Journal of Computational Physics 372 (2018) 587–602

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Multigrid renormalization

Michael Lubasch [a,*], Pierre Moinier [b], Dieter Jaksch [a,c,d]

[a] Clarendon Laboratory, University of Oxford, Parks Road, Oxford OX1 3PU, United Kingdom
[b] BAE Systems MAI, Computational Engineering, Buckingham House, FPC 267 PO Box 5, Filton, Bristol BS34 7QW, United Kingdom
[c] Centre for Quantum Technologies, National University of Singapore, 3 Science Drive 2, 117543, Singapore
[d] Keble College, University of Oxford, Parks Road, Oxford OX1 3PG, United Kingdom

A R T I C L E   I N F O

Article history:
Received 22 February 2018
Received in revised form 12 June 2018
Accepted 24 June 2018
Available online 27 June 2018

Keywords:
Multigrid methods
Numerical renormalization group
Density matrix renormalization group
Variational renormalization group methods
Matrix product states
Quantics tensor trains

A B S T R A C T

We combine the multigrid (MG) method with state-of-the-art concepts from the variational formulation of the numerical renormalization group. The resulting MG renormalization (MGR) method is a natural generalization of the MG method for solving partial differential equations. When the solution on a grid of N points is sought, our MGR method has a computational cost scaling as $\mathcal{O}(\log(N))$, as opposed to $\mathcal{O}(N)$ for the best standard MG method. Therefore MGR can exponentially speed up standard MG computations. To illustrate our method, we develop a novel algorithm for the ground state computation of the nonlinear Schrödinger equation. Our algorithm acts variationally on tensor products and updates the tensors one after another by solving a local nonlinear optimization problem. We compare several different methods for the nonlinear tensor update and find that the Newton method is the most efficient as well as precise. The combination of MGR with our nonlinear ground state algorithm produces accurate results for the nonlinear Schrödinger equation on $N = 10^{18}$ grid points in three spatial dimensions.

1. Introduction

Multigrid (MG) methods [1] are among the most efficient tools for the boundary value problem of (nonlinear) partial differential equations [2]. The standard MG method discretizes space by a finite number of grid points N and provides a recipe for approaching the solution in continuous space by successively increasing N. The ultimate goal of MG methods is the continuous solution which is obtained in the limit $N \to \infty$. MG methods are iterative solvers and their computational cost scales better than for others, e.g. direct solvers, namely linearly with N in the best case. This allows MG methods to go to larger values of N and closer to the continuum limit $N \to \infty$ than alternative methods.

Around the same time when MG methods were first formulated to tackle the limit $N \to \infty$ grid points in mathematics, new renormalization group (RG) techniques [3] were invented to address the limit of $N \to \infty$ quantum particles in theoretical physics. The invention of the numerical renormalization group (NRG) [3] led to a breakthrough when it solved the Kondo problem – of a single magnetic quantum impurity in a nonmagnetic metal – which was an outstanding problem in condensed matter theory until then. While the NRG has proven to be a very powerful tool for a variety of quantum impurity problems [4], it has fundamental difficulties with more general quantum many-body problems such as e.g. quantum spin chains. For the latter systems, most of these difficulties could be overcome when the density matrix renormalization group

* Corresponding author.
E-mail address: michael.lubasch@physics.ox.ac.uk (M. Lubasch).

https://doi.org/10.1016/j.jcp.2018.06.065
0021-9991/© 2018 Elsevier Inc. All rights reserved.

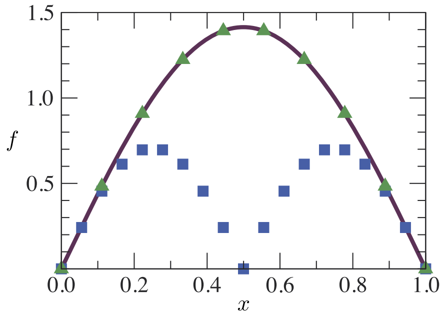

Fig. 1. Exact ground state wave function $f(x)$ for a free quantum particle in a box (solid line). The numerical renormalization group (NRG) approximates the solution on a finer grid using the solutions from a coarser grid as depicted by the squares, which correspond to the lowest eigenstate on the coarser grid. More precisely, NRG uses many low-lying eigenstates on the coarser grid to approximate the solution on the finer grid. The standard NRG approach fails because all combinations of these eigenstates have a node at $x = 0.5$ which is absent in the desired solution on the finer grid. Applying a variety of boundary conditions on the coarser grid improves NRG for this problem [14]. However, the optimal boundary conditions on the coarser grid need to be found empirically, and so this problem is a classic example for the limitations of NRG. Our multigrid renormalization (MGR) method approximates the solution on a finer grid using the solution from a coarser grid as depicted by the triangles in a natural and optimal way. Our MGR method uses the same boundary conditions on the coarser grid as on the finer grid, and these are, in fact, the desired boundary conditions for the continuous solution.

(DMRG) was invented [5,6] as a modified version of NRG – reference [7] summarizes the early success of DMRG. However, also DMRG has fundamental limitations, and many of them could be overcome when yet another key discovery was made: namely that DMRG can be equally formulated as a variational algorithm over an ansatz called matrix product states [8]. This led to new variational renormalization group (RG) methods that generalized and extended DMRG – references [9–11] cover this topic comprehensively. Note that also NRG could be improved by reformulation as a variational RG method [12,13].

In this article, we generalize and extend the MG method by combining it with state-of-the-art concepts from variational RG methods. The resulting MG renormalization (MGR) method has a computational cost that formally scales like $\mathcal{O}(\log(N))$ and can thus be exponentially faster than the MG method. Our MGR method is a natural combination of MG and variational RG methods, and it is optimal in a certain sense that we will explain later. As an example, Fig. 1 compares the performance of NRG and our MGR method for a free quantum particle in a box. This problem has often been used to illustrate fundamental limitations of NRG [14]. Our MGR method overcomes these limitations.

Later in this article, we will see that MGR gives rise to tensor product states. This variational ansatz is also known as tensor network states or as tensor networks in the physics community – and references [9,10,15] give a comprehensive overview of the physics perspective. An important subset of tensor product states is called matrix product states [8,16–18]. These are also known as tensor train decompositions [19] or as quantics tensor train decompositions [20,21] in the mathematics community. Here we will use existing results from the mathematics community. However, our motivation, derivations, and applications originate from a physics perspective.

It is also interesting to note that MGR produces a function representation that was originally proposed in the context of quantum computation [22–24]. There, this representation was used to show that a quantum computer achieves an exponential compression rate of lossless compression compared to a classical computer. Here, by using tensor product states, we will understand that MGR achieves an exponential compression rate of lossy compression on a classical computer.

This article is structured as follows. In section 2 we present the general MGR concept. We show how MGR naturally leads to tensor product states in subsection 2.1 and formulate the MGR method for the Poisson equation in subsection 2.2. MGR is then used to construct a new algorithm for the ground state computation of the nonlinear Schrödinger equation in section 3. There we first introduce quantities that measure the accuracy of MGR in subsection 3.1, then present our new algorithm in subsection 3.2, and finally analyse its performance in subsection 3.3.

2. Multigrid renormalization (MGR) method

In this section we present the MGR method, which is a general concept that can be applied to any (nonlinear) partial differential equation.

2.1. MGR gives rise to tensor product states

In the following we use the finite difference method to represent (nonlinear) partial differential equations. We denote the sought solution function by f and assume that it is defined on the interval $[0, 1]$ in each of its variables – any function can be scaled to satisfy this with all its variables. We define the vector $|f\rangle_h$ as the values of f on an equidistant grid of spacing h. Furthermore each variable of f shall be discretized by $N = 2^L$ points, i.e. $h = 1/N = 1/2^L$ for each variable. If L, N, or h is given explicitly then we drop the subindex h in $|f\rangle_h$ and write $|f\rangle$ to simplify the notation. We first present our method for a function f that depends only on one variable x and later discuss the extension to more variables.

MG methods find the solution on a fine grid with the help of exact solutions from coarse grids. A crucial ingredient is the prolongation operator $\mathcal{P}$ that maps a function from a coarse grid of spacing h to a finer grid of spacing $h/2$: $\mathcal{P}|f\rangle_h = |f\rangle_{h/2}$.

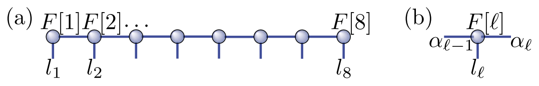

Fig. 2. Matrix product state $\sum_{\alpha_1,\alpha_2,\ldots,\alpha_7}\left(F[1]^{l_1}_{\alpha_1}F[2]^{l_2}_{\alpha_1,\alpha_2}F[3]^{l_3}_{\alpha_2,\alpha_3}\ldots F[8]^{l_8}_{\alpha_7}\right)$ of length $L=8$ (a) and a tensor $F[\ell]^{l_\ell}_{\alpha_{\ell-1},\alpha_\ell}$ from an interior site $1<\ell<L$ (b). In this graphical representation of a tensor product state, each node (blue sphere) corresponds to a tensor and each edge (blue line) corresponds to an index. Therefore the rank of a tensor, i.e. its total number of indices, is given by the total number of edges connected to that tensor. When a tensor has an open edge then this represents a degree of freedom. We denote the corresponding open indices by Latin letters and define that each open index takes values from $\{0,1,\ldots,d-1\}$. E.g. in (a) each open index l_1, l_2, ..., or l_L can take any value from $\{0,1,\ldots,d-1\}$ and so this matrix product state represents $N=d^L$ numbers. Note that d can be seen as the dimensionality of a vector space in which one open index lives. When two tensors are connected via an edge then the corresponding two indices are summed over. We denote these connecting indices by Greek letters and define that each connecting index takes values from $\{1,2,\ldots,\chi\}$. E.g. in (a) the tensors $F[1]$ and $F[2]$ are connected via one edge which corresponds to the sum over the index α_1, i.e. $\sum_{\alpha_1}\left(F[1]^{l_1}_{\alpha_1}F[2]^{l_2}_{\alpha_1,\alpha_2}\right)$. In practice the connecting indices can live in vector spaces of different dimensionalities and then χ is defined as the largest of these dimensionalities. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Usually MG methods also require a restriction operator that maps a function from a fine grid to a coarser grid [1,2,25] but we will not need this restriction operator here.

We now formulate an exact MG method that starts from a large grid spacing $h=0.5$ and then refines the grid in an exact way. Initially, we want to store the function values of f at $x=0.0$ and 0.5 in a vector $|f\rangle_{h=0.5}$. This vector lives in a two-dimensional vector space that is spanned by two basis vectors $|0\rangle$ and $|1\rangle$. We define that the basis vector $|l_1\rangle$ – where $l_1=0$ or 1 – corresponds to $x=l_1 2^{-1}$ on the grid. We write $|f\rangle_{h=0.5}=\sum_{l_1=0}^{1}F[1]^{l_1}|l_1\rangle$ where the tensor $F[1]^{l_1}$ has one index l_1 that takes two values 0 or 1 for the two components of $|f\rangle_{h=0.5}$ with respect to the basis vectors $|0\rangle$ and $|1\rangle$. We define a new tensor $F[1]^{l_1}_{\alpha_1}$ where l_1 is the same as before and α_1 takes two values, and we write any new two basis vectors $|\alpha_1\rangle=\sum_{l_1=0}^{1}F[1]^{l_1}_{\alpha_1}|l_1\rangle$ in terms of the old basis vectors $|0\rangle$ and $|1\rangle$. Now we decrease the grid spacing by a factor 0.5 such that $h=0.25$. For this refined grid, we want to store the function values of f at $x=0.0$, 0.25, 0.5, and 0.75 in a vector $|f\rangle_{h=0.25}$. This vector lives in a four-dimensional vector space that we construct as the tensor product of the previous vector space with basis vectors $|\alpha_1\rangle$ and a new two-dimensional vector space with basis vectors $|0\rangle$ and $|1\rangle$. We define that the basis vector $|l_1\rangle|l_2\rangle$ – where $l_1=0$ or 1 and $l_2=0$ or 1 – corresponds to $x=l_1 2^{-1}+l_2 2^{-2}$ on the grid. We write $|f\rangle_{h=0.25}=\sum_{l_2}\sum_{\alpha_1}F[2]^{l_2}_{\alpha_1}|\alpha_1\rangle|l_2\rangle$ where the tensor $F[2]^{l_2}_{\alpha_1}$ has one index α_1 for the previous basis and one index l_2 for the new basis. We define a new tensor $F[2]^{l_2}_{\alpha_1,\alpha_2}$ where α_1 and l_2 are the same as before and α_2 takes four values, and we write any new four basis vectors $|\alpha_2\rangle=\sum_{l_2}\sum_{\alpha_1}F[2]^{l_2}_{\alpha_1,\alpha_2}|\alpha_1\rangle|l_2\rangle$ in terms of the old basis vectors. We continue this procedure. In each step the grid spacing h decreases by a factor of 0.5 and the dimensionality of the corresponding vector $|f\rangle$ grows by a factor of 2. Thus in step ℓ, the grid spacing reads $h=1/2^\ell$ and $|f\rangle_{h=1/2^\ell}$ lives in a 2^ℓ-dimensional vector space. We define that the basis vector $|l_1\rangle|l_2\rangle\ldots|l_\ell\rangle$ – where $l_1=0$ or 1, $l_2=0$ or 1, and so on – corresponds to $x=l_1 2^{-1}+l_2 2^{-2}+\ldots+l_\ell 2^{-\ell}$ on the grid. We write $|f\rangle_{h=1/2^\ell}=\sum_{l_\ell}\sum_{\alpha_{\ell-1}}F[\ell]^{l_\ell}_{\alpha_{\ell-1}}|\alpha_{\ell-1}\rangle|l_\ell\rangle$. And we define a new tensor $F[\ell]^{l_\ell}_{\alpha_{\ell-1},\alpha_\ell}$ to write any new 2^ℓ basis vectors $|\alpha_\ell\rangle=\sum_{l_\ell}\sum_{\alpha_{\ell-1}}F[\ell]^{l_\ell}_{\alpha_{\ell-1},\alpha_\ell}|\alpha_{\ell-1}\rangle|l_\ell\rangle$.

If we want to store all the tensors $F[1]$, $F[2]$, ...on our computer, we need to be aware that in step ℓ the tensor $F[\ell]$ has the size 4^ℓ, and so the required memory grows exponentially with ℓ. To prevent this from happening we now define a maximum dimensionality χ for the vector space spanned by the new basis vectors $|\alpha_\ell\rangle$. The χ is introduced to reduce the exponential scaling to a polynomial scaling. For the initial steps $\ell\le\log_2(\chi)$ there are 2^ℓ new basis vectors, but for the later steps $\ell>\log_2(\chi)$ there are just χ new basis vectors. Let us denote the dimensionality of the vector space spanned by $|0\rangle$ and $|1\rangle$ by d, where $d=2$ here. Then each tensor $F[\ell]$ has at most the size $d\chi^2$. Therefore if we stop our procedure at step L – when $N=2^L$ – our memory requirements are upper-bounded by $\mathcal{O}(Ld\chi^2)$. We refer to our new MG procedure as a MG renormalization (MGR) method because it allows us to go to much larger values of L by using renormalized degrees of freedom, given by the at most χ-dimensional vector spaces.

Importantly, we can choose these χ-dimensional vector spaces in an optimal way by making use of the fact that MGR produces tensor product states. If we stop our MGR method at step L we can write out the vector $|f\rangle_{h=1/2^L}=\sum_{l_L}\sum_{\alpha_{L-1}}F[L]^{l_L}_{\alpha_{L-1}}|\alpha_{L-1}\rangle|l_L\rangle$ as

$$|f\rangle_{h=1/2^L}=\sum_{l_1,l_2,\ldots,l_L}\sum_{\alpha_1,\alpha_2,\ldots,\alpha_{L-1}}\left(F[1]^{l_1}_{\alpha_1}F[2]^{l_2}_{\alpha_1,\alpha_2}F[3]^{l_3}_{\alpha_2,\alpha_3}\ldots F[L]^{l_L}_{\alpha_{L-1}}\right)|l_1\rangle|l_2\rangle\ldots|l_L\rangle. \quad (1)$$

This is a matrix product state [8,16–18] – also known as tensor train decomposition [19] or as quantics tensor train decomposition [20,21] – of bond dimension χ. In a tensor product state, the term bond dimension denotes the maximum dimensionality of all the internal indices that connect different tensors. Fig. 2 shows a graphical representation. Because MGR produces tensor product states we can reformulate finite difference approaches to (nonlinear) partial differential equations as variational algorithms over tensor product states. This allows us to make use of the plethora of already existing variational renormalization group techniques that determine the χ-dimensional vector spaces in an optimal way – see e.g. reference [9] and references therein.

Our MGR method uses tensor product states as a variational ansatz for the function f and approaches the continuous solution via iteration of two steps: firstly minimizing a cost function and secondly prolonging to the next finer grid. The cost

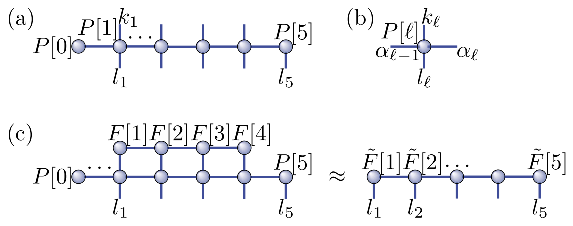

Fig. 3. (a) Matrix product operator [29] $\sum_{\alpha_0,\alpha_1,\ldots,\alpha_4}\left(P[0]_{\alpha_0}P[1]^{k_1,l_1}_{\alpha_0,\alpha_1}\ldots P[5]^{l_5}_{\alpha_4}\right)$ for prolongation from $L=4$ to $L+1=5$. (b) A tensor $P[\ell]^{k_\ell,l_\ell}_{\alpha_{\ell-1},\alpha_\ell}$ from an interior site $0<\ell<L+1$. (c) We prolong $|f\rangle = F[1]F[2]F[3]F[4]$ to $|\tilde{f}\rangle = \tilde{F}[1]\tilde{F}[2]\tilde{F}[3]\tilde{F}[4]\tilde{F}[5]$ by minimizing the cost function $c(|\tilde{f}\rangle) = \|\mathcal{P}|f\rangle - |\tilde{f}\rangle\|^2$ via the alternating minimization procedure of reference [29].

function is a problem-specific function $c(f)$. Here, we always choose the cost function for a problem in such a way that the desired solution f minimizes the cost function. An example cost function is the ground state energy $\langle f|H|f\rangle/\langle f|f\rangle$ where H denotes a Hamiltonian (e.g. from finite difference discretization of a linear Schrödinger equation). This cost function often occurs in physical problems (particularly in quantum physics) and is minimal for the exact ground state. If we are using a variational ansatz to approximate the exact ground state we can minimize the ground state energy via the variational parameters of that ansatz. Using tensor product states we minimize the cost function in the spirit of alternating least squares as proposed in reference [26]: We move from one tensor to the next and for each tensor minimize the cost function via the variational parameters of that tensor only, i.e. leaving the variational parameters of all other tensors fixed. E.g., if we are using the matrix product state of equation (1) as a variational ansatz to minimize a cost function, then we start at the tensor $F[1]$ and minimize the cost function using only the variational parameters of $F[1]$. Then we move to $F[2]$ and minimize the cost function using only the variational parameters of $F[2]$, then to $F[3]$, and so on, until $F[L]$, which is when we move backwards, i.e. $F[L-1]$, then $F[L-2]$, and so on, until $F[1]$. We repeat this procedure several times until our cost function does not change significantly anymore. In the context of tensor product states, several minimization procedures have been proposed, see e.g. references [27,28] for methods using Monte Carlo sampling of the gradient. However, to the best of our knowledge, the alternating minimization procedure of reference [26] is the most efficient and accurate approach for the minimization of many different cost functions. And so we use this method for the minimization of all our cost functions in this article.

After minimizing the cost function of our problem, we prolong f to the next finer grid. We can define our prolongation operator in many different ways. Here we define the prolongation $\mathcal{P}|f\rangle_h = |f\rangle_{h/2}$ in such a way that the new vector $|f\rangle_{h/2}$ has the same values as $|f\rangle_h$ on the same grid points and on the new grid points linearly interpolates between the values of $|f\rangle_h$ on the two adjacent grid points. Expressed in terms of vector components, the prolongation operator $\mathcal{P}$ maps the vector $|f\rangle_h$ with components $|f\rangle^l_h$ to the vector $|f\rangle_{h/2}$ with components $|f\rangle^{2l}_{h/2} = |f\rangle^l_h$ and $|f\rangle^{2l+1}_{h/2} = \left(|f\rangle^l_h + |f\rangle^{l+1}_h\right)/2$. In the tensor product formalism, the operator $\mathcal{P}$ is a tensor product operator, i.e. the generalization of the concept of tensor product states to operators. In the context of matrix product states, $\mathcal{P}$ is a matrix product operator [29] of the form shown in Fig. 3. $\mathcal{P}$ prolongs from L to $L+1$ and is represented by a matrix product operator $\sum_{\alpha_0,\alpha_1,\ldots,\alpha_L}\left(P[0]_{\alpha_0}P[1]^{k_1,l_1}_{\alpha_0,\alpha_1}P[2]^{k_2,l_2}_{\alpha_1,\alpha_2}\ldots P[L]^{k_L,l_L}_{\alpha_{L-1},\alpha_L}P[L+1]^{l_{L+1}}_{\alpha_L}\right)$. The tensors have the entries $P[0]_1 = 1$, $P[\ell]^{0,0}_{1,1} = 1 = P[\ell]^{1,1}_{1,1} = P[\ell]^{1,0}_{1,2} = P[\ell]^{0,1}_{2,2}$ for $0<\ell<L+1$, $P[L+1]^0_1 = 1$ and $P[L+1]^1_1 = 0.5 = P[L+1]^1_2$, and all other tensor entries are zero. Remember that according to our notational convention the α indices run from 1 to at most χ whereas the k and l indices take values from 0 to $d-1$, as explained in Fig. 2. We see that our matrix product operator for $\mathcal{P}$ has bond dimension 2. We prolong $|f\rangle_h$ to $|\tilde{f}\rangle_{h/2}$ approximately, by minimizing the cost function $c(|\tilde{f}\rangle_{h/2}) = \|\mathcal{P}|f\rangle_h - |\tilde{f}\rangle_{h/2}\|^2$ via the alternating minimization procedure of reference [29]. This approximate prolongation has a computational cost that scales like $\mathcal{O}(Ld\chi^3) + \mathcal{O}(Ld^2\chi^2)$. An exact prolongation is not computationally efficient. If we were to compute $\mathcal{P}|f\rangle_h = |f\rangle_{h/2}$ exactly, then the bond dimension χ of $|f\rangle_{h/2}$ would be a factor of 2 larger than the bond dimension of $|f\rangle_h$. Thus, for an exact prolongation, the bond dimension doubles after each prolongation, and so χ grows exponentially with the number of prolongations. Remarkably, for all cases in this article, the approximation errors of our approximate prolongations turned out to be negligibly small, as we have checked explicitly.

In our MGR method, we can choose among many different tensor product ansatzes for higher spatial dimensions dim > 1. In this article, we simply use matrix product states with $d = 2^{\text{dim}}$ as our ansatz for all higher dimensions. Each index l_ℓ now represents a multi-index $l_\ell = (l^x_\ell, l^y_\ell)$ in two or $l_\ell = (l^x_\ell, l^y_\ell, l^z_\ell)$ in three spatial dimensions. And the individual indices in this multi-index specify the spatial coordinates (x, y) in two or (x, y, z) in three spatial dimensions as $x = \sum_{\ell=1}^L l^x_\ell 2^{-l^x_\ell}$, $y = \sum_{\ell=1}^L l^y_\ell 2^{-l^y_\ell}$, and $z = \sum_{\ell=1}^L l^z_\ell 2^{-l^z_\ell}$. For the mapping from the multi-index to the single index, one usually chooses either lexicographic order – where the last index runs fastest (which corresponds to row-major order for a two-dimensional array) – or colexicographic order – where the first index runs fastest (which corresponds to column-major order for a two-dimensional array) – and in our case it is not important which order one chooses. The prolongation operators for higher spatial dimensions are obtained from the matrix product operator $\mathcal{P}$ for one spatial dimension. We obtain the

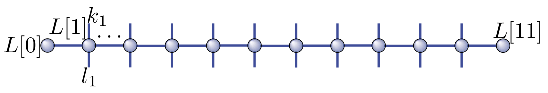

Fig. 4. Laplace operator after discretization for $L = 10$ – i.e. for d^{10} grid points where $d = 2^{\text{dim}}$ – represented by a matrix product operator $\sum_{\alpha_0,\alpha_1,\ldots,\alpha_{10}} L[0]_{\alpha_0} L[1]^{k_1,l_1}_{\alpha_0,\alpha_1} L[2]^{k_2,l_2}_{\alpha_1,\alpha_2} \ldots L[10]^{k_{10},l_{10}}_{\alpha_9,\alpha_{10}} L[11]_{\alpha_{10}}$. The entries of each tensor $L[\ell]$ for $0 \le \ell \le L+1$ are defined and explained in the text.

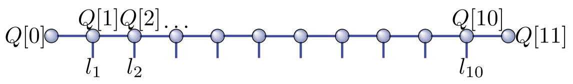

Fig. 5. Polynomial function after discretization for $L = 10$ – i.e. for d^{10} grid points where $d = 2^{\text{dim}}$ – represented by a matrix product state $\sum_{\alpha_0,\alpha_1,\ldots,\alpha_{10}} Q[0]_{\alpha_0} Q[1]^{l_1}_{\alpha_0,\alpha_1} Q[2]^{l_2}_{\alpha_1,\alpha_2} \ldots Q[10]^{l_{10}}_{\alpha_9,\alpha_{10}} Q[11]_{\alpha_{10}}$. The entries of each tensor $Q[\ell]$ for $0 \le \ell \le L+1$ are defined and explained in the text.

matrix product operator $\mathcal{P} \otimes \mathcal{P}$ of bond dimension 4 in two spatial dimensions and $\mathcal{P} \otimes \mathcal{P} \otimes \mathcal{P}$ of bond dimension 8 in three spatial dimensions, where $\otimes$ denotes the Kronecker product.

2.2. MGR for the Poisson equation

The Poisson equation is a paradigmatic application for MG methods. After discretization, this equation reads $\Delta_h |f\rangle_h = |g\rangle_h$ where Δ_h represents the Laplace operator, $|g\rangle_h$ and boundary conditions are given, and $|f\rangle_h$ is sought. We define the Laplace operator via the usual finite difference approximation of the second derivative from a Taylor expansion. Expressed in terms of vector components, our Laplace operator Δ_h maps the vector $|f\rangle_h$ with components $|f\rangle^l_h$ to the vector $\Delta_h|f\rangle_h$ with components $(\Delta_h|f\rangle_h)^l = \left(|f\rangle^{l+1}_h - 2|f\rangle^l_h + |f\rangle^{l-1}_h\right)/h^2$. This finite difference approximation of the second derivative Δ_h has an error $\mathcal{O}(h^2)$ that we can reduce systematically by decreasing h, i.e. increasing L. We choose the function g on the right-hand side as a polynomial because then we know the exact solution f of our Poisson equation. This allows us to analyze the performance of our MGR method by comparing to the exact solution. We choose the boundary conditions $f(x=-h) = 0 = f(x=1)$ – and similarly in higher dimensions for y and z – as they can be implemented easily in our representation of the Laplace operator.

The Laplace operator is written as a matrix product operator as shown in Fig. 4, see also references [19,20,30]. In one spatial dimension this reads $\sum_{\alpha_0,\alpha_1,\ldots,\alpha_L} L[0]_{\alpha_0} L[1]^{k_1,l_1}_{\alpha_0,\alpha_1} L[2]^{k_2,l_2}_{\alpha_1,\alpha_2} \ldots L[L]^{k_L,l_L}_{\alpha_{L-1},\alpha_L} L[L+1]_{\alpha_L}$. The tensors have the entries $L[0]_1 = 1$, $L[\ell]^{0,0}_{1,1} = 4 = L[\ell]^{1,1}_{1,1} = L[\ell]^{1,0}_{2,2} = L[\ell]^{0,1}_{1,2} = L[\ell]^{1,0}_{1,3} = L[\ell]^{0,1}_{3,3}$ for $0 < \ell < L+1$, $L[L+1]_1 = -2$ and $L[L+1]_2 = 1 = L[L+1]_3$, and all other tensor entries are zero. This matrix product operator has bond dimension 3. Our Laplace operators for higher spatial dimensions are constructed from this matrix product operator Δ_h for one spatial dimension. We obtain the matrix product operator $\Delta_h \otimes \mathbb{1} + \mathbb{1} \otimes \Delta_h$ of bond dimension 6 in two spatial dimensions and $\Delta_h \otimes \mathbb{1} \otimes \mathbb{1} + \mathbb{1} \otimes \Delta_h \otimes \mathbb{1} + \mathbb{1} \otimes \mathbb{1} \otimes \Delta_h$ of bond dimension 9 in three spatial dimensions, where $\mathbb{1}$ denotes the identity operator for one spatial dimension.

Polynomial functions are written as matrix product states as shown in Fig. 5, see also references [21,31]. We consider a polynomial $q(x) = \sum_{l=0}^{\kappa} c_l x^l$ of degree κ. After discretization, the matrix product state

$$\sum_{\alpha_0,\alpha_1,\ldots,\alpha_L} Q[0]_{\alpha_0} Q[1]^{l_1}_{\alpha_0,\alpha_1} Q[2]^{l_2}_{\alpha_1,\alpha_2} \ldots Q[L]^{l_L}_{\alpha_{L-1},\alpha_L} Q[L+1]_{\alpha_L}$$

for such a polynomial has bond dimension $\kappa+1$. The tensors have the entries $Q[0]_1 = 1$, $Q[\ell]^0_{\alpha,\alpha} = 1 = Q[\ell]^1_{\alpha,\alpha} \forall \alpha \in \{1,2,\ldots,\kappa+1\}$ and $Q[\ell]^1_{\alpha,\beta} = \binom{\beta-1}{\alpha-1} \cdot 2^{-(\beta-\alpha)\ell} \forall \alpha < \beta : \beta \in \{2,3,\ldots,\kappa+1\}$ for $0 < \ell < L+1$, $L[L+1]_\alpha = c_{\alpha-1} \forall \alpha \in \{1,2,\ldots,\kappa+1\}$, and all other tensor entries are zero. Here, we construct higher-dimensional polynomial functions from the matrix product state Q_h for one spatial dimension. We use $Q^x_h \otimes Q^y_h$ in two and $Q^x_h \otimes Q^y_h \otimes Q^z_h$ in three spatial dimensions. Therefore our higher-dimensional polynomials are separable products of one-dimensional polynomials, which suffice for this analysis here.

Here we compare the two cost functions $c_1(|f\rangle) = \| \Delta|f\rangle - |g\rangle \|^2$ and $c_2(|f\rangle) = \langle f|\Delta|f\rangle - 2\langle f|g\rangle$. Note that the first cost function can be written out as $c_1(|f\rangle) = \langle f|\Delta^T\Delta|f\rangle - \langle f|\Delta^T|g\rangle - \langle g|\Delta|f\rangle + \langle g|g\rangle$ – where the superscript T denotes transposition – and is minimal and zero for the solution of the Poisson equation. The second cost function is minimal and possibly non-zero (having the value $-\langle f|g\rangle$) for the solution of the Poisson equation. This statement is true because the gradient of c_2 with respect to $|f\rangle$ is $2\Delta|f\rangle - 2|g\rangle$ and vanishes at the minimum $|f^{\min}\rangle$ of c_2 such that $2\Delta|f^{\min}\rangle - 2|g\rangle = 0$ which is the desired solution of the Poisson equation $\Delta|f^{\min}\rangle = |g\rangle$. We minimize these cost functions using the alternating minimization procedure of reference [26]. During this procedure, the tensors of our tensor product state are updated one after another and in each tensor update the cost function is minimized variationally using only the parameters of the tensor that is updated while the parameters of all the other tensors are fixed. For each tensor update, we compare the pseudoinverse, steepest descent, and conjugate gradient. The pseudoinverse has the computational cost $\mathcal{O}(d^3\chi^6)$ and finds the optimal tensor parameters in one step. Steepest descent as well as conjugate gradient require ν steps to find the

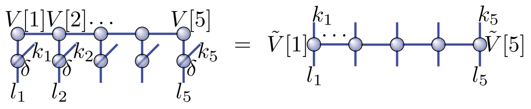

Fig. 6. The matrix product state $V[1]V[2]\dots V[5]$ is transformed into the matrix product operator $\tilde{V}[1]\tilde{V}[2]\dots\tilde{V}[5]$ that is diagonal and has the values of the matrix product state on its diagonal. This is achieved by contracting each individual tensor of the matrix product state with the δ tensor as shown. The δ tensor has the entries $\delta_{\alpha,\beta,\gamma} = 1 \forall \alpha = \beta = \gamma$ and all other entries are zero.

optimal tensor parameters and then their computational cost reads $\mathcal{O}(\nu d\chi^3) + \mathcal{O}(\nu d^2\chi^2)$. With steepest descent ν can get very large for the convergence of each tensor update to good tensor parameters. With conjugate gradient we make use of the fact that $-\Delta$ is positive definite – and symmetric – and then we can always find the optimal tensor parameters in each tensor update for $\nu = d\chi^2$.

The performance of the tensor train decomposition, i.e. matrix product states, in the context of the Poisson equation has been discussed in reference [32], however, without MGR. Here we want to identify the advantages of our MGR method. MGR starts from the exact solution on a coarse grid – i.e. the exact solution of the coarse grained finite difference problem – and then successively refines the grid and finds solutions on the refined grids, until the solution for the desired number of grid points $N = 2^L$ is found. In this article, our MGR method always starts from $L = 3$ and tensors with random entries, and we choose $\chi \geq d$ and perform alternating minimization until convergence to the exact solution. We then repeatedly implement prolongation and minimization operations, as described in section 2.1, until the desired value of L has been reached. We compare our MGR method to the usual procedure which starts directly at the desired final value of L, uses tensors with random entries, and then performs alternating minimization. We define that the alternating minimization procedure has converged when the relative change of the cost function is smaller than a given convergence precision. In all our numerical experiments, we observe that, for a given convergence precision, MGR converges faster than the usual procedure and to lower final values of the cost function. E.g. for convergence precision 10^{-4} with MGR, when the conjugate gradient method is used for the tensor update, and we consider a problem in one spatial dimension with a random polynomial $|g\rangle_h$, we find that the number of required alternating minimization sweeps is ≈ 10 for $L = 6$, 12 for $L = 7$, 21 for $L = 8$, 29 for $L = 9$, and 39 for $L = 10$. For the same convergence precision without MGR and the same problem, we find that the number of required alternating minimization sweeps is ≈ 29 for $L = 6$, 57 for $L = 7$, and > 100 for $L = 8$. Although the quantitative details of our analysis (e.g. the required number of sweeps and final value of the cost function) are problem-specific (and depend on the spatial dimensionality, $|g\rangle_h$, and so on), qualitatively we always observe the same: MGR improves the convergence and final result. We find that the advantages of MGR are most visible when steepest descent or conjugate gradient are used for the tensor update. In these cases, it is not possible for us to converge with the usual procedure when L is larger than ≈ 8. In contrast, MGR converges well for all values of L that we have considered, namely up to $L = 20$.

3. MGR for the ground state computation of the nonlinear Schrödinger equation

In this section MGR is used to develop a new algorithm for the ground state computation of the nonlinear Schrödinger equation. This equation is defined via a Hamiltonian operator that, after discretization, takes on the form

$$H(|f\rangle) = -\frac{1}{2}\Delta + V + g|f|^2. \tag{2}$$

H depends on the function $|f\rangle$ that it acts upon and thus represents a nonlinear function. Δ is the Laplace operator, V represents an external potential and $g|f|^2$ the interactions. Here we use this nonlinear Schrödinger equation to describe Bose–Einstein condensates and references [33,34] provide details on the physical meaning of the individual parts of H (note that equation (2) is also used in other contexts, e.g. nonlinear optics [35]). We are interested in the ground state, which is the eigenstate $|f\rangle$ corresponding to the smallest possible eigenvalue E in the equation $H(|f\rangle)|f\rangle = E|f\rangle$.

The individual parts of H in equation (2) can be written as tensor product operators. We already discussed the Laplace operator in section 2.2. We assume a simple form of the potential V which, most of the time, will be a polynomial that can be constructed as explained in section 2.2. This construction gives a matrix product state for V. For equation (2) we need to transform this matrix product state into a matrix product operator that has the values of the matrix product state on its diagonal. That is achieved with the help of a simple δ tensor which has the entries $\delta_{\alpha,\beta,\gamma} = 1 \forall \alpha = \beta = \gamma$ and all other entries are zero – note that this tensor is also known as the copy tensor [36,37]. Then the construction of V as a matrix product operator is straightforward as can be seen in Fig. 6. The interaction term $|f|^2$ needs to be constructed as a matrix product operator that has the modulus squared of the entries of $|f\rangle$ on its diagonal. Using the δ tensor this construction is also straightforward as shown in Fig. 7.

3.1. Quantifying the accuracy of MGR

A standard algorithm for the ground state computation of the nonlinear Schrödinger equation is imaginary time evolution [33,38]. In this section we use this algorithm to identify quantities for measuring the accuracy of MGR. Imaginary

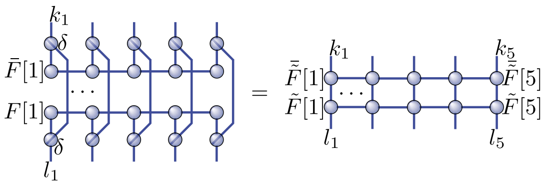

Fig. 7. The matrix product state $F[1]F[2]\dots F[5]$ is transformed into a tensor product operator that is diagonal and has the values of $|f|^2$ on its diagonal. Here $\bar{F}[1]\bar{F}[2]\dots\bar{F}[5]$ is the complex conjugate of $F[1]F[2]\dots F[5]$ and is constructed from the individual tensors $\bar{F}[\ell]$ that contain the complex conjugates of the entries of $F[\ell]$. The δ tensor is the same as in Fig. 6. We contract each individual tensor $F[\ell]$ and $\bar{F}[\ell]$ with a δ tensor to obtain $\tilde{F}[\ell]$ and $\tilde{\bar{F}}[\ell]$, respectively, as shown.

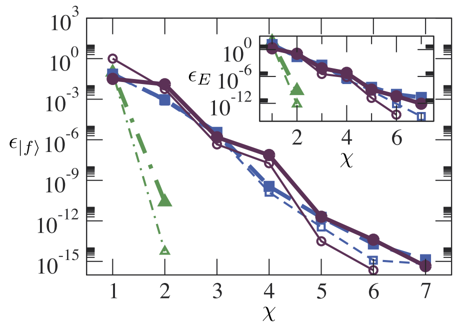

Fig. 8. Infidelity $\epsilon_{|f\rangle}(\chi) := 1 - |\langle f^{\chi}|f^{\text{exact}}\rangle|$ (main) and relative energy error $\epsilon_E(\chi) := |E^{\chi} - E^{\text{exact}}|/|E^{\text{exact}}|$ (inset) as functions of the bond dimension χ: We consider $g = 0$ (dash-dotted, triangles), $g = 10$ (dashed, squares), and $g = 100$ (solid, circles), for $L = 6$ (thin lines, open symbols), and $L = 8$ (thick lines, filled symbols). We observe that each error decreases exponentially with χ, for all values of g and L. We also observe that, for a given value of g, the errors are almost the same for both sizes $L = 6$ and 8. For $g = 0$ the numerically exact solution is obtained with $\chi = 2$. $\chi = 4$ to 7 are required to reach machine accuracy for $g = 10$ and 100.

time evolution is based on the fact that $e^{-tH}|f\rangle$ converges to the eigenstate of H corresponding to its smallest eigenvalue, i.e. the ground state, after long enough time t, if the initial state $|f\rangle$ is not orthogonal to that eigenstate. Note that it is important to normalize $|f\rangle$ such that $\langle f|f\rangle = 1$ is true during the propagation, as otherwise the norm of $|f\rangle$ can increase or decrease very quickly during imaginary time evolution. We split the total time t into N_τ small time steps τ, i.e. $t = N_\tau \tau$, such that $e^{-tH}|f\rangle = (e^{-\tau H})^{N_\tau}|f\rangle$ can be solved by applying the operator $e^{-\tau H}$ to the initial state $|f\rangle$ for N_τ times. We use the approximation $e^{-\tau H} \approx \mathbb{1} - \tau H$ with an approximation error $\mathcal{O}(\tau^2)$ for each time step τ. In imaginary time evolution the propagation for the initial time steps does not need to be accurate because we are only interested in an accurate final state of the evolution. So we start the evolution with a large value of τ for the initial steps and then successively decrease τ during the evolution such that the final steps are much more accurate than the initial ones. More precisely, we start the propagation with a large value of $\tau = 0.5$ (which is large for all H, i.e. g and V, considered here), converge the evolved state $|f\rangle$ by propagating it for large enough N_τ time steps, then decrease τ by a factor 1/2, converge $|f\rangle$, and so on, until all the results that we are interested in have converged. In our tensor product formalism, we find the evolved state $|\tilde{f}\rangle$ after each time step from the state $|f\rangle$ before the time step by minimizing the cost function $c(|\tilde{f}\rangle) = \| |\tilde{f}\rangle - (\mathbb{1} - \tau H)|f\rangle\|^2$ where H is defined in equation (2). We minimize this cost function by means of the alternating minimization procedure of reference [26]. Then we normalize the evolved state such that $\langle \tilde{f}|\tilde{f}\rangle = 1$ to avoid numerical problems. Thus for each time step, our time evolution algorithm has a computational cost that scales like $\mathcal{O}(Ld^2\chi^5)$. Note that the dominant contribution to the computational cost comes from the contraction of $|f|^2$.

Here we only want to analyse different quantities for measuring the accuracy of MGR and so we use a particularly simple external potential V in all numerical experiments of this section. Throughout this section, in equation (2), V is the box potential: i.e. V is 0 everywhere inside the interval $[0, 1)$ and infinitely large outside this interval.

When the exact solution $|f^{\text{exact}}\rangle$ is known, then we can compute the infidelity $\epsilon_{|f\rangle}(\chi) := 1 - |\langle f^{\chi}|f^{\text{exact}}\rangle|$ and the relative energy error $\epsilon_E(\chi) := |E^{\chi} - E^{\text{exact}}|/|E^{\text{exact}}|$. Here, we compute the exact reference solution $|f^{\text{exact}}\rangle$ by means of imaginary time evolution of a vector of dimension $N = 2^L$ which is initialized with random entries. The errors $\epsilon_{|f\rangle}$ and ϵ_E are shown in Fig. 8 for several values of g and L. We observe that all errors decrease exponentially as a function of χ. Because of this numerical evidence we can thus assume an upper bound of each error $\epsilon(\chi) \le c_0 \exp(-c_1\chi)$ where c_0 and c_1 are positive and real. This implies that, to achieve a certain error, i.e. accuracy, ϵ, we need to choose χ as

$$\chi(\epsilon) = \left\lceil \left(\log_e(c_0) - \log_e(\epsilon)\right)/c_1 \right\rceil \tag{3}$$

where $\lceil$ and $\rceil$ denote the ceiling function and we assume that $\epsilon < 1$ such that $-\log_e(\epsilon) > 0$. To gain a more intuitive understanding of equation (3), we rewrite $-\log_e(\epsilon) = -log_{10}(\epsilon)/log_{10}(e)$, because we can understand $-log_{10}(\epsilon)$ as the number of digits that are correct for accuracy ϵ. Then equation (3) reads $\chi(\epsilon) = \lceil \tilde{c}_0 - \tilde{c}_1 \log_{10}(\epsilon) \rceil$ where $\tilde{c}_0 = \log_e(c_0)/c_1$

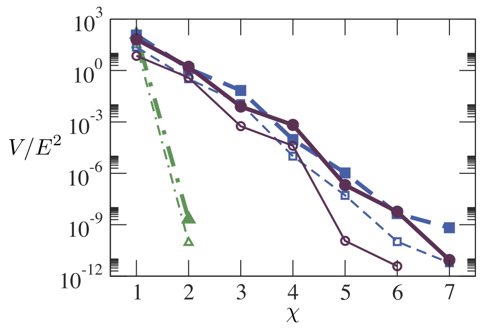

Fig. 9. Normalized variance $V/E^2(\chi) := \langle f^\chi|H^2|f^\chi\rangle/\langle f^\chi|H|f^\chi\rangle^2 - 1$ as a function of the bond dimension χ: We consider the same systems as in Fig. 8. We observe that the normalized variance qualitatively behaves the same as the quantities shown in Fig. 8. This suggests that the quantity shown here can be used to identify convergence of our method when the exact solution $|f^{\text{exact}}\rangle$ is not known.

and $\tilde{c}_1 = 1/(c_1 \log_{10}(e))$. We conclude that, if the error ϵ decreases exponentially as a function of χ, then χ is a linear function of $-\log_{10}(\epsilon)$, i.e. χ grows linearly with the number of correct digits in the solution. Equivalently, in this case, if we set the target accuracy ϵ, then the required χ to achieve this accuracy is a linear function of $-\log_{10}(\epsilon)$. Furthermore we can see in Fig. 8 that the errors do not depend significantly on L. Although we only compare $L = 6$ and 8 (because we can only compute the exact solutions $|f^{\text{exact}}\rangle$ for these relatively small values of L), we can clearly expect similar error functions for larger values of L from the results shown in Fig. 8. The fact that ϵ is independent of L implies that, for a given value of χ we obtain the same accuracy for all possible values of L, i.e. number of grid points $N = 2^L$. We conclude that tensor product states are an efficient ansatz for the problems considered here.

For large values of L, i.e. number of grid points $N = 2^L$, we do not know $|f^{\text{exact}}\rangle$ and therefore cannot compute $\epsilon_{|f\rangle}$ or ϵ_E. So we need to come up with alternative quantities to estimate the accuracy of tensor product state approximations. The normalized variance $V/E^2(\chi) := \langle f^\chi|H^2|f^\chi\rangle/\langle f^\chi|H|f^\chi\rangle^2 - 1$ vanishes for eigenstates of H. In reference [39] such a variance was used to study ground state convergence with tensor product states. Fig. 9 indicates that also in our case the normalized variance can be used to analyze convergence to the ground state. However, it is important to keep in mind that this quantity measures convergence to an eigenstate which does not have to be the ground state. Furthermore, in our case the computation of $\langle f^\chi|H^2|f^\chi\rangle$ has a high computational cost which is dominated by the scaling $\mathcal{O}(d^2\chi^7)$ of the contraction for $\langle f^\chi||f|^2|f|^2|f^\chi\rangle$. This high computational cost makes the computation of the normalized variance difficult for large values of χ and so we will not use this quantity in the remainder of this article. Nevertheless this can be a useful quantity in the context of other (nonlinear) partial differential equations if it can be computed more efficiently there.

The quantities that we use later to analyse the accuracy of MGR follow from a transformation of our matrix product state of equation (1) into its canonical form [17,18,40]

$$|f\rangle = \sum_{l_1,l_2,\ldots,l_L} \sum_{\alpha_1,\alpha_2,\ldots,\alpha_{L-1}} \left(\Gamma[1]^{l_1}_{\alpha_1}\lambda[1]_{\alpha_1}\Gamma[2]^{l_2}_{\alpha_1,\alpha_2}\lambda[2]_{\alpha_2}\ldots\Gamma[L]^{l_L}_{\alpha_{L-1}}\right)|l_1\rangle|l_2\rangle\ldots|l_L\rangle \tag{4}$$

where each matrix $\lambda[l]$ is diagonal with nonnegative entries that correspond to the Schmidt coefficients for the bipartition of the state between $1, 2, \ldots, l$ and $l+1, l+2, \ldots, L$. A general indicator for the convergence of a state $|f^\chi\rangle$ with increasing χ is the convergence of Schmidt coefficients $\lambda[l]_\alpha$ and von Neumann entropies $S(l) := -\sum_\alpha(\lambda^2[l]_\alpha \log_2(\lambda^2[l]_\alpha))$ for all bipartitions of the state. Notice that $S(l)$ is also known as the entanglement entropy $S(l) := -\mathrm{tr}(\rho(l)\log_2(\rho(l)))$ where $\rho(l) := \mathrm{tr}_{1,2,\ldots,l}(|f\rangle\langle f|)$. Fig. 10 shows example entanglement entropies and Schmidt coefficients for some of our problems. With the help of the squared Schmidt coefficients $\lambda^2[l]_\alpha$, we can estimate the error [19,41]

$$\||f^{\text{exact}}\rangle - |f^\chi\rangle\|^2 \leq 2\sum_{l=1}^{L-1}\sum_{\alpha_l=\chi+1}^{\min(d^l,d^{L-l})} \lambda^2[l]_{\alpha_l}. \tag{5}$$

Although we do not know the Schmidt coefficients $\lambda[l]_\alpha$ for $\alpha > \chi$, we do know them for $\alpha = 1$ to χ and can extrapolate their behavior for $\alpha = \chi + 1$ to $\min(d^l, d^{L-l})$. We can convince ourselves that our extrapolation is correct by increasing χ and checking that the new Schmidt coefficients lie on our extrapolated curve. We have thoroughly investigated this procedure of extrapolating and checking the behavior of Schmidt coefficients, and in all cases our extrapolations were very accurate. Therefore quickly decreasing Schmidt coefficients are an indicator for good accuracy of a matrix product state approximation.

Moreover, for all the problems considered here, we observe that the Schmidt coefficients for all bipartitions decrease exponentially as a function of χ. We thus assume that real and positive numbers c_0 and c_1 exist such that $\lambda[l]_\alpha \leq c_0\exp(-c_1\alpha)$ $\forall l$. Then the error of equation (5) has the upper bound

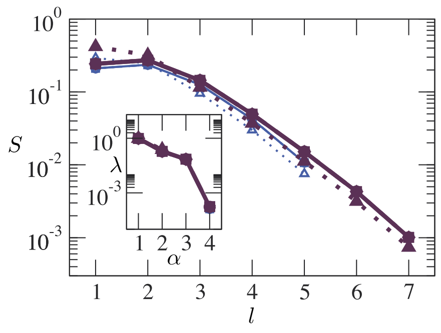

Fig. 10. Entanglement entropy $S(l) := -\mathrm{tr}(\rho(l)\log_2(\rho(l)))$ where $\rho(l) := \mathrm{tr}_{1,2,\ldots,l}(|f\rangle\langle f|)$ as a function of the level l (main) and Schmidt coefficients $\lambda[l]_\alpha$ for level $l = 2$ (inset): We consider $g = 100$, for $L = 6$ (thin lines, open symbols) and 8 (thick lines, filled symbols), and $\chi = 2$ (dotted, triangles), 3 (dash-dotted, diamonds), 4 (dashed, squares), and $\chi_{\max}$ (solid, circles), where $\chi_{\max} = 8$ for $L = 6$ and $\chi_{\max} = 16$ for $L = 8$. In the inset, we show the Schmidt coefficients for level 2 because the exact solutions have the largest entanglement entropy for level 2. We observe that, for each value of L, all curves for $\chi \geq 3$ lie on top of each other. We also observe that, for each value of χ, our results for $L = 6$ and 8 are almost the same. We conclude that the problems considered here are only weakly entangled and can thus be described using low values of χ that are almost independent of the multigrid size L. Therefore, our method converges rapidly with increasing bond dimension χ, almost independently of L.

$$\begin{aligned}
\||f^{\mathrm{exact}}\rangle - |f^{\chi}\rangle\|^2 &< 2c_0^2(L-1)\sum_{\alpha=\chi+1}^{\infty}\exp(-2c_1\alpha) \\
&= 2c_0^2(L-1)\exp(-2c_1(\chi+1))\sum_{\alpha=0}^{\infty}\exp(-2c_1\alpha) \\
&= 2c_0^2(L-1)\exp(-2c_1(\chi+1))/(1-\exp(-2c_1)) \\
&= 2c_0^2(L-1)\exp(-2c_1\chi)/(\exp(2c_1)-1) \\
&= \tilde{c}_0(L-1)\exp(-\tilde{c}_1\chi)
\end{aligned}$$

where $\tilde{c}_0 = 2c_0^2/(\exp(2c_1)-1)$ and $\tilde{c}_1 = 2c_1$. We conclude that if all Schmidt coefficients decrease exponentially as a function of χ, then also the error defined in equation (5) decreases exponentially as a function of χ. Using equation (3), this implies that χ depends linearly on $-\log(\epsilon)$ – where ϵ now denotes the error defined in equation (5) – and thus the required χ depends linearly on the number of desired correct digits.

Looking at all Schmidt coefficients is more informative than looking at just the von Neumann entropies. Quick convergence of von Neumann entropies with increasing χ is a necessary but not sufficient criterion for good accuracy of a matrix product state approximation [42].

3.2. Direct energy minimization via nonlinear tensor updates

Imaginary time evolution is a reliable but inefficient algorithm for ground state computation, because the ground state is obtained in an indirect way via propagation in imaginary time for typically many time steps. Direct minimization of the energy is more efficient as it aims at the ground state as directly as possible. For the nonlinear Schrödinger equation with the Hamiltonian of equation (2), the energy is $E(|f\rangle) := \langle f|(-\frac{1}{2}\Delta + V + g|f|^2)|f\rangle$ for normalized functions $|f\rangle$, i.e. that fulfill $\langle f|f\rangle = 1$. The ground state minimizes $\langle f|(-\frac{1}{2}\Delta + V + \frac{g}{2}|f|^2)|f\rangle$ under the constraint $\langle f|f\rangle = 1$ (notice that this ground state is equivalent to the normalized eigenstate of the Hamiltonian in equation (2) corresponding to the smallest eigenvalue) [43].

We investigated several different methods for solving this constrained minimization problem from reference [44]. The method that worked best substitutes the constrained minimization problem by the unconstrained minimization of a penalty function $P := \langle f|(-\frac{1}{2}\Delta + V + \frac{g}{2}|f|^2)|f\rangle + \eta(\langle f|f\rangle - 1)^2$ where η is a new parameter. In the limit $\eta \to \infty$ the minimum of this unconstrained minimization problem coincides with the minimum of the original constrained minimization problem. Notice that a penalty function is a special cost function that results from replacing a constrained minimization problem by an unconstrained minimization problem and that depends on a penalty parameter in such a way that the desired solution is obtained when this penalty parameter is infinite. To compute the ground state, we therefore need to minimize P for successively growing values of η until the energy $E(|f\rangle)$ has converged to its smallest value and $\langle f|f\rangle = 1$ is fulfilled well enough. For each value of η, we minimize P via an alternating minimization procedure that sweeps over the tensors and for each tensor minimizes P via the variational parameters of this tensor only, i.e. keeping the parameters of all other tensors fixed. Here each tensor update is a nonlinear optimization problem. In the following we compare steepest descent, Fletcher–Reeves conjugate gradient, Polak–Ribière conjugate gradient, and Newton method when solving the nonlinear equations. Additionally we compare the direct minimization of the penalty function with MGR and without MGR.

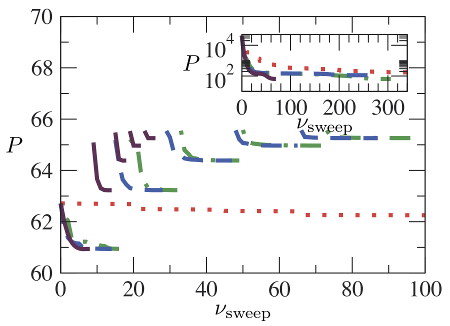

Fig. 11. Penalty function $P := \langle f|(-\frac{1}{2}\Delta + V + \frac{g}{2}|f|^2)|f\rangle + \eta(\langle f|f\rangle - 1)^2$ as a function of the number of sweeps ν_{sweep} – where during each sweep we perform one tensor update of each tensor in $|f\rangle$ going from left to right and then we perform one tensor update of each tensor in $|f\rangle$ going from right to left – for our MGR method (main) and without our MGR method (inset). We consider the box potential V, with $g = 100$, and set $\eta = 10^8$, $L = 8$, and $\chi = 4$. We compare four different methods for the nonlinear tensor update: Each tensor update consists of $\nu_{\text{update}} = 4$ steepest descent steps (dotted), $\nu_{\text{update}} = 4$ Fletcher–Reeves conjugate gradient steps (dash-dotted), $\nu_{\text{update}} = 4$ Polak–Ribière conjugate gradient steps (dashed), and $\nu_{\text{update}} = 1$ Newton method step (solid). Main (our MGR method): We start from the exact solution for $L = 3$ (from imaginary time evolution), prolong it to $L = 4$, minimize the penalty function for $L = 4$ until convergence, prolong the solution to $L = 5$, minimize the penalty function for $L = 5$ until convergence, prolong the solution to $L = 6$, and so on. For each L the first plotted value of P is computed directly after the prolongation. E.g. for the Newton method (solid), the first continuous line (from $\nu_{\text{sweep}} = 0$ to ≈ 10) corresponds to $L = 4$, the second continuous line (from $\nu_{\text{sweep}} \approx 10$ to ≈ 15) corresponds to $L = 5$, the third continuous line (from $\nu_{\text{sweep}} \approx 15$ to ≈ 20) corresponds to $L = 6$, the fourth continuous line (from $\nu_{\text{sweep}} \approx 20$ to ≈ 25) corresponds to $L = 7$, and the fifth continuous line (from $\nu_{\text{sweep}} \approx 25$ to ≈ 30) corresponds to $L = 8$. Inset: We start from a random state for $L = 8$ and minimize the penalty function until convergence. We define convergence when the relative change of the penalty function from one sweep to the next is smaller than 10^{-6}.

Our results are shown in Fig. 11. We observe that our MGR method converges reliably as all four methods of the nonlinear tensor update considered here converge to the same final result of P within reasonable relative precisions. We take the final result of the Newton method as the reference value as it is the lowest value of P. Then we obtain a relative precision of $2 \cdot 10^{-3}$ for steepest descent, $4 \cdot 10^{-5}$ for Fletcher–Reeves conjugate gradient, and $2 \cdot 10^{-5}$ for Polak–Ribière conjugate gradient. The steepest descent method requires the largest number of sweeps (more than 2000) for convergence and converges to the least accurate final result (i.e. the highest value of P). Both conjugate gradient methods require similar numbers of sweeps for convergence (around 90) and converge to similar final results. The Polak–Ribière is a little bit faster and more precise than the Fletcher–Reeves conjugate gradient method. The Newton method requires the smallest number of sweeps for convergence and gives the most accurate final result (i.e. the lowest value of P). Notice that each time L is increased by 1 the penalty function converges to a larger value. This is not an indication of the result getting worse but arises because of the normalization of $|f\rangle$ changing with L.

Each time L is increased by 1, the Newton method requires approximately 5 sweeps until convergence while each conjugate gradient method requires approximately 20 sweeps until convergence. Firstly, this suggests that our MGR method requires a number of sweeps that grows linearly with L: $\nu_{\text{sweep}} \propto L$ (notice that the accuracy of our finite discretization method grows exponentially with L). Secondly, this suggests that the Newton method converges the fastest: The computational cost of each tensor update is $\mathcal{O}(\chi^6)$ in the Newton method and $\mathcal{O}(\chi^5)$ in both conjugate gradient methods, and we performed $\nu_{\text{update}} = 4$ $(= \chi)$ updates per tensor for both conjugate gradient methods and $\nu_{\text{update}} = 1$ update per tensor for the Newton method. We can read off from the plot that, still, each conjugate gradient method requires approximately 4 times more sweeps than the Newton method: Thus, in total, each conjugate gradient method needs approximately 16 $(= \chi^2)$ times more tensor updates than the Newton method.

We can see in the inset of Fig. 11 that without our MGR method no reliable convergence occurs. The four methods considered here for the nonlinear tensor update do not converge to the same values of P. Both the steepest descent and the Polak–Ribière conjugate gradient method converge to relatively high values of P, i.e. the methods possibly get stuck in high-lying local minima of P. The Newton method achieves the lowest value of P and this value coincides with the one obtained from our MGR method within a relative precision of $5 \cdot 10^{-11}$. Without our MGR method significantly more sweeps are required for convergence than with our MGR method.

Convergence can be quantified via the norm of the normalized gradient of the penalty function $\|\delta P/P\| := \sqrt{\sum_{l=1}^{L} \|\partial P/\partial \vec{F}[l]\|^2}/|P|$. This quantity is shown in Fig. 12. We draw the same conclusion from Fig. 12 as we did from Fig. 11: The combination that works best is MGR with the Newton method for the nonlinear tensor update. This combination is applied to a finer grid, namely $L = 20$, in Fig. 13. Fig. 13 illustrates a fact that is true for all problems considered in this article: For a given convergence precision the number of sweeps required by our MGR method scales linearly with L. Therefore our MGR method features an exponential speedup compared to the original MG method, namely from a computational cost scaling like $\mathcal{O}(N)$ to $\mathcal{O}(L) = \mathcal{O}(\log(N))$.

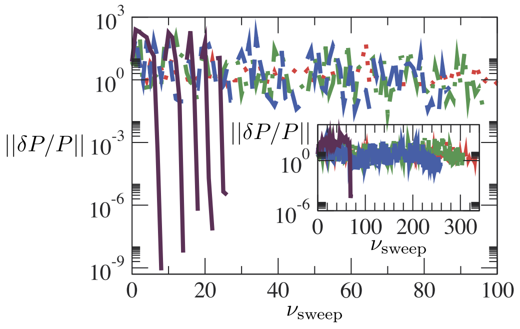

Fig. 12. Norm of the normalized gradient of the penalty function $\|\delta P/P\| := \sqrt{\sum_{l=1}^{L} \|\partial P/\partial \vec{F}[l]\|^2}/|P|$ as a function of the number of sweeps ν_{sweep} for our MGR method (main) and without our MGR method (inset). We consider the same systems as in Fig. 11. Both with and without our MGR method, the Newton method converges to a small penalty function gradient, i.e. a proper extremum of the penalty function. With our MGR method this convergence is more systematic, quicker, and finishes in a lower final value than without our MGR method. All the other tensor update methods converge less systematically, slower, and to significantly larger penalty function gradients than the Newton method.

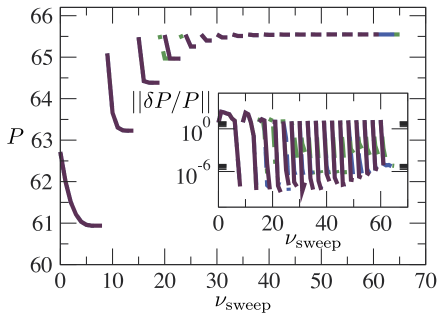

Fig. 13. Penalty function P (main) and norm of the normalized gradient of the penalty function $\|\delta P/P\|$ (inset) as a function of the number of sweeps ν_{sweep} for our MGR method with $\nu_{\text{update}} = 1$ Newton method step per tensor update. We consider the box potential V, with $g = 100$, set $\eta = 10^8$, and $L = 20$, and compare $\chi = 4$ (dash-dotted), 8 (dashed), and 16 (solid). For our MGR method we use the same procedure as in the main plot of Fig. 11 with the same convergence precision 10^{-6}. We observe that our results for the different values of χ lie on top of each other for all values of $L \leq 20$. This suggests that increasing χ further, i.e. setting $\chi > 16$, would have a negligible effect on all our results: Therefore our MGR method has converged with $\chi = 16$. We also observe that the convergence of our MGR method with increasing χ occurs for all values of $L \leq 20$ equally fast: This suggests that the convergence with χ is independent of L. Furthermore we read off from the plot that our MGR method requires a number of sweeps ν_{sweep} to converge that grows linearly with L: $\nu_{\text{sweep}} \propto L$.

3.3. Performance

We now analyse the performance of our MGR method by computing ground states of the nonlinear Schrödinger equation in one, two, and three spatial dimensions. We consider three different external potentials: a box, a well, and a double-well. These external potentials are defined and shown for one spatial dimension in Fig. 14. Notice that the behavior of the solution for a double-well potential depends on the ratio between barrier height and zero-point energy of the unperturbed single-well oscillator: if the barrier is too low, then we just have a perturbed (anharmonic) single-well problem, whereas if the barrier is too high, then the problem reduces to two well-isolated single-well problems. Our double-well potential is at the cross-over between these two regimes. Our external potentials for higher spatial dimensions are constructed from the ones for one spatial dimension by taking the same potential in each dimension. E.g. the double-well potential reads $V_{\text{dw}}(x) + V_{\text{dw}}(y)$ in two and $V_{\text{dw}}(x) + V_{\text{dw}}(y) + V_{\text{dw}}(z)$ in three spatial dimensions, where V_{dw} is defined in the caption of Fig. 14. The corresponding matrix product operators are constructed in the same way that we used for the higher-dimensional Laplace operator in section 2.2. E.g. for the double-well potential we use $V_{\text{dw}} \otimes \mathbb{1} + \mathbb{1} \otimes V_{\text{dw}}$ in two and $V_{\text{dw}} \otimes \mathbb{1} \otimes \mathbb{1} + \mathbb{1} \otimes V_{\text{dw}} \otimes \mathbb{1} + \mathbb{1} \otimes \mathbb{1} \otimes V_{\text{dw}}$ in three spatial dimensions, where V_{dw} is the matrix product operator for one spatial dimension. We choose these external potentials because the corresponding nonlinear Schrödinger equations describe Bose–Einstein condensates in laser potentials as they are currently realised in many experiments all around the world. Figs. 14 and 15 illustrate how our solutions look like for one and two spatial dimensions, respectively.

We investigate the convergence of our solution function $|f^{\chi}\rangle$ with increasing bond dimension χ and number of grid points N in Fig. 16. This allows us to give precise estimates for the ground state energies in continuous space, see Table 1. We observe in Fig. 16 that the entanglement entropy grows when the spatial dimension increases. This indicates that, to obtain a fixed accuracy for the solution, in higher spatial dimensions we need larger values of χ. Table 1 confirms this: For a fixed value of χ, the accuracy decreases with increasing spatial dimensionality. In this analysis we set the maximum χ to 24 for all spatial dimensions. However, we have converged results for $\chi = 64$ in one and $\chi = 48$ in two spatial dimensions. For all problems considered here, we obtained machine precision of our solutions in one and two spatial dimensions. Only

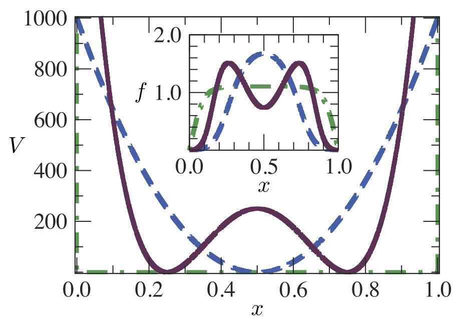

Fig. 14. Main: Potentials considered here: box $V_b(x)$ (dash-dotted), well $V_w(x) = 4000(x - 0.5)^2$ (dashed), and double-well $V_{dw}(x) = 250 - 8000(x - 0.5)^2 + 64000(x - 0.5)^4$ (solid). Inset: Ground state wave functions $f(x)$ of the one-dimensional nonlinear Schrödinger equation with $g = 100$ for the potentials of the main plot. We minimized the penalty function P with $\eta = 10^8$, using our MGR method with $\nu_{\text{update}} = 1$ Newton method step per tensor update and convergence precision 10^{-6}, i.e. the same procedure as in Fig. 13. These results correspond to $L = 20$ and $\chi = 24$. We systematically compared the results shown here with the corresponding results for $L = 12$ and 20 using $\chi = 12$, 16, and 20, and we found no significant differences between all these results. Therefore we assume that choosing $L > 20$ or $\chi > 24$ will have a negligible effect on the results shown here. Thus our MGR method has converged with increasing L and χ to the true continuous ground state wave function.

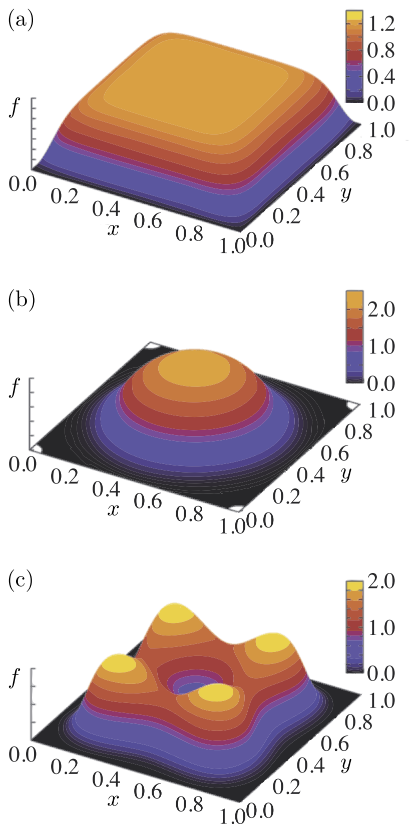

Fig. 15. Ground state wave functions $f(x, y)$ of the two-dimensional Schrödinger equation with $g = 100$ for the following potentials: box (a), well (b), and double-well (c) (as defined in Fig. 14). We minimized the penalty function P with $\eta = 10^8$, using our MGR method with $\nu_{\text{update}} = 1$ Newton method step per tensor update and convergence precision 10^{-6}, i.e. the same procedure as in Fig. 13. These results correspond to $L = 20$ and $\chi = 24$ restricted to $L = 10$. We systematically compared the results shown here with the corresponding results for $L = 12$ and 20 using $\chi = 12$, 16, and 20, and we found no significant differences between all these results. Therefore we assume that choosing $L > 20$ or $\chi > 24$ will have a negligible effect on the results shown here. Thus our MGR method has converged with increasing L and χ to the true continuous ground state wave function.

in three spatial dimensions, where $d = 8$, it is difficult to obtain converged results for $\chi > 24$ because of limited computing power. Nevertheless it is remarkable that such a small value of $\chi = 24$ suffices to describe $N = 8^{20} \approx 10^{18}$ grid points in three spatial dimensions and produces ground state energy accuracies that are better than one percent.

Our results in Fig. 16 and Table 1 indicate that for higher spatial dimensions a different tensor product state ansatz might be better suited. Two alternative ansatzes are presented in Fig. 17. A particularly promising ansatz is the finite projected

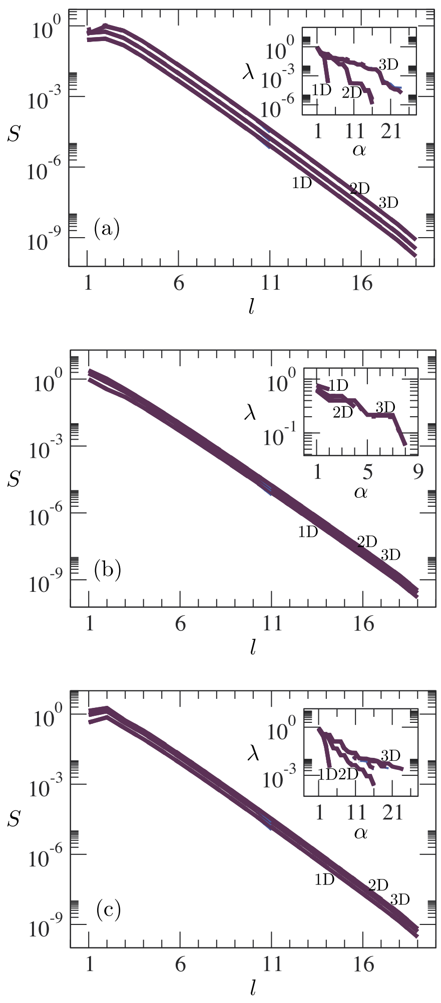

Fig. 16. Entanglement entropy $S(l) := -\mathrm{tr}(\rho(l)\log_2(\rho(l)))$ where $\rho(l) := \mathrm{tr}_{1,2,\ldots,l}(|f\rangle\langle f|)$ as a function of the level l (main) and Schmidt coefficients $\lambda[l]_\alpha$ (inset) for level $l = 2$ (a), 1 (b), and 2 (c): We consider the box $\hat{V}_\mathrm{b}$ (a), well $\hat{V}_\mathrm{w}$ (b), and double-well $\hat{V}_\mathrm{dw}$ (c) (as defined in Fig. 14), with $g = 100$ for $L = 12$ (thin lines) and 20 (thick lines), and $\chi = 12$ (dotted), 16 (dash-dotted), 20 (dashed), and 24 (solid). In the inset, we show the Schmidt coefficients for the level with the largest entanglement entropy. These results were obtained via minimization of the penalty function P with $\eta = 10^8$, using our MGR method with $\nu_\mathrm{update} = 1$ Newton method step per tensor update and convergence precision 10^{-6}, i.e. the same procedure as in Fig. 13. We observe that in 1D and 2D all curves lie on top of each other. In 3D the curves corresponding to $\chi > 16$ lie on top of each other and only the $\chi = 12$ and 16 curves can be slightly off. We conclude that the convergence of our MGR method with χ is independent of L and that our MGR method has converged with $\chi = 24$, i.e. setting $\chi > 24$ will have a negligible effect on all our results.

Table 1
Ground state energies E for the nonlinear Schrödinger equation with $g = 100$, for the problems described in the text. We minimized the penalty function P with $\eta = 10^8$, using our MGR method with $\nu_\mathrm{update} = 1$ Newton method step per tensor update and convergence precision 10^{-6}, i.e. the same procedure as in Fig. 13. The number in brackets denotes the uncertainty of the last digit. E.g. for 1D Box $E = 122.09942(7)$ means that the true result is in the interval [122.09935, 122.09949]. We construct these numbers from the ground state energies for $L = 20$ and $\chi = 24$ (for 1D Box: $E = 122.09942$) and the corresponding ground state energies for $L = 19$ and $\chi = 20$ (for 1D Box: $E = 122.09935$). We assume that the energy error decreases sufficiently fast with increasing number of grid points N and bond dimension χ. Then the energy values provided here with error bars represent the true ground state energies for continuous space $N \to \infty$ and infinite bond dimension $\chi \to \infty$.

Dimensionality	Box	Well	Double-well
1D	122.09942(7)	288.05273(3)	264.67755(3)
2D	145.0192(2)	515.2060(2)	444.6485(5)
3D	197.0(3)	759.5(5)	640(1)

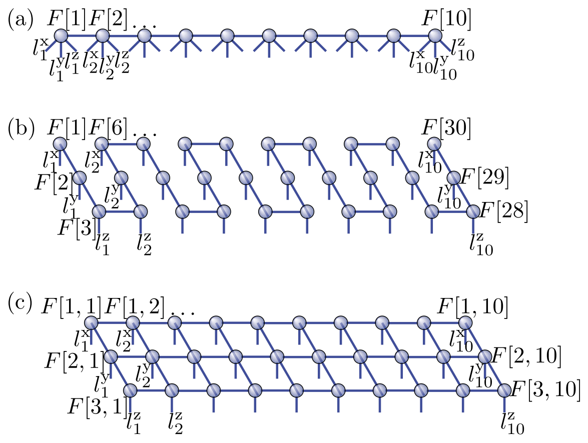

Fig. 17. Different tensor product state ansatzes for three spatial dimensions. (a) Matrix product state, as used in this article. (b) Matrix product state, constructed as for quantum many-body systems in two spatial dimensions [47]. Compared to (a), (b) has the advantage that $d = 2$ – and not $d = 8$ as in (a) – and therefore has a lower computational cost. (c) Projected entangled pair state [45,46]. Compared to (a) and (b), (c) has the advantage that it can capture larger von Neumann entropies. In particular, (c) can capture a larger increase of von Neumann entropies with the spatial dimension.

entangled pair state [45,46] shown in Fig. 17 (c). This ansatz can capture much larger von Neumann entropies in higher spatial dimensions than matrix product states.

4. Conclusions and outlook

We have shown that MG methods for solving (nonlinear) partial differential equations generate functions in the form of tensor product states. Therefore these methods can alternatively be formulated as variational algorithms over tensor product states. This has the advantage that finer grids can be reached than with previous MG methods. We have constructed our MGR method as a natural combination of MG and variational renormalization group concepts. Our MGR method generalizes and extends the MG method. Moreover, our MGR method has a computational cost that scales like $\mathcal{O}(\log(N))$ for N grid points, whereas for the fastest previous MG methods the computational cost scales like $\mathcal{O}(N)$. We have developed MGR algorithms for the linear Poisson and nonlinear Schrödinger equation. For all problems considered in this article, we have verified that the computational cost indeed scales like $\mathcal{O}(\log(N))$ and, thus, our MGR method is exponentially faster than previous MG methods.

In this article, we have only used matrix product states as our tensor product state ansatz for functions in any spatial dimension. We have identified Schmidt coefficients and von Neumann entropies as useful quantifiers for the accuracy of a solution. For our ground state solutions of the nonlinear Schrödinger equation, we have often seen a significant increase of von Neumann entropies with the spatial dimension. This can be better captured by another tensor product state ansatz called projected entangled pair states [45,46]. It would be very interesting to apply the state-of-the-art finite projected entangled pair state algorithms of references [48,49] to partial differential equations.

In all our algorithms, we have found that the Newton method is more efficient and accurate for nonlinear tensor updates than steepest descent and conjugate gradient based approaches. This might also be the case in other tensor product state algorithms. For example, infinite projected entangled pair states [50] are composed of repeating unit cells of tensors. So far, most algorithms that use this ansatz to approximate ground states are based on imaginary time evolution. Only recently algorithms have been proposed that rely on the more efficient direct energy minimization, see e.g. references [39,51,52]. In this case, nonlinear tensor updates are required and we believe that the Newton method could accelerate the existing algorithms.

All problems that we have considered in this article have had relatively smooth solutions. It would be interesting to analyze the performance of our MGR method in other examples like e.g. turbulence (see e.g. reference [53] for details on how turbulence can be observed in experiments described by nonlinear Schrödinger equations). The optimal tensor product state algorithm for dynamical problems is known [54,55]. And so it would be straightforward to analyze the performance of our MGR method for such time-dependent problems. However, there are many other interesting applications for MGR. For example, it would be exciting to apply MGR to the systematic construction of new density functionals for density functional theory, using the methods developed in reference [56]. While the scenario of reference [56] was limited to one spatial dimension and a relatively rough grid for the spatial coordinates as well as the density values, MGR can handle continuous functions in three spatial dimensions. By combining MGR with the procedure presented in reference [56], we could construct a hierarchy of successively more precise exchange-correlation energy functionals for continuous density values in three spatial dimensions: an outstanding goal in density functional theory. Another exciting future direction of research would be to address the natural question whether or not it is possible to apply our MGR method to the computation of excited states.

Acknowledgements

ML and DJ acknowledge funding from the NQIT (Networked Quantum Information Technologies) Hub of the UK National Quantum Technology Programme and from the EPSRC Tensor Network Theory Grant (EP/K038311/1). All tensor product algorithms of this article were programmed using the Tensor Network Theory Library [57].

References

[1] A. Brandt, Multi-level adaptive solutions to boundary-value problems, Math. Comput. 31 (1977) 333–390.
[2] W. Hackbusch, Multi-Grid Methods and Applications, Springer-Verlag, New York, 1985.
[3] K.G. Wilson, The renormalization group: critical phenomena and the Kondo problem, Rev. Mod. Phys. 47 (1975) 773–840.
[4] R. Bulla, T.A. Costi, T. Pruschke, Numerical renormalization group method for quantum impurity systems, Rev. Mod. Phys. 80 (2008) 395–450.
[5] S.R. White, Density matrix formulation for quantum renormalization groups, Phys. Rev. Lett. 69 (1992) 2863–2866.
[6] S.R. White, Density-matrix algorithms for quantum renormalization groups, Phys. Rev. B 48 (1993) 10345–10356.
[7] U. Schollwöck, The density-matrix renormalization group, Rev. Mod. Phys. 77 (2005) 259–315.
[8] S. Östlund, S. Rommer, Thermodynamic limit of density matrix renormalization, Phys. Rev. Lett. 75 (1995) 3537–3540.
[9] F. Verstraete, V. Murg, J.I. Cirac, Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems, Adv. Phys. 57 (2008) 143–224.
[10] J.I. Cirac, F. Verstraete, Renormalization and tensor product states in spin chains and lattices, J. Phys. A, Math. Theor. 42 (2009) 504004.
[11] U. Schollwöck, The density-matrix renormalization group in the age of matrix product states, Ann. Phys. 326 (2011) 96–192.
[12] A. Weichselbaum, F. Verstraete, U. Schollwöck, J.I. Cirac, J. von Delft, Variational matrix-product-state approach to quantum impurity models, Phys. Rev. B 80 (2009) 165117.
[13] I. Pižorn, F. Verstraete, Variational numerical renormalization group: bridging the gap between NRG and density matrix renormalization group, Phys. Rev. Lett. 108 (2012) 067202.
[14] S.R. White, R.M. Noack, Real-space quantum renormalization groups, Phys. Rev. Lett. 68 (1992) 3487–3490.
[15] R. Orús, A practical introduction to tensor networks: matrix product states and projected entangled pair states, Ann. Phys. 349 (2014) 117–158.
[16] M. Fannes, B. Nachtergaele, R.F. Werner, Finitely correlated states on quantum spin chains, Commun. Math. Phys. 144 (1992) 443–490.
[17] G. Vidal, Efficient classical simulation of slightly entangled quantum computations, Phys. Rev. Lett. 91 (2003) 147902.
[18] D. Pérez-García, F. Verstraete, M.M. Wolf, J.I. Cirac, Matrix product state representations, Quantum Inf. Comput. 7 (2007) 401–430.
[19] I.V. Oseledets, Tensor-train decomposition, SIAM J. Sci. Comput. 33 (2011) 2295–2317.
[20] I.V. Oseledets, Approximation of $2^d \times 2^d$ matrices using tensor decomposition, SIAM J. Matrix Anal. Appl. 31 (2010) 2130–2145.
[21] B.N. Khoromskij, $\mathcal{O}(d\log(N))$-quantics approximation of N–d tensors in high-dimensional numerical modeling, Constr. Approx. 34 (2011) 257–280.
[22] S. Wiesner, Simulations of many-body quantum systems by a quantum computer, arXiv:quant-ph/9603028v1, 1996.
[23] C. Zalka, Threshold estimate for fault tolerant quantum computing, arXiv:quant-ph/9612028v2, 1997.
[24] C. Zalka, Simulating quantum systems on a quantum computer, Proc. R. Soc. Lond. A 454 (1998) 313–322.
[25] W. Hackbusch, U. Trottenberg, Multigrid Methods, Springer-Verlag, New York, 1982.
[26] F. Verstraete, D. Porras, J.I. Cirac, Density matrix renormalization group and periodic boundary conditions: a quantum information perspective, Phys. Rev. Lett. 93 (2004) 227205.
[27] A.W. Sandvik, G. Vidal, Variational quantum Monte Carlo simulations with tensor-network states, Phys. Rev. Lett. 99 (2007) 220602.
[28] N. Schuch, M.M. Wolf, F. Verstraete, J.I. Cirac, Simulation of quantum many-body systems with strings of operators and Monte Carlo tensor contractions, Phys. Rev. Lett. 100 (2008) 040501.
[29] F. Verstraete, J.J. García-Ripoll, J.I. Cirac, Matrix product density operators: simulation of finite-temperature and dissipative systems, Phys. Rev. Lett. 93 (2004) 207204.
[30] V.A. Kazeev, B.N. Khoromskij, Low-rank explicit QTT representation of the Laplace operator and its inverse, SIAM J. Matrix Anal. Appl. 33 (2012) 742–758.
[31] I.V. Oseledets, Constructive representation of functions in low-rank tensor formats, Constr. Approx. 37 (2013) 1–18.
[32] I.V. Oseledets, S.V. Dolgov, Solution of linear systems and matrix inversion in the TT-format, SIAM J. Sci. Comput. 34 (2012) A2718–A2739.
[33] F. Dalfovo, S. Giorgini, L.P. Pitaevskii, S. Stringari, Theory of Bose–Einstein condensation in trapped gases, Rev. Mod. Phys. 71 (1999) 463–512.
[34] L. Pitaevskii, S. Stringari, Bose–Einstein Condensation, Clarendon Press, Oxford, 2003.
[35] A. Scott, Encyclopedia of Nonlinear Science, Routledge, New York, 2005.
[36] S. Al-Assam, S.R. Clark, C.J. Foot, D. Jaksch, Capturing long range correlations in two-dimensional quantum lattice systems using correlator product states, Phys. Rev. B 84 (2011) 205108.
[37] S.J. Denny, J.D. Biamonte, D. Jaksch, S.R. Clark, Algebraically contractible topological tensor network states, J. Phys. A, Math. Theor. 45 (2012) 015309.
[38] M. Edwards, K. Burnett, Numerical solution of the nonlinear Schrödinger equation for small samples of trapped neutral atoms, Phys. Rev. A 51 (1995) 1382.
[39] L. Vanderstraeten, J. Haegeman, P. Corboz, F. Verstraete, Gradient methods for variational optimization of projected entangled-pair states, Phys. Rev. B 94 (2016) 155123.
[40] F. Verstraete, J.I. Cirac, J.I. Latorre, E. Rico, M.M. Wolf, Renormalization-group transformations on quantum states, Phys. Rev. Lett. 94 (2005) 140601.
[41] F. Verstraete, J.I. Cirac, Matrix product states represent ground states faithfully, Phys. Rev. B 73 (2006) 094423.
[42] N. Schuch, M.M. Wolf, F. Verstraete, J.I. Cirac, Entropy scaling and simulability by matrix product states, Phys. Rev. Lett. 100 (2008) 030504.
[43] E.H. Lieb, J. Yngvason, Ground state energy of the low density Bose gas, Phys. Rev. Lett. 80 (1998) 2504.
[44] I. Griva, S.G. Nash, A. Sofer, Linear and Nonlinear Optimization, 2nd edition, Soc. Indust. Appl. Math., Philadelphia, 2009.
[45] F. Verstraete, J.I. Cirac, Renormalization algorithms for quantum many-body systems in two and higher dimensions, arXiv:cond-mat/0407066v1, 2004.
[46] V. Murg, F. Verstraete, J.I. Cirac, Variational study of hard-core bosons in a two-dimensional optical lattice using projected entangled pair states, Phys. Rev. A 75 (2007) 033605.
[47] E.M. Stoudenmire, S.R. White, Studying two-dimensional systems with the density matrix renormalization group, Annu. Rev. Condens. Matter Phys. 3 (2012) 111–128.
[48] M. Lubasch, J.I. Cirac, M.-C. Bañuls, Unifying projected entangled pair state contractions, New J. Phys. 16 (2014) 033014.
[49] M. Lubasch, J.I. Cirac, M.-C. Bañuls, Algorithms for finite projected entangled pair states, Phys. Rev. B 90 (2014) 064425.
[50] J. Jordan, R. Orús, G. Vidal, F. Verstraete, J.I. Cirac, Classical simulation of infinite-size quantum lattice systems in two spatial dimensions, Phys. Rev. Lett. 101 (2008) 250602.
[51] P. Corboz, Variational optimization with infinite projected entangled-pair states, Phys. Rev. B 94 (2016) 035133.
[52] W.-Y. Liu, S.-J. Dong, Y.-J. Han, G.-C. Guo, L. He, Gradient optimization of finite projected entangled pair states, Phys. Rev. B 95 (2017) 195154.

[53] N. Navon, A.L. Gaunt, R.P. Smith, Z. Hadzibabic, Emergence of a turbulent cascade in a quantum gas, Nature 539 (2016) 72–75.
[54] C. Lubich, I.V. Oseledets, B. Vandereycken, Time integration of tensor trains, SIAM J. Numer. Anal. 53 (2015) 917–941.
[55] J. Haegeman, C. Lubich, I.V. Oseledets, B. Vandereycken, F. Verstraete, Unifying time evolution and optimization with matrix product states, Phys. Rev. B 94 (2016) 165116.
[56] M. Lubasch, J.I. Fuks, H. Appel, A. Rubio, J.I. Cirac, M.-C. Bañuls, Systematic construction of density functionals based on matrix product state computations, New J. Phys. 18 (2016) 083039.
[57] S. Al-Assam, S.R. Clark, D. Jaksch, The tensor network theory library, J. Stat. Mech. Theory Exp. 2017 (2017) 093102.