Preconditioned low-rank methods for high-dimensional elliptic PDE eigenvalue problems*

Daniel Kressner[1] Christine Tobler[2]

August 12, 2011

Abstract

We consider elliptic PDE eigenvalue problems on a tensorized domain, discretized such that the resulting matrix eigenvalue problem $Ax = \lambda x$ exhibits Kronecker product structure. In particular, we are concerned with the case of high dimensions, where standard approaches to the solution of matrix eigenvalue problems fail due to the exponentially growing degrees of freedom. Recent work shows that this curse of dimensionality can in many cases be addressed by approximating the desired solution vector x in a low-rank tensor format. In this paper, we use the hierarchical Tucker decomposition to develop a low-rank variant of LOBPCG, a classical preconditioned eigenvalue solver. We also show how the ALS and MALS (DMRG) methods known from computational quantum physics can be adapted to the hierarchical Tucker decomposition. Finally, a combination of ALS and MALS with LOBPCG and with our low-rank variant is proposed. A number of numerical experiments indicate that such combinations represent the methods of choice.

1 Introduction

This paper is concerned with the solution of matrix eigenvalue problems arising from the discretization of high-dimensional elliptic PDE eigenvalue problems. A typical example is given by

$$\begin{aligned} -\Delta u(\xi) + V(\xi)u(\xi) &= \lambda\, u(\xi) && \text{in } \Omega \subset \mathbb{R}^d, \\ u(\xi) &= 0 && \text{on } \partial\Omega, \end{aligned} \tag{1}$$

for a certain potential $V : \Omega \to \mathbb{R}$. In the case of a tensorized domain Ω, a standard discretization by, e.g., finite differences leads to a matrix eigenvalue problem of the form

$$(\mathcal{A}_L + \mathcal{A}_V)x = \lambda x, \tag{2}$$

where $\mathcal{A}_L$ and $\mathcal{A}_V$ are the discretized Laplace operator and potential, respectively. If Ω is a tensorized domain, such as $\Omega = [0,1]^d$, and the potential can be written as

$$V(\xi) = \sum_{j=1}^{s} V_j^{(1)}(\xi_1) V_j^{(2)}(\xi_2) \cdots V_j^{(d)}(\xi_d)$$

[1]Chair of Numerical Algorithms and HPC, MATHICSE, EPF Lausanne, CH-1015 Lausanne, Switzerland. daniel.kressner@epfl.ch

[2]Seminar for Applied Mathematics, D-MATH, ETH Zurich, Raemistr. 101, CH-8092 Zurich. ctobler@math.ethz.ch

*Supported by the SNF research module *Preconditioned methods for large-scale model reduction* within the SNF ProDoc *Efficient Numerical Methods for Partial Differential Equations*. DK acknowledges hospitality by the Hausdorff Institute for Mathematics, Bonn, Germany.

then such a discretization results in highly structured matrices:

$$\mathcal{A}_L = \sum_{i=1}^{d} \underbrace{I \otimes \cdots \otimes I}_{d-i \text{ times}} \otimes A_L^{(i)} \otimes \underbrace{I \otimes \cdots \otimes I}_{i-1 \text{ times}}, \tag{3}$$

$$\mathcal{A}_V = \sum_{j=1}^{s} A_{V,j}^{(d)} \otimes \cdots \otimes A_{V,j}^{(2)} \otimes A_{V,j}^{(1)}, \tag{4}$$

where $-A_L^{(i)} \in \mathbb{R}^{n_i \times n_i}$ is the finite-difference discretization of the 1D-Laplace operator and $A_{V,j}^{(i)} \in \mathbb{R}^{n_i \times n_i}$ is a diagonal matrix with the sampled function $V_j^{(i)}(\xi_i)$ on the diagonal. Note that the eigenvector x in (2) has length $n_1 n_2 \cdots n_d$ and can thus be reshaped into a tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$. From this point of view, (2) becomes a linear operator eigenvalue problem on the vector space of tensors.

In general, we aim at computing the smallest eigenvalue for a linear eigenvalue problem of the form

$$\mathcal{A}(\mathcal{X}) = \lambda \mathcal{X}, \tag{5}$$

where we view the matrix $\mathcal{A}$ as a linear operator $\mathcal{A} : \mathbb{R}^{n_1 \times \cdots \times n_d} \to \mathbb{R}^{n_1 \times \cdots \times n_d}$. We assume that $\mathcal{A}$ is symmetric and has a representation compatible with low-rank tensors, such as the sum of Kronecker products from above. With increasing d, the number of entries in $\mathcal{X}$ grows exponentially. This excludes the use of classical iterative methods [1, 6] already for moderate values of d. During the last years, a number of concepts and algorithms have been developed to cope with this curse of dimensionality. The underlying idea is to assume that $\mathcal{X}$ can be well approximated in a low-rank tensor format, with significantly less degrees of freedom, and to construct a solver that searches for such an approximation within the low-rank tensor format. Such methods have been investigated intensively for the solution of eigenvalue problems in computational quantum chemistry, including DMRG for matrix product states and tensor networks, see [30, 22] and the references therein. A number of recent, more mathematically oriented papers illustrate how these ideas can be adapted and transferred to other applications [3, 10, 12, 13, 14, 27]. In particular, Oseledets and Khoromskij [27] discuss the use of LOBPCG to solve reduced eigenvalue problems arising in DMRG (also called MALS). In this paper, we adapt MALS to the hierarchical Tucker decomposition [8, 11] and propose a low-rank variant of LOBPCG [17]. This low-rank variant can either be applied directly to the high-dimensional eigenvalue problem or used in combination with MALS.

The rest of this paper is organized as follows. In Section 2, we briefly summarize the hierarchical Tucker decomposition, which will serve as the low-rank tensor format throughout this paper. Based on this decomposition, Section 3 introduces our low-rank variant of the LOBPCG method. In Section 4, we introduce ALS and MALS in this setting and propose combinations with LOBPCG and with our low-rank variant. Finally, a number of numerical experiments in Section 5 show the effectiveness of such combinations.

2 Preliminaries

In the following, we provide a very brief description of the hierarchical Tucker decomposition (HTD) and refer to [8, 20] for more details.

A tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ has d different *modes* $1, \ldots, d$. Consider a splitting of these modes into two disjoint sets: $\{1, \ldots, d\} = t \cup s$ with $t = \{t_1, \ldots, t_k\}$ and $s = \{s_1, \ldots, s_{d-k}\}$. The *matricization* with respect to this splitting is obtained by merging the first set of modes into row

indices and the second set into column indices:

$$X^{(t)} \in \mathbb{R}^{(n_{t_1}\cdots n_{t_k})\times(n_{s_1}\cdots n_{s_{d-k}})} \quad \text{with} \quad \Big(X^{(t)}\Big)_{(i_{t_1},\ldots,i_{t_k}),(i_{s_1},\ldots,i_{s_{d-k}})} := \mathcal{X}_{i_1,\ldots,i_d}$$

for any indices $i_1, \ldots, i_d$ in the multi-index set $\mathfrak{I} = \{1, \ldots, n_1\} \times \cdots \times \{1, \ldots, n_d\}$.

HTD employs a hierarchy of matricizations, motivated by the following nestedness property:

$$\operatorname{span}\big(X^{(t)}\big) \subset \operatorname{span}\big(X^{(t_r)} \otimes X^{(t_l)}\big)$$

for any $t = t_l \cup t_r$ with two disjoint sets $t_l, t_r \subset \{1, \ldots, d\}$. Given any bases U_t, U_{t_l}, U_{t_r} for the column spaces of $X^{(t)}, X^{(t_l)}, X^{(t_r)}$, this implies the existence of a so called *transfer matrix* B_t such that

$$U_t = (U_{t_r} \otimes U_{t_l})B_t, \quad B_t \in \mathbb{R}^{r_{t_l} r_{t_r} \times r_t}, \tag{6}$$

where r_t, r_{t_l}, r_{t_r} denote the ranks of the corresponding matricizations. Alternatively, B_t can be reshaped into a so called *transfer tensor* $\mathcal{B}_t$ of size $r_{t_l} \times r_{t_r} \times r_t$.

Applying (6) recursively, until t_l and t_r become singletons, leads to the HTD. For example, for $d = 4$, repeated application of (6) leads to the HTD

$$\begin{aligned} \operatorname{vec}(\mathcal{X}) = X^{(1234)} &= (U_{34} \otimes U_{12})B_{1234} \\ U_{12} &= (U_2 \otimes U_1)B_{12} \\ U_{34} &= (U_4 \otimes U_3)B_{34} \\ \Rightarrow \quad \operatorname{vec}(\mathcal{X}) &= (U_4 \otimes U_3 \otimes U_2 \otimes U_1)(B_{34} \otimes B_{12})B_{1234}. \end{aligned} \tag{7}$$

Such a (non-unique) recursive construction of HTD leads to a hierarchical splitting of the modes $1, \ldots, d$, which is represented as a binary tree $\mathcal{T}$, the so called *dimension tree*, as follows: Each node of $\mathcal{T}$ corresponds to a subset of $\{1, \ldots, d\}$, with the root node given by $\{1, \ldots, d\}$ itself. Each parent node is the disjoint union of its two children and each leaf node is a singleton, see also Figure 1. Having prescribed a maximal rank r_t for each node $t \in \mathcal{T}$, the set of hierarchical Tucker tensors of hierarchical rank at most $(r_t)_{t\in\mathcal{T}}$ is defined as

$$\mathcal{H}\text{-Tucker}\big((r_t)_{t\in\mathcal{T}}\big) = \Big\{\mathcal{X} \in \mathbb{R}^{n_1\times\cdots\times n_d} : \operatorname{rank}\big(X^{(t)}\big) \leqslant r_t \text{ for all } t \in \mathcal{T}\Big\}. \tag{8}$$

In [8], algorithms are described to compress a tensor (either given explicitly or in HTD) to a tensor in HTD with prescribed hierarchical ranks. This operation is called *low-rank truncation*.

It is often convenient to illustrate tensor decompositions by means of tensor network diagrams (also called Penrose diagrams), see Figure 1 for an example. Such a diagram represents a tensor in terms of contractions of other smaller-order tensors. Each node in the diagram represents a tensor and each edge represents a mode. An edge connecting two nodes corresponds to the contraction of these tensors in the associated pair of modes.

HTD can also be used for the compression of a matrix $\mathcal{A}$ representing a linear operator $\mathbb{R}^{n_1\times\cdots\times n_d} \to \mathbb{R}^{n_1\times\cdots\times n_d}$. For this purpose, the indices of an entry $a_{i_1,\ldots,i_d;j_1,\ldots,j_d}$ are shuffled such that $\widetilde{a}_{i_1,j_1;\ldots;i_d,j_d} = a_{i_1,\ldots,i_d;j_1,\ldots,j_d}$. After merging pairs of indices i_μ, j_μ, this corresponds to a tensor $\widetilde{\mathcal{A}} \in \mathbb{R}^{n_1^2\times\cdots\times n_d^2}$, to which the HTD can be applied. Having a linear operator represented in low-rank HTD allows the efficient application of this operator to low-rank HTD vectors. The described embedding of a linear operator into a tensor is a common technique in the simulation of quantum spin systems (see, e.g.,[28]) and was also proposed in [26] for the so called TT format.

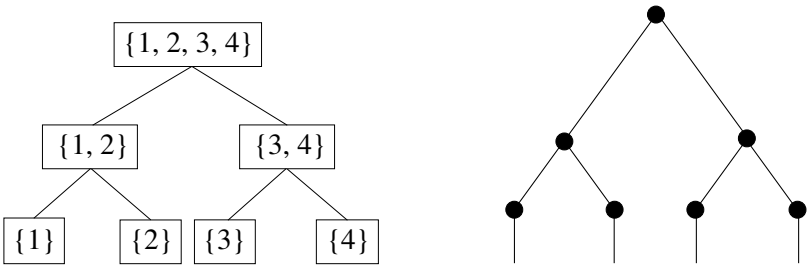

Figure 1: Dimension tree and tensor network diagram for a fourth-order tensor in HTD (7).

Algorithm 1:
LOBPCG with block size 1.
Input: Functions for applying matrices A, B^{-1} to a vector; starting vector x_0 with $\|x_0\|_2 = 1$.
Output: Approximate smallest eigenpair $(\lambda_{\min}, x)$

$\lambda_0 = \langle x_0, x_0 \rangle_A$
$p_0 = 0$
for $k = 0, 1, \ldots$ (until converged) **do**
  $r_k = B^{-1}(Ax_k - \lambda_k x_k)$
  $U = \begin{bmatrix} x_k, & r_k, & p_k \end{bmatrix}$
  $\tilde{A} = U^T A U, \ \widetilde{M} = U^T U$
  Find eigenpair (λ_{k+1}, y), with $\|y\|_2 = 1$, for smallest eigenvalue of matrix pencil $\tilde{A} - \lambda \widetilde{M}$.
  $p_{k+1} = y_2 \cdot r_k + y_3 \cdot p_k$
  $x_{k+1} = y_1 \cdot x_k + p_{k+1}$
  $x_{k+1} \leftarrow x_{k+1} / \|x_{k+1}\|_2$
end for
Return $(\lambda_{\min}, x) = (\lambda_{k+1}, x_{k+1})$.

Algorithm 2:
Tensor LOBPCG with low-rank truncation in HTD.
Input: Functions for applying $\mathcal{A}, \mathcal{B}^{-1}$ to a tensor; function for evaluating $\langle \mathcal{X}, \mathcal{Y} \rangle_{\mathcal{A}}$ of two tensors $\mathcal{X}, \mathcal{Y}$ in HTD; starting vector $\mathcal{X}_0$ in HTD with $\langle \mathcal{X}_0, \mathcal{X}_0 \rangle = 1$.
Output: Approximate smallest eigenpair $(\lambda_{\min}, \mathcal{X})$

$\lambda_0 = \langle \mathcal{X}_0, \mathcal{X}_0 \rangle_{\mathcal{A}}$
$\mathcal{P}_0 = 0 \cdot \mathcal{X}$
for $k = 0, 1, \ldots$ (until converged) **do**
  $\mathcal{R}_k = \mathcal{B}^{-1}(\mathcal{A}(\mathcal{X}_k) - \lambda_k \mathcal{X}_k)$, $\quad \mathcal{R}_k \leftarrow \mathcal{T}(\mathcal{R}_k)$
  $\mathcal{U}_1 = \mathcal{X}_k, \mathcal{U}_2 = \mathcal{R}_k, \mathcal{U}_3 = \mathcal{P}_k$
  $\tilde{A}_{ij} = \langle \mathcal{U}_i, \mathcal{U}_j \rangle_{\mathcal{A}}, \ \widetilde{M}_{ij} = \langle \mathcal{U}_i, \mathcal{U}_j \rangle$
  Find eigenpair (λ_{k+1}, y), with $\|y\|_2 = 1$, for smallest eigenvalue of matrix pencil $\tilde{A} - \lambda \widetilde{M}$.
  $\mathcal{P}_{k+1} = y_2 \cdot \mathcal{R}_k + y_3 \cdot \mathcal{P}_k$ $\quad \mathcal{P}_{k+1} \leftarrow \mathcal{T}(\mathcal{P}_{k+1})$
  $\mathcal{X}_{k+1} = y_1 \cdot \mathcal{X}_k + \mathcal{P}_{k+1}$ $\quad \mathcal{X}_{k+1} \leftarrow \mathcal{T}(\mathcal{X}_{k+1})$
  $\mathcal{X}_{k+1} \leftarrow \mathcal{X}_{k+1} / \sqrt{\langle \mathcal{X}_{k+1}, \mathcal{X}_{k+1} \rangle}$
end for
Return $(\lambda_{\min}, \mathcal{X}) = (\lambda_{k+1}, \mathcal{X}_{k+1})$.

3 Locally optimal block preconditioned CG (LOBPCG)

In principle, any iterative method for solving linear systems or eigenvalue problems can be combined with a low-rank tensor format by representing each iterate in this format, as proposed, e.g., in [2, 10, 16, 19]. This significantly reduces the computational cost for all basic operations (addition, scalar product, ...) of the method and may lead to significant speedup. However, as the rank usually grows rapidly with each iteration, repeated low-rank truncations of the iterates are necessary. When convergence is slow (e.g., due to a poor choice of preconditioner), these truncations may lead to significant perturbations of the iterates especially in the transient phase of the method. These perturbations may or may not severely affect the convergence of the method; the understanding of this phenomenon is still rather incomplete.

In the following, we discuss the combination of HTD with LOBPCG [17], an iterative method for computing the smallest eigenvalue(s) of a symmetric matrix A. In contrast to standard Krylov subspace methods, LOBPCG allows for the direct use of a preconditioner B. Algorithm 1 provides a summary of LOBPCG. For simplicity, this paper focuses on the computation of a single eigenvalue, and we therefore restrict ourselves to block size 1.

We combine LOBPCG with HTD for the solution of a linear symmetric eigenvalue problem $\mathcal{A}(\mathcal{X}) = \lambda \mathcal{X}$ with $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and a given preconditioner $\mathcal{B}$. The algorithmic description given in Algorithm 2 is nearly identical with Algorithm 1, with the notable difference that the iterates are repeatedly truncated to low-rank HTD by means of a truncation operator $\mathcal{T}$. This truncation can be tuned with user-specified options, the prescribed maximal hierarchical rank and maximal truncation error. The evaluation of $\mathcal{A}(\mathcal{X}), \mathcal{B}^{-1}(\mathcal{X})$ typically represents the major part

of the computational cost. The following details are essential for the successful implementation of Algorithm 2:

- To avoid numerical instabilities due to ill-conditioning of the matrix U_k in Algorithm 1, it is often [1, 17] recommended to orthogonalize U_k before calculating (λ_{k+1}, y). However, in the context of low-rank tensors, orthogonalization by, e.g., Gram-Schmidt, is not practical, as the ranks would grow significantly and additional truncation would destroy orthogonality.
- As the application of $\mathcal{A}$ and $\mathcal{B}^{-1}$ usually increases the rank of a tensor significantly, we truncate the tensors $\mathcal{R}_k$ and $\mathcal{P}_k$ before setting up the 3×3 eigenvalue problem. Depending on the nature of $\mathcal{A}$ and $\mathcal{B}^{-1}$, truncation might even be required *during* the application of these operators. While the convergence of LOBPCG is quite tolerant to such truncations, the reduced Gram matrices $\widetilde{A}$ and $\widetilde{M}$ need to be calculated exactly to guarantee the accuracy of the eventually attained eigenpair approximation. In particular, this requires computing the inner products $\langle \mathcal{U}_i, \mathcal{U}_j \rangle_{\mathcal{A}}$ *without* truncation. This turns out to be possible even for high-order tensors in HTD in a number of situations, for example when $\mathcal{A}$ is a short sum of Kronecker products or, more generally, when $\mathcal{A}$ is represented in low-rank HTD [20]. The user is therefore required to provide not only functions for applying $\mathcal{A}, \mathcal{B}^{-1}$ but also a function for evaluating $\mathcal{A}$-inner products with tensors in HTD; an operation that will be available in the `htucker` toolbox [20].

3.1 Numerical experiments

In the following, Algorithm 2 is applied to two quite different examples.

Example 3.1. *Consider the PDE eigenvalue problem (1) with* $\Omega = [0, \pi]^d$ *and potential* $V(\xi) = q \cdot \prod_{i=1}^{d} \sin(\xi_i)$ *for some constant* $q > 0$*. We aim to compute the smallest eigenvalue and its eigenvector.*

Similarly as in the TT format [15, 26], the discretization $\mathcal{A}_L$ *of the Laplace operator can be represented by a tensor in HTD, with all hierarchical ranks equal to* 2 *[20]. The potential* V *is separable and therefore leads to a matrix* $\mathcal{A}_V$ *that can be written as the Kronecker product of* d *matrices. As a preconditioner in LOBPCG, we use the results from [7] to construct an approximate inverse of the* $\mathcal{A}_L$ *having the form*

$$\mathcal{A}_L^{-1} = \int_0^\infty \exp(-t\mathcal{A}_L)\,dt \approx \sum_{j=-M}^{M} \omega_j \exp(-\alpha_j A_L^{(d)}) \otimes \cdots \otimes \exp(-\alpha_j A_L^{(1)}) =: \mathcal{B}^{-1}, \tag{9}$$

for a certain, optimized choice of coefficients $\alpha_j, \omega_j > 0$*. Note that all matrices* $A_L^{(i)}$ *are symmetric positive definite and comparably small. Therefore, the matrix exponentials can evaluated in a stable and efficient manner by calling the* MATLAB *function* `expm`*. In applications with large and possibly nonsymmetric* $A_L^{(i)}$*, different methods need to be used for approximating the matrix exponentials, see [25] for an overview. Alternatively, the tensor Krylov subspace method from [18] could be used for directly approximating the action of* $\mathcal{A}_L^{-1}$*.*

We choose $d = 10$ *and discretize with* $n = 128$ *uniformly spaced nodes in each dimension, using finite differences. The preconditioner (9) uses* $M = 10$*, and the maximally allowed hierarchical rank is set to* 50 *throughout all iterations. Figure 2 illustrates the performance of Algorithm 2 applied to this problem, using truncation with a relative error* `eps` *smaller than* 10^{-2}*,* 10^{-4} *or* 10^{-8}*, respectively. The plots show the residual and maximal rank at every step.*

It can be observed that the convergence depends strongly on the choice of `eps`*. In the case* $q = 1000$*, choosing* `eps` *too large may even lead to stagnation. Since the preconditioner is less effective*

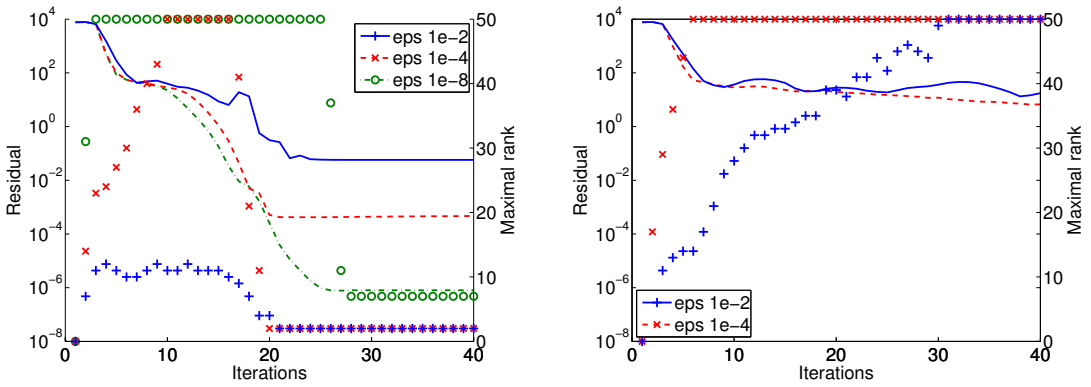

Figure 2: Algorithm 2 (LOBPCG with low-rank truncation) applied to Example 3.1, with $q = 1$ (left plot) and $q = 1000$ (right plot). The lines represent the residual, while the dots represent the maximal rank in each iteration.

for this case, this may also serve as a confirmation that the availability of a good preconditioner is even more critical in our low-rank variant of LOBPCG. Furthermore, the hierarchical ranks tend to grow quite rapidly in the transient phase of the iteration, only decreasing when the asymptotic regime of convergence is reached. ◇

The intermediate rank growth of Algorithm 2 and the observation that overly aggressive truncation potentially leads to stagnation, has motivated and still motivates the search for other methods. However, there are particular settings to which other methods, such as the ones described below, are not applicable. Such a setting arises from an approach to calculate the position and the value of the smallest entry of a given tensor $\mathcal{C}$ in HTD. Following a suggestion in [4, Sec. 6.2.2] for determining the largest entry of a high-order tensor, we define a diagonal matrix $\mathcal{A}$ with the entries of $\mathcal{C}$ on the diagonal. Clearly, the smallest eigenvalue of $\mathcal{A}$ is the smallest entry of $\mathcal{C}$. Also, the corresponding eigenvector will have exactly one nonzero entry at the corresponding position, provided that the smallest eigenvalue is simple. Hence, the desired eigenvector of $\mathcal{A}$ is a tensor with all hierarchical ranks equal to 1.

Example 3.2. *We consider a discretized stationary heat equation with conductivity coefficient depending on p parameters, as described in more detail in [19, Sec. 4]. Sampling the solutions on a tensorized grid of parameter values results in a tensor $\mathcal{Y}$ of order $d = p + 1$. The first mode of this tensor, $\mathcal{Y}(:, i_1, \ldots, i_p)$, contains the temperature distribution in the physical domain for certain parameter values $\alpha_{i_1}^{(1)}, \ldots, \alpha_{i_p}^{(p)}$. The mean temperature across the computational domain is then given by*

$$\mathcal{C}(i_1, \ldots, i_p) = \frac{1}{N} \sum_{j=1}^{N} \mathcal{Y}(j, i_1, \ldots, i_p).$$

It is now of interest to find the parameter values which give minimal mean temperature, corresponding to finding the smallest entry of $\mathcal{C}$. Using the low-rank CG method described in [19], the tensor $\mathcal{C}$ is already in HTD, which we truncate further up to a relative error $\mathsf{eps} = 10^{-4}$.

In our experiments, we have considered the cases $p = 4$ and $p = 9$ with $n = 101$ samples for each parameter. Figure 3 shows the convergence of Algorithm 2 applied to the diagonal embedding $\mathcal{A}$ of $\mathcal{C}$ as described above. For the repeated truncation to HTD, we used a maximal rank of 50 *and a truncation tolerance of* $\mathsf{eps} = 10^{-1}$. *It can be observed that the ranks increase and then*

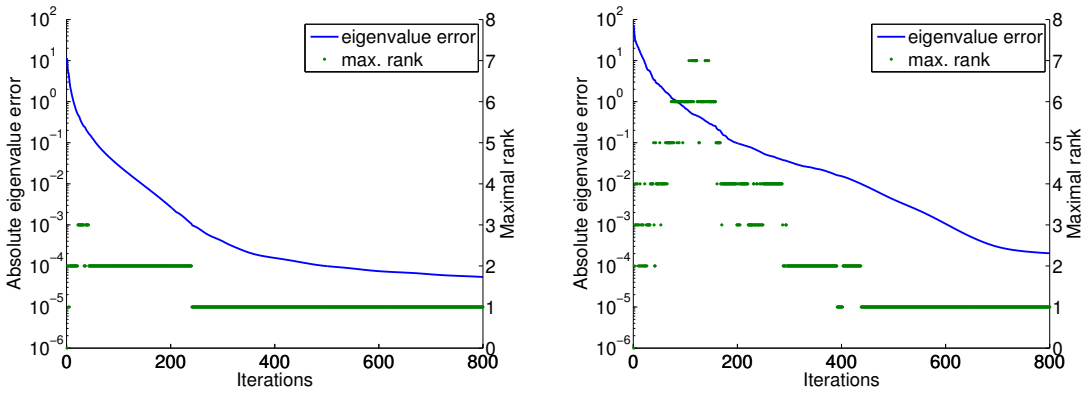

Figure 3: Algorithm 2 applied to determining the smallest entry of a tensor, for a tensor of order $d = 4$ (left plot) and $d = 9$ (right plot).

decrease again, eventually reaching rank 1. *At this point, the iteration should be continued until all matrices U_t at leaf nodes in the dimension tree of $\mathcal{X}_k$ have one clearly dominant entry. The positions of this entries in the leaf matrices give the coordinates of the smallest entry, which can then be used to extract the exact smallest value from $\mathcal{C}$.* ◇

4 Alternating optimization

One of the drawbacks of directly combining LOBPCG (or other iterative methods) with HTD is the intermediate rank growth observed in the numerical examples above. Avoiding this rank growth, a rather different class of methods originates from Computational Quantum Mechanics, in particular the Density Matrix Renormalization Group (DMRG) method. In these methods, the problem at hand is posed as an optimization problem, which is then constrained to the low-rank tensor structure imposed on the solution. ALS (Alternating Linear Scheme) and MALS (Modified ALS) are particular instances of this class, and have been proposed in [12, 3, 14, 27] in combination with the TT low-rank tensor format. In the following, we will discuss the use of ALS and MALS in combination with HTD.

4.1 ALS combined with HTD

Computing the smallest eigenvalue of a symmetric matrix is equivalent to minimizing the Rayleigh quotient $\langle \mathcal{X}, \mathcal{A}(\mathcal{X})\rangle / \langle \mathcal{X}, \mathcal{X}\rangle$. Following the principle outlined above, the Rayleigh quotient is restricted to all tensors in HTD with prescribed hierarchical ranks r_t:

$$\min \left\{ \frac{\langle \mathcal{X}, \mathcal{A}(\mathcal{X})\rangle}{\langle \mathcal{X}, \mathcal{X}\rangle} \,:\, \mathcal{X} \in \mathcal{H}\text{-Tucker}\big((r_t)_{t\in\mathcal{T}}\big),\ \mathcal{X} \neq 0 \right\} \tag{10}$$

where $\mathcal{H}$-Tucker is defined as in (8).

The general principle of ALS for low-rank structures that are represented by tensor networks is to alternately optimize individual nodes of the tensor network. For this purpose, the admissible set of the optimization problem is restricted such that only the tensor associated with the selected node is variable while the tensors associated with all other nodes remain fixed. One sweep of ALS is completed after visiting each node once. For a tensor $\mathcal{X}$ in HTD, every step of ALS

corresponds to the selection of one node in the dimension tree. Formulating the associated restricted optimization problem is highly technical and we will only illustrate the procedure for a non-leaf node t in the following; the formulation is similar for a leaf node.

Each non-leaf node t of a dimension tree corresponds to a transfer tensor $\mathcal{B}_t \in \mathbb{R}^{r_{t_l} \times r_{t_r} \times r_t}$, where t_l and t_r are the left and right children of t, respectively. When selecting such a node in ALS, only this transfer tensor is allowed to vary in the optimization problem (10) while all other transfer tensors and the leaf bases remain fixed. Recalling the nestedness property (6), the matricization $X^{(t)}$ can be decomposed as

$$X^{(t)} = U_t V_t^T = \big(U_{t_r} \otimes U_{t_l}\big) B_t V_t^T,$$

for some matrix V_t, where $B_t \in \mathbb{R}^{r_{t_l} r_{t_r} \times r_t}$ is a matricization of $\mathcal{B}_t$. Vectorizing this relation yields

$$\operatorname{vec}(\mathcal{X}) = \big(\underbrace{V_t \otimes U_{t_r} \otimes U_{t_l}}_{=:\mathcal{U}_t} \big) \operatorname{vec}(B_t), \tag{11}$$

provided that we are in a situation where $t_l = \{1, \dots, i\}$ and $t_l = \{i+1, \dots, j\}$ for some $1 \leqslant i < j \leqslant d$. In other situations, a similar relation holds after permuting the dimensions appropriately. Using (11), the restricted optimization problem takes the form

$$\min \left\{ \frac{y^T \widetilde{\mathcal{A}}_t y}{y^T \widetilde{\mathcal{M}}_t y} \,:\, y \in \mathbb{R}^{r_{t_l} r_{t_r} r_t},\, y \neq 0 \right\}, \tag{12}$$

with the reduced matrices

$$\widetilde{\mathcal{A}}_t := \mathcal{U}_t^T \mathcal{A} \mathcal{U}_t, \quad \widetilde{\mathcal{M}}_t := \mathcal{U}_t^T \mathcal{U}_t. \tag{13}$$

Clearly, the solution of (12) is given by an eigenvector y belonging to the smallest eigenvalue of the matrix pencil $\widetilde{\mathcal{A}}_t - \lambda \widetilde{\mathcal{M}}_t$, provided that this pencil is not singular. In one step of ALS, the transfer tensor $\mathcal{B}_t$ is replaced by the tensor representation of this eigenvector y.

It remains to explain how the reduced matrices in (13) are computed. This can only be performed efficiently if $\mathcal{A}$ can be written as a (short) sum of Kronecker products or if $\mathcal{A}$ is in HTD. In the following, we focus on the first case and restrict ourselves to one term in the sum: $\mathcal{A} = A_d \otimes \cdots \otimes A_2 \otimes A_1$ with $A_i \in \mathbb{R}^{n_i \times n_i}$. Using (11), we obtain

$$\widetilde{\mathcal{A}}_t = \big(V_t \otimes U_{t_r} \otimes U_{t_l}\big)^T \mathcal{A} \big(V_t \otimes U_{t_r} \otimes U_{t_l}\big) = \widehat{A}_t \otimes \widetilde{A}_{t_r} \otimes \widetilde{A}_{t_l},$$

where

$$\widetilde{A}_{t_l} = U_{t_l}^T \Big(\bigotimes_{i \in t_l} A_i \Big) U_{t_l}, \quad \widetilde{A}_{t_r} = U_{t_r}^T \Big(\bigotimes_{i \in t_r} A_i \Big) U_{t_r}, \quad \widehat{A}_t = V_t^T \Big(\bigotimes_{i \notin t} A_i \Big) V_t.$$

These reduced matrices exhibit the same structure as the Gram matrices introduced in [8, P. 2045]. Such Gram matrices are computed by tensor contractions as explained in more detail in [20]. An intuitive way to see this is to note that the inner product $\langle \mathcal{X}, \mathcal{A}(\mathcal{X}) \rangle$ can be expressed as a tensor network, see Figure 4 for an illustration. It follows that $\widetilde{A}_{t_l}, \widetilde{A}_{t_r}, \widehat{A}_t$ are subnetworks, which can be evaluated efficiently by matrix-matrix products. In particular, note that the matrix V_t does not need to be constructed explicitly for the computation of $\widehat{A}_t$. Without going into details, it is important to reuse parts of these computations in subsequent steps of ALS, which can be achieved if the nodes are visited in a certain order. Traversing the dimension tree in a depth-first search (DFS) fashion gives a suitable ordering, see also Figure 5.

The reduced matrix $\widetilde{\mathcal{M}}_t$ in (13) can be calculated in the same way as $\widetilde{\mathcal{A}}_t$, simply by replacing the factors A_i by identity matrices.

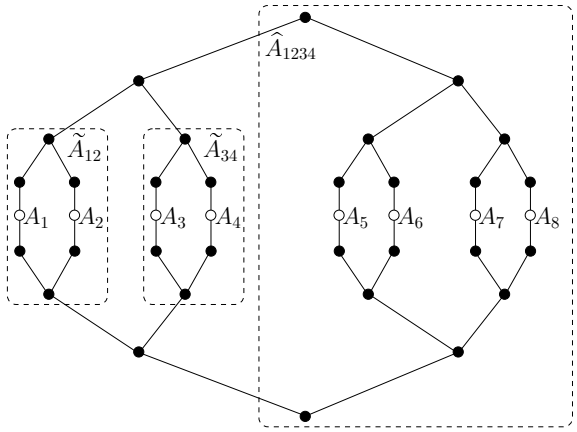

Figure 4: Tensor network corresponding to the inner product $\langle \mathcal{X}, \mathcal{A}(\mathcal{X}) \rangle$ with $\mathcal{X}$ in HTD and $\mathcal{A} = A_d \otimes \cdots \otimes A_1$. The subnetworks corresponding to the Gram matrices $\widetilde{A}_{t_l}, \widetilde{A}_{t_r}, \widehat{A}_t$ are marked by dashed lines.

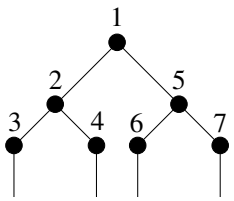

Figure 5: Illustration of the DFS ordering, in which the nodes are traversed in one sweep of ALS, for a tensor of order 4 in HTD.

Remark 4.1. *The computation of $\widetilde{\mathcal{M}}_t$ can be avoided if the columns of $\mathcal{U}$ form an orthonormal basis, which is equivalent to requiring that the columns of V_t, U_{t_r}, U_{t_l} form orthonormal bases. This property also helps to avoid $\mathcal{U}$ becoming nearly rank-deficient, which would lead to numerical instabilities in the eigenvalue computation for the reduced pencil $\widetilde{\mathcal{A}}_t - \lambda \widetilde{\mathcal{M}}_t$. The orthonormality of $\mathcal{U}$ can always be achieved by applying an orthogonalization procedure very similar to [8, Alg. 3]. Analogously to the computation of Gramians, the computational cost of this orthogonalization procedure can be kept small if the nodes are visited in DFS order, see Figure 5.* ◇

4.2 MALS combined with HTD

MALS introduces the following modification to the ALS described above: Instead of a node, an edge of the dimension tree is selected in every step. The nodes connected by this edge are combined to form a tensor of higher order (order 3 if one of the nodes is a leaf, order 4 otherwise). Similarly to ALS, the optimization of the Rayleigh quotient is performed only with respect to this combined tensor. This is, again, equivalent to the solution of a reduced eigenvalue problem (12). Next, an SVD is applied to approximate the matricization of the resulting eigenvector y by a product of two low-rank matrices. These two factors are tensorized and replace the tensors at the nodes connected to the selected edge. More details can be found, e.g., in [12, Pg. 10]. All other aspects of ALS, such as the construction of the reduced Gram matrices and orthogonalization, extend immediately to MALS.

Clearly, one step of MALS is more expensive than one step of ALS, simply because of the enlarged reduced matrices. However, MALS offers a number of distinct advantages: First of all, the low-rank approximations for splitting the combined tensors allow an adaptive choice of the hierarchical ranks in the course of the method. Moreover, since the optimizations are performed on much larger subspaces in every step, MALS can be expected to converge faster and is less likely to get stuck in local minima.

4.3 Combination with LOBPCG

In ALS and more pronouncedly in MALS, the solution of the reduced eigenvalue problem (12) may become very expensive even for moderate hierarchical ranks. If r denotes the maximal hierarchical rank and n the maximal size of the tensor then a direct eigenvalue solver [6] would require $O(n^3r^3)$ operations in the case of a leaf node and $O(r^9)$ operations in the case of a non-leaf node in one step of ALS. For MALS, the number of operations grows to $O(n^3r^6)$ and $O(r^{12})$, respectively. The use of a Krylov subspace method, as implemented in ARPACK (MATLAB's `eigs`), can reduce this cost to a certain extent. However, searching for the smallest eigenvalue usually requires the use of shift-and-invert techniques, which requires the explicit construction and factorization of the possibly dense matrix $\widetilde{\mathcal{A}}_t$.[1] Instead, we will use, as proposed in [27], the LOBPCG method for solving (12).

In MALS, the matricization of the desired eigenvector y can be expected to admit a good low-rank approximation. This can be exploited to reduce the computational cost further, by applying the low-rank LOBPCG proposed in Algorithm 2. Note that a variation of Algorithm 2 needs to be used, replacing the HTD by a low-rank *matrix* format.

It is not advisable to use LOBPCG in all situations, especially when the size p of the reduced eigenvalue problem is small. Following [3], we suggest the following criteria:

$p < 1000$:	use dense symmetric eigenvalue solver (MATLAB's `eig`);
$p < 4000$:	use shift-and-invert Arnoldi (MATLAB's `eigs`);
$p < 50000$:	use standard LOBPCG;
$p \geqslant 50000$:	use low-rank LOBPCG (only within MALS).

Both, standard and low-rank LOBPCG greatly benefit from the use of a good preconditioner. For this purpose, a given preconditioner $\mathcal{P}$ for the original matrix $\mathcal{A}$ can be turned into a preconditioner $\mathcal{U}^T\mathcal{P}\mathcal{U}$ for $\mathcal{U}^T\mathcal{A}\mathcal{U}$. This choice is motivated by the following result: If $\mathcal{A}, \mathcal{P}$ are symmetric positive definite then

$$\kappa\Big((\mathcal{U}^T\mathcal{P}\mathcal{U})^{-1}\,\mathcal{U}^T\mathcal{A}\mathcal{U}\Big) \leqslant \kappa(\mathcal{P}^{-1}\mathcal{A}), \quad \text{for} \quad \mathcal{U}^T\mathcal{U} = I.$$

This follows immediately from an eigenvalue interlacing property for matrix pencils, see, e.g., [21].

4.4 Implementation details

The implementation of ALS and MALS depends on a number of parameter choices. In particular, the stopping criterion for LOBPCG (or any other iterative method) applied to the reduced eigenvalue problem (12) is critical to the execution time of the overall method.

[1] Note that $\widetilde{\mathcal{A}}_t$ is represented as a sum of Kronecker products and this property could be used in an iterative method for solving linear systems with $\widetilde{\mathcal{A}}_t$. However, such an inner iteration would require additional parameter choices and it may not always lead to computational benefits.

Stopping criteria for LOBPCG in ALS. As suggested in [3] for the case of ALS applied to linear systems, we use the following adaptive choice for terminating LOBPCG: Let res be an estimate for the norm of the residual for the eigenvalue/eigenvector approximation to $\mathcal{A}$ obtained after completing the previous sweep of ALS. Then LOBPCG is stopped when the residual for the reduced eigenvalue problem becomes smaller than $\gamma \cdot \mathsf{res}$, where γ is a user-specified parameter ($\gamma = 10^{-2}$ in all numerical examples). Note that the choice of γ may need to be adjusted to the problem to achieve optimal performance. Furthermore, LOBPCG is stopped after a maximum number of iterations (chosen to be 100 in all numerical examples).

Truncation criteria for MALS. Independent of the particular inner solver, the splitting of the combined tensor in every step of MALS requires a choice of tolerance eps for neglecting small singular values in the low-rank approximation. It is important to adjust eps in every sweep of MALS [3]. In the beginning of the iteration, eps should be chosen quite large to avoid an excessive growth of ranks. Once the iteration settles to convergence, eps should be decreased to attain good asymptotic convergence and accuracy. The choice of when to decrease ϵ is quite subtle. Currently, a heuristics is used for this purpose; a theoretically justified automated choice of eps remains open. Apart from eps, the maximally allowed rank is limited by a user-specified value.

Stopping criteria for (low-rank) LOBPCG in MALS. For the low-rank truncation in Algorithm 2 (low-rank LOBPCG), we use the same criteria as explained for low-rank truncation in MALS. In the choice of stopping criteria, it is important to take into account that it will be impossible to attain an arbitrarily small residual, due to the low-rank structure imposed on the eigenvector of the reduced eigenvalue problem. Hence, instead of requiring that the residual is smaller than a certain tolerance, we choose to terminate Algorithm 2 when stagnation occurs. More specifically, the algorithm is stopped when the residual does not decrease within s steps ($s = 10$ in all numerical examples).

When using standard LOBPCG in MALS, we could in principle use the same stopping criterion as in ALS. However, such a choice may lead to an unnecessarily high accuracy compared to the approximation error introduced when splitting the combined tensor. To avoid this waste of LOBPCG iterations, we use the stopping criterion

$$\|x_k - x_{k-s}\| \leqslant \mathsf{eps},$$

with eps and s defined as above.

5 Numerical Experiments

Most of our experiments will be concerned with the PDE eigenvalue problem (1), which was already used in Example 3.1. In all examples, a random starting vector was chosen (with prescribed ranks for ALS, with all hierarchical ranks equal to 2 for MALS). Note that an intelligent choice of starting vector would possibly include components from eigenvectors of the discretized Laplace operator. We have deliberately decided not to do this, in order to reflect the more typical setting when no particularly good choice of starting vector is available.

All numerical experiments have been performed in MATLAB, version 7.7.0.471, on an Intel Xeon DP X5450 with 3 GHz and 2 × 6MB L2 Cache. In the following, we will refer to the following quantities.

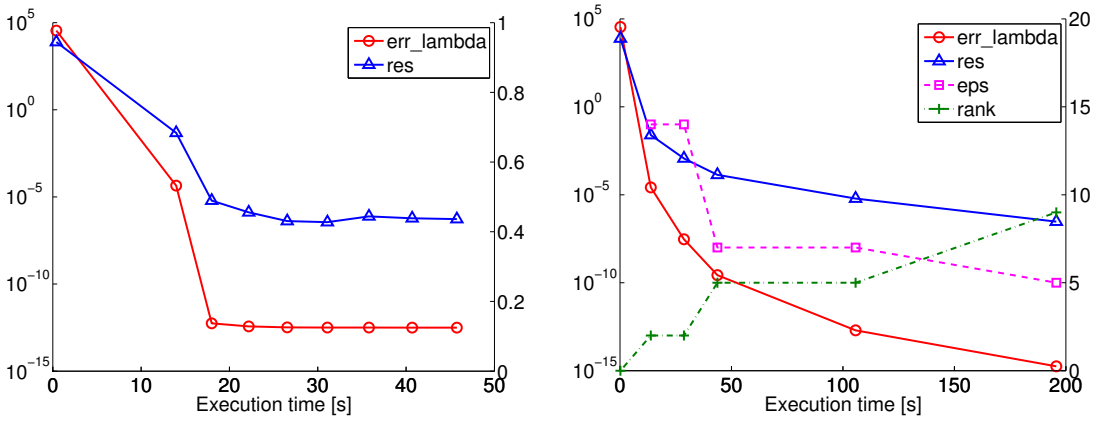

Figure 6: ALS (left plot) and MALS (right plot), applied to Example 3.1 – PDE eigenvalue problem with sine potential and $q = 1$.

err_lambda: Absolute eigenvalue error $|\lambda - \lambda_k|$ after the kth sweep of (M)ALS. As the exact value of λ is not known, an estimate based on the minimum of all computed Rayleigh quotients is used.

res $= \|\mathcal{A}(\mathcal{X}_k) - \lambda_k \mathcal{X}_k\|_2$**:** Residual norm after the kth sweep of (M)ALS.

nr_iter: Maximal number of LOBPCG iterations for solving the reduced eigenvalue problems within one sweep. Note that no number of iterations are provided when MATLAB's `eig` or `eigs` are used for the solution.

eps: Tolerance for low-rank truncations as explained in Section 4.4.

rank: Maximal hierarchical rank in one sweep of MALS.

Example 5.1. *This example is a continuation of Example 3.1, to which we now apply ALS and MALS. In ALS, all hierarchical ranks are set to* 7 *for* $q = 1$ *and to* 40 *for* $q = 1000$*, while MALS is used with a maximal hierarchical rank of* 30*. The choice of* eps *for MALS is varied in the course of the iteration as shown in the plots. If not stated otherwise, we have used the preconditioner* $\mathcal{U}^T \mathcal{P} \mathcal{U}$ *explained in Section 4.3, where* $\mathcal{P}$ *coincides with the Laplace-based preconditioner used in Example 3.1.*

The obtained results are displayed in Figures 6 and 7. For $q = 1$*, both ALS and MALS nearly reach the eventually attained accuracy within only two sweeps. For* $q = 1000$*, the convergence is still very satisfactory even though the quality of the preconditioner is worse. Compared to low-rank LOBPCG applied to the full eigenvalue problem, see Example 3.1, the convergence of (M)ALS is significantly more robust and faster.*

In the following, we will explore some of the choices made in the design of our (M)ALS methods in Section 4. First, Figure 8 shows the convergence for 10 *runs of (M)ALS where in every sweep the nodes of the hierarchical tree are traversed in a fixed but random ordering. This is compared with our choice of traversing the tree in DFS ordering, which was needed to keep the computational cost minimal. Apparently, the choice of ordering has little influence on the overall convergence of ALS and MALS.*

Second, to investigate the impact of using the preconditioner $\mathcal{U}^T \mathcal{P} \mathcal{U}$ *in LOBPCG, we repeat our experiments with no preconditioner, see Figure 9. It turns out that the availability of*

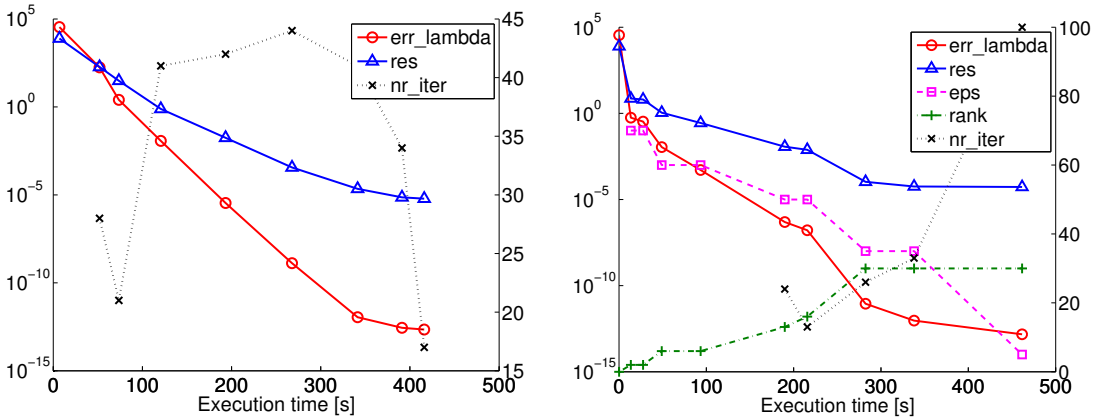

Figure 7: ALS (left plot) and MALS (right plot), applied to Example 3.1 – PDE eigenvalue problem with sine potential and $q = 1000$.

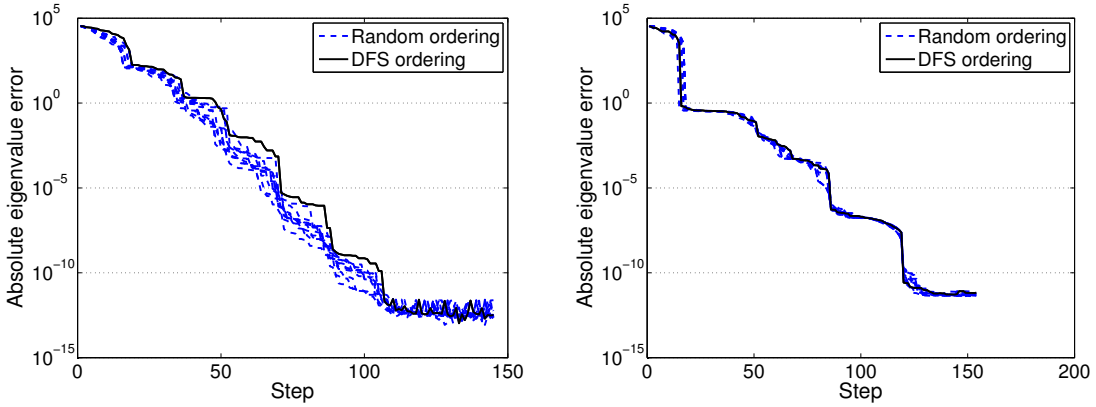

Figure 8: ALS (left plot) and MALS (right plot) with DFS ordering and 10 random orderings applied to Example 3.1 – PDE eigenvalue problem with sine potential and $q = 1000$. In contrast to all other plots, the eigenvalue approximation error is displayed for every step of (M)ALS.

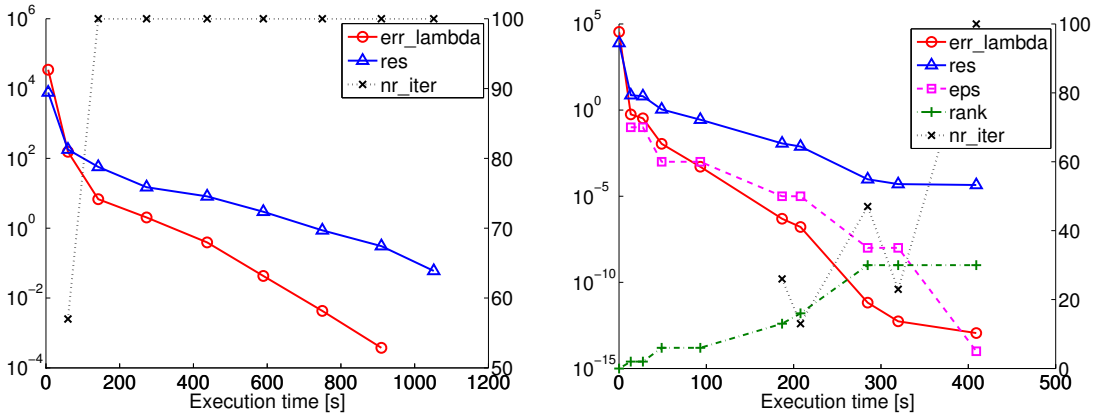

Figure 9: ALS (left plot) and MALS (right plot), applied to Example 3.1 – PDE eigenvalue problem with sine potential and $q = 1000$. In contrast to Figure 7, no preconditioner is used.

a preconditioner is critical for the performance of ALS. Since the maximal number of allowed iterations for LOBPCG is nearly always attained, the obtained approximation to the solution of the reduced eigenvalue problem is poor, resulting in a significantly slower convergence of ALS. In contrast, the availability of a preconditioner seems to have almost no influence on the convergence of MALS. However, it would be misleading to conclude that MALS generally requires no preconditioning. In this particular example, the hierarchical ranks grow quite slowly in the beginning so that `eig(s)` *instead of LOBPCG is used in the critical first stage of MALS sweeps. Only at the* 5*th sweep does the method switch to LOBPCG, but by then the current iterate of MALS already represents a rather good approximation. In such a case, the reduced eigenvalue problems can be expected to feature a much more narrow eigenvalue distribution and demand no preconditioning.* ◇

Example 5.2. *We consider the same discretized PDE eigenvalue problem as in Example 3.1, but with the sine potential replaced by a Henon-Heiles potential as defined in [24, 29, 5]:*

$$V(\xi) = \frac{1}{2}\sum_{j=1}^{n} \sigma_j \xi_j^2 + \sum_{j=1}^{n-1} \Big(\sigma_* (\xi_j \xi_{j+1}^2 - \frac{1}{3}\xi_j^3) + \frac{\sigma_*^2}{16}(\xi_j^2 + \xi_{j+1}^2)^2 \Big).$$

In our experiments we have chosen $\sigma_j = 1$, $\sigma_* = 0.2$, *and the computational domain* $\Omega = [-10, 2]^d$. *The discretized potential* $\mathcal{A}_V$, *see (4), has the structure*

$$\sum_{i=1}^{d} \underbrace{I \otimes \cdots \otimes I}_{d-i \text{ times}} \otimes C_i \otimes \underbrace{I \otimes \cdots \otimes I}_{i-1 \text{ times}} + \sum_{i=1}^{d-1} \underbrace{I \otimes \cdots \otimes I}_{d-i-1 \text{ times}} \otimes B_{i+1} \otimes A_i \otimes \underbrace{I \otimes \cdots \otimes I}_{i-1 \text{ times}}.$$

Any matrix with such a structure can be represented exactly in HTD with rank 4, *see [20, 27], leading to a significantly reduced cost compared to the straightforward Kronecker product presentation. Note that the first sum of the potential can be absorbed into the discretized Laplace operator and accounted for in the construction of the preconditioner. Although we make use of this property in our implementation, we have not observed a dramatic positive effect on the convergence. We choose* $d = 20$ *and discretize with* $n = 128$ *uniformly spaced nodes in each dimension. In the ALS method, we set all hierarchical ranks to* 40, *while a maximal hierarchical rank of* 50 *is used in MALS. Figure 10 contains the obtained results. Interestingly, MALS is* 2 *to*

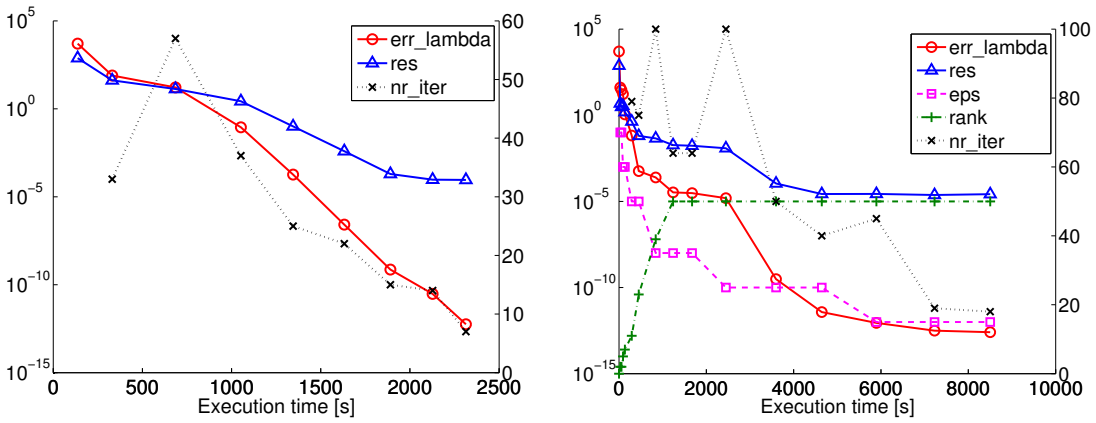

Figure 10: ALS (left plot) and MALS (right plot), applied to Example 5.2 – PDE eigenvalue problem with Henon-Heiles potential.

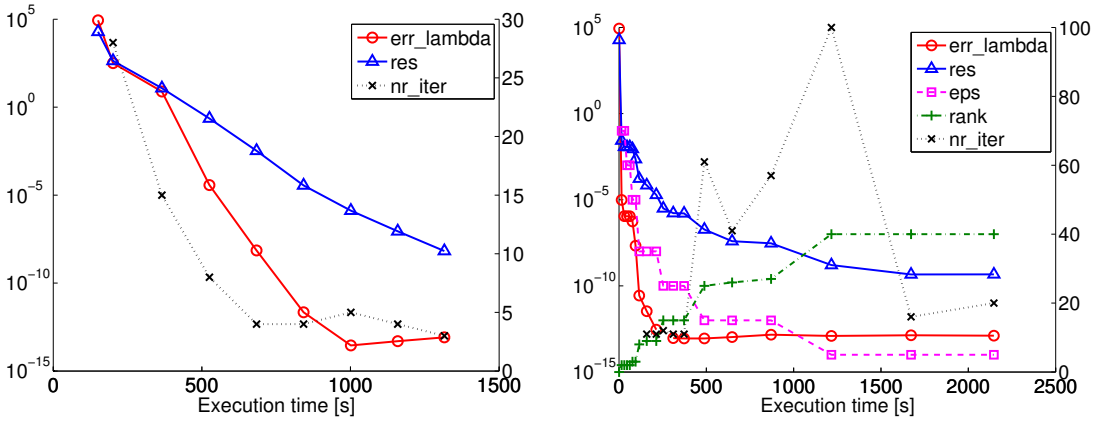

Figure 11: ALS (left plot) and MALS (right plot), applied to Example 5.3 – PDE eigenvalue problem with potential $1/\|\xi\|_2$.

3 *times slower than ALS due to rapid rank growth in the first sweeps until the imposed maximal rank* 50 *is reached.* ◇

Example 5.3. *As a final PDE eigenvalue example, we consider the hydrogen-like potential* $V(\xi) = 1/\|\xi\|_2$ *which features a singularity in the chosen domain* $\Omega = [-1, 1]^d$. *Although the discretization of this potential has full hierarchical rank, there exist highly accurate low-rank approximations. Such approximations can be constructed by means of an exponential sum, see [9]. In our experiments, we use* 10 *terms of this sum, leading to a hierarchical rank of* 10. *We choose* $d = 10$ *and once again discretize with* $n = 128$ *uniformly spaced nodes in each dimension. In the ALS method, we set all hierarchical ranks to* 30, *while a maximal hierarchical rank of* 40 *is used in MALS. As can be seen in Figure 11, MALS converges somewhat faster. Compared to the results for the sine potential with* $d = 10$ *(see Figures 6 and 7), the execution time is significantly higher due to the more complicated nature of the potential.* ◇

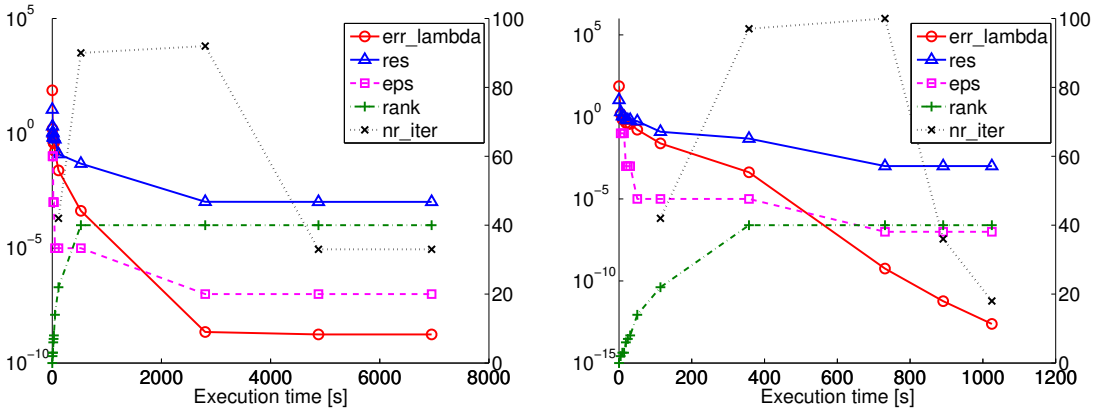

Figure 12: MALS with full LOBPCG (left plot) and with low-rank LOBPCG (right plot), for the spin system from Example 5.4.

Example 5.4. *Our final example is a spin system from [14], with* $d = 64$ *and* $n = 2$*:*

$$\mathcal{A} = \sum_{i=1}^{d} \underbrace{I \otimes \cdots \otimes I}_{d-i \text{ times}} \otimes \sigma_x \otimes \underbrace{I \otimes \cdots \otimes I}_{i-1 \text{ times}} + \sum_{i=1}^{d-1} \underbrace{I \otimes \cdots \otimes I}_{d-i-1 \text{ times}} \otimes \sigma_z \otimes \sigma_z \otimes \underbrace{I \otimes \cdots \otimes I}_{i-1 \text{ times}}.$$

where $\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ *and* $\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ *are Pauli matrices. Note that, as for the Henon-Heiles potential, this matrix has an HTD with hierarchical ranks* 4*. Moreover,* $\mathcal{A}$ *is not positive definite and we search for the smallest eigenvalue, which is negative, and for its eigenvalue. We do not make use of any preconditioner.*

The small sizes $n = 2$ *of the tensor cause significant overhead in HTD, and we therefore reshape the problem such that the desired eigenvector is represented by a tensor of order* $d = 16$ *with* $n = 16$*.*

The convergence of MALS is displayed in Figure 12, using a maximal hierarchical rank of 40*. To demonstrate the significance of employing a low-rank version of LOBPCG for the reduced eigenvalue problems, we deliberately turned off the use of Algorithm 2 in the left plot. In contrast, the right plot uses the strategy proposed in this paper, which implies that Algorithm 2 is used in the last* 4 *sweeps. It is obvious from the execution times that the addition of low-rank LOBPCG results in significant speed-up for larger ranks.* ◇

6 Conclusions

We have presented and compared a number of low-rank tensor techniques for computing the smallest eigenvalue of a symmetric, discretized high-dimensional eigenvalue problem.

It has turned out that the most straightforward approach, combining a classical iterative method with low-rank truncation, exhibits convergence and robustness issues but may be the only choice for certain applications. Although we have only explored the use of LOBPCG, we expect similar findings for methods based on Krylov subspaces, which, however, have the additional complication of not admitting a natural way to incorporate preconditioners.

We have developed a novel combination of (M)ALS with HTD and demonstrated that it offers a very satisfying approach to a number of applications. Moreover, it has been shown that the

use of a low-rank LOBPCG method for solving subproblems in MALS is crucial in applications featuring high ranks.

While it is conceptually not difficult to extend the presented algorithms to the computation of several eigenvalues, at least for LOBPCG, the nonsymmetric case remains open. An interesting approach to nonsymmetric eigenvalue problems has recently been proposed by Meerbergen and Spence in [23]. Currently, this approach is limited to second order tensors (i.e., low-rank matrices) and its extension to high-order tensors remains to be explored.

Acknowledgments: The authors thank Mischa Obrecht, who studied the combination of iterative methods for solving matrix eigenvalue problems with HTD in the context of a semester project.

References

[1] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide.* SIAM, Philadelphia, PA, 2000.

[2] J. Ballani and L. Grasedyck. A projection method to solve linear systems in tensor format. Preprint 46, DFG-Schwerpunktprogramm 1324, May 2010.

[3] S. Dolgov and I. V. Oseledets. Solution of linear systems and matrix inversion in the tt-format. Preprint 19/2011, Max-Planck-Institut für Mathematik in den Naturwissenschaften, May 2011.

[4] M. Espig. *Effiziente Bestapproximation mittels Summen von Elementartensoren in hohen Dimensionen.* PhD thesis, Fakultät für Mathematik und Informatik, Universität Leipzig, 2008.

[5] E. Faou, V. Gradinaru, and C. Lubich. Computing semi-classical quantum dynamics with hagedorn wavepackets. Preprint 29, DFG-Schwerpunktprogramm 1324, November 2009.

[6] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[7] L. Grasedyck. Existence and computation of low Kronecker-rank approximations for large linear systems of tensor product structure. *Computing*, 72(3-4):247–265, 2004.

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

[9] W. Hackbusch. Approximation of $1/x$ by exponential sums. Available from `http://www.mis.mpg.de/scicomp/EXP\_SUM/1\_x/tabelle`. Retrieved August 2008.

[10] W. Hackbusch, B. N. Khoromskij, S. A. Sauter, and E. E. Tyrtyshnikov. Use of tensor formats in elliptic eigenvalue problems. Preprint 78/2008, Max-Planck-Institut für Mathematik in den Naturwissenschaften, 2008.

[11] W. Hackbusch and S. Kühn. A new scheme for the tensor representation. *J. Fourier Anal. Appl.*, 15(5):706–722, 2009.

[12] S. Holtz, T. Rohwedder, and R. Schneider. The alternating linear scheme for tensor optimisation in the TT format. Preprint 71, DFG-Schwerpunktprogramm 1324, December 2010.

[13] S. Holtz, T. Rohwedder, and R. Schneider. On manifolds of tensors of fixed tt-rank. Preprint 61, DFG-Schwerpunktprogramm 1324, September 2010.

[14] T. Huckle, K. Waldherr, and T. Schulte-Herbrüggen. Computations in quantum tensor networks. Technical report, Institut für Informatik, TU München, 2010.

[15] V. A. Kazeev and B. N. Khoromskij. On explicit qtt representation of laplace operator and its inverse. Preprint 75/2010, Max-Planck-Institut für Mathematik in den Naturwissenschaften, 2010.

[16] B. N. Khoromskij and C. Schwab. Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs. *SIAM J. Sci. Comput.*, 33(1):364–385, 2011.

[17] A. V. Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput.*, 23(2):517–541, 2001.

[18] D. Kressner and C. Tobler. Krylov subspace methods for linear systems with tensor product structure. *SIAM J. Matrix Anal. Appl.*, 31(4):1688–1714, 2010.

[19] D. Kressner and C. Tobler. Low-rank tensor Krylov subspace methods for parametrized linear systems. Technical report 2010-16, Seminar for Applied Mathematics, ETH Zurich, June 2010.

[20] D. Kressner and C. Tobler. `htucker` – a MATLAB toolbox for tensors in hierarchical Tucker format. Technical report, Seminar for Applied Mathematics, ETH Zurich, 2011. In preparation, see `http://www.sam.math.ethz.ch/NLAgroup/htucker\_toolbox.html`.

[21] R.-C. Li, Y. Nakatsukasa, N. Truhar, and S. Xu. Perturbation of partitioned hermitian definite generalized eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 32(2):642–663, 2011.

[22] K. H. Marti, B. Bauer, M. Reiher, M. Troyer, and F. Verstraete. Complete-graph tensor network states: a new fermionic wave function ansatz for molecules. *New Journal of Physics*, 12(10):103008, 2010.

[23] K. Meerbergen and A. Spence. Shift-and-invert iteration for purely imaginary eigenvalues with application to the detection of Hopf bifurcations in large scale problems. *SIAM J. Matrix Anal. Appl.*, 31(4):1982–1999, 2010.

[24] H.-D. Meyer, U. Manthe, and L.-S. Cederbaum. The multiconfigurational time-dependent Hartree approach. *Chem. Phys. Lett.*, 165:73–78, 1990.

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

[26] I. V. Oseledets. Compact matrix form of the d-dimensional tensor decomposition. Preprint 09-01, Institute of Numerical Mathematics RAS, Moscow, Russia, 2009.

[27] I. V. Oseledets and B. N. Khoromskij. DMRG+QTT approach to high-dimensional quantum molecular dynamics. Preprint 69/2010, Max-Planck-Institut für Mathematik in den Naturwissenschaften, 2010.

[28] B. Pirvu, V. Murg, J. I. Cirac, and F. Verstraete. Matrix product operator representations. *New Journal of Physics*, 12(2):025012, 2010.

[29] A. Raab and H.-D. Meyer. A numerical study on the performance of the multiconfigurational time-dependent Hartree method for density operators. *J. Chem. Phys.*, 112:10718–10729, 2000.

[30] U. Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326, 2011.