Thomas Mach and Jens Saak

Towards an ADI iteration for Tensor Structured Equations

MPIMD/11-12v2.1 July 18, 2014

Authors Addresses:

Thomas Mach
Dept. Computer Science,
KU Leuven,
Celestijnenlaan 200A, 3000 Leuven, Belgium

thomas.mach@gmail.com

Jens Saak
Computational Methids in Systems and Control Theory
Max Planck Institute for Dynamics of Complex Technical Systems,
39106 Magdeburg, Germany,

and

Faculty for Mathematics
Technische Universität Chemnitz,
09107 Chemnitz, Germany.

saak@mpi-magdeburg.mpg.de

Imprint:

Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg

Publisher:
Max Planck Institute for
Dynamics of Complex Technical Systems

Address:
Max Planck Institute for
Dynamics of Complex Technical Systems
Sandtorstr. 1
39106 Magdeburg

http://www.mpi-magdeburg.mpg.de/preprints/

Towards an ADI iteration for Tensor Structured Equations[*]

Thomas Mach[†]

Jens Saak[‡]

July 18, 2014

Abstract

In this paper a generalization of the alternating directions implicit (ADI) iteration to higher dimensional problems is presented. This generalization solves approximately equations of the form

$$(I \otimes \cdots \otimes I \otimes A_1 + I \otimes \cdots \otimes I \otimes A_2 \otimes I + \ldots + A_d \otimes I \otimes \cdots \otimes I) \operatorname{vec}(X) = \operatorname{vec}(B),$$

with B given in the tensor train format. The solution X is approximated by $\tilde{X}$ computed also in the tensor train format. The accuracy of $\tilde{X}$ depends exponentially on the local rank of $\tilde{X}$ and on the rank of B. This is proven by adapting a result for right-hand sides of low Kronecker rank to low tensor train rank. Further a convergence proof is given for the generalized ADI iteration in the single shift case and show first ideas for more sophisticated shift strategies. The conditioning of tensor-structured equations is investigated by generalizing results for the matrix equations case. Finally numerical results show the potential and limits of this new generalization of the ADI method.

1 Introduction

This paper aims to generalize the alternating direction implicit (ADI) iteration for Lyapunov matrix equations to higher dimensional problems [37]. Shortly after the

[*]This work represents equal share of both authors. The work was started while the first author did his PhD in Magdeburg. The research was partially supported by DFG research stipend MA 5852/1-1.

[†]Dept. Computer Science, KU Leuven, Celestijnenlaan 200A, 3000 Leuven, Belgium (`thomas.mach@gmail.com`).

[‡]Max Planck Institute for Dynamics of Complex Technical Systems, 39106 Magdeburg, Germany, (`saak@mpi-magdeburg.mpg.de`) and Chemnitz University of Technology, 09107 Chemnitz, Germany.

invention of the ADI iteration as partial differential equation solver for two dimensional problems there have been attempts to generalize the method to three spatial dimensions, see, e.g., [11, 23]. Thirty years later Wachspress used the ADI iteration to solve matrix equations [37]. These matrix equations are of the form $AX + XA^T = B$ or after vectorization $(I \otimes A + A \otimes I)\mathrm{vec}(X) = \mathrm{vec}(B)$. If the right-hand side B is of low-rank, then the singular values of the solution X decay, under certain conditions, rapidly, see [1, 13, 31]. If the matrix A is sparse, then the ADI iteration is capable to solve these matrix equations comparable efficient. Lyapunov matrix equations can also be solved with the sign-function method, see, e.g., [6, 8, 33]. This has been shown to be also successful in case there the matrix A is an hierarchical matrix, [3, 5]. The sign-function method is also capable to solve Sylvester equations $(A_1X + XA_2^T = B)$ with hierarchical matrices A_1, A_2, see [4].

This paper deals with two subjects. On the one hand we will solve matrix equations with coefficient matrices of a particular structure; a Kronecker structure, analog to the Laplace operator, of the form

$$A = \Big(I \otimes \cdots \otimes I \otimes A_1 + I \otimes \cdots \otimes I \otimes A_2 \otimes I + \ldots + A_{\frac{d}{2}} \otimes I \otimes \cdots \otimes I\Big).$$

On the other hand we will directly solve linear equations of the form

$$\begin{aligned}(I \otimes \cdots \otimes I \otimes A_1 + I \otimes \cdots \otimes I \otimes A_2 \otimes I + \ldots \\ \ldots + A_d \otimes I \otimes \cdots \otimes I)\,\mathrm{vec}(X) = \mathrm{vec}(B),\end{aligned} \quad (1)$$

where all A_k are real. The vectorization of the first matrix equations directly provide us with linear equations of the second form. Hence, we are dealing only with one subject. However, we would like to ask the reader to keep in mind that our primary aim is the solution of matrix equations.

Tensor structured equations like (1) occur, e.g., in the solution of particular PDEs on d-dimensional hypercubes discretized by finite elements/differences. We will apply a generalized ADI iteration to these equations. Therefore we require a stronger condition on the right-hand side. It is not enough to have only low-rank structure. We will need tensor-train structure, which can be regarded as one possible generalization of low-rank structure to higher dimensions. We have, however to note, that our numerical examples show that it is more efficient to apply a DMRG (density matrix renormalization group) solver [40], see [10, 16] for a description of the used DMRG solver for tensor-trains. Nonetheless the generalization is of theoretical interest, since it provides new insights into the ADI method.

Equations like (1) have been investigate in [12, 19] and more recently in [10]. In [12] Grasedyck showed that the solution of a linear system with tensor product structure has a low Kronecker-rank approximation if the right-hand side is of low Kronecker-rank. The solution is approximated by a quadrature formula on an integral expression of the solution analog to (25). The solution of tensor structured equations using a Krylov subspace method was recently investigated by Tobler and Kressner in [19]. Further Dolgov and Oseledets investigate the solution of linear systems with coefficient matrices in tensor-train matrix format in [10].

In this paper we will try out a different approach based on the ADI iteration, since that is closer to our first way of interpreting these equations. The alternating direction implicit (ADI) iteration is an efficient algorithm for the solution of equations $Ax = b$, where $A = H + V$ with commuting H and V for which the solution of linear systems is cheap compared to solving with A directly. The ADI iteration originates from solving Poison's problem over the unit square. There the first summand is responsible for fulfilling the Laplace in x-direction and the second summand for the y-direction. The idea of the ADI is to alternatingly solve, the equation in x-direction and y-direction. We will generalize this idea to d dimensional problems, where we will sweep through all d dimensions.

The aforementioned Poison equation in 2D serves as our first example. We will give a detailed derivation in Section 1.2.

Example 1.1. *The stationary 2D heat equation over the unit square leads to the discrete linear system of equations $Au = f$ where*

$$A = (I \otimes \Delta_{1,h} + \Delta_{1,h} \otimes I)\,. \tag{2}$$

The Lyapunov equation related to a linear control system with the instationary heat equation consequently results as:

$$\underbrace{(I \otimes \Delta_{1,h} + \Delta_{1,h} \otimes I)}_{\Delta_{2,h}} X + X(I \otimes \Delta_{1,h} + \Delta_{1,h} \otimes I) = BB^T, \tag{3}$$

where B is the discretized input operator.

Obviously $\Delta_{2,h}$, X and B are matrices with row/column indices over the discretized unit square Ω_h. Further we obviously get an equation of the desired structure if we vectorize (3). This observation will be used in Section 3.

The remainder of the paper is structured as follows. The next subsections in this section review the necessary preliminaries that are the ADI method, the low-rank ADI, and the tenor-train decomposition. In Section 2 the conditioning of the problem will be analyzed. Section 3 is devoted to the generalization of the ADI method. In Section 4 numerical results are shown that the generalized ADI method is inferior to existing DMRG solvers, but superior to highly optimized implementations of the ADI iteration for matrix equations.

1.1 Notation

We denote with capital letters $M, T, \ldots$ matrices, tensors, and sometimes vectors—skinny matrices. The calligraphic letters $\mathcal{I}, \mathcal{J}$, and $\mathcal{K}$ are used for index sets.

We call a matrix Hurwitz if the real parts of all its eigenvalues are negative.

If $A_i = A_j$ for all pairs $i, j \in \{1, \ldots, d\}$, then we call the resulting problem of form (1) *Lyapunov tensor equation*, since this leads for $d = 2$ to the Lyapunov matrix equation. Analog the other equations are referenced as *Sylvester tensor equation.*

1.2 Classic ADI and Lyapunov Equations

Originally the ADI iteration has been developed to solve finite difference discretizations of Poisson's equation [29]:

$$\begin{aligned} -\Delta \mathbf{u} &= \mathbf{f} && \text{in } \Omega \subset \mathbb{R}^d, d = 2 \\ \mathbf{u} &= 0 && \text{on } \partial\Omega. \end{aligned}$$

For $d = 1$ using centered differences on an equidistant mesh with nodes $x_i \in \Omega$ and mesh width $h \in \mathbb{R}$ it is well known, see, e.g., [21,36], that this forms the linear system $\Delta_{1,h} u = h^2 f$, where $u_i = \mathbf{u}(x_i)$, $f_i = \mathbf{f}(x_i)$, $i = 1, \ldots, n$, the boundary conditions are reflected by $u_0 = u_{n+1} = 0$ and we have

$$\Delta_{1,h} = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}.$$

In 2D, when applying the 5-point differences star, it is equally well known that the resulting linear system of equations looks like $\Delta_{2,h} u = h^2 f$, where

$$\Delta_{2,h} = \begin{bmatrix} K & -I & & & \\ -I & K & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & K & -I \\ & & & -I & K \end{bmatrix} \quad \text{and} \quad K = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{bmatrix}.$$

It is an easy exercise to proof that then in fact we have

$$\Delta_{2,h} = \underbrace{(\Delta_{1,h} \otimes I)}_{=:H} + \underbrace{(I \otimes \Delta_{1,h})}_{=:V}. \tag{4}$$

This leads to the idea of an iteration capable of exploiting the tridiagonal structure of $\Delta_{1,h}$, which finally gave rise to the classical two step ADI iteration described by

$$\begin{aligned} (H + p_i I)\, u_{i+\frac{1}{2}} &= (p_i I - V)\, u_i + \tilde{f}, \\ (V + p_i I)\, u_{i+1} &= (p_i I - H)\, u_{i+\frac{1}{2}} + \tilde{f}, \end{aligned}$$

for certain shift parameters $p_i \in \mathbb{C}$, where the optimal parameters solve

$$\min_{\{p_1,\ldots,p_\ell\} \subset \mathbb{C}} \quad \max_{\lambda \in \Lambda(H), \mu \in \Lambda(V)} \left| \prod_{k=0}^{\ell} \frac{(p_k - \mu)(p_k - \lambda)}{(p_k + \lambda)(p_k + \mu)} \right|. \tag{5}$$

In the special case of our example we have $\Lambda(V) = \Lambda(H) = \Lambda(\Delta_{1,h})$ and thus (5) simplifies to

$$\min_{\{p_1,\ldots,p_\ell\} \subset \mathbb{C}} \quad \max_{\lambda \in \Lambda(\Delta_{1,h})} \left| \prod_{k=0}^{\ell} \frac{(p_k - \lambda)}{(p_k + \lambda)} \right|. \tag{6}$$

This approach has also been generalized to 3D problems, [11].

Wachspress [37,38] first observed that the vectorization

$$[(I \otimes F) + (F \otimes I)] \operatorname{vec} X = -\operatorname{vec}(GG^T), \tag{7}$$

of the Lyapunov equation

$$FX + XF^T = -GG^T \tag{8}$$

yields exactly the same structure as (4) and thus the Lyapunov equation is suitable for applying the ADI method. This observation lead to the ADI iteration for the Lyapunov equation:

$$\begin{aligned}(F + p_i I)\, X_{i+\frac{1}{2}} &= -GG^T - X_i \left(F^T - p_i I\right),\\ (F + \overline{p_i} I)\, X_{i+1} &= -GG^T - X_{i+\frac{1}{2}}^T \left(F^T - \overline{p_i} I\right),\end{aligned}$$

where $\overline{p_i}$ denotes the complex conjugate of $p_i \in \mathbb{C}$.

Additionally, Wachspress [38,39] provides a way to compute the optimal (in terms of the global convergence rate) ℓ shift parameters such that the ℓ-th iterate meets a prescribed relative error bound, provided the spectrum of F is real. For complex spectra with moderately sized imaginary parts of the eigenvalues, an asymptotically optimal choice is given. A fast heuristic choice of the parameters was introduced by Penzl [32], and Sabino [35] treats a potential theory based selection algorithm. The contribution in [7] combines the optimality of the Wachspress parameters and the fast computability of Penzl's parameters.

1.3 The Basic Idea of Low Rank ADI

The two key observations towards a low rank version of this iteration are that on the one hand [22,30] after rewriting it into a one step iteration

$$\begin{aligned}X_i = &-2\operatorname{Re}\,(p_i)\,(F + p_i I)^{-1} GG^T (F + p_i I)^{-T}\\ &+ (F + \overline{p_i} I)^{-1}(F - \overline{p_i} I) X_{i-1} (F - p_i I)^T (F + p_i I)^{-T},\end{aligned} \tag{9}$$

we find that (9) is symmetric. On the other hand the solution's singular values decay rapidly such that it allows for a good low rank approximation, see, e.g., [1,13,31].

Thus assuming $X_i = Z_i Z_i^H$ and $Y_0 = 0$ we can write the iteration in terms of the factors Z_i as

$$\begin{aligned}Z_1 &= \sqrt{-2\operatorname{Re}\,(p_1)}(F + p_1 I)^{-1} G,\\ Z_i &= \left[\sqrt{-2\operatorname{Re}\,(p_i)}(F + p_i I)^{-1} G,\ (F + p_i I)^{-1}(F - \overline{p_i} I) Z_{i-1}\right].\end{aligned}$$

Hence, we can write the ADI iteration such that it forms the factors by successively adding a fixed number of columns in every step. In the current formulation, however, all columns have to be processed in every step, which makes the iteration increasingly expensive. Now let $n_p \in \mathbb{N}$ be the number of shift parameters we have at hand.

Then defining the matrices $T_i := (F - \overline{p_i} I)$ and inverse matrices $S_i := (F + p_i I)^{-1}$ following [22] one can express the n_p-th, i.e., the final, iterate as

$$Z_{n_p} = \Big[S_{n_p} \sqrt{-2\,\mathrm{Re}\,(p_{n_p})} G, \;\; S_{n_p}(T_{n_p} S_{n_p-1}) \sqrt{-2\,\mathrm{Re}\,(p_{n_p-1})} G, \; \dots, \\ S_{n_p} T_{n_p} \cdots S_2 (T_2 S_1) \sqrt{-2\,\mathrm{Re}\,(p_1)} G \Big].$$

Observing that the S_j and T_j commute, one can rearrange these matrices. Note that every block of the dimension of G essentially contains its left neighbor, i.e., predecessor in the iteration. Thus one finds that the factor can be rewritten in the form

$$Z_{n_p} = \left[z_{n_p}, \; P_{n_p-1} z_{n_p}, \; P_{n_p-2}(P_{n_p-1} z_{n_p}), \; \dots, \; P_1(P_2 \cdots P_{n_p-1} z_{n_p}) \right], \tag{10}$$

where $z_{n_p} = \sqrt{-2 p_{n_p}} S_{n_p} G$ and one only needs to apply a *step operator*

$$\begin{aligned} P_i := & \frac{\sqrt{-2\,\mathrm{Re}\,(p_i)}}{\sqrt{-2\,\mathrm{Re}\,(p_{i+1})}} (F + p_i I)^{-1} (F - \overline{p_{i+1}} I) \\ = & \frac{\sqrt{-2\,\mathrm{Re}\,(p_i)}}{\sqrt{-2\,\mathrm{Re}\,(p_{i+1})}} \left[I - (p_i + \overline{p_{i+1}})(F + p_i I)^{-1} \right], \end{aligned} \tag{11}$$

to compute the new columns in every step.

Especially note that only the new columns need to be processed after rearrangement. In summary this forms an iteration that computes a low rank factorization of the solution exploiting the low rank structure of the right-hand side. This is exactly what we want to pursue in Section 3 for more general tensor structures of the right-hand side, although the direct computation will not be possible in more general cases. To this end the next subsection briefly reviews tensor structures, which will be used in this paper.

1.4 Tensor Structure

Following [14] we define a *d-mode tensor* $T \in \mathbb{R}^{\mathcal{I}}$ as vector over the product index set

$$\mathcal{I} = \mathcal{I}_1 \otimes \mathcal{I}_2 \otimes \cdots \otimes \mathcal{I}_d,$$

where $\mathcal{I}_i$ are index sets. If we split $\mathcal{I}$ into

$$\mathcal{I} = \mathcal{J} \otimes \mathcal{K},$$

with $t \subset \{1, \dots, d\}$, $\mathcal{J} = \bigotimes_{i \in t} \mathcal{I}_i$ and $\mathcal{K} = \bigotimes_{i \notin t} \mathcal{I}_i$, then there is a *matricization* $M \in \mathbb{R}^{\mathcal{J} \times \mathcal{K}}$ of T. We write for $t = \{i_1, \dots, i_j\}$

$$M = T(i_1, \dots, i_j; i_{j+1}, \dots, i_d)$$

to separate the indices into row indices (before the semicolon), and column indices (after the semicolon). Sometimes this splitting of a tensor into a matrix with product index sets as row and column index is also called *tensor unfolding*, e.g., in [9].

We use the following notation of a product of tensor T with matrix $M \in \mathbb{R}^{n_j \times n_j}$ from [9]:

$$(T \times_j M)_{i_1,\ldots,i_d} := \sum_{\alpha} T_{i_1,\ldots,i_{j-1},\alpha,i_{j+1},\ldots,i_d} M_{i_j,\alpha}.$$

Further we use the product of a matrix $M \in \mathbb{R}^{N \times N}$ with $N = \prod\limits_{k=1}^{d} n_k$ and the vectorization of a tensor $T \in R^{n_1 \times \cdots \times n_d}$:

$$(MT)_{i_1,\ldots,i_d} = (M\mathrm{vec}(T))_{i_1,\ldots,i_d} = \sum_{j_1,\ldots,j_d=1}^{n_1,\ldots,n_d} M_{i_1,\ldots i_d;j_1\ldots,j_d} T_{j_1,\ldots,j_d}.$$

If it is clear from the context that T is a tensor, then we omit the vec and assume that the result is again a tensor and not the vectorization.

For simplicity we assume that all $\mathcal{I}_i$ are of dimension n. Storing the n^d entries of a tensor becomes expensive for large d due to the exponential growth. This is often called *the curse of dimensionality*.

There are different approaches to overcome this problem, e.g., the usage of the canonical form or the tensor-train representation of the tensor. If

$$T = \sum_{\alpha=1}^{r} U_1^{(\alpha)} \otimes U_2^{(\alpha)} \otimes \cdots \otimes U_d^{(\alpha)},$$

then T is a tensor of *tensor rank* or *canonical rank* r. The right-hand side is the canonical form of the tensor. The computation of the canonical form of a tensor is unstable and difficult, except for $d = 1, 2$.

Recently *tensor-trains* [24, 26] have been introduced by Oseledets and Tyrtshnikov. Tensor-trains are a representation of tensors, which does not suffer from the curse of dimensionality. The tensor-train decomposition (TT decomposition) is given by

$$T\left(i_1, i_2, \ldots, i_d\right) \approx \sum_{\alpha_1,\ldots,\alpha_{d-1}} G_1(i_1,\alpha_1) G_2(\alpha_1, i_2, \alpha_2) \cdots \\ \cdots G_{d-1}(\alpha_{d-2}, i_{d-1}, \alpha_{d-1}) G_d(\alpha_{d-1}, i_d) = \tilde{T}.$$

This means that the tensor is represented by $(d-2)$ tensors of size $r_{j-1} \times n_j \times r_j$ and by two tensors of size $n_1 \times r_1$ and $r_{d-1} \times n_d$, so that $(d-2)nr^2 + 2nr$ $(r_j \leq r, n_j \leq n \ \forall j)$ entries have to be stored. The r_j are called the tensor train ranks, or short TT-ranks, of $\tilde{T}$. For $d = 2$ the approximation by tensor trains is equal to the low rank approximation by a factorization AB^T or $\mathrm{vec}\left(AB^T\right)$ respectively.

If T is available one can compute the TT decomposition by a sequence of SVDs of different matricizations of the tensor. There is a second approach using a generalization of the cross approximation [28].

If we vectorize the matrix equation in Example 1.1, we get a linear system with a coefficient matrix, which is a matricization of an 8-mode tensor given in canonical form,

$$\begin{pmatrix} & \underbrace{I \otimes I \otimes I \otimes \Delta_{1,h}}_{=H} & + & \underbrace{I \otimes I \otimes \Delta_{1,h} \otimes I}_{=V} & + \\ + & \underbrace{I \otimes \Delta_{1,h} \otimes I \otimes I}_{=R} & + & \underbrace{\Delta_{1,h} \otimes I \otimes I \otimes I}_{=Q} & \end{pmatrix} \underbrace{\mathrm{vec}(X)}_{=u} = \underbrace{\mathrm{vec}(B)}_{=f}. \tag{12}$$

In the last part of the Section 3 we will assume that the right-hand side B and the solution X are tensors in the tensor train format and the coefficient matrix is given in the canonical form

$$T = \sum_{\alpha=1}^{r} U_1^{(\alpha)} \otimes U_2^{(\alpha)} \otimes \cdots \otimes U_d^{(\alpha)},$$

with $r = d$ and

$$U_j^{(\alpha)} = \begin{cases} I, & \alpha \neq j, \\ A_j, & \alpha = i. \end{cases}$$

The ideas of the tensor-train decomposition have been generalized to tensor-train matrices, see [24]. Let $M = T(i_1, i_2, \ldots, i_d; j_1, \ldots, j_d)$. Then one can write M as a tensor-train matrix in the form:

$$M = T\left(i_1, i_2, \ldots, i_d; j_1, \ldots, j_d\right) \approx \sum_{\alpha_1, \ldots, \alpha_{d-1}} G_1(i_1, j_1, \alpha_1) G_2(\alpha_1, i_2, j_2, \alpha_2) \cdots$$
$$\cdots G_{d-1}(\alpha_{d-2}, i_{d-1}, j_{d-1}, \alpha_{d-1}) G_d(\alpha_{d-1}, i_d, j_d).$$

2 Conditioning of the Problem

The TT decomposition is capable to overcome the cures of dimensionality in out problem by reducing the computational complexity tremendously, but there is more we have to take care off. The conditioning of the problem is also effected by growing d. In this section we will investigate the effect of large d on the conditioning of the linear system. We will generalize some results that have been derived for Lyapunov equations (8) in [41]. The initial idea there is to investigate the vectorized form (7) of (8), which coincides with (20) for $d = 2$. We will follow the presentation there and show that it is a straight forward argumentation to extend the results to $d > 2$.

Let A be defined as

$$A = I \otimes \cdots \otimes I \otimes A_1 + I \otimes \cdots \otimes I \otimes A_2 \otimes I + \ldots + A_d \otimes I \otimes \cdots \otimes I, . \tag{13}$$

The matrix A is defined by the small matrices A_i. Thus we simplify the notation by using $\mathcal{A} := \{A_1, A_2, \ldots, A_d\}$. It is well known as Stéphanos's theorem, see, e.g., [17,19], that the eigenvalues of A are the sum of the eigenvalues of A_1 to A_d

$$\lambda_i(A) = \lambda_{i_1}(A_1) + \lambda_{i_2}(A_2) + \cdots + \lambda_{i_d}(A_d), \tag{14}$$

with $i = i_1 + i_2 n_1 + \cdots + i_d \prod_{j=1}^{d-1} n_j$. A direct consequence is that in perfect analogy to the Lyapunov/Sylvester matrix equation, the eigenvalues of A are non-zero and hence invertible if $A_1, \ldots, A_d$ are Hurwitz.

Based on this representation of the eigenvalues we can bound the smallest eigenvalue from above by

$$\min_l |\lambda_l(A)| = \min_{l_1,\ldots,l_d} \left| \sum_{k=1}^{d} \lambda_l(A_k) \right| \leq \sum_{k=1}^{d} \min_{l_k} |\lambda_{l_k}(A_k)|. \tag{15}$$

Now defining

$$m_k = \underset{j \in \{i=1,\ldots,n_k \,:\, \mathrm{Im}\,(\lambda_i(A_k)) \geq 0\}}{\operatorname{argmax}} |\lambda_j(A_k)| \tag{16}$$

and assuming that A_k are Hurwitz for all $k = 1, \ldots, d$, we find

$$\max_l |\lambda_l(A)| \geq \left| \sum_{k=1}^{d} \lambda_{m_k}(A_k) \right| \geq \left| \sum_{k=1}^{d} \mathrm{Re}\,(\lambda_{m_k}(A_k)) \right|. \tag{17}$$

Where the first inequality holds since a particular value is always smaller than the maximum over a set including this value. The second one holds due to the fact that we leave away the positive (by definition of the m_k) imaginary parts. Note that we assume real data, i.e., real matrices A_k and thus the restriction to the positive imaginary part eigenvalues is no restriction since these come in pairs anyway. Our choice here guarantees that we pick the one giving the larger result in the sum.

Furthermore, we can generalize the definition of the sep operator to

$$\mathrm{sep}(\mathcal{A}) := \min_X \frac{\|X \times_1 A_1 + X \times_2 A_2 + \cdots + X \times_d A_d\|_F}{\|X\|_F} = \min_X \frac{\|A \mathrm{vec}(X)\|_2}{\|\mathrm{vec}(X)\|_2},$$

and find that then we have

$$\left\|A^{-1}\right\|_2^{-1} = \sigma_{\min}(A) = \min_y \frac{\|Ay\|_2}{\|y\|_2} = \mathrm{sep}(\mathcal{A}). \tag{18}$$

Here the first two equalities are the obvious consequences from the definitions and the last equality follows immediately from the vectorization of the above definition.

The following lemma shows that the normality of all A_k is inherited to A. This lemma will allow us to compute the condition number in the normal case.

Lemma 2.1. *If all A_i, $i = 1, \ldots, d$ are normal, then also A is normal.*

Proof. By using the defining property for A normal, $AA^T = A^T A$, one can easily see that we have for $d = 2$:

$$\begin{aligned}
AA^T &= (A_2 \otimes I + I \otimes A_1)\,(A_2 \otimes I + I \otimes A_1)^T \\
&= (A_2 \otimes I + I \otimes A_1)\,\left(A_2^T \otimes I + I \otimes A_1^T\right) \\
&= \left(A_2 A_2^T \otimes II\right) + \left(A_2 I \otimes I A_1^T\right) + \left(I A_2^T \otimes A_1 I\right) + \left(II \otimes A_1 A_1^T\right) \\
&= \left(A_2^T A_2 \otimes II\right) + \left(I A_2 \otimes A_1^T I\right) + \left(A_2^T I \otimes I A_1\right) + \left(II \otimes A_1^T A_1\right) \\
&= (A_2 \otimes I + I \otimes A_1)^T\,(A_2 \otimes I + I \otimes A_1) \\
&= A^T A.
\end{aligned}$$

For higher d the argumentation is analogous. □

Now for the normal case, i.e., all the A_j are normal and thus A is normal, we can just pull back the 2-norm condition $\kappa_2(A)$ to the eigenvalues of the A matrix via

$$\kappa_2(A) = \|A\|_2 \left\|A^{-1}\right\|_2 = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} = \frac{\max\limits_{l} |\lambda_l(A)|}{\min\limits_{l} |\lambda_l(A)|} \overset{\forall i,j\ A_i = A_j}{=} \frac{\max\limits_{\ell_u} |\lambda_{\ell_u}(A_1)|}{\min\limits_{\ell_l} |\lambda_{\ell_l}(A_1)|}.$$

We note that this equality may only be helpful in the Lyapunov case, i.e., when all the A_k are the same, since computing all eigenvalues of A from those of the A_k $(k = 1, \ldots, d)$ is an $\mathcal{O}(d \cdot N)$ operation.

We will now face the non-normal case, for which we will derive computable bounds on the condition number. We denote by $\mathcal{T}$ the set of normalized eigenvectors of A. We then find

$$\begin{aligned}
\sigma_{\max}(A) = \|A\|_2 = \sup_{y \in \mathbb{R}^N} \frac{\|Ay\|_2}{\|y\|_2} \geq \sup_{y \in \mathcal{T}} \|Ay\|_2 = \max_{l} |\lambda_l(A)| \\
\sigma_{\min}(A) = \|A\|_2 = \sup_{y \in \mathbb{R}^N} \frac{\|Ay\|_2}{\|y\|_2} \leq \sup_{y \in \mathcal{T}} \|Ay\|_2 = \min_{l} |\lambda_l(A)|
\end{aligned}$$

We have already noted that A_j Hurwitz for all $j = 1, \ldots, d$ is sufficient for the solubility of the tensor equation. Taking this assumption we establish the following proposition:

Theorem 2.2. *Assume A_j Hurwitz for all $j = 1, \ldots, d$, then the lower bound*

$$\kappa_2(A) \geq \frac{\sum\limits_{k=1}^{d} \operatorname{Re}\left(-\lambda_{m_k}(A_k)\right)}{\sum\limits_{k=1}^{d} \min\limits_{l_k} |\lambda_{l_k}(A_k)|},$$

with m_k defined as in (16), holds for the 2-norm condition number of A.

Proof. Insert the above bounds on the singular values and the bounds (15), (17) into

$$\kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}, \tag{19}$$

exploiting that all A_j are Hurwitz. □

Obviously we may underestimate the real condition number by this bound. On the other hand it is as easily computable as the exact condition number in the normal case. However, we can also give the strict condition number in very similar, although practically useless terms. Noting that we can express the maximal singular value of A in analogous manner as the minimal one in (18) and using (19) we end up with

$$\kappa_2(A) = \frac{\max\limits_{\|X\|_F=1} \|\mathfrak{A}(X)\|_F}{\operatorname{sep}(\mathcal{A})}$$

The upper bound employs the *logarithmic norms* $\mu(A_j)$ (see [41] and references therein) of the A_j matrices, which in case of the 2-norm reduces to

$$\mu_2(A_j) = \frac{1}{2}\lambda_{\max}(A_j + A_j^T).$$

Note that the logarithmic norm is not a real norm, since it is allowed to take negative values. The upper bound to the 2-norm condition number is then given by

Theorem 2.3. *Assume $A_j \in \mathbb{R}^{n_j \times n_j}$ Hurwitz for all $j = 1, \dots, d$ and thus $\mu(A_j) < 0$ for all $j = 1, \dots, d$. Then the 2-norm condition number for A is bounded from above by*

$$\kappa_2(A) \le -\frac{2\sum\limits_{k=1}^{d} \sigma_{\max}(A_k)}{\sum\limits_{k=1}^{d} \lambda_{\max}(A_k + A_k^T)} \le -\frac{2\max\limits_k \sigma_{\max}(A_k)}{\min\limits_k \lambda_{\max}(A_k + A_k^T)}.$$

Proof. The upper bound for $\|A\|_2$ follows directly by applying the triangular inequality

$$\|A\|_2 \le \sum_{k=1}^{d} \|A_k\|_2 = \sum_{k=1}^{d} \sigma_{\max}(A_k).$$

For the upper bound on $\left\|A^{-1}\right\|_2$ analogous to [41] we consider the tensor equation

$$\sum_{j=1}^{d} H \times_j A_j = -I.$$

From Lemma 3.2 we know that the solution can be expressed as the integral

$$H = \int_0^\infty I \times_1 \exp(A_1 t) \times_2 \cdots \times_d \exp(A_d t) dt.$$

By definition of H and due to the fact that all A_j are Hurwitz, we have

$$\left\|A^{-1}\right\| = \|H\|,$$

following the argumentation in [2]. Thus we only need to bound $\|H\|_2$ to get the desired bound for $\left\|A^{-1}\right\|_2$. To get this we exploit that $\|\exp(A_j t)\|_2 \leq \exp(\mu_2(A_j)t)$ (from the defining propertiy of $\mu_2(A_j)$) in

$$\|H\|_2 \leq \int_0^\infty \prod_{k=1}^d \exp(\mu_2(A_k)t)dt \leq -\frac{1}{\sum_{k=1}^d \mu_2(A_k)} \leq -\frac{2}{\sum_{k=1}^d \lambda_{\max}(A_k + A_k^T)}.$$

Insertion of the two estimates in

$$\kappa_2(A) = \|A\|_2 \left\|A^{-1}\right\|_2,$$

completes the proof, since the second part of the bound is obvious. $\square$

We conclude this section with a corollary on the simplifications in the Lyapunov case.

Corollary 2.4. *Let $A_0 := A_1 = A_2 = \cdots = A_d \in \mathbb{R}^{n\times n}$ be Hurwitz, then the upper and lower bounds on the 2-norm condition number read*

$$\frac{\max_l \operatorname{Re}\left(-\lambda_l(A_0)\right)}{\min_l |\lambda_l(A_0)|} \leq \kappa_2(A) \leq -\frac{2\sigma_{\max}(A_0)}{\lambda_{\max}(A_0 + A_0^T)}$$

These are exactly the bounds derived in [41] for the special case $d = 2$, the Lyapunov matrix equation.

3 ADI for Tensor Structures

In this section we will finally generalize the ADI iteration to tensor structures. In the next subsection we will briefly explain the generalization for $d = 4$. Therefore we use the 4D heat equation. Afterwards we present the generalization for higher dimensional problems.

3.1 ADI for 4D Heat Equation

Let us recall Example 1.1. There we saw that the 2D heat equation leads to the Lyapunov equation (12),

$$\underbrace{(I \otimes \Delta_{1,h} + \Delta_{1,h} \otimes I)}_{\Delta_{2,h}} X + X(I \otimes \Delta_{1,h} + \Delta_{1,h} \otimes I) = BB^T,$$

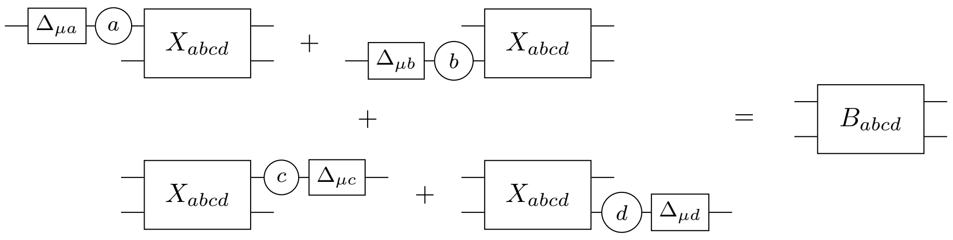

Figure 1: Lyapunov equation.

which is in vectorized form (12):

$$\left(\begin{array}{cccc} & \underbrace{I\otimes I\otimes I\otimes \Delta_{1,h}}_{=H} & + & \underbrace{I\otimes I\otimes \Delta_{1,h}\otimes I}_{=V} \quad + \\ + & \underbrace{I\otimes \Delta_{1,h}\otimes I\otimes I}_{=R} & + & \underbrace{\Delta_{1,h}\otimes I\otimes I\otimes I}_{=Q} \end{array}\right)\underbrace{\mathrm{vec}(X)}_{=u} = \underbrace{\mathrm{vec}(B)}_{=f}.$$

This is also the 4D heat equation, since the structure of this equation is equivalent to

$$\Delta_{4,h}u = f.$$

The tensor formulation of this equation as tensor network is shown in Figure 1, where we use the graphical notation used in [27]. Tensors are depicted by rectangles. The summation over indices is shown by links with small circles giving the index. Similar notations are used in [18]. We search for a 4-mode tensor X, since the right-hand side B is also a 4-mode tensor. We assume that the right hand side B has a tensor train structure with low local rank. On the left-hand side of the equation there are matrix-like multiplications of X with $\Delta_{1,h}$ for each mode.

Generalizing ADI for this structure leads to the following iteration scheme:

$$\begin{aligned}
(H + I\otimes I\otimes I\otimes p_i I)X_{i+\frac{1}{4}} &= (p_i I - V - R - Q)X_i + B\\
(V + I\otimes I\otimes p_i I\otimes I)X_{i+\frac{1}{2}} &= (p_i I - H - R - Q)X_{i+\frac{1}{4}} + B\\
(R + I\otimes p_i I\otimes I\otimes I)X_{i+\frac{3}{4}} &= (p_i I - H - V - Q)X_{i+\frac{1}{2}} + B\\
(Q + p_i I\otimes I\otimes I\otimes I)X_{i+1} &= (p_i I - H - V - R)X_{i+\frac{3}{4}} + B.
\end{aligned}$$

The main advantage of this scheme as compared to the standard ADI, where we have $H = \Delta_{2,h}\otimes I$, is that we only have to solve very simple structured systems of the form

$$\left(I\otimes\cdots\otimes I\otimes \Delta_{1,h}\otimes I\otimes\cdots\otimes I + p_i I_{4n}\right)X = B + \dots .$$

The matrix on the left-hand side is the Kronecker product of identity matrices and one tridiagonal matrix. Solving with $(H + I\otimes I\otimes I\otimes p_i I)$ simplifies to

$$\left(\Delta_{1,h} + p_i I\right)X_{d;abc} = h_{d;abc},$$

where $X_{d;abc}$ is the matricization of the tensor into a fat rectangular matrix, there d is the row index and abc the column index. If X and the right-hand side h are in tensor train representation, this further simplifies to

$$\begin{aligned} X_j &= h_j \quad \forall j \neq 4 \\ (\Delta_{1,h} + p_i I)\, X_4 &= h_4. \end{aligned}$$

The equations for V, R and Q can be handled analogously.

The assumption that X and B are given in tensor train representation is the natural generalization of the assumption in LRCF-ADI that BB^T is a low rank factorization and the solution X can be given in low rank Cholesky factorized form.

3.2 ADI for Multidimensional Equations

Now we will generalize the algorithm from the previous subsection to higher dimensional equations of the form:

$$\begin{aligned} &(I \otimes \cdots \otimes I \otimes \Delta_{1,h} + I \otimes \cdots \otimes I \otimes \Delta_{1,h} \otimes I + \ldots \\ &\qquad \ldots + \Delta_{1,h} \otimes I \otimes \cdots \otimes I)\, \mathrm{vec}(X) = \mathrm{vec}(B). \end{aligned} \tag{20}$$

Like in the 4-mode case, we have to solve d equations in each iteration. Now they have the more general form

$$(S_k + p_i I)\, X_{i+\frac{k}{d}} = \left(p_i I - \sum_{\substack{j=1 \\ j\neq k}}^{d} S_j \right) X_{i+\frac{k-1}{d}} + B \quad \forall k \in \{1, \ldots, d\},$$

with

$$S_k = I \otimes \cdots \otimes I \otimes \Delta_{1,h} \otimes \underbrace{I \otimes \cdots \otimes I}_{k-1 \text{ factors}}.$$

After we have cycled through all dimensions/directions once, we choose the next shift and start again with dimension/direction $k = 1$. Again the right-hand side has the same simple structure as in the 4-mode case.

The algorithm can further be generalized by replacing $\Delta_{1,h}$ with regular matrices $A_i \in \mathbb{R}^{n_i \times n_i}$, so that equations like

$$A\, \mathrm{vec}(X) = \mathrm{vec}(B), \tag{21}$$

where

$$A = I \otimes \cdots \otimes I \otimes A_1 + I \otimes \cdots \otimes I \otimes A_2 \otimes I + \ldots + A_d \otimes I \otimes \cdots \otimes I, \tag{22}$$

can be solved.

Equation (21) is solvable if and only if A is regular. As we have seen in (14), this is for instance the case in all A_k are Hurwitz.

One of the major drawbacks of ADI methods is that one requires good shifts for fast convergence. In the next subsection we present a simple shift strategy.

3.3 ADI Shift Parameters and Convergence of our Method

Computing good shifts is a tough task in the ADI in general. Even for $d = 2$ there is a lack in easy and cheap computable shifts. Hence this topic is not in the focus of this paper. However, we need shifts for the preliminary numerical experiments in the next section.

Let $E^{(i)}$ be the residual in the ith step defined by $E^{(0)} = X^{(0)} - X$. Then the error propagation operator $G_{1:\ell}$ for the first ℓ steps of the ADI iteration is

$$G_{1:\ell} := \prod_{i=1}^{\ell} G_1,$$
$$G_i := \prod_{k=1}^{d} \Big(p_i I - \sum_{\substack{j=1 \\ j\neq k}}^{d} \times_j A_j \Big) \times_k (A_k + p_i I).$$

We have $E^{(\ell)} = G_{1:\ell} E^{(0)}$. In the Lyapunov matrix equation case the norm of G_ℓ for shifts $p_1, \ldots, p_\ell$ is bounded by (see, e.g. [38])

$$\|G_\ell\|_2 \leq \max_{\lambda,\mu \in \Lambda(A)} \left| \prod_{i=1}^{\ell} \frac{(p_i - \mu)(p_i - \lambda)}{(p_i + \lambda)(p_i + \mu)} \right|.$$

It is straight forward to generalizes this for arbitrary d and normal A to

$$\|G_\ell\|_2 \leq \max_{\lambda_k \in \Lambda(A_k),\, k=1,\ldots,d} \left| \prod_{j=1}^{\ell} \prod_{l=1}^{d} \frac{p_j - \sum\limits_{k\neq l} \lambda_k}{p_j + \lambda_l} \right|. \tag{23}$$

The following theorem shows that in the single shift case the ADI iteration converges.

Theorem 3.1. *Let $A_1, \ldots, A_d$ be Hurwitz matrices with $\Lambda(A_k) \subset \mathbb{R}^-$ $\forall k$. Let further p be a shift with*

$$\infty < p \leq |\lambda_i(A)| \leq 0 \quad \forall i = 1, \ldots, N.$$

Then $\|G_1\|_2 < 1$ and so the ADI iteration converges.

Proof. Equation (23) simplifies to

$$\|G_1\|_2 \leq \max_{\substack{\lambda_k \in \Lambda(A_k), \\ k=1,\ldots,d}} \left| \prod_{l=1}^{d} \frac{p - \sum\limits_{k} \lambda_k + \lambda_l}{p + \lambda_l} \right| = \max_{\substack{\lambda_k \in \Lambda(A_k), \\ k=1,\ldots,d}} \left| \prod_{l=1}^{d} \left(1 - \frac{\sum\limits_{k} \lambda_k}{p + \lambda_l} \right) \right|.$$

Since the A_k are Hurwitz, we have $|\lambda_l| < 0$ and $|\lambda_i(A)| < 0$. The shift p is also smaller than zero and so we subtract a positive number from one. Since $p \leq |\lambda_i(A)|$ $\forall i$ the absolute value of the denominator is larger than the absolute value of the numerator and so we get

$$\left| \frac{\sum_k \lambda_k}{p + \lambda_l} \right| < 1.$$

The proof is completed by the argument that $\lim_{l \to \infty} \left\| (G_1)^l \right\| = 0$. □

The shift p can be chosen larger, since the bound is not tight. In the Lyapunov case ($A_k = A_0 \ \forall k$) we have a tight bound for a single shift p if A_0 is a real Hurwitz matrix. The shift p must fulfill

$$p < \frac{d-2}{2} \left| \lambda_{\min} \right|, \qquad \text{with } \lambda_{\min} := \lambda_{\min}(A_0) = \min_i \lambda_i(A_0),$$

since otherwise the quotient $\left| \frac{d\lambda_{\min}(A_0)}{p + \lambda_{\min}(A_0)} \right| \geq 2$ and $\|G\|_2 \geq 1$. One observes that for $d = 2$ this condition reduces to $p < 0$. Thus there is no additional restriction, since p is chosen negative anyway.

More sophisticated shift strategies need the adaption of Penzl's heuristic shifts [32] or the (asymptotically) optimal Wachspress shifts [22, 38] for the standard Lyapunov type matrix equation. Both strategies (for $d = 2$) are based on the rational min max problem:

$$\min_{\{p_1,\ldots,p_\ell\} \subset \mathbb{C}} \quad \max_{\lambda_k \in \Lambda(A_k), k=1,\ldots,d} \quad \left| \prod_{j=0}^{\ell} \frac{p_j - \sum_{k \neq l} \lambda_k}{p_j + \lambda_l} \right|, \tag{24}$$

Penzl's idea for the heuristic with respect to (5) is essentially the restriction of $\{p_1, \ldots, p_\ell\}$ and λ to a subset $\mathcal{R}$ of Λ and the successive evaluation of the rational function, to choose an even smaller subset of $\mathcal{R}$ as the actual shifts. It is obvious that we can do the same for (24). This may become expensive for very large d, though. Due to the notably more complex structure of the rational functions there is no obvious extension of the Wachspress shift strategy for the tensor case. The further investigation of shift strategies is beyond the scope of this paper. Besides the generalization of the shift parameter choice an important ingredient for the efficiency of our method is the generalization of the low rank solution structure corresponding to the low rank form of the right-hand side. In the next subsection we discuss the effects of choosing B to be of tensor train structure, especially we show that then X can be approximated by a tensor train of low local rank, as well.

3.4 ADI for Tensors given as Tensor Trains

If we assume that B is dense and compute a dense solution X, then the algorithm will never be efficient due to the curse of dimensionality. Hence we will now assume that B is given in TT decomposition, with small TT-ranks. Under this assumption Algorithm 1 computes an approximate solution $\tilde{X}$ to X in tensor train form.

This makes of course only sense if the solution can be approximated by a tensor train with low TT-rank. In Theorem 3.3 this will be shown. The argumentation uses the ideas from [12], where it is shown that the solution for a right-hand side of low Kronecker-rank is also of low Kronecker-rank. Therefore we need the following lemma cf. [12, 19], which we proof like in [20, Theorem 2], where the proof is given for the Sylvester equation.

Lemma 3.2. *Let A_j be Hurwitz. The tensor equation*

$$\sum_{j=1}^{d} X \times_j A_j = B$$

has the solution

$$X = -\int_0^\infty B \times_1 \exp(A_1 t) \times_2 \cdots \times_d \exp(A_d t) dt. \tag{25}$$

Proof. We define the function

$$Z(t) = B \times_1 \exp(A_1 t) \times_2 \cdots \times_d \exp(A_d t).$$

Differentiating $Z(t)$ obviously gives [9, Property 2]

$$\dot{Z}(t) = \sum_{j=1}^{d} Z(t) \times_j A_j.$$

Further we have

$$\begin{aligned} Z(\infty) - Z(0) &= \int_0^\infty \dot{Z}(t) dt, \\ 0 - B &= \sum_{j=1}^{d} \underbrace{\int_0^\infty Z(t) dt}_{=-X} \times_j A_j. \end{aligned}$$

□

We assume B to be of low TT-rank that means

$$B(i_1, i_2, \ldots, i_d) = \sum_{\alpha_1, \ldots, \alpha_{d-1}=1}^{r_1, \ldots, r_{d-1}} G_1(i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_2) \cdots G_d(\alpha_{d-1}, i_d), \tag{26}$$

with small $r_1, \ldots, r_{d-1}$. The solution X is then

$$X = -\int_0^\infty \sum_{\alpha_1,\ldots,\alpha_{d-1}=1}^{r_1,\ldots,r_{d-1}} \sum_{\beta_1} G_1(\beta_1,\alpha_1) \left(\exp(A_1 t)\right)_{i_1,\beta_1} \sum_{\beta_2} G_2(\alpha_1,\beta_2,\alpha_2) \left(\exp(A_2 t)\right)_{i_2,\beta_2} \tag{27}$$

$$\cdots \sum_{\beta_{d-1}} G_{d-1}(\alpha_{d-2},\beta_{d-1},\alpha_{d-1}) \left(\exp(A_{d-1} t)\right)_{i_{d-1},\beta_{d-1}} \sum_{\beta_d} G_d(\alpha_{d-1},\beta_d) \left(\exp(A_d t)\right)_{i_d,\beta_d} dt.$$

Further we will need the *Dunford-Cauchy representation* of the matrix exponential [17, Theorem 6.2.28]

$$\exp(tA) = \frac{1}{2\pi\imath} \oint_\Gamma \exp(t\lambda) \left(\lambda I - A\right)^{-1} d_\Gamma \lambda. \tag{28}$$

The following theorem is the generalization of [12, Lemma 6/7] to tensor trains.

Theorem 3.3. *(cf. [12, Lemma 6/7])*
Let $A_1, \ldots, A_d$ be matrices such that the matrix A from (22) has a spectrum $\sigma(A)$ contained in $[-\lambda_{\min}, -\lambda_{\max}] \oplus \imath\,[-\mu, \mu] \subset \mathbb{C}^-$. Let Γ be the boundary of $[-2\lambda_{\min}/\lambda_{\max} - 1, -1] \oplus \imath\,[-\mu-1, \mu+1]$. Let B be a tensor of tensor train structure like in (26). Further let $k \in \mathbb{N}$ and the quadrature points and weights like in

$$\begin{aligned} h_{st} &:= \pi/\sqrt{k}, \\ t_j &:= \log\left(\exp(jh_{st}) + \sqrt{1+\exp(2jh_{st})}\right), \\ w_j &:= h_{st}/\sqrt{1+\exp(-2jh_{st})}. \end{aligned}$$

Then the solution X can be approximated by

$$\tilde{X}\left(i_1, i_2, \ldots, i_d\right) = -\sum_{j=-k}^{k} \frac{2w_j}{\lambda_{\max}} \sum_{\alpha_1,\ldots,\alpha_{d-1}=1}^{r_1,\ldots,r_{d-1}} H_1(i_1,\alpha_1) H_2(\alpha_1,i_2,\alpha_2) \cdots H_d(\alpha_{d-1},i_d), \tag{29}$$

with

$$H_p(\alpha_{p-1}, i_p, \alpha_p) := \sum_{\beta_p} \left(\exp\left(\frac{2t_j}{\lambda_{\max}} A_p\right)\right)_{i_p,\beta_p} G_p(\alpha_{p-1},\beta_p,\alpha_p)$$

with the approximation error

$$\left\| X - \tilde{X} \right\|_2 \leq \frac{C_{st}}{\pi\lambda_{\max}} \exp\left(\frac{2\mu\lambda_{\max}^{-1}+1}{\pi} - \pi\sqrt{k}\right) \oint_\Gamma \left\| \left(\lambda I - 2A/\lambda_{\max}\right)^{-1} \right\|_2 d_\Gamma \lambda \sum_m \left\| b_m \right\|_2,$$

where $B = \sum_m b_m$ is the canonical decomposition of the right-hand side B into rank-1 tensors.

Proof. Let $B = \sum_m b_m$ be the canonical decomposition of the right-hand side tensor train B. Obviously, this sum has not more than $r_1 r_2 \cdots r_{d-1}$ summands. Let y_m be the solution of

$$\sum_{j=1}^{d} y_m \times_j A_j = b_m.$$

Then we have

$$X = \sum_m y_m.$$

From [12, Lemma 5, 6 and 7] it is known that there is an approximation $\tilde{y}_m$ with

$$\|y_m - \tilde{y}_m\|_2 \le \frac{C_{\mathrm{st}}}{\pi \lambda_{\max}} \exp\left(\frac{2\mu\lambda_{\max}^{-1} + 1}{\pi} - \pi\sqrt{k}\right) \oint_{\Gamma} \left\| \left(\lambda I - 2A/\lambda_{\max}\right)^{-1} \right\|_2 d_{\Gamma}\lambda \, \|b_m\|_2 \,.$$

The approximation $\tilde{X} = \sum_m y_m$ can be computed by (29). The approximation error of $\tilde{X}$ follows by summing over m. □

Remark 3.4. *Lemma 3.2 gives an explicit formula for the solution X. This explicit formula can be used to compute the solution X, cf. [12] where this is done for B of low Kronecker rank.*

Remark 3.5. *The TT-ranks of X in Theorem 3.2 depend on k and so on the number of quadrature points. Obviously, the TT-ranks of X are in general large than the TT-ranks of B.*

The proof tells us that for B of low TT-rank, there is an approximation to the solution of low TT-rank and that the error decays exponentially. In the next section we will investigate numerical properties of Algorithm 1, which computes this TT approximation.

If $A_1, \ldots, A_d$ are Hurwitz and the imaginary parts of the eigenvalues of the A_j are bounded by $\frac{\mu}{d}$, then the eigenvalues of A lie in a rectangle as required by the theorem above. Large imaginary parts of the eigenvalues increase the μ and so the factor $\exp\left(2\mu\lambda_{\max}^{-1}/\pi\right)$. Compared with this large real parts of the eigenvalues only increase the length of Γ. However a bad conditioning of the problem leads to higher TT-ranks.

Algorithm 1: ADI for Tensor Trains.

```
Input: $\{A_1, \ldots, A_d\}$, with $\lambda_{\max}(A) \leq 1$, tensor train $B$, accuracy $\epsilon$
Output: tensor train $\tilde{X}$, with $\tilde{X} \approx X$, with $X$ the solution of
        $(I \otimes \cdots \otimes I \otimes A_1 + \ldots + A_d \otimes I \otimes \cdots \otimes I) X = B$
forall the $j \in \{1, \ldots, d\}$ do
    $e_j := \Lambda(A_j)$ ;            /* eigenvalues of small dense matrices */
    $X_j^{(0)} :=$ zeros$(n, 1, 1)$ ;     /* initial tensor train of tt-rank 1 */
end
$r^{(0)} := B$;
forall the $j \in \{1, \ldots, d\}$ do
    $y := X^{(0)}$;
    $y_j := A_j y_j$;                  /* $y_j$ in suitably reshaped form */
    $r^{(0)} := r^{(0)} - y$;                            /* truncated sum */
end
$i := 0$;
while $\|r^{(i)}\| > \epsilon$ do
    $i++$;
    Choose the shift $p_i$;
    forall the $k \in \{1, \ldots, d\}$ do
        $h := B + p_i X^{(i-1+\frac{k-1}{d})}$;
        forall the $j \in \{1, \ldots, d\} \setminus \{k\}$ do
            $y := X^{(i-1+\frac{k-1}{d})}$;
            $y_j := A_j y_j$;          /* $y_j$ in suitably reshaped form */
            $h := h - y$;                                /* truncated sum */
        end
        $X^{(i-1+\frac{k}{d})} := h$;
        $X_k^{(i-1+\frac{k}{d})} := (A_k + p_i) \backslash X_k^{(i-1+\frac{k}{d})}$; /* solve small lin. system */
    end
    $r^{(i)} := B$;
    forall the $j \in \{1, \ldots, d\}$ do
        $y := X^{(i)}$;
        $y_j := A_i y_j$;              /* $y_j$ in suitably reshaped form */
        $r^{(i)} := r^{(i)} - y$;                        /* truncated sum */
    end
end
```

4 Numerical Results

For the numerical computations we use the tensor train toolbox by I.V. Oseledets et al. [25]. We implement Algorithm 1 in MATLAB®. We test the algorithm with the d-dimensional variants of Example 1.1, where $A_j = \Delta_{1,h}\ \forall j$, $B = [0, \cdots, 0, 1]^T$. The relative norm of the residual is computed by

$$\frac{\left\|\Delta_{d,h}X^{(i)} - B\right\|_2}{\left\|\Delta_{d,h}X^{(0)} - B\right\|_2} = \frac{\left\|\Delta_{d,h}X^{(i)} - B\right\|_2}{\left\|B\right\|_2} = \left\|\Delta_{d,h}X^{(i)} - B\right\|_2,$$

since $\left\|B\right\|_2 = 1$.

We observe that the first iterates X have large TT-ranks. The solution X has a low TT-rank property. In the first steps we are far away from the solution, thus we can not expect that our iterates are of low rank. Since the accuracy of the result does not depend on the truncation in the first steps, but on the truncation error in the last steps, it makes sense to allow large truncation errors in the earlier steps, cf. [15]. In practice we observe a slight increase in the number of iterations, but the first iterations become much cheaper, which reduces the overall computation time.

We use the residual for the termination criterion, computing the residual with full accuracy of 10^{-15} in each step. We terminate the iteration once the residual drops below 10^{-9}, even if the last step was not done with the full accuracy. As shifts we choose randomly eigenvalues of A, which can be easily computed via the eigenvalues of $A_j = \Delta_{1,h}$. Since these results may depend on the randomly chosen shifts p, we do five runs and average. The last line represents the solution of a linear system of dimension $10^{500} \times 10^{500}$. We observe that the computation time grows like d^3. The number of sweeps for large d is equal to 5. In each sweep we solve d equations. For each of these equations we have to perform $d-1$ updates for the computation of the right-hand side. Each of these updates has a complexity linear in d, such that the whole algorithm is of cubic complexity in d. We further observe that the DMRG solver from the TT-toolbox is superior to the Tensor ADI, except that the solver was not able to handle the last matrix due to a failed singular value decomposition. The DMRG solver is superior since each DMRG sweep is of linear complexity in d. We observed that in this example for large d 3 to 5 DMRG sweeps are sufficient.

Finally we compare our new algorithm with the existing ones in MATLAB. Therefore we use the MATLAB function `lyap` as well as the package M.E.S.S. [34] for the formulation as matrix equation. The results are shown in Figure 2.

d	$t_{\text{Tensor ADI}}$ in s	$t_{\text{DMRG solver}}$ in s	$t_{\text{TADI}}/t_{\text{DMRG}}$
2	0.6577	9.463045e-10	201.0
INV 2	0.0545	2.920905e-10	
4	16.3887	9.258344e-10	112.0
INV 4	2.5425	5.748835e-09	
5	30.9652	9.709418e-10	79.0
INV 5	3.3336	4.689450e-09	
6	40.9748	9.114350e-10	55.0
INV 6	4.0122	3.217614e-09	
8	36.4464	9.227478e-10	24.0
INV 8	5.3617	3.767449e-09	
10	23.2325	7.281864e-10	13.0
INV 10	5.5565	4.375187e-09	
15	12.9703	2.660179e-10	7.0
INV 15	8.4590	2.758010e-09	
20	26.8764	1.207994e-10	7.0
INV 20	11.0341	5.578970e-09	
25	50.4179	9.267272e-11	7.0
INV 25	9.9951	2.229231e-09	
30	66.5421	8.846127e-10	6.0
INV 30	14.4459	2.241105e-09	
35	130.7518	2.093638e-10	7.0
INV 35	15.1731	2.848897e-09	
40	194.8815	1.178674e-10	7.0
INV 40	16.6687	5.940820e-09	
45	222.1944	5.293162e-10	6.0
INV 45	22.3505	3.585824e-09	
50	302.1157	9.206113e-10	6.0
INV 50	21.0924	2.722696e-09	
55	492.8092	1.265739e-10	7.0
INV 55	24.2897	1.794095e-09	
60	636.1085	9.321389e-11	7.0
INV 60	26.7702	1.231573e-09	
65	647.4493	8.083744e-10	6.0
INV 65	28.5015	8.704940e-10	
70	1001.0012	9.957955e-11	7.0
INV 70	29.8156	1.453762e-09	
75	988.1161	7.879275e-10	6.0
INV 75	30.2188	1.062754e-09	
80	1194.4299	8.107509e-10	6.0
INV 80	64.4583	1.575450e-09	
85	1362.4583	6.781870e-10	6.0
INV 85	72.8823	1.604087e-09	
90	1992.0864	6.514858e-11	7.0
INV 90	66.3502	1.486337e-09	
95	2325.1152	7.656727e-11	7.0
INV 95	81.8164	1.436866e-09	
100	2692.6057	1.239966e-10	7.0
INV 100	91.6408	1.835052e-09	
150	9325.1941	6.099611e-11	7.0
INV 150	135.4045	6.520941e-09	
200	19873.7025	1.125717e-10	7.0
INV 200	58.5566	2.716880e-09	

Table 1: Numerical results, 10 inner discretizations points per direction.

$n = 10$				
d	Tensor-ADI	sparse MESS (Penzl shifts)	lyap	DMRG solver
2	0.3100	0.0240 (0.0028)	0.0005	tba
4	3.1304	0.0109 (0.0492)	0.0124	tba
6	8.1469	0.0758 (0.0937)	7.1645	tba
8	5.4582	5.8634 (1.0972)	13698.2117	tba
10	5.3060	3445.5234 (249.4638)	out of time	tba
$n = 15$				
d	Tensor-ADI	sparse MESS (Penzl shifts)	lyap	DMRG solver
2	0.7999	0.0638 (0.0767)	0.0990	tba
4	8.4896	0.0167 (0.0567)	0.0811	tba
6	13.5902	0.3852 (0.2642)	345.1757	tba
8	7.5854	217.3529 (18.5596)	out of time	tba

Table 2: Computation time in s for our new algorithm and MATLAB functions.

5 Conclusions

We have presented a generalization of the alternating direction implicit iteration to higher dimensions. We computed an approximation of the solution X in tensor train format. The existence of such an approximation was proven by a generalization of a theorem by Grasedyck for right-hand sides and solution of low Kronecker rank to low tensor train rank.

The generalization of results regarding the condition number of matrix equations leads to similar results for the tensor structured equation. In the Lyapunov case we even get exactly the same results, i.e., the conditioning of the problem is dimension independent.

In Theorem 3.1 we prove the convergence of the generalized ADI iteration. We present first ideas for shift strategies, which are tested in a first numerical experiment. The numerical example showed that the ADI iteration is of cubic complexity in d. For large d the method is much cheaper than the matrix equation solvers. However the method is not able to compete with DMRG solvers, which are of linear complexity in d.

However, if a matrix equation has tensor-structured coefficient matrices or coefficient matrices what have a good approximation by a tensor-train matrix, e.g., like in LyaPack `demo_l1` [32], then exploiting this structure by the usage of tensor techniques, by Tensor ADI or DMRG solver, might lead to huge speed-ups.

References

[1] A. Antoulas, D. Sorensen, and Y. Zhou, *On the decay rate of Hankel singular values and related issues*, Sys. Control Lett., 46 (2002), pp. 323–342.

[2] R. Bathia, *A note on the Lyapunov equation*, Linear Algebra Appl., 259 (1997), pp. 71–76.

[3] U. Baur, *Control-Oriented Model Reduction for Parabolic Systems*, Dissertation, Inst. f. Mathematik, TU Berlin, http://opus.kobv.de/tuberlin/volltexte/2008/1760/, Jan. 2008.

[4] ——, *Low rank solution of data-sparse Sylvester equations*, Numer. Lin. Alg. Appl., 15 (2008), pp. 837–851.

[5] U. Baur and P. Benner, *Factorized solution of Lyapunov equations based on hierarchical matrix arithmetic*, Computing, 78 (2006), pp. 211–234.

[6] P. Benner, J. Claver, and E. Quintana-Ortí, *Efficient solution of coupled Lyapunov equations via matrix sign function iteration*, in Proc. 3rd Portuguese Conf. on Automatic Control CONTROLO'98, Coimbra, A. D. et al., ed., 1998, pp. 205–210.

[7] P. Benner, H. Mena, and J. Saak, *On the parameter selection problem in the Newton-ADI iteration for large-scale Riccati equations*, Electr. Trans. Num. Anal., 29 (2008).

[8] R. Byers, *Solving the algebraic Riccati equation with the matrix sign function*, Linear Algebra Appl., 85 (1987), pp. 267–279.

[9] L. De Lathauwer, B. De Moor, and J. Vandewalle, *On the best rank-*1 *and rank-*$(r_1, r_2, \ldots, r_N)$ *approximation of higher-order tensors*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1324–1342.

[10] S. Dolgov and I. V. Oseledets, *Solution of linear systems and matrix inversion in the TT-format*, Preprint 2011-19, Max-Planck-Institut für Mathematik in den Naturwissenschaften, Leipzig, May 2011. Available at `www.mis.mpg.de/preprints/2011/preprint2011_{}19.pdf`.

[11] J. Douglas, *Alternating direction methods for three space variables*, Numer. Math., 4 (1962), pp. 41–63.

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

[13] ——, *Existence of a low rank or H-matrix approximant to the solution of a Sylvester equation*, Numer. Lin. Alg. Appl., 11 (2004), pp. 371–389.

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

[15] W. Hackbusch, B. Khoromskij, and E. E. Tyrtyshnikov, *Approximate iterations for structured matrices*, Numer. Math., 109 (2008), pp. 365–383.

[16] S. Holtz, T. Rohwedder, and R. Schneider, *The alternating linear scheme for tensor optimization in the tensor train format*, SIAM J. Sci. Comput., 34 (2012), pp. 683–713.

[17] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, 1994.

[18] R. Hübener, V. Nebendahl, and W. Dür, *Concatenated tensor network states*, New J. Phys., 12 (2010), p. 025004.

[19] D. Kressner and C. Tobler, *Krylov subspace methods for linear systems with tensor product structure*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1688–1714.

[20] P. Lancaster and M. Tismenetsky, *The Theory of Matrices*, Academic Press, Orlando, 2nd ed., 1985.

[21] R. J. LeVeque, *Finite difference methods for ordinary and partial differential equations*, in Finite difference methods for ordinary and partial differential equations : steady-state and time-dependent problems, SIAM, Philadelphia, PA, 2007.

[22] J.-R. Li and J. White, *Low rank solution of Lyapunov equations*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 260–280.

[23] R. E. Lynch, J. R. Rice, and D. H. Thomas, *Tensor product analysis of partial difference equations*, Bull. Amer. Math. Soc, 70 (1964), pp. 378–384.

[24] I. V. Oseledets, *Approximation of $2^d \times 2^d$ matrices using tensor decomposition*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2130–2145.

[25] ———, *TT-toolbox 2.1.* http://spring.inm.ras.ru/osel/, 2011.

[26] 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.

[27] ———, *Tensor tree decomposition does not need a tree*, Preprint 2009-09, Russian Academy of Sciences - Institute of Numerical Mathematic, Russian Academy of Sciences, Sept. 2009. Available at http://pub.inm.ras.ru/pub/inmras2009-08.pdf.

[28] ———, *TT-cross approximation for multidimensional arrays*, Linear Algebra Appl., 432 (2010), pp. 70–88.

[29] D. Peaceman and H. Rachford, *The numerical solution of elliptic and parabolic differential equations*, J. Soc. Indust. Appl. Math., 3 (1955), pp. 28–41.

[30] T. Penzl, *A cyclic low rank Smith method for large sparse Lyapunov equations*, SIAM J. Sci. Comput., 21 (2000), pp. 1401–1418.

[31] ———, *Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case*, Sys. Control Lett., 40 (2000), pp. 139–144.

[32] ———, Lyapack *Users Guide*, Tech. Rep. SFB393/00-33, Sonderforschungsbereich 393 *Numerische Simulation auf massiv parallelen Rechnern*, TU Chemnitz, 09107 Chemnitz, Germany, 2000. Available from http://www.tu-chemnitz.de/sfb393/sfb00pr.html.

[33] J. Roberts, *Linear model reduction and solution of the algebraic Riccati equation by use of the sign function*, Internat. J. Control, 32 (1980), pp. 677–687. (Reprint of Technical Report No. TR-13, CUED/B-Control, Cambridge University, Engineering Department, 1971).

[34] J. Saak, M. Köhler, P. Kürschner, H. Mena, and P. Benner, *M.E.S.S. matrix equations sparse solvers library 1.0.* http://svncsc.mpi-magdeburg.mpg.de/trac/messtrac/wiki/, 2013. in preparation.

[35] J. Sabino, *Solution of Large-Scale Lyapunov Equations via the Block Modified Smith Method*, PhD thesis, Rice University, Houston, Texas, June 2007. available from: http://www.caam.rice.edu/tech_{}reports/2006/TR06-08.pdf.

[36] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1980. 2nd printing 1983.

[37] E. L. Wachspress, *Iterative solution of the Lyapunov matrix equation*, Appl. Math. Letters, 107 (1988), pp. 87–90.

[38] ———, *The ADI model problem*, 1995. Available from the author.

[39] ———, *ADI iteration parameters for the Sylvester equation*, 2000. Available from the author.

[40] S. R. White, *Density matrix formulation for quantum renormalization groups*, Phys. Rev. Lett., 69 (1992), p. 2863.

[41] Y. Zhou, *Numerical Methods for Large Scale Matrix Equations with Applications in LTI System Model Reduction*, PhD thesis, Rice University, Houston, Texas, May 2002.