Sergey V. Dolgov

A tensor decomposition algorithm for large ODEs with conservation laws

Abstract: We propose an algorithm for solution of high-dimensional evolutionary equations (ODEs and discretized time-dependent PDEs) in the Tensor Train (TT) decomposition, assuming that the solution and the right-hand side of the ODE admit such a decomposition with a low storage. A linear ODE, discretized via one-step or Chebyshev differentiation schemes, turns into a large linear system. The tensor decomposition allows to solve this system for several time points simultaneously using an extension of the Alternating Least Squares algorithm. This method computes a reduced TT model of the solution, but in contrast to traditional offline-online reduction schemes, solving the original large problem is never required. Instead, the method solves a sequence of reduced Galerkin problems, which can be set up efficiently due to the TT decomposition of the right-hand side. The reduced system allows a fast estimation of the time discretization error, and hence adaptation of the time steps. Besides, conservation laws can be preserved exactly in the reduced model by expanding the approximation subspace with the generating vectors of the linear invariants and correction of the euclidean norm. In numerical experiments with the transport and the chemical master equations, we demonstrate that the new method is faster than traditional time stepping and stochastic simulation algorithms, whereas the invariants are preserved up to the machine precision irrespectively of the TT approximation accuracy.

Keywords: high–dimensional problems, tensor train format, DMRG, alternating iteration, differential equations, conservation laws

1 Introduction

Large-scale evolutionary equations for many-body systems arise ubiquitously in the numerical modeling. The cases of particular interest and difficulty involve many configuration coordinates in the state space. For instance, the time-dependent *Schroedinger* equation describes the wavefunction, which depends on

Sergey V. Dolgov, University of Bath, Claverton Down, BA2 7AY, Bath, United Kingdom (s.dolgov@bath.ac.uk). The author acknowledges funding from the EPSRC fellowship EP/M019004/1.

all positions of all quantum particles or states of spins. Another important example is the joint probability density function, which is driven by the *Fokker-Planck* or *master* equations in continuous or discrete spaces, respectively. The solution of a problem with d configuration variables is a d-variate function. When d is much larger than 3, a uniform discretization would require $\mathcal{O}(n^d)$ degrees of freedom. Typical examples in quantum physics involve d being of the order of hundreds, and the straightforward computation with n^d unknowns is impossible.

To cope with such *high-dimensional* problems, one has to employ *data-sparse* techniques, i.e. describe the solution by much fewer unknowns than n^d. Different approaches exist for this task. Among the most successful ones we may identify Monte Carlo (and Quasi Monte Carlo) methods [34, 12], Sparse Grids [43, 3], and tensor product representations. In this paper, we adopt the latter framework.

Tensor product decompositions rely on the idea of separation of variables: a d-variate array (or *tensor*) can be defined or approximated by sums of products of univariate factors. Extensive information can be found in recent reviews and books, e.g. [29, 15, 27]. A promising potential of the tensor product methods stems from the fact that a univariate factor is defined by only n values. If a tensor can be approximated up to the required accuracy with a moderate number of factors, the memory and complexity savings can be outstanding.

There exist different tensor product *formats*, i.e. rules that map univariate factors to the initial array. In case of two dimensions, one ends up with the low-rank dyadic factorization of a matrix. A straightforward extension of such sum of direct products of vectors in higher dimensions is called the CP format [18]. However, the CP approximation problem may be ill-posed [5]. This issue is circumvented in recurrent two-dimensional factorizations, where one can enforce a certain stable form of the representation. In this paper, we focus on the simplest example, the so-called *Tensor Train* (TT) decomposition [36]. It was rediscovered several times, and the most important analogs in quantum physics are *Matrix Product States* (MPS) [10] and *Density Matrix Renormalization Group* (DMRG) [48]. This format possesses all the power of recurrent factorizations, but algorithms are easier to describe. For higher flexibility in particular problems, one can use more general tree-based constructions, such as *HT* [13] or *Extended TT/QTT-Tucker* [6] formats.

DMRG is not only the name of the representation, but also a variety of computational tools. It was originally developed for finding ground states (lowest eigenpairs) of high-dimensional Hamiltonians of spin chains. The main idea behind DMRG is the alternating optimization of a function (e.g. Rayleigh quotient) over the factors of a tensor decomposition. It was noticed that this method may manifest a remarkably fast convergence, and later extensions to the energy function followed [22, 19].

Besides the stationary problems, the same framework was applied to the dynamical spin Schroedinger equation. Two conceptually similar techniques, the *time-evolving block decimation* (TEBD) [46] and the *time-dependent DMRG* (tDMRG) [49] take into account the nearest-neighbor form of the Hamiltonian in order to split the operator exponent into two parts using the Trotter decompositions. Each part can then be integrated exactly, followed by the separation of variables via the truncated singular value decomposition. These methods perform very well for short times, but in a long time integration the error may accumulate, and the storage of the tensor product decomposition grows dramatically [40].

To avoid this problem, one can use the so-called *Dirac-Frenkel* principle [28, 30]. This scheme projects the dynamical equations onto the tangent space of the Riemannian manifold, induced by the tensor decomposition. The storage of the format is now fixed, but the approximation errors can be difficult to control.

As an alternative approach, we consider time as an extra variable and discretize an ODE into a system of algebraic equations on many time steps simultaneously [47, 8, 24]. If the original ODE is linear, so is this system. A handful of time steps allows to estimate the time discretization error and adapt the time grid accordingly. However, the global state-time system is non-symmetric and requires a reliable solution algorithm in the tensor format. We use an extension of DMRG, the so-called *Alternating Minimal Energy* (AMEn) method [9]. It augments the tensor format of the solution by the tensor format of the global residual. This improves the convergence and allows to adapt the tensor format storage up to a desired accuracy tolerance.

The residual is not the only quantity we can enrich the solution with. The approximation error of the tensor decomposition is distributed evenly in all components of the solution. However, it might be beneficial to compute some parts of the solution with a higher accuracy. For example, the exact ODE may possess certain *conservation laws* (e.g. phase [39] or normalization), which are worth to be preserved in a numerical scheme. We show that the basis vectors of the co-kernel of the ODE matrix can be inserted into the TT representation of the solution in addition to the residual. This preserves the corresponding linear invariants. The second norm of the solution can then be corrected by rescaling.

The paper is structured as follows. In the next section we formulate the ODE problem, investigate its properties related to the first- and the second-order invariants, show the Galerkin model reduction concept and how the invariants can be preserved in the reduced system, and suggest an adaptive linear discretization in time. Section 3 starts with a brief introduction to tensor product formats and methods and presents the new tAMEn algorithm (the name is motivated by tDMRG). Section 4 demonstrates supporting numerical examples, followed by the conclusion in Section 5.

2 Ordinary differential equations

Our problem of interest is a linear system of ODEs,

$$\frac{dx}{dt} = A(t)x, \qquad x(0) = x_0, \tag{1}$$

solved on $t \in [0, T]$, where $A(t) \in \mathbb{C}^{N\times N}$ is a stable matrix. Throughout the paper, x and other quantities denoted by small letters are $N \times 1$ vectors, such that the inner products can be consistently written as $c^*x \in \mathbb{C}^{1\times 1}$. Up to technical changes, the formulation (1) can be extended to ODEs with forcing, $dx/dt = Ax + f(t)$, or weakly nonlinear systems, where $A(t) = A(t, \check{x}(t))$ depends on the solution from the previous Picard iteration.

2.1 Conservation laws and Galerkin reduction

Our goal will be to approximate the ODE solution in a compressed data-sparse form. A particular question of interest is the following: if the system preserves some quantities in time, is it possible to maintain this property in data-sparse algorithms, which are based on the Galerkin projection approach?

The simplest conservation laws are defined by linear functions of the solution and its euclidean norm. Given some *detecting* vector $c \neq 0$, the linear function can be written as c^*x. It corresponds, for example, to the probability normalization in the Fokker-Planck equation: x represents the discretized probability density function, and $\sum_{i=1}^{N} x(i) = c^*x = 1$, with c being a vector of all ones. For a time-invariant system $dx/dt = Ax$, a sufficient condition for conservation of c^*x is the nullspace equation $A^*c = 0$.

Among the second-order invariants, we consider the euclidean (Frobenius) norm of the solution, $\|x\| = \sqrt{x^*x}$. The conservation law $\|x(t)\| = \|x_0\|$ is a well-known property of the Schroedinger equation $dx/dt = \mathrm{i}Hx$, where i is the imaginary unity, and $H = H^\top \in \mathbb{R}^{N\times N}$. A sufficient condition is the skew-symmetry of the matrix, $A = -A^*$.

An abstract Galerkin reduction can be written as follows. Given an orthogonal set of columns $X \in \mathbb{C}^{N\times r}$, $X^*X = I$, we replace the large system (1) by a reduced ODE[1],

$$\frac{dv}{dt} = (X^*AX)\, v, \qquad v(0) = v_0 = X^*x_0. \tag{2}$$

1 for simplicity, we consider the time-invariant ODE in this section.

Numerical treatment of this equation is cheap if the basis size is small, $r \ll N$. The approximate solution of the initial problem (1) writes as $\tilde{x}(t) = Xv(t) \approx x(t)$. Many approaches exist to determine the basis sets X, see e.g. [1, 2]. The Krylov method for the computation of the matrix exponential [32] belongs to this class as well. Another celebrated technique is the Proper Orthogonal Decomposition (POD) [31, 42, 25, 35], which extracts principal components from a set of *snapshots* $\{x(t_j)\}_{j=1}^{\mathcal{J}}$ using the singular value decomposition.

The accuracy $\|x - \tilde{x}\|$ of the reduced model depends on the approximation capacity of the basis set. In this paper, we employ a tensor product algorithm, which is similar to POD but computes both the basis and the reduced solution adaptively without solving the large original problem. Most importantly, it belongs to the Galerkin projection framework (2). Here we show how to preserve first and second order invariants with an arbitrary Galerkin basis.

Suppose we are given vectors $C = \begin{bmatrix} c_1 & \cdots & c_M \end{bmatrix}$ such that $A^*C = 0$. Let us *expand* the basis by concatenating C and X and orthogonalizing the result,

$$\begin{bmatrix} C & X \end{bmatrix} = \hat{X}R, \quad \hat{X}^*\hat{X} = I \quad \text{(QR decomposition)}. \tag{3}$$

Since the first M columns of $\hat{X}$ belong to the kernel of A^*, the reduced matrix writes

$$\hat{X}^*A\hat{X} = \begin{bmatrix} \mathcal{C}^*A\mathcal{C} & \mathcal{C}^*A\mathcal{X} \\ \mathcal{X}^*A\mathcal{C} & \mathcal{X}^*A\mathcal{X} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ \mathcal{X}^*A\mathcal{C} & \mathcal{X}^*A\mathcal{X} \end{bmatrix}, \quad \text{where} \quad \hat{X} = \begin{bmatrix} \mathcal{C} & \mathcal{X} \end{bmatrix}.$$

In order to derive the reduced solution $v(t) = \exp\left(t\hat{X}^*A\hat{X}\right)v_0$ in the expanded basis, consider one recursion step for the exponential series. For any $k = 1, 2, \ldots,$

$$\begin{bmatrix} 0 & 0 \\ (\mathcal{X}^*A\mathcal{X})^{k-1}\,\mathcal{X}^*A\mathcal{C} & (\mathcal{X}^*A\mathcal{X})^k \end{bmatrix} \begin{bmatrix} 0 & 0 \\ \mathcal{X}^*A\mathcal{C} & \mathcal{X}^*A\mathcal{X} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ (\mathcal{X}^*A\mathcal{X})^k\,\mathcal{X}^*A\mathcal{C} & (\mathcal{X}^*A\mathcal{X})^{k+1} \end{bmatrix},$$

and hence we obtain

$$\exp\left(t\hat{X}^*A\hat{X}\right) = I + \sum_{k=1}^{\infty} \frac{\left(t\hat{X}^*A\hat{X}\right)^k}{k!} = \begin{bmatrix} I & 0 \\ \sum\limits_{k=1}^{\infty} \frac{t(t\mathcal{X}^*A\mathcal{X})^{k-1}}{k!}\mathcal{X}^*A\mathcal{C} & \exp\left(t\mathcal{X}^*A\mathcal{X}\right) \end{bmatrix}. \tag{4}$$

Since the first row contains only the identity, the linear invariants $\mathcal{C}^*x_0$ are explicitly preserved in the solution, $v(t) = \begin{bmatrix} \mathcal{C}^*x_0 \\ w(t) \end{bmatrix}$.

The skew-symmetry, yielding conservation of the second norm, is even easier to consider, since for any Galerkin projection, $(X^*AX)^* = X^*A^*X = -X^*AX$, and hence $\|v(t)\| = \|X^*x_0\|$. Moreover, $\|\tilde{x}(t)\| = \|v(t)\| = \|X^*x_0\|$ if X is

orthogonal. Thus, it is enough to guarantee $\|X^* x_0\| = \|x_0\|$. A simple way to do this is to rescale the projected initial state. However, this requires a certain care if we need to preserve both the second norm and the linear invariants. Given $v_0 = \begin{bmatrix} \mathcal{C}^* x_0 \\ \mathcal{X}^* x_0 \end{bmatrix}$, we can rescale only the bottom part. This means finding $\theta > 0$ such that

$$\|\hat{v}_0\|^2 = \|\mathcal{C}^* x_0\|^2 + \theta^2 \|\mathcal{X}^* x_0\| = \|x_0\|^2, \quad \text{hence} \quad \theta = \frac{\sqrt{\|x_0\|^2 - \|\mathcal{C}^* x_0\|^2}}{\|\mathcal{X}^* x_0\|}, \tag{5}$$

and the rescaled initial state reads

$$\hat{v}_0 = \begin{bmatrix} \mathcal{C}^* x_0 \\ \theta \mathcal{X}^* x_0 \end{bmatrix}.$$

Due to orthogonality of $\mathcal{C}$ and $\mathcal{X}$, it holds that $\|\mathcal{C}^* x_0\| \le \|\hat{X}^* x_0\| \le \|x_0\|$, and hence θ is well-defined when $x_0 \notin \operatorname{span}(\mathcal{C})$. Otherwise, $\|\mathcal{X}^* x_0\| = 0$, and the rescaling is not needed.

2.2 Linear discretization in time

Assuming the solution $x(t)$ to be continuous, we can introduce a time discretization grid $\mathbf{t} = \{t_j\}_{j=1}^{\mathcal{J}} \in [0, T]$ and collocate the solution on this grid, $\{x_j\} = \{x(t_j)\}$. An approximate solution at any time can be computed by the polynomial interpolation,

$$x(t) \approx \sum_{j=1}^{\mathcal{J}} x_j p_j(t), \tag{6}$$

where $p_j(t)$ are polynomials, centered at t_j, such as the global Lagrange polynomials or local splines. Since both ODE (1) and the interpolation (6) is linear in x, the discrete system is linear as well, and can be generally written as

$$Bx = f, \qquad B = I_N \otimes S - (I_N \otimes P)\, A(\mathbf{t}), \quad f = x_0 \otimes (Se), \tag{7}$$

where $A(\mathbf{t})$ is a block-diagonal matrix constructed from the ODE matrices at the grid points,

$$A(\mathbf{t}) = \begin{bmatrix} A(t_1) & & \\ & \ddots & \\ & & A(t_{\mathcal{J}}) \end{bmatrix}, \quad \text{and} \quad x = \begin{bmatrix} x(t_1) \\ \vdots \\ x(t_{\mathcal{J}}) \end{bmatrix} \tag{8}$$

is the vector of all snapshots stacked together, $S \in \mathbb{R}^{\mathcal{J} \times \mathcal{J}}$ is the stiffness matrix corresponding to the time derivative, $P \in \mathbb{R}^{\mathcal{J} \times \mathcal{J}}$ is the mass matrix,

$e = (1, \dots, 1)^\top \in \mathbb{R}^{\mathcal{J}}$ is a vector of all ones, and $\otimes$ is the Kronecker product. For a time-invariant ODE (8) simplifies to $A(\mathbf{t}) = A \otimes I_{\mathcal{J}}$. For example, Euler and Crank-Nicolson schemes belong to this class with $S = \text{tridiag}(-1, 1, 0)$, and $P = \frac{T}{\mathcal{J}} I$ for the implicit Euler scheme on a grid $t_j = Tj/\mathcal{J}$, and

$$P = \frac{T}{2(\mathcal{J}-1)} \begin{bmatrix} 0 & & & \\ 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{bmatrix}$$

for the Crank-Nicolson scheme on a grid $t_j = T(j-1)/(\mathcal{J}-1)$. With these schemes we can take linear splines at $t_j \le t \le t_{j+1}$ in the interpolation (6).

Alternatively, for the Chebyshev grid with nodes $t_j = \frac{T}{2}(1 - \cos(\pi j/\mathcal{J}))$ we obtain the spectral differentiation matrix [45, Chapter 6] $S = \{dp_j(t_i)/dt\}_{i,j=1}^{\mathcal{J}}$, where p_j is the Lagrange polynomial centered at t_j, and $P = I$. An advantage of the spectral discretization is the rapid convergence (exponential in $\mathcal{J}$, see [44] and [45, Theorem 6]) when the solution is analytic on the Bernstein ellipse extension of $[0, T]$. On the other hand, lower order schemes lead to sparse matrices and lower condition numbers in (7).

The Galerkin reduction (2) can be combined with (7) straightforwardly. Given an orthogonal basis matrix X, we assemble and solve the $r\mathcal{J} \times r\mathcal{J}$ system

$$\left(I_r \otimes S - (I_r \otimes P)\,(X \otimes I_{\mathcal{J}})^* A(\mathbf{t})(X \otimes I_{\mathcal{J}})\right) v = v_0 \otimes (Se), \tag{9}$$

where $v_0 = X^* x_0$. Both linear and quadratic invariants can be preserved as shown in (3) and (5), respectively.

Remark 1. *Low-order schemes are often preferred to the spectral discretization because of the particular sparsity of the stiffness and mass matrices, e.g.* bidiagonality*, which allows to solve* (7) *step by step. However, in this paper we solve* (7) *indirectly via iterative tensor product algorithms (see Sec. 3.3), which require a single system of equations, defining the entire solution. On the other hand, tensor decompositions allow more freedom in the choice of S and P due to the reduced cost; in fact, solving the global system* (7) *can be faster and more accurate than the step by step integration [8], since it allows to take more accurate time discretization.*

Remark 2. *If the ODE solution lacks smoothness, more sophisticated Discontinuous Galerkin techniques may be required [41, 24]. Otherwise, the collocation leads to easier pointwise construction of the matrix* (8)*, compared to the computation of the Galerkin coefficients.*

An analog of the Runge's rule [16] can be used for estimating the discretization error. Consider two grids with $\mathcal{J}$ and $2\mathcal{J}$ points, $\{t_j\}_{j=1}^{\mathcal{J}}$ and $\{t_i^*\}_{i=1}^{2\mathcal{J}}$. Given an approximation $y(t) \approx dx/dt$ on the coarse grid $\{t_j\}$ (in our case $y(t) = A(t)x(t)$), we can take the difference on the fine grid $|dx/dt(t_i^*) - y(t_i^*)|$ as our error estimate. For evaluating the quantities on $\{t_i^*\}$ we construct the fine-grid differentiation matrix $\hat{S} \in \mathbb{R}^{2\mathcal{J}\times 2\mathcal{J}}$ and the interpolation matrix $\hat{P} \in \mathbb{R}^{2\mathcal{J}\times\mathcal{J}}$, which maps from $\{t_j\}$ to $\{t_i^*\}$. Then the estimate can be computed from the snapshots as follows,

$$\mathcal{E}_{\mathcal{J},T} = \left\| \left[I_N \otimes (\hat{S}\hat{P}) - (I_N \otimes \hat{P}) A(\mathbf{t}) \right] x - x_0 \otimes (\hat{S}\hat{e}) \right\|, \tag{10}$$

where $\hat{e}$ is a vector of all ones of size $2\mathcal{J}$. For the Chebyshev discretization, for example, $\hat{P}_{i,j} = p_j(t_i^*)$.

3 Tensor product representations and methods

3.1 Vectors and tensors

The unknowns in the whole discrete solution can be enumerated by at least two independent indices, corresponding to the state space and time. Assuming that $i = 1, \dots, N$ enumerates the state components of the solution, $x_i(t)$, and that the time points are enumerated by an index $j = 1, \dots, \mathcal{J}$, we can consider the solution as a matrix $X = [x_i(t_j)]$. Moreover, we will assume (and exploit) that the state space can be further factorised into d independent indices $i_1, \dots, i_d$, running from 1 to $n_1, \dots, n_d$, respectively. An equivalence between *digits* $i_1, \dots, i_d$ and the original index i holds due to the standard positional expression,

$$i = (i_1 - 1)n_2 \cdots n_d + (i_2 - 1)n_3 \cdots n_d + \cdots + i_d. \tag{11}$$

However, the solution can now be also seen as a *tensor*, $\mathbf{x} = [\mathbf{x}(i_1, \dots, i_d, j)] \in \mathbb{C}^{n_1 \times \cdots \times n_d \times \mathcal{J}}$. The multi-index expansion can arise for example from a discretization of PDEs: if a PDE $\frac{\partial x}{\partial t}(q_1, \dots, q_d, t) = Ax(q_1, \dots, q_d, t)$ is discretized in $q_1, \dots, q_d$ by collocation on a Cartesian product of independent univariate grids $\{q_k(i_k)\}$, $k = 1, \dots, d$, the nodal values of x can be collected into a tensor $\mathbf{x}$, as described above.

To write the global state-time system (7) consistently, we need to reshape the whole tensor $\mathbf{x}$ into a vector x of size $(n_1 \cdots n_d)\mathcal{J} \times 1$. We can extend (11) to any set of indices, introducing a general *multi-index*

$$\overline{i_p \dots i_q} = (i_p - 1)n_{p+1} \cdots n_q + \cdots + i_q, \quad q \geq p. \tag{12}$$

Now we can address the solution by either of the equivalent forms $x(\overline{i_1 \dots i_d, j})$, $X(\overline{i_1 \dots i_d}, j)$ or $\mathbf{x}(i_1, \dots, i_d, j)$.

3.2 Tensor Train decomposition

The Tensor Train (TT) [36], or Matrix Product States (MPS) [10] decomposition for the tensor $\mathbf{x}$ (resp. vector x) is defined as follows,

$$x(\overline{i_1 \dots i_d, j}) = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_d=1}^{r_d} \mathbf{x}^{(1)}_{\alpha_1}(i_1)\mathbf{x}^{(2)}_{\alpha_1,\alpha_2}(i_2)\cdots\mathbf{x}^{(d)}_{\alpha_{d-1},\alpha_d}(i_d)\mathbf{x}^{(d+1)}_{\alpha_d}(j). \quad (13)$$

The summation indices $\alpha_k = 1, \dots, r_k$, $k = 1, \dots, d$, are called the *rank* indices, and their ranges r_k are the *tensor train* ranks (TT ranks). The right-hand side consists of the TT *blocks* $\mathbf{x}^{(k)} \in \mathbb{C}^{r_{k-1}\times n_k \times r_k}$. Introducing uniform bounds $r_k \le r$, $n_k \le n$, we can estimate the storage complexity of the TT decomposition as $\mathcal{O}(dnr^2)$. If the rank bound r is small, this is much lower than $N\mathcal{J} = \mathcal{O}(n^d\mathcal{J})$ in the straightforward storage of x.

The matrix B from (7) can be seen as a $(2d+2)$-dimensional tensor and decomposed in a slightly different *matrix* TT decomposition,

$$B(\overline{i_1 \dots i_d, j}, \overline{i_1 \dots i_d, j'}) = \sum_{\gamma_1=1}^{\mathcal{R}_1} \cdots \sum_{\gamma_d=1}^{\mathcal{R}_d} \mathbf{B}^{(1)}_{\gamma_1}(i_1, i'_1)\cdots\mathbf{B}^{(d)}_{\gamma_{d-1},\gamma_d}(i_d, i'_d)\mathbf{B}^{(d+1)}_{\gamma_d}(j, j'). \quad (14)$$

The matrix TT decomposition is introduced for consistency with the Kronecker product when $\mathcal{R}_1 = \cdots = \mathcal{R}_d = 1$ and multiplication with a "vector" TT decomposition of x (13). Assuming an upper bound $\mathcal{R}_k \le \mathcal{R}$, we can estimate the storage of (14) by $\mathcal{O}(dn^2\mathcal{R}^2)$ for a dense matrix, and by $\mathcal{O}(dn\mathcal{R}^2)$ for a sparse matrix.

The multi-index notation (12) allows to notice that the TT decomposition can be seen as a low-rank decomposition of a matrix $X^{\{k\}} = \left[X(\overline{i_1 \dots i_k}, \overline{i_{k+1} \dots j})\right]$ for any $k = 1, \dots, d$. We can group the left, respectively, right subset of TT blocks into *interface matrices*, or simply *interfaces*

$$X^{(\le k)}(\overline{i_1 \dots i_k}, \alpha_k) = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_{k-1}=1}^{r_{k-1}} \mathbf{x}^{(1)}_{\alpha_1}(i_1)\cdots\mathbf{x}^{(k)}_{\alpha_{k-1},\alpha_k}(i_k),$$

$$X^{(>k)}(\alpha_k, \overline{i_{k+1} \dots j}) = \sum_{\alpha_{k+1}=1}^{r_{k+1}} \cdots \sum_{\alpha_d=1}^{r_d} \mathbf{x}^{(k+1)}_{\alpha_k,\alpha_{k+1}}(i_{k+1})\cdots\mathbf{x}^{(d+1)}_{\alpha_d}(j). \quad (15)$$

We can naturally extend this definition to $X^{(<k)} = X^{(\le k-1)}$ and $X^{(\ge k)} = X^{(>k-1)}$. Then we can write $X^{\{k\}} = X^{(\le k)}X^{(>k)}$. Moreover, the interface matrices allow to see the TT decomposition as a *linear map* of each TT block $\mathbf{x}^{(k)}$. Indeed, reshaping it into a vector $x^{(k)}(\overline{\alpha_{k-1} i_k \alpha_k}) = \mathbf{x}^{(k)}_{\alpha_{k-1},\alpha_k}(i_k)$, we can write $x = X_{\ne k}x^{(k)}$, where $X_{\ne k}$ is a *frame* matrix

$$X_{\ne k} = X^{(<k)} \otimes I_{n_k} \otimes \left(X^{(>k)}\right)^\top. \quad (16)$$

3.3 Computing TT decompositions by alternating iteration

Although a TT approximation can be computed for any tensor via a sequence of singular value decompositions (SVD) [36], this is rarely efficient or even possible when the tensor is large. The aim of the tensor product methodology is to avoid fully stored tensors at all stages of computations. One of the most successful approaches traces back to the alternating least squares optimization over the tensor decomposition blocks. It was then generalized to the Alternating Linear Scheme (ALS) [19]. A similar algorithm, called Density Matrix Renormalization Group (DMRG) [48, 22], was proposed in quantum physic for calculation of ground states, i.e. lowest eigenvalues of high-dimensional Hamiltonians.

Let us consider the linear system $Bx = f$ as an overdetermined equation on a particular block $x^{(k)}$ in the TT decomposition (13); the linearity established in the previous subsection makes this equation linear, $(BX_{\neq k})x^{(k)} = f$. This equation can be resolved in different ways (e.g. by least squares), but practically the cheapest option is to use the same frame matrix,

$$\left(X_{\neq k}^* B X_{\neq k}\right) x^{(k)} = X_{\neq k}^* f. \tag{17}$$

This reduction can be justified by relation to the minimization of the energy function $x^*Bx - 2\mathrm{Re}\, x^*f$ when the matrix B is symmetric (hermitian) positive definite (SPD). However, the projection formalism (17) is more general and can be applied also if B is not SPD, which is the case for (7). The *alternating* iteration is realised by sweeping through different blocks, $k = 1, \dots, d+1$, and backwards from $k = d+1$ to $k = 1$ until convergence, solving (17) in each step.

Three essential details make the alternating iteration actually useful:

– efficient assembly of (17);
– orthogonality of $X_{\neq k}$ and efficient solution of (17);
– adaptation of TT ranks of x.

The frame matrix, composed from the interface matrices (15) via Kronecker products, can be seen as a special TT decomposition with the same number of blocks as in x. In turn, the matrix B and right-hand side f are assumed to be available in the TT format as well, such as (14). This allows to compute $X_{\neq k}^* B X_{\neq k}$ and $X_{\neq k}^* f$ efficiently, using only multiplications of the TT blocks. Moreover, sequential iteration over $k = 1, 2, \dots$ allows to reuse partial products of the interfaces of x, B and f and make the algorithm even more efficient, with the total asymptotic complexity linear in d [19, 37].

The TT representation is not unique; any partition of identity can be inserted between adjacent TT blocks, e.g. $X^{\{k\}} = \left(X^{(\leq k)} R\right)\left(R^{-1} X^{(>k)}\right)$, without changing the whole tensor. However, the matrix R changes the interfaces, and

we can choose it in order to empower the representation with desirable properties. For example, we can make $X^{(<k)}$ and $X^{(>k)}$ orthogonal by performing QR decompositions of appropriately reshaped TT blocks. By construction (16), $X_{\neq k}$ is orthogonal, too. The orthogonality of the projection (17) leads to a well conditioned reduced problem, which can be solved iteratively (we employ the BiCGstab algorithm) using fast matrix-vector products due to the TT structure inherited from the original problem [37].

For high-dimensional problems it is difficult to guess all d rank parameters. It becomes necessary to adapt them during the computations in such a way that the TT solution is within the desired distance from the exact solution. If we possess a solution with a satisfactory accuracy but overly large TT ranks, it is easy to reduce them via SVD [36]. It is more important therefore to develop a procedure for increasing the ranks. The DMRG method addresses this problem by reducing the system to a two-dimensional block (merged from $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(k+1)}$), which can be split via SVD up to a desired threshold. However, this requires solving a larger problem on the merged block. The Alternating Minimal Energy (AMEn) algorithm [9] solves one-dimensional problems in each step, but augments the TT blocks of the solution by the TT blocks of an approximate *global* residual $z \approx f - Bx$. Since f, B and x are all represented in the TT format, the residual can be approximated efficiently by the *second* ALS iteration, applied to a simpler problem $Iz = f - Bx$. Given a TT decomposition

$$z(\overline{i_1 \dots i_d, j}) = \sum_{\beta_1,\dots,\beta_d=1}^{\rho_1,\dots,\rho_d} \mathbf{z}^{(1)}_{\beta_1}(i_1) \cdots \mathbf{z}^{(d+1)}_{\beta_d}(j)$$

from the previous iteration, we define the interface matrices $Z^{(<k)}$ and $Z^{(>k)}$ similarly to (15), and update the k-th TT block of the residual by projecting

$$z^{(k)} = \left(Z^{(<k)} \otimes I \otimes (Z^{(>k)})^\top \right)^* (f - Bx). \tag{18}$$

Performing this process simultaneously with the computation of the solution blocks (17), we ensure that $Z^{(<k)}$ and $Z^{(>k)}$ are sufficiently good bases for the residuals in all steps. In turn, projecting the residual onto the solution interface,

$$\zeta^{(k)} = \left(X^{(<k)} \otimes I \otimes (Z^{(>k)})^\top \right)^* (f - Bx), \tag{19}$$

we can expand the solution TT block,

$$\mathbf{x}^{(k)}(i_k) = \begin{bmatrix} \mathbf{x}^{(k)}(i_k) & \boldsymbol{\zeta}^{(k)}(i_k) \end{bmatrix}, \qquad \mathbf{x}^{(k+1)}(i_{k+1}) = \begin{bmatrix} \mathbf{x}^{(k+1)}(i_{k+1}) \\ 0 \end{bmatrix}. \tag{20}$$

This allows to increase the solution TT ranks (by the ranks of $\boldsymbol{\zeta}^{(k)}$), and also improves convergence in difficult cases, since the basis of the reduction (17) contains now the residual of the original problem.

3.4 tAMEn: extended time integrator

The time-dependent version of the AMEn algorithm combined two enrichments of the solution: by the residual (20) and by the co-kernel vectors (3). Assume the latter to be given in a compatible TT format,

$$c_m(\overline{i_1 \dots i_d}) = \sum_{\beta_1,\dots,\beta_{d-1}=1}^{\rho_1,\dots,\rho_{d-1}} \mathbf{c}^{(1)}_{\beta_1}(i_1)\mathbf{c}^{(2)}_{\beta_1,\beta_2}(i_2)\cdots\mathbf{c}^{(d)}_{\beta_{d-1},m}(i_d),$$

where $m = 1,\dots,M$ enumerates different vectors c_m. In the course of the alternating iteration from $k = 1$ to $k = d$, the combined enrichment is performed as follows,

$$\mathbf{x}^{(k)}(i_k) = \begin{bmatrix}\mathbf{x}^{(k)}(i_k) & \boldsymbol{\zeta}^{(k)}(i_k) & \mathcal{C}_k\mathbf{c}^{(k)}(i_k)\end{bmatrix}, \qquad \mathbf{x}^{(k+1)}(i_{k+1}) = \begin{bmatrix}\mathbf{x}^{(k+1)}(i_{k+1}) \\ 0 \\ 0\end{bmatrix}, \tag{21}$$

where $\mathcal{C}_k = (X^{(<k)})^* C^{(<k)}$ is the projection onto the left interface of the solution. We can see that $c_m \in \operatorname{span}(X^{(<k)} \otimes I_{n_k\cdots n_d})$ for all $k = 1,\dots,d$ and $m = 1,\dots,M$. For $k = 1$, for example, we can write

$$c_m = \left(\mathbf{x}^{(1)} \otimes I_{n_2\cdots n_d}\right)\tilde{c}_m, \quad \tilde{c}_m = \begin{bmatrix}0 \\ 0 \\ c_m^{(>1)}\end{bmatrix},$$

where $c_m^{(>1)}$ is the $(n_2\cdots n_d\rho_1)\times 1$ vectorisation of the interface matrix $C_m^{(>1)}$. By induction, this extends to $k > 1$. In order to maintain orthogonality of the interfaces, we perform the QR decomposition of $\mathbf{x}^{(k)}$ after the enrichment (21).

For $k = d+1$, we can notice that the frame matrix reduces to $X^{(\le d)} \otimes I$. Therefore, the local problem (17) for $x^{(d+1)}$ is nothing else than the reduced discretized ODE (9) with the interface being the Galerkin basis, $X = X^{(\le d)}$, and the last TT block being the unknown, $v = x^{(d+1)}$. The enrichment (21) ensures that this basis contains also the co-kernel matrix C. Moreover, the second norm of the right hand side (reduced initial state) can be corrected according to (5). If we stop the alternating iteration at this step, the error in the linear invariants and the second norm depends only on the accuracy of the solution of (9) and the time discretization, but not on the accuracy of the TT decomposition. If the last TT rank r_d is reasonably small, we can take sufficiently large $\mathcal{J}$ and solve (9) directly, which yields the machine precision accuracy in the conservation laws.

Fig. 1: tAMEn algorithm

Require: Initial state x_0, matrix $A(t)$ and right hand side $f(t)$ in the TT format, final time T, accuracy threshold ε, discretization points $\mathbf{t} \in [0,1]$ and matrices $S, P, \hat{S}$ and $\hat{P}$, co-kernel basis C in the TT format.

Ensure: Time splitting points $T_0 = 0 < T_1 < \cdots < T_L = T$, solutions $x_\ell(t)$ in the TT format.

Set $t = 0$, $T_0 = 0$, $\ell = 1$, $h_1 = T$.
while $t < T$ **do**
  Rescale $\mathbf{t}, S, P, \hat{S}$ and $\hat{P}$ from $[0,1]$ to $[T_{\ell-1}, T_{\ell-1} + h_\ell]$.
  Form $B = I \otimes S - (I \otimes P)\mathrm{diag}(A(\mathbf{t}))$ and $f = x_{\ell-1} \otimes (Se)$.
  Set $x = x_{\ell-1} \otimes e$.
  for iter $= 1, 2, \ldots,$ **do**
    Set $x_{prev} = x$.
    for $k = d+1, d, \ldots, 2$ **do**
      Orthogonalize $X^{(>k)}$ and $Z^{(>k)}$, see [36, Section 3].
    end for
    for $k = 1, 2, \ldots, d$ **do** ▷ Solve
      Solve $(X^*_{\neq k} B X_{\neq k}) x^{(k)} = X_{\neq k} f$, as defined in (7) and (16).
      Compute truncated SVD of $\mathbf{x}^{(k)}$ up to ε.
      Compute residual blocks as shown in (18) and (19).
      Enrich $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(k+1)}$ as shown in (21).
      Orthogonalize $X^{(<k+1)}$ and $Z^{(<k+1)}$, see [36, Section 3].
    end for
    Correct the norm of $v_0 = (X^{(\leq d)})^* x_{\ell-1}$ as shown in (5).
    Solve $(X^*_{\neq d+1} B X_{\neq d+1}) x^{(d+1)} = \hat{v}_0 \otimes (Se)$.
    Compute the error estimate (22) and $h_{\ell+1} = h_\ell \cdot (\varepsilon / \mathcal{E}_{\mathcal{J},h_\ell})^{1/q}$.
    if $\mathcal{E}_{\mathcal{J},h_\ell} \leq \varepsilon$ **then**
      if $\|x - x_{prev}\| < \varepsilon \|x\|$ **then** ▷ This step converged, accept it
        Set $x_\ell = x$, $T_\ell = T_{\ell-1} + h_\ell$, $t = t + h_\ell$, $\ell = \ell + 1$, and break.
      end if
    else ▷ Reject the step
      Set $h_\ell = h_{\ell+1}$ and break.
    end if
  end for
end while

The time discretization error (10) can be also estimated from the reduced system. Instead of the full solution, we consider only the last TT block, and replace the state matrix by its projection[2],

$$\mathcal{E}_{\mathcal{J},T} = \left\| \left(I_{r_d} \otimes \hat{S}\hat{P} - \left[(X^{(\le d)})^* A X^{(\le d)} \right] \otimes \hat{P} \right) x^{(d+1)} - \hat{v}_0 \otimes (\hat{S}\hat{e}) \right\|. \quad (22)$$

This estimate can be used for refining the number of time points $\mathcal{J}$ or the length of the time interval. Instead of solving (7) on the whole desired interval $[0, T]$, we can split it into a sequence of subintervals $[0, T_1], \dots, [T_{L-1}, T_L]$, taking the solution at the last time point of the previous interval as the initial state in the next interval. We determine an optimal splitting using the local error control with rejections [4]. We aim to maintain the error in the next time interval below a desired threshold, $\mathcal{E}_{\mathcal{J},h_{\ell+1}} \le \varepsilon$, so we adjust the next interval length as follows,

$$h_{\ell+1} = T_{\ell+1} - T_\ell = h_\ell \left(\frac{\varepsilon}{\mathcal{E}_{\mathcal{J},h_\ell}} \right)^{1/q}. \quad (23)$$

The parameter q reflects the order of convergence of the time scheme, which is 1 for the Euler method, 2 for the Crank-Nicolson scheme, and $\mathcal{J}$ for the Chebyshev differentiation. Moreover, if it appears that $\mathcal{E}_{\mathcal{J},h_\ell} > \varepsilon$, such solution is *rejected*, the *current* interval is shrunk according to (23), and the solution is started again from $T_{\ell-1}$. The entire procedure is written in Fig. 1. Assuming the index ranges from Sec. 3.2 ($r_k \le r$, $\mathcal{R}_k \le \mathcal{R}$, $n_k \le n$), the computational complexity of Alg. 1, inherited from AMEn [9], reads

$$\mathcal{O}(dn(\mathcal{R}r^3 + \mathcal{R}^2 r^2)).$$

4 Numerical experiments

We have implemented Algorithm 1 in Matlab. This research made use of the Balena High Performance Computing (HPC) Service at the University of Bath. We carried out the computations on one core of the `Balena` node, an Intel Xeon E5-2650 CPU at 2.6GHz. The code is available from `http://github.com/dolgov/tamen`.

2 For non-autonomous ODEs the estimate can be extended accordingly.

4.1 Convection

Our first example is the transport equation in the periodic domain $[-10, 10]^2$ with the central difference discretization scheme,

$$\frac{dx}{dt} = (\nabla_n \otimes I_n + I_n \otimes \nabla_n)\, x, \quad \nabla_n = \frac{1}{2h} \begin{bmatrix} 0 & 1 & \cdots & & -1 \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ 1 & & \cdots & -1 & 0 \end{bmatrix} \in \mathbb{R}^{n \times n}, \tag{24}$$

where $h = 20/n$ is the mesh step of the uniform grid $q_k(i_k) = -10 + h(i_k - 1)$, $i_k = 1, \ldots, n$, $k = 1, 2$, and the Gaussian initial state $x_0 = \exp(-q_1^2 - q_2^2)$. This example is chosen for the following reasons. First, the exact solution repeats with the period of $T_p = 20$, hence we can estimate the error of our scheme as the difference between the solution after a number of periods and the initial state, $\|x(T) - x_0\|$ for $T = 20m$, $m \in \mathbb{N}$. Second, (24) possesses both types of invariants: the solution mass $c^* x = c^* x_0$, where $c = (1, \ldots, 1)^\top$, and the second norm, $\|x\|_2 = \|x_0\|_2$. Third, the discrete solution of the pure convection is prone to developing spurious oscillations when the discretization is not accurate enough. For the central difference scheme, this requires taking rather fine grids, with n ranging from 1024 to 4096, which makes the problem large enough to apply the tensor decompositions.

In order to increase the efficiency of the TT methods, we apply them to the *quantized* tensors [26]: instead of separating just two indices in $X(i_1, i_2)$, we split each index into its binary digits,

$$i_k = 2^{L-1}(i_{k,1} - 1) + 2^{L-2}(i_{k,2} - 1) + \cdots + i_{k,L}, \quad i_{k,l} \in \{1, 2\},$$

and consider the solution as a $2L$-dimensional tensor, $\mathbf{x}(i_{1,1}, \ldots, i_{2,L})$. Now the TT decomposition can reduce the storage cost down to $\mathcal{O}(r^2 L) = \mathcal{O}(r^2 \log n)$, in contrast to $\mathcal{O}(rn)$ in the low-rank decomposition of $X(i_1, i_2)$ or n^2 in the full representation of x. The ODE matrix can be constructed in this quantized TT representation exactly [23].

First, we confirm conservation of the invariants. We fix the time interval splitting to $h_\ell = 0.2$ for all $\ell = 1, \ldots, T/h_\ell$, the spatial grid size $n = 4096$, and vary the number of Chebyshev or Crank-Nicolson points, as well as the accuracy threshold ε. In Fig. 2 we show how the errors in both conserving quantities evolve with time. We observe that the error in the invariants is much smaller than the tensor approximation threshold in all cases. However, insufficient number of Chebyshev points can increase the error in the second norm (Fig. 2, left). The

Fig. 2: Convection example. Degeneracy of $\|u\|_2$ (solid lines) and $c^* u$ (dashed lines) vs. time for Chebyshev (left) and Crank-Nicolson (right) schemes.

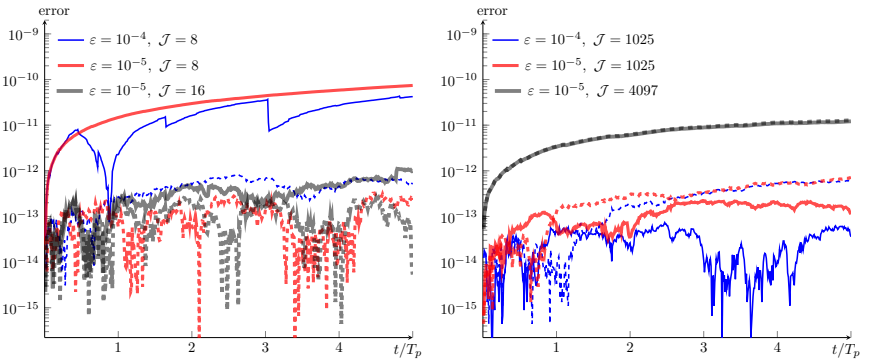
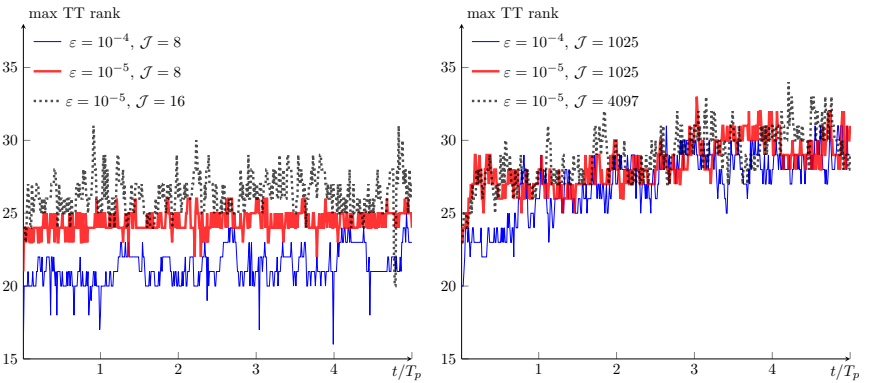


Fig. 3: Convection example, TT ranks for Chebyshev (left) and Crank-Nicolson (right) schemes.



Crank-Nicolson scheme preserves both invariants up to the machine precision for any number of points. In fact, it manifests the opposite situation that the errors are larger for $\mathcal{J} = 4097$ points due to a larger condition number of the matrix in (7). This shows that, although the explicit cost of the tensor schemes depends mildly on the grid sizes, it is still recommended to avoid too fine grids due to the conditioning issues.

The evolution of TT ranks with time is shown in Fig. 3. Since the shape of the exact solution remains unchanged, its ranks should be the same for all times. We see that the ranks are indeed stable with time, in particular with the Chebyshev scheme. In the Crank-Nicolson scheme, the ranks grow slightly

Fig. 4: Convection example, CPU times in seconds (left) and time intervals found in the adaptive regime (right) with Chebyshev and Crank-Nicolson (CN) schemes.

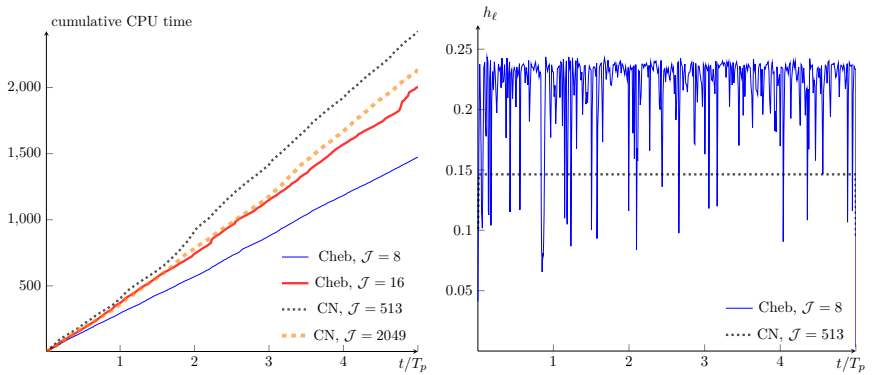


Table 1: Convection, CPU times (seconds) and errors for different time interval lengths.

Scheme	**Chebyshev,** $\mathcal{J} = 8$				**Crank-Nicolson,** $\mathcal{J} = 513$			
Time step	0.1	0.2	0.4	100_{adap}	0.1	0.2	0.4	100_{adap}
CPU time	**1098.8**	**1391.1**	**5069.9**	**2014.7**	**4326.0**	**2213.7**	**10246.1**	**2652.9**
$\frac{10^4 \cdot \|x - x_\star\|}{\|x_\star\|}$	**0.85**	**6.22**	**5.65**	**2.81**	**3.33**	**3.52**	**2.22**	**2.62**

towards the end of the 5-period interval. This is also reflected by a slightly larger CPU time, see Fig. 4 (left).

Now we consider how the tAMEn algorithm depends on the time interval splitting. In Table 1 we show the CPU times and the errors of $x(T)$ with respect to the reference solution $x_\star$, computed with the Chebyshev scheme with $\mathcal{J} = 16$ points on $h_\ell = 0.2$ and $\varepsilon = 10^{-7}$. For small time steps ($h_\ell = 0.1, 0.2$ and 0.4), we turn the adaptation off. However, we also start from the entire interval 100 and let the algorithm split it automatically. Due to rejections of some time steps, the CPU time of the adaptive method is larger than the cost of the optimal splitting ($h_\ell = 0.1$ for Chebyshev and $h_\ell = 0.2$ for Crank-Nicolson schemes), but the overhead never exceeds a factor of 2. Moreover, the adaptive algorithm is faster than the non-adaptive one with improperly chosen time steps. Fig. 4 (right) shows the time steps determined by the adaptive method. We see that the average step lies between 0.1 and 0.2. Interestingly, the low-order Crank-Nicolson scheme is more robust in estimating the error, and hence the time step.

Finally, we benchmark tAMEn against the standard Crank-Nicolson method without the TT decomposition and the Riemannian TT time integrator [30]. We split the time into intervals of length $h_\ell = 0.2$, but each interval is further

Table 2: Convection example. CPU times (seconds) and errors in different methods and parameters for time splitting $h_\ell = 0.2$.

	tAMEn		KSL		Full CN	
	Cheb, $\mathcal{J}=8$	**CN,** $\mathcal{J}=513$	$\mathcal{J}=16$	$\mathcal{J}=512$	$\mathcal{J}=16$	$\mathcal{J}=64$
CPU time	**1391.1**	**2213.7**	**694.9**	**14159**	**170732**	**102294**
$\frac{10^3\cdot\|x(T)-x_0\|}{\|x_0\|}$	**2.36**	**2.16**	**583.8**	**7.44**	**16.0**	**2.93**

partitioned into $\mathcal{J}$ individual time steps, on which the full Crank-Nicolson or Riemannian integration is carried out. The Riemannian integrator projects the dynamical equations directly onto the Riemannian manifold of the TT representation, using the so-called *Dirac-Frenkel* principle [28]. The projected equations can be split with respect to the different TT blocks and solved subsequently, using the so-called KSL propagator [30][3] This scheme works with the TT decomposition of only one snapshot at a time, which requires smaller TT ranks. However, it requires integrating backward in time, which can introduce numerical instabilities for large time steps. In Table 2 we see that for $\mathcal{J}=16$ the solution becomes qualitatively incorrect. For a smaller time step the scheme is stable, but a large number of time steps leads to a large computational time. The full Crank-Nicolson method is even slower, since each time step is more expensive. In fact, the CPU time is larger for smaller number of time steps. This is due to a larger condition number of the matrix in the implicit step.

4.2 Chemical master equation

In the second experiment, we investigate an example with a steady state, the chemical master equation (CME), describing stochastic kinetics model of the λ-phage virus [17, 21, 7]. Using the Finite State Projection [33], the CME is turned into a large-scale ODE,

$$\frac{dx}{dt} = Ax, \qquad A = \sum_{m=1}^{M} \left(J^{z_1^m} \otimes \cdots \otimes J^{z_d^m} - I \right) \mathrm{diag}(w^m), \tag{25}$$

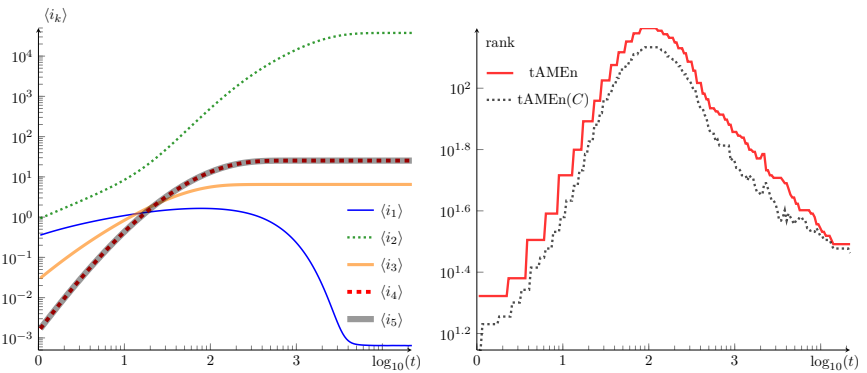
Here, J^z is the order-z shift matrix, defined as follows: $J^0 = I$, $J^1 = \mathrm{tridiag}(1,0,0)$, $J^z = (J^1)^z$ for $z>1$, and $J^z = (J^{-z})^\top$ for $z<0$. The vector $\mathbf{z}^m = (z_1^m, \ldots, z_d^m)$ is the so-called *stoichiometric* vector, $w^m = w^m(i_1,\ldots,i_d)$ is

3 The multi-dimensional Matlab version `tt_ksl_ml.m` was implemented by the author in collaboration with I. Oseledets, and is available within TT-Toolbox.

Table 3: Reactions in the λ-phage model.

Generation		Destruction	
$w^1 = \dfrac{0.06}{0.12 + i_2}$,	$\mathbf{z}^1 = \mathbf{e}_1$	$w^2 = 0.0025 \cdot i_1$,	$\mathbf{z}^2 = -\mathbf{e}_1$
$w^3 = \dfrac{(1 + i_5) \cdot 0.6}{0.6 + i_1}$,	$\mathbf{z}^3 = \mathbf{e}_2$	$w^4 = 0.0007 \cdot i_2$,	$\mathbf{z}^4 = -\mathbf{e}_2$
$w^5 = \dfrac{0.15 \cdot i_2}{i_2 + 1}$,	$\mathbf{z}^5 = \mathbf{e}_3$	$w^6 = 0.0231 \cdot i_3$,	$\mathbf{z}^6 = -\mathbf{e}_3$
$w^7 = \dfrac{0.3 \cdot i_3}{i_3 + 1}$,	$\mathbf{z}^7 = \mathbf{e}_4$	$w^8 = 0.01 \cdot i_4$,	$\mathbf{z}^8 = -\mathbf{e}_4$
$w^9 = \dfrac{0.3 \cdot i_3}{i_3 + 1}$,	$\mathbf{z}^9 = \mathbf{e}_5$	$w^{10} = 0.01 \cdot i_5$,	$\mathbf{z}^{10} = -\mathbf{e}_5$

Fig. 5: CME example, $\langle i_k \rangle$ (left) and maximal TT ranks in tAMEn with and without C-enrichment (right)



the *propensity* rate of the m-th reaction, and $\mathrm{diag}(w^m)$ constructs a $N \times N$ diagonal matrix from all elements of w^m. The total size of the problem is $N = \prod_{k=1}^{d} n_k$, since each index is assumed to vary in the range $i_k = 0, \ldots, n_k - 1$. The indices $i_1, \ldots, i_d$ denote the so-called *copy numbers* (numbers of molecules) of d reacting species (e.g. proteins), and the solution $\mathbf{x}(i_1, \ldots, i_d, t)$ is the distribution function, which defines the probability that at the time t, the system contains i_1 molecules of the first protein, i_2 molecules of the second species, and so on.

The particular λ-phage model contains $d = 5$ species and $M = 10$ reactions. The stoichiometric vectors and propensities are given in Table 3 ($\mathbf{e}_1, \ldots, \mathbf{e}_5$ are unit vectors of size 5).

As the initial state, we choose all-zero copy numbers with probability 1, i.e. $x_0(i_1, \ldots, i_5) = 1$ when $i_1 = \cdots = i_5 = 0$, and 0, otherwise. Under certain conditions [14], fulfilled for the λ-phage model, and infinite ranges of i_k, the CME

(25) converges to a unique stationary state x_∞. For practical computations, we truncate the state space to $n_1 \times \cdots \times n_5 = 128 \times 65536 \times 64 \times 64 \times 64$, respectively, since the probability outside this box is negligible. In order to preserve existence of the stationary state [20], we adjust the propensities of the generation reactions such that

$$w^{2k-1}(i_1, \dots, i_d) = 0 \quad \text{if} \quad i_k = n_k - 1, \quad k = 1, \dots, d.$$

This also guarantees that $A^* e = 0$, therefore the probability *normalization* $e^* x = 1$ is conserved.

The statistical outputs of interest are the *mean copy numbers*,

$$\langle i_k \rangle = \frac{\mathtt{i}_k^* x}{e^* x}, \quad \mathtt{i}_k = \mathsf{e}^{(1)} \otimes \cdots \otimes \mathsf{e}^{(k-1)} \otimes \{i_k\} \otimes \mathsf{e}^{(k+1)} \otimes \cdots \otimes \mathsf{e}^{(d)} \in \mathbb{R}^N, \quad (26)$$

where $\mathsf{e}^{(p)}$ are the all-ones vectors of size n_p. In order to preserve the normalization, we add the vector of ones to the enrichment (21). However, we can also keep the quantities of interest in the TT representation in order to make statistics more accurate. Therefore, we use 6 enrichment columns, $C = \begin{bmatrix} e & \mathtt{i}_1 & \cdots & \mathtt{i}_5 \end{bmatrix}$. The Quantized TT representation of C has TT ranks up to 6, and the ranks of the residual (18) are set to 1. We compare tAMEn implementations with and without the additional enrichment by C. For the fair comparison, we set the residual ranks equal to those of C plus 1 in the version without the C-enrichment.

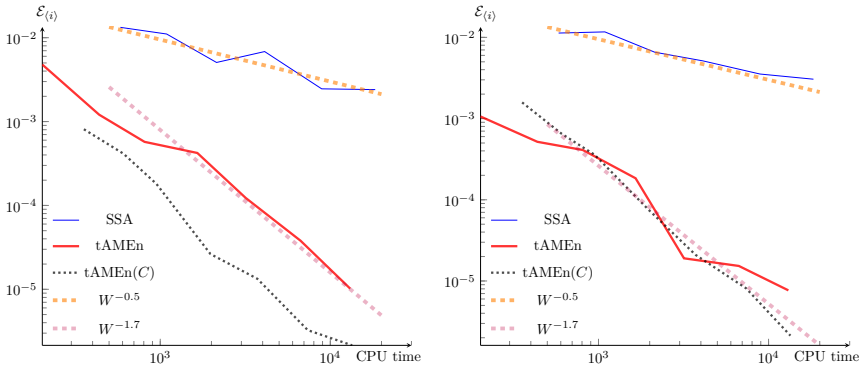
The tAMEn algorithm is run in the fully adaptive regime using the Chebyshev time discretisation scheme with $\mathcal{J} = 8$ points in each interval. The final time $T = 22000$. We estimate the errors directly in the quantities of interest and report the log-average of individual errors,

$$\mathcal{E}_{\langle i \rangle}(t) = \exp\left(\frac{1}{5} \sum_{k=1}^{5} \log \frac{\left| \langle i_k(t) \rangle - \langle i_k^\star(t) \rangle \right|}{\langle i_k^\star(t) \rangle} \right). \quad (27)$$

We vary the accuracy thresholds ε from 10^{-2} to 10^{-5}, and use the values computed with $\varepsilon = 3 \cdot 10^{-7}$ as the reference $\langle i_k^\star \rangle$. In addition to the two versions of tAMEn, we present the results of the classical Stochastic Simulation Algorithm (SSA) [11] for comparison.

In Fig. 5 we show the evolution of the mean copy numbers and TT ranks. Interestingly, the ranks with the C-enrichment are even smaller, since the specialized frame matrices constitute better bases for the solution. The computational times and errors are shown in Fig. 6. We see that the normalization-preserving solution is systematically more efficient in terms of the cost/accuracy ratio, compared to the residual-only enrichment. Moreover, the direct solution of the CME in the TT format is much faster than the stochastic simulation, since large times and copy numbers require a large number of trajectories and time steps in SSA.

Fig. 6: CME example, errors (27) in the mean copy numbers for $t = 2000$ (left) and $t = 22000$ (right) versus computational Work (CPU time) for SSA and tAMEn with and without the C-enrichment.



5 Conclusion

We have proposed and studied an alternating iterative algorithm for approximate solution of ordinary differential equations in the TT format. The method combines advances of DMRG techniques and classical iterative methods of linear algebra. Started from the solution at the previous time interval as the initial guess, it often converges in 2—4 iterations, and delivers accurate solution even for strongly non-symmetric matrices in the right-hand side of an ODE. The numerical experiments reveal a promising potential of this method in long time simulations when the solution admits a low-rank decomposition. For example, nuclear magnetic resonance models can be approached directly, without any a priori reduction of the original Hilbert space [39].

The main limiting factor of the method is the TT ranks of the solution. A theoretical rank bound remains in general an open question, especially if the ODE is nonlinear. While for some problems (e.g. convection) it follows readily from the continuous equation, analysis of realistic cases may be difficult. A beneficial feature of the new algorithm is that it can determine the TT ranks adaptively in the course of computations.

Existing convergence estimates for tensor algorithms in high dimensions are far from being sharp. Although the time stepping scheme might be easier for both local [38] and global [9] convergence analysis under an assumption of small enough time step, the particular criterion for “small enough” might be too restrictive. In practice, the method converges robustly for stable first-order

systems. Unstable, higher-order and differential-algebraic systems require further investigation, and potentially modification of the algorithm.

The method possesses a simple mechanism for maintaining linear conservation laws in the reduced tensor model exactly, provided that the generating vectors admit low-rank representations. In principle, this covers most of the needs in statistical problems, where the solution defines a probability distribution, and the invariants are means of some functions w.r.t. this distribution. However, it is unclear whether general nonlinear invariants can be preserved in a tensor product representation. Some abelian and non-abelian symmetries in quantum physics can be preserved by further splitting of TT blocks into invariant sectors [40].

References

[1] A. C. Antoulas, D. C. Sorensen, and S. Gugercin. A survey of model reduction methods for large-scale systems. *Contemporary mathematics*, 280:193–220, 2001.

[2] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.

[3] H.-J. Bungatrz and M. Griebel. Sparse grids. *Acta Numerica*, 13(1):147–269, 2004.

[4] G. D. Byrne and A. C. Hindmarsh. A polyalgorithm for the numerical solution of ordinary differential equations. *ACM Trans. Math. Softw.*, 1(1):71–96, 1975.

[5] V. de Silva and L.-H. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM J. Matrix Anal. Appl.*, 30(3):1084–1127, 2008.

[6] S. Dolgov and B. Khoromskij. Two-level QTT-Tucker format for optimized tensor calculus. *SIAM J. on Matrix An. Appl.*, 34(2):593–623, 2013.

[7] S. Dolgov and B. Khoromskij. Simultaneous state-time approximation of the chemical master equation using tensor product formats. *Numer. Linear Algebra Appl.*, 22(2):197–219, 2015.

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

[9] S. V. Dolgov and D. V. Savostyanov. Alternating minimal energy methods

for linear systems in higher dimensions. *SIAM J. Sci. Comput.*, 36(5):A2248–A2271, 2014.

[10] M. Fannes, B. Nachtergaele, and R.F. Werner. Finitely correlated states on quantum spin chains. *Comm. Math. Phys.*, 144(3):443–490, 1992.

[11] D.T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J Comput. Phys.*, 22(4):403–434, 1976.

[12] I.G. Graham, F.Y. Kuo, D. Nuyens, R. Scheichl, and I.H. Sloan. Quasi-Monte Carlo methods for elliptic PDEs with random coefficients and applications. *J. Comput. Phys.*, 230(10):3668–3694, 2011.

[13] L. Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM J. Matrix Anal. Appl.*, 31(4):2029–2054, 2010.

[14] A. Gupta and M. Khammash. Determining the long-term behavior of cell populations: a new procedure for detecting ergodicity in large stochastic reaction networks. *IFAC Proceedings Volumes*, 47(3):1711 – 1716, 2014.

[15] W. Hackbusch. *Tensor Spaces And Numerical Tensor Calculus*. Springer–Verlag, Berlin, 2012.

[16] G. Hall and J.M. Watt. *Modern numerical methods for ordinary differential equations*. Clarendon Press, 1976.

[17] M. Hegland, C. Burden, L. Santoso, S. MacNamara, and H. Booth. A solver for the stochastic master equation applied to gene regulatory networks. *Journal of Computational and Applied Mathematics*, 205(2):708 – 724, 2007.

[18] F. L. Hitchcock. Multiple invariants and generalized rank of a p-way matrix or tensor. *J. Math. Phys*, 7(1):39–79, 1927.

[19] S. Holtz, T. Rohwedder, and R. Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM J. Sci. Comput.*, 34(2):A683–A713, 2012.

[20] T. Jahnke. On reduced models for the chemical master equation. *Multiscale Modeling and Simulation*, 9(4):1646–1676, 2011.

[21] T. Jahnke and W. Huisinga. A dynamical low-rank approach to the chemical master equation. *Bulletin of Mathematical Biology*, 70:2283–2302, 2008.

[22] E. Jeckelmann. Dynamical density–matrix renormalization–group method. *Phys. Rev. B*, 66:045114, 2002.

[23] V. Kazeev, B. Khoromskij, and E. Tyrtyshnikov. Multilevel Toeplitz matrices generated by tensor-structured vectors and convolution with logarithmic complexity. *SIAM J. Sci. Comput.*, 35(3):A1511–A1536, 2013.

[24] V. Kazeev, O Reichmann, and Ch. Schwab. hp-DG-QTT solution of high-dimensional degenerate diffusion equations. Tech. Report 2012-11, ETH SAM, Zürich, 2012.

[25] G. Kerschen, J. Golinval, A. Vakakis, and L. Bergman. The method of proper

orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: An overview. *Nonlinear Dynamics*, 41(1):147–169, 2005.

[26] B. N. Khoromskij. $\mathcal{O}(d \log n)$–Quantics approximation of N–d tensors in high-dimensional numerical modeling. *Constr. Approx.*, 34(2):257–280, 2011.

[27] B. N. Khoromskij. Tensor numerical methods for multidimensional PDEs: theoretical analysis and initial applications. *ESAIM: Proc.*, 48:1–28, 2015.

[28] O. Koch and Ch. Lubich. Dynamical tensor approximation. *SIAM J. Matrix Anal. Appl.*, 31(5):2360–2375, 2010.

[29] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, 2009.

[30] Ch. Lubich, I. Oseledets, and B. Vandereycken. Time integration of tensor trains. *SIAM J. Numer. Anal.*, 53(2):917–941, 2015.

[31] J. L. Lumley. The structure of inhomogeneous turbulent flows. *Atmospheric turbulence and radio wave propagation*, pages 166–178, 1967.

[32] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

[33] B. Munsky and M. Khammash. The finite state projection algorithm for the solution of the chemical master equation. *The Journal of chemical physics*, 124:044104, 2006.

[34] H. Niederreiter. Quasi–Monte Carlo methods and pseudo–random numbers. *Bull. AMS*, 84(6):957–1041, 1978.

[35] A. Nouy. A priori model reduction through proper generalized decomposition for solving time-dependent partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 199(23):1603–1626, 2010.

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

[37] I. V. Oseledets and S. V. Dolgov. Solution of linear systems and matrix inversion in the TT-format. *SIAM J. Sci. Comput.*, 34(5):A2718–A2739, 2012.

[38] T. Rohwedder and A. Uschmajew. On local convergence of alternating schemes for optimization of convex problems in the tensor train format. *SIAM J. Num. Anal.*, 51(2):1134–1162, 2013.

[39] D. V. Savostyanov, S. V. Dolgov, J. M. Werner, and I. Kuprov. Exact NMR simulation of protein-size spin systems using tensor train formalism. *Phys. Rev. B*, 90:085139, 2014.

[40] U. Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 2011.

[41] D. Schötzau. *hp-DGFEM for parabolic evolution problems. Applications to*

diffusion and viscous incompressible fluid flow. PhD thesis, ETH, Zürich, 1999.

[42] L. Sirovich. Turbulence and the dynamics of coherent structures. *Quarterly of applied mathematics*, 45:561–571, 1987.

[43] S. A. Smolyak. Quadrature and interpolation formulas for tensor products of certain class of functions. *Dokl. Akad. Nauk SSSR*, 148(5):1042–1053, 1963. Transl.: Soviet Math. Dokl. 4:240-243, 1963.

[44] E. Tadmor. The exponential accuracy of Fourier and Chebychev differencing methods. *SIAM J. Numer. Anal.*, 23:1–23, 1986.

[45] L. N. Trefethen. *Spectral methods in MATLAB*. SIAM, Philadelphia, 2000.

[46] G. Vidal. Efficient simulation of one-dimensional quantum many-body systems. *Phys. Rev. Lett.*, 93:040502, 2004.

[47] T. von Petersdorff and Ch. Schwab. Numerical solution of parabolic equations in high dimensions. *ESAIM: Mathematical Modelling and Numerical Analysis*, 38(01):93–127, 2004.

[48] S. R. White. Density-matrix algorithms for quantum renormalization groups. *Phys. Rev. B*, 48(14):10345–10356, 1993.

[49] S. R. White and A. E. Feiguin. Real-time evolution using the density matrix renormalization group. *Phys. Rev. Lett.*, 93:076401, 2004.