\mathbf{F}(t, \mathbf{A}(t)) &= \mathbf{F}(t, \mathbf{U}_1\widetilde{\mathbf{S}}(t)\mathbf{V}_1^\top + \mathbf{R}(t)) \\ &= \mathbf{F}(t, \mathbf{U}_1\widetilde{\mathbf{S}}(t)\mathbf{V}_1^\top) + \mathbf{D}(t)\end{aligned}$$

with the defect

$$\mathbf{D}(t) := \mathbf{F}(t, \mathbf{U}_1\widetilde{\mathbf{S}}(t)\mathbf{V}_1^\top + \mathbf{R}(t)) - \mathbf{F}(t, \mathbf{U}_1\widetilde{\mathbf{S}}(t)\mathbf{V}_1^\top).$$

With the Lipschitz constant L of $\mathbf{F}$, the defect is bounded by

$$\|\mathbf{D}(t)\| \le L\|\mathbf{R}(t)\| \le 2L(Bh + \vartheta).$$

We compare the two differential equations

$$\begin{aligned}\dot{\widetilde{\mathbf{S}}}(t) &= \mathbf{U}_1^\top \mathbf{F}(t, \mathbf{U}_1\widetilde{\mathbf{S}}(t)\mathbf{V}_1^\top)\mathbf{V}_1 + \mathbf{U}_1^\top \mathbf{D}(t)\mathbf{V}_1, & \widetilde{\mathbf{S}}(t_0) &= \mathbf{U}_1^\top \mathbf{Y}_0 \mathbf{V}_1, \\ \dot{\mathbf{S}}(t) &= \mathbf{U}_1^\top \mathbf{F}(t, \mathbf{U}_1\mathbf{S}(t)\mathbf{V}_1^\top)\mathbf{V}_1, & \mathbf{S}(t_0) &= \mathbf{U}_1^\top \mathbf{Y}_0 \mathbf{V}_1.\end{aligned}$$

By construction, the solution of the first differential equation at time t_1 is $\widetilde{\mathbf{S}}(t_1) = \mathbf{U}_1^\top \mathbf{A}_1 \mathbf{V}_1$. The solution of the second differential equation is $\mathbf{S}_1$ as given by the S-step of the integrator. With the Gronwall inequality we obtain

$$\|\mathbf{S}_1 - \mathbf{U}_1^\top \mathbf{A}_1 \mathbf{V}_1\| \le \int_{t_0}^{t_1} e^{L(t_1 - s)}\, \|\mathbf{D}(s)\|\, ds \le e^{Lh}\, 2L(Bh + \vartheta)h.$$

The result now follows using the definition of ϑ. □

Using the Lipschitz continuity of the function $\mathbf{F}$, we pass from the local to the global errors by the standard argument of Lady Windermere's fan [9, Section II.3] and thus conclude the proof of Theorem 7.

3.5 Symmetric and skew-symmetric low-rank matrices

We now assume that the right-hand side function in (1) is such that one of the following conditions holds,

$$\mathbf{F}(t, \mathbf{Y}^\top)^\top = \mathbf{F}(t, \mathbf{Y}) \qquad \text{for all } \ \mathbf{Y} \in \mathbb{R}^{n\times n} \tag{8}$$

or

$$\mathbf{F}(t, \mathbf{Y}^\top)^\top = -\mathbf{F}(t, -\mathbf{Y}) \qquad \text{for all } \ \mathbf{Y} \in \mathbb{R}^{n\times n}. \tag{9}$$

Under these conditions, solutions to (1) with symmetric or skew-symmetric initial data remain symmetric or skew-symmetric, respectively, for all times. We also have preservation of (skew-)symmetry for the new integrator, which does not hold for the projector-splitting integrator.

Theorem 5 *Let* $\mathbf{Y}_0 = \mathbf{U}_0\mathbf{S}_0\mathbf{U}_0^\top \in \mathbb{R}^{n\times n}$ *be symmetric or skew-symmetric and assume that the function* $\mathbf{F}$ *satisfies property* (8) *or* (9)*, respectively. Then, the approximation* $\mathbf{Y}_1$ *obtained after one time step of the new integrator is symmetric or skew-symmetric, respectively.*

Proof Let us just consider the skew-symmetric case (9). (The symmetric case is analogous.) The L-step is the K-step for the transposed function $\mathbf{G}(t,\mathbf{Y}) = \mathbf{F}(t,\mathbf{Y}^\top)^\top$, and so the skew-symmetry of $\mathbf{S}_0$ and property (9) imply that $\mathbf{L}(t_1) = -\mathbf{K}(t_1)$, which further yields $\mathbf{V}_1 = \mathbf{U}_1$ and $\mathbf{M} = \mathbf{N}$. These identities show that the initial value and the right-hand side function of the differential equation for $\mathbf{S}(t)$ are skew-symmetric, which implies that $\mathbf{S}(t)$ and hence $\mathbf{S}_1$ are still skew-symmetric. Altogether, the algorithm gives us the skew-symmetric result $\mathbf{Y}_1 = \mathbf{U}_1\mathbf{S}_1\mathbf{U}_1^\top$. □

Under condition (8) or (9), the new integrator coincides with the (skew)-symmetry preserving low-rank matrix integrator of [5].

4 Recap: the Tucker tensor projector-splitting integrator

The solution $A(t) \in \mathbb{R}^{n_1\times\cdots\times n_d}$ of a tensor differential equation

$$\dot{A}(t) = F(t, A(t)), \qquad A(0) = A_0 \tag{10}$$

is approximated by the solution $Y(t) \in \mathbb{R}^{n_1\times\cdots\times n_d}$ of multilinear rank $\mathbf{r} = (r_1,\dots,r_d)$ of the differential equation projected to the tangent space of the manifold of rank-$\mathbf{r}$ tensors at the current approximation ([13], cf. also [2]),

$$\dot{Y}(t) = \mathrm{P}(Y(t))F(t,Y(t)), \qquad Y(t_0) = Y_0, \tag{11}$$

where Y_0 is a rank-$\mathbf{r}$ approximation to A_0. Tensors $Y(t)$ of multilinear rank $\mathbf{r}$ are represented in the Tucker form [7], written here in a notation following [14]:

$$\begin{aligned} &Y(t) = C(t)\,\mathsf{X}_{i=1}^{d}\,\mathbf{U}_i(t), \\ &i.e., \quad y_{i_1,\dots,i_d}(t) = \sum_{j_1,\dots,j_d} c_{j_1,\dots,j_d}(t)\,u_{i_1,j_1}(t)\dots u_{i_d,j_d}(t), \end{aligned} \tag{12}$$

where the slim basis matrices $\mathbf{U}_i \in \mathbb{R}^{n_i\times r_i}$ have orthonormal columns and the smaller core tensor $C(t) \in \mathbb{R}^{r_1\times\cdots\times r_d}$ is of full multilinear rank $\mathbf{r}$.

The orthogonal tangent space projection $\mathrm{P}(Y)$ is given as an alternating sum of $2d-1$ subprojections [15], and like in the matrix case, a projector-splitting integrator with favourable properties can be formulated and efficiently implemented [15,18]. The algorithm runs through the modes $i = 1,\dots,d$ and solves differential equations for matrices of the dimension of the slim basis matrices and for the core tensor, alternating with orthogonalizations of slim matrices. Like the matrix projector-splitting integrator, also the Tucker tensor projector-splitting integrator has the exactness property and a robust error bound independently of small singular values of matricizations of the core tensor [18, Theorems 4.1 and 5.1].

5 A new robust low-rank Tucker tensor integrator

The low-rank numerical integrator defined in Section 3 for the matrix case extends in a natural way to the Tucker tensor format, and this extension still has the exactness property and robust error bounds that are independent of small singular values of matricizations of the core tensor.

In comparison with the Tucker integrator of [15] and [18], the new Tucker tensor integrator has the following favourable properties:

1. The solution of the differential equations for the $n_i \times r_i$ matrices can be done in parallel for $i = 1, \dots, d$, and also the QR decompositions can be done in parallel.
2. No differential equations are solved backward in time. No differential equations for $r_i \times r_i$ matrices need to be solved.
3. The integrator preserves (anti-)symmetry if the differential equation does.

On the other hand, in contrast to the projector-splitting Tucker integrator there is apparently no efficient way to construct a time-reversible integrator from this new Tucker integrator.

5.1 Formulation of the algorithm

One time step of integration from time t_0 to $t_1 = t_0 + h$ starting from a Tucker tensor of multilinear rank $(r_1, \dots, r_d)$ in factorized form, $Y_0 = C_0 \mathsf{X}_{i=1}^{d} \mathbf{U}_i^0$, computes an updated Tucker tensor of multilinear rank $(r_1, \dots, r_d)$ in factorized form, $Y_1 = C_1 \mathsf{X}_{i=1}^{d} \mathbf{U}_i^1$, in the following way:

1. Update the basis matrices $\mathbf{U}_i^0 \to \mathbf{U}_i^1$ for $i = 1, \dots, d$ in parallel:

   Perform a QR factorization of the transposed i-mode matricization of the core tensor:

   $$\mathbf{Mat}_i(C_0)^\top = \mathbf{Q}_i \mathbf{S}_i^{0,\top}.$$

   With $\mathbf{V}_i^{0,\top} = \mathbf{Q}_i^\top \bigotimes_{j \neq i}^{d} \mathbf{U}_j^{0,\top} \in \mathbb{R}^{r_i \times n_{\neg i}}$ (which yields $\mathbf{Mat}_i(Y_0) = \mathbf{U}_i^0 \mathbf{S}_i^0 \mathbf{V}_i^{0,\top}$)

   and the matrix function $\mathbf{F}_i(t, \cdot) := \mathbf{Mat}_i \circ F(t, \cdot) \circ \mathit{Ten}_i$, integrate from $t = t_0$ to t_1 the $n_i \times r_i$ matrix differential equation

   $$\dot{\mathbf{K}}_i(t) = \mathbf{F}_i(t, \mathbf{K}_i(t) \mathbf{V}_i^{0,\top}) \mathbf{V}_i^0, \qquad \mathbf{K}_i(t_0) = \mathbf{U}_i^0 \mathbf{S}_i^0.$$

   Perform a QR factorization $\mathbf{K}_i(t_1) = \mathbf{U}_i^1 \mathbf{R}_i^1$ and compute the $r_i \times r_i$ matrix $\mathbf{M}_i = \mathbf{U}_i^{1,\top} \mathbf{U}_i^0$.
2. Update the core tensor $C_0 \to C_1$:

   Integrate from $t = t_0$ to t_1 the $r_1 \times \cdots \times r_d$ tensor differential equation

   $$\dot{C}(t) = F\left(t, C(t) \mathsf{X}_{i=1}^{d} \mathbf{U}_i^1\right) \mathsf{X}_{i=1}^{d} \mathbf{U}_i^{1,\top}, \quad C(t_0) = C_0 \mathsf{X}_{i=1}^{d} \mathbf{M}_i$$

   and set $C_1 = C(t_1)$.

To continue in time, we take Y_1 as starting value for the next step and perform another step of the integrator.

We observe that, in contrast to the Tucker integrators of [18,15], the factors $\mathbf{U}_i \in \mathbb{R}^{n_i \times r_i}$ are updated simultaneously for $i = 1, \dots, d$.

5.2 Exactness property

The following result extends the exactness results of Theorem 3 and [18, Theorem 4.1] to the new Tucker tensor integrator.

Theorem 6 (Exactness property) *Let $A(t) = C(t) \bigtimes_{i=1}^{d} \mathbf{U}_i(t)$ be of multilinear rank $(r_1, \ldots, r_d)$ for $t_0 \le t \le t_1$. Moreover, assume that the $r_i \times r_i$ matrix $\mathbf{U}_i(t_1)^\top \mathbf{U}_i(t_0)$ is invertible for each $i = 1, \ldots, d$. With $Y_0 = A(t_0)$, the new Tucker integrator with rank $(r_1, \ldots, r_d)$ for $\dot{Y}(t) = \mathrm{P}(Y(t))\dot{A}(t)$ with starting value $Y_0 = A(t_0)$ is then exact: $Y_1 = A(t_1)$.*

Proof For each $i = 1, \ldots, d$, we apply Lemma 1 to $\mathbf{Mat}_i(\dot{A}(t))$

$$\mathbf{Mat}_i(A(t_1) \times_i \mathbf{U}_i^1 \mathbf{U}_i^{1,\top}) = \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} \mathbf{Mat}_i(A(t_1)) = \mathbf{Mat}_i(A(t_1)).$$

We tensorize in the i-th mode and obtain

$$A(t_1) \times_i \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} = A(t_1), \qquad i = 1, \ldots, d\ .$$

With $Y_0 = A(t_0)$ we obtain from the second substep of the algorithm

$$\begin{aligned} Y_1 &= C_1 \bigtimes_{i=1}^{d} \mathbf{U}_i^1 \\ &= \Big(Y_0 \bigtimes_{i=1}^{d} \mathbf{U}_i^{1,\top} + (A(t_1) - A(t_0)) \bigtimes_{i=1}^{d} \mathbf{U}_i^{1,\top} \Big) \bigtimes_{i=1}^{d} \mathbf{U}_i^1 \\ &= \Big(A(t_1) \bigtimes_{i=1}^{d} \mathbf{U}_i^{1,\top} \Big) \bigtimes_{i=1}^{d} \mathbf{U}_i^1 \\ &= A(t_1) \bigtimes_{i=1}^{d} \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} = A(t_1)\,, \end{aligned}$$

which proves the exactness. □

5.3 Robust error bound

The robust error bounds from Theorem 4 and [18, Theorem 5.1] extend to the new Tucker tensor integrator as follows. The norm $\|B\|$ of a tensor B used here is the Euclidean norm of the vector of entries of B.

Theorem 7 (Robust error bound) *Let $A(t)$ denote the solution of the tensor differential equation* (10)*. Assume the following:*

1. *F is Lipschitz-continuous and bounded.*
2. *The non-tangential part of $F(t, Y)$ is ε-small:*

$$\|(I - \mathrm{P}(Y))F(t, Y)\| \le \varepsilon$$

   for all Y of multilinear rank $(r_1, \ldots, r_d)$ in a neighbourhood of $A(t)$ and $0 \le t \le T$.
3. *The error in the initial value is δ-small:*

$$\|Y_0 - A_0\| \le \delta.$$

Let Y_n denote the approximation of multinear rank $(r_1, \ldots, r_d)$ to $A(t_n)$ at $t_n = nh$ obtained after n steps of the new Tucker integrator with step-size $h > 0$. Then, the error satisfies for all n with $t_n = nh \leq T$

$$\|Y_n - A(t_n)\| \leq c_0\delta + c_1\varepsilon + c_2 h,$$

where the constants c_i only depend on the Lipschitz constant L and bound B of F, on T, and on the dimension d. In particular, the constants are independent of singular values of matricizations of the exact or approximate solution.

The proof of Theorem 7 proceeds similar to the proof of Theorem 4 for the matrix case. We begin with two key lemmas and are then in a position to analyse the local error produced after one time step. We denote the solution value at t_1 by A_1. The basis matrix computed in the first part of the integrator is denoted by $\mathbf{U}_i^1$ for each $i = 1, \ldots, d$.

Lemma 5 *For each $i = 1, \ldots, d$, the function $\mathbf{F}_i(t, \cdot) := \mathbf{Mat}_i \circ F(t, \cdot) \circ \mathit{Ten}_i$ fulfills Conditions* 1. - 2. *of Theorem* 2*, and the initial matrix $\mathbf{Y}_{(i)}^0 = \mathbf{Mat}_i(Y_0)$ fulfills Condition* 3. *of that theorem.*

Proof For each $i = 1 \ldots d$, it holds that for $\mathbf{Y}_{(i)} = \mathbf{Mat}_i(Y)$,

$$\|\mathbf{F}_i(t, \mathbf{Y}_{(i)})\| = \|F(t, Y)\|.$$

The boundedness and Lipschitz condition of the matrix-valued function $\mathbf{F}_i$ follows from the boundedness and Lipschitz condition of the tensor-valued function F.

Condition 2. follows with the help of the correspondingly defined projection

$$\mathrm{P}_i(\mathbf{Y}_{(i)}) := \mathbf{Mat}_i \circ \mathrm{P}(Y) \circ \mathit{Ten}_i \quad \text{for} \quad \mathbf{Y}_{(i)} = \mathbf{Mat}_i(Y),$$

which is an orthogonal projection onto a subspace of the tangent space at $\mathbf{Y}_{(i)}$ of the manifold of rank-r_i matrices of dimension $n_i \times n_{\neg i}$. Denoting the orthogonal projection onto this tangent space by $\mathrm{P}_{(i)}(\mathbf{Y}_{(i)})$, we thus have

$$\|\big(\mathbf{I} - \mathrm{P}_{(i)}(\mathbf{Y}_{(i)})\big)\mathbf{F}_i(t, \mathbf{Y}_{(i)})\| \leq \|\big(\mathbf{I} - \mathrm{P}_i(\mathbf{Y}_{(i)})\big)\mathbf{F}_i(t, \mathbf{Y}_{(i)})\| = \|(\mathbf{I} - \mathrm{P}(Y))F(t, Y)\| \leq \varepsilon.$$

Condition 3. holds due to the invariance of the Frobenius norm under matricization,

$$\|\mathbf{Y}_{(i)}^0 - \mathbf{Mat}_i(A_0)\| = \|\mathbf{Mat}_i(Y_0 - A_0)\| = \|Y_0 - A_0\| \leq \delta,$$

and so we obtain the stated result. □

Lemma 6 *The following estimate holds with ϑ of* (6)*:*

$$\|A_1 \bigtimes_{i=1}^d \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} - A_1\| \leq d\,\vartheta,$$

where c only depends on d and a bound for hL.

Proof From Lemma 5 and Lemma 2,

$$\|\mathbf{U}_i^1 \mathbf{U}_i^{1,\top} \mathbf{Mat}_i(A_1) - \mathbf{Mat}_i(A_1)\| \leq \vartheta, \qquad i = 1, \dots, d.$$

The norm is invariant under tensorization and so the bound is equivalent to

$$\|A_1 \times_i \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} - A_1\| \leq \vartheta, \qquad i = 1, \dots, d.$$

To conclude, we observe

$$\begin{aligned}
&\|A_1 \bigtimes_{i=1}^d \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} - A_1\| \\
&\leq \|A_1 \bigtimes_{i=1}^d \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} - A_1 \bigtimes_{i=1}^{d-1} \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} + A_1 \bigtimes_{i=1}^{d-1} \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} - A_1\| \\
&\leq \|A_1 \bigtimes_{i=1}^d \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} - A_1 \bigtimes_{i=1}^{d-1} \mathbf{U}_i^1 \mathbf{U}_i^{1,\top}\| + \|A_1 \bigtimes_{i=1}^{d-1} \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} - A_1\| \\
&\leq \|(A_1 \times_d \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} - A_1) \bigtimes_{i=1}^{d-1} \mathbf{U}_i^1 \mathbf{U}_i^{1,\top}\| + \|A_1 \bigtimes_{i=1}^{d-1} \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} - A_1\| \\
&\leq \|A_1 \times_d \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} - A_1\| + \|A_1 \bigtimes_{i=1}^{d-1} \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} - A_1\| \\
&\leq \vartheta + \|A_1 \bigtimes_{i=1}^{d-1} \mathbf{U}_i^1 \mathbf{U}_i^{1,\top} - A_1\|,
\end{aligned}$$

and the result follows by an iteration of this argument. □

We are now in a position to analyse the local error produced after one time step of the integrator.

Lemma 7 (Local error) *If $A_0 = Y_0$, the following local error bound holds for the new Tucker tensor integrator:*

$$\|Y_1 - A_1\| \leq \hat{c}\, h(BLh + \varepsilon),$$

where $\hat{c}$ only depends on d and a bound of hL. In particular, the constant is independent of singular values of the exact or approximate solution.

We omit the proof because, up to modifications analogous to those in the proof of Lemma 4, the result follows as in [5, Section 5.3] for the (anti-)symmetry preserving Tucker integrator, using the two previous lemmas.

Using the Lipschitz continuity of the function F, we pass from the local to the global errors by the standard argument of Lady Windermere's fan [9, Section II.3] and thus conclude the proof of Theorem 7.

5.4 Symmetric and anti-symmetric low-rank Tucker tensors

For permutations $\sigma \in S_d$, we use the notation $\sigma(Y) = \big(y_{i_{\sigma(1)},\dots,i_{\sigma(d)}}\big)$ for tensors $Y = (y_{i_1,\dots,i_d}) \in \mathbb{R}^{n \times \cdots \times n}$ of order d. A tensor Y is called *symmetric* if $\sigma(Y) = Y$ for all $\sigma \in S_d$, and is called *anti-symmetric* if $\sigma(Y) = (-1)^{\mathrm{sign}(\sigma)}\, Y$ for all $\sigma \in S_d$.

We now assume that the right-hand side function in (10) is such that one of the following conditions holds: For all permutations $\sigma \in S_d$ and all tensors $Y \in \mathbb{R}^{n \times \cdots \times n}$ of order d,

$$\sigma\big(F(t, \sigma(Y))\big) = F(t, Y) \tag{13}$$

or

$$\sigma\big(F(t, \sigma(Y))\big) = (-1)^{\mathrm{sign}(\sigma)}\, F(t, Y) \tag{14}$$

Under these conditions, solutions to (1) with symmetric or anti-symmetric initial data remain symmetric or anti-symmetric, respectively, for all times. We also have preservation of (anti-)symmetry for the new integrator, which does not hold for the projector-splitting integrator.

Theorem 8 *Let Y_0 be symmetric or anti-symmetric and assume that the function F satisfies property* (13) *or* (14)*, respectively. Then, the approximation Y_1 obtained after one time step of the new integrator is symmetric or anti-symmetric, respectively.*

The simple proof is similar to the matrix case and is therefore omitted.

Under condition (13) or (14), the new integrator coincides with the (anti)-symmetry preserving low-rank Tucker tensor integrator of [5].

6 Numerical Experiments

In this section, we present results of different numerical experiments. The experiments were done using Matlab R2017a software with TensorLab package v3.0 [19].

6.1 Robustness with respect to small singular values

In the first example, the time-dependent matrix is given explicitly as

$$\mathbf{A}(t) = \left(e^{t\mathbf{W}_1}\right) e^t \mathbf{D} \left(e^{t\mathbf{W}_2}\right)^\top, \quad 0 \le t \le 1 \ .$$

The matrix $\mathbf{D} \in \mathbb{R}^{N\times N}$ is diagonal with entries $d_j = 2^{-j}$. The matrices $\mathbf{W}_1 \in \mathbb{R}^{N\times N}$ and $\mathbf{W}_2 \in \mathbb{R}^{N\times N}$ are skew-symmetric and randomly generated. We note that $e^t 2^{-j}$ are the singular values of $A(t)$. We choose $N = 100$ and final time $T = 1$. We compare the new low-rank integrator presented in Section 3 with a numerical solution obtained with the classical fourth-order explicit Runge-Kutta method applied to the system of differential equations for dynamical low-rank approximation as derived in [12, Proposition 2.1].

The computational cost of the new low-rank matrix integrator is governed by the matrix multiplications as appearing on the right-hand side of the differential equations for the substeps given in Section 3.1, by the two QR-decompositions of $N \times r$ matrices and the construction of the matrices $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{r\times r}$ together with the initial value of the S-step. The computational cost of the Runge–Kutta integrator applied to the system of [12, Proposition 2.1] is dominated by the matrix multiplications appearing on the right-hand side of the system, which are of the same dimensions as in the new integrator. Therefore, the computational costs per step of the two integrators are similar.

The numerical results for different ranks are shown in Figure 1. In contrast to the Runge–Kutta method, the new low-rank integrator does not require a step-size restriction in the presence of small singular values. The same favourable behaviour was shown for the projector-splitting integrator in [11].

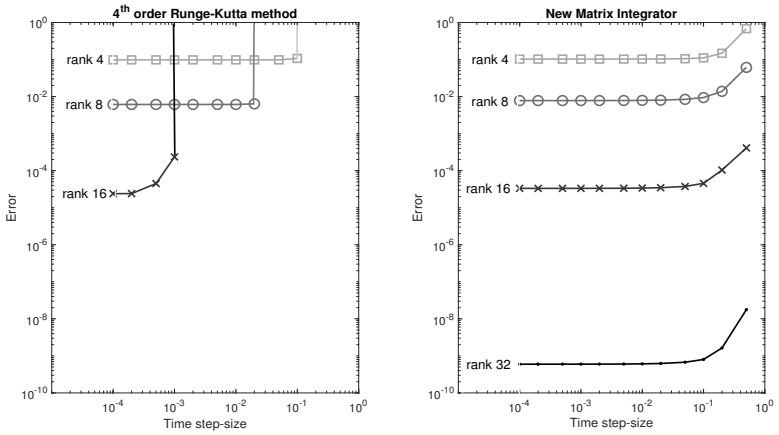


Fig. 1 Comparison of the explicit Runge Kutta method (left) and the proposed new integrator (right) for different approximation ranks and step sizes in the case of a given time-dependent matrix.

6.2 Error behaviour

In the second example, we integrate a (non-stiff) discrete Schrödinger equation in imaginary time,

$$\dot{Y} = -\mathrm{H}[Y], \quad Y(t_0) = C_0 \bigtimes_{i=1}^{d} \mathbf{U}_i^0 \ .$$

Here,

$$\begin{aligned}
&\mathrm{H}[Y] = -\frac{1}{2}\sum_{j=1}^{d} \left(Y \times_j \mathbf{D}\right) + Y \bigtimes_{i=1}^{d} \mathbf{V}_{cos} \in \mathbb{R}^{N\times\cdots\times N}, \\
&\mathbf{D} = \texttt{tridiag}(-1, 2, -1) \in \mathbb{R}^{N\times N}, \\
&\mathbf{V}_{\cos} := \mathrm{diag}\{1 - \cos(\frac{2\pi j}{N})\}, \quad j = -N/2, \ldots, N/2 - 1 \ .
\end{aligned}$$

The function H arises from the Hamiltonian $\mathcal{H} = -\frac{1}{2}\Delta_{\mathrm{discrete}} + V(x)$ on a equidistant space grid with the torsional potential $V(x_1, \ldots, x_d) = \prod_{i=1}^{d}(1 - \cos(x_i))$.

For each $i = 1, \ldots, d$, the orthonormal matrices $\mathbf{U}_i^0 \in \mathbb{R}^{N\times N}$ are randomly generated. The core tensor $C_0 \in \mathbb{R}^{N\times N\times N}$ has only non-zero diagonal elements set equal to $(C_0)_{jjj} = 10^{-j}$ for $j = 1, \ldots N$ in the case $d = 3$, and analogously in the matrix case $d = 2$.

The reference solution was computed with the Matlab solver `ode45` and stringent tolerance parameters {'RelTol', 1e-10, 'AbsTol', 1e-10} . The differential equations appearing in the definition of a step of the new matrix and Tucker integrators have all been solved either with a single step of a second- or fourth-order explicit Runge–Kutta method.

We choose $N = 100$, final time $T = 0.1$ and $d = 2, 3$. The multi-linear rank is chosen such that $r_1 = r_2 = \cdots = r_d$. The singular values of the matricization in the first mode of the reference solution and the absolute errors $\|Y_n - A(t_n)\|_F$ at time $t_n = T$ of the approximate solutions for different ranks, calculated with

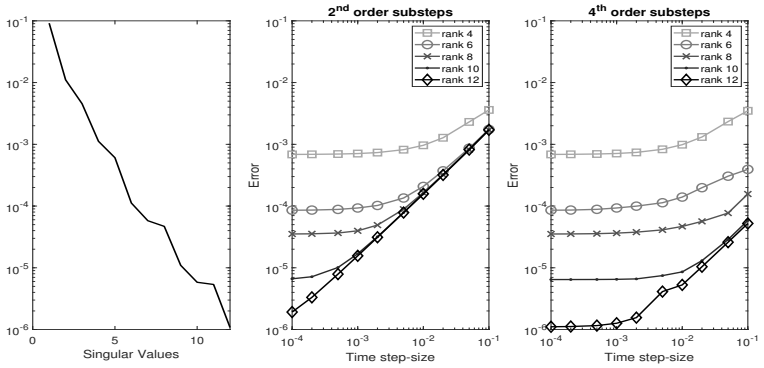


Fig. 2 First twelve singular values of the reference solution at time $T = 0.1$ and approximation errors for different ranks, time-integration methods in the substeps of the new matrix integrator, and step-sizes for the matrix differential equation ($d = 2$).

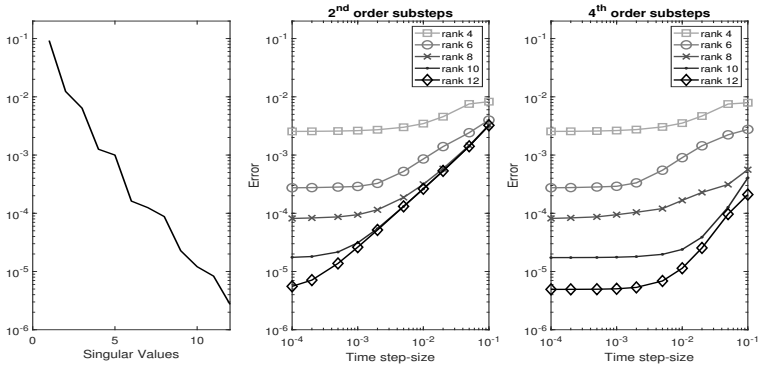


Fig. 3 First twelve singular values of the matricisization in first mode of the reference solution at time $T = 0.1$ and approximation errors for different multi-linear ranks, time-integration methods in the substeps of the new fixed-rank Tucker tensor integrator and step-sizes for the tensor differential equation ($d = 3$).

different step-sizes and different time integration methods, are shown in Figure 2 for the matrix case($d = 2$), and in Figure 3 for the tensor case($d = 3$). The figures clearly show that solving the substeps with higher accuracy allows us to take larger step-sizes to achieve a prescribed error.

6.3 Comparison with the matrix projector-splitting integrator over different ranks

In the last example, we compare the matrix projector splitting integrator with the new matrix integrator of Section 3. Here, the complex case is considered: in the definition of the sub-problems appearing in the new matrix integrator, it is sufficient to replace the transpose with the conjugate transpose.

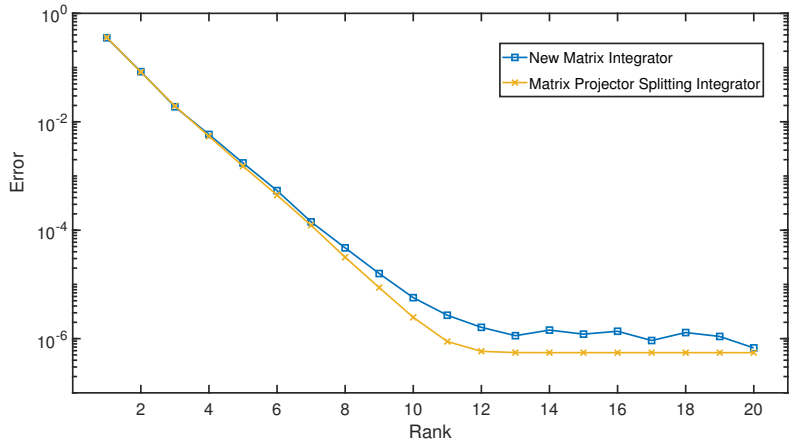


Fig. 4 Approximation errors for different ranks at final time $T = 5$ of the low-rank approximation computed with the matrix projector splitting integrator and the new matrix integrator.

We consider a Schrödinger equation as in [11, Section 4.3],

$$
\begin{aligned}
&i\partial_t u(x,t) = -\frac{1}{2}\Delta u(x,t) + \frac{1}{2}x^\top \mathbf{A} x\, u(x,t), \quad x \in \mathbb{R}^2, t > 0,\\
&u(x,0) = \pi^{-\frac{1}{2}} \exp\big(\frac{1}{2}x_1^2 + \frac{1}{2}(x_2-1)^2\big),\\
&\mathbf{A} = \begin{pmatrix} 2 & -1 \\ -1 & 3 \end{pmatrix}.
\end{aligned}
$$

The problem is discretized with a Fourier collocation method with a grid of $N \times N$ points; the solution is essentially supported within $\Omega = [-7.5, 7.5]^2$. We choose the final time $T = 5$ and $N = 128$, which makes the problem moderately stiff. First, we compute a reference solution with an Arnoldi method and a tiny time-step size $h = 10^{-4}$. Then, for each rank from 1 until 20, we compute a low-rank approximation with the matrix projector splitting integrator and the new matrix integrator. The lower-dimensional sub-problems appearing in the definition of the two integrators are solved with an Arnoldi method and time-step size $h = 0.005$. For each rank, the absolute error in Frobenius norm of the two given approximations at the final time $T = 5$, with respect to the reference solution, are shown in Figure 4.

Acknowledgements The last numerical example is based upon the original source code of [11, Section 4.3]; we would like to thank Hanna Walach for kindly providing it.

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) — Project-ID 258734477 — SFB 1173.

References

1. M. Bachmayr, H. Eisenmann, E. Kieri, and A. Uschmajew. Existence of dynamical low-rank approximations to parabolic problems. *Math. Comp.*, page https://doi.org/10.1090/mcom/3626, 2021.

2. M. H. Beck, A. Jäckle, G. A. Worth, and H.-D. Meyer. The multiconfiguration time-dependent Hartree (MCTDH) method: a highly efficient algorithm for propagating wavepackets. *Physics reports*, 324(1):1–105, 2000.
3. M. H. Beck and H.-D. Meyer. An efficient and robust integration scheme for the equations of motion of the multiconfiguration time-dependent Hartree (MCTDH) method. *Z. Physik D*, 42(2):113–129, 1997.
4. M. Billaud-Friess, A. Falcó, and A. Nouy. A geometry based algorithm for dynamical low-rank approximation. *arXiv preprint arXiv:2001.08599*, 2020.
5. G. Ceruti and C. Lubich. Time integration of symmetric and anti-symmetric low-rank matrices and Tucker tensors. *BIT Numer. Math.*, 60:591–614, 2020.
6. G. Ceruti, C. Lubich, and H. Walach. Time integration of tree tensor networks. *SIAM J. Numer. Anal.*, 59(1):289–313, 2021.
7. L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.
8. J. Haegeman, C. Lubich, I. Oseledets, B. Vandereycken, and F. Verstraete. Unifying time evolution and optimization with matrix product states. *Phys. Rev. B*, 94(16):165116, 2016.
9. E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I. Nonstiff problems*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1993.
10. B. N. Khoromskij, I. V. Oseledets, and R. Schneider. Efficient time-stepping scheme for dynamics on TT-manifolds. *Preprint 2012-24, MPI Math. Naturwiss. Leipzig*, 2012.
11. E. Kieri, C. Lubich, and H. Walach. Discretized dynamical low-rank approximation in the presence of small singular values. *SIAM J. Numer. Anal.*, 54(2):1020–1038, 2016.
12. O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM J. Matrix Anal. Appl.*, 29(2):434–454, 2007.
13. O. Koch and C. Lubich. Dynamical tensor approximation. *SIAM J. Matrix Anal. Appl.*, 31(5):2360–2375, 2010.
14. T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, 2009.
15. C. Lubich. Time integration in the multiconfiguration time-dependent Hartree method of molecular quantum dynamics. *Appl. Math. Res. Express. AMRX*, 2015(2):311–328, 2015.
16. C. Lubich and I. V. Oseledets. A projector-splitting integrator for dynamical low-rank approximation. *BIT*, 54(1):171–188, 2014.
17. C. Lubich, I. V. Oseledets, and B. Vandereycken. Time integration of tensor trains. *SIAM J. Numer. Anal.*, 53(2):917–941, 2015.
18. C. Lubich, B. Vandereycken, and H. Walach. Time integration of rank-constrained Tucker tensors. *SIAM J. Numer. Anal.*, 56(3):1273–1290, 2018.
19. N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer. Tensorlab 3.0. *available online, URL: www. tensorlab. net*, 2016.