Approximating Functions of MPOs

Moritz August\textsuperscript{1}, joint work with M. C. Bañuls\textsuperscript{2} and T. Huckle\textsuperscript{1}
Technical University of Munich \textsuperscript{1}
Max Planck Institute of Quantum Optics \textsuperscript{2}
Condensed Matter Theory Seminar
Outline

Motivation

A bit of Theory

Global Lanczos

Gauss Quadrature

The Algorithm

Numerical Experiments

Approximating the Entropy

A Note on the Runtime

Summary & Outlook
Motivation I

\[ H = \frac{1}{4N} \left( \sum_i \sigma_x^{[i]} \right)^2 + \frac{\hbar}{2} \sum_i \sigma_z^{[i]} \]

There are some interesting functions in quantum physics that look like \( \text{Tr} f(A) \), for example

- Entropy: \( -\text{Tr} A \log A \)
- Trace Norm: \( \text{Tr} \sqrt{A^\dagger A} \)
- Expectation values: \( \text{Tr} AO \) w.r.t. given \( O \)

While \( \text{Tr} A \) is easy for MPOs, \( \text{Tr} f(A) \) is not.

Functions of this form are also used in computer science and engineering to compute some properties of graphs

- Network Topology
- Social Networks
- Molecules

Of course, computing these things functions is easy for small matrices.
Motivation II

What to do when matrices get large, i.e., \( \gg 2^{10} \) dimensions?

- Diagonalization not possible anymore
- Approximation algorithms based on Monte Carlo and Krylov methods
- \( \text{Tr} f(A) \approx \sum_i u_i^\dagger f(A) u_i \), \( u_i \) starting vector for Krylov method
- Much work on this by Bellalij, Reichel, Golub, Saad, etc.
- \( f \) is always assumed to be analytic and smooth

These methods can be adapted to the tensor network world, but they don’t work.

- Variance of the estimator decreases too slowly
- Too many samples needed (no logarithmic scaling in dimensions)
- Added approximation error by the tensor networks representation

Still, computing/approximating these functions would be nice. Can we maybe avoid sampling?
Outline

Motivation

A bit of Theory

Global Lanczos

Gauss Quadrature

The Algorithm

Numerical Experiments

Approximating the Entropy

A Note on the Runtime

Summary & Outlook
Global Lanczos I

A generalization of the normal Lanczos method to the case of (Hermitian) matrices.

• Starting with $U \in \mathbb{C}^{N \times M}$, builds basis $U_K = [U_1, \cdots, U_K] \in \mathbb{C}^{N \times KM}$ of
  $\mathcal{K} \mathcal{R}_K = \text{span}\{ U, AU, A^2 U, \ldots, A^{K-1} U \}$

• Makes use of the Frobenius inner product defined as $\langle U, V \rangle_F = \text{Tr} U^\dagger V$

• Induces the Frobenius norm $\| U \|_F = \langle U, U \rangle^{1/2}$

• Only complete matrices are orthogonalized against each other

• All important results carry over from the standard Lanczos algorithm

• The algorithm produces a tridiagonal matrix

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

Just like the Lanczos algorithm, only with matrices

**Algorithm 1**: Global Lanczos Algorithm

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

1. $U_0 \leftarrow 0$;
2. $V_0 \leftarrow U$;
3. for $i \leftarrow 1$; $i \leq K$ do
   4. $\beta_i \leftarrow \|V_{i-1}\|_F$;
   5. if $\beta_i = 0$ then
      6. break ;
   7. end
   8. $U_i \leftarrow V_{i-1}/\beta_i$;
   9. $V_i \leftarrow AU_i - \beta_i U_{i-1}$ ;
10. $\alpha_i \leftarrow \langle U_i, V_i \rangle$ ;
11. $V_i \leftarrow V_i - \alpha_i U_i$ ;
12. end

**Output**: Orthonormal Basis $U_K \in \mathbb{C}^{N \times KM}$, Tridiagonal Matrix $T_K \in \mathbb{R}^{KM \times KM}$
Gauss Quadrature I

We now turn to the seemingly unrelated Gauss quadrature

It is well known that we can write

\[ \mathcal{I} f := \text{Tr}(f(A)) \approx \text{Tr}(U^\dagger f(A) U) = \int_a^b f(\lambda) d\mu(\lambda) \]

- Eigenvalues \( \lambda_1 = a < \lambda_2 < \cdots < \lambda_N = b \)
- \( \mu(\lambda) \) piecewise constant distribution function with jumps at \( \lambda_i \)
- \( U \) is a suitable submatrix of some unitary matrix
- The larger \( U \) is, the better

We can approximate this integral via Gauss quadrature (and some of its extensions)
Gauss Quadrature II

We need to find $\theta_i$, $\omega_i$ such that

$$G_K f := \sum_{i=1}^{K} \omega_i f(\theta_i) \approx \mathcal{I} f$$

- Solution is to construct polynomials $\{p_0, \cdots, p_K\}$ orthonormal w.r.t. $\mu(\lambda)$
- Roots of $p_K$ are the optimal $\theta_i$
- Polynomials follow recurrence relation that yields a recurrence matrix

$$R_K = \begin{bmatrix}
\alpha_1 & \beta_2 & 0 \\
\beta_2 & \alpha_2 & \cdots \\
\cdots & \cdots & \beta_K \\
0 & \beta_K & \alpha_K
\end{bmatrix},$$

- Eigenvalues of $R_K$ are roots of $p_k$, $\omega_i$ are squared first elements of Eigenvectors
- It can be shown that $U_i$ from the Lanczos method are such polynomials
- It follows $\text{Tr} f(A) \approx \text{Tr}(U^\dagger f(A)U) = \int_a^b f(\lambda) d\mu(\lambda) \approx e_1^T f(T_K)e_1$
Outline

Motivation

A bit of Theory

Global Lanczos

Gauss Quadrature

The Algorithm

Numerical Experiments

Approximating the Entropy

A Note on the Runtime

Summary & Outlook
The Algorithm I

The whole global Lanczos method is based on matrix addition, matrix-matrix and scalar-matrix multiplication. So, we can rewrite it for MPOs.

- $\langle U, V \rangle_F = \text{Tr} U^* V$ is just the inner product of vectorized MPOs, i.e., MPS
- $V/\beta_i$ corresponds to scaling one site of MPO $V$
- $AU$ can be rewritten as $(I \otimes A) U_{\text{vec}}$ such that $(I \otimes A)$ is an MPO and $U_{\text{vec}}$ is an MPS
- So we can use standard techniques to find a good representation for given bond dimension
- $\alpha A + \beta U$ can also be tackled by turning $A$ and $U$ into MPS
- We also know how to optimize this for given bond dimension

Great, we know we can use the algorithm for MPOs. But why should this perform any better than previous approaches?
The Algorithm II

Because now we do not need to sample anymore!

Working with MPOs, we can afford a unitary $U$ of size equal to $A$, such that

$$\mathcal{I} f = \text{Tr}(f(A)) = \text{Tr}(U^\dagger f(A)U) \approx e_1^T f(T_K)e_1 = \mathcal{G} f.$$  

Trivial choice: Identity MPO with $A_{i,j}^{i,k} = \delta_{j,k}$

But we introduce a new source of error via the bond dimension:

- Naive Multiplication/Addition causes $D$ to grow exponentially/linearly
- Main challenge is to find best way to keep $D$ within reasonable bounds
- Approach taken here: Perform operation and truncation simultaneously
- We force $D \leq D_{\text{max}}$ and build the Krylov basis up to dimension $K$
- $D_{\text{max}}$ and $K$ are the two parameters controlling the accuracy of the approximation

Connection to Gauss quadrature provides useful interpretation in terms of bounds.
The Algorithm III

**Algorithm 2: Approximation Algorithm**

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

1. $U_0 \leftarrow 0$
2. $V_0 \leftarrow U$
3. $D \leftarrow D_{\text{init}}$
4. for $i \leftarrow 1; i \leq K$ do
   5. $\beta_i \leftarrow \sqrt{\text{contract}(V_{i-1}, V_{i-1})}$
   6. if $\beta_i = 0$ then
      7. break
   end
   8. $U_i \leftarrow \text{multiplyScalar}(1/\beta_i, V_{i-1})$
   9. $D \leftarrow \min(D_{\text{max}}, D \cdot D_A)$
   10. $V_i \leftarrow \text{multiplyAndOptimize}(A, U_i, D)$
    11. $D \leftarrow \min(D_{\text{max}}, D + D_{U_{i-1}})$
    12. $V_i \leftarrow \text{sumAndOptimize}(V_i, -\beta_i U_{i-1}, D)$
    13. $\alpha_i \leftarrow \text{contract}(U_i, V_i)$
    14. $D \leftarrow \min(D_{\text{max}}, D + D_{U_i})$
    15. $V_i \leftarrow \text{sumAndOptimize}(V_i, -\alpha_i U_i, D)$
    16. $V \Lambda V^* \leftarrow \text{spectralDecomposition}(T_i)$
    17. $\mathcal{G} f \leftarrow \beta_i^2 e_1^T V f(\Lambda) V^* e_1$
    18. if checkStop($\mathcal{G} f, \Lambda, \mathcal{S}$) then
       19. break
    end
5. end
6. end

**Output**: Approximation $\mathcal{G} f$ of $\text{Trf}(A)$
The Algorithm IV

To be computationally feasible, the algorithm must be in $\mathcal{O}(\text{poly}(L))$, $L$ being number of sites

<table>
<thead>
<tr>
<th>Operation</th>
<th>$\mathcal{O}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>contract</td>
<td>$\mathcal{O}(L D_{\text{max}}^3 d^2)$</td>
</tr>
<tr>
<td>multiplyAndOptimize</td>
<td>$\mathcal{O}(L D_{\text{max}}^3 D_A d^2)$</td>
</tr>
<tr>
<td>sumAndOptimize</td>
<td>$\mathcal{O}(L D_{\text{max}}^3 d^2)$</td>
</tr>
<tr>
<td>multiplyScalar</td>
<td>$\mathcal{O}(D_{\text{max}}^2 d)$</td>
</tr>
<tr>
<td>spectralDecomposition</td>
<td>$\mathcal{O}(K^3)$</td>
</tr>
<tr>
<td>checkStop</td>
<td>$\mathcal{O}(1)$</td>
</tr>
</tbody>
</table>

We require $D_{\text{max}}, D_A, K \in \mathcal{O}(L)$, thus we find that the algorithm is indeed in $\mathcal{O}(KLD_{\text{max}}^3 D_A d^2) \subseteq \mathcal{O}(\text{poly}(L))$. 

M. Sc. Moritz August (TUM) | august@in.tum.de | ENB PhD-Programme 'Exploring Quantum Matter'
Outline

Motivation

A bit of Theory

Global Lanczos

Gauss Quadrature

The Algorithm

Numerical Experiments

Approximating the Entropy

A Note on the Runtime

Summary & Outlook
Numerical Experiments I

To illustrate some properties of the algorithm, we look at Gibbs states of the transverse Ising model.

So, we have

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

and

\[ A = \frac{e^{-\beta H}}{\text{Tr} e^{-\beta H}}, \beta \text{ being the inverse temperature.} \]

Good toy example because we have an analytic solution to compare against.

Let’s approximate the entropy \( S(A) = -\text{Tr} A \log A \). There are two possible ways to do that.

- Option 1: Obtain \( A \) from imaginary time evolution, approximate \( S(A) \)
- Option 2: Realize that \( S(A) \) is actually a function of \( H \)
Numerical Experiments II

It’s much nicer if we can compute a function of $H$ rather than $A$

- $H$ has typically very low bond dimension and is exact
- We can do the imaginary time evolution exactly for $T_K$
- But now $f(A)$ involves itself a trace, however

$$-\text{Tr} A \log A = -\text{Tr} \frac{e^{-\beta H}}{\text{Tr} e^{-\beta H}} \log \frac{e^{-\beta H}}{\text{Tr} e^{-\beta H}} = \frac{\text{Tr} e^{-\beta H \beta H}}{\text{Tr} e^{-\beta H}} + \log \text{Tr} e^{-\beta H}$$

- We approximate $\text{Tr} e^{-\beta H \beta H}$ and $\text{Tr} e^{-\beta H}$ in one run and combine the results

Absorbing the imaginary time evolution can be applied to other functions and Hamiltonians, too.

All following results were computed on an Intel i7-4790 @ 3.6 GHz x 8 with 32 GB DDR3 RAM. Code was compiled with the Intel Compiler, using Intel MKL and PRIMME
Relative Error over $K$ for $\beta = 0.1$
Relative Error over $K$ for $\beta = 0.5$
Relative Error over $D_{\text{max}}$ for $\beta \in \{0.1, 0.5\}$ and $L = 50$
$G$ over $L$
Spectra of Gibbs states for $\beta \in \{0.1, 0.5\}$ and $L = 10$
Avg. runtime of single iteration over $D_{max}$ for $L = 50$
Avg. runtime of single iteration over $L$ for $D_{\text{max}} = 30$
Outline

Motivation

A bit of Theory

Global Lanczos

Gauss Quadrature

The Algorithm

Numerical Experiments

Approximating the Entropy

A Note on the Runtime

Summary & Outlook
Summary

A brief listing of the most important points of the talk

- Algorithm is based on global Lanczos method and Gauss quadrature
- Reformulation for MPOs does away with sampling
- Can be applied to arbitrary Input-MPOs
- Runtime is polynomial in $L$
- Imaginary time evolution can be absorbed into approximated function
- Good results already for small $D_{max}$ and $K$
- Approximation accuracy depends on spectral properties
- Can be run on single workstation, cluster helpful for large systems
Conclusions and Future Work

A few things I would like to do next

• Try to analyze propagation of truncation errors better
• Understand connection between condition number and bond dimension better
• Try to improve performance for very large $L$ and $D$
• Are there specific classes of inputs for which it works/fails?
• Find more interesting functions to approximate

If you have any questions/suggestions please feel free to ask or have a look at arXiv:1610.06086

Thank you for your attention!
Literature I

Some large-scale matrix computation problems.

Bounding matrix functionals via partial global block lanczos decomposition.

M. Bachmayr, R. Schneider, and A. Uschmajew.
Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations.

V. Druskin.
On monotonicity of the lanczos approximation to the matrix exponential.

L. Elbouyahyaoui, A. Messaoudi, and H. Sadok.
Algebraic properties of the block gmres and block arnoldi methods.

L. Grasedyck, D. Kressner, and C. Tobler.
A literature survey of low-rank tensor approximation techniques.
Literature II

J. J. García-Ripoll.
Time evolution of matrix product states.

K. Jbilou, A. Messaoudi, and H. Sadok.
Global fom and gmres algorithms for matrix equations.

C. Meyer.
The idea behind krylov methods.

I. Oseledets.
Tensor-train decomposition.

U. Schollwöck.
The density-matrix renormalization group in the age of matrix product states.

F. Verstraete, V. Murg, and I. Cirac.
Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems.