Time integration of Tree Tensor Networks

Gianluca Ceruti, EPF Lausanne.
(based on joint work with Ch. Lubich and H. Walach)

Napoli, 15 Febbraio 2022.
Problem

We consider

\[ \dot{A}(t) = F(t, A(t)), \quad A(t_0) = A_0 \in \mathbb{C}^{n_1 \times \cdots \times n_d} \]

arising from, e.g.
- discrete Schrödinger equation \((d = 3N\text{ particles})\).
- discrete kinetic equations \((d = 6)\).

Direct computational treatment is infeasible for large \(d\) and/or \(n_i\).

We start from the real case \(d = 2\).
Dynamical low-rank approximation

Given a matrix differential equation

\[ \dot{A}(t) = F(t, A(t)), \quad A(t_0) = A_0 \in \mathbb{R}^{m \times n} \]

we aim to approximate \( A(t) \approx Y(t) \in \mathcal{M}_r \) (rank \( r \)) by requiring

\[ \dot{Y}(t) \in T_{Y(t)} \mathcal{M}_r \ 	ext{such that} \ ||\dot{Y}(t) - F(t, Y(t))|| = \text{min!} \]

\[ F(Y) \]

\[ T_{Y,M} \]

Koch, Lubich 2007
Recalling

\[ \dot{Y}(t) \in T_{Y(t)} \mathcal{M}_r \quad \text{such that} \quad \| \dot{Y}(t) - F(t, Y(t)) \| = \min! \]

it is equivalent to

\[ \dot{Y} = P(Y) F(t, Y), \]

where \( P(Y) \) is the orthogonal projection onto the tangent space at \( Y = USV^\top \) given by

\[ P(Y)Z = ZV V^T - UU^T ZV V^T + UU^T Z. \]
Matrix projector-splitting integrator

Idea: Split $P(Y)$ into its three parts, solve separately.

Efficiently implementable integrator such that

+ Reproduces rank-$r$ matrices exactly.
+ Robust error bound (independent of small singular values).
- Backward substep (problematic for dissipative problems).

Lubich, Oseledets 2014
Kieri, Lubich, Walach 2016
From matrices to tree tensor networks

- Low-rank matrices
  - Tucker tensors
    - Tree tensor networks
Low-rank matrices

Let \( Y \in \mathbb{R}^{n_1 \times n_2} \) be a matrix of multilinear rank \((r_1, r_2)\)

\[
Y = U_1 S U_2^\top \in \mathbb{R}^{n_1 \times n_2}
\]

where

\[
U_i \in \mathbb{R}^{n_i \times r_i}, \quad \forall i = 1, 2,
\]

\[
S \in \mathbb{R}^{r_1 \times r_2}.
\]
Tucker tensors

Let $Y \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a tensor of multilinear rank $(r_1, \ldots, r_d)$

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

where

$$\mathbf{U}_i \in \mathbb{R}^{n_i \times r_i} \quad \forall i = 1, \ldots, d,$$

$$\mathbf{C} \in \mathbb{R}^{r_1 \times \cdots \times r_d}.$$
Starting from $Y^0 = C^0 X_{i=1}^d U_i^0$:

Set $C_0^0 = C^0$.

For $i = 1, \ldots, d$, update $U_i^0 \rightarrow U_i^1$ and modify $C_{i-1}^0 \rightarrow C_i^0$.

Update $C_d^0 \rightarrow C^1$.

After one time step, this yields $Y^1 = C^1 X_{i=1}^d U_i^1$.
Curse of the dimensionality of the rank

Let $C \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ and $r := \max(r_1, r_2, \ldots, r_d)$,

$$\text{size} \ (C) = r^d.$$ 

The size of the core tensor grows exponentially with $d$. 
Tree tensor network - Preparation:
Graphical representation

Figure: Graphical representation of a tree $\bar{\tau}$ with three subtrees and set of leaves $\mathcal{L} = \{1, 2, 3, 4, 5, 6\}$. 
Tree tensor network - Preparation:
Graphical representation

Figure: Graphical representation of a tree tensor network with the set of leaves $\mathcal{L} = \{1, 2, 3, 4, 5, 6\}$. 
Tree tensor network - Definition:
Graphical representation
Tree tensor network - Definition

Definition (Tree tensor network)

For a given tree $\bar{\tau} \in \mathcal{T}$ and basis matrices $U_\ell$ and connection tensors $C_\tau$ as described 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

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

(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 $I_\tau$ the identity matrix of dimension $r_\tau$, and

$$X_\tau := C_\tau \times_0 I_\tau X_{\tau_1} \times_1 \ldots \times_m U_{\tau_1} \in \mathbb{R}^{r_\tau \times n_{\tau_1} \times \cdots \times n_{\tau_m}},$$

$$U_\tau := \text{Mat}_0(X_\tau)^\top \in \mathbb{R}^{n_\tau \times r_\tau}.$$
Tree tensor network integrator - Tucker integrator first

\[ C^0_{\tau} \]

\[ U^0_{\tau_1} \]

\[ U^0_{\tau_2} \]

\[ U^0_6 \]

\[ C^0_{\tau} \]

\[ C^0_{\tau_1} \]

\[ C^0_{\tau_2} \]

\[ U^0_1 \]

\[ U^0_2 \]

\[ U^0_3 \]

\[ U^0_4 \]

\[ U^0_5 \]

\[ U^0_6 \]
Figure: We apply the Tucker integrator on the smaller tree tensor network.
Definition of $F_{\tau_i}$ and $Y_{\tau_i}^0$

Let $\tau = (\tau_1, \ldots, \tau_m)$ and $i = 1, \ldots, m$. We recursively define

\[
F_{\tau_i} := \pi_{\tau, i}^\dagger \circ F_\tau \circ \pi_{\tau, i},
\]

\[
Y_{\tau_i}^0 := \pi_{\tau, i}^\dagger(Y_\tau^0).
\]
Definition of $F_{\tau_i}$ and $Y^0_{\tau_i}$ - Prolongation

Let $\tau = (\tau_1, \ldots, \tau_m)$ and $i = 1, \ldots, m$. We recursively define

$$F_{\tau_i} := \pi^{\dagger}_{\tau, i} \circ F_\tau \circ \pi_{\tau, i},$$

$$Y^0_{\tau_i} := \pi^{\dagger}_{\tau, i}(Y^0_\tau).$$
Definition of $F_{\tau_i}$ and $Y_{\tau_i}^0$ - Restriction

Let $\tau = (\tau_1, \ldots, \tau_m)$ and $i = 1, \ldots, m$. We recursively define

$$F_{\tau_i} := \pi_{\tau, i}^\dagger \circ F_\tau \circ \pi_{\tau, i},$$

$$Y_{\tau_i}^0 := \pi_{\tau, i}^\dagger(Y_\tau^0).$$
Consider a tree $\tau = (\tau_1, \ldots, \tau_m)$. We define

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

We introduce the **prolongation**

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

and the **restriction**

$$\pi_{\tau,i}^\dagger(Z_\tau) := \text{Ten}_0\left((\mathbf{Mat}_i(Z_\tau)\mathbf{V}_{\tau_i}^0)^\top\right) \in \mathcal{V}_{\tau_i} \quad \text{for} \quad Z_\tau \in \mathcal{V}_\tau.$$
Lemma

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

$$\pi^\dagger_{\tau, i}(\pi_{\tau, i}(Y_{\tau_i})) = Y_{\tau_i} \quad \text{for all } Y_{\tau_i} \in V_{\tau_i}$$

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

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

The recursive tree tensor network integrator is derived as a recursive application of the Tucker integrator.

Due to its recursive derivation, it preserves the exactness property and it remains robust with respect to the presence of small singular values in the matricizations of the connection tensors, as the matrix and the Tucker projector splitting integrator.

C., Lubich, Walach 2021
Thanks for your attention!
Matrix projector-splitting integrator

1. **K-step**: Update $U_0 \rightarrow U_1, S_0 \rightarrow \hat{S}_1$
   Integrate to $t = t_1$ the $m \times r$ differential equation
   \[ \dot{K}(t) = F(t, K(t) V_0^T) V_0, \quad K(t_0) = U_0 S_0 \]
   and perform a **QR factorization** $K(t_1) = U_1 \hat{S}_1$.

2. **S-step**: Update $\hat{S}_1 \rightarrow \tilde{S}_0$
   Integrate to $t = t_1$ the $r \times r$ differential equation
   \[ \dot{S}(t) = -U_1^T F(t, U_1 S(t) V_0^T) V_0, \quad S(t_0) = \hat{S}_1 \]
   \[ (i!) \]

3. **L-step**: Update $V_0 \rightarrow V_1, \tilde{S}_0 \rightarrow S_1$
   Integrate to $t = t_1$ the $r \times n$ differential equation
   \[ \dot{L}^T(t) = U_1^T F(t, U_1 L(t)^T), \quad L^T(t_0) = \tilde{S}_0 V_0^T \]
   and perform a **QR factorization** $L(t_1) = V_1 S_1^T$.

---

*Lubich, Oseledets 2014*
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 \textit{Tucker tensor} $Y = \mathbf{C} \mathbf{X}^d_{i=1} \mathbf{U}_i$, i.e., $y_{k_1, \ldots, k_d} = \sum_{l_1=1}^{r_1} \cdots \sum_{l_d=1}^{r_d} c_{l_1, \ldots, l_d} u_{k_1, l_1} \cdots u_{k_d, l_d}$, where the \textit{basis matrices} $\mathbf{U}_i \in \mathbb{R}^{n_i \times r_i}$ have orthonormal columns and the \textit{core tensor} $\mathbf{C} \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ has full multilinear rank $(r_1, \ldots, r_d)$. 
Dynamical low-rank approximation of tensors in Tucker format is equivalent to

\[ \dot{Y} = P(Y)F(t, Y), \quad Y(t_0) = Y_0 \in \mathcal{M}_r. \]

The orthogonal projection \( P(Y) \) onto \( \mathcal{T}_Y \mathcal{M}_r \) is given by

\[ P(Y) = \sum_{i=1}^{d} \left( P_i^+(Y) - P_i^-(Y) \right) + P_0(Y). \]
(d=3) Nested Tucker integrator

Lubich, Vandereycken, Walach 2018
The result of the Tucker tensor integrator after one time step can be expressed in a compact way as

\[ Y^1 = \Psi \circ \Phi^{(d)} \circ \cdots \circ \Phi^{(1)}(Y^0). \]
Algorithm 1: Subflow $\Phi^{(i)}$

Data: $Y^0 = C^0 X^d_{j=1} U^0_j$ in factorized form, $F(t, Y), t_0, t_1$

Result: $Y^1 = C^1 X^d_{j=1} U^1_j$ in factorized form

begin
  set $U^1_j = U^0_j \quad \forall j \neq i$
  compute the QR decomposition $\text{Mat}_i(C^0)^\top = Q^0_i S^0_i,^\top \in \mathbb{R}^{r-i \times r_i}$
  set $K^0_i = U^0_i S^0_i \in \mathbb{R}^{n_i \times r_i}$
  solve the $n_i \times r_i$ matrix differential equation
  $\dot{K}_i(t) = F_i(t, K_i(t))$ with initial value $K_i(t_0) = K^0_i$
  and return $K^1_i = K_i(t_1)$; here
  $F_i(t, K_i) = \text{Mat}_i(F(t, \text{Ten}_i(K_i(t))V^0_i,^\top))V^0_i$ with
  $V^0_i,^\top = \text{Mat}_i(\text{Ten}_i(Q^0_i,^\top)X_{j\neq i} U^0_j)$
  compute the QR decomposition $K^1_i = U^1_i \hat{S}^1_i$
  solve the $r_i \times r_i$ matrix differential equation
  $\dot{S}_i(t) = -\hat{F}_i(t, S_i(t))$ with initial value $S_i(t_0) = \hat{S}^1_i$
  and return $\hat{S}_i^0 = S_i(t_1)$; here
  $\hat{F}_i(t, S_i) = U^1_i,^\top F_i(t, U^1_i S_i)$
  set $C^1 = \text{Ten}_i(\hat{S}_i^0 Q^0_i,^\top)$

end


**Algorithm 2: Subflow $\Psi$**

**Data:** $Y^0 = C^0 X_{j=1}^d U_j^0$ in factorized form, $F(t, Y)$, $t_0$, $t_1$

**Result:** $Y^1 = C^1 X_{j=1}^d U_j^1$ in factorized form

begin

- set $U_j^1 = U_j^0 \ \forall j = 1, \ldots, d.$
- solve the $r_1 \times \cdots \times r_d$ tensor differential equation
  \[ \dot{C}(t) = \tilde{F}(t, C(t)) \]  with initial value $C(t_0) = C^0$
  and return $C^1 = C(t_1)$; here
  \[ \tilde{F}(t, C) = F(t, C X_{j=1}^d U_j^1) X_{j=1}^d U_j^{1,\top} \]

end
Recursive TTN integrator

The recursive TTN integrator is derived as a recursive application of the Nested Tucker integrator

\[ Y_{\tau}^1 = \psi_{\tau} \circ \phi_{\tau}^{(m)} \circ \cdots \circ \phi_{\tau}^{(1)}(Y_{\tau}^0). \]
Algorithm 2: Subflow $\Phi^{(i)}_\tau$

**Data:** tree $\tau = (\tau_1, \ldots, \tau_m)$, TTN in factorized form

$\begin{align*}
Y^0_\tau &= C^0_\tau \times_0 I \times_{j=1}^m U^0_{\tau_j} \quad \text{with} \quad U^0_{\tau_j} = \text{Mat}_0(X^0_{\tau_j})^T, \\
\text{function} \quad F_\tau(t, Y_\tau), t_0, t_1
\end{align*}$

**Result:** TTN $Y^1_\tau = C^1_\tau \times_0 I \times_{j=1}^m U^1_{\tau_j}$ with $U^1_{\tau_j} = \text{Mat}_0(X^1_{\tau_j})^T$

in factorized form

begin

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

compute the QR factorization $\text{Mat}_i(C^0_\tau)^T = Q^0_{\tau_i} S^0_{\tau_i}^T$

set $Y^0_{\tau_i} = X^0_{\tau_i} \times_0 S^0_{\tau_i}^T$

if $\tau_i = \ell$ is a leaf, then solve the $n_\ell \times r_\ell$ matrix differential equation

$\begin{align*}
\dot{Y}_{\tau_i}(t) &= F_{\tau_i}(t, Y_{\tau_i}(t)) \\
\text{with initial value} \quad Y_{\tau_i}(t_0) &= Y^0_{\tau_i} \\
\text{and return} \quad Y^1_{\tau_i} &= Y_{\tau_i}(t_1)
\end{align*}$

else

compute $Y^1_{\tau_i} = \text{Recursive TTN Integrator} (\tau_i, Y^0_{\tau_i}, F_{\tau_i}, t_0, t_1)$

compute the QR factorization $\text{Mat}_0(C^1_{\tau_i})^T = \tilde{Q}^1_{\tau_i} \tilde{S}^1_{\tau_i}$, where $C^1_{\tau_i}$ is the connecting tensor of $Y^1_{\tau_i}$

set $U^1_{\tau_i} = \text{Mat}_0(X^1_{\tau_i})^T$, where the TTN $X^1_{\tau_i}$ is obtained from $Y^1_{\tau_i}$ by replacing the connecting tensor with $\tilde{C}^1_{\tau_i} = \text{Ten}_0(\tilde{Q}^1_{\tau_i}, T)$

solve the $r_{\tau_i} \times r_{\tau_i}$ matrix differential equation

$\begin{align*}
\dot{S}_{\tau_i}(t) &= -\tilde{F}_{\tau_i}(t, S_{\tau_i}(t)) \quad \text{with initial value} \quad S_{\tau_i}(t_0) = \tilde{S}^1_{\tau_i} \\
\text{and return} \quad \tilde{S}^0_{\tau_i} &= S_{\tau_i}(t_1); \quad \text{here} \\
\tilde{F}_{\tau_i}(t, S_{\tau_i}) &= U^1_{\tau_i} \text{Mat}_0(F_{\tau_i}(t, X^1_{\tau_i} \times_0 S^T_{\tau_i}))^T
\end{align*}$

set $C^1_\tau = \text{Ten}_i(\tilde{S}^0_{\tau_i}, Q^0_{\tau_i}, T)$

end
Tensor Trains represented as TTNs

Figure: Tensor train represented in hierarchical Tucker (HT) format.