Max-Planck-Institut für Mathematik in den Naturwissenschaften Leipzig

Efficient time-stepping scheme for dynamics on TT-manifolds

by

Boris N. Khoromskij, Ivan V. Oseledets, and Reinhold Schneider

Preprint no.: 24 2012

Efficient time-stepping scheme for dynamics on TT-manifolds *

I. V. Oseledets †, B. N. Khoromskij ‡, R. Schneider §

April 20, 2012

Abstract

Solution of non-stationary problems in high dimensions is feasible only if certain low-parametric nonlinear approximation to the solution is used. Thus, even if the initial system is linear, defining equations for the model parameters are nonlinear. The general concept is to use the *Dirac-Frenkel principle* to obtain the approximate trajectory on the manifold. In this paper, this approach is analyzed for the dynamical approximation of high-dimensional tensors in the so-called *Tensor Train (TT)-format*. We obtain an explicit system of ODEs describing the evolution of parameters, defining the solution, and propose an efficient and stable numerical scheme to solve this system. The efficiency of the proposed method is illustrated by the numerical examples.

TT-format, Dirac-Frenkel, Splitting scheme, higher dimensions

1 Introduction

This paper aims at constructing efficient numerical methods for the solution of high-dimensional time-dependent problems of form

$$\frac{dx}{dt} = Ax, \quad x(0) = y_0, \tag{1}$$

where $x = x(q_1, \ldots, q_d, t)$ is an unknown multivariate function of d variables. The problems of form (1) play a crucial role in many applications, especially in quantum chemistry, where numerical methods for the approximate solution of electronic and molecular Schrödinger equations are of great importance, for example in the presence of certain time-dependent sources. Another application is related to the Fokker-Planck and master-type equations in multiconfiguration dynamics.

It is interesting to note that the solution of the non-stationary problem is often used to compute the approximation to the *spectrum* of the operator A. This method will be discussed later on in the numerical examples. Equations of the form (1) appear in the computation of the molecular spectra [24, 1]. The main difficulty in solving (1) is the high-dimensionality of

*This work was partially supported by RFBR grants 12-01-00546, 11-01-12137,11-01-00549, by Rus. Gov. Contracts P1112, 14.740.11.0345, by Rus. President grant MK-140.2011.1, by Dmitriy Zimin Dynasty Foundationn

†ivan.oseledets@gmail.com, Institute of Numerical Mathematics, Russia, 119333, Moscow, Gubkina, 8

‡bokh@mis.mpg.de, Max-Planck Institute for Mathematics in Sciences, Germany, 04103 Leipzig, Inselstraße, 22

§schneidr@math.tu-berlin.de, TU Berlin, Straße des 17. Juni 136, 10623 Berlin, Germany,

the problem. If a tensor-product grid with n nodes in each direction is used, then the total number of unknowns would be n^d, which is large even for small n and d. There are several possibilities to overcome this *curse of dimensionality*. One case use basis functions of special type (radial basis functions, sparse grids, etc), or use methods to approximate the discrete solution. Such kind of methods have been around in quantum chemistry for quite a long time. One of the most effective ones is the so-called multiconfigurational time-dependent Hartree-method (MCTDH) [23] which has proven its effectiveness in many cases. The discretization of a multivariate function on a tensor-product grid leads to high-dimensional arrays that we will call tensors. The number of elements of a tensor grows exponentially in the number of indices. Thus, this tensors have to be approximated by certain *data-sparse* representations.

Several formats have been proposed to represent a tensor in a data-sparse way. They include canonical and Tucker formats, the two formats with well-established properties and application areas, see the review [21] for more details. They have known drawbacks: the canonical format can not be computed via stable algorithms, and the Tucker format in high dimensions is subject to the curse of dimensionality. The $\mathcal{H}$-Tucker [16, 13] and Tree-Tucker [27] formats were proposed as alternatives.

These formats depend on specially chosen *dimension trees* and require recursive procedures. To avoid the recursion, it was proposed to use a simple matrix product form of the tensor decomposition [25, 26], that was called the *Tensor Train format*, or simply the TT-format (also known in other areas as Matrix Product States (MPS) [28] and linear tensor network [31]).

A tensor $\mathbf{A}$ is said to be in the TT-format, if its elements are defined by formula

$$A(i_1, \ldots, i_d) = G_1(i_1) G_2(i_2) \ldots G_d(i_d), \tag{2}$$

where $G_k(i_k)$ is an $r_{k-1} \times r_k$ matrix for each fixed $i_k, 1 \leq i_k \leq n_k$. To make the matrix-by-matrix product in (2) a scalar, boundary conditions $r_0 = r_d = 1$ have to be imposed. The numbers r_k are called *TT-ranks* and $G_k(i_k)$ — cores of the TT-decomposition of a given tensor. If $r_k \leq r, n_k \leq n$, then the storage of the TT-representation requires $\leq dnr^2$ memory cells. If r is small, then this is much smaller than the storage of the full array, n^d.

The TT-format was introduced in [25, 26] as an alternative to two commonly used formats: the canonical format and the Tucker format. These two formats can be considered as different generalizations of the singular value decomposition (SVD) from matrices (i.e., $d = 2$) to higher order tensors. The tensor is said to be in the canonical format, if

$$A(i_1, \ldots, i_d) = \sum_{\alpha=1}^{r} U_1(i_1, \alpha) \ldots U_d(i_d, \alpha). \tag{3}$$

This representation is often referred to as *CANDECOMP/PAFAFAC*, or simply the *CP model* [4, 17]. If r, the canonical rank, is small, then the number of parameters depends on d linearly. The Tucker model [30, 5] is the representation of form

$$A(i_1, \ldots, i_d) = \sum_{\alpha_1, \ldots, \alpha_d} G(\alpha_1, \ldots, \alpha_d) U_1(i_1, \alpha_1) \ldots U_d(i_d, \alpha_d),$$

where U_k are called *Tucker factors*, and the tensor $G(\alpha_1, \ldots, \alpha_d)$ — *Tucker core*.

The canonical format is a good candidate for low-parametric representation of tensors. It can be used to approximate high-dimensional operators and their inverses (see, for example, [2, 3, 11, 14, 15, 12]) using certain analytical expansions. However, the canonical format has a serious drawback: there are no robust algorithms to compute the canonical approximation numerically. In fact, the best approximation problem in the canonical format can be ill-posed [6] and the best approximation may not even exist.

The TT-format comes with all basic linear algebra operations: addition, dot product, matrix-by-vector multiplication (we will clarify what is meant by "matrix in the TT-format" later on). Thus, one can use Krylov time methods for solving the nonstationary problem of form (1). In this paper we will use another approach.

The tensors of fixed TT-ranks form a low-dimensional manifold $\mathcal{M} = \mathbb{TT}_{\mathbf{r}}$ in the space of all d-dimensional tensors [18]. We are looking for the approximate solution of (1) that lies on this manifold for every t: the real trajectory $y(t)$ should be projected onto $\mathcal{M}$.

There are several possibilities to perform such projection. A natural way is to just perform best approximation for the particular time t by using the robust approximation procedures, that are readily available for the TT-format. However, as in two and three-dimensional cases, this approach may not be optimal: we are interested in the approximation of the trajectory in "whole", not in the approximation of the individual parts of it. If the "true" solution is close to the manifold for all t there is no problem. However, in several important problems (including time-dependent Schrödinger equation), it may not be true since the solution may incorporate a large noise, coming, for example, from artificial reflections from the boundaries. At some moment, these unwanted components may become comparable to the "true" signal that we are interested in, and the optimal approximation will approximate noise, rather then maintain the closeness to the original trajectory.

Thus, the approximation of the time-dependent problems is defined using the so-called *Dirac-Frenkel variational principle*, which states that the optimal solution $x(t)$ should satisfy

$$\langle \frac{dy}{dt} - Ay, \delta y \rangle = 0, \quad \delta y \in T_y\mathcal{M}, \tag{4}$$

where $T_y\mathcal{M}$ is the tangent space of the manifold $\mathcal{M}$ taken at the point y. Provided that y_0 belongs to $\mathcal{M}$, (4) defines a trajectory on $\mathcal{M}$ that is "close" to the true solution $y(t)$ in the variational sense. We refer the reader to the book [22] for more details. Another approach can be based on standard time-stepping scheme with approximation, and it was considered in [8] for several parabolic problems.

Since (4) defines a trajectory on the manifold of tensors with small TT-ranks, the solution can be defined by a small number of parameters, i.e., the cores G_k that will now depend on time t. Thus, it is important to write down the equations for these quantities. The analogous procedure is well-established for the case of the Tucker format which is in fact used in the MCTDH method, where the equations for the factors and for the core are derived and then solved numerically [1]. The benefit of using the TT-format instead of the Tucker format is that with a fixed rank the number of parameters in the TT-format does not grow exponentially in the number of degrees of freedom. By increasing the rank the accuracy of the approximation is also increasing, thus we can study the quality of the approximation numerically. Note that in some cases there are theoretical estimates on the structure of the solution [10].

The paper is organized as follows. In Section 2 basic notation (which is not yet standard) is presented, along with known facts about the TT-format. In Section 3 a dynamical TT-approximation is described and the equations of motion for the cores are obtained for the general time-dependent system. In Section 4 the linear case is described in more details. It is the central part of the paper: an efficient splitting scheme is presented for solving the obtained equations of motion in a stable way, and also its properties are investigated. In Section 5 numerical experiments are given.

[1] Note, that this approach is not applicable to the canonical format, since the tensors of fixed canonical rank do not form a manifold

2 Notation and basic facts

2.1 Parameter-dependent matrices

In this section several basic facts and definitions are recollected. Through the paper, any multiindex object, i.e. $G_k(i_k)$ is a matrix, depending on parameters, and these parameters are listed in the brackets. The size of this parameter-dependent matrix should be clear from the context (i.e, $r_{k-1} \times r_k$). The number of parameters can be arbitrary. A parameter-dependent matrix $G_k(i_k)$ is in fact a collection of n_k matrices of the same size. This convention allows us to replace certain summations by matrix products, simplifying the presentation. For example, multiplication of two parameter-dependent matrices yields a new parameter-dependent matrix with larger number of parameters:

$$W(i_k, i_{k+1}) = G_k(i_k)G_{k+1}(i_{k+1}).$$

Individual elements of the parameter-dependent matrices are referred to as $G_k(i_k)_{(\alpha_{k-1},\alpha_k)}$. This is an important operation, and we will use a special notation for that.

Definition 2.1. *For a given parameter-dependent matrix $G_k(i_k)$, ${}^{<}G_k^{>}(\alpha_{k-1}, i_k, \alpha_k)$ is a three-dimensional tensor with elements*

$${}^{<}G_k^{>}(\alpha_{k-1}, i_k, \alpha_k) = G_k(i_k)_{(\alpha_{k-1},\alpha_k)}. \tag{5}$$

Note, that this notation is consistent: ${}^{<}G_k^{>}(\alpha_{k-1}, i_k, \alpha_k)$ is a parameter-dependent matrix of size 1×1, i.e., a scalar. The notation (5) can be extended to extract either row or column index of a parameter dependent matrix.

Definition 2.2. *Given a parameter-dependent matrix $G_k(i_k)$, by ${}^{<}G_k(\alpha_{k-1}, i_k)$ and $G_k^{>}(i_k, \alpha_k)$ we denote parameter-dependent matrices of sizes $1 \times r_k$ and $r_{k-1} \times 1$ respectively with elements*

$${}^{<}G_k(\alpha_{k-1}, i_k) = G_k(i_k)_{(\alpha_{k-1},:)}, \quad G_k^{>}(i_k, \alpha_k) = G_k(i_k)_{(:,\alpha_k)}.$$

The size of ${}^{<}G_k(\alpha_{k-1}, i_k)$ is $1 \times r_k$, and the size of $G_k^{>}(i_k, \alpha_k)$ is $r_{k-1} \times 1$. One should be very careful, since matrix-by-matrix product is noncommutative, and the order is very important with this notation.

Using the above conventions, the TT-format can be written in a simple "separable" form:

$$A(i_1, \ldots, i_d) = G_1(i_1)G_2(i_2)\ldots G_d(i_d). \tag{6}$$

In this paper, we will need the following "basic identity".

Lemma 2.3. *Suppose $A_1(i_1), \ldots, A_p(i_p)$ and $B_1(i_1), \ldots, B_p(i_p)$ are parameter-dependent matrices such that the matrix-by-matrix products*

$$A(i_1, \ldots, i_p) = A_1(i_1)A_2(i_2)\ldots A_p(i_p)$$

and

$$B(i_1, \ldots, i_p) = B_1(i_1)B_2(i_2)\ldots B_p(i_p)$$

are 1×1 matrices, i.e., scalars. Then,

$$\Big(A_1(i_1)\ldots A_p(i_p)\Big)\Big(B_1(i_1)\ldots B_p(i_p)\Big) = \Big(A_1(i_1) \otimes B_1(i_1)\Big)\ldots\Big(A_p(i_p) \otimes B_p(i_p)\Big). \tag{7}$$

Proof. It is sufficient to note that for scalars, which are 1×1 matrices, the product is equal to the tensor product, and use the following well-known property of the tensor product:

$$(AB) \otimes (CD) = (A \otimes C)(B \otimes D),$$

where A, B, C, D are arbitrary matrices such that products AB and CD are defined. □

Definition 2.4. *(TT-manifold) Let $\mathbf{r} = (r_1, \ldots, r_{d-1})$ be a given* rank parameter. *Then $\mathbb{TT}_{\mathbf{r}}$ is the set of tensors that can be represented in the form (6).*

2.2 Vectors and matrices in the TT-format

The functions and their discretizations are in this paper are directly related to d-dimensional tensors. For brevity, we will use the term "vector in the TT-format". By that we implicitly assume, that the vector in question has length $N = n_1 n_2 \dots n_d$ and it can be considered as an $n_1 \times \dots \times n_d$ tensor which has low TT-ranks. Square matrices acting on such vectors have size $N \times N$ and their elements can be naturally indexed by a $2d$-tuple $(i_1, \dots, i_d, j_1, \dots, j_d)$, $1 \le i_k, j_k \le n_k$, i.e. $M(i_1, \dots, i_d, j_1, \dots, j_d)$. A matrix M is said to be in the TT-format, if

$$M(i_1, \dots, i_d, j_1, \dots, j_d) = M_1(i_1, j_1) M_2(i_2, j_2) \dots M_d(i_d, j_d), \tag{8}$$

where $M_k(i_k, j_k)$ is an $r_{k-1} \times r_k$ matrix, and $r_0 = r_d = 1$. If all TT-ranks are equal to 1, then (8) reduces to

$$M = M_1 \otimes \dots \otimes M_d,$$

i.e. it has Kronecker rank 1. Thus, (8) is a generalization of a standard low-rank approximation of high-dimensional operators [32, 2].

2.3 Left and right orthogonalization of the TT-format

The TT-representation (6) is non-unique, since it is invariant under a transformation

$$G'_k(i_k) := G_k(i_k) S, \quad G'_{k+1}(i_{k+1}) := S^{-1} G_{k+1}(i_{k+1})$$

for any nonsingular matrix S. Using such transformations, certain cores of the TT-representation can be made *orthogonal*. There are two types of orthogonality of cores: left-orthogonality and right-orthogonality. First, introduce a notion of left and right unfolding of the core.

Definition 2.5. *(Left and right unfoldings) For a given parameter-dependent matrix $G_k(i_k)$, its left unfolding, denoted by $L(G_k(i_k))$ is defined as*

$$L(G_k(i_k)) = \begin{pmatrix} G_k(1) \\ G_k(2) \\ \vdots \\ G_k(n_k) \end{pmatrix}. \tag{9}$$

The matrix $L(G_k(i_k))$ has size $(r_{k-1} n_k) \times r_k$. The right unfolding $R(G_k(i_k))$ is defined as

$$R(G_k(i_k)) = \begin{pmatrix} G_k(1) & G_k(2) & \cdots & G_k(n_k) \end{pmatrix}. \tag{10}$$

The matrix $R(G_k(i_k))$ has size $r_{k-1} \times (n_k r_k)$.

Using $L(\cdot)$ and $R(\cdot)$ operators, it is easy to define left and right orthogonality of cores.

Definition 2.6. *(Left and right orthogonality) The core $G_k(i_k)$ is said to be left-orthogonal, if $L(G_k(i_k))$ has orthonormal columns, and right-orthogonal, if $R(G_k(i_k))$ has orthonormal rows.*

The cores $G_1(i_1), \dots G_k(i_k)$ can be made left-orthogonal by equivalent transformations, starting from the first core. Indeed $G_1(i_1)$ can be written as

$$G_1(i_1) = Q_1(i_1) R,$$

where $Q_1(i_1)$ is left-orthogonal by applying the QR-decomposition to a $G_1^{(L)}$. Then, the matrix R is incorporated into the second core:

$$G'_2(i_2) = R G_2(i_2),$$

and the same procedure is applied to the modified core $G'_2(i_2)$, and the second and the third core will be modified, an so on. This procedure is called *orthogonalization* of cores [26], and can be implemented in $\mathcal{O}(dnr^3)$ complexity.

3 Dynamical TT-approximation

3.1 Equations of motion in the general case

In this section a general framework for the dynamical low-rank approximation in the TT-format will be derived, following the ideas of [20, 19, 22].

Suppose that $X(t)$ is an $n_1 \times n_2 \times \dots n_d$ d-dimensional tensor depending on time, i.e., a solution to a certain nonstationary problem. We want to approximate $X(t)$ by $Y(t)$ that belongs to a manifold $\mathcal{M}$ of tensors will fixed TT-ranks. Of course, it can be done "pointwise" by the TT-SVD algorithm, but then the dependence of the TT-factors will not be smooth, just as in the two-dimensional case. For the approximation the analogue of the Dirac-Frenkel variational principle will be used. The projected (approximate) trajectory $Y(t)$ should satisfy the variational condition

$$\langle \frac{dX}{dt} - \frac{dY}{dt}, \delta Y \rangle = 0, \quad \delta Y \in \mathcal{T}_Y \mathcal{M}, \tag{11}$$

where

$$\delta Y = \delta Y_1(i_1) Y_2(i_2) \dots Y_d(i_d) + Y_1(i_1) \delta Y_2(i_2) \dots Y_d(i_d) + \dots + Y_1(i_1) \dots \delta Y_d(i_d). \tag{12}$$

If we impose left-orthogonality conditions, (12) has to be equipped with *gauge conditions* of form

$$\sum_{i_k} Y_k^*(i_k) \delta Y_k(i_k) = 0, \quad \sum_{i_k} Y_k^*(i_k) Y_k(i_k) = I_{r_k}. \tag{13}$$

The equations of (12) and (13) have to be solved for a given $\delta Y \in \mathcal{T}_Y \mathcal{M}$ to get $\delta Y_k(i_k)$. Let us perform an auxiliary orthogonalization of $Y_d(i_d), \dots, Y_2(i_2)$ from right to left,

$$Y_k(i_k) \dots Y_d(i_d) = S_{k-1} Z_k(i_k) \dots Z_d(i_d), \quad k = 2, \dots, d, \tag{14}$$

where the cores $Z_k(i_k)$ are right orthogonal. Then, equations (12) are rewritten as

$$\begin{aligned} \delta Y = &\Big(\delta Y_1(i_1) S_1\Big) Z_2(i_2) \dots Z_d(i_d) + Y_1(i_1) \Big(\delta Y_2(i_2) S_2\Big) Z_3(i_3) \dots Z_d(i_d) + \\ &+ \dots + Y_1(i_1) \dots \delta Y_d(i_d). \end{aligned} \tag{15}$$

To get $\delta Y_k(i_k)$, contract (15) from the left with $Y_1(i_1), \dots, Y_{k-1}(i_{k-1})$, and from the right with $Z_d(i_d), \dots, Z_{k+1}(i_{k+1})$:

$$\delta Y_k(i_k) S_k = P_k(i_k) - Y_k(i_k) \Phi_k,$$

where

$${}^{<}P_k^{>}(\alpha_{k-1}, i_k, \alpha_k) = \sum_{i_s, s \neq k} [Y_{k-1}^{>}(i_{k-1}, \alpha_{k-1})]^* \dots Y_1^*(i_1) \delta Y(i_1, \dots, i_d) Z_d^*(i_d) \dots [{}^{<}Z_{k+1}(\alpha_k, i_{k-1})]^*.$$

Since $[Y_{k-1}^{>}(i_{k-1}, \alpha_{k-1})]^* \dots Y_1^*(i_1)$ and $Z_d^*(i_d) \dots [Z_{k+1}^{>}(\alpha_k, i_{k-1})]^*$ are numbers, one have

$$\begin{aligned} [Y_{k-1}^{>}(i_{k-1}, \alpha_{k-1})]^* \dots Y_1^*(i_1) &= \overline{Y_1(i_1) \dots Y_{k-1}^{>}(i_{k-1}, \alpha_{k-1})}, \\ Z_d^*(i_d) \dots [Z_{k+1}^{>}(\alpha_k, i_{k-1})]^* &= \overline{Z_{k+1}^{>}(\alpha_k, i_{k-1}) \dots Z_d(i_d)}, \end{aligned}$$

where the overline denotes the complex conjugate. To determine Φ_k it is sufficient to use the gauge conditions (13). Thus, we have

$$\begin{aligned}
\delta Y_k(i_k) S_k &= P_k(i_k) - Y_k(i_k)\Phi_k =: F_k(i_k), \quad k = 1, \ldots, d-1, \\
\delta Y_d(i_d) &= P_d(i_d), \\
\overline{{}^{<}P_k^{>}(\alpha_{k-1}, i_k, \alpha_k)} &:= \sum_{i_s, s\neq k} \overline{\delta Y(i_1, \ldots, i_d)} Y_1(i_1) \ldots Y_{k-1}^{>}(i_{k-1}, \alpha_{k-1})^{<} Z_{k+1}(\alpha_k, i_{k+1}) \ldots Z_d(i_d), \\
\Phi_k &= \sum_{i_k} Y_k^*(i_k) P_k(i_k).
\end{aligned} \tag{16}$$

Note, that $F_k(i_k)$ can be written as

$$F_k(i_k) = P_k(i_k) - Y_k(i_k) \sum_{i'_k} Y_k^*(i'_k) P_k(i'_k),$$

and since $L(Y_k(i_k))$ is a matrix with orthonormal columns (due to left-orthogonality), each column of the matrix $L(F_k(i_k))$ is just a projection of the corresponding column of $L(P_k(i_k))$ onto the space, orthogonal to the column space of the matrix $L(Y_k(i_k))$,

$$L(F_k(i_k)) = L(\Pi^{\perp}_{Y_k(i_k)} P_k(i_k)), \tag{17}$$

where by $\Pi^{\perp}_{Y_k(i_k)}$ is the orthogonal projector, discussed above. The form (17) is very important, since the projector is invariant under multiplication of $Y_k(i_k)$ from the right by any non-singular matrix, i.e.

$$\Pi^{\perp}_{Y_k(i_k)} = \Pi^{\perp}_{Y_k(i_k)W}.$$

The final form for the equations of motion looks like

$$\begin{aligned}
\frac{dY_k(i_k)}{dt} S_k &= \Pi^{\perp}_{Y_k(i_k)} P_k(i_k), \quad k = 1, \ldots, d-1, \\
\frac{dY_d(i_d)}{dt} &= P_d(i_d), \\
\overline{{}^{<}P_k^{>}(\alpha_{k-1}, i_k, \alpha_k)} &= \sum_{i_s, s\neq k} \overline{\frac{dX(i_1, \ldots, i_d)}{dt}} Y_1(i_1) \ldots Y_{k-1}^{>}(i_{k-1}, \alpha_{k-1})^{<} Z_{k+1}(\alpha_k, i_{k+1}) \ldots Z_d(i_d),
\end{aligned} \tag{18}$$

where $Y_k(i_k)$, S_k, Φ_k, P_k, Z_k depend on time t. Note, that Z_k and S_k are auxiliary variables defined by the orthogonalization procedure (14), and can be considered as known functions of Y_k.

From now on, we consider the case when $X(t)$ is the solution of the equation of form

$$\frac{dX}{dt} = F(X, t). \tag{19}$$

3.2 Difficulties in the solution of the equations of motion

Equation (18) can be used to compute the dynamics of the "reduced system", provided that $P_k(i_k)$ can be computed cheaply. If it is the case, one can use any suitable ODE solver. However, this is not a good idea due to several reasons. First of all, the matrices S_k can be ill-conditioned, thus, the system in the variables $Y_1(i_1), \ldots, Y_d(i_d)$ will be stiff. For example, the following situation is not uncommon: the initial value $Y(0)$ has a very nice structure (say, rank-1), but during time evolution the one has to use larger ranks. To put the solution

into the manifold with higher ranks, the simplest choice is to use pad $Y_k(i_k)$ by zeros. In this case, the matrices S_k will be singular.

Another source of stiffness is that the different $Y_k(i_k)$ can correspond to different physical properties (for example, different scales), thus each of them would require different time steps. For example, the solution may be constant along one direction, and vary a lot along another — the time step should be taken as the smallest among two of them.

Therefore, a specific time-stepping scheme should be constructed. A natural idea is to use splitting: first make a time step in the first block variable $Y_1(i_1)$, modify the right-hand side, then in the second, and so on. The simplest splitting scheme has first order in time, but it is simple and cheap to implement, and we will show also, that it allows to avoid the inversion of the matrices S_k and also for certain important cases, will have some conservation properties. A benefit of the splitting scheme is that by using so-called Strang-Marchuk splitting the second order in the timestep can be achived at no computational cost.

3.3 Splitting scheme

A natural idea to solve the system of equations (18) is to use the splitting scheme. The idea is as follows. First, we make a time step in the first core, $Y_1(i_1)$ from t to $t + \widehat{\tau}$ with all other cores fixed, then in the second core from $t + \widehat{\tau}$ to $t + 2\widehat{\tau}$, and so on. In order to approximate the evolution of the full system on a time interval $[t, t + \tau]$ the fractional time step $\widehat{\tau} = \frac{\tau}{d}$ should be used.

The whole time interval is split into d intervals. at the k-th step, only the k-th core is modified accordind to the equation

$$\frac{dY_k(i_k)}{dt} S_k = \Pi^{\perp}_{Y_k(i_k)} P_k(i_k), \quad k = 1, \ldots, d-1, \tag{20}$$

where $P_k(i_k)$ is defined by (18) with $Y_1(i_1), \ldots, Y_{k-1}(i_{k-1}), Z_{k+1}(i_{k+1}), \ldots, Z_d(i_d)$ taken from the previous $(k-1)$ time step, i.e., do not depend on. The $\frac{dX}{dt} = F(X, t)$ term (see (19)) depends on the whole tensor, not on the particular component. Thus, it is convenient to rewrite (20) in a new variable

$$\widehat{Y}_k(i_k) = Y_k(i_k) S_k.$$

$$\frac{d\widehat{Y}_k(i_k)}{dt} = \Pi^{\perp}_{\widehat{Y}_k(i_k)} P_k(i_k), \quad k = 1, \ldots, d-1, \tag{21}$$

The main advantage of this tranformation is that the right-hand side can be computed in a stable way without inversion of the matrix S_k. Indeed, the projector does not change and can be computed in a stable way via the QR-decomposition. $P_k(i_k)$ depends only on the full tensor, which can be written as

$$Y = Y_1(i_1) \ldots \widehat{Y_k}(i_k) Z_{k+1}(i_{k+1}) \ldots Z_d(i_d).$$

The question is how to compute the right hand side for each equation (21) in a fast way. We will show how it can be done for a very important practical case, when the initial system is a linear system.

After $\widehat{Y}_k(i_k)$ has been modified, it is easy to recover $Y_k(i_k)$ as the Q-factor of $L(\widehat{Y}_k(i_k))$.

3.4 The second-order splitting scheme

The splitting scheme described in the previous section has only first order in time. However, it is very simple to modify it to achieve the second order in the time step, using the so-called *Strang splitting*. Its idea is as follows. For a general system of form

$$\frac{du}{dt} = A(u,t) + B(u,t),$$

we make three fractional steps: half-step in $A(u,t)$, then full step in $B(u,t)$ and finally half-step in $A(u,t)$. This scheme has second order in time. In our settings we have not two, but d operators in the right-hand side. It means, that we make a time step of length $\frac{\tau}{2d}$ in all cores except the last one (starting from 1 and going to the $(d-1)$-th core), and a time step $\frac{\tau}{d}$ in the last core, and then the process is repeated in the opposite direction: we make a time step of length $\frac{\tau}{2d}$ in the $d-1$ core, $d-2$ core and so on until the first core is reached. This is exactly how the Strang splitting looks for this particular system of ODEs we are trying to solve, and the cost is increased only twice, but the benefit is great: instead of the first order in time we have the second order in time.

4 The linear case

4.1 Equations of motion for the linear system case

Now let us concentrate on the numerical solution method in the linear case, when the tensor Y is the optimal dynamical low-rank approximation to the solution $x(t)$ of a linear high-dimensional equation of form

$$\frac{\partial x}{\partial t} = Ax, \quad x \in \mathbb{C}^{n_1 \otimes n_2 \otimes \ldots \otimes n_d}, \quad x(0) = x_0. \tag{22}$$

We will suppose that both A and x_0 are given in the TT-format, and the TT-ranks that specify the manifold are also given.

The right-hand side of the equations (18), defined by $P_k(i_k)$, can be calculated in a fast way. Indeed, if A is in the TT-format with cores $A_k(i_k, j_k)$, and Y has cores $Y_k(i_k)$, the cores $W_k(i_k)$ of the tensor, corresponding to the matrix-by-vector product are [26].

$$W_k(i_k) = \sum_{j_k} A_k(i_k, j_k) \otimes Y_k(j_k). \tag{23}$$

In other words, $\frac{dX}{dt}$ is also in the TT-format with the cores $W_k(i_k)$ given by (23). Therefore,

$$\begin{aligned} {}^{<}P_k^{>}(\alpha_{k-1}, i_k, \alpha_k) = \sum_{i_s, s \neq k} & \overline{Y_1(i_1)} \ldots \overline{Y_{k-1}^{>}(i_{k-1}, \alpha_{k-1})} \\ & W_1(i_1) \ldots W_d(i_d) {}^{<}\overline{Z_{k+1}(\alpha_k, i_{k+1})} \ldots \overline{Z_d(i_d)}. \end{aligned}$$

Now, represent the product $W_1(i_1) \ldots W_d(i_d)$ as

$$W_1(i_1) \ldots W_d(i_d) = \sum_{\beta_{k-1}, \beta_k} W_1(i_1) \ldots W_{k-1}^{>}(i_{k-1}, \beta_{k-1}) {}^{<}W_k^{>}(\beta_{k-1}, i_k, \beta_k) {}^{<}W_{k+1}(\beta_k, i_{k+1}) \ldots W_d(i_d).$$

The summation over $i_1, \ldots, i_{k-1}$ can be then separated from the summation over $i_{k+1}, \ldots, i_d$, and the expession for $P_k(\alpha_{k-1}, i_k, \alpha_k)$ can be written as

$$P_k(i_k) = \Psi_{k-1}^{(L)} W_k(i_k) \Psi_{k+1}^{(R)}, \tag{24}$$

where

$$\begin{aligned} {}^{<}\Psi_{k-1}^{(L)>}(\alpha_{k-1}, \beta_{k-1}) &= \sum_{i_1, \ldots, i_{k-1}} \overline{Y_1(i_1)} \ldots \overline{Y_{k-1}^{>}(i_{k-1}, \alpha_{k-1})} W_1(i_1) \ldots W_{k-1}^{>}(i_{k-1}, \beta_{k-1}), \\ {}^{<}\Psi_{k+1}^{(R)>}(\alpha_k, \beta_k) &= \sum_{i_{k+1}, \ldots, i_d}^{<} W_{k+1}(\beta_k, i_{k+1}) \ldots W_d(i_d) \overline{Z_{k+1}(\alpha_k, i_{k+1})} \ldots \overline{Z_d(i_d)} \end{aligned} \tag{25}$$

Using the "basic identity", we have

$$^{<}\Psi_{k-1}^{(L)>}(\alpha_{k-1},\beta_{k-1}) = \sum_{i_1,\ldots,i_{k-1}} \overline{Y_1(i_1)}\ldots\overline{Y_{k-1}^{>}(i_{k-1},\alpha_{k-1})}W_1(i_1)\ldots W_{k-1}^{>}(i_{k-1},\beta_{k-1}) = \Big(\sum_{i_1}\overline{Y_1(i_1)}\otimes W_1(i_1)\Big)\ldots\Big(\sum_{i_{k-1}}\overline{Y_{k-1}^{>}(i_{k-1},\alpha_{k-1})}\otimes W_{k-1}^{>}(i_{k-1},\beta_{k-1})\Big).$$

Analogous representation holds for $\Psi^{(R)}$. Finally, we have the following compact expression for the matrices $\Psi^{(L)}$, $\Psi^{(R)}$ (we also plugged in the expression (23) for $W_k(i_k)$).

$$\begin{aligned} ^{<}\Psi_{s}^{(L)}(\alpha_s) &= \sum_{\alpha_{s-1},i_s,j_s} {}^{<}\Psi_{s-1}^{(L)}(\alpha_{s-1})\Big(\overline{^{<}Y_s^{>}(\alpha_{s-1},i_s,\alpha_s)}\otimes A_s(i_s,j_s)\otimes Y_s(j_s)\Big),\\ \Psi_{s}^{(R)>}(\alpha_{s-1}) &= \sum_{\alpha_s,i_s,j_s}\Big(\overline{^{<}Z_s^{>}(\alpha_{s-1},i_s,\alpha_s)}\otimes A_s(i_s,j_s)\otimes Y_s(j_s)\Big)\Psi_{s+1}^{(R)>}(\alpha_s),\\ \Psi_0^{(L)} &= 1,\quad \Psi_{d+1}^{(R)} = 1. \end{aligned} \tag{26}$$

4.2 Splitting scheme for the linear case

The splitting schemes (both first and second order) in the linear case reduce to the solution of problems of form

$$\frac{d\widehat{Y}_k(i_k)}{dt} = \Pi^{\perp}_{\widehat{Y}_k(i_k)}\Big(\Psi_{k-1}^{(L)}\Big(\sum_{j_k}A_k(i_k,j_k)\otimes\widehat{Y}_k(j_k)\Big)\Psi_{k+1}^{(R)}\Big), \tag{27}$$

and the sought core is recovered as a Q-factor:

$$[L(Y_k(i_k)), S_k] = \mathtt{QR}(L(\widehat{Y}_k(i_k)),\quad k=1,\ldots,d-1. \tag{28}$$

For the last core, the equation is just a linear system of ODEs:

$$\frac{dY_d(i_d)}{dt} = \Psi_{d-1}^{(L)}\Big(\sum_{j_d}A_d(i_d,j_d)\otimes Y_d(j_d)\Big), \tag{29}$$

The complete numerical scheme is summarized in the Algorithm 1 for the first-order scheme.

4.3 Energy conservation

One of the important applications of the dynamical low-rank approximation is the quantum molecular dynamics, when $A = iH$, and H is a Hermitian matrix. In this case, the solution operator

$$S(t) = e^{iHt},$$

is unitary, and the norm of the solution is constant. A good feature of the proposed time-splitting schemes in the linear case is that they also conserve the norm for the skew-Hermitian matrices A.

Theorem 4.1. *Let $H = H^*$ be a matrix in the TT-format, $A = iH$, then for any starting vector Y in the TT-format the approximation $\widehat{Y}_1$ generated by Algorithm 1 has the same norm as Y for any τ.*

Algorithm 1 First order splitting scheme for dynamical TT-approximation

Require: Matrix A in the TT-format with cores $A_k(i_k, j_k)$, vector Y in the TT-format with cores $Y_1(i_1), \ldots, Y_d(i_d)$, time step length τ

Ensure: Dynamical low TT-rank approximation $\widetilde{Y}$ to the solution of the initial value problem

$$\frac{dx}{dt} = Ax, \quad x(0) = y,$$

at time τ.

1: {Preliminary step}
2: $\widetilde{\tau} := \frac{\tau}{d}$
3: Orthogonalize Y from right to left:

$$Y = \widehat{Y}_1(i_1) Z_2(i_2) \ldots Z_d(i_d),$$

where $Z_2, \ldots, Z_d$ are right-orthogonal.
4: $\widetilde{Y}_1(i_1) := \widehat{Y}_1(i_1)$.
5: During orthogonalization, also compute $\Psi_k^{(R)}, k = 2, \ldots, d+1$ matrices using (26).
6: Set $\Psi_0^{(L)} = 1$.
7: **for** $k = 1$ to $d-1$ **do**
8: Solve ODE (27) on the interval $[0, \widetilde{\tau}]$ using any suitable time scheme with $\widetilde{Y}_k(i_k)$ as the initial value.
9: $[L(\widetilde{Y}_k(i_k)), S_k] = \mathtt{QR}(L(\widehat{Y}_k(i_k))$.
10: $\widetilde{Y}_{k+1}(i_{k+1}) := S_k Z_{k+1}(i_{k+1})$.
11: Compute $\Psi_k^{(L)}$ using (26).
12: **end for**
13: Solve ODE (29) using $\widetilde{Y}_d(i_d)$ as the initial value.
14: $\widetilde{Y}_d(i_d) := \widehat{Y}_d(i_d)$.

Proof. After the right-to-left orthogonalization (construction of $Z_2, \ldots, Z_d$) the norm of $\widetilde{Y}$ is equal to the norm of the first core:

$$||\widetilde{Y}||_F = ||\widetilde{Y}_1||_F = ||S_1||_F.$$

Due to (28), the norm of $\widetilde{Y}_1$ is not changing during the "local" solve, thus after $\widetilde{Y}_2$ has been computed (at the end of step 1 of Algorithm 4.1, the norm of $\widetilde{Y}_2$ is equal to the norm of the full tensor:

$$||\widetilde{Y}_2||_F = ||\widetilde{Y}||_F.$$

The norm does not change until k becomes equal to d, where there is no projector in the equation (29). If we stack all elements $Y_d(i_d)$ into a long vector q, then (29) is a linear initial value problem:

$$\frac{dq}{dt} = Bq,$$

for some matrix B. It is not difficult to show (see [9]) that B is a Galerkin matrix for A:

$$B = Q^* A Q,$$

where Q is an orthogonal matrix. Therefore, since $A^* = -A$, B is skew-Hermitian, and the norm q is constant in time. Therefore, the whole time-stepping procedure does not change the norm, provided that each local system is integrated exactly. □

Remark 4.2. *In practice, Theorem 4.1 is true only approximately. If the local systems (27),(28) are solved using some ODE solver with accuracy ε, one can expect that the norm may change also by the order of $\mathcal{O}(\varepsilon)$. We will illustrate this behaviour in the numerical experiments.*

5 Numerical experiments

5.1 High-dimensional Henon-Heiles potential, heat equation

As a first example, consider a heat equation

$$\frac{d\psi}{dt} = -H\psi, \tag{30}$$

with Henon-Heiles Hamiltonian

$$H = -\frac{1}{2}\Delta + V, \tag{31}$$

where

$$V(q_1, \dots, q_d) = \frac{1}{2}\sum_{k=1}^{d} q_d^2 + \lambda(\sum_{i=1}^{d-1} q_i^2 q_{i+1} - \frac{1}{3}q_i^3), \tag{32}$$

and $\lambda = 0.111803$.

Equation (31) will be discretized by the tensor-product spectral Chebyshev elements (see [29]) on an interval $[-7, 7]$. In the following we choose 28 spectral elements in each mode. The initial condition is set to be a product of a shifted Gaussian (which has rank 1) and a polynomial of degree 3. We are interested in the computational complexity scaling with respect to the dimension d. The manifold is chosen to be $\mathbb{TT}_{\mathbf{r}}$, where $\mathbf{r} = (10, \dots, 10)$, i.e. all TT-ranks are equal to 10.

Table 1: Timings for the heat equation with the Henon-Heiles Hamiltonian, time interval [0,1], $\tau = 1 \cdot 10^{-2}$, the manifold has ranks 10, $d = 2, 4, \dots, 64$.

Dimension	Time (sec)
2	2.77
4	21.39
8	64.82
16	142.2
32	346.9
64	832.31

We can see, that the time scaling is slightly worse than linear. It is due to the fact that the time for local adaptive ODE solvers may slightly increase with increasing dimension.

5.2 Dependence on the timestep

Now let us confirm that the proposed scheme has indeed the second order in time. The matrix H is the discretized Henon-Heiles Hamiltonian with dimension $d = 8$, the time interval is again set to $[0, 1]$, and we compare the final values $\psi(1)$ for different timesteps τ. For the reference solution, we use the solution computed with $\tau = 10^{-3}$. The results are presented in Table 2.

Table 2: The dependence of the error on the time step for the Henon-Heiles oscillator ($d = 8$), time interval $[0, 0.1]$.

τ	Error
1.000e-01	3.137e-03
5.000e-02	7.969e-04
2.500e-02	2.000e-04
1.250e-02	5.001e-05
6.250e-03	1.247e-05
3.125e-03	3.081e-06
1.563e-03	7.335e-07

5.3 The degenerate case

The proposed scheme has an important advantage over analogous approaches, known in the dynamical low-rank approximation. Since it is based on projectors, it can easily work in the *degenerate case*, when the rank is overestimated, that means, that solution in time has smaller rank, then the TT-ranks chosen (thus, all matrices S_k are singular). Such kind of situation can appear in several cases. The first case is when the initial approximation has larger ranks than the final ("stationary") solution. The second case is when the TT-ranks are underestimated: initial approximation has smaller ranks then $\psi(t)$ for sufficiently large t (i.e., the manifold was chosen incorrectly). A simple solution is to add fictious zero components to the initial approximation. This immediately leads to singularities, and is usually solved by certain regularization techniques. Our approach does not need regularization, and the same scheme works in both cases. One can also imagine more complicated dynamics, when $\psi(t)$ has smaller ranks at some times. In this case, regularization techniques are not applicable, and the largest possible rank has to be chosen. Our scheme makes it possible.

5.3.1 Rank overestimation

As a numerical illustration for the first case (rank overestimation), we will use the simplest multidimensional harmonic oscillator. In the operator form,

$$H = -\frac{1}{2}\Delta + V,$$

where

$$V = \frac{1}{2}\Big(q_1^2 + \ldots + q_d^2\Big).$$

The operator H is discretized via Chebyshev spectral elements. The initial value is chosen as a product of a shifted Gaussian and a degree-2 polynomial. It can be shown, that such kind of functions has TT-ranks bounded by 3. The solution $\psi(t)$ converges to the scalar multiple of a first eigenfunction which is a Gaussian function, which is separable and has TT-ranks exactly equal to 1. The error can be measured by the residue

$$\varepsilon(t) = \frac{||H\psi - \lambda\psi|||}{||\psi||}.$$

Figure 1 shows the evolution of singular values of different unfoldins of a tensor in time. It can be seen, that the solution converges to a rank-1 eigenvector, and the second singular values are numerically equal to zero, but it does not influence the convergence of the splitting scheme. The results are presented on Figure 1.

,

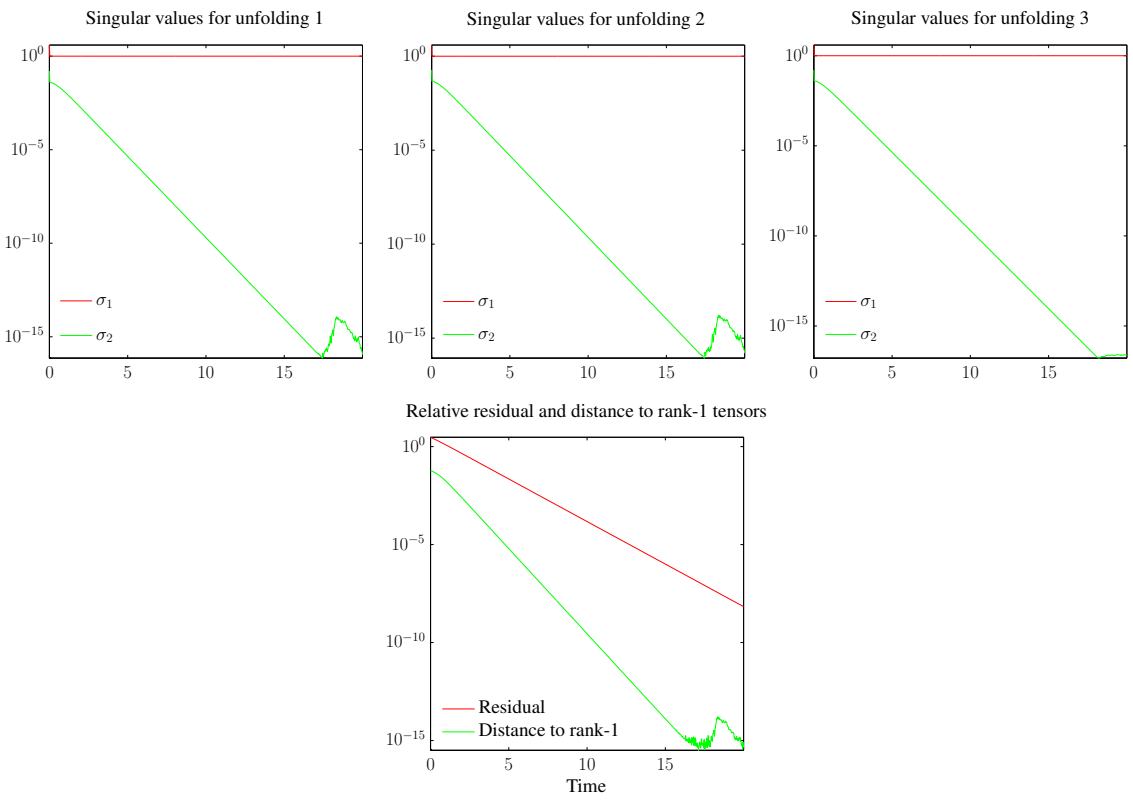

Figure 1: The dynamics of singular values of unfoldings (Heat equation with harmonic oscillator Hamiltonian, $T = 20$, $d = 4$, $\tau = 0.05$)

5.3.2 Rank underestimation

For the second case, when the rank $\psi(t)$ is larger, then the rank of the initial value $\psi(0)$, we have chosen the Henon-Heiles oscillator again, and set the initial condition to be a Gaussian function, shifted by 2 in each direction. It has rank 1, while the first eigenvector of the Henon-Heiles oscillator has non-trivial ranks. In order to put the solution onto the manifold, we explicitly added fictious zero components to the initial solution, making all TT-ranks to be equal to 10. The results are presented on Figure 2.

5.4 Time-dependent Schrödinger equation

5.4.1 Norm conservation in the skew-Hermitian case

In this subsection we will consider the application of the proposed numerical scheme to the complex case, i.e. the equation of form

$$\frac{d\psi}{dt} = iH\psi, \quad \psi(0) = \psi_0.$$

where $H = H^*$ is the Hamiltonian. Again, H is chosen to be the Henon-Heiles Hamiltonian (31). The result are presented on Figure 3.

5.4.2 Changing the manifold

It is interesting to consider the following numerical example. In the case when the solution is "going away" from the manifold, we can instantly change the manifold and increase the ranks.

,

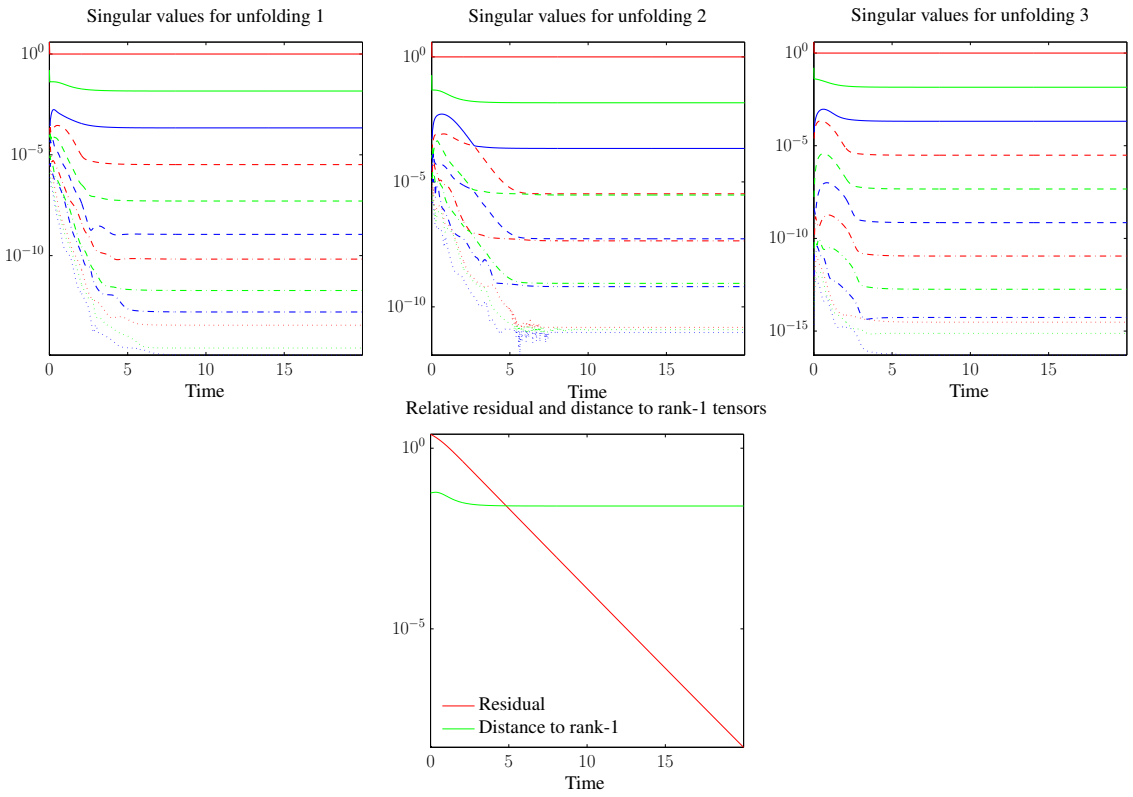

Figure 2: The dynamics of singular values of unfoldings (heat equation with Henon-Heiles Hamiltonian, $T = 20$, $d = 4$, $\tau = 0.05$)

In order to see, what happens, let us take theinitial TT-ranks to be equal to 5, the time interval $[0, 10]$, and at the time moment $t = 5$ the TT-ranks are increased to 10 by adding zero components. The results are presented in Figure 4. It can be seen, that the "bump" produces a peak in the singular values of the solution. This is due to fact, that adding zero components leads (at the first iteration step) to projection of the local matrix-by-vector product onto a *random* subspace. However, the effect is then quickly eliminated, and the first 5 singular values proceed along the same curves.

6 Conclusions and future work

In this paper we have presented an efficient time-stepping scheme for the dynamical TT-approximation. This scheme is simple, has the second order in the time step, does not suffer from the curse of dimensionality, works with tensor "as whole" (thus, applies to the degenerate case, when certain singular values of unfoldings vanish), and it is conservative in the skew-Hermitian case. We demonstrated the efficiency of our approach on several examples. In future work, we plan to adapt the developed method to the DMRG-type algorithm (to allow the smooth changing of the manifold) and also to stationary problems, like linear systems and eigenvalue problems.

The approach considered in this paper can be extended to more general tensor manifolds, in particular, to the two-level Tucker-TT-QTT format [7].

,

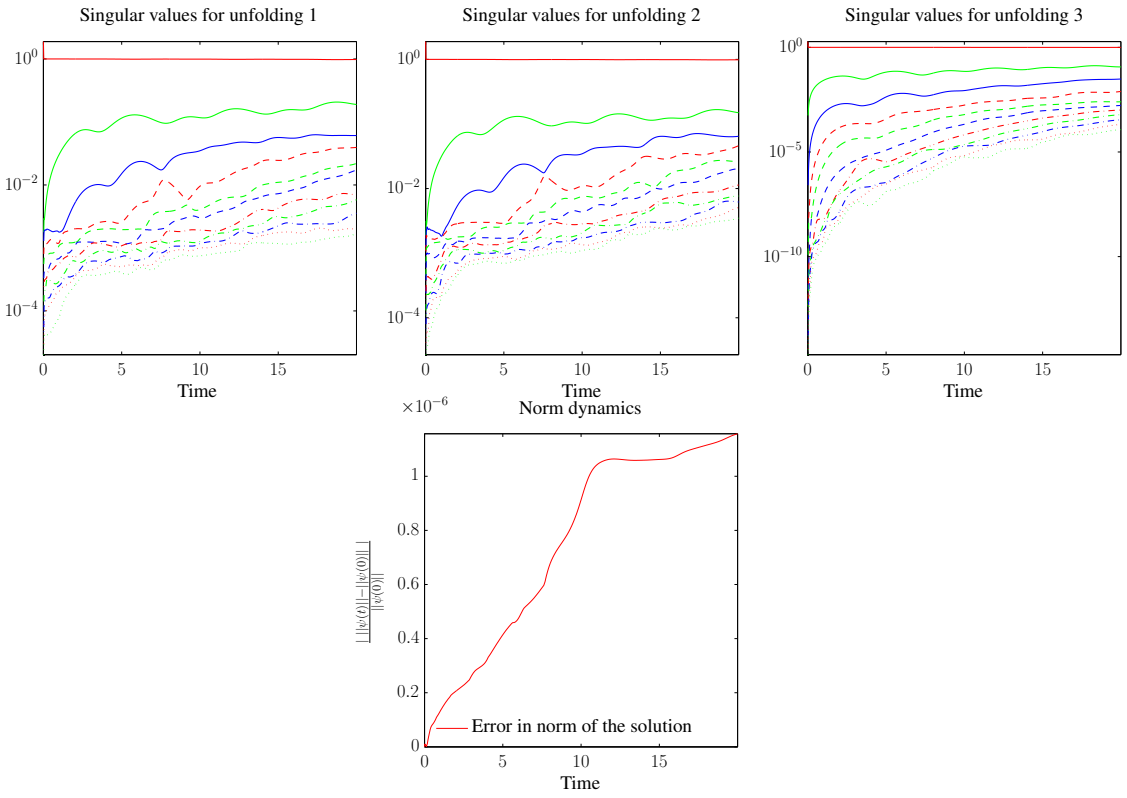

Figure 3: The dynamics of the singular values of unfoldings ($T = 20$, $d = 4$, $\tau = 0.05$, $r = 10$) and norm evolution in the complex case.

,

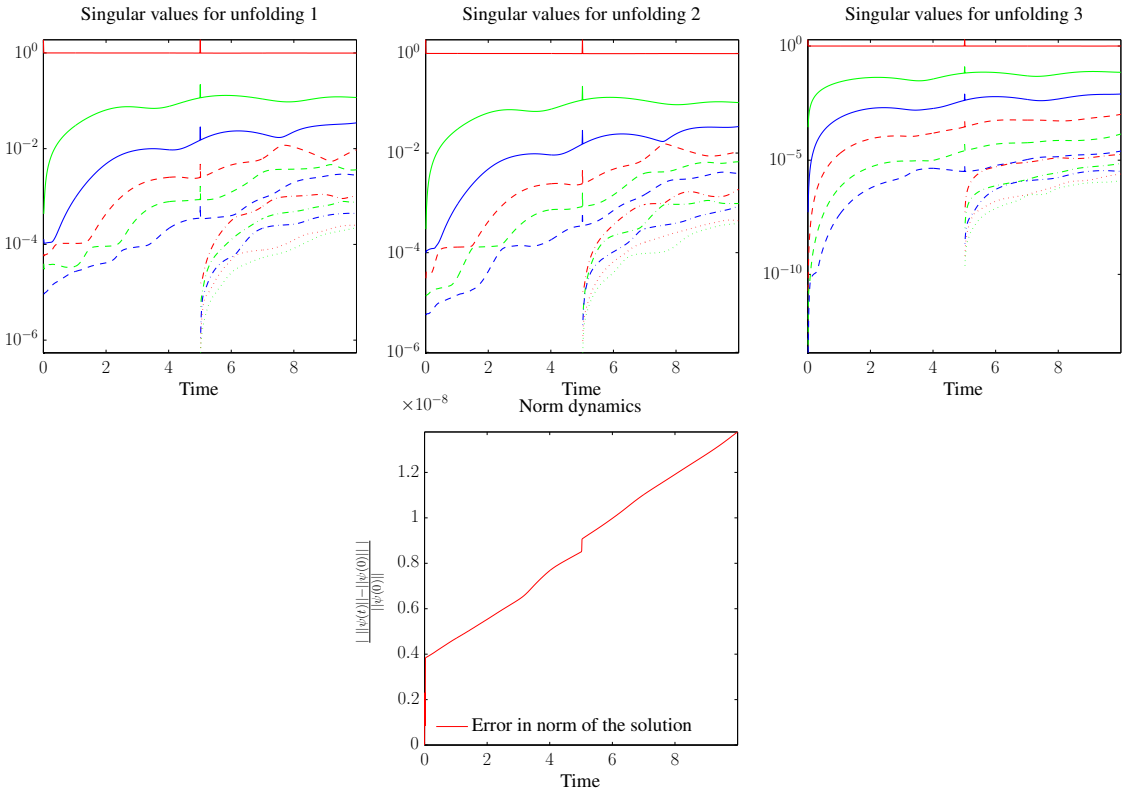

Figure 4: The dynamics of the singular values of unfoldings with a bump from $r = 5$ to $r = 10$ at $t = 5$. ($d = 4$, $\tau = 0.025$).

References

[1] O. Alon, A. Streltsov, and L. Cederbaum, *Multiconfigurational time-dependent Hartree method for bosons: Many-body dynamics of bosonic systems*, Phys. Rev. A, 77 (2008).

[2] G. Beylkin and M. J. Mohlenkamp, *Numerical operator calculus in higher dimensions*, Proc. Nat. Acad. Sci. USA, 99 (2002), pp. 10246–10251.

[3] ——, *Algorithms for numerical analysis in high dimensions*, SIAM J. Sci. Comput., 26 (2005), pp. 2133–2159.

[4] J. D. Caroll and J. J. Chang, *Analysis of individual differences in multidimensional scaling via n-way generalization of Eckart–Young decomposition*, Psychometrika, 35 (1970), pp. 283–319.

[5] L. de Lathauwer, B. de Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278.

[6] 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 (2008), pp. 1084–1127.

[7] S. Dolgov and B. Khoromskij, *Two-level Tucker-TT-QTT format for optimized tensor calculus*, Preprint 19, MPI MIS, 2012.

[8] S. Dolgov, B. N. Khoromskij, and I. V. Oseledets, *Fast solution of multi-dimensional parabolic problems in the TT/QTT–format with initial application to the Fokker-Planck equation*, Preprint 80, MPI MIS, Leipzig, 2011.

[9] S. V. Dolgov and I. V. Oseledets, *Solution of linear systems and matrix inversion in the TT-format*, Preprint 19, MPI MIS, Leipzig, 2011.

[10] I. Gavrilyuk and B. Khoromskij, *Quantized-TT-Cayley transform for computing the dynamics and the spectrum of high-dimensional Hamiltonians*, Comput. Methods in Appl. Math., 11 (2011), pp. 273–290.

[11] I. P. Gavrilyuk, W. Hackbusch, and B. N. Khoromskij, *Tensor-product approximation to the inverse and related operators in high-dimensional elliptic problems*, Computing, (2005), pp. 131–157.

[12] L. Grasedyck, *Existence and computation of low Kronecker-rank approximations for large systems in tensor product structure*, Computing, 72 (2004), pp. 247–265.

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

[14] W. Hackbusch and B. N. Khoromskij, *Low-rank Kronecker-product approximation to multi-dimensional nonlocal operators. I. Separable approximation of multi-variate functions*, Computing, 76 (2006), pp. 177–202.

[15] W. Hackbusch, B. N. Khoromskij, and E. E. Tyrtyshnikov, *Hierarchical Kronecker tensor-product approximations*, J. Numer. Math., 13 (2005), pp. 119–156.

[16] W. Hackbusch and S. Kühn, *A new scheme for the tensor representation*, J. Fourier Anal. Appl., 15 (2009), pp. 706–722.

[17] R. A. Harshman, *Foundations of the PARAFAC procedure: models and conditions for an explanatory multimodal factor analysis*, UCLA Working Papers in Phonetics, 16 (1970), pp. 1–84.

[18] S. Holtz, T. Rohwedder, and R. Schneider, *On manifolds of tensors of fixed TT–rank*, Numer. Math., (2011).

[19] O. Koch and C. Lubich, *Dynamical low rank approximation*, SIAM J. Matrix Anal. Appl., (2007).

[20] ——, *Dynamical low rank approximation of tensors*, SIAM J. Matrix Anal. Appl., (2007).

[21] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Review, 51 (2009), pp. 455–500.

[22] C. Lubich, *From quantum to classical molecular dynamics: reduced models and numerical analysis*, EMS, Zürich, 2008.

[23] H.-D. Meyer, F. Gatti, and G. A. Worth, *Multidimensional Quantum Dynamics: MCTDH Theory and Applications*, Wiley-VCH, Weinheim, 2009.

[24] M. Nest and H.-D. Meyer, *Benchmark calculations on high-dimensional Henon-Heiles potentials with the multi-configuration time dependent Hartree (MCTDH) method*, J. Chem. Phys., 117 (2002), p. 10499.

[25] I. V. Oseledets, *Compact matrix form of the d-dimensional tensor decomposition*, Preprint 2009-01, INM RAS, Moscow, 2009.

[26] ——, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.

[27] I. V. Oseledets and E. E. Tyrtyshnikov, *Breaking the curse of dimensionality, or how to use SVD in many dimensions*, SIAM J. Sci. Comput., 31 (2009), pp. 3744–3759.

[28] S. Östlund and S. Rommer, *Thermodynamic limit of density matrix renormalization*, Phys. Rev. Lett., 75 (1995), pp. 3537–3540.

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

[30] L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.

[31] C. F. van Loan, *Tensor network computations in quantum chemistry*, Preprint , 2008.

[32] C. F. van Loan and N. Pitsianis, *Approximation with Kronecker products*, in Linear algebra for large scale and real-time applications (Leuven, 1992), vol. 232 of NATO Adv. Sci. Inst. Ser. E Appl. Sci., Dordrecht, 1993, pp. 293–314.