An Introduction to Matrix Product State Algorithms

Nicholas Woodford
Supervisor: Takis Angelides

Abstract

Tensor Networks were developed in Many-Body Quantum Physics for describing many-body wave-functions. Matrix Product States are a simple 1D ansatz which lends itself to efficient computation and as such have been explored for many applications in MBQP and beyond. I give an overview of some of the most common MPS algorithms and their uses: the Tensor Cross Interpolation for efficiently finding MPS representations of functions; Density Matrix Renormalisation Group for finding the ground state of Hamiltonians; and Time Evolving Block Decimation for time evolution. Uses of MPS for manipulating functions are presented as well as applications to the Ising model and for simulating quantum electrodynamics with the Schwinger model.

September, 2025

Contents

1 Introduction

The general quantum state of a system of N qubits (two level systems) can be written as:

$$|\psi\rangle = \sum_{\sigma_0\sigma_1...\sigma_{N-1}} \psi_{\sigma_0\sigma_1...\sigma_{N-1}} |\sigma_0\sigma_1...\sigma_{N-1}\rangle$$

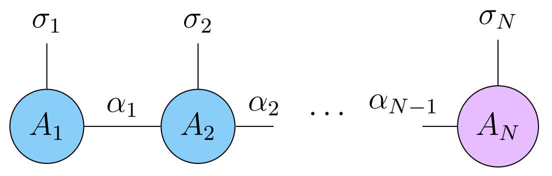
The Hilbert space of such a system has dimensionality 2^N. Motivated by the exponential scaling of the Hilbert space, Tensor Network techniques originated in Quantum Many Body Physics as a technique for working with large quantum systems by decomposing this large Hilbert space into many smaller tensors for more efficient storage and manipulation. More recently, these techniques have been developed for use outside of quantum many body physics and for applications in areas such as quantum information, machine learning and fluid dynamics[1].

In this report we will look at three algorithms which utilise the MPS ansatz and some of their applications. Firstly, the Tensor Cross Interpolation algorithm which aims to efficiently learn the MPS representation of a tensor [2]. Secondly, the Density Matrix Renormalisation Algorithm which approximates the ground state and ground state energy of a Hamiltonian expressed as a MPO through a variational approach [3]. And finally, the Time Evolving Block Decimation algorithm which time evolves an MPS state according to a Hamiltonian expressed as an MPO [4].

2 Tensor Networks

2.1 Matrix Product States and Operators

A matrix product state (MPS) is a one dimensional tensor network. MPS take a tensor of N dimensions each taking values between 1 and d and represent it as a product of tensors each with a physical index of dimension d and link indices between tensors with bond dimensions D_i. Assuming all link indices are the same, decomposing a tensor into a string of N tensor, all with bond dimensions D, takes NdD^2 points. It is simple to see how this can represent for example a 1D chain of spin-1/2 particles with physical indices of dimension 2 and link indices representing the entanglement between sites.



$$= A_{\sigma_0\alpha_0}A_{\sigma_1\alpha_0\alpha 1}...A_{\sigma_{N-1}\alpha_{N-2}}$$

By using an arbitrarily large bond dimension, an MPS can in principle perfectly represent any tensor, however the bond dimension will scale exponentially for a given N. But by limiting the bond dimensions at the cost of some error, we can reduce the size of the MPS.

Compressible, or low-rank, tensors can be approximated with small D below an error threshold [2]. Physically relevant tensors often have a structure which lends to this compression, making the MPS form an effective one for representing and manipulating these tensors.

The bond dimension of an MPS is closely linked to the entanglement entropy of that system it is representing. If we split the system into two subsystems of the site up to and including site i and those from site $i+1$ to N we can expressed the state in the basis of $|I\rangle$ and $|J\rangle$ with a matrix $\psi_{I,J}$ storing the coefficients. By performing a single value decomposition on this matrix, the system can be expressed in a new basis where the matrix is diagonal with entries λ_a. The entanglement entropy is defined as $S = -\mathrm{Tr}_L(\rho_l \mathrm{ln} \rho_L)$ where ρ_L is the reduced density matrix of the left part of the system. Therefore, in this basis where the state is diagonal:

$$S = -\sum_a |\lambda_a|^2 \mathrm{ln}(|\lambda_a|^2) \tag{1}$$

The entanglement entropy is at a maximum for an equal superposition state where all λ_a are equal to $1/\sqrt{D}$ by the normalisation constraint. This places an upper bound on the entanglement entropy which can be expressed by a given bond dimension [5]:

$$S \leq \mathrm{ln} D \tag{2}$$

Systems that lend themselves to efficient (compressed) MPS representations are therefore ones with low entanglement entropy - such as local or quasilocal systems - so that D can remain small while still capturing most of the entanglement information.

A Matrix Product Operator (MPO) acts on a MPS to create a new MPS. Therefore each tensor has two site indices and the operator can be written as

$$\hat{O} = \sum_{\sigma\sigma'\alpha} W^{\sigma_0'}_{\sigma_0\alpha_0} W^{\sigma_1'}_{\sigma_1\alpha_0\alpha_1} ... W^{\sigma_{N-1}'}_{\sigma_{N-1}\alpha_{N-2}} \left|\sigma_0'...\sigma_{N-1}'\right\rangle \left\langle\sigma_0'...\sigma_{N-1}'\right| \tag{3}$$

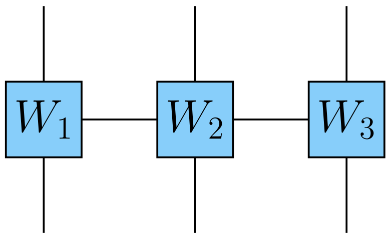


Figure 1: A Matrix Product Operator

It is simple to turn a Hamiltonian of Pauli operators into an MPO, with physical indices of each tensor representing the indices of the Paulis operating on that site. These operators are then arranged in matrices for each site to reproduce the desired interactions of the Hamiltonian. Here ITensors [6] will be used to automate the process. Importantly, with local, for example nearest-neighbour, Hamiltonians, the bond dimension is constant and does not scale with the system size.

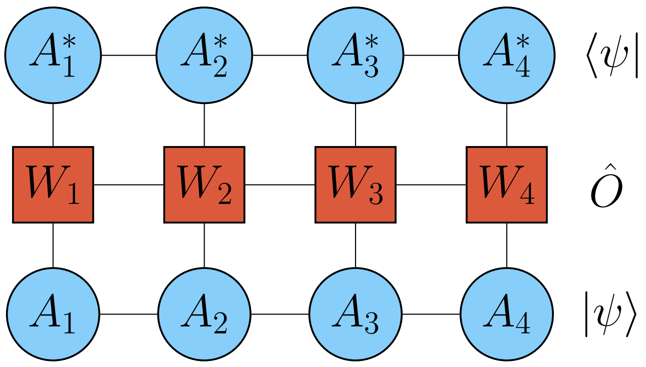


Figure 2: Calculating the expectation value $\langle\psi|\,\hat{O}\,|\psi\rangle$ of an MPO requires contracting it between the two MPS representing the bra and the ket of the state ψ

Consider the Transverse Field Ising Model Hamiltonian:

$$\hat{H} = J\sum_{i=0}^{N} Z_i Z_{i+1} + g_x \sum X_i \tag{4}$$

We have three operators which need to be arranged in matrices to create the tensors of the MPO: JZ, Z, $g_x X$ and I. These operators need to be arranged so that when the matrices are multiplied together they recreate the sums of the Hamiltonian. For example, at the first site we have only three operators that may be applied at the site which we arrange into the row vector, similarly at the final site we arrange them into a column vector:

$$W_1 = \begin{bmatrix} g_x X_1 & JZ_1 & I_1 \end{bmatrix} \quad , \quad W_N = \begin{bmatrix} I_N \\ Z_N \\ g_x X_N \end{bmatrix} \tag{5}$$

For the other sites, we arrange the operators so they combine correctly with others. There are four combinations again looking to the left, and three to the right. Looking leftwards, the necessary sums are $g_x X_{n-1} \to I_n$, $JZ_{n-1} \to Z_n$ and $I_{n-1} \to g_x X_n$ or JZ_n. To the right we must make the same mapping but from site n to $n+1$. The corresponding matrices are:

$$W_n = \begin{bmatrix} I_n & 0 & 0 \\ Z_n & 0 & 0 \\ g_x X_n & JZ_n & I_n \end{bmatrix} \tag{6}$$

2.2 Contracting two MPS

Contracting two MPS or an MPS with an MPO efficiently is very important for the efficiency of tensor network calculations. The order of tensor contractions must be chosen to minimise the size of any two tensors being contracted so that the computation cost does not grow. By starting at one end of the MPS, first contracting the vertical bond between the two MPS chains, then the two horizontal bonds to the next sites, a loop is formed which can contract the entire chain without growing the size of the tensors being contracted [3]. Using this algorithm makes contraction of two MPS with physical site dimension d, bond dimensions D and length L have a time complexity $O(dD^3L)$ and a memory cost of $O(D^2dL)$.

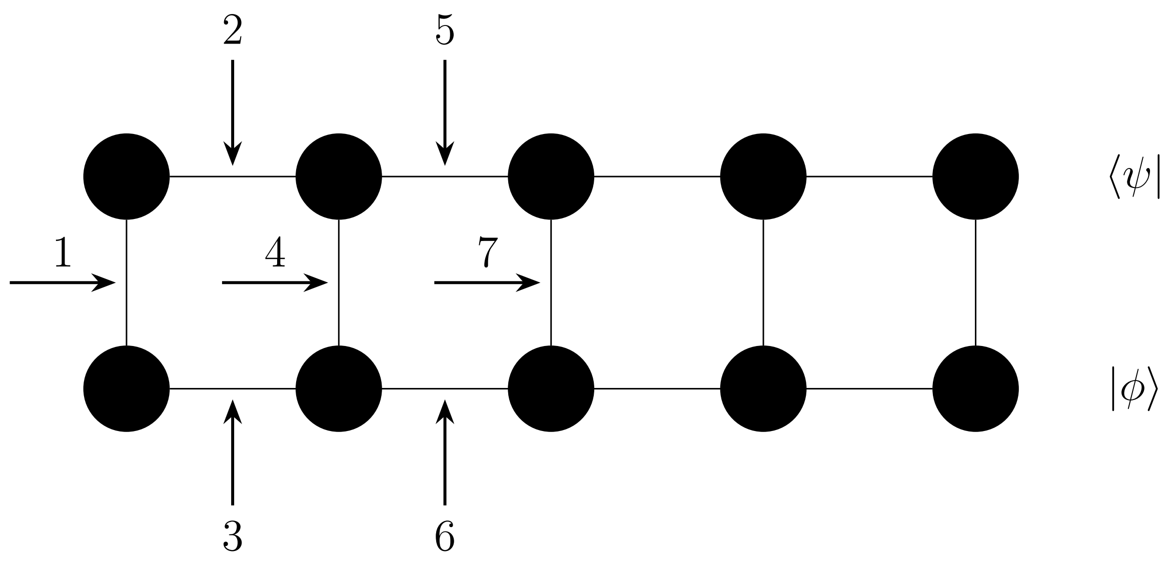


Figure 3: Ordering of bonds to contract for efficient contraction of two MPS as in [3]

2.3 Canonical Forms

As discussed in [3], tensors in an MPS are not unique and have a gauge freedom. For two adjacent matrices the MPS is unchanged under the transformation

$$M^{\sigma_i}M^{\sigma_{i+1}} \rightarrow (M^{\sigma_i}X)(X^{-1}M^{\sigma_{i+1}}) \tag{7}$$

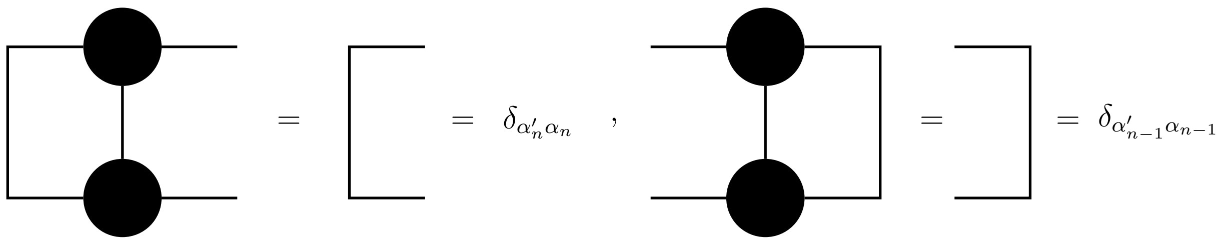
As long as X is invertible. This allows us to choose our tensors and three special cases exist which simplify calculations. An MPS tensor is said to be in left canonical form if it satisfies

$$A^*_{\sigma_n\alpha_{n-1}\alpha'_n}A_{\sigma_n\alpha_{n-1}\alpha'_n} = \delta_{\alpha'_n\alpha_n} \tag{8}$$

And in right canonical form if

$$A^*_{\sigma_n\alpha'_{n-1}\alpha_n}A_{\sigma_n\alpha_{n-1}\alpha_n} = \delta_{\alpha'_{n-1}\alpha_{n-1}} \tag{9}$$

MPO tensors can satisfy similar equations by summing over both physical indices and one link index. An MPS or MPO which has some sites in left canonical form and some in right canoncial form is said to be in mixed canonical form. This property is also know as orthogonality and if an MPS or MPO has tensors up to and including site n in left canonical form, or tensors from n to N in right canonical form then the site n is known as the orthogonality center.



$= \delta_{\alpha'_n \alpha_n}$,
$= \delta_{\alpha'_{n-1} \alpha_{n-1}}$

Figure 4: Left and right canonical form for MPS tensors

Starting at the leftmost (rightmost) site, performing an SVD and keeping only the unitary S (V) matrix on the site puts the first (last site in left (right) canonical form. The SV (US) matrices are absorbed into the next site where the process is repeated until the desired orthognality center is reached.

2.4 Quantics

A tensor is a useful structure when working with discretised points such as in condensed matter models where each dimensions can represent for example a spin site. But we can use tensor network tools for continuous functions if we discretise them onto a lattice using what is know as the quantics tensor representation [2]. For functions where a high resolutions is required, such as when there exists structures with different length scales, this can be useful. The function variables $\mathbf{x}$ are represented through binary (or more generally any base number system, where the base will be the physical dimensions of the tensors) digits $\boldsymbol{\sigma}$.

In one dimension, a variable is discretized onto a grid of 2^R points, the index $m = 1, ..., 2^R$ gives the site on the grid and therefore the value of the input variable to the function. With R bits, the index m is expressed as in [2]:

$$m(\sigma_1, ..., \sigma_R) = \sum_{r=1}^{R} \sigma_r 2^{R-r} \tag{10}$$

The vector $\boldsymbol{\sigma}$ allows the function $f(x)$ to be represented as a tensor $F_{\boldsymbol{\sigma}} = f(x(\boldsymbol{\sigma}))$ where the tensor has R indices of dimension $d = 2$.

When the input to the function has more than one dimension, a string of R bits is needed for each input dimension N, but the same process is used to create the tensor. The tensor $F_{\boldsymbol{\sigma}}$ can then be represented as a MPS of length $R \times N$. However if it is to be approximated by a tensor $\tilde{F}_{\boldsymbol{\sigma}}$, the error in the approximation for a given bond dimension can depend strongly on the choice of ordering of the bits. For example the interleaved quantics representation groups bits of the same scale close together in the MPS which allows smaller bond dimensions if the function has small interaction between length scales - such as in non-turbulent fluid flows. But if we have two variables x and y that are independent then having the first R bits represent x and the next R represent y can lead to a more compressible representation.

3 Tensor Cross Interpolation

TCI is an active learning algorithm which learns an efficient representation of the tensor. Rather than needing the entire large tensor as an input, it acts as function and is evaluated only for specific inputs when necessary. The algorithm aims to minimise calls to the tensor while constructing the MPS representation by learning 'pivots' which maximise the accuracy of the representation. Here an overview of the working of the algorithm are presented, more detailed explanations and proofs can be found in [7], [2] and [8]. The algorithm takes some function $f(x)$ as its input which can be viewed as a tensor $F_{\boldsymbol{\sigma}}$ using a quantics representation, the algorithm then creates an MPS state which aims to approximate the tensor:

$$F_{\boldsymbol{\sigma}} \approx \tilde{F}_{\boldsymbol{\sigma}} = \prod_{l=1}^{\mathcal{L}} M_l^{\sigma_l} \tag{11}$$

3.1 Matrix Cross Interpolation

Cross Interpolation aims to produce a factorisation of an $M \times N$ matrix A through a small number of queries to A. A truncated Single Value Decomposition, $A = USV^{\dagger}$, which keeps only the D largest singular values of S is optimal as an approximation but CI has error at most $O(D^2)$ the optimal error.

Let $\mathcal{I} = \{i_1, i_2, ...i_D\}$ (and $\mathcal{J} = \{j_1, j_2, ...j_D\}$) denote a list of D rows (columns) of A, and $\mathbb{I}$ and $\mathbb{J}$ be the list of all rows and columns in A. $A(\mathcal{I}, \mathcal{J})$ is the submatrix of A containing the rows $\mathcal{I}$ and columns $\mathcal{J}$. The cross interpolation formula is [7]:

$$A = A(\mathbb{I}, \mathbb{J}) \approx A(\mathbb{I}, \mathcal{J})A(\mathcal{I}, \mathcal{J})^{-1}A(\mathcal{I}, \mathbb{J}) \tag{12}$$

$A(\mathcal{I}, \mathcal{J})$ is known as the pivot matrix and its elements pivots where pivots are chosen to minimise the error of the interpolation. The interpolation is exact at the rows and columns included in the pivot matrix and is exact for the entire matrix if it has rank D where D is the number of rows (and columns) included in the interpolation - see [7] for proof.

Although useful, the CI decomposition can be unstable due to the need to invert the pivot matrix which can be unstable. Instead, below we use a Partial rank-revealing LU decomposition (prrLU or just LU) which is equivalent and more stable, see [2] for more detail. LU decomposes a matrix A into $P^{-1}LDUQ^{-1}$, where P and Q are permutation matrices for the rows and columns respectively, L (U) is lower (upper) triangular and D is diagonal. prrLU implements this by using Gaussian elimination, starting by moving the largest values to the first value in D and continuing to select largest values for D, whilst constructing P, L, U, Q, until the largest value is smaller than a tolerance or it has selected a pre-chosen maximum number of values. In this sense it is partially rank revealing as the full rank is not calculated but an approximate rank is found is values fall below the tolerance.

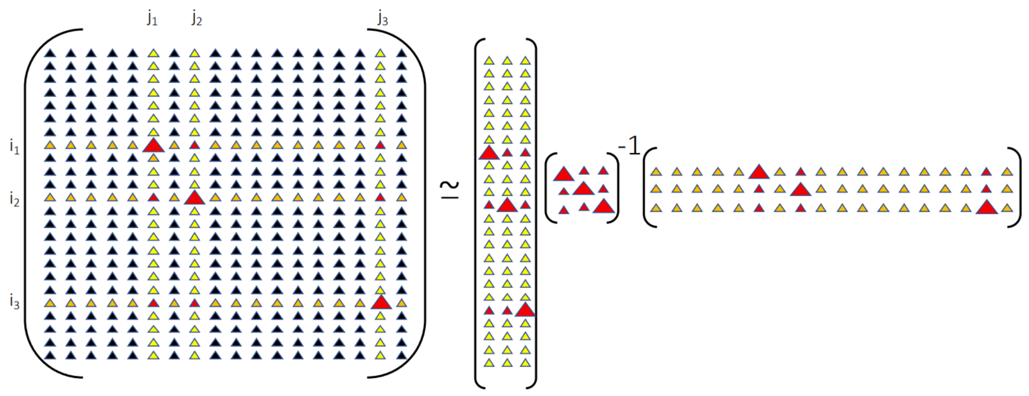


Figure 5: Diagram of the Matrix Cross Interpolation formula from [7]

3.2 The TCI algorithm

Throughout the TCI algorithm, the different indices must be kept track of, mainly the multi-indices which refer to values which the function is evaluated on rather than location in which the values sit in the tensors of the MPS. The following notations for doing this are taken from [2], the most important of which are the pivot lists $\mathcal{I}_l$ and $\mathcal{J}_l$.

We have an index σ_l ($l \in \{1, 2, ..., \mathcal{L}\}$) which takes d different values from a set $\mathbb{S}_l$, for example $\mathcal{L}$ qubit sites which take $d = 2$ values 0 or 1 meaning $\mathbb{S} = \{0, 1\}$. $\mathbb{I}_l = \mathbb{S}_1 \times ... \times \mathbb{S}_l$ is a set of row multi-indices up to site l, with elements $i \in \mathbb{I}_l$ is a row multi-index $i = (\sigma_1, ..., \sigma_l)$. Similarly $\mathbb{J}_l = \mathbb{S}_l \times ... \times \mathbb{S}_{\mathcal{L}}$ is a set of column multi-indices from site l upwards. $i_l \oplus j_{l+1} \equiv (\sigma_1, ..., \sigma_{\mathcal{L}})$ is the concatenation of complementary multi-indices to span the full configuration space.

For each l, we define a list of pivot rows $\mathcal{I}_l \in \mathbb{I}_l$ and pivot columns $\mathcal{J}_l \in \mathbb{J}_l$. These subsets of the full lists are generated initially by the algorithm from lists of random pivots $\boldsymbol{\sigma}$. Each pivot $\boldsymbol{\sigma}$ is split at each site l to produce the elements i_l and j_l of the lists $\mathcal{I}_l$ and $\mathcal{J}_l$.

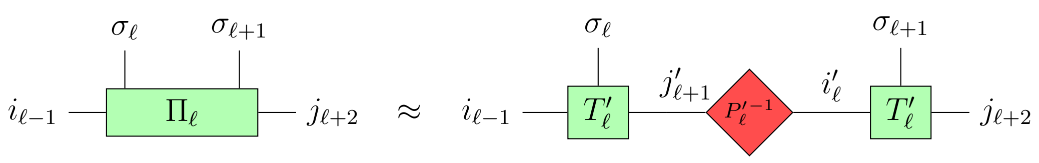
Once we have the initial pivot lists, the MPS can be constructed, this will then be improved iteratively to decrease the error of the approximation. To construct the initial MPS, we construct the pivot matrix $P_l = F(\mathcal{I}_l, \mathcal{J}_{l+1})$ and the tensor $T_l = F(\mathcal{I}_{l-1}, \mathbb{S}_l, \mathcal{J}_{l+1})$ by calling the function at specific inputs (for a system with $d = 2$, $\mathbb{S}_l = 0, 1$). For the P_l matrices a prrLU decomposition is performed where small pivots are removed to update the pivot list before P_l is reconstructed, this is to ensure that P_l is not singular and can be inverted. This is done at each site l and the initial MPS is defined as

$$F_{\boldsymbol{\sigma}} \approx \tilde{F}_{\boldsymbol{\sigma}} = T_1^{\sigma_1} P_1^{-1} ... T_l^{\sigma_l} P_l^{-1} T_{l+1}^{\sigma_{l+1}} ... P_{\mathcal{L}-1}^{-1} T_{\mathcal{L}}^{\sigma_{\mathcal{L}}} \tag{13}$$

We have decomposed the tensor at each site with the pivot matrix P_l. This is a tensor train where each T_l has a physical index but each P_l^{-1} only has link indices. Adjacent P_l^{-1} and T_l are contracted together to form an MPS.

Now an initial MPS has been created, the algorithm continues to improve the pivots by removing pivots which have little effect on the error to increase the compression of the MPS form. To improve the MPS, the algorithm sweeps back and forth, now performing a two-site TCI. Π_l tensors are constructed with two site indices: $\Pi_l = F(\mathcal{I}_{l-1}, \mathbb{S}_l, \mathbb{S}_{l+1}, \mathcal{J}_{l+2})$. The Π_l tensor is viewed as a matrix $F(I_{\ell-1} \times \mathbb{S}_\ell, \mathbb{S}_{\ell+1} \times J_{\ell+2})$ and again it is prrLU decomposed to remove pivots from the pivot lists for the next sweep. The dcomposed components are used to turn Π_l back into two tensors $T_l P_l^{-1}$ and T_{l+1}. Sweeping back and forth, pivots are slowly removed to reduce the bond dimensions. These bond dimensions are tracked and the algorithm is said to have converged when these do not change for a full sweep.

$$\big[\widetilde{\Pi}_\ell\big]_{i_{\ell-1}\,\sigma_\ell\,\sigma_{\ell+1}\,j_{\ell+2}} \approx \big[T'^{\sigma_\ell}_\ell\big]_{i_{\ell-1} j'_{\ell+1}} \big(P'_\ell\big)^{-1}_{j'_{\ell+1} i'_\ell} \big[T'^{\sigma_{\ell+1}}_\ell\big]_{i'_\ell j_{\ell+2}}$$



As long as the pivot lists are nested - discussion in [2] - the error from the prrLU decomposition is equal to the error of approximating $F_{\boldsymbol{\sigma}}$ with $\tilde{F}_{\boldsymbol{\sigma}}$. In the algorithm as presented here, error is controlled by the maximum bond dimensions and the cutoff value for the prrLU decompositions and pivots are only ever removed, never added. Different methods of choosing pivots can be implemented, such as schemes where pivots are added, or global pivots specified such as if $F_{\boldsymbol{\sigma}}$ is a sparse tensor and some non-zero values are given as initial pivots. The algorithm (2-site, full pivoting) has a cost of $O(D^3 d^2 \mathcal{L})$ and makes $O(D^2 d^2 \mathcal{L})$ calls to the function [2].

As can be seen in fig.6, even with a small number of sites and physical dimensions, TCI can effectively approximate highly oscillatory functions. For larger values of d and N, the quantics representation has a higher resolution but the compression of the MPS and the percentage of the domain that must be evaluated decreases making the TCI algorithm an effective method of producing very high resolution representations of functions.

3.3 Integration

Numerical integration using standard Monte-Carlo integration suffers from the numerical sign problem when attempting to approximate integrals of highly oscillating functions. When a function is highly oscillating the number of sampling points needed to improve the accuracy of the integral approximation scales poorly [7]. However using a quantics representation can gives very high resolution and when an efficient MPS form can be found this can be used to approximate integrals efficiently, for example for the integration of Feynman diagrams as explored in [7].

Starting with a multidimensional integral $\int_D d^N \mathbf{x} f(\mathbf{x})$, over a domain D, we can use quantics to express this as an integral over $N \times L$ integrals all over the domain $[0, 1]$. This integral is approximated as a Riemann sum across all the possible bit strings $\boldsymbol{\sigma}$. The second

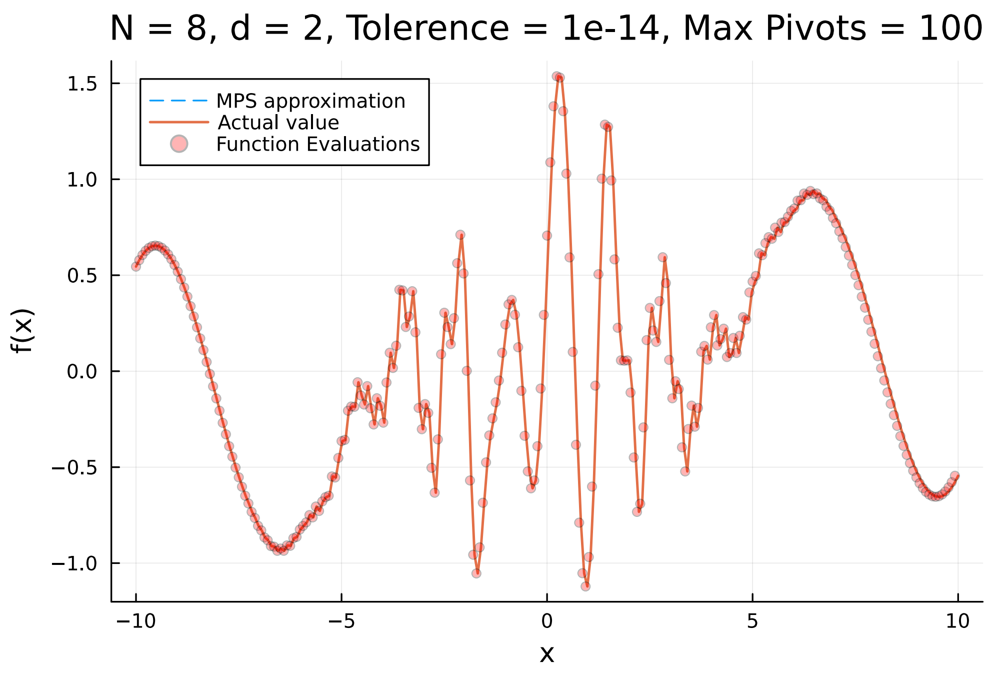
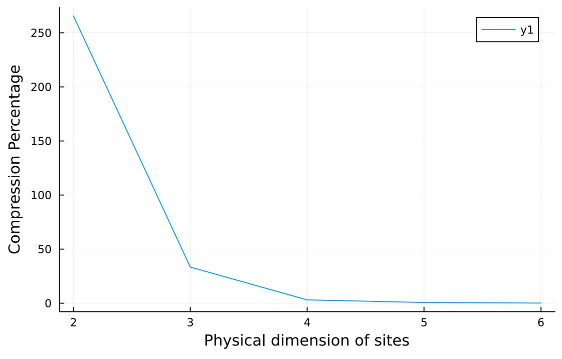
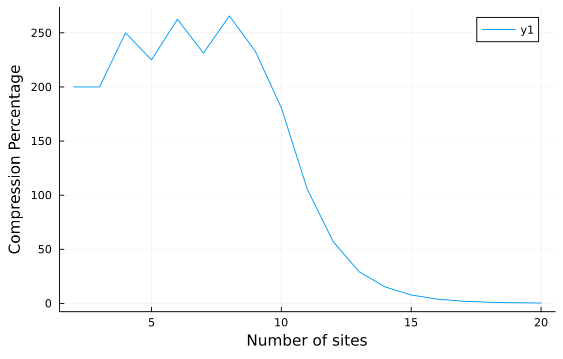


Figure 6: TCI approximation of $f(x) = \sin(5x) + 0.5\cos(3x^2)e^{-0.1x^2} + 0.2\arctan(x) + 0.1x\cos(x)$



(a)



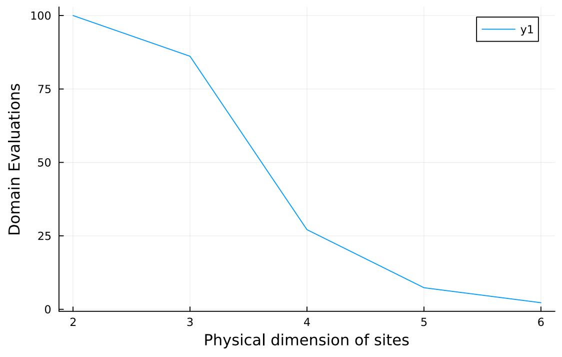
(b)

Figure 7: The size of the MPS representation $O(DdN)$ as a percentage of the total domain d^N

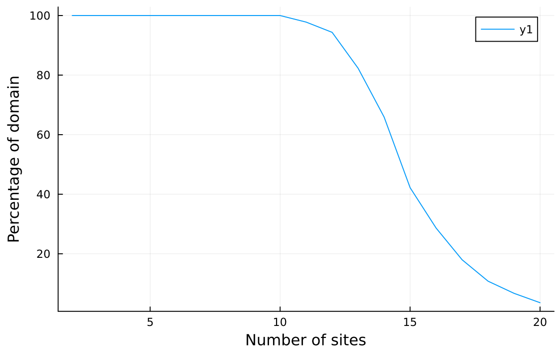
approximation is then viewing the function as a tensor and using the TCI algorithm to approximate it with an MPS, $\tilde{F_{\boldsymbol{\sigma}}} = \prod_l M_l^{\sigma_l}$. The sum can be placed in front of the product to sum each tensor across the possible physical index values 0 and 1 which are then multiplied together to find the integral.

$$\int_D d^N\mathbf{x} f(\mathbf{x}) \approx \frac{1}{2^L}\sum_{\boldsymbol{\sigma}} f(\mathbf{x}(\boldsymbol{\sigma})) = \frac{1}{2^L}\sum_{\boldsymbol{\sigma}} F_{\boldsymbol{\sigma}} \approx \frac{1}{2^L}\sum_{\boldsymbol{\sigma}} \tilde{F}_{\boldsymbol{\sigma}} = \frac{1}{2^L}\prod_{l=1}^{L}\left[\sum_{\sigma_l=1}^{2} M_l^{\sigma_l}\right] \tag{14}$$

The first $\approx$ refers to the discretisation error which decreases $O(1/2^L)$ and the second



(a)



(b)

Figure 8: Percentage of the quantics domain of d^N sites evaluated as a function of d and N

to the TCI error which is controlled by the bond dimensions of the MPS. The sum can be viewed as a tensor contraction with an MPS of trivial bond dimension 1 between every site and the value 1 when $\sigma_l = 0$ or 1 - known as a Trace MPS. The cost of this sum is then $O(DdL)$ where D is the bond dimension and d is the dimensions of the physical site index which we have been taking to be 2. Once the contraction has been done (see section 2.2), the integral has been calculated.

Unlike Monte-Carlo methods, using an MPS for integration does not suffer from the sign problem as the function is not randomly sampled. As long as the MPS form is expressive and efficient then the integration will be precise.

3.4 Fourier Transform

The Fourier transform finds applications across physics and engineering, with the Fast Fourier Transform algorithm allowing for real time numerical calculations of Fourier transforms. The Quantum Fourier Transform performs the transform with a quantum circuit, aiming for further speed up. As in [9], the transform can be expressed as the unitary matrix

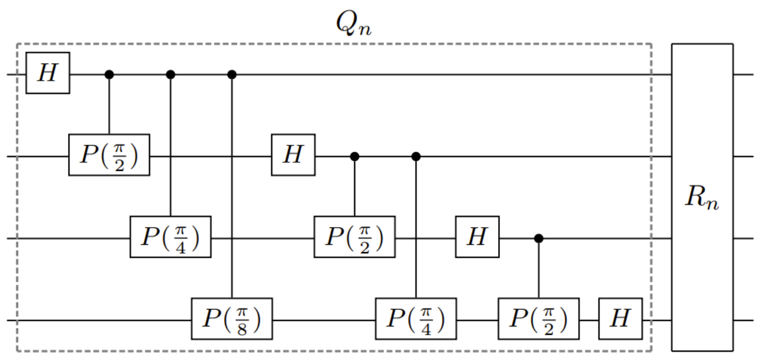
$$F_n = \frac{1}{\sqrt{2^n}} \sum_{p=0}^{2^n-1} \sum_{q=0}^{2^n-1} |p\rangle \langle q| \,, \tag{15}$$

where p and q represent the bit strings of the state of the qubits. This can thought of as a linear map from one state $|q\rangle = |q_1 q_2 ... q_n\rangle$ to another,

$$\begin{aligned} F_n |q_1 q_2 \cdots q_n\rangle = & \frac{1}{\sqrt{2^n}} \left(|0\rangle + e^{2\pi i 0 \cdot q_n} |1\rangle\right) \\ & \otimes \left(|0\rangle + e^{2\pi i 0 \cdot q_{n-1} q_n} |1\rangle\right) \\ & \otimes \cdots \left(|0\rangle + e^{2\pi i 0 \cdot q_1 q_2 \cdots q_n} |1\rangle\right) . \end{aligned} \tag{16}$$

This provides a clearer view to a quantum circuit to implement the transform, which we aim to simulate using an MPO.

The Quantum Fourier Transform (QFT) circuit is comprised of two sections, the first which applies Hadamard and phase gates in periodic structure, Q_n, and the second which reverses the ordering of the bits, R_n. As explored in [9], it is this reversal that takes lots of entanglement generation and the main part of the circuit actually has low entanglement. Since Q_n has low entanglement entropy, it was be efficiently approximated as an MPO. After simulating Q_n with an MPO, R_n can be implemented by reversing the sites of the MPS in order to simulate the full circuit classically.



$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad\qquad P(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$$

Figure 9: QFT circuit diagram taken from [9]

A quantum circuit diagram already has a tensor network type structure where different gates apply locally to different sites. In order to then express this as an MPO a series of SVDs are performed to contract the circuit. Using ITensors it is also possible to simply add gates to sites to produce an MPO of the circuit. With this MPO and an MPS of a function from the TCI algorithm contracting the two produces the Fourier transform of the function.

As discussed in [9], for greater than around $n = 18$ qubits for data of size 2^n, simulating QFT with tensor networks can provide speed up over the FFT algorithm - even when accounting for the time for TCI to produce the MPS state.

4 Density Matrix Renormalisation Group

4.1 The DMRG Algorithm

The DMRG algorithm uses a variational approach to find the ground state: starting with a random MPS, individual tensors in the MPS are replaced to minimise the energy with respect to each site [5] [3]. For a Hamiltonian H, the energy we want to minimise is given by

$$E = \frac{\langle\psi| H |\psi\rangle}{\langle\psi|\psi\rangle} \tag{17}$$

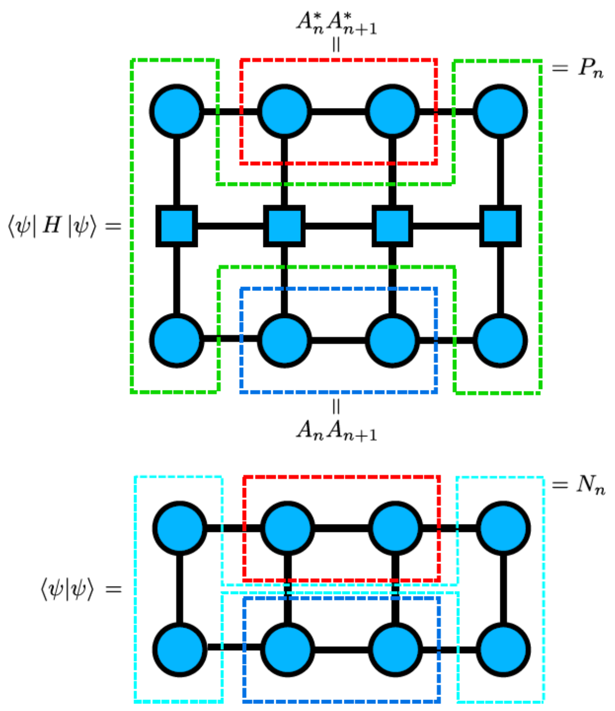


Figure 10: Definitions of P_n and N_n, taken from [5]

Rather than solving for the entire state at once, DMRG considers two adjacent sites n and $n+1$. Reframing the problem for these two sites and defining the tensors P_n and N_n as in fig.10 the energy is expressed as the ratio of the expectation values and these tensors between two sites. Differentiating with respect to the sites and setting to 0 gives the equation

$$P_n(A_n A_{n+1}) = E N_n(A_n A_{n+1}). \tag{18}$$

By choosing our tensors such that the MPS is in a mixed canonical form with the orthogonality center at site n, the tensor $N_n = \sum_{i \neq n,n+1} A_i A_i^*$ becomes the identity. This means that by reshaping P_n into a matrix the minimising equation becomes the simple eigenvalue equation

$$P_n(A_n A_{n+1}) = E(A_n A_{n+1}). \tag{19}$$

After solving this eigenvalue equation, an SVD is performed on the resulting state and the bond dimensions between sites n and $n+1$ are truncated to reduce the size of the MPS. By using a two-site DMRG algorithm rather than simply considering one site at a time, we can adjust the accuracy and compression of the MPS by changing the bond dimensions between sites when we truncate during the SVD.

By sweeping through all the sites, ensuring to update the orthogonality center each time, the global MPS can be optimised to find the ground state of the Hamiltonian and its energy.

Once the ground state ψ has been found, it is simple to then create a new Hamiltonian

$$H \to H' = H + \lambda \left|\psi\right\rangle \left\langle\psi\right| . \tag{20}$$

As long as the Lagrange multiplier λ is larger than the energy gap between the ground state energy and the first excited state energy, the new Hamiltonian will have the first excited state of H as its ground state, and DMRG can be used to find it.

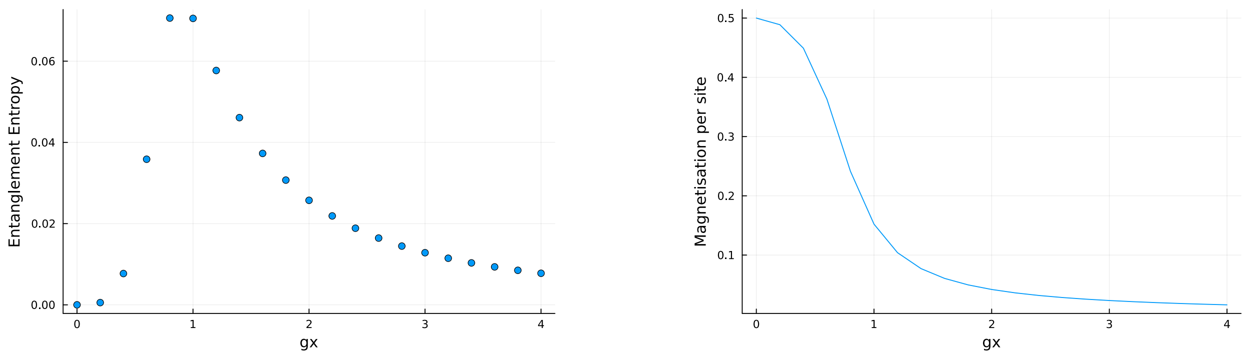
Another method to find excited states, and the method that is implemented by the ITensors library[10], is to enforce an orthogonality constraint. By searching for the lowest energy state that is orthogonal to the ground state, the first excited state is found. At each step when individual sites are updated, the new tensors must ensure the MPS remains orthogonal to the ground state MPS. If we start with a random MPS orthogonal to the ground state and keep the orthogonality centers of the two states at the sites being updated, then the orthogonality constraint is automatically held at all other sites and need only be enforced at sites n and $n+1$.

4.2 The Ising Model

The Transverse Ising Model is a simple spin model with nearest neighbour interactions in the z direction and an external field applied perpendicular to it, here we will also apply a small field along z to ensure the uniqueness of the ground state. The 1D Hamiltonian we will use is

$$\hat{H} = J \sum_{i=1}^{N-1} Z_i Z_{i+1} + g_x \sum_{i=1}^{N} X_i + g_z \sum_{i=1}^{N} Z_i, \tag{21}$$

where J represents nearest neighbour coupling and will be sent to -1, g_x represents the transverse field strength and g_z the longitudinal field strength. If $J > g_x$ then the nearest neighbour interactions dominate and the ground state is (anti)ferromagnetic for positive (negative) J. If $g_x > J$ then the transverse field dominates and the system becomes



(a) Entanglement Entropy (b) Magnetisation

Figure 11: Phase change in the Ising model as g_x increases as seen from the expectation of entanglement entropy and magnetisation per site with $N = 16$

paramagnetic as spins begin to align along the field direction. As the field strength grows from 0 we see a phase change at $g_x/J = \pm 1$ with distinct changes in magnetisation and entanglement entropy. Fig.11b shows plots created using the DMRG algorithm to find the ground state of the Ising model with $N = 16$ for varying field strengths between 0 and 4. As seen in fig.11b, magnetisation (along z) decreases since the spins are pushed to align along the field, and entanglement entropy spikes at the phase change.

4.3 The Schwinger Model

The Schwinger model is a 1+1D quantum electrodynamics model of one fermion flavour. It has been the subject of much research due to its relevance in theoretical and experimental physics and its similarities to QCD. It allows us to study many properties of QED in a lower-dimensional setting and we will explore some of these using the DMRG algorithm. For a detailed discussion of the model see [11] and [5].

The Hamiltonian density is given by

$$\mathcal{H} = -i\bar{\psi}\gamma^1 \left(\partial_1 - igA_1\right)\psi + m\bar{\psi}\psi + \frac{1}{2}\left(\dot{A}_1 + \frac{g\theta}{2\pi}\right)^2 . \tag{22}$$

Where the first term is a kinetic term, followed by the coupling between ψ and the gauge field A_μ, the third term is the mass term and finally the dynamic electric field density and the static background electric field controlled by the topological term $\theta \in [0, 2\pi]$. We use the temporal gauge $A_0 = 0$ so the field is static. Physical states must obey Gauss' law $\delta_1 \dot{A}_1 = -g\psi^\dagger\psi$ and since charge is conserved, we see that the physical subspace has 0 charge since otherwise there would be infinite field energy [5].

In the large mass limit the kinetic term can be ignored and it is less probable to produce charged particles. With a small background field the energy comes only from this field, but at higher field strengths it becomes energetically favourable to produce a pair of charged particles which screen the background field and lower the overall energy. In the continuum this phase transition happens at $\theta = \pi$, but only with the lattice mass greater than the critical value $m_c \approx 0.33$ below which the phase transition become a smooth transition [11]. Hence we have a first order phase transition with $m > m_c$ at $l_0 = \theta/2\pi = 0.5$ and a second phase transition at $m = 0.33$.

To simulate this using Tensor Networks, the Hamiltonian density needs to be discretised onto a lattice. However simply discretising onto a lattice causes the fermion doubling problem - discussed in [5] - and so we use a Wilson Fermions approach using $2N$ sites to represent a lattice of N points to avoid this. By mapping the fermionic operators onto spin operators we produce the Hamiltonian:

$$
\begin{aligned}
H_W = x(r-1) &\sum_{n=0}^{N-2} \left(\sigma^+_{2n} Z_{2n+1} Z_{2n+2} \sigma^-_{2n+3} + \text{h.c. }\right) \\
&+ \frac{x(r+1)}{2} \sum_{n=0}^{N-2} (X_{2n+1}X_{2n+2} + Y_{2n+1}Y_{2n+2}) \\
&+ \left(\frac{m_{\text{lat}}}{g}\sqrt{x} + xr\right) \sum_{n=0}^{N-1} (X_{2n}X_{2n+1} + Y_{2n}Y_{2n+1}) \\
&+ \frac{1}{2} \sum_{n=0}^{2N-1} \sum_{k=n+1}^{2N-1} \left(N - \left\lceil \frac{k+1}{2} \right\rceil + \lambda\right) Z_n Z_k \\
&+ l_0 \sum_{n=0}^{2N-3} \left(N - \left\lceil \frac{n+1}{2} \right\rceil\right) Z_n \\
&+ l_0^2(N-1) + \frac{1}{4}N(N-1) + \frac{\lambda N}{2}
\end{aligned}
\tag{23}
$$

Where $x = 1/(ag)^2$, with a as the lattice spacing, and r is the Wilson parameter which we set to 1. To see the phase changes we track two operators: particle number P_W and the electric field density L_W.

$$
P_W = N + \frac{1}{2} \sum_{n=0}^{N-1} (X_{2n}X_{2n+1} + Y_{2n}Y_{2n+1}) \tag{24}
$$

$$
L_W = l_0 + \frac{1}{2} \sum_{k=0}^{\lceil N/2 \rceil - 1} (Z_{2k} + Z_{2k+1}), \tag{25}
$$

For smaller values of N we also see lattice effects which change the location of the phase changes, this why we see the phase change at $l_0 \approx 1.5$ in fig.12. We see in fig.12 that there is a sharp and clear phase transition, where the particle number increases to two as we have pair production and the field strength decreases as the field is screened by the charges of the particles. However in fig.13, below the critical mass, this transition becomes smoother.

5 Time Evolving Block Decimation

5.1 The TEBD Algorithm

The Time Evolving Block Decimation (TEBD) algorithm aims to time evolve an MPS under the influence of a Hamiltonian. After using DMRG to calculate the ground state of a Hamiltonian TEBD can then be used to evolve it in time as the Hamiltonian changes. A full overview of the algorithm can be found in [4]

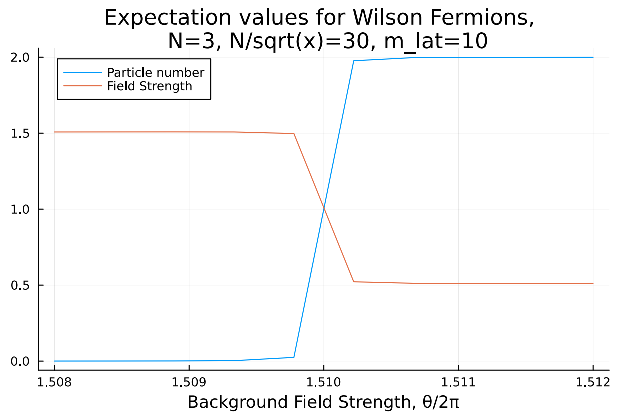
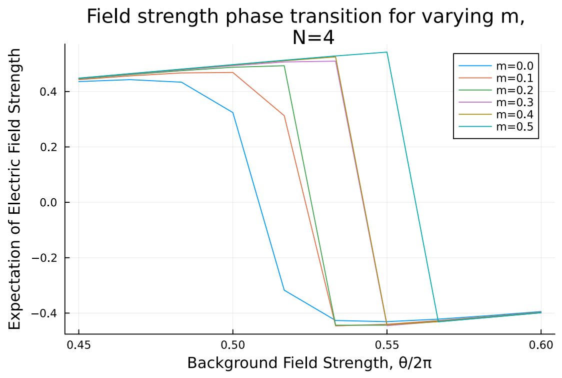
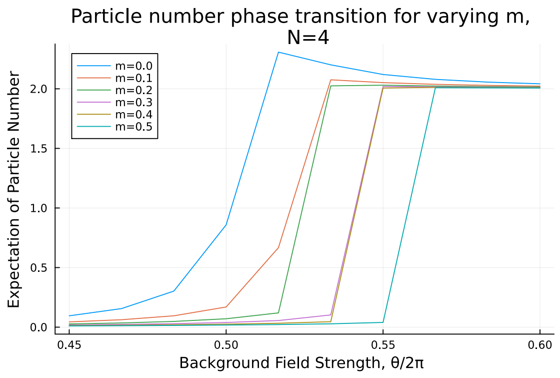


Figure 12: The first order phase transition in the Schwinger model



(a) Electric Field Strength Expectation values



(b) Particle Expectation values

Figure 13: Ground state expectation values for the Schwinger model for varying field strength with different masses

If we have some Hamiltonian $\hat{H}$ it has a time evolution operator $\hat{U}(\delta) = e^{-i\delta\hat{H}}$. This exponentiation can be approximated by splitting the Hamiltonian into two parts, $\hat{H} = \hat{H}_1 + \hat{H}_2$.

$$\hat{U}(\delta) = e^{-i\delta\hat{H}_1}e^{-i\delta\hat{H}_2}e^{-i\delta^2[\hat{H}_1,\hat{H}_2]} \approx e^{-i\delta\hat{H}_1}e^{-i\delta\hat{H}_2} \equiv \hat{U}^{\text{TEBD}_1}(\delta), \tag{26}$$

which has error $O(\delta^2)$. Second order TEBD uses

$$\hat{U}^{\text{TEBD}_2} \equiv e^{-i\frac{\delta}{2}\hat{H}_1}e^{-i\delta\hat{H}_2}e^{-i\frac{\delta}{2}\hat{H}_1}, \tag{27}$$

which has error $O(\delta^3)$. But if we are trying to time evolve the state for some time T with timesteps T/δ then the error per time step is $O(\delta^2)$. By choosing the partition of the Hamiltonian such that $\hat{H}_1$ and $\hat{H}_2$ are internally commuting then we can easily construct low bond dimension MPOs of each exponential since every term in them can be individually exponentiated. For example a two site Hamiltonian $\hat{H} = \sum h_{j,j+1}$ can be split into $\hat{H}_{\text{even}}$ and $\hat{H}_{\text{odd}}$ which contain the even and odd js respectively. All these terms then commute and an MPO for each exponential can be easily constructed. Time evolving the state is then as easy as repeatedly contracting the MPOs with the MPS to construct the state at each time step. After each time step it is important to normalise the state as the errors due to the truncation of bond dimensions can lead to the state becoming unnormalised.

The time cost of TEBD is dominated by the SVDs when applying gates, making it $O(ND^3d^3)$ where D is the bond dimensions and d is the physical site dimensions.

5.2 Imaginary Time Evolution

By making the transformation $\delta \to -i\delta$ to apply imaginary time evolution, rather than applying a phase $e^{-i\delta E_i}$ to each eigenstate in the state being evolved, an exponential decay term $e^{-\delta E_i}$ is applied. The ground state, having the lowest energy, has the slowest decay rate and in the limit $\delta \to \infty$ has an exponentially larger coefficient than all other states. Therefore by applying imaginary time evolution with the TEBD algorithm to a random initial MPS, as long as the starting state has some non-zero projection onto the ground state of the Hamiltonian, after many timesteps the final state will be an almost exact approximation of the ground state.

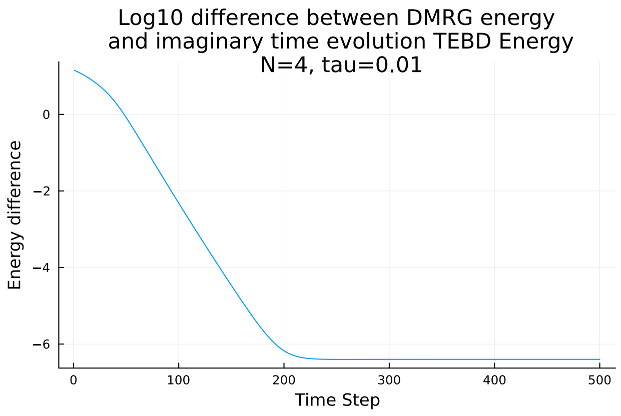
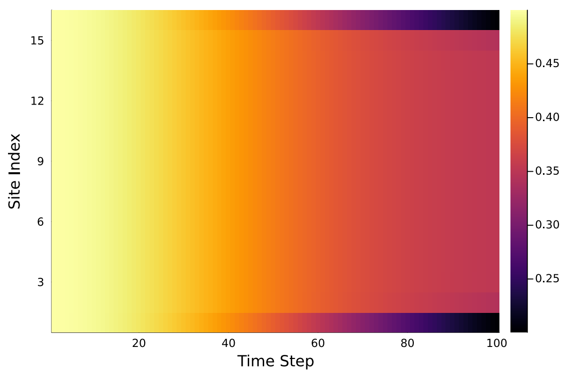


Figure 14: Imaginary Time Evolution with the TEBD Algorithm

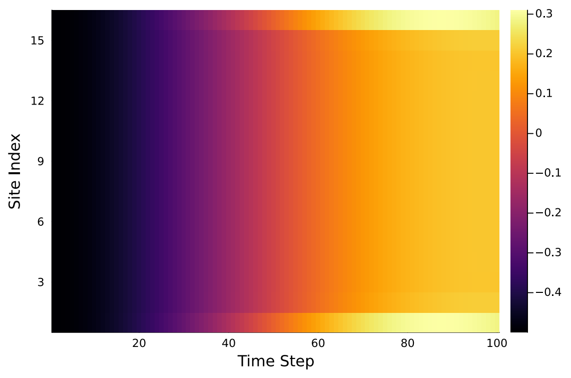
In fig.14 the convergence of imaginary time evolution for the Ising model is shown and a clear exponential can be seen in the energy difference as the other eigenstates in the random initial state decay exponentially.

5.3 Ising Model Quenches

With no external field, the Ising model with $J < 0$ gives a ferromagnetic ground state where all the spins have the same orientation. By initially finding the ground state with no external field and then quenching the Hamiltonian by turning on an external field at $T = 0$ we can use TEBD to see the system evolve into the ground state of this new system.



(a) Transverse field less than J

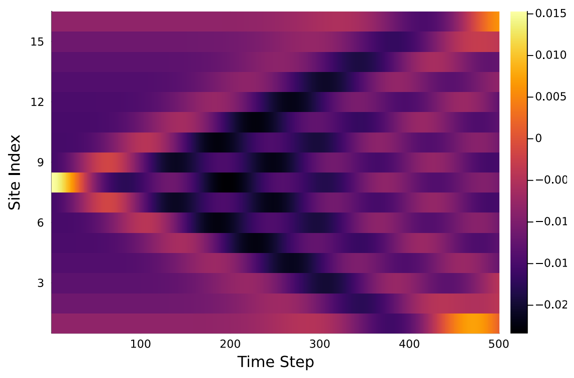


(b) Transverse field greater than J

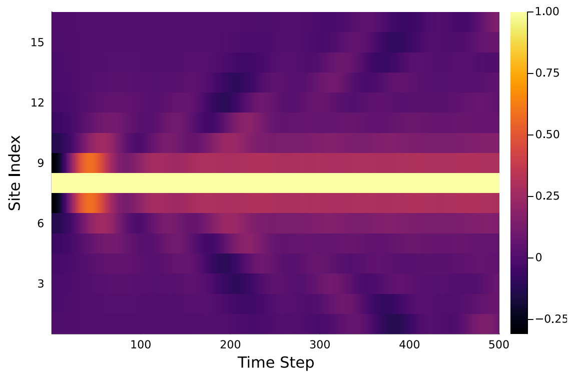
Figure 15: Quenching the Ising model from no external field to transverse external field greater or less than J

5.4 Spinons

By flipping one spin in the ground state of a 1D spin chain, the resulting state is a low-lying excited state. When transverse terms are present, the spin-flip causes the propagation of a particles known as spinons or magnons. By using TEBD after flipping one spin in the ground state of the Transverse Field Ising model, we can see two spinons propagate out from the center as explained in [12]. When reaching the boundaries they reflect and interference patterns can be seen. Correlations also grow from the center point with each time step at a fixed velocity which changes with the strength of the external field. Correlations are quantified with the correlation coefficient $C_{ij} = \langle \sigma_i^z \sigma_j^z \rangle - \langle \sigma_i^z \rangle \langle \sigma_j^z \rangle$.

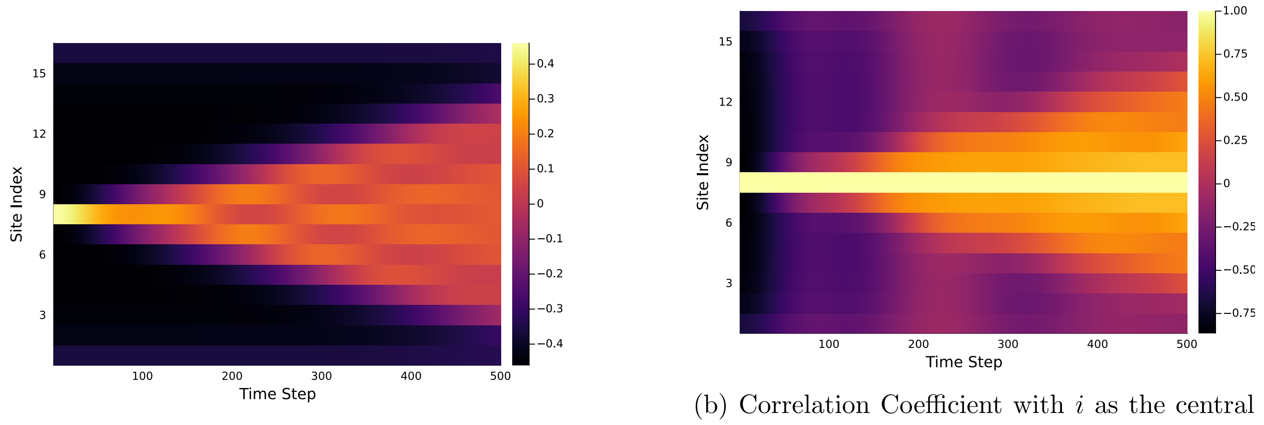


(a) Magnetisation



(b) Correlation Coefficient with i as the central site

Figure 16: Spinons in the Ising model with transverse field = 1.7



(a) Magnetisation

(b) Correlation Coefficient with i as the central site

Figure 17: Spinons in the Ising model with transverse field = 0.7

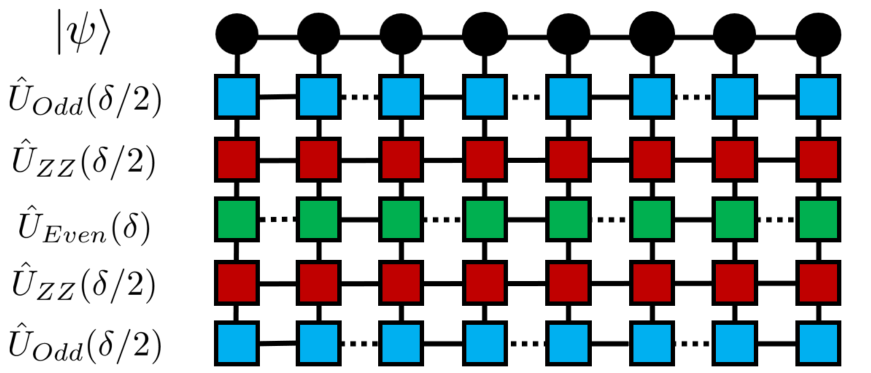


Figure 18: Diagram of applying one time step to evolve the Schwinger model

5.5 Time Dependant Schwinger

The Schwinger model contains long range ZZ terms in the sum:

$$\frac{1}{2}\sum_{n=0}^{2N-1}\sum_{k=n+1}^{2N-1}(N-\lceil\frac{k+1}{2}\rceil+\lambda)Z_nZ_k \tag{28}$$

Although they can be included and exponentiated individually by using swaps to move sites closer together before applying the gates, this is costly in time complexity. Instead, the Hamiltonian is split into 3 terms, H_{Odd}, H_{Even} and H_{ZZ}. The ZZ Hamiltonian is implemented as an MPO which can represent the terms with small error due to the decaying coefficients. In this way we can use TEBD to time evolve the Schwinger model and probe the dynamics [13].

By quenching the model from no external field to different values of l_0, we can see how pair production occurs by tracking the expectation of particle number. As in [13], we observe the stable vacuum for small external fields. For larger fields, particle number initially increases linearly but then saturates and we see oscillations in number.

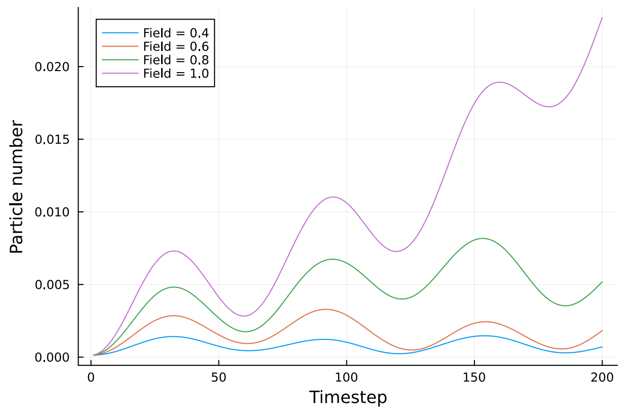


Figure 19: Pair production induced by external fields

6 Conclusion

This report provides an overview of three common and useful tensor network algorithms which exploit the properties of MPS to efficiently approximate functions and simulate physical systems. Many more applications are possible than those discussed in the report, however to better understand how these algorithms simulate continuous systems on a lattice, a full exploration of lattice effects and convergence to the continum limit should be explored as in [14].

References

[1] Aleksandr Berezutskii, Minzhao Liu, Atithi Acharya, Roman Ellerbrock, Johnnie Gray, Reza Haghshenas, Zichang He, Abid Khan, Viacheslav Kuzmin, Dmitry Lyakh, et al. Tensor networks for quantum computing. *Nature Reviews Physics*, pages 1–13, 2025.

[2] Yuriel Núñez Fernández, Marc K Ritter, Matthieu Jeannin, Jheng-Wei Li, Thomas Kloss, Thibaud Louvet, Satoshi Terasaki, Olivier Parcollet, Jan von Delft, Hiroshi Shinaoka, et al. Learning tensor networks with tensor cross interpolation: new algorithms and libraries. *SciPost Physics*, 18(3):104, 2025.

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

[4] Andrew John Daley, Corinna Kollath, Ulrich Schollwöck, and Guifré Vidal. Time-dependent density-matrix renormalization-group using adaptive effective hilbertspaces. *Journal of Statistical Mechanics: Theory and Experiment*, 2004(04):P04005, 2004.

[5] Takis Angelides. *Tensor network and quantum simulations of 1+1-dimensional quantum electrodynamics*. PhD thesis, der Humboldt-Universität zu Berlin, 2025.

[6] Matthew Fishman, Steven White, and Edwin Miles Stoudenmire. The itensor software library for tensor network calculations. *SciPost Physics Codebases*, page 004, 2022.

[7] Yuriel Núñez Fernández, Matthieu Jeannin, Philipp T Dumitrescu, Thomas Kloss, Jason Kaye, Olivier Parcollet, and Xavier Waintal. Learning feynman diagrams with tensor trains. *Physical Review X*, 12(4):041018, 2022.

[8] Shuta Matsuura, Hiroshi Shinaoka, Philipp Werner, and Naoto Tsuji. Tensor cross interpolation approach for quantum impurity problems based on the weak-coupling expansion. *Physical Review B*, 111(15):155150, 2025.

[9] Jielun Chen, EM Stoudenmire, and Steven R White. Quantum fourier transform has small entanglement. *PRX Quantum*, 4(4):040318, 2023.

[10] Matthew Fishman, Steven R. White, and E. Miles Stoudenmire. The ITensor Software Library for Tensor Network Calculations. *SciPost Phys. Codebases*, page 4, 2022.

[11] Takis Angelides, Lena Funcke, Karl Jansen, and Stefan Kühn. Computing the mass shift of wilson and staggered fermions in the lattice schwinger model with matrix product states. *Physical Review D*, 108(1):014516, 2023.

[12] Teresa Kulka, Miłosz Panfil, Mona Berciu, and Krzysztof Wohlfeld. Nature of spinons in 1d spin chains. *Physical Review Letters*, 134(23):236504, 2025.

[13] Giuseppe Magnifico, Marcello Dalmonte, Paolo Facchi, Saverio Pascazio, Francesco V Pepe, and Elisa Ercolessi. Real time dynamics and confinement in the $\mathbb{Z}_n$ schwinger-weyl lattice model for 1+ 1 qed. *Quantum*, 4:281, 2020.

[14] Elisa Ercolessi, Paolo Facchi, Giuseppe Magnifico, Saverio Pascazio, and Francesco V. Pepe. Phase transitions in Z_n gauge models: Towards quantum simulations of the schwinger-weyl qed. *Phys. Rev. D*, 98:074503, Oct 2018.