Karlsruhe Institute of Technology

Wave phenomena
analysis and numerics

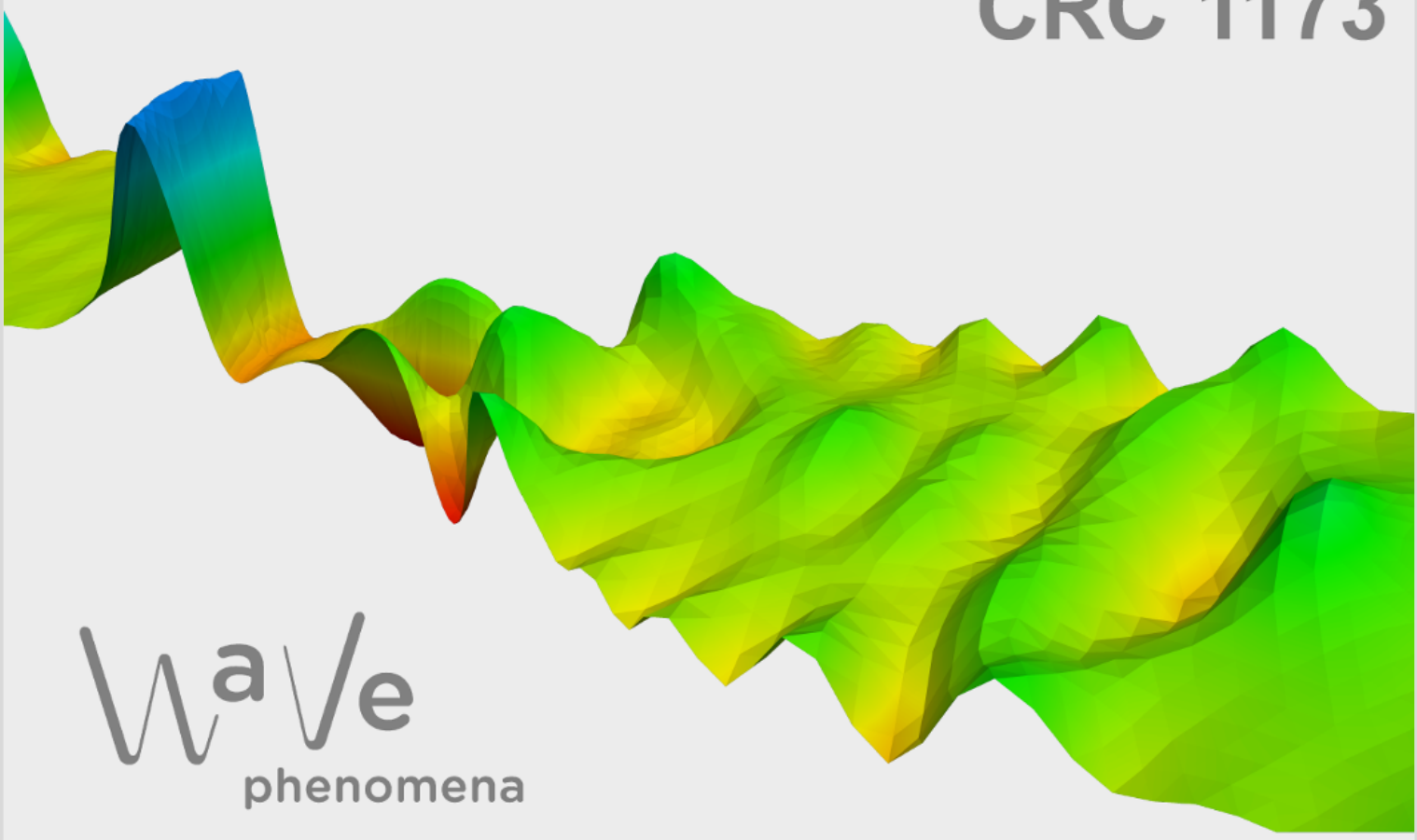
Time integration of tree tensor networks

Gianluca Ceruti, Christian Lubich, Hanna Walach

CRC Preprint 2020/5, February 2020

KARLSRUHE INSTITUTE OF TECHNOLOGY

CRC 1173



Participating universities

Universität Stuttgart

Funded by

ISSN 2365-662X

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 to large time-dependent data tensors and unknown solutions of 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 Tucker tensors and is shown to inherit their favorable properties. The integrator is based on recursively applying the 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 $A(t) \in \mathbb{R}^{n_1 \times \dots \times n_d}$,

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

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., [14, 15]. 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) 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 [12] and was used in [17] 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 [11]. The projector-splitting integrator was extended to tensor trains / matrix product states in [18]; see also [9] for a description of the algorithm in a physical idiom. The projector-splitting integrator was extended to Tucker tensors of fixed multilinear rank in [16]. A reinterpretation was given in [19], 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 [11, 19]. 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.

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 [19] 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 [19], which in turn was proved using the exactness property for the matrix projector-splitting integrator that was discovered and proved in [17].

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 [19], which in turn relies on such an error bound for the fixed-rank matrix projector-splitting integrator proved in [11], 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 [24]. 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 [17], 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 [17, 18, 19], 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 precisely this integrator (given in an *ad hoc* formulation) are already reported in [4] for the Vlasov–Poisson equation of plasma physics, for a tree tensor network where 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.

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

2.1. Matrices of rank r. The singular value decomposition 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 real $m \times n$ matrices of rank (exactly) r are known to form a smooth embedded manifold in $\mathbb{R}^{m\times n}$ [10].

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 $(r_1, \ldots, r_d)$ 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 $i = 1, \ldots, d$, where $n_{\neg i} = \prod_{j\neq i} n_j$. We recall that the ith matricization aligns in the kth row (for $k = 1, \ldots, n_i$) all entries of A that

have the index k in the ith position, usually ordered co-lexicographically. The inverse operation is retensorisation 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 [3] that the tensor A has multilinear rank $(r_1, \ldots, r_d)$ if and only if it can be factorized as a *Tucker tensor* (we adopt the shorthand notation from [13])

$$A = C \bigtimes_{i=1}^{d} \mathbf{U}_i, \qquad \text{i.e.,} \quad a_{k_1,\ldots,k_d} = \sum_{l_1=1}^{r_1} \cdots \sum_{l_d=1}^{r_d} c_{l_1,\ldots,l_d} u_{k_1,l_1} \ldots u_{k_d,l_d}, \tag{2.1}$$

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 $(r_1, \ldots, r_d)$. (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.)

A useful formula for the matricization of Tucker tensors is

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

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

The tensors of given dimensions $(n_1, \ldots, n_d)$ and fixed multilinear rank $(r_1, \ldots, r_d)$ 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 [22], but tree tensor networks were actually already used a few years earlier in the multilayer MCTDH method of chemical physics [24]. In the mathematical literature, tree tensor networks with binary trees have been studied as 'hierarchical tensors' [8] and with general trees as tensors in 'tree-based tensor format' [5, 6]. We remark that Tucker tensors and matrix product states / tensor trains [21, 20] are particular instances of tree tensor networks, whose trees are trees of minimal height (bushes) and binary trees of maximal height, respectively. As there does not appear to exist a firmly established notation for tree tensor networks, we give a formulation from scratch that we find useful for our purposes.

DEFINITION 2.1 (Ordered trees with distinct leaves). *Let $\mathcal{L}$ be a given finite set, to which we refer as the set of leaves. We define the set $\mathcal{T}$ of trees with leaves in $\mathcal{L}$ recursively as follows:*

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

(ii) If, for some $m \ge 2$,

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

then the ordered m*-tuple*

$$\tau := (\tau_1, \ldots, \tau_m) \in \mathcal{T}, \qquad L(\tau) := \dot{\bigcup_{i=1}^{m}} L(\tau_i) \ .$$

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 distinct leaves. (Note that $m = 1$ is excluded: the 1-tuple $\tau = (\tau_1)$ is considered as identical to τ_1.) $L(\tau)$ is the set of leaves of the tree τ.

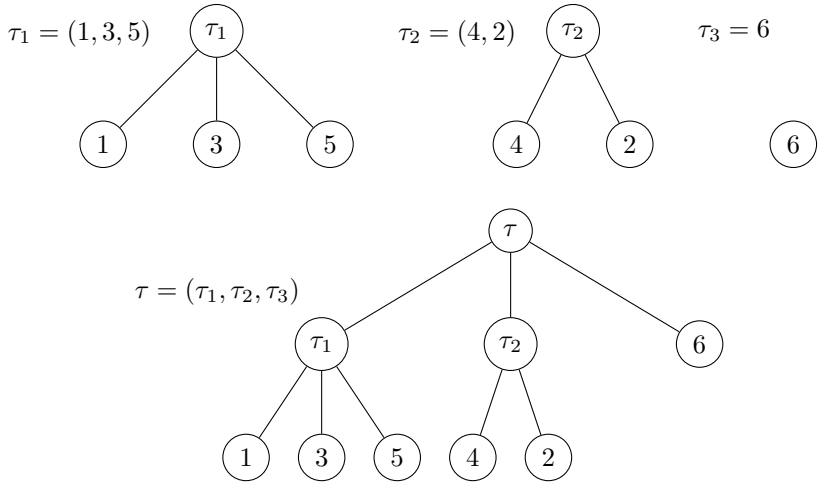


FIG. 2.1. *Graphical representation of a tree and three subtrees with the set of leaves* $\mathcal{L} = \{1,2,3,4,5,6\}$.

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

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

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 $\sigma, \tau \in \mathcal{T}$,

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

On a given tree $\bar\tau \in \mathcal{T}$, we work with the following set of data:

- To each leaf 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.
- 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.

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* $\bar\tau \in \mathcal{T}$*, we recursively define a tree tensor network* $X_{\bar\tau}$ *as follows:*

(i) If $\tau = \ell \in \mathcal{L}$*, we set*

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

(ii) If $\tau = (\tau_1, \dots, \tau_m) \le \bar\tau$, we set $n_\tau = \prod_{i=1}^m n_{\tau_i}$ and $\mathbf{I}_\tau$ the identity matrix of dimension r_τ, and

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

We note that the expression on the righthand side of the definition of X_τ 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 $k = 1, \dots, r_\tau$. 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.

It is favorable to work with orthonormal matrices, so that each tensor X_τ is in the Tucker tensor 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 [24].

A further consequence of Lemma 2.4 is that every tree tensor network has an orthonormal representation. This is shown using a QR decomposition of non-orthonormal matrices $\mathbf{Mat}_0(C_{\tau_i})^\top$ and including the non-orthonormal factor in the tensor C_τ of the parent tree $\tau = (\tau_1, \dots, \tau_m)$. It is of full tree rank (r_τ) if all these matrices are of full rank.

We rely on the following property, which we can be proved as in [23], where binary trees are considered (this corresponds to the case $m = 2$ above); see also [5].

LEMMA 2.5. *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 tree tensor networks for a tree $\bar\tau \in \mathcal{T}$ of given dimensions $(n_\ell)_{\ell \in L(\bar\tau)}$ and ranks $(r_\tau)_{\tau \in T(\bar\tau)}$ 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 [16, 19], 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 Tucker Tensors 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,

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

The nested 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 in the following algorithmic scheme.

Algorithm 1: Time step of the Tucker integrator

Data: Core tensor C^0 and orthonormal-basis matrices $\mathbf{U}_i^0$ of $Y^0 = C^0 \bigtimes_{i=1}^{d} \mathbf{U}_i^0$, righthand side function $F(t,Y)$, t_0, t_1
Result: Core tensor C^1 and orthonormal-basis matrices $\mathbf{U}_i^1$ of $Y^1 = C^1 \bigtimes_{i=1}^{d} \mathbf{U}_i^1$

```
begin
    Set C_0^0 = C^0
    for i = 1 ... d do
        Update U_i^0 → U_i^1, Modify C_{i-1}^0 → C_i^0
    end
    Update C_d^0 → C^1
end
```

The update process in the Tucker integrator is the most intensive and technical part of the algorithm. In order to reduce the complexity of the explanation we introduce the subflows $\Phi^{(i)}$ and Ψ, corresponding to the update of the basis matrices and of the core tensor, respectively. We set $r_{\neg i} = \prod_{j \neq i} r_i$.

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

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 $\mathbf{U}_j^1 = \mathbf{U}_j^0 \quad \forall j = 1, \ldots, d.$

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 [19] 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 $(1+d)$-dimensional Tucker tensor 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 holds.

LEMMA 3.1. *The action of the subflow $\Phi^{(0)}$ on Y^0 is 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 compute the corresponding QR decomposition,

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

To conclude, we solve the differential equation

$$\begin{aligned} \dot{\mathbf{S}}_0(t) &= -\, \mathbf{I}_r^\top \, \mathbf{Mat}_0(F(t, \mathrm{Ten}_0(\mathbf{I}_r \, \mathbf{S}_0(t)\mathbf{V}_0^{0,\top}))\mathbf{V}_0^0, \\ \mathbf{S}_0(t_0) &= \mathbf{K}_0^1 \, . \end{aligned}$$

We now observe that due to the presence of the negative sign, the solution of the equation can be directly computed, i.e.

$$\mathbf{S}_0(t_1) = \mathbf{K}_0(t_0) = \mathbf{S}_0^0.$$

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

During the update process the structure of the initial data is preserved and we can now introduce the extended Tucker integrator as follows.

Algorithm 4: 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

  end

  Compute $Y^1 = \Psi(Y^{[d]})$ in factorized form

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 updates the orthonormal-basis matrices of the leaves and the orthonormality-constrained core tensors of the other vertices of the tree by a recursive TTN integrator.

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.

This leads to the definition of the recursive tree tensor network (TTN) integrator. It has the same general structure as the extended Tucker integrator.

Algorithm 5: Recursive TTN Integrator

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 $Y_\tau^{[0]} = Y_\tau^0$
  for $i = 1 \ldots m$ **do**
    compute $Y_\tau^{[i]} = \Phi_\tau^{(i)}(Y_\tau^{[i-1]})$ in factorized form
  end
  compute $Y_\tau^1 = \Psi_\tau(Y_\tau^{[m]})$ in factorized form
end

The difference to the Tucker integrator is that now the subflow $\Phi_\tau^{(i)}$ is no longer the same as for the extended Tucker integrator, 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.

Algorithm 6: 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 $Y_{\tau_i}^1 = \textit{Recursive TTN Integrator}\ (\tau_i, Y_{\tau_i}^0, F_{\tau_i}, t_0, t_1)$

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 $\mathbf{U}_{\tau_i}^1 = \mathbf{Mat}_0(X_{\tau_i}^1)$, 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 7: 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 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_τ^0 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 $\Phi_\tau^{(i)}$ of the recursive TTN integrator. For the initial data $Y_\tau^0=C_\tau^0\times_0\mathbf{I}_\tau\bigtimes_{i=1}^m\mathbf{U}_{\tau_i}^0$ the subflow $\Phi_\tau^{(i)}$ given by Algorithm 6 first computes the QR decomposition

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

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

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

we then have the SVD-like decomposition

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

with the (computationally inaccessible) matrix

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

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

Like in Algorithm 2 for the subflow $\Phi^{(i)}$ of the Tucker integrator, we consider the differential equation for $\mathbf{K}_{\tau_i}(t)\in\mathbb{R}^{n_{\tau_i}\times r_{\tau_i}}$,

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

where

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

Algorithm 6 retensorizes this differential equation and solves it approximately by recurrence down to the leaves. By definition of the tree tensor network, there exists $X_{\tau_i}^0\in\mathcal{V}_{\tau_i}$ such that

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

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 6. We introduce

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

By substitution, (4.3) can be rewritten as

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

where

$$\begin{aligned} F_{\tau_i}(t, Y_{\tau_i}) &= \mathrm{Ten}_0\big(\mathbf{F}_{\tau_i}(t, \mathbf{Mat}_0(Y_{\tau_i})^\top\big) \\ &= \mathrm{Ten}_0\Big(\big(\mathbf{Mat}_i(\ F_\tau(t, \mathrm{Ten}_i(\mathbf{Mat}_0(Y_{\tau_i})^\top \mathbf{V}^{0,\top}_{\tau_i})))\mathbf{V}^0_{\tau_i}\big)^\top\Big)\ . \end{aligned} \tag{4.4}$$

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

$$\pi_{\tau,i}(Y_{\tau_i}) := \mathrm{Ten}_i\big((\mathbf{V}^0_{\tau_i}\,\mathbf{Mat}_0(Y_{\tau_i}))^\top\big) \in \mathcal{V}_\tau \quad \text{ for } \ Y_{\tau_i} \in \mathcal{V}_{\tau_i} \tag{4.5}$$

and the *restriction*

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

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,*

$$\pi^\dagger_{\tau,i}(\pi_{\tau,i}(Y_{\tau_i})) = Y_{\tau_i} \qquad \textit{for } \ Y_{\tau_i} \in \mathcal{V}_{\tau_i} \tag{4.7}$$

$$\langle \pi_{\tau,i}(Y_{\tau_i}), Z_\tau \rangle_{\mathcal{V}_\tau} = \langle Y_{\tau_i}, \pi^\dagger_{\tau,i}(Z_\tau) \rangle_{\mathcal{V}_{\tau_i}} \quad \textit{for } \ Y_{\tau_i} \in \mathcal{V}_{\tau_i},\ Z_\tau \in \mathcal{V}_\tau. \tag{4.8}$$

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 (4.7). 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 (4.8). The norm equality follows from the definition (4.5) and the fact that the matrix $\mathbf{V}^0_{\tau_i}$ has orthonormal columns. The norm bound follows from (4.6) on noting the general matrix norm inequality $\|\mathbf{AB}\|_F \le \|\mathbf{A}\|_2\,\|\mathbf{B}\|_F$ and the fact that $\|\,\mathbf{V}^{0,\top}_{\tau_i}\,\|_2 = 1$. ∎

We emphasize that the mappings $\pi_{\tau,i}$ and $\pi^\dagger_{\tau,i}$ depend on the initial data Y^0_τ. We observe that we can write (4.4) more compactly as $F_{\tau_i} = \pi^\dagger_{\tau,i} \circ F_\tau \circ \pi_{\tau,i}$. For the initial data we find from (4.2) and (4.3) that

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

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* $Y^0_{\bar\tau} \in \mathcal{M}_{\bar\tau}$*, we define recursively for each tree* $\tau = (\tau_1, \dots, \tau_m) \in T(\bar\tau)$ *and for* $i = 1, \dots, m$

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

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. [7].

An important observation is the following.

LEMMA 4.3. *If the initial tree tensor network* $Y^0_{\bar\tau}$ *has full tree rank* $(r_\sigma)_{\sigma\le\bar\tau}$*, then* Y^0_τ *has full tree rank* $(r_\sigma)_{\sigma\le\tau}$ *for every subtree* $\tau \le \bar\tau$*.*

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

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

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

In terms of the manifold (see Lemma 2.5)

$$\mathcal{M}_\tau = \mathcal{M}(\tau, (n_\ell)_{\ell\in L(\tau)}, (r_\sigma)_{\sigma\le\tau}) \tag{4.9}$$

of tree tensor networks for the tree $\tau \in \mathcal{T}$ of given dimensions $(n_\ell)_{\ell\in L(\tau)}$ and full tree rank $(r_\sigma)_{\sigma\le\tau}$, Lemma 4.3 can be restated as saying that for $\tau = (\tau_1, \dots, \tau_m)$ and $Y^0_\tau \in \mathcal{M}_\tau$, the restriction $\pi^\dagger_{\tau,i}(Y^0_\tau)$ 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^0_τ (recall that the chosen restriction operator $\pi^\dagger_{\tau,i}$ depends on Y^0_τ). In particular, a loss of rank occurs if for some j, the basis matrices are such that $\mathbf{U}^{0,\top}_{\tau_j}\, \mathbf{U}_{\tau_j}$ is a singular $r_{\tau_j} \times r_{\tau_j}$ matrix. However, for the prolongation we have the following.

LEMMA 4.4. *Let* $\tau = (\tau_1, \dots, \tau_m) \in \mathcal{T}$ *and* $i = 1, \dots, m$*. 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 $\mathbf{V}^0_{\tau_i}$ and writing $\mathbf{U}_{\tau_i} := \mathbf{Mat}_0(Y_{\tau_i})^\top$,

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

which is of full tree rank. ∎

4.3. QR decomposition. Given a tree $\tau = (\tau_1, \dots, \tau_m) \in \mathcal{T}$, a critical step in the recursive TTN integrator is computing the QR-decomposition of $\mathbf{K}_\tau(t_1)$.
The matrix $\mathbf{K}_\tau(t_1)$ can be extremely large and this may affect the computational cost of the recursive TTN integrator. This difficulty is overcome if the tree tensor

network is orthonormal: the QR-decomposition of the full matrix is equivalent to the QR-decomposition of the matricization of a small core tensor. In fact, by construction we have that

$$\mathbf{K}_\tau(t_1) = \mathbf{Mat}_0(Y_\tau)^\top,$$

where

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

This implies that

$$\mathbf{K}_\tau(t_1) = \Big(\bigotimes_{i=1}^m \mathbf{U}_{\tau_i} \Big) \mathbf{Mat}_0(C_\tau)^\top.$$

We recall that the Kronecker product of orthonormal matrices is orthonormal. To conclude, it suffices to perform a QR-decomposition of the small matrix $\mathbf{Mat}_0(C_\tau)^\top \in \mathbb{R}^{r_{\tau_1} \cdots r_{\tau_m} \times r_\tau}$.

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 $Y \in \mathcal{V}_{\tau_i}$ is a new larger tree tensor having core $\mathrm{Ten}_i(\mathbf{Q}_{\tau_i}^{0,\top})$.

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 $\tau = (\tau_1, \ldots, \tau_m)$ with the same dimensions n_τ and n_{τ_i} but possibly different bond dimensions 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

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

If we define the small matrix

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

and we recall that by definition of the tree tensor network,

$$\exists Z_{\tau_i} \in \mathcal{V}_{\tau_i} \;:\; \mathbf{W}_{\tau_i} = \mathbf{Mat}_0(Z_{\tau_i})^\top,$$

we have that,

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

The action of the restriction $\pi_{\tau_i}^\dagger$ on the tree tensor network $Z_\tau \in \mathcal{V}_\tau$ is reduced to the multiplication of the small core tensor G_{τ_i} of Z_{τ_i} with the tiny matrix $\mathbf{R}_{\tau_i}$.

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 [17], tensor trains / matrix product states [18], and Tucker tensors [19]. 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 $Y_{\bar\tau}^0 = A(t_0)$, for which we consider the restricted tensor networks $A_\tau(t) := \big(A(t)\big)_\tau$ defined by the restrictions (4.6) associated with $Y_{\bar\tau}^0$ 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 $t_1 > t_0$:

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

THEOREM 5.1 (Exactness). *Let $A(t)$ be a continuously differentiable time-dependent family of tree tensor networks $A(t)$ of full tree rank $(r_\tau)_{\tau\leq\bar\tau}$ for $t_0 \leq t \leq t_1$, 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 [19] 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 $\tau = (\tau_1, \ldots, \tau_m) \in \mathcal{T}$, then we set $h(\tau) = 1 + \max\{h(\tau_1), \ldots, h(\tau_m)\}$.

We note that, since the restricitions $\pi_\tau^\dagger$ for $\tau \leq \bar\tau$ 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 $\tau = (\tau_1, \ldots, \tau_m)$ 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 $(r_\tau, r_{\tau_1}, \ldots, r_{\tau_m})$ 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 [19] and hence reproduces $A_\tau(t_1)$ exactly by [19, Theorem 4.1].

(ii) For trees of height $k \geq 2$ 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 $\tau = (\tau_1, \ldots, \tau_m)$ 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 $(r_\tau, r_{\tau_1}, \ldots, r_{\tau_m})$ at both $t = t_0$ and $t = t_1$. From the exactness result of [19, 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 $\bar{\tau}$, 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 [19], which in turn was based on the corresponding result for matrices proved in [11]. We recall the notation $\mathcal{V}_\tau$ for the tensor space (4.1) and $\mathcal{M}_\tau$ for the tree tensor network manifold (4.9). 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,

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

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

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 $\varepsilon > 0$,

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

for all $(t, Y) \in [0, t^*] \times \mathcal{M}$ in some neighborhood of the exact solution $(t, A(t))$.

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:

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

We write Y^n for the numerical approximation obtained after n time steps of the TTN integrator with step size $h > 0$.

THEOREM 6.1. *Under the above assumptions, the errors of the recursive TTN integrator at $t_n = nh$ are 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 proof works recursively, based on the corresponding result for Tucker tensors given in [19] 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 $\tau \leq \bar{\tau}$, 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 $\tau \leq \bar{\tau}$, with the same L, B, and ε,*

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

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

and

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

for all $(t, Y_\tau) \in [0, t^*] \times \mathcal{M}_\tau$ in some neighborhood of the restricted exact solution.

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

LEMMA 6.3. *Let $\tau = (\tau_1, \ldots, \tau_m) \in \mathcal{T}$ and $i = 1, \ldots, m$. 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:

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

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,

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

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

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

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

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

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

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

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 $t_0 \leq t \leq t^*$ 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 [11] and [19].

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 [19], for which the error estimate has been proved in [19].

For trees $\tau = (\tau_1, \ldots, \tau_m)$ 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 [11, 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. In various specific tree formats, the recursive TTN integrator has already been applied to problems appearing in kinetic theory and quantum dynamics. The numerical integrator presented in this paper generalizes and provides a systematic theoretical approach to the results presented in [4, 18, 19]. In the following numerical examples, we choose the tree of Figure 2.1.

The dimensions of the leaves and the ranks are taken the same for all the nodes and are fixed to $n = 16$ and $r = 5$.

7.1. Tree tensor addition. Let $\bar{\tau}$ be the given tree with associated rank $r_{\bar{\tau}} = 1$ and let $A \in \mathcal{M}_{\bar{\tau}}$ and $B \in \mathcal{T}_A\mathcal{M}_{\bar{\tau}}$. We consider the addition of two given tensors,

$$C = A + B,$$

where $C \in \mathcal{V}_{\bar{\tau}}$ is a tree tensor network not necessarily of the same rank and we want to compute a tree tensor network retraction to $\mathcal{M}_{\bar{\tau}}$. 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 the retraction 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.

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.

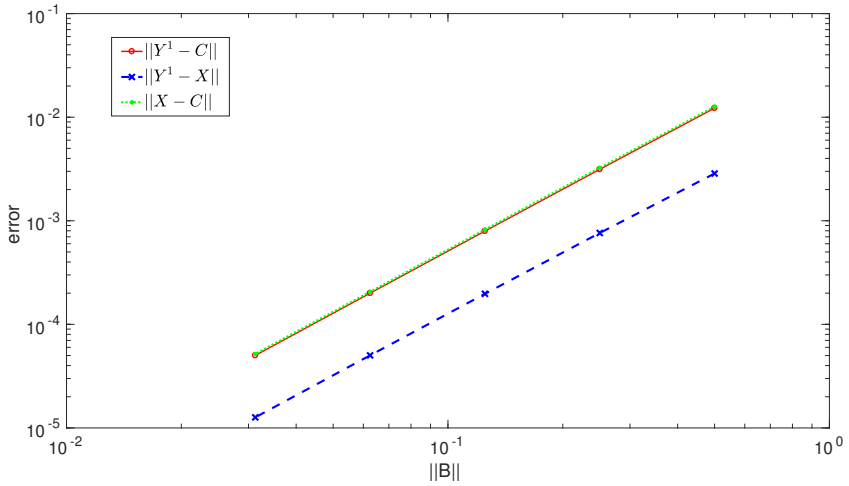


FIG. 7.1. *Error of retracted tree tensor network sum.*

We observe that decreasing the norm of the tensor B reduces the approximation error as expected, proportional to $\|B\|^2$.

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 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

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

and the connection tensors changed according to

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

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.

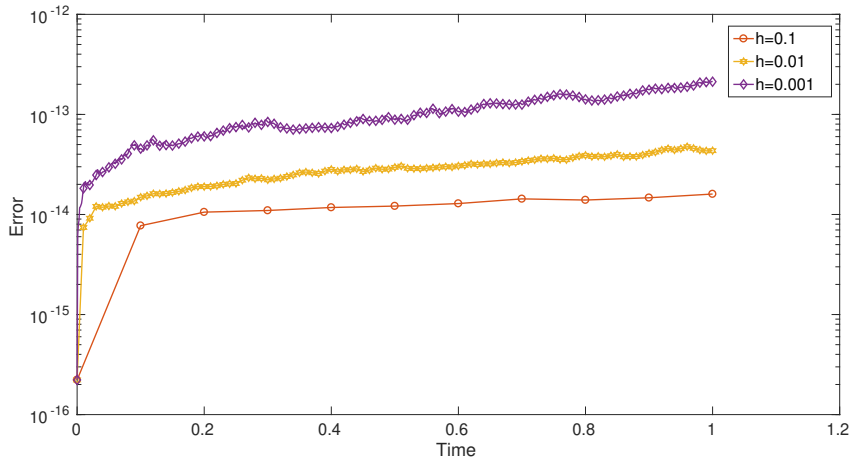


FIG. 7.2. *Error vs. time in a case of exactness up to round-off errors.*

Acknowledgements. 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] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21:1253–1278, 2000.
[4] 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.
[5] A. Falcó, W. Hackbusch, and A. Nouy. Geometric structures in tensor representations (final release). *arXiv preprint arXiv:1505.03027*, 2015.
[6] A. Falcó, W. Hackbusch, and A. Nouy. Tree-based tensor formats. *SeMA J.*, pages 1–15, 2018.
[7] W. Hackbusch. *Multigrid methods and applications*, volume 4 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1985.
[8] W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*. Springer, 2012.
[9] 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.
[10] U. Helmke and J. B. Moore. *Optimization and dynamical systems*. Communications and Control Engineering Series. Springer-Verlag, London, 1994.
[11] E. Kieri, C. Lubich, and H. Walach. Discretized dynamical low-rank approximation in the presence of small singular values. *SIAM J. Numer. Anal.*, 54:1020–1038, 2016.

[12] O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM J. Matrix Anal. Appl.*, 29(2):434–454, 2007.
[13] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51:455–500, 2009.
[14] 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.
[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*, 2015:311–328, 2015.
[17] C. Lubich and I. V. Oseledets. A projector-splitting integrator for dynamical low-rank approximation. *BIT*, 54:171–188, 2014.
[18] C. Lubich, I. V. Oseledets, and B. Vandereycken. Time integration of tensor trains. *SIAM J. Numer. Anal.*, 53:917–941, 2015.
[19] C. Lubich, B. Vandereycken, and H. Walach. Time integration of rank-constrained Tucker tensors. *SIAM J. Numer. Anal.*, 56:1273–1290, 2018.
[20] I. V. Oseledets. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33(5):2295–2317, 2011.
[21] 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.
[22] 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.
[23] A. Uschmajew and B. Vandereycken. The geometry of algorithms using hierarchical tensors. *Linear Algebra Appl.*, 439(1):133–166, 2013.
[24] H. Wang and M. Thoss. Multilayer formulation of the multiconfiguration time-dependent Hartree theory. *J. Chem. Phys.*, 119(3):1289–1299, 2003.