Thomas Mach and Jens Saak

Towards an ADI iteration for Tensor Structured Equations

Max Planck Institute Magdeburg Preprints

MPIMD/11-12 **March 15, 2012**

Impressum:

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

www.mpi-magdeburg.mpg.de/preprints

TOWARDS AN ADI ITERATION FOR TENSOR STRUCTURED EQUATIONS*

THOMAS MACH† AND JENS SAAK‡

Abstract. We present a generalization of the alternating directions implicit (ADI) iteration to higher dimensional problems. We solve equations of the form

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

with B given in the tensor train format. The solution X is computed in the tensor train format, too. The accuracy of X depends exponentially on the local rank of X and on the rank of B. To prove this we adapt a result for right hand sides of low Kronecker rank to low tensor train rank. Further we give a convergence proof 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 we present first numerical results.

Key words. alternating directions implicit, ADI, tensor structured equations, tensor trains

AMS subject classifications. 15A24, 15A72, 65F30

1. Introduction. This paper deals with the numerical solution of linear equations with coefficient matrices of tensor product structure. Such tensor structured equations occur, e.g., in the solution of PDEs on d-dimensional hypercubes discretized by finite elements/differences. In [5] 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 Equation (2.7). The solution of tensor structured equations using a Krylov subspace method was recently investigated by Tobler and Kressner in [13]. Further Dolgov and Oseledets investigate the solution of linear systems with a coefficient matrix in tensor-train matrix format in [4].

Here we will present a different approach based on the ADI iteration. 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 1.1.

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

$$A = (I \otimes \Delta_{1,h} + \Delta_{1,h} \otimes I)\,. \tag{1.1}$$

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

$$\underbrace{(I \otimes \Delta_{1,h} + \Delta_{1,h} \otimes I)}_{\Delta_{2,h}} X + X(I \otimes \Delta_{1,h} + \Delta_{1,h} \otimes I) = BB^T, \tag{1.2}$$

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. In Section 2 we will use this special structure to look at them as tensors with 4 modes.

*This work represents equal share of both authors.

†Max Planck Institute for Dynamics of Complex Technical Systems, 39106 Magdeburg, Germany, (thomas.mach@googlemail.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.

1.1. Classic ADI and Lyapunov Equations. Originally the ADI iteration has been developed to solve finite difference discretizations of Poisson's equation [23]:

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

For $d = 1$ using centered differences on an equidistant mesh with nodes $x_i \in \Omega$ and mesh width $h \in \mathbb{R}$ this is well known, see, e.g., [29, 16], to form the linear system of equations $\Delta_{1,h} u = h\, f$, 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

$$\Delta_{2,h} = \underbrace{(\Delta_{1,h} \otimes I)}_{=:H} + \underbrace{(I \otimes \Delta_{1,h})}_{=:V}. \tag{1.3}$$

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

$$\min_{\{p_1,\ldots,p_\ell\} \subset \mathbb{C}} \quad \max_{\lambda \in \Lambda(H), \mu \in \Lambda(V)} \quad \left| \prod_{k=0}^{\ell} \frac{(p_k - \mu)(p_k - \lambda)}{(p_k + \lambda)(p_k + \mu)} \right|. \tag{1.4}$$

In the special case of our example we have $\Lambda(V) = \Lambda(H) = \Lambda(\Delta_{1,h})$ and thus (1.4) simplifies to

$$\min_{\{p_1,\ldots,p_\ell\} \subset \mathbb{C}} \quad \max_{\lambda \in \Lambda(\Delta_{1,h})} \quad \left| \prod_{k=0}^{\ell} \frac{(p_k - \lambda)}{(p_k + \lambda)} \right|. \tag{1.5}$$

Wachspress [30, 31] first observed that the vectorization

$$[(I \otimes F) + (F \otimes I)]\, \mathrm{vec} X = -\mathrm{vec}(GG^T), \tag{1.6}$$

of the Lyapunov equation

$$FX + XF^T = -GG^T \tag{1.7}$$

yields exactly the same structure as (1.3) and thus the Lyapunov equation is an ADI model problem. 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 [31, 32] 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 [25], and Sabino [27] treats a potential theory based selection algorithm. The contribution in [2] combines the optimality of the Wachspress parameters and the fast computability of Penzl's parameters.

1.2. 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 [24, 17] after rewriting it into a one step iteration

$$\begin{aligned}X_i = &-2\operatorname{Re}(p_i)(F + p_i I)^{-1} GG^T (F + p_i I)^{-T}\\ &+ (F + \overline{p_i} I)^{-1}(F - \overline{p_i} I) X_{i-1} (F - p_i I)^T (F + p_i I)^{-T},\end{aligned} \tag{1.8}$$

we find that (1.8) is symmetric. On the other hand the solutions singular values decay rapidly such that it allows for a good low rank approximation [6].

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 [17] one can express the n_p-th, i.e., the final, iterate as

$$\begin{aligned}Z_{n_p} = \Big[&S_{n_p}\sqrt{-2\operatorname{Re}(p_{n_p})} G,\ \ S_{n_p}(T_{n_p} S_{n_p-1})\sqrt{-2\operatorname{Re}(p_{n_p-1})} G,\ \ldots,\\ &S_{n_p} T_{n_p} \cdots S_2 (T_2 S_1)\sqrt{-2\operatorname{Re}(p_1)} G\Big].\end{aligned}$$

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

$$Z_{n_p} = \left[z_{n_p},\ P_{n_p-1} z_{n_p},\ P_{n_p-2}(P_{n_p-1} z_{n_p}),\ \ldots,\ P_1(P_2 \cdots P_{n_p-1} z_{n_p})\right], \tag{1.9}$$

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

$$\begin{aligned}P_i &:= \frac{\sqrt{-2\operatorname{Re}(p_i)}}{\sqrt{-2\operatorname{Re}(p_{i+1})}}(F + p_i I)^{-1}(F - \overline{p_{i+1}} I)\\ &= \frac{\sqrt{-2\operatorname{Re}(p_i)}}{\sqrt{-2\operatorname{Re}(p_{i+1})}}\left[I - (p_i + \overline{p_{i+1}})(F + p_i I)^{-1}\right],\end{aligned} \tag{1.10}$$

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 2 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 some tensor structures available in the literature.

1.3. Tensor Structure. Following [7] we define a *d-mode tensor* $T \in \mathbb{R}^I$ as vector over the product index set

$$I = I_1 \otimes I_2 \otimes \cdots \otimes I_d,$$

where I_i are index sets. If we split I into

$$I = J \otimes K,$$

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

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

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., [3].

We use the following notation of a product of tensor T with matrix $M \in \mathbb{R}^{n_j \times n_j}$ from [3]:

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

Further we use the product of a matrix $M \in \mathbb{R}^{N \times N}$ with $N = \prod_{k=1}^{d} n_k$ 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 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 named *the curse of dimensionality*, cf. [20].

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* [20, 18] 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

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

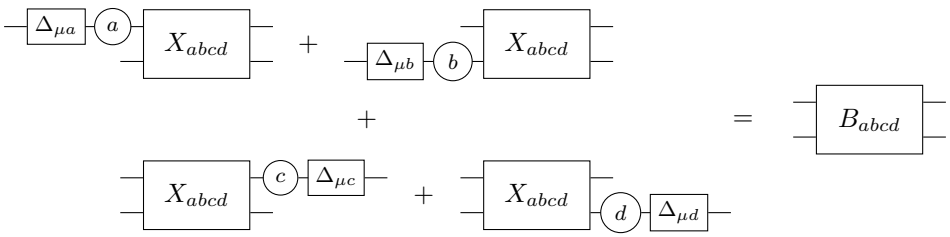

FIG. 2.1. *Lyapunov equation.*

This means that the tensor is represented by $(d-2)$ tensors of size $r_{j-1} \times n_j \times r_j$ and by 2 tensors of size $n_1 \times r_1$ and $r_{d-1} \times n_d$, so that only $(d-2)nr^2 + 2nr$ $(r_j \leq r, n_j \leq n \ \forall j)$ 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 resp. $\operatorname{vec}(AB^T)$.

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 [22].

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,

$$\left(\begin{array}{ccccc} \underbrace{I \otimes I \otimes I \otimes \Delta_{1,h}}_{=H} & + & \underbrace{I \otimes I \otimes \Delta_{1,h} \otimes I}_{=V} & + & \\ & + & \underbrace{I \otimes \Delta_{1,h} \otimes I \otimes I}_{=R} & + & \underbrace{\Delta_{1,h} \otimes I \otimes I \otimes I}_{=Q} \end{array} \right) \underbrace{\operatorname{vec}(X)}_{=u} = \underbrace{\operatorname{vec}(B)}_{=f}. \quad (1.11)$$

In the last part of the next chapter 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

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

2. ADI for Tensor Structures. In this section we will generalize the ADI iteration to tensor structures. We use the 2D heat equation to explain the ideas. Afterwards we present the generalization for higher dimensional problems.

2.1. ADI for 2D Heat Equation. We have seen in Equation (1.11) the vectorization of Equation (1.2), see Example 1.1. The structure of this equation is equivalent to the equation

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

The tensor formulation of this equation as tensor network is shown in Figure 2.1, where we use the graphical notation from [21]. Tensors are depicted by rectangles. The summation over indices is shown by links with small circles giving the index. Similar notations are used in [12]. 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

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

The matrix of 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

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

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 analog to 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.

2.2. ADI for Multidimensional Equations. Now we will generalize the algorithm from the previous subsection to higher dimensional equations of the form:

$$(I \otimes \cdots \otimes I \otimes \Delta_{1,h} + I \otimes \cdots \otimes I \otimes \Delta_{1,h} \otimes I + \ldots + \Delta_{1,h} \otimes I \otimes \cdots \otimes I) \operatorname{vec}(X) = \operatorname{vec}(B). \quad (2.1)$$

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

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

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

$$A \operatorname{vec}(X) = \operatorname{vec}(B), \quad (2.2)$$

where

$$A = I \otimes \cdots \otimes I \otimes A_1 + I \otimes \cdots \otimes I \otimes A_2 \otimes I + \ldots + A_d \otimes I \otimes \cdots \otimes I, \quad (2.3)$$

can be solved.

Equation (2.2) is solvable if and only if A is regular. Therefore the eigenvalues of A have to be non-zero. It is well known as Stéphanos' theorem, see, e.g., [13, 11], that the eigenvalues of A are the sum of the eigenvalues of A_1 to A_d

$$\lambda_i(A) = \lambda_{i_1}(A_1) + \lambda_{i_2}(A_2) + \cdots + \lambda_{i_d}(A_d), \tag{2.4}$$

with $i = i_1 + i_2 n_1 + \cdots + i_d \prod_{j=1}^{d-1} n_j$. The multidimensional case follows from $d = 2$ by recursion.

REMARK 2.1. *In perfect analogy to the Lyapunov/Sylvester equation case, the eigenvalues of A are non-zero if $A_1, \ldots, A_d$ are Hurwitz.* Further in analogy to the two-dimensional case we will call Equation (2.2) a *Sylvester tensor equation* and in the case of $A_1 = A_2 = \cdots = A_d$ a *Lyapunov tensor equation.*

Fast convergence of the ADI iteration can only be expected when using suitable shifts. In the next subsection we present a simple shift strategy.

2.3. ADI Shift Parameters and Convergence of our Method. Computing good shifts is a tough task in the ADI in general. This topic is not in the focus of this paper. Nevertheless, we want to state some remarks.

Let G_ℓ be the error propagation operator for ℓ steps of the ADI iteration. In the Lyapunov matrix equation case the norm of G_ℓ for shifts $p_1, \ldots, p_\ell$ is bounded by (see, e.g. [31])

$$\|G_\ell\|_2 \leq \max_{\lambda,\mu \in \Lambda(A)} \left| \prod_{j=0}^{\ell} \frac{(p_j - \mu)(p_j - \lambda)}{(p_j + \lambda)(p_j + \mu)} \right|.$$

It is straight forward to generalizes this for arbitrary d to

$$\|G_\ell\|_2 \leq \max_{\lambda_k \in \Lambda(A_k),\, k=1,\ldots,d} \left| \prod_{j=0}^{\ell} \prod_{l=0}^{d} \frac{p_j - \sum_{k \neq l} \lambda_k}{p_j + \lambda_l} \right|. \tag{2.5}$$

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

THEOREM 2.2. *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 (2.5) simplifies to

$$\|G_1\|_2 \leq \max_{\substack{\lambda_k \in \Lambda(A_k),\\ k=1,\ldots,d}} \left| \prod_{l=0}^{d} \frac{p - \sum_k \lambda_k + \lambda_l}{p + \lambda_l} \right| = \max_{\substack{\lambda_k \in \Lambda(A_k),\\ k=1,\ldots,d}} \left| \prod_{l=0}^{d} \left(1 - \frac{\sum_k \lambda_k}{p + \lambda_l} \right) \right|.$$

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

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

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

$$p < \frac{d-2}{2} \lambda_{\min},$$

since otherwise the quotient $\frac{d\lambda_{\min}(A_0)}{p+\lambda_{\min}} \geq 2$ and $\|G\|_2 \geq 1$. Note: For $d = 2$ we have $p < 0$ and this is no additional restriction, since p has to be chosen negative anyway.

More sophisticated shift strategies need the adaption of Penzl's heuristic shifts [25] or the (asymptotically) optimal Wachspress shifts [31, 17] for the standard Lyapunov type matrix equation. Both strategies (for $d = 2$) are based on the rational min max problem:

$$\min_{\{p_1,\dots,p_\ell\}\subset\mathbb{C}} \quad \max_{\lambda_k\in\Lambda(A_k),k=1,\dots,d} \quad \left|\prod_{j=0}^{\ell} \frac{p_j - \sum_{k\neq l}\lambda_k}{p_j + \lambda_l}\right|, \tag{2.6}$$

Penzl's idea for the heuristic with respect to (1.4) is essentially the restriction of $\{p_1, \dots, p_\ell\}$ 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 (2.6). 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.

2.4. ADI for Tensors given as Tensor Trains. Now we assume that b is given as TT decomposition, with small TT-ranks, then Algorithm 1 computes an approximate solution X in tensor train form.

Now we will show that the solution X for a right hand side B with low TT-rank is of low TT-rank. This argumentation will use ideas from [5], 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. [5, 13], which we proof like in [14, Theorem 2], where the proof is given for the Sylvester equation.

LEMMA 2.3. *Let A_j be Hurwitz. The tensor equation*

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

has the solution

$$X = -\int_0^\infty B \times_1 \exp(A_1 t) \times_2 \cdots \times_d \exp(A_d t) dt. \tag{2.7}$$

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 [3, Property 2]

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

Further we have

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

Algorithm 1: ADI for Tensor Trains.

Input: $\{A_1, \ldots, A_d\}$, with $\lambda_{\max}(A) \leq 1$, tensor train B, accuracy ϵ
Output: tensor train X, with $(I \otimes \cdots \otimes I \otimes A_1 + \ldots + A_d \otimes I \otimes \cdots \otimes I)\, X = B$

```
forall j ∈ {1,...,d} do
    e_j := Λ(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 j ∈ {1,...,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)‖ > ε do
    i++;
    Choose the shift p_i;
    forall k ∈ {1,...,d} do
        h := B + p_i X^(i−1+(k−1)/d);
        forall j ∈ {1,...,d} \ {k} do
            y := X^(i−1+(k−1)/d);
            y_j := A_j y_j;                  /* y_j in suitably reshaped form */
            h := h − y;                      /* truncated sum */
        end
        X^(i−1+k/d) := h;
        X_k^(i−1+k/d) := (A_k + p_i) \ X_k^(i−1+k/d);   /* solve the small linear system */
    end
    r^(i) := B;
    forall j ∈ {1,...,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
```

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

$$B\,(i_1, i_2, \ldots, i_d) = \sum_{\alpha_1,\ldots,\alpha_{d-1}=1}^{r_1,\ldots,r_{d-1}} G_1(i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_2) \cdots G_d(\alpha_{d-1}, i_d), \tag{2.8}$$

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

$$\begin{aligned} X = -\int_0^\infty \sum_{\alpha_1,\ldots,\alpha_{d-1}=1}^{r_1,\ldots,r_{d-1}} & \sum_{\beta_1} G_1(\beta_1, \alpha_1)\, (\exp(A_1 t))_{i_1,\beta_1} \sum_{\beta_2} G_2(\alpha_1, \beta_2, \alpha_2)\, (\exp(A_2 t))_{i_2,\beta_2} \\ & \cdots \\ & \sum_{\beta_{d-1}} G_{d-1}(\alpha_{d-2}, \beta_{d-1}, \alpha_{d-1})\, (\exp(A_{d-1} t))_{i_{d-1},\beta_{d-1}} \\ & \sum_{\beta_d} G_d(\alpha_{d-1}, \beta_d)\, (\exp(A_d t))_{i_d,\beta_d}\, dt. \end{aligned} \tag{2.9}$$

Further we will need the *Dunford-Cauchy representation* of the matrix exponential [11, Theorem 6.2.28]

$$\exp(tA) = \frac{1}{2\pi\imath} \oint_{\Gamma} \exp(t\lambda) \left(\lambda I - A\right)^{-1} d_{\Gamma}\lambda. \tag{2.10}$$

THEOREM 2.4. *(cf. [5, Lemma 6/7])*
Let $A_1, \dots, A_d$ be matrices such that the matrix A from Equation (2.3) 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 Equation (2.8). 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

$$\tilde{X}\left(i_1, i_2, \dots, i_d\right) = -\sum_{\alpha_1,\dots,\alpha_{d-1}=1}^{r_1,\dots,r_{d-1}} H_1(i_1, \alpha_1) H_2(\alpha_1, i_2, \alpha_2) \cdots H_{d-1}(\alpha_{d-2}, i_{d-1}, \alpha_{d-1}) H_d(\alpha_{d-1}, i_d),$$

with

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

with the approximation error

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

Proof. The following proof is a straight forward application of Lemma 5, 6 and 7 from [5].

The quadrature formula (t_j, w_j) can be found in [28, Example 4.2.11], with $d = \pi/2$ (d here is a parameter from Stenger's quadrature formula and not the dimension), $\alpha = \beta = \epsilon = 1$, $n = N = M = k$. The quadrature formula is used to approximate $\frac{1}{r}$ resp. the inverse of matrices by

$$\frac{1}{r} = \int_0^{\infty} \exp(-tr) dt \approx \sum_{j=-k}^{k} w_j \exp(t_j r).$$

The quadrature error is bounded by [28, (4.2.60)]

$$\left| \int_0^{\infty} \exp(tr) dt - \sum_{j=-k}^{k} w_j \exp(t_j r) \right| \leq C_3 \exp(-\pi\sqrt{k}),$$

with, cf. [8],

$$C_3 \leq C_{\mathrm{st}} \exp(\left|\Im(z)\right|/\pi).$$

In [9, D.4.3] it is shown that $d = \pi/2$ is optimal, since for larger d the quadrature error depends on r exponentially. Together with (2.10) scaled by $\frac{2}{\lambda_{\max}}$ we get the formula for $\tilde{X}$.

For the error holds

$$\left\|X-\tilde{X}\right\|_2 = \left\| \sum_{\alpha_1,\dots,\alpha_{d-1}=1}^{r_1,\dots,r_{d-1}} \prod_{p=1}^{d} \sum_{\beta_p} G_p(\alpha_{p-1},\beta_p,\alpha_p) \left[-\int_0^\infty \exp\left(\frac{2t}{\lambda_{\max}}A_p\right) + \sum_{j=-k}^{k} w_j \exp\left(\frac{2t_j}{\lambda_{\max}}A_p\right)\right]_{i_p,\beta_p} \right\|_2 .$$

The Dunford-Cauchy formula and the quadrature error give

$$\begin{aligned}
&\left\| -\int_0^\infty \exp\left(\frac{2t}{\lambda_{\max}}A_p\right) + \sum_{j=-k}^{k} w_j \exp\left(\frac{2t_j}{\lambda_{\max}}A_p\right)\right\| \\
&\leq \frac{1}{2\pi}\left\| -\oint_\Gamma \int_0^\infty \exp(t\lambda)\left(\lambda I - \frac{2A_p}{\lambda_{\max}}\right)^{-1} d_\Gamma\lambda + \sum_{j=-k}^{k} w_j \oint_\Gamma \exp(t_j\lambda)\left(\lambda I - \frac{2A_p}{\lambda_{\max}}\right)^{-1} d_\Gamma\lambda \right\| \\
&\leq \frac{1}{2\pi} C_{\mathrm{st}} \exp(|\Im(z)|/\pi)\exp(-\pi\sqrt{k}) \oint_\Gamma \left\| \left(\lambda I - \frac{2A_p}{\lambda_{\max}}\right)^{-1}\right\|_2 d_\Gamma\lambda.
\end{aligned}$$

Summing over p completes the proof. □

REMARK 2.5. *Theorem 2.3 gives an explicit formula for the solution X. This explicit formula can be used to compute the solution X, cf. [5] where this is done for B of low Kronecker rank. This approach will be investigated in the future.* 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 $A_1,\dots,A_d$ are Hurwitz and the imaginary parts of the eigenvalues of the A_j's 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 increases 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.

3. Conditioning the Problem. Up to this point we have always tackled the curse of dimensionality to make up a solver capable of solving the large tensor equations. A second and similarly crucial issue when solving increasingly large linear systems is the conditioning of the equation. The study of the condition will therefore be the focus of this section. We will generalize some results that have been derived for Lyapunov equations (1.7) in [33]. The initial idea there is to investigate the vectorized form (1.6) of equations (1.7), which coincides with (2.1) for $d=2$. We will follow the presentation there and show that it is a straight forward argumentation to extend the results to $d>2$.

To this end we recall Equations (2.3) and (2.4) and define $\mathcal{A} := \{A_1, A_2, \dots, A_d\}$. It is then obvious that

$$\min_l |\lambda_l(A)| = \min_{l_1,\dots,l_d} \left| \sum_{k=1}^{d} \lambda_l(A_k) \right| \leq \sum_{k=1}^{d} \min_{l_k} |\lambda_{l_k}(A_k)|. \tag{3.1}$$

Now defining

$$m_k = \operatorname*{argmax}_{j\in\{i=1,\dots,n_k\,:\,\mathrm{Im}\,(\lambda_i(A_k))\geq 0\}} |\lambda_j(A_k)| \tag{3.2}$$

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

$$\max_l |\lambda_l(A)| \geq \left| \sum_{k=1}^{d} \lambda_{m_k}(A_k) \right| \geq \left| \sum_{k=1}^{d} \mathrm{Re}\,(\lambda_{m_k}(A_k)) \right|. \tag{3.3}$$

Where the first inequality holds since we maximize only over a subset of $\{1,\dots,N\}$, where N denotes the dimension of A given as $N = \prod_{k=1}^{d} n_k$. 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

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

and find that then we have

$$\left\|A^{-1}\right\|_2^{-1} = \sigma_{\min}(A) = \min_y \frac{\|Ay\|_2}{\|y\|_2} = \operatorname{sep}(\mathcal{A}). \tag{3.4}$$

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.

LEMMA 3.1. *If all A_i, $i = 1,\dots,d$ are normal, then also A is normal.*

Proof. For $d = 2$ and $A_1 = A_2$ the proof directly follows from a result, e.g., in [15]. We present the calculations for $d = 3$, the extension to large d is obvious.

$$\begin{aligned}
AA^T =& \left(A_3 \otimes I \otimes I + I \otimes A_2 \otimes I + I \otimes I \otimes A_1\right)\left(A_3 \otimes I \otimes I + I \otimes A_2 \otimes I + I \otimes I \otimes A_1\right)^T \\
=& \left(A_3 \otimes I \otimes I + I \otimes A_2 \otimes I + I \otimes I \otimes A_1\right)\left(A_3^T \otimes I \otimes I + I \otimes A_2^T \otimes I + I \otimes I \otimes A_1^T\right) \\
=& \left(A_3A_3^T \otimes II \otimes II\right) + \left(A_3I \otimes IA_2^T \otimes II\right) + \left(A_3I \otimes II \otimes IA_1^T\right) \\
&+ \left(IA_3^T \otimes A_2I \otimes II\right) + \left(II \otimes A_2A_2^T \otimes II\right) + \left(II \otimes A_2I \otimes IA_1^T\right) \\
&+ \left(IA_3^T \otimes II \otimes A_1I\right) + \left(II \otimes IA_2^T \otimes A_1I\right) + \left(II \otimes II \otimes A_1A_1^T\right) \\
=& \left(A_3^TA_3 \otimes II \otimes II\right) + \left(IA_3 \otimes A_2^TI \otimes II\right) + \left(IA_3 \otimes II \otimes A_1^TI\right) \\
&+ \left(A_3^TI \otimes IA_2 \otimes II\right) + \left(II \otimes A_2^TA_2 \otimes II\right) + \left(II \otimes IA_2 \otimes A_1^TI\right) \\
&+ \left(A_3^TI \otimes II \otimes IA_1\right) + \left(II \otimes A_2^TI \otimes IA_1\right) + \left(II \otimes II \otimes A_1^TA_1\right) \\
=& \left(A_3 \otimes I \otimes I + I \otimes A_2 \otimes I + I \otimes I \otimes A_1\right)^T \left(A_3 \otimes I \otimes I + I \otimes A_2 \otimes I + I \otimes I \otimes A_1\right) \\
=& A^TA
\end{aligned}$$

□ 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

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

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 ($k = 1,\dots,d$) is an $\mathcal{O}(d \cdot N)$ operation.

Now allowing the non-normal case, 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 in Remark 2.1 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 3.2. *Assume A_j Hurwitz for all $j = 1, \ldots, d$, then the lower bound*

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

with m_k defined as in (3.2), holds for the 2-norm condition number of A.

Proof. Insert the above bounds on the singular values and the bounds (3.1), (3.3) into

$$\kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}, \tag{3.5}$$

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 non useful terms. Noting that we can express the maximal singular value of A in analogous manner as the minimal one in (3.4) and using (3.5) we end up with

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

The upper bound employs the *logarithmic norms* $\mu(A_j)$ (see [33] 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 3.3. *Assume $A_j \in \mathbb{R}^{n_j \times n_j}$ Hurwitz for all $j = 1, \ldots, d$ and thus $\mu(A_j) < 0$ for all $j = 1, \ldots, d$. Then the 2-norm condition number for A is bounded from above by*

$$\kappa_2(A) \leq -\frac{2\sum_{k=1}^{d} \sigma_{\max}(A_k)}{\sum_{k=1}^{d} \lambda_{\max}(A_k + A_k^T)} \leq -\frac{2 \max_k \sigma_{\max}(A_k)}{\min_k \lambda_{\max}(A_k + A_k^T)}.$$

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

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

For the upper bound on $\left\|A^{-1}\right\|_2$ analogous to [33] we consider the tensor equation

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

From Lemma 2.3 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 [1]. 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. □ We conclude this section with a remark about simplifications in the bounds for the case where all A_j are equal.

COROLLARY 3.4. *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*

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

Note that these are exactly the bounds derived in [33] for the Lyapunov equation.

4. Numerical Results. For the numerical computations we use the tensor train toolbox by I.V. Oseledets [19]. 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

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

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

We observe (see Figure 4.1) 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. [10]. 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. The results are listed in Table 4.1. 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.

Finally we compare our new algorithm with the existing ones in MATLAB. Therefore we use the MATLAB operator backslash for the full system and the MATLAB function `lyap` as well as the package M.E.S.S. [26] for the formulation as matrix equation.

We do not compare our method with the tensor-structured equation solvers from [5] and [13], since such a comparison would only be fair if one used a good shift strategy. The shift strategy is work in progress and we will report such a comparison as soon as possible.

d	t in s	residual	mean(#it)
2	0.3887 e+00	7.015 e−10	112.8
5	5.3975 e+00	7.467 e−10	45.8
8	6.0073 e+00	6.936 e−10	12.8
10	3.6624 e+00	7.685 e−10	6.8
15	7.1693 e+00	3.579 e−10	5.0
20	1.6102 e+01	2.716 e−10	5.0
25	3.1421 e+01	2.437 e−10	5.0
30	5.1158 e+01	3.293 e−10	5.0
40	1.1793 e+02	2.525 e−10	5.0
50	2.2682 e+02	2.049 e−10	5.0
75	7.1918 e+02	4.036 e−10	5.0
100	1.6997 e+03	1.864 e−10	5.0
150	5.5375 e+03	1.801 e−10	5.0
200	1.2795 e+04	1.472 e−10	5.0
250	2.4991 e+04	1.816 e−10	5.0
300	4.2979 e+04	2.535 e−10	5.0
500	1.9515 e+05	2.039 e−10	5.0

TABLE 4.1
Numerical results, 10 inner discretizations points per direction.

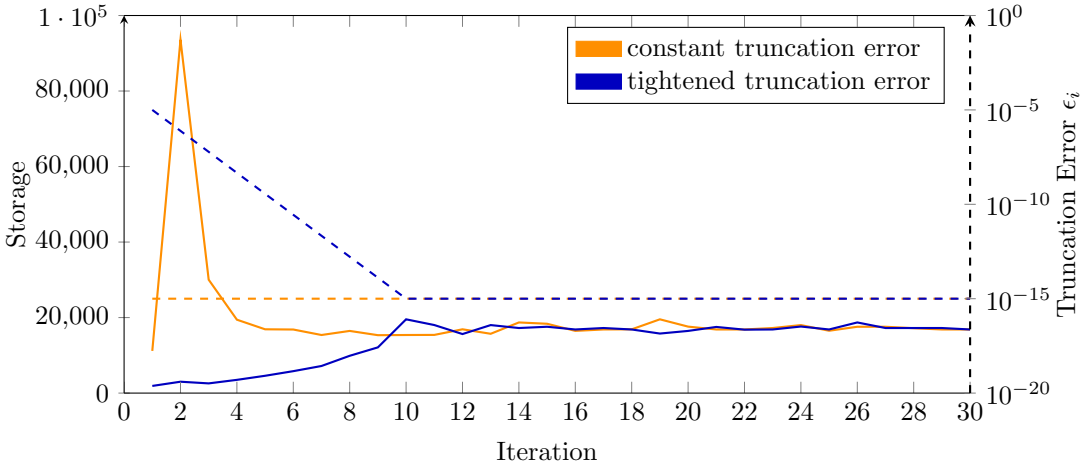

FIG. 4.1. *Memory (left axis, solid line) used for $X^{(i)}$ depending on the truncation error ϵ_i (right axis, dashed line).*

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 L. Grasedyck for right hand sides and solution of low Kronecker rank to low tensor train rank.

In Theorem 2.2 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.

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.

It is of high interest to generalize the method further. First, one may investigate the effects

$n = 10$					
d	Tensor-ADI	sparse backslash	sparse MESS (Penzl shifts)	dense backslash	lyap
2	0.3100	0.0006	0.0240 (0.0028)	0.0003	0.0005
4	3.1304	0.1695	0.0109 (0.0492)	6.3308	0.0124
6	8.1469	out of memory	0.0758 (0.0937)	out of memory	7.1645
8	5.4582	out of memory	5.8634 (1.0972)	out of memory	13698.2117
10	5.3060	out of memory	3445.5234 (249.4638)	out of memory	out of time
$n = 15$					
d	Tensor-ADI	sparse backslash	sparse MESS (Penzl shifts)	dense backslash	lyap
2	0.7999	0.0038	0.0638 (0.0767)	0.0538	0.0990
4	8.4896	4.9191	0.0167 (0.0567)	out of memory	0.0811
6	13.5902	out of memory	0.3852 (0.2642)	out of memory	345.1757
8	7.5854	out of memory	217.3529 (18.5596)	out of memory	out of time

TABLE 4.2
Computation time in s for our new algorithm and MATLAB functions.

of a coefficient matrix expanded by a summand $N \otimes N \otimes \cdots \otimes N$. Such a term would be related to convection in our model problem.

Further, the investigation of more sophisticated shift strategies adapting the ideas of Penzl and Wachspress may lead to a more efficient method.

REFERENCES

[1] R. BATHIA, *A note on the Lyapunov equation*, Linear Algebra Appl., 259 (1997), pp. 71–76.
[2] 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).
[3] 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.
[4] S. DOLGOV AND I. V. OSELEDETS, *Solution of linear systens 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`.
[5] L. GRASEDYCK, *Existence and computation of low Kronecker-rank approximations for large linear systems of tensor product structure*, Computing, 72 (2004), pp. 247–265.
[6] ———, *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.
[7] ———, *Hierarchical singular value decomposition of tensors*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2029–2054.
[8] L. GRASEDYCK, W. HACKBUSCH, AND B. KHOROMSKIJ, *Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices*, Computing, 70 (2003), pp. 121–165.
[9] W. HACKBUSCH, *Hierarchische Matrizen. Algorithmen und Analysis*, Springer-Verlag, Berlin, 2009.
[10] W. HACKBUSCH, B. KHOROMSKIJ, AND E. E. TYRTYSHNIKOV, *Approximate iterations for structured matrices*, Numer. Math., 109 (2008), pp. 365–383.
[11] R. A. HORN AND C. R. JOHNSON, *Topics in Matrix Analysis*, Cambridge University Press, 1994.
[12] R. HÜBENER, V. NEBENDAHL, AND W. DÜR, *Concatenated tensor network states*, New Journal of Physics, 12 (2010), p. 025004.
[13] 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.
[14] P. LANCASTER AND M. TISMENETSKY, *The Theory of Matrices*, Academic Press, Orlando, 2nd ed., 1985.
[15] A. J. LAUB, *A Schur method for solving algebraic Riccati equations*, IEEE Trans. Automat. Control, AC-24 (1979), pp. 913–921.
[16] 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.
[17] J.-R. LI AND J. WHITE, *Low rank solution of Lyapunov equations*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 260–280.
[18] I. V. OSELEDETS, *Approximation of $2^d \times 2^d$ matrices using tensor decomposition*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2130–2145.
[19] ———, *TT-toolbox 2.1.* `http://spring.inm.ras.ru/osel/`, 2011.
[20] 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.
[21] ———, *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`.
[22] ———, *TT-cross approximation for multidimensional arrays*, Linear Algebra Appl., 432 (2010), pp. 70–88.
[23] D. Peaceman and H. Rachford, *The numerical solution of elliptic and parabolic differential equations*, J. Soc. Indust. Appl. Math., 3 (1955), pp. 28–41.
[24] T. Penzl, *A cyclic low rank Smith method for large sparse Lyapunov equations*, SIAM J. Sci. Comput., 21 (2000), pp. 1401–1418.
[25] ———, 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`.
[26] 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/`, 2011. in preparation.
[27] 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`.
[28] F. Stenger, *Numerical methods based on Sinc and analytic functions*, Springer, 1993.
[29] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1980. 2nd printing 1983.
[30] E. Wachspress, *Iterative solution of the Lyapunov matrix equation*, Appl. Math. Letters, 107 (1988), pp. 87–90.
[31] ———, *The ADI model problem*, 1995. Available from the author.
[32] ———, *ADI iteration parameters for the Sylvester equation*, 2000. Available from the author.
[33] Y. Zhou, *Numerical Methods for Large Scale Matrix Equations with Applications in LTI System Model Reduction*, PhD thesis, Rice University, Houston, Texas, May 2002.