On the Approximation of Functionals of Very Large Hermitian Matrices represented as Matrix Product Operators

Moritz August
Department of Informatics, Technical University of Munich, 85748 Garching, Germany (august@in.tum.de)

Mari Carmen Bañuls
Max Planck Institute for Quantum Optics, 85748 Garching, Germany (mari.banuls@mpq.mpg.de)

Thomas Huckle
Department of Informatics, Technical University of Munich, 85748 Garching, Germany (huckle@in.tum.de)

We present a method to approximate functionals $\mathsf{Tr}f(A)$ of very high-dimensional hermitian matrices A represented as Matrix Product Operators (MPOs). Our method is based on a reformulation of a block Lanczos algorithm in tensor network format. We state main properties of the method and show how to adapt the basic Lanczos algorithm to the tensor network formalism to allow for high-dimensional computations. Additionally, we give an analysis of the complexity of our method and provide numerical evidence that it yields good approximations of the entropy of density matrices represented by MPOs while being robust against truncations.

I. INTRODUCTION

Approximating functionals of very large matrices is an important problem in many fields of science, such as network analysis [BFRR14, FMRR13, EH10, New10] or quantum mechanics [Sch11, VMC08]. In many cases, the respective matrices are hermitian due to either the underlying physical properties of the systems they describe or the way they are constructed from, e. g., a graph. Naturally, as the dimensionality of the matrices becomes very high, i. e., several tens or hundreds of thousands and above, explicit methods of function evaluation, like exact diagonalization, break down and approximations must be made.

One paradigm for the approximation of high-dimensional matrices that has gained a lot of attention especially in the quantum information, condensed matter and numerical linear algebra communities are tensor network representations [VMC08, Sch11, GKT13, BSU16, Ose11, HWSH13]. Among the class of tensor networks, matrix product states (MPS) and matrix product operators (MPO) count among the best established methods. These representations approximate large tensors by contractions of multiple low-rank tensors in a row and have been shown to yield efficient parametrizations of many relevant states of quantum many-body systems [VC06, PGVWC06, Has07].

In this work, we introduce a novel method to approximate functionals of the form $\mathsf{Tr}f(A)$ where we assume f to be smooth and defined on the spectrum of A as well as A to be hermitian and given in MPO-format. For our method, we have reformulated a block version of the Lanczos algorithm in MPO/MPS-representation. This particular block Lanczos algorithm will be referred to as global Lanczos algorithm in the following and has already been used to approximate functionals of the given form for explicitly stored matrices [BRRS15, EMS09]. Rewriting it for the tensor network formalism however allows us to consider block vectors of size identical to A which was previously prohibitive. Our method is thus able to approximate $\mathsf{Tr}f(A)$ for certain $f(A)$ needing only one carefully selected starting block vector. This means that we get rid of the approximation error induced by having to combine the results obtained for multiple different starting block vectors. At the same time, we find the numerical error induced by the MPS/MPO-representation to be comparably small. Our method can be applied whenever A is efficiently approximated by an MPO. We will in the following however focus on the case where A has directly been defined as MPO.

The rest of this work is structured as follows: after a basic introduction to matrix product states and operators in Section II, we will introduce the block Lanczos method we employ in this work in Section III. In Section IV, we will then provide an explanation of how one can use the method to approximate functionals of the given type. Following this, we will then state our method in Section V, show how we have reformulated the global Lanczos method in the tensor network formalism and discuss its properties as well as give an analysis of its complexity. Finally, in Section VI we will provide numerical evidence for the fast and robust convergence of our method for the case of the trace-norm and von-Neumann entropy of quantum mechanical density matrices. We conclude with a discussion of the results in Section VII.

II. MATRIX PRODUCT STATES AND OPERATORS

In the area of tensor networks, MPS and MPOs form a well-established class of tensor decompositions that allow for efficient and stable approximation of very high-dimensional vectors and matrices respectively. While they are commonly used in theoretical and numerical quantum many body physics to model, e.g., ground and

thermal states [VPC04, ZV04, PMCV10, FNW92, Vid03, VGRC04, Sch11, VMC08, VC06], they also have been independently introduced in the numerical mathematics community under the name of Tensor Trains (TT) [Ose11] as a general approximation tool. Since our work is mainly motivated by applications in quantum physics, we will adapt the respective terminology in the following.

A matrix product state is a decomposition of a vector $v \in \mathbb{C}^N$ such that

$$v_i = v_{i_1 \dots i_L} = \mathsf{Tr} A_1^{i_1} A_2^{i_2} \cdots A_L^{i_L}$$

where the index i is split up into L sub-indices of dimensionality d, called physical indices, $i_1, \dots, i_L$. $A_1, \dots, A_L \in \mathbb{C}^{d \times D \times D}$ we will refer to as the sites of the MPS and D is called the bond dimension. This concept of splitting up the index i is known under the name of quantized tensor trains (QTT) [Kho11] in the numerical community. While in principle every site may have its own bond dimensions, as long as they allow for contraction with neighbouring sites, for simplicity and without loss of generality we will assume all sites to have identical bond dimension D. The physical dimension d is likewise assumed to be identical for all sites. It is important to note that $N = L^d$, as this relation forms the basis for the ability of MPS/MPOs to represent vectors and matrices of very high dimensionality.

A slightly different representation can be chosen, where

$$v_i = v_{i_1 \dots i_L} = A_1^{i_1} A_2^{i_2} \cdots A_L^{i_L}$$

with $A_1 \in \mathbb{C}^{d \times 1 \times D}$ and $A_L \in \mathbb{C}^{d \times D \times 1}$. In physical terms, the former representation corresponds to systems with closed boundary-conditions (CBC) whereas the latter assumes open boundaries (OBC). It is clear that OBC is just a special case of PBC. For the remainder of this work, we assume open boundaries without loss of generality. Following this decomposition, a particular element of v is described by a chain of matrix multiplications, explaining the name of the representation.

Now, the whole vector v can be written as

$$\begin{aligned}
v &= \sum_{i_1,\dots,i_L}^{d} \left(A_1^{i_1} A_2^{i_2} \cdots A_L^{i_L}\right) \left(e_{i_1} \otimes e_{i_2} \otimes \cdots \otimes e_{i_L}\right) \\
&= \sum_{i_1,\dots,i_L}^{d} \sum_{k_2,\dots,k_{L-1}}^{D} \left(A_{1,1k_2}^{i_1} A_{2,k_2k_3}^{i_2} \cdots A_{L,k_{L-1}1}^{i_L}\right) \\
&\qquad \cdot \left(e_{i_1} \otimes e_{i_2} \otimes \cdots \otimes e_{i_L}\right) \\
&= \sum_{k_2,\dots,k_{L-1}}^{D} \left(\sum_{i_1}^{d} A_{1,1k_2}^{i_1} e_{i_1}\right) \otimes \left(\sum_{i_2}^{d} A_{2,k_2k_3}^{i_2} e_{i_2}^{i_2}\right) \\
&\qquad \otimes \cdots \otimes \left(\sum_{i_L}^{d} A_{L,k_{L-1}1}^{i_L} e_{i_L}\right) \\
&= \sum_{k_2,\dots,k_{L-1}}^{D} u_{1,k_2} \otimes u_{2,k_2k_3} \otimes \cdots \otimes u_{L,k_{L-1}}
\end{aligned}$$

where e_j denotes the jth column of the identity matrix. This expression sheds some light on the underlying tensor product structure of MPS and facilitates comparisons with other tensor decomposition schemes.

We now turn our attention to the representation of operators and matrices respectively. Abstractly, one can define an MPO as an operator with an MPS representation in a direct product basis of the operator linear space. More concretely, for the representation of a matrix $M \in \mathbb{C}^{N \times N}$ as an MPO, the approach presented above can easily be adapted to yield

$$M_{ij} = M_{i_1 \dots i_L j_1 \dots j_L} = A_1^{i_1 j_1} A_2^{i_2 j_2} \cdots A_L^{i_L j_L}$$

where i, j have been split up as before and $A_1, \dots, A_L \in \mathbb{C}^{d \times d \times D \times D}$. In analogy to the case for a vector, we can write the whole matrix as

$$\begin{aligned}
M &= \sum_{i_1,\dots,i_L}^{d} \sum_{j_1,\dots,j_L}^{d} \left(A_1^{i_1 j_1} A_2^{i_2 j_2} \cdots A_L^{i_L j_L}\right) \\
&\qquad \cdot \left(e_{i_1} \otimes e_{i_2} \otimes \cdots \otimes e_{i_L}\right)\left(e_{j_1}^T \otimes e_{j_2}^T \otimes \cdots \otimes e_{j_L}^T\right) \\
&= \sum_{i_1,\dots,i_L}^{d} \sum_{j_1,\dots,j_L}^{d} \sum_{k_2,\dots,k_{L-1}}^{D} \left(A_{1,1k_2}^{i_1 j_1} A_{2,k_2k_3}^{i_2 j_2} \cdots A_{L,k_{L-1}1}^{i_L j_L}\right) \\
&\qquad \cdot \left(e_{i_1} e_{j_1}^T\right) \otimes \left(e_{i_2} e_{j_2}^T\right) \otimes \cdots \otimes \left(e_{i_L} e_{j_L}^T\right) \\
&= \sum_{k_2,\dots,k_{L-1}}^{D} U_{1,k_2} \otimes U_{2,k_2k_3} \otimes \cdots \otimes U_{L,k_{L-1}}
\end{aligned}$$

where e_j is again the jth column of the identity and e_j^T is its transpose. Note that this also holds true for other product bases, like for instance the Pauli basis. Making use of these formulations, it is easy to show that basic operations such as scalar multiplication, addition and inner product as well as the multiplication of an MPS by an MPO or of two MPOs can be performed in the formalism. The addition and non-scalar multiplication however lead to an increase in the bond dimension D. For the addition of two MPS/MPOs with bond dimensions D and D', the new bond dimension is $D'' = D + D'$ and for the multiplication, $D'' = D \cdot D'$ [Sch11]. This can again easily be verified.

It is obvious from the above explanation that the bond dimension is the decisive factor for the expressive power of the formalism. An exact representation of a vector (operator) as an MPS (MPO) is always possible if we allow the bond dimension, D, to be big enough, which may mean exponentially large in L, up to $d^{N/2}$ [VPC04]. When the maximum value of D is limited to some small value (*truncated*) not all vectors or operators can be represented, what may give raise to approximation errors. We will in the following denote that some vector v or matrix M is approximated with bond dimension D by writing $v[D]$ and $M[D]$ respectively. Nevertheless, it has been found that MPS/MPOs often yield good approximations for $D \in \mathcal{O}(poly(L))$ leading to the total number of parameters $LdD^2 \in \mathcal{O}(poly(L))$ as opposed to d^L or

Algorithm 1: Original Lanczos Algorithm

Input : Matrix $A \in \mathbb{C}^{N\times N}$, Starting Vector $u \in \mathbb{C}^N$, Number of Dimensions K

$u_0 \leftarrow 0$;
$v_0 \leftarrow u$;
for $i \leftarrow 1; i \leq K$ **do**
  $\beta_i \leftarrow \|v_{i-1}\|$;
  if $\beta_i = 0$ **then**
    break ;
  end
  $u_i \leftarrow v_{i-1}/\beta_i$;
  $v_i \leftarrow Au_i - \beta_i u_{i-1}$;
  $\alpha_i \leftarrow u_i^* v_i$;
  $v_i \leftarrow v_i - \alpha_i u_i$;
end

Output: Orthonormal Basis $U_K \in \mathbb{C}^{N\times K}$, Tridiagonal Matrix $T_K \in \mathbb{R}^{K\times K}$

d^2L for the whole vector or matrix, respectively. This constitutes another main reason for their usefulness as an efficient approximation scheme.

Naturally, many methods have been developed to find optimal and canonical representations for a given D both in the numerical and the quantum physics community. The most important algorithms for optimizing a given MPS/MPO with respect to some error function and bond-dimension thereby rely on local updates of the individual A_i, with all other sites being treated as constants, rather than considering all parameters simultaneously. These algorithms, starting with the left- or rightmost site, generally sweep back and forth over the chain of sites, updating one site per step, until convergence. As all sites not considered in a given step are treated as fixed, this sweeping scheme allows for reusage of previously computed values in a dynamical programming fashion. As explaining the details and the complexity of these algorithms exceeds the scope of this section, we refer the interested reader to the overview articles [Sch11, VMC08, BSU16, GKT13].

III. THE GLOBAL LANCZOS ALGORITHM

The idea of employing variants of the Krylov method to solve various types of problems, for instance solving linear systems [SS86, Mey98, KT11], finding eigenvectors [CRS94, Arn51, Lan50, Kry31] or approximating the action of an exponential operator onto a vector [GR06], is already well-established. All the (non-block) methods have in common that, starting with a given matrix $A \in \mathbb{C}^{N\times N}$ and an initial vector $u \in \mathbb{C}^N$, they construct an orthonormal basis $U_K = [u_1, u_2, \ldots, u_K] \in \mathbb{C}^{N\times K}$ of the Krylov space $\mathcal{KR}_K = \{u, Au, A^2u, \ldots, A^{K-1}u\}$, where K is the dimension of the space. We will now first describe the original Lanczos method and then generalize our findings to the global algorithm. The original Lanczos algorithm is a variant of the general Krylov method that assumes A to be hermitian and in our case makes use of the Gram-Schmidt (GS) orthogonalization method to construct U_K. During the process of building U_K, the Lanczos method produces a matrix $T_K \in \mathbb{R}^{K\times K}$ that is given by

$$T_K = \begin{bmatrix} \alpha_1 & \beta_2 & & \mathbf{0} \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_K \\ \mathbf{0} & & \beta_K & \alpha_K \end{bmatrix},$$

where α_i and β_i are defined as in Algorithm 1. Reminding ourselves of the orthonormality of U_K and the hermiticity of A, it is easy to verify that

$$U_K^* A U_K = T_K,$$

where U^* represents the Hermitian conjugate, and that hence A is similar to T_N, T_N being the tridiagonal matrix of full dimension N. This provides an interesting perspective on the Lanczos method as a projection onto a lower-dimensional space and a first glance at how one may approximate the spectrum of A via the eigenvalues of T_K, called the Ritz values of A [Saa92, TBI97]. Note that T_K has to be real because the β_i are vector-norms and the α_i have to be real due to the hermiticity of A and its similarity to T_N. We now define an extended version $\widehat{T}_K$ of T_K by

$$\widehat{T}_K = \begin{bmatrix} \alpha_1 & \beta_2 & & \mathbf{0} \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_K \\ \mathbf{0} & & \beta_K & \alpha_K \\ 0 & \ldots & 0 & \beta_{K+1} \end{bmatrix}$$

and use it to state the partial Lanczos decomposition

$$AU_K = U_{K+1}\widehat{T}_K = U_K T_K + \beta_{K+1} u_{K+1} e_K^T$$

that links U_K to U_{K+1} for $K < N$. The decomposition makes it clear that the improvement from one step of the algorithm to the next is proportional to β_{K+1}. In the case where $\beta_{K+1} = 0$, the decomposition is exact and U_K spans an eigenspace of A. For the decomposition, we have implicitly used a three-term recurrence relation that can be read off directly from Algorithm 1. By construction it holds that

$$Au_i = \beta_{i+1}u_{i+1} + \beta_i u_{i-1} + \alpha_i u_i$$

which can be rewritten to yield

$$\beta_{i+1}u_{i+1} = (A - \alpha_i I_N)u_i - \beta_i u_{i-1} \tag{1}$$

where I_N is the $N \times N$ identity matrix. This relation plays an important role in our algorithm and we will get back to it when we explain how we approximate functionals in the next section.

However, before explaining the rigorous approach, we will provide two more intuitive arguments to show why Krylov methods can be used for our purpose. From the above explanations, it is evident that every vector $v \in \text{span}\,(U_K)$ is given by

$$v = \sum_{i=0}^{K-1} c_i (A^i u) = \left(\sum_{i=0}^{K-1} c_i A^i \right) u = p_v(A) u$$

where $p_v(A)$ is some polynomial of A with degree $K-1$. Note that there exist instances of v for which $p_v(A)$ yields a power-series approximation around 0 of degree $K-1$ of some desired function $f(A)$. This is a key insight if one wants to understand why Krylov methods are very useful for problems like the ones mentioned above. It is obvious from this formulation that K has a strong impact on the error of the approximation of $f(A)$ which naturally leads to asking whether bounds can be derived for K with respect to certain functions. It can be seen that the K needed for exactness can be determined from the degree of the minimal polynomial of A [Mey98]. We have seen above that $U_K^* A U_K = T_K$. Therefore it holds that

$$\begin{aligned} U_K^* f(A) U_K &\approx U_K^* \left(\sum_{i=0}^{K-1} c_i A^i \right) U_K \\ &\approx \sum_{i=0}^{K-1} c_i (U_K^* A U_K)^i \\ &= \sum_{i=0}^{K-1} c_i T_K^i \\ &\approx f(T_K), \end{aligned}$$

where we again assume some power-series approximation of $f(A)$ of degree $K-1$ and make use of the fact that $\lim_{K \to N} U_K^* U_K = I$.

After having reviewed the original algorithm, we now proceed to state the global block version used in this work. In order to do so, we will from now on assume to be given an *initial matrix* $U \in \mathbb{C}^{N \times M}$. Starting from U, the algorithm will then build up a basis of matrices $\mathbf{U}_K = [U_1, \cdots, U_K]$ with $\mathbf{U}_K \in \mathbb{C}^{N \times KM}$. While there exist several block-versions of the Lanczos algorithm [EMS09, GLS94, Mon95, GLO81, CD74], we will only consider the one presented in [BRRS15] as it does not require the columns of U_i to be orthogonal, which would be prohibitive for very large matrices. Now, we first need to state the inner product with respect to which the individual U_i must be orthonormal and define it to be

$$\langle U_i, U_j \rangle = \mathsf{Tr} U_i^* U_j$$

where $U_i, U_j \in \mathbb{C}^{N \times M}$. This induces the well known Frobenius norm

$$\|U_i\|_F = \langle U_i, U_i \rangle^{1/2}$$

Algorithm 2: Global Lanczos Algorithm

Input : Matrix $A \in \mathbb{C}^{N \times N}$, Starting Matrix $U \in \mathbb{C}^{N \times M}$, Number of Dimensions K

$U_0 \leftarrow 0$;
$V_0 \leftarrow U$;
for $i \leftarrow 1; i \leq K$ **do**
  $\beta_i \leftarrow \|V_{i-1}\|_F$;
  if $\beta_i = 0$ **then**
    break ;
  end
  $U_i \leftarrow V_{i-1} / \beta_i$;
  $V_i \leftarrow A U_i - \beta_i U_{i-1}$;
  $\alpha_i \leftarrow \langle U_i, V_i \rangle$;
  $V_i \leftarrow V_i - \alpha_i U_i$;
end

Output: Orthonormal Basis $\mathbf{U}_K \in \mathbb{C}^{N \times KM}$, Tridiagonal Matrix $T_K \in \mathbb{R}^{KM \times KM}$

and hence this definition of the inner product is a straight-forward generalization of the one used in Algorithm 1. Naturally, one may also choose different inner products [EMS09]. For this work, we do however choose the Frobenius norm as it can be efficiently computed for MPOs. Equipped with this definition, we can see that Algorithm 2 is in fact a direct generalization to the matrix-case. As such we find that after K steps, the method has produced the reduction of A T_K and yields the partial global Lanczos decomposition

$$A \mathbf{U}_K = \mathbf{U}_K \tilde{T}_K + \beta_{K+1} U_{K+1} E_K^T$$

where we define $\tilde{T}_K = T_K \otimes I_M \in \mathbb{R}^{KM \times KM}$ and $E_K^T = [\mathbf{0}, \cdots, \mathbf{0}, I_M] \in \mathbb{R}^{M \times KM}$. Furthermore, it again holds that

$$\beta_{i+1} U_{i+1} = (A - \alpha_i I_N) U_i - \beta_i U_{i-1} \tag{2}$$

and

$$\mathbf{U}_K^* A \mathbf{U}_K = T_K$$

if we apply the inner product defined previously. From now on, we will implicitly make use of this inner product whenever appropriate. Then, all other observations made for the original Lanczos method above carry over to the global Lanczos case.

IV. INTRODUCING GAUSS-TYPE QUADRATURE

In the previous section, we have sketched that for the Lanczos method $U_K^* f(A) U_K \approx f(T_K)$ and hence

$$u_1^* f(A) u_1 = e_1^T U_K^* f(A) U_K e_1 \approx e_1^T f(T_D) e_1,$$

which generalizes to the global Lanczos method. However, there exists a well studied and more rigorous approach to function approximation via the Lanczos

methods [BFG96, GM09, GM94]. This approach revolves around the connection between T_K and Gauss-type quadrature rules.

To establish the link between both methods, we start by observing that

$$\begin{aligned} u_1^* f(A) u_1 &= u_1^* V f(\Lambda) V^* u_1 \\ &= \sum_{i=1}^{N} f(\lambda_i)\mu_i^2 \\ &= \int_a^b f(\lambda)\mathrm{d}\mu(\lambda) \end{aligned}$$

with $V\Lambda V^*$ being the spectral decomposition of A, $\mu_i = e_i^T V^* u_1$ and

$$\mu(\lambda) = \begin{cases} 0 & \text{if } \lambda < \lambda_1 = a \\ \sum_{i=1}^{j} \mu_i^2 & \text{if } \lambda_j \leq \lambda < \lambda_{j+1} \\ \sum_{i=1}^{N} \mu_i^2 & \text{if } b = \lambda_N \leq \lambda \end{cases}$$

being a piecewise constant and nondecreasing distribution function. Here we assume the eigenvalues of A to be ordered ascendingly. We can use this result to obtain

$$\begin{aligned} \mathcal{I}f &\coloneqq \mathsf{Tr}(U_1^* f(A) U_1) \\ &= \sum_{i=1}^{N} e_i^* U_1^* V f(\Lambda) V^* U_1 e_i \\ &= \sum_{i=1}^{N} \int_a^b f(\lambda)\mathrm{d}\mu_i(\lambda) \\ &= \int_a^b f(\lambda)\mathrm{d}\sum_{i=1}^{N} \mu_i(\lambda) \\ &= \int_a^b f(\lambda)\mathrm{d}\mu(\lambda) \end{aligned} \tag{3}$$

where we define $\mu_i(\lambda)$ analogously to the case above and $\mu(\lambda) \coloneqq \sum_{i=1}^{N} \mu_i(\lambda)$. This Riemann-Stieltjes integral can now be tackled via Gauss-type quadrature.

The most general formulation of a Gauss-type quadrature rule is

$$\mathcal{G}f \coloneqq \sum_{i=1}^{K} \omega_i f(\theta_i) + \sum_{j=1}^{M} \nu_j f(\tau_j)$$

where θ_i and τ_j are called the nodes and ω_i and ν_j the weights of the quadrature. The remainder of such an approximation is given by

$$\begin{aligned} \mathcal{R}f &\coloneqq \int_a^b f(\lambda)\mathrm{d}\mu(\lambda) - \mathcal{G}f \\ &= \frac{f^{2K+M}(\eta)}{(2K+M)!} \int_a^b \prod_{i=1}^{M}(\lambda - \tau_i) \left(\prod_{j=1}^{K} (\lambda - \theta_j) \right)^2 \mathrm{d}\mu(\lambda) \end{aligned}$$

where $\lambda_1 < \eta < \lambda_N$. Setting $M = 0$ yields the Gauss quadrature, while using prescribed nodes results, e. g., in the Gauss-Radau and Gauss-Lobatto quadratures [DR07, BRRS15]. For the Gauss quadrature, the sign of the remainder thus is given by the sign of the $2K$-th derivative of f. Therefore, it is easy to determine whether, for a given f, the Gauss quadrature provides a lower or upper bound of the correct value. By application of the Weierstrass-theorem, it can also be shown that

$$\lim_{K\to\infty} \mathcal{G}_K f = \mathcal{I}f$$

where $\mathcal{G}_K f$ is the Gauss quadrature with K nodes. Furthermore, $\mathcal{G}_K f$ is exact for all polynomials of degree at most $2K-1$ [DR07]. It is well known that in order to determine the ω_i and θ_i that satisfy this property, one can construct a sequence of polynomials $\{p_0, \cdots, p_K\}$ that are orthonormal in the sense that

$$\int_a^b p_i(\lambda) p_j(\lambda)\mathrm{d}\mu(\lambda) = \delta_{ij}$$

and that satisfy a recurrence relation given by

$$\beta_j p_j(\lambda) = (\lambda - \alpha_{j-1}) p_{j-1}(\lambda) - \beta_{j-1} p_{j-2}(\lambda), \tag{4}$$

where $p_{-1}(\lambda) \equiv 0$ and $p_0(\lambda) \equiv 1$. Then, the roots of p_K ca be shown to be the optimal θ_i [DR07, GM09]. Now, the recurrence relation yields a recurrence matrix T_K defined by

$$T_K = \begin{bmatrix} \alpha_1 & \beta_2 & & \mathbf{0} \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_K \\ \mathbf{0} & & \beta_K & \alpha_K \end{bmatrix}$$

whose eigenvalues are the zeroes of $p_K(\lambda)$ and consequentially the θ_i of $\mathcal{G}f$ [GM09]. The ω_i are given by the squared first elements of the normalized eigenvectors of T_K and so,

$$\mathcal{G}f = e_1^T f(T_K) e_1 = e_1^T V f(\Lambda) V^* e_1,$$

where $V\Lambda V^*$ is the spectral decomposition of T_K.

Now, the U_i from the global Lanczos method can be expressed by

$$U_i = p_{i-1}(A) U$$

with p_{i-1} being some polynomial of degree $i-1$. Then, it is clear that

$$\langle p_{i-1}(A)U, p_{j-1}(A)U \rangle = \langle U_i, U_j \rangle = \delta_{ij}$$

and taking into account the above derivations

$$\begin{aligned} \langle p_{i-1}(A)U, p_{j-1}(A)U \rangle &= \mathsf{Tr}(U^* p_{i-1}(A)^* p_{j-1}(A) U) \\ &= \int_a^b p_{i-1}(\lambda) p_{j-1}(\lambda)\mathrm{d}\mu(\lambda). \end{aligned}$$

Algorithm 3: Approximation Algorithm

Input : MPO $A[D_A] \in \mathbb{C}^{N\times N}$, Starting orthogonal MPO $U[D_{init}] \in \mathbb{C}^{N\times N}$, Number of Dimensions K, Maximal Bond-Dimension D_{max}, Stopping Criteria $\mathcal{S}$

$U_0 \leftarrow 0$;
$V_0 \leftarrow U$;
$D \leftarrow D_{init}$;
for $i \leftarrow 1; i \leq K$ **do**
  $\beta_i \leftarrow \sqrt{\texttt{contract}(V_{i-1}, V_{i-1})}$;
  if $\beta_i = 0$ **then**
    break ;
  end
  $U_i \leftarrow \texttt{multiplyScalar}(1/\beta_i, V_{i-1})$;
  $D \leftarrow \min(D_{max}, D \cdot D_A)$;
  $V_i \leftarrow \texttt{multiplyAndOptimize}(A, U_i, D)$;
  $D \leftarrow \min(D_{max}, D + D_{U_{i-1}})$;
  $V_i \leftarrow \texttt{sumAndOptimize}(V_i, -\beta_i U_{i-1}, D)$;
  $\alpha_i \leftarrow \texttt{contract}(U_i, V_i)$;
  $D \leftarrow \min(D_{max}, D + D_{U_i})$;
  $V_i \leftarrow \texttt{sumAndOptimize}(V_i, -\alpha_i U_i, D)$;
  $V\Lambda V^* \leftarrow \texttt{spectralDecomposition}(T_i)$;
  $\mathcal{G}f \leftarrow \beta_1^2 e_1^T V f(\Lambda) V^* e_1$;
  if $\texttt{checkStop}(\mathcal{G}f, \Lambda, \mathcal{S})$ **then**
    break ;
  end
end

Output: Approximation $\mathcal{G}f$ of $\mathsf{Tr}f(A)$

Hence, the global Lanczos method produces orthonormal polynomials [Lan50] that in addition satisfy the recurrence relation stated in Equation 4 by construction, as we have shown in Section III. The T_K obtained by the global Lanczos algorithm is thus the recurrence matrix needed to perform the Gauss quadrature. If we choose $U \in \mathbb{C}^{N\times N}$ to be orthonormal, it follows that

$$\mathsf{Tr}f(A) = \mathsf{Tr}(U^* f(A) U) = \int_a^b f(\lambda) \mathrm{d}\mu(\lambda) \approx e_1^T f(T_K) e_1.$$

V. ASSEMBLING THE PARTS

Now that we have reviewed the relevant theoretical aspects, we will proceed by showing how we put together the pieces to obtain our algorithm. The whole algorithm is shown in Algorithm 3.

Since the global Lanczos method is based on matrix-matrix multiplications, additions of matrices and multiplications of matrices by scalars, these operations have to be formulated for the MPO-case. As the bond dimension of the basis-MPOs grows with the number of multiplications and additions, we need to keep track of the bond dimensions and perform projections onto lower bond dimensions whenever necessary. Thus, the input parameters of our method are the MPO $A[D_A] \in \mathcal{C}^{N\times N}$, an orthogonal starting MPO $U[D_{init}] \in \mathcal{C}^{N\times N}$, the maximal Krylov-dimension K, a set of stopping criteria $\mathcal{S}$ and finally the maximal bond-dimension D_{max} of the U_i. Note that we assume $U[D_{init}]$ to be orthogonal and of the same dimension as $A[D_A]$. This allows us to replace the approximation that had to be made previously by combining the estimations for several starting matrices by the exact computation. This is only possible due to the fact that we translate the Lanczos method to the MPO-formalism. Besides the case of very large matrices that can be explicitly stored but are too large for multiplications with equally-sized matrices, this is especially important for the case where the respective matrices are only given in MPO-format and N is of the order of several millions, as in the case of quantum many body systems. While we introduce some approximation error by using MPOs, we will show in Section VI that these errors can be comparably small already for low bond dimensions in cases of practical interest. In the following, we will omit the declaration of the bond-dimension of an MPO whenever it increases clarity.

While in principle every orthogonal MPO can serve as a starting point, in this work we choose $U[D_{init}]$ to be the identity matrix because it has a minimal MPO-formulation of $A_i^{jk} = \delta_{jk}$. This allows us to start from the minimal bond-dimension $D_{init} = 1$ and thus maximizes the amount of relevant information we can store for a given D_{max}. In certain cases it might however be possible to choose a better starting MPO, e. g., when A is very close to being diagonal. For the implementation of the inner product and norm used in the global Lanczos algorithm, we observe that

$$\langle U_i, U_j \rangle = \sum_{k=1}^{N} \sum_{l=1}^{N} U_{i,kl} U_{j,lk} = U_{i,vec}^* U_{j,vec}$$

where $U_{i,vec}$ and $U_{j,vec}$ are the vectorized versions of U_i and U_j respectively. This allows us to make use of an efficient, exact way of computing the inner product of MPS [Sch11] by rewriting the $A_i^{j_i k_i}$ as $A_i^{j_i'}$ with $\dim j_i' = \dim j_i \cdot \dim k_i$ and hence vectorizing the MPO. This functionality is implemented in `contract`(). The implementation of the scalar multiplication `multiplyScalar`() is straightforward as it corresponds to multiplying an arbitrary A_i, we choose A_1 for simplicity, of the respective MPO by the scalar at hand.

A bit more care has to be taken when implementing the functions `multiplyAndOptimize`() and `sumAndOptimize`(), the multiplication and summation of MPOs respectively. One possiblity would be to first perform the respective operation exactly, i.e., use the bond-dimension required for the exact result, and to project the resulting MPO onto the current D via the singular value decomposition (SVD) of its A_i in a second step. It has however been found that performing the projection simultaneously to the multiplication or summation at the level of the individual A_i yields superiour results, see [VMC08, Sch11, Wal14]. In case of the multiplication, we implement this strategy by solving the optimization

problem

$$\min_{\tilde{A}_i} \|AU_j[D_{old}] - \tilde{U_j}[D_{new}]\|_F^2$$

where D_{old} is the bond dimension used previous to the multiplication and D_{new} is the bond dimension used for the optimization. $\tilde{U_j}[D_{new}]$ denotes the result of the multiplication of A on U_j and the $\tilde{A}_i$ are its tensors. The implementation hence performs the multiplication of the MPOs at tensor level and directly optimizes the resulting tensors for the chosen bond dimension by employing the sweeping scheme sketched in Section II. In order to apply this algorithm to the case of MPO-MPO multiplication, we rewrite AU_j as

$$(I \otimes A)U_{j,vec} = \begin{bmatrix} A & 0 & & \mathbf{0} \\ 0 & A & \ddots & \\ & \ddots & \ddots & 0 \\ \mathbf{0} & & 0 & A \end{bmatrix} \begin{bmatrix} U_{j,1} \\ U_{j,2} \\ \vdots \\ U_{j,N} \end{bmatrix}$$

with $U_{j,k}$ being the kth column vector of U_j. Due to technical reasons, we do in fact use $U_j \otimes I$ and A_{vec}. For the summation, we apply the same strategy and solve

$$\min_{\tilde{A}_i} \| \left(U_j[D_{old}] + \sum_k \gamma_k U_k[D_{old}^k] \right) - \tilde{U_j}[D_{new}] \|_F^2$$

where D_{old} is the bond dimension used before the addition, D_{old}^k are some other previously used bond dimensions, D_{new} is the bond-dimension to be used for the optimization and $\gamma_k \in \mathbb{C}$ are some scalars. $\tilde{U_j}[D_{new}]$, similar to before, represents the outcome of the summation and the $\tilde{A}_i$ are its tensors.

As it can be seen in Algorithm 3, we allow for exact multiplication and summation as long as the resulting bond-dimension does not grow beyond D_{max}. This however happens quickly, since D grows exponentially with the number of iterations, and so, most of the U_i will be represented based on D_{max}. This underlines the importance of D_{max} for the accuracy of the approximation.

After the algorithm has completed one iteration of the global Krylov method, the spectral decomposition of T_i is performed and the current approximation is computed. Based on the approximation and the eigenvalues of T_i the algorithm then determines if it should be stopped in `checkStop`. Here we have to account for several factors.

Firstly, we know that $\mathcal{G}f$ converges to the correct value in absence of approximation errors. So, the algorithm can terminate when the distance between the previous and current $\mathcal{G}f$ becomes smaller than some ϵ.

Secondly, it is clear that the projection of the generated MPOs down to D_{max} introduces an approximation error. While it is possible to obtain the error of the optimization problems described above, it is not clear how the accumulated error influences $\mathcal{G}f$ precisely. However, a possible way of detecting when the approximation error has become too large is to check for the violation of some theoretical constraints. For instance, in case of a positive A, we know that the Ritz-values A must be positive as well. If T_i starts to have negative eigenvalues, we thus know that the total approximation error has reached a level that leads to unreasonable results. The same reasoning could be applied for other intervals in which one knows the spectrum of A to be in.

We have also seen in the previous section that, depending on the sign of the derivative of f in (λ_1, λ_N), $\mathcal{G}f$ yields an upper/lower bound and that it converges to the true value. Based on this it is possible to show that $\mathcal{G}_M < \mathcal{G}_{M+1}$ for the lower-bound case and $\mathcal{G}_M > \mathcal{G}_{M+1}$ for the case of an upper bound [GM09]. This provides another stopping-criterion.

As the accumulation of truncation errors can lead to unreasonable results even before the violation of the above property, we propose to keep a moving average over the last k approximations and employ the 3σ-rule to detect improbable results. This heuristic is justified by the guaranteed convergence in the absence of numerical errors.

After having explained the algorithm, a few remarks are in order:

- In this version of the algorithm, we only consider the Gauss quadrature. This is mainly due to the fact that obtaining good lower or upper bounds on the spectrum of A is in general not possible because of the size of its dimensions. Analogously to [BRRS15], our algorithm can nevertheless be adapted to perform Gauss-Radau or Gauss-Lobatto approximations.

- To improve numerical stability and prevent 'ghost' eigenvalues from occuring, it could be beneficial to perform reorthogonalization. Due to the MPO-representation, this would however be very costly and not necessarily result in a large improvement. Thus, we do not consider this extension. It can however be easily added to the algorithm.

- In the presented algorithm, we stick to the canonical way of orthogonalizing the new matrix against the old matrices individually. In the case of exactly stored matrices/vectors, this scheme increases numerical stability. Since we now employ approximations of the exact matrices, it might however be worth considering to compute α_i first and then optimize the sum containing both U_i and U_{i-1}. The advantage of being able to optimize the whole expression at once might outweigh the disadvantage of orthogonalizing against both matrices simultaneously. On the other hand side, computing α_i first might lead to different and possibly worse results.

- As we have stated above, it is possible to obtain approximation errors from both `multiplyAndOptimize` and `sumAndOptimize`. But these errors naturally only refer to the current optimization and do not allow for strict bounds

TABLE I: A listing of the complexity of the subfunctions of Algorithm 3. $L = \log(N)$ is the number of tensors of the MPOs, d is the physical dimension. For simplicity, all U_i are assumed to have bond-dimension D_{max} and T_i is assumed to be in $\mathbb{R}^{K\times K}$.

`contract`	$\mathcal{O}(LD_{max}^3 d^2)$
`multiplyAndOptimize`	$\mathcal{O}(LD_{max}^3 D_A d^2)$
`sumAndOptimize`	$\mathcal{O}(LD_{max}^3 d^2)$
`multiplyScalar`	$\mathcal{O}(D_{max}^2 d)$
`spectralDecomposition`	$\mathcal{O}(K^3)$
`checkStop`	$\mathcal{O}(1)$

on the overall error. One could of course try to increase the bond dimension for each individual optimization until its error converges to make sure the partial result is close to exact. The problem here is that due to the possibly exponential growth of the bond dimension needed for exactness, D_{max} is typically reached within very few iterations. From this point on, it is not possible any more to increase D and so, the information about the error provides little useful information. This is why we have resorted to the approach of checking for the violation of theoretically guaranteed behaviour.

- From the above explanations it is clear that K and D_{max} are the parameters that control the accuracy of the approximation. For the algorithm to be of use for very high-dimensional matrices, we must impose the restriction that $K, D_{max} \in \mathcal{O}(poly(L))$. This property is particularly relevant for quantum mechanical simulations where N grows exponentially with the number of particles.

We will conclude this section with an analysis of the complexity of our algorithm. The complexities of the subfunctions of Algorithm 3 are listed in Table I. For the analysis of `multiplyAndOptimize`, we have assumed D_A to be of the same order as D_{max}. If it were significantly larger, the complexity would change to $\mathcal{O}(LD_{max}^2 D_A^2 d^2)$. Note that this analysis does not extend to the number of sweeps necessary for the optimizations to converge. For the spectral decomposition, we have for simplicity assumed all T_i to be of size $K \times K$. Combining all the different results, we thus find that the overall complexity of Algorithm 3 is in $\mathcal{O}(KLD_{max}^3 D_A d^4)$ with $L = \log(N)$ and since we require $K, D_{max} \in \mathcal{O}(\log N)$, this is equivalent to $\mathcal{O}(poly(L))$.

VI. NUMERICAL RESULTS

In this section we present numerical results obtained for a challenging problem of relevance in quantum many body physics. Our goal thereby is to study the convergence of the results with increasing K and D_{max}. The problem we consider is the approximation of the von Neumann entropy. For a quantum state ρ [Nota], the von Neumann entropy is given by $S = -\mathsf{Tr}\rho\log\rho$. In the following, we will focus on the case of states of the form $\rho = e^{-\beta H} / \mathsf{Tr}e^{-\beta H}$, i.e., thermal equilibrium states for a Hamiltonian H, at a certain inverse temperature, β [Notb]. Here, we assume H to be the Ising Hamiltonian with open boundary conditions that is given by

$$H = J\sum_{i=1}^{L-1}\sigma_i^x\sigma_{i+1}^x + g\sum_{i=1}^{L}\sigma_i^z + h\sum_{i=1}^{L}\sigma_i^x$$

where $\sigma^{\{x,y,z\}}$ are the Pauli matrices

$$\sigma^x = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}, \sigma^y = \begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}, \sigma^z = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}.$$

Note however that here by $\sigma_i^{\{x,y,z\}}$ we actually denote the tensor product $I_1 \cdots \otimes I_{i-1} \otimes \sigma_i^{\{x,y,z\}} \otimes I_{i+1} \otimes \cdots \otimes I_L$ and analogously for $\sigma_i^{\{x,y,z\}}\sigma_{i+1}^{\{x,y,z\}}$ for simplicity of notation. The Hamiltonian describes a chain of spin particles with nearest neighbour interactions and two magnetic fields acting only on the individual particles. This choice of H has the advantage that it is exactly solvable for $h = 0$, a case commonly known as 'transverse field Ising', and thus opens the possibility to obtain a reference solution for system sizes for which it could otherwise not be obtained [Sac07, Kar06, SIC12]. It is possible to find an MPO-approximation to the thermal equilibrium state, ρ, by means of standard MPS-techniques [VGRC04, ZV04, GR06]. It is customary to use a *purification* ansatz for this purpose, where $\rho(\beta/2)$ is approximated by standard imaginary time evolution, and the whole state is then written as $\rho \propto \rho(\beta/2)^*\rho(\beta/2)$. In the context of our algorithm, nevertheless, applying exactly this ρ involves a larger cost and worse numerical condition. Instead, we apply the method as described above to $\rho(\beta/2)$ and absorb the necessary squaring into the function that is to be approximated. In our case this means that instead of computing $f(\lambda_i) = \lambda_i\log\lambda_i$ for each Ritz-value, we simply compute $f(\lambda_i)' = \lambda_i^2\log\lambda_i^2$. This allows us to apply the algorithm to a possibly much more benign input at the cost of an only slightly more complicated function. Due to truncation errors, the operator $\rho(\beta/2)$ may not be exactly Hermitian. This can be easily accounted for by taking its Hermitian part, $\frac{1}{2}[\rho(\beta/2)^* + \rho(\beta/2)]$, which is an MPO with at most twice the original bond dimension. In our experiments we however did not find this to be necessary.

Apart from the entropy, another interesting function to examine would have been the trace norm given by $||\rho||_1 = \mathsf{Tr}\sqrt{\rho^*\rho}$, i. e., the sum of the singular values. But as we only consider positive matrices in this scenario, this sum is equal to the sum of the eigenvalues which we know to be equal to 1 due to the normalization of the thermal

state. In fact, we find that for $\rho(\beta)$ α_1 as computed by our algorithm is given by

$$\alpha_1 = \mathsf{Tr} U_1^* V_1 = \mathsf{Tr} U_1^* \rho U_1 = \frac{1}{\beta_1^2} \mathsf{Tr} U^* \rho U = \frac{1}{\beta_1^2} \mathsf{Tr} \rho.$$

So, in this case the algorithm computes the exact result in one step. We verified this result numerically and found it to hold for all considered cases. It is also easily possible to adapt the above trick to compute the trace norm from $\rho(\frac{\beta}{2})$. We found this to approach to work very well but do not present results here as computing the value directly as shown above is of course preferable. In future work, the trace norm might be used as a distance measure between different thermal states.

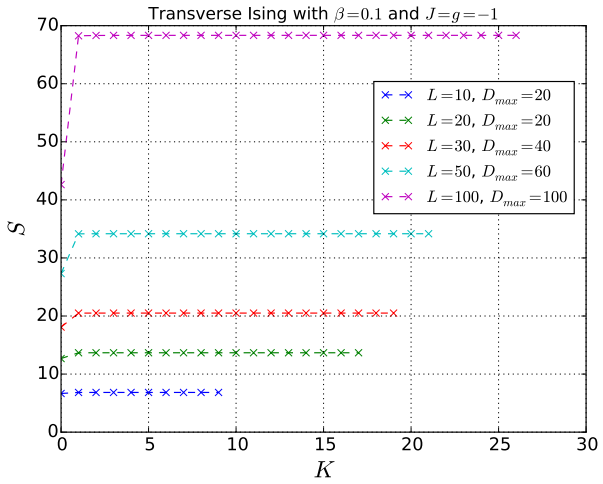
As was explained in Section IV, the $2K$-th derivative of f determines whether $\mathcal{G}f$ poses a lower or upper bound of the true value. In our case and for $K > 1$, it is given by

$$\frac{d^{2K} - \lambda_i^2 \ln \lambda_i^2}{d^{2K} \lambda_i} = \frac{4(2K-3)!}{\lambda_i^{2K-2}}$$

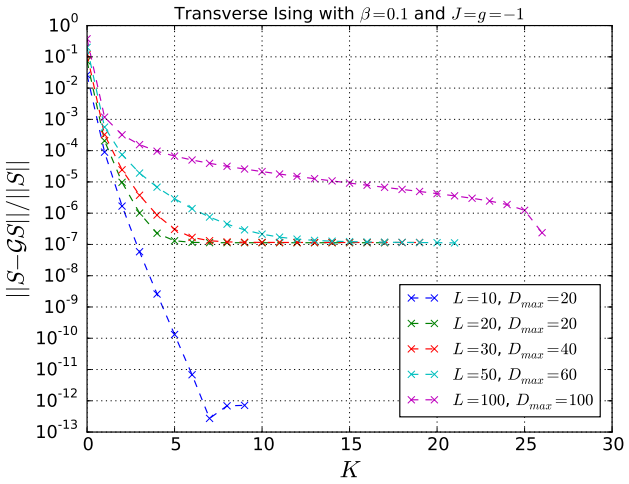
with λ_i being the i-th eigenvalue of ρ. Hence, we can expect our algorithm to provide increasingly tight lower bounds for the correct value. We use the violation of this property as a stopping criterion to account for the situation when truncation errors become too large. Additionally, we keep the average of the last three or four, depending on β, approximations and employ the aforementioned 3σ-rule. In case these stopping criteria are not met, we terminate the algorithm when the difference between successive approximations is below 10^{-10}.

In our experiments we considered systems of size $L \in \{10, 20, 30, 50, 100\}$, Hamiltonian parameters $J = g = 1$ and inverse temperatures $\beta \in \{0.1, 1.0\}$. The bond dimension used to obtain $\rho(\frac{\beta}{2})$ was set to 20 for all cases. The convergence of the approximation as well as the relative error for $\beta = 0.1$ and $\beta = 1$ are shown in Figure 1 and Figure 2, respectively. Note that in order to compute the relative error, we used numerical diagonalization for $L = 10$ and the analytical solution [Sac07] for $L > 10$. Figure 3 shows the change of the relative error of the final approximations with growing D_{max} for $L = 100$ and $\beta = 0.1$.

For the case of $\beta = 0.1$ we observe fast convergence to good approximations in K and D_{max}, as shown in Figure 1. The maximal bond dimension required for good convergence only grows mildly with L allowing our method to scale very well with the size of the input. The plots in Figure 1b show a plateau in the relative error at 10^{-7}. This corresponds to the non-vanishing difference between the exact solution and the numerical MPO used as input. Notice that for $L = 10$, where the input is exact, the method is able to achieve a smaller error. Figure 3 illustrates that while our method achieves a good error even for a small D_{max} of 20, it profits from an increase of the maximal bond dimension. The error decreases by two orders of magnitude from 10^{-5} to about 10^{-7} when



(a) Entropy over K.

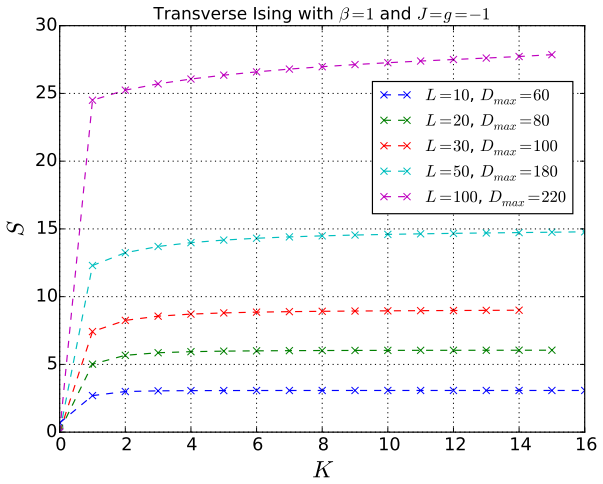


(b) Relative error in the entropy over K.

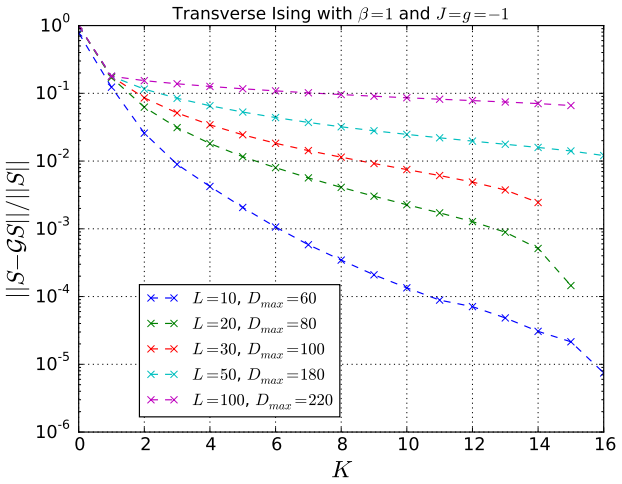
FIG. 1: Convergence behavior of the algorithm for $L \in \{10, 20, 30, 50, 100\}$, $\beta = 0.1$ and varying Krylov-dimension K. In (a), the convergence of the approximation is depicted. In (b), the convergence of the relative error is shown.

D_{max} is raised from 20 to 180 which still constitutes a strong truncation. It is conceivable that a further increase of the maximal bond dimension would improve the accuracy. We also found that D_{max} limits the number of basis MPOs that can be successfully orthogonalized and therefore effectively controls the maximally reachable K. Hence, D_{max} can be regarded as the decisive parameter of our method.

The results for the larger inverse temperature $\beta = 1$ paint a slightly different picture. While the overall behavior of our method remains the same and Figure 2a depicts good convergence especially for $L < 100$, Figure 2b shows that the relative error achieved is noticeably worse than for the case of $\beta = 0.1$. It also seems that larger val-



(a) Entropy over K.



(b) Relative error in the entropy over K.

FIG. 2: Convergence behavior of the algorithm for $L \in \{10, 20, 30, 50, 100\}$, $\beta = 1$ and varying Krylov-dimension K. In (a), the convergence of the approximation is depicted. In (b), the convergence of the relative error is shown.

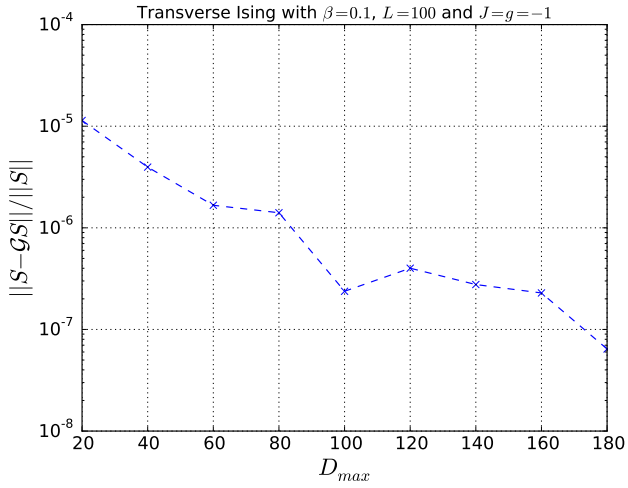


FIG. 3: Relative error in the entropy for $L = 100$ and $\beta = 0.1$ over the maximal bond dimension D_{max}.

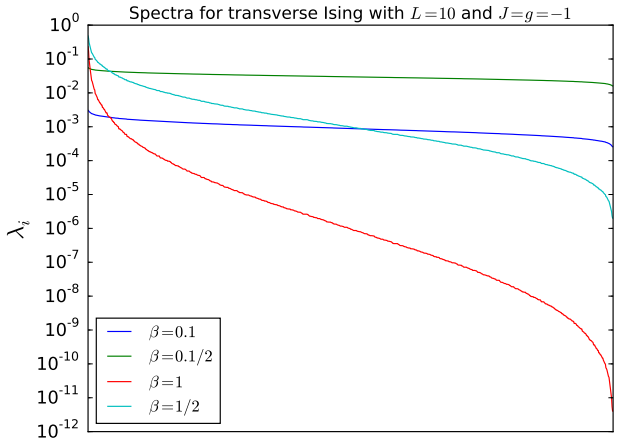


FIG. 4: The spectra of $\rho(\beta)$ and $\rho(\frac{\beta}{2})$ for $L = 10$.

ues of D_{max} are required to achieve reasonable results. This phenomenon naturally becomes more pronounced with larger L.

We conjecture that the difference in the performance observed for the two considered values of β has two main reasons. Firstly, the bond dimension required for a good approximation of ρ grows with larger β. This might in turn increase the value of D_{max} required for good accuracy, and, correspondingly, increase the approximation error incurred by ρ, so that the computed function will be farther from the analytical solution. Secondly, the spectral properties of the obtained MPOs for the two considered cases are significantly different. In Figure 4, we show the spectra of ρ for both values of β and $\beta/2$ for $L = 10$, respectively. It is clearly visible that $\beta = 0.1$ poses a much more benign case. This is underlined by the condition numbers, which are roughly 11.9, $5.68 \cdot 10^{10}$, 3.5 and $2.38 \cdot 10^5$ for $\beta = 0.1$, $\beta = 1$, $\beta = 0.05$ and $\beta = 0.5$, respectively. They show that $\beta = 1$ in fact yields highly ill-conditioned MPOs, functions of which are hard to approximate. These considerations also make it clear that by absorbing the necessary squaring of the eigenvalues into the function, we obtain much more well-conditioned input MPOs of lower bond dimension. Hence, we can conclude that our method, while being influenced by both aforementioned factors, is relatively robust and even works reasonably well for very difficult cases.

We do not provide a comparison to other methods at this point, because of the simple reason that to the best

of the authors knowledge there is no other algorithm that can solve the considered kind of problem for $L \gg 20$. For $L \leq 20$ we however expect the existing highly optimized methods to outperform our method in terms of runtime.

VII. DISCUSSION

In this work, we have introduced a method to approximate functionals of the form $\mathsf{Tr} f(A)$ for matrices of dimension much larger than 2^{20}. We started by giving an overview over the mathematical and algorithmic ideas behind the method. Following this, a detailed description of the algorithm together with an analysis of its complexity was provided. We then presented numerical results for a challenging problem in quantum many body physics. These results indicate that our method is able to produce good approximations for a number of Krylov steps and a maximal bond dimension logarithmic in the size of the matrix as long as the matrix exhibits some structure that can be expressed well in the MPO/MPS-formalism and is moderately well-conditioned. It was also shown that the maximal allowed bond dimension is the decisive parameter of the algorithm.

There are several ways to build upon this work. Firstly, an investigation of preconditioning methods suitable for our method could be fruitful. Secondly, a more thorough analysis of the effect of the approximation error introduced by the tensor network formalism on the approximation error of the Gauss quadrature would be an interesting addition. Thirdly, the connection of the approximability of a matrix by an MPO to the convergence behavior of our method could provide deeper understanding. Fourthly, it could be investigated which of the many improvements over the normal Gauss quadrature, as for instance [RST15], can be incorporated into our algorithm to make better use of the expensive information obtained in the Krylov iteration. Finally, the method naturally could be applied to solve practical problems of interest.

While our method was tested for a quantum mechanical problem, it is of course general in nature and can be applied to any case where the matrix in question can be formulated as an MPO or well approximated by one. Especially for matrices of dimension larger than 2^{10} that however can still be explicitly stored, it might be interesting to consider computing the desired function for the MPO-representation.

ACKNOWLEDGEMENTS

This work was partly funded by the *Elite Network of Bavaria*(ENB) via the doctoral programme *Exploring Quantum Matter*.

[Arn51] W. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of applied mathematics*, 9(1):17–29, 1951.

[BFG96] Z. Bai, G. Fahey, and G. Golub. Some large-scale matrix computation problems. *Journal of Computational and Applied Mathematics*, 74(1):71–89, 1996.

[BFRR14] J. Baglama, C. Fenu, L. Reichel, and G. Rodriguez. Analysis of directed networks via partial singular value decomposition and gauss quadrature. *Linear Algebra and its Applications*, 456:93–121, 2014.

[BKS07] C. Bekas, E. Kokiopoulou, and Y. Saad. An estimator for the diagonal of a matrix. *Applied numerical mathematics*, 57(11):1214–1229, 2007.

[BRRS15] M. Bellalij, L. Reichel, G. Rodriguez, and H. Sadok. Bounding matrix functionals via partial global block lanczos decomposition. *Applied Numerical Mathematics*, 94:127–139, 2015.

[BSU16] M. Bachmayr, R. Schneider, and A. Uschmajew. Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. *Foundations of Computational Mathematics*, pages 1–50, 2016.

[CD74] J. Cullum and W. Donath. A block lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large, sparse, real symmetric matrices. In *Decision and Control including the 13th Symposium on Adaptive Processes, 1974 IEEE Conference on*, pages 505–509. IEEE, 1974.

[CRS94] D. Calvetti, L. Reichel, and D. Sorensen. An implicitly restarted lanczos method for large symmetric eigenvalue problems. *Electronic Transactions on Numerical Analysis*, 2(1):21, 1994.

[DR07] P. Davis and P. Rabinowitz. *Methods of numerical integration*. Courier Corporation, 2007.

[Dru08] V. Druskin. On monotonicity of the lanczos approximation to the matrix exponential. *Linear Algebra and its Applications*, 429(7):1679–1683, 2008.

[EH10] E. Estrada and D. Higham. Network properties revealed through matrix functions. *SIAM review*, 52(4):696–714, 2010.

[EMS09] L. Elbouyahyaoui, A. Messaoudi, and H. Sadok. Algebraic properties of the block gmres and block arnoldi methods. *Electronic Transactions on Numerical Analysis*, 33(207-220):4, 2009.

[FMRR13] C. Fenu, D. Martin, L. Reichel, and G. Rodriguez. Block gauss and anti-gauss quadrature with application to networks. *SIAM Journal on Matrix Analysis and Applications*, 34(4):1655–1684, 2013.

[FNW92] Mark Fannes, Bruno Nachtergaele, and Reinhard F Werner. Finitely correlated states on quantum spin chains. *Communications in mathematical physics*, 144(3):443–490, 1992.

[GKT13] L. Grasedyck, D. Kressner, and C. Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

[GLO81] G. Golub, F. Luk, and M. Overton. A block lanczos method for computing the singular values and corresponding singular vectors of a matrix. *ACM Transactions on Mathematical Software (TOMS)*, 7(2):149–169, 1981.

[GLS94] R. Grimes, J. Lewis, and H. Simon. A shifted block lanczos algorithm for solving sparse symmetric generalized eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, 15(1):228–272, 1994.

[GM94] G. Golub and G. Meurant. Matrices, moments and quadrature. *PITMAN RESEARCH NOTES IN MATHEMATICS SERIES*, pages 105–105, 1994.

[GM09] G. Golub and G. Meurant. *Matrices, moments and quadrature with applications*. Princeton University Press, 2009.

[GOS12] S. Goreinov, I. Oseledets, and D. Savostyanov. Wedderburn rank reduction and krylov subspace method for tensor approximation. part 1: Tucker case. *SIAM Journal on Scientific Computing*, 34(1):A1–A27, 2012.

[GR06] J. J. García-Ripoll. Time evolution of matrix product states. *New Journal of Physics*, 8(12):305, 2006.

[Has07] M. Hastings. An area law for one-dimensional quantum systems. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(08):P08024, 2007.

[Hut90] M. Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

[HWSH13] T. Huckle, K. Waldherr, and T. Schulte-Herbrüggen. Computations in quantum tensor networks. *Linear Algebra and its Applications*, 438(2):750–781, 2013.

[JMS99] K. Jbilou, A. Messaoudi, and H. Sadok. Global fom and gmres algorithms for matrix equations. *Applied Numerical Mathematics*, 31(1):49–63, 1999.

[Kar06] D. Karevski. Ising quantum chains. *arXiv preprint cond-mat/0611327*, 2006.

[Kho11] B. Khoromskij. O (dlog n)-quantics approximation of nd tensors in high-dimensional numerical modeling. *Constructive Approximation*, 34(2):257–280, 2011.

[Kry31] A. Krylov. On the numerical solution of the equation by which the frequency of small oscillations is determined in technical problems. *Izv. Akad. Nauk SSSR Ser. Fiz.-Mat*, 4:491–539, 1931.

[KT11] D. Kressner and C. Tobler. Low-rank tensor krylov subspace methods for parametrized linear systems. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1288–1316, 2011.

[Lan50] C. Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.

[Mar12] D. Martin. *Quadrature approximation of matrix functions, with applications*. PhD thesis, Kent State University, 2012.

[Mey98] C. Meyer. The idea behind krylov methods. *The American mathematical monthly*, 105(10):889–899, 1998.

[Mon95] P. Montgomery. A block lanczos algorithm for finding dependencies over gf (2). In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 106–120. Springer, 1995.

[New10] M. Newman. *Networks: an introduction*. Oxford university press, 2010.

[Nota] ρ is a positive operator with unit trace, representing an ensemble of pure states.

[Notb] Note that β is not related to the β_i computed by our algorithm.

[Ose11] I. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[PGVWC06] D. Perez-Garcia, F. Verstraete, M. Wolf, and I. Cirac. Matrix product state representations. *arXiv preprint quant-ph/0608197*, 2006.

[PMCV10] B. Pirvu, V. Murg, I. Cirac, and F. Verstraete. Matrix product operator representations. *New Journal of Physics*, 12(2):025012, 2010.

[RST15] L. Reichel, M. Spalević, and T. Tang. Generalized averaged gauss quadrature rules for the approximation of matrix functionals. *BIT Numerical Mathematics*, pages 1–23, 2015.

[Saa92] Y. Saad. *Numerical methods for large eigenvalue problems*, volume 158. SIAM, 1992.

[Sac07] S. Sachdev. *Quantum phase transitions*. Wiley Online Library, 2007.

[Sch11] U. Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 2011.

[SIC12] S. Suzuki, J. Inoue, and B. Chakrabarti. *Quantum Ising phases and transitions in transverse Ising models*, volume 862. Springer, 2012.

[SS86] Y. Saad and M. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[TBI97] L. Trefethen and D. Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[TS12] J. Tang and Y. Saad. A probing method for computing the diagonal of a matrix inverse. *Numerical Linear Algebra with Applications*, 19(3):485–501, 2012.

[VC06] F. Verstraete and I. Cirac. Matrix product states represent ground states faithfully. *Physical Review B*, 73(9):094423, 2006.

[VGRC04] F. Verstraete, J. Garcia-Ripoll, and I. Cirac. Matrix product density operators: simulation of finite-temperature and dissipative systems. *Physical review letters*, 93(20):207204, 2004.

[Vid03] G. Vidal. Efficient classical simulation of slightly entangled quantum computations. *Physical Review Letters*, 91(14):147902, 2003.

[Vid04] G. Vidal. Efficient simulation of one-dimensional quantum many-body systems. *Physical review letters*, 93(4):040502, 2004.

[VMC08] F. Verstraete, V. Murg, and I. Cirac. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics*, 57(2):143–224, 2008.

[VPC04] F. Verstraete, D. Porras, and I. Cirac. Density matrix renormalization group and periodic boundary conditions: a quantum information perspective. *Physical review letters*, 93(22):227205, 2004.

[Wal14] K. Waldherr. *Numerical Linear and Multilinear Algebra in Quantum Control and Quantum Tensor Networks*. Verlag Dr. Hut, 2014.

[ZV04] M. Zwolak and G. Vidal. Mixed-state dynamics in one-dimensional quantum lattice systems: a time-dependent superoperator renormalization algorithm. *Physical review letters*, 93(20):207205, 2004.