

Computing vibrational eigenstates with tree tensor network states (TTNS)

Henrik R. Larsson^{1, a)}

Division of Chemistry and Chemical Engineering, California Institute of Technology, Pasadena, CA 91125, USA

(Dated: 27 November 2019)

We present how to compute vibrational eigenstates with tree tensor network states (TTNS), the underlying *ansatz* behind the multilayer multi-configuration time-dependent Hartree (ML-MCTDH) method. The eigenstates are computed with an algorithm that is based on the density matrix renormalization group (DMRG). We apply this to compute the vibrational spectrum of acetonitrile (CH_3CN) to high accuracy and compare TTNS with matrix product states (MPS), the *ansatz* behind the DMRG. The presented optimization scheme converges much faster than ML-MCTDH-based optimization. For this particular system, we found no major advantage of the more general TTNS over MPS. We highlight that for both TTNS and MPS, the usage of an adaptive bond dimension significantly reduces the amount of required parameters. We furthermore propose a procedure to find good trees.

I. INTRODUCTION

The multi-configuration time-dependent Hartree (MCTDH) method has been established as a very powerful method for performing molecular quantum dynamics simulations.^{1–4} Mathematically, the MCTDH *ansatz* is based on a tensor decomposition, namely the Tucker decomposition.⁵ In its standard form, it can be applied to up to 12-dimensional systems.⁶ With mode-combination, that is, a redistribution of dimensions within the tensor decomposition, it has been successfully applied to up to about 80-dimensional model systems.^{7–9} However, MCTDH inherently scales exponentially and cannot be applied to larger and more complex systems. Major improvements could be obtained with a generalization of MCTDH, namely the multilayer MCTDH (ML-MCTDH) method.^{10–14} In mathematics, this approach is known as hierarchical Tucker decomposition^{15–17} whereas in physics, it is known as tree tensor network states (TTNS) decomposition,^{18–20} a subset of the more general tensor network states (TNS) decomposition.^{21,22} With ML-MCTDH, scientists were able to simulate model systems with more than 1000 dimensions.^{10,12,23–25}

Independently, related tensor decompositions have also been developed in condensed matter physics, electronic structure theory and other fields.^{21,22,26,27} In particular, successful in these fields is the density matrix renormalization group (DMRG).^{27–29} The DMRG is based on a so-called matrix-product state (MPS) tensor decomposition.²⁷ This is known in mathematics as tensor train (TT) decomposition.³⁰ While MPS actually are a subset of TTNS and thus very much related to ML-MCTDH, the DMRG uses a completely different approach for performing quantum simulations. ML-MCTDH is based on the Dirac-Frenkel time-dependent

variational principle³ and gives highly nonlinear equations of motions whereas the DMRG, essentially, converts the nonlinear (typically time-independent) equations to fixed-point iterations, similar to the Hartree-Fock self-consistent field algorithm.³¹

There has not been much overlap between ML-MCTDH and DMRG developments. Only recently, developments for MPS and for the DMRG have been transferred to molecular quantum dynamics.^{32–40} However, much remains to be done in order to systematically compare and apply the different approaches to molecular quantum dynamics. In particular, we are not aware of any *direct* comparisons between MPS and ML-MCTDH/TTNS for molecular systems.

This paper is aimed to make a step into this direction. The aim of this paper is threefold: (1) We present and apply the highly successful diagrammatic notation^{18,19,22} in the context of molecular quantum dynamics. This powerful notation is very common in physics, but has, so far, been used in molecular quantum dynamics only as a pictorial tool^{11,41} and not for deriving equations. (2) We use this notation to transfer the essentials of the DMRG algorithms to TTNS in order to be able to compute vibrational spectra (i.e., to solve the time-independent Schrödinger equation). We also focus on using adaptive tensor sizes throughout the simulations and how to find good trees. (3) With the same methodology and code, we directly compare MPS with TTNS for the 12-dimensional CH_3CN molecule.

To distinguish our methodology from ML-MCTDH, we use the term TTNS for our work in the following, even though ML-MCTDH is based on TTNS for solving the time-dependent Schrödinger equation. Both TTNS and ML-MCTDH thus have the same overall computational scaling. The only difference is the way to apply tensor decompositions to quantum systems.

With ML-MCTDH, vibrational spectra have been computed using combinations of time-independent Krylov subspace techniques and imaginary time evolution based on the time-dependent variational

^{a)}Electronic mail: larsson [a t] caltech . edu

principle.^{42–44} The nonlinear nature of the ML-MCTDH equations of motions makes it often difficult to use it for computing many eigenstates. In contrast, since the very beginning of TTNS,^{19,20,45–48} they have been used with DMRG algorithms that directly solve the time-independent Schrödinger equation by iteratively solving many quadratic (eigenvalue) problems. This should allow for a more direct and straightforward calculation of vibrational spectra with many high-lying excited states.

This work thus aims to complement the already well-established ML-MCTDH method and to give an alternative approach to compute spectra. Note that approaches based on related tensor decompositions (the Candecomp format) already provide another alternative.^{41,49–51}

The remainder of this Article is organized as follows: Section II gives a detailed overview of the used tensor decompositions and tensor network states, the diagrammatic notation to present tensor network states and DMRG-like optimizations for TTNS. Section III shows an application of TTNS to the computation of the vibrational spectrum of acetonitrile, CH₃CN, and compares both TTNS with MPS and the ML-MCTDH-based optimization with the DMRG-like optimization. Section IV concludes and gives an outlook.

II. THEORY

In the following, the general notation and theory is discussed. After an introduction of tensor decompositions, tensor network states and the diagrammatic notation in Section II A, the exploitation of gauge degrees of freedom are discussed in Section II B. Section II C discusses the used algorithm for ground state minimization and Section II D discusses how to obtain excited states. This Section ends with a description of how to adapt the number of parameters in the tensor networks in Section II E and how to find good trees in Section II F. Some of the used notation and symbols are summarized in Table I. Here, we mostly (but not exclusively) use the notation from the ML-MCTDH context.^{11,12,52}

A. Tensor decompositions

To solve Schrödinger’s equation, the F -dimensional wavefunction typically is represented *via* a Galerkin approach. That is, a direct product of (often orthogonal) “primitive” bases $\{|\chi_{\alpha_f}^{(f)}\rangle\}_{\alpha_f=1}^{N_f}$ of finite size N_f for dimension $f \in [1, F]$ is used as representation:

$$|\Psi\rangle = \sum_{\alpha_1}^{N_1} \sum_{\alpha_2}^{N_2} \cdots \sum_{\alpha_F}^{N_F} C_{\alpha_1 \alpha_2 \dots \alpha_F} \bigotimes_f^F |\chi_{\alpha_f}^{(f)}\rangle. \quad (1)$$

The entries of the real- or complex-valued coefficient tensor \mathbf{C} are then to be determined.

TABLE I. Some of the notation used in this work. A non-exhaustive list of alternative symbols are also given. “#” stands for “number.”

description	symbol	alternative symbols
physical dimension	F	D
physical index	f	κ
physical basis	$ \chi^{(f)}\rangle$	$ \sigma\rangle$
physical basis dimension	$N^{[f]}$	n_f, k
# of layers	L	
layer	l	
horizontal position in layer l	κ	n
bond dim./rank/# of SPFs	$n^{[l;\kappa]}$	m, M, D, χ, r
dimension of node/tensor/site	$d^{[l;\kappa]}$	
node/SPF tensor/site	$A^{[l;\kappa]}$	$\chi^{[l,\kappa]}, \Lambda^{[l,\kappa]}$

While this approach is very simple and numerically robust, the problem lies in the the coefficient tensor \mathbf{C} whose size scales exponentially as N_{mean}^F where N_{mean} is the geometric mean of the number of employed basis functions per dimension. For molecular quantum dynamics, N_{mean} typically is ~ 10 and this approach is then limited to about $F = 9$ dimensional systems. Tensor decompositions lower the scaling and enable studies for much larger systems.

For introducing tensor decompositions, specifically tensor networks, we consider a $F = 3$ dimensional wavefunction represented in a finite basis as shown in Eq. (1). The coefficient tensor then has entries $C_{\alpha\beta\gamma}$. In the so-called Tucker decomposition, which is the underlying decomposition of the MCTDH approach, \mathbf{C} is factorized in an up to F -dimensional so-called core-tensor $\mathbf{A}^{[1]}$ and several smaller-dimensional, “auxiliary” tensors $\mathbf{A}^{[2,\kappa]}$:

$$C_{\alpha\beta\gamma} \approx \sum_{ijk} A_{ijk}^{[1]} A_{\alpha i}^{[2,1]} A_{\beta j}^{[2,2]} A_{\gamma k}^{[2,3]}, \quad (2)$$

where $i \in [1, n^{[2,1]}]$, $j \in [1, n^{[2,2]}]$ and $k \in [1, n^{[2,3]}]$. Here, $n^{[l,\kappa]}$ are called “bond dimensions”, “ranks” or “number of single-particle functions (SPFs)” for a particular tensor $\mathbf{A}^{[l,\kappa]}$. We will use the term bond dimension in the following. In $n^{[l,\kappa]}$ and $\mathbf{A}^{[l,\kappa]}$, l is the *layer* and κ the horizontal position in a particular layer. The Tucker decomposition thus is a two-layer approach. The notion of layers will become more clear shortly. Note that we use here a more elaborate notation because this will be required for the following more intricate tensor decompositions.

Since the core tensor $\mathbf{A}^{[1]}$ has the same dimensionality as the original coefficient tensor \mathbf{C} , the Tucker decomposition does not avoid exponential scaling with dimensionality. Nevertheless, the problem size typically is reduced due to the introduction of the auxiliary tensors $\mathbf{A}^{[2,\kappa]}$, which are contracted with the core tensor in order to restore \mathbf{C} . For many problems, the required bond dimensions $n^{[l,\kappa]}$ are much smaller than the physical dimensions $N^{[f]}$ of the coefficient tensor. This drastically reduces the

size of the core tensor, compared to the coefficient tensor. Despite this reduction of problem size, the exponential scaling of the core tensor typically limits the Tucker decomposition to about 15-dimensional systems,^{6,53} or, for model systems, to about 80-dimensional systems.⁷⁻⁹

The exponential scaling of the core tensor can be avoided by lowering its dimensionality. This is achieved by introducing higher-dimensional auxiliary tensors. They are then decomposed further using, again, a Tucker decomposition; for example,

$$C_{\alpha\beta\gamma} \approx \sum_{ij} A_{ij}^{[1]} A_{\alpha i}^{[2,1]} B_{\beta\gamma j} \quad (3)$$

$$= \sum_{ij} A_{ij}^{[1]} A_{\alpha i}^{[2,1]} \sum_{kl} A_{klj}^{[2,2]} A_{\beta k}^{[3,1]} A_{\gamma l}^{[3,2]} \quad (4)$$

Here, the core tensor $\mathbf{A}^{[1]}$ is only two-dimensional and the higher-dimensional auxiliary tensor \mathbf{B} is again Tucker-decomposed into $\mathbf{A}^{[2,2]}$, $\mathbf{A}^{[3,1]}$ and $\mathbf{A}^{[3,2]}$.⁵⁴ This type of decomposition is the underpinning of the ML-MCTDH method and of TTNS. Note that in many TTNS calculations in electronic structure theory,^{46,48,55} (almost) every tensor in the TTNS typically is connected with a physical dimension whereas in vibrational dynamics, only the tensors in the lowest layers typically are connected with a physical dimension.¹³ This difference, however, is not fundamental but rather depends on the best way to represent the physical system (Fermions with a “generic” Hamiltonian vs. distinguishable particles with a Hamiltonian that leads to some approximate grouping of particles, in terms of the coupling).

As a special case of TTNS, one can formulate a tensor decomposition where each value of $C_{\alpha\beta\gamma}$ is reconstructed by computing a trace of product of matrices:

$$C_{\alpha\beta\gamma} \approx \sum_i A_{\alpha i}^{[1]} \sum_j A_{\beta ij}^{[2]} A_{\gamma j}^{[3]}. \quad (5)$$

The trace of matrix products can be understood by considering $\mathbf{A}_{\beta::}^{[2]}$ as a matrix and $\mathbf{A}_{\alpha:}^{[1]}$ ($\mathbf{A}_{\gamma:}^{[3]}$) as row (column) vectors. Hence, this decomposition is called matrix product state (MPS) and is the *ansatz* behind the DMRG.

Since it is quite tedious to deal with these decompositions using equations, here, we instead use a diagrammatic notation.^{21,22} Fig. 1 gives an overview of it. There, each node represents a tensor. A node with an asterisk marks complex conjugation.⁵⁶ In the following, we use the terms node and tensor interchangeably. In the context of DMRG, nodes are called “sites.” Each vertex represents a tensor dimension. Vertices that connect two nodes/tensors represent a contraction, that is, a summation between the two tensors over the particular dimension.

The previously introduced examples of tensor network states are shown in Fig. 2 using the diagrammatic notation. There, each tensor $\mathbf{A}^{[l,\kappa]}$ is labeled by layer l and horizontal position κ in the tensor network. The dangling bonds represent the physical dimensions of size $N^{[l]}$. The

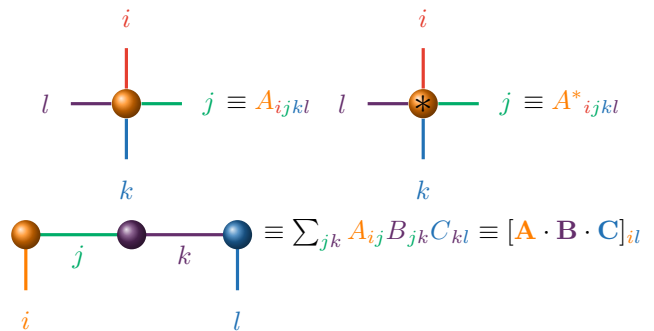


FIG. 1. Diagrammatic notation used in this work. The upper panel shows the four-dimensional tensor A_{ijkl} (left) and its complex conjugate (right) represented in diagrammatic notation. The lower panel shows a particular tensor contraction, the il th entry of the matrix product \mathbf{ABC} . See text for further details.

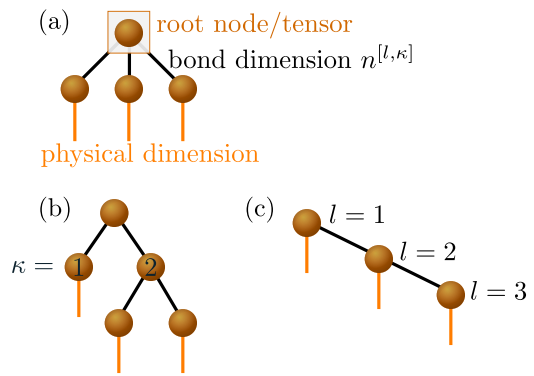


FIG. 2. Examples of tensor network states/decompositions in diagrammatic notation. (a) shows the Tucker decomposition from Eq. (2), (b) shows the tree tensor network state (TTNS) from Eq. (4) and (c) shows the matrix product state (MPS) from Eq. (5). The symbols for the bond dimension $n^{[l,\kappa]}$ (shown in A) in layer l (shown in (c)) and horizontal position κ (shown in (b)) are also exemplified.

vertices that connect tensors introduce “virtual” bond dimensions of size $n^{[l,\kappa]}$. Contracting all virtual bonds finally restores the core tensor \mathbf{C} . The notion of root tensor will become clear in the following Section II B.

Another notation is to associate each tensor $\mathbf{A}^{[l,\kappa]}$ with the representation of some basis function. In the context of DMRG, these functions are called renormalized basis functions whereas in the context of MCTDH, they are known as single-particle functions (SPFs), $|\phi\rangle$. SPFs can be understood as a variationally optimized basis and are represented either by SPFs in higher layers or by the primitive basis, e.g., $|\phi_i^{[l,\kappa]}\rangle = \sum_{\alpha} A_{\alpha i}^{[l+1,\kappa_{l+1}]} |\chi_{\alpha}^{(\kappa_{l+1})}\rangle$. While this basis notation is very powerful as well and can go hand-in-hand with the diagrammatic notation, here, we mostly use the latter. However, for some cases, the former notation is simpler and hence will be used occasionally. With SPFs, the wavefunction from the three-dimensional example can be described in terms of mul-

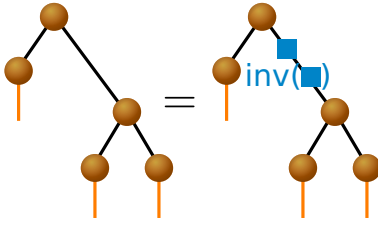


FIG. 3. Example of the gauge degree of freedom. A matrix (shown as blue rectangle) and its inverse is inserted into a particular bond. The matrix and its inverse can be absorbed into the neighboring tensors without changing the final, contracted tensor.

tidimensional configurations $|\Phi_{ijk}\rangle$ that are decomposed as product of SPFs:

$$|\Psi\rangle = \sum_{ijk} A_{ijk}^{[1]} |\Phi_{ijk}^{[1]}\rangle = \sum_{ijk} A_{ijk}^{[1]} |\phi_i^{[1,1]}\rangle |\phi_j^{[1,2]}\rangle |\phi_k^{[1,3]}\rangle \quad (6)$$

$$= \sum_{\alpha\beta\gamma} \sum_{ijk} A_{ijk}^{[1]} A_{\alpha i}^{[2,1]} A_{\beta j}^{[2,2]} A_{\gamma k}^{[2,3]} |\chi_\alpha^{(1)}\rangle |\chi_\beta^{(2)}\rangle |\chi_\gamma^{(3)}\rangle. \quad (7)$$

B. Canonicalization

The presented tensor networks actually have some intrinsic redundancy: An insertion of an invertible matrix and its inverse between each bond does not change the overall wave function; see Fig. 3. This so-called gauge degree of freedom can be exploited to improve numerics.^{3,27} In this Section, we discuss two ways to improve numerics and will later in Section II C 2 discuss a third way. The first way is to reshape each tensor into a matrix and restrict this matricized⁵ tensor to be orthogonal, c.f. Fig. 4. In the used tensor diagrams, the matricization is implicitly declared by the location of the bonds on each tensor. Bonds pointing away from the root node (typically downwards) define a multi-index that is associated with the rows of the matrix. The bond that points towards the root node is associated with the columns of the matrix.⁵⁷ Note that in all tensor networks shown here, for each node, there is only one bond pointing to the root node (meaning that the tensor network does not contain loops). The orthogonalization of the matricized tensors means nothing else than that the SPFs are orthogonal. It does massively simplify many expressions. For example, the calculation of the norm, $\sqrt{\langle\Psi|\Psi\rangle}$, boils down to the norm calculation of the root tensor (viewed as a vector). This is shown in Fig. 5.

The orthogonalization towards a specific (root/central) node is called canonicalization. A change of the canonical form (change of the root node) is possible *via* a QR or similar matrix decomposition,⁵⁸ see Fig. 6: Starting from the previous root node, the tensor is matricized such that the bond pointing toward the new root node represents the columns (step *I* in Fig. 6). Then, a QR decomposition is performed: $\mathbf{A} = \mathbf{Q}\mathbf{R}$. The orthogonal matrix

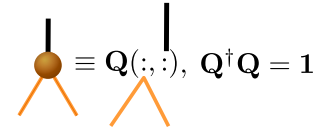


FIG. 4. Representation of a tensor as orthogonal matrix (matricization). The two lower bonds represent the rows (as multi-index) of the matrix \mathbf{Q} and the upper bond represent the columns.

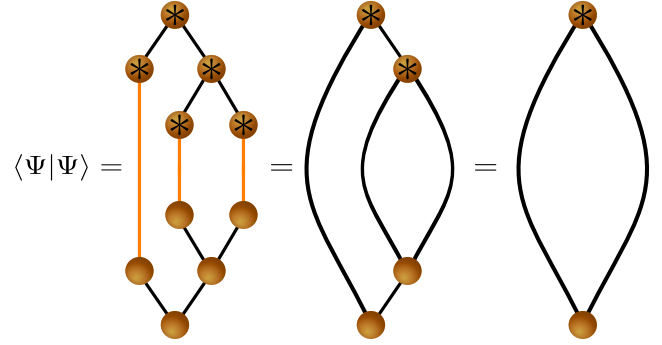


FIG. 5. Norm calculation for a tensor network state with orthogonalized tensors. The contraction of the tensor network simplifies due to the orthogonality of each tensor; compare with Fig. 4.

\mathbf{Q} becomes the new tensor and the triangular matrix \mathbf{R} is absorbed into the neighbor (step *II* in Fig. 6). This procedure is repeated until the desired new root node is reached.

While the canonicalization makes every (matricized) tensor orthogonal, any unitary transformation between two connected tensors is still possible. Thus, a second way to exploit the redundancy is to fix this unitary transformation. In the context of MCTDH, this unitary de-

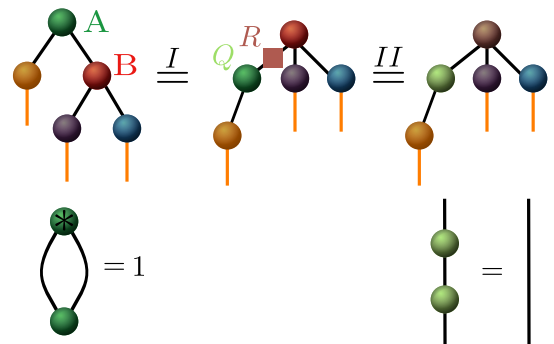


FIG. 6. Change of canonicalization (root node) from tensor \mathbf{A} to tensor \mathbf{B} . The upper panel show the steps performed in the canonicalization: Reshaping and QR decomposition in step *I* and absorption of the \mathbf{R} matrix in step *II*. The lower panels show the particular orthogonality conditions (isometries) of tensor \mathbf{A} for the canonical forms where \mathbf{A} is root node (left) and where \mathbf{B} is root node (right). A single line represents a unit matrix.

gree of freedom is fixed by adding an additional gauge operator to the Hamiltonian. Some gauges simplify the solution of the differential equations.^{3,59} In contrast to MCTDH, here, this gauge is *not* fixed directly. Finding most suitable gauges is subject to future work. In the context of MCTDH and improved relaxation, so-called energy orbitals are used to fix the gauge. They diagonalize the separable part of the Hamiltonian.⁶⁰

An alternative is to use natural orbitals that diagonalize reduced density matrices. While natural orbitals are not used directly in this work, changing the canonical form (during the sweeps to be introduced in Section II C 2) actually leads to this gauge for the root node.

C. Ground state optimization

Having introduced TTNS, we now discuss how to optimize for the ground state. That is, the goal is to minimize the expectation value of the Hamiltonian \hat{H} :

$$E = \min_{\Psi} \frac{\langle \Psi | \hat{H} | \Psi \rangle}{\langle \Psi | \Psi \rangle} \quad (8)$$

For that, we first discuss possible forms of \hat{H} in Section II C 1 and then discuss the employed sweep algorithm in Section II C 2.

1. Hamiltonian

In the primitive basis $\{|\chi_{\alpha_f}^{(f)}\rangle\}_{\alpha_f=1}^{N_f}$ (see Eq. (1)), the Hamiltonian \hat{H} has the following matrix representation:

$$H_{\alpha_1 \alpha_2 \dots \alpha_F, \alpha'_1 \alpha'_2 \dots \alpha'_F} = \bigotimes_{f=1}^F \bigotimes_{f'=1}^F \langle \chi_{\alpha_f}^{(f)} | \hat{H} | \chi_{\alpha'_{f'}}^{(f')} \rangle, \quad (9)$$

For methods based on tensor decompositions of Ψ , working equations become much simplified if \hat{H} takes a similar (not necessarily identical) tensor decomposition as Ψ itself. Hence, we assume here that \hat{H} can be decomposed as a sum of direct products of one-dimensional operators or matrices in finite basis representation (SOP):

$$H_{\alpha_1 \alpha_2 \dots \alpha_F, \alpha'_1 \alpha'_2 \dots \alpha'_F} \approx \sum_{s=1}^{N_{\text{PF}}} c_s \bigotimes_{f=1}^F h_{\alpha_f, \alpha'_f}^{(f,s)}, \quad (10)$$

Such a decomposition can be achieved, for example, using the “potfit” algorithm⁶¹ or variants thereof^{52,62–65} that Tucker-decompose the potential in grid representation.⁶⁶ The potfit procedure is shown diagrammatically in Fig. 7. Another alternative is the related Candecomp format.⁵ There, the circle-shaped tensor in Fig. 7 would be diagonal. Most optimal decompositions would probably be decompositions of the Hamiltonian using similar tree tensor networks (tree tensor network operators). This is called multilayer potfit for

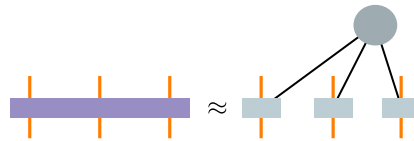


FIG. 7. Tucker decomposition (“potfit”) of the Hamiltonian for a three-dimensional example (this leads to a six-dimensional Hamiltonian tensor to be decomposed).

MCTDH.⁵² It has previously been used for the special case of matrix products/tensor trees by Rakhuba and Oseledets.³² Other kinds of approximations to avoid the tensor decomposition at all may also be possible in certain cases.^{11,67} For many-body decompositions of the Hamiltonian, so-called complementary operators can also be used.⁶⁸ While all of this would certainly decrease the overall computational effort we here stick to the standard SOP format.

2. Sweep algorithm

We now address the actual ground state minimization. The tensor decomposition turns the eigenvalue problem into a nonlinear optimization problem. However, the decomposition also clearly identifies ways to approximately decouple the tensors and to optimize the tensor network tensor by tensor, thereby solving many smaller subproblems instead of solving one big nonlinear problem. This idea has already been used in the context of improved relaxation in MCTDH,^{60,69} where the root node is solved separately from the other nodes (see also Refs. 70–72).

Keeping all tensors but the root tensor fixed results in a standard eigenvalue problem for the optimal values of the root tensor^{69,71} with an effective Hamiltonian \mathcal{H} obtained by representing the full Hamiltonian in configuration representation (c.f. Eq. (6)):

$$\mathcal{H} = \langle \Phi^{[1]} | \hat{H} | \Phi^{[1]} \rangle \quad (11)$$

This is shown diagrammatically in Fig. 8. It can be computed in a recursive way as explained by Manthe.¹¹

In improved relaxation in MCTDH, the minimization of the remaining tensors is then performed via a normal nonlinear optimization for each tensor, typically either by imaginary time evolution⁶⁹ or by Jacobi rotations.^{71,72} This nonlinear optimization is most of the time not the main effort in the optimization for standard MCTDH. However, for ML-MCTDH/TTNS the nonlinear optimization becomes non-trivial, especially for excited states. Furthermore, improved relaxation relaxes the SPFs in each degree of freedom independently from each other and thus neglects the coupling between them.

Here, we perform a different approach,^{19,20,45,47,55} borrowed from the DMRG algorithm.^{28,29} After minimization of the root node while keeping all other nodes fixed, we change the canonical form to another node. This is

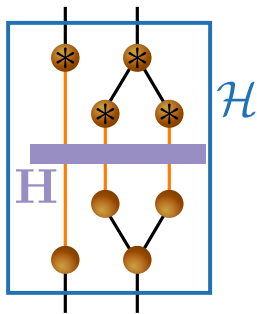


FIG. 8. Effective Hamiltonian \mathcal{H} from Eq. (11) (blue rectangle) in diagrammatic notation. The purple block represents the full Hamiltonian \mathbf{H} (see Fig. 7 for simplifications). The three-dimensional wavefunction is represented by the tensor network (b) shown in Fig. 2.

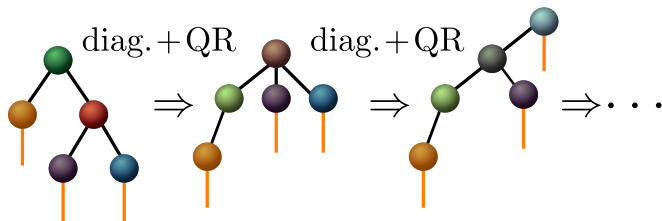


FIG. 9. Sweep through a tensor network. At each step, the root node is replaced by the eigenstate of the effective Hamiltonian \mathcal{H} (Fig. 8) and a QR decomposition is performed to change the root node (Fig. 6).

then minimized by diagonalizing an effective Hamiltonian \mathcal{H} . This procedure is repeated for all tensors, resulting in a sweep through the whole tree; see Fig. 9. The sweeps are repeated until convergence. Thus, the nonlinear optimization problem of minimizing all tensors at once is converted into a fixed-point iteration with successive quadratic (eigenvalue) problems.

Note that the convergence of the sweep algorithm can be improved by using a two-site variant where two neighboring tensors are first contracted, then diagonalized simultaneously and afterwards decomposed using a singular value decomposition (SVD).⁵⁸ This leads to faster and more stable convergence but is also more costly. Perturbative approaches are also possible to improve convergence of the one-site algorithm.^{73,74}

For MPS, which form linear tensor networks, the way to sweep through the network is clearly identified by sweeping from one end to the other end (forward sweep) and back (backward sweep). For trees, there are several possibilities. Gerster *et al.* start from the highest layer and optimize the tree layer by layer, finally optimizing the tensors connected to physical dimensions (similar to breadth-first search).⁴⁷ Here, we generalize the MPS sweep using a more depth-first search approach such that each tensor is diagonalized as many times as it has bonds per forward and backward sweep: The largest linear chain in the tree is identified and the optimization starts at one end of this chain. One then sweeps through this

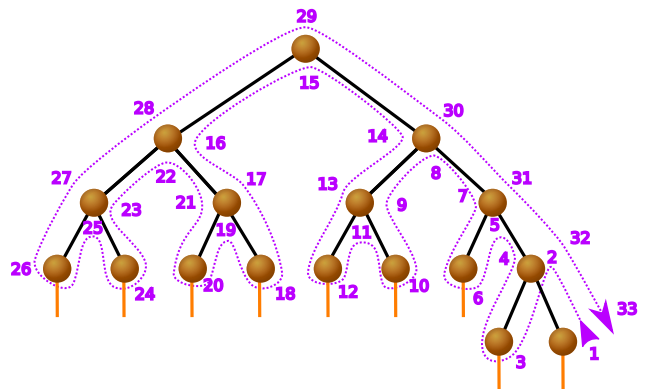


FIG. 10. Example of the sweep schedule (purple path; starting in the lower right corner) used in this work. For each number, the corresponding (closest) tensor is optimized.

chain, enters branches whenever they occur and sweeps through the branches forwards and backwards. This procedure is repeated recursively. This is very similar to the procedure used by Nakatani and Chan.⁴⁶ An example is shown in Fig. 10.

We note that the sweep algorithm has overall the same computational scaling as the standard ML-MCTDH algorithm (namely n^{d+1} for the diagonalization with Hamiltonians in SOP form; n is the geometric mean of the bond dimensions on each tensor and d is the dimensionality of the tensor). The sweep algorithm thus just presents an alternative way to obtain eigenstates. It may, however, be advantageous as only many eigenvalue problems need to be solved instead of having the necessity to solve nonlinear equations directly. The sweep algorithm also takes some coupling into account by allowing, after each update of a tensor, for a feedback to the neighboring tensor to be updated. In contrast to the ML-MCTDH algorithm, no regularization in the differential equations is required. Even though the regularization in MCTDH can be improved,^{75–80} here, this is not even necessary.

D. Excited states optimization

In principle, excited states can be computed using the same procedure as the ground state optimization. One simply targets a state whose energy is closest to the energy of interest in the effective Hamiltonian \mathcal{H} . However, this procedure does not work in general. Problems occur when there are degeneracies in the spectrum or whenever there is a high density of states. Then, root flipping can occur which means that the optimization oscillates between different states. There are various methods to avoid these problems.^{37,43,60,81–85} In the following, we describe three methods. The first two methods are based on making use of the previously computed states whereas the third method is based on state averaging. Two of the methods will be combined in Section III. We note that, as the used sweep algorithm simply is a straightforward

generalization of the DMRG algorithm, any improvement from the DMRG literature^{37,81} can straightforwardly be implemented for TTNS. Here, we stick to more basic methods in order to present the overall methodology.

1. Projecting out previously computed states

A very common approach is to project the previously computed states $|I\rangle$ out of the solution space.^{37,85} For that, we first define the projection operator \hat{P} on the previously computed states $|I\rangle$ as

$$\hat{P} = \sum_I |I\rangle\langle I|. \quad (12)$$

The Hamiltonian is then modified to

$$\hat{H}^P = (\hat{1} - \hat{P})\hat{H}(\hat{1} - \hat{P}). \quad (13)$$

This approach has some practical problems. For example, \hat{H}^P includes a null space that may interfere with the state of interest. Furthermore, \hat{H}^P is more complicated than the approach presented in the next section and numerical difficulties arise once $\{|I\rangle\}$ are not fully orthogonal.

2. Shifting the spectrum

As an alternative to the projection method, the projector can be used not to project the states out but to shift the energies E_I of the states $|I\rangle$ by an amount S .⁸⁶

$$\hat{H}^S = \hat{H} + \sum_I (E_I + S)|I\rangle\langle I|. \quad (14)$$

If S is larger than the difference between the energy levels of the states $|I\rangle$ and the targeted state, the targeted states becomes the ground state of \hat{H}^S and can thus easily be retrieved.

Inserting Eq. (14) into Eq. (11), the matrix elements of the effective Hamiltonian \mathcal{H}^S take the form of

$$\mathcal{H}_{AB}^S = \langle \Phi_A^{[1]} | H | \Phi_B^{[1]} \rangle + \sum_I (E_I + S) \langle \Phi_A^{[1]} | I \rangle \langle I | \Phi_A^{[1]} \rangle \quad (15)$$

$$= \mathcal{H}_{AB} + \sum_I (E_I + S) \langle \Phi_A^{[1]} | I \rangle \langle I | \Phi_A^{[1]} \rangle. \quad (16)$$

Note that after the optimization, the newly computed eigenstate is orthogonal to the previously computed states $\{|I\rangle\}$. Then, $\langle \Phi_A^{[1]} | I \rangle$ typically approaches zero and \mathcal{H}^S approaches \mathcal{H} .

3. State average calculations

Another, well-known way to obtain excited states is to describe several eigenstates simultaneously with the same tensor network. This is known as state averaging.^{42,60,70,87} It can be achieved by adding to the root node an additional ‘‘physical’’ bond that indicates the particular state. This has been used both for MCTDH and ML-MCTDH, and there, it is easily implemented.^{42,60,87}

For the sweep algorithm in TTNS, the implementation is less obvious as the root node changes during the sweep. In principle, the state dimension could stay on one particular node. However, the states then become nonorthogonal once the canonical form changes. We found that this leads to arbitrarily small weights of the particular states and thus an unstable optimization.

To make the optimization stable, it is required to move the state dimension from one node to another. This has already been used in DMRG calculations by means of taking averages of density matrices.⁸⁸ Here, instead of using density matrices, we use a more simpler and less costly approach based on a SVD. The procedure is described in Fig. 11: In step A , the old root node G with the state dimension is first transposed (step A_1 ; note the change of the order of the bonds in Fig. 11) and then decomposed using a SVD (steps A_2 and A_3):

$$\underbrace{\mathbf{G}}_{A_1} = \underbrace{\mathbf{U}\mathbf{s}\mathbf{V}^\dagger}_{A_2} = \underbrace{\mathbf{U}\tilde{\mathbf{V}}}_{A_3}. \quad (17)$$

The remaining dimensions (the rows of \mathbf{U}) are now separated from the state dimension and the dimension connecting to the next node (the columns of $\tilde{\mathbf{V}}$). In the final step B , $\tilde{\mathbf{V}}$ is absorbed into the neighboring node, which becomes the new root node.

One drawback of this procedure is that the transposition of the previous root tensor in step A_1 changes the ordering of dimensions in the tensor network. This means that the initial and final tensor networks shown in Fig. 11 are not related via a gauge transform. Therefore, the required bond dimension to represent the same physical state to a given accuracy changes (in the example in Fig. 11, the dimension of the purple bond differs from that of the gray bond) and often grows. To avoid a growth of the bond dimension, it is thus required to truncate it (by means of discarding some singular values/some rows of $\tilde{\mathbf{V}}$). Then, $|\Psi\rangle$ varies whenever the canonical form changes and the energy of the state depends on the position of the root node in the tree. Typically, the energy is lowest in the center of a MPS⁸⁸ but this is less obvious in a tree. Furthermore, it is crucial to diagonalize the new root node once the state dimension has moved. Canonicalizing from one node to a not directly connected node without diagonalization will deteriorate the state if the bond dimension is truncated.

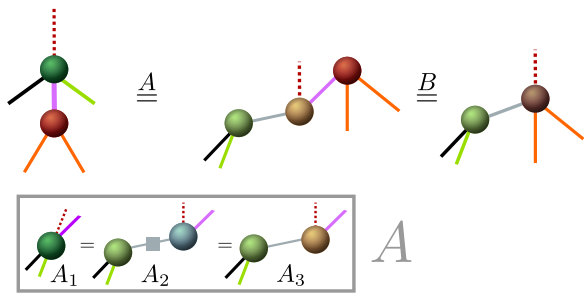


FIG. 11. Canonicalization with state averaging. For clarity, the bonds are colored. The dashed bond represent the state dimension. The substeps for step A are shown in the lower panel (gray rectangle). The gray square in step A_2 corresponds to a diagonal matrix (the singular values). See text for details.

E. Adaption of the bond dimension

A drawback of complicated tensor network states such as TTNS is the number of input parameters. For example, the required bond dimensions for each node need to be set. While this can be done in an iterative manner by observing the natural weights (eigenvalues of the density matrix) after the TTNS is converged, this procedure is cumbersome. Instead, we here make use of an adaptive bond dimension during the optimization procedure.^{89–92} For state-averaged states, this can easily be implemented by truncating the singular values s_i in Eq. (17) such that only singular values larger than some parameter ϵ are included.⁹³ For TTNS without state averaging, bond adaption is achieved by observing the natural weights for the bond between the root node \mathbf{A} and the neighboring node \mathbf{B} that will become the new root node during the sweep. This allows to reduce the bond dimension (by discarding some natural orbitals). In the current implementation, increasing the bond dimension is achieved by performing a SVD of the matricized, combined tensor \mathbf{AB} . Improved methods may be used in future.⁸⁹ The two-site algorithm offers the most straightforward and robust way to adapt the bond between two nodes. It is, however, also more costly.

F. Finding good trees

Another drawback of TTNS over simpler methods is that it is not always obvious how to set up an optimal tree, meaning a tree “topology” that has low bond dimensions for a given accuracy.⁹⁰ The same holds for the ordering of the physical dimensions in a MPS.^{94,95} Here, we propose a simple way to improve (to “disentangle”) the tree, meaning that smaller bond dimensions for a given accuracy are required. Other procedures based on quantum information theory are also possible.⁵⁵ Our procedure is based on randomly permuting dimensions of neighboring tensors in the tree. Once some eigenstate

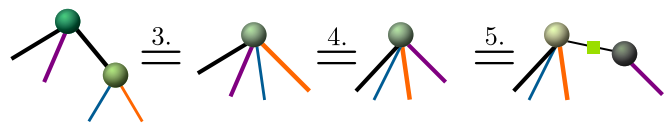


FIG. 12. Procedure to optimize a tree. After randomly selecting a tensor (1.; not shown) and canonicalizing it (2.; not shown), it is contracted with a randomly chosen neighbor (3.). Then (4.), their dimensions are permuted. Finally (5.), a SVD along a random bond dimension is performed to check the required bond dimension.

has been obtained with an initial tree that is based on an educated guess, the following (see also Fig. 12) is iterated many times:

1. Randomly select tensor \mathbf{X} and a neighboring tensor \mathbf{Y} .
2. Canonicalize to \mathbf{X} .
3. Contract \mathbf{X} with \mathbf{Y} to get the combined tensor \mathbf{Z} (see Fig. 12).
4. Randomly permute the dimensions of \mathbf{Z} , e.g., $Z_{ijkl} \rightarrow \tilde{Z}_{klji}$ (see Fig. 12).
5. Perform a SVD for a randomly chosen matricization of $\tilde{\mathbf{Z}}$: $\tilde{Z}_{klji} = \sum_x U_{kljx} s_x V_{xi} = \sum_x U_{kljx} \tilde{V}_{xi}$ (see Fig. 12).
6. Set the bond dimension of this new configuration as the number of singular values that are larger than some parameter ϵ . If this is smaller than the previous bond dimension: Accept the new configuration, replacing \mathbf{X} and \mathbf{Y} by \mathbf{U} and $\tilde{\mathbf{V}}$. Otherwise, discard it.

For particular tensors \mathbf{X} and \mathbf{Y} , steps 4 to 6 can be repeated several times (microiterations) before two new tensors are selected (macroiterations).

The optimization procedure makes use of “greedy” optimization. As new configurations are only accepted if they actually lower the bond dimension, the optimization likely gets stuck in a local minimum. This is because only neighboring tensors are combined. However, the algorithm can simply be improved by accepting worse configurations with some probability, making it similar to simulated annealing.⁹⁶

Since two nodes need to be contracted for this algorithm, it scales as n^{2d} . This is larger than the normal scaling of the optimization, which is n^{d+1} . However, by computing eigenvalues of the reduced density matrix instead of computing a SVD and using iterative eigensolvers, the scaling can be reduced. At any rate, the optimization procedure is very fast because the prefactor of the scaling is orders of magnitudes smaller, compared to that of the eigenvalue optimization.

III. APPLICATIONS

In the following, we present results for a quartic force field of acetonitrile,^{97,98} which is a twelve-dimensional problem. This system has been established as a benchmark model, allowing for a comparison to a variety of methods.^{32,34,41,49,91,98} Here, we compare both TTNS and MPS against Smolyak quadrature performed by Avila and Carrington⁹⁸ and previous tensor train/MPS calculations performed by Rakhuba and Oseledets.³² The goal is to compute the lowest 84 eigenstates of the Hamiltonian to sub- cm^{-1} accuracy. While the correlation in this system is not very strong, difficulties arise due to a high density of states with many nearly-degenerate states, combined with the high accuracy demand. We note that the desired accuracy only serves as benchmark purpose, to make the problem more difficult and to compare with previous highly accurate calculations. The system parameters are described in Section III A. Section III B describes the particular methodology and combinations of the algorithms presented in Section II. The results are given in Section III C.

A. System parameters

We took the parameters of the quartic force field from the Heidelberg MCTDH package.⁹⁹ The $J = 0$ Hamiltonian, using normal coordinates and neglecting cross terms in the kinetic energy operator, has then 323 product terms. We used the same Gauß-Hermite discrete variable representation (DVR)¹⁰⁰ employed by Leclerc and Carrington and Rakhuba and Oseledets.^{32,49} The basis sizes were $N_i \in \{9, 7, 9, 9, 9, 9, 7, 7, 9, 9, 27, 27\}$ for the modes with harmonic frequencies (in cm^{-1}) $\omega_1 = 3065$, $\omega_2 = 2297$, $\omega_3 = 1413$, $\omega_4 = 920$, $\omega_5 = \omega_6 = 3149$, $\omega_7 = \omega_8 = 1487$, $\omega_9 = \omega_{10} = 1061$, $\omega_{11} = \omega_{12} = 361$. The TTNS we used is shown in Fig. 13 in panel (a). The particular way of setting up the tree roughly followed previous investigations with similar methods.^{41,91} For the MPS, we used the same basis ordering as used by Rakhuba and Oseledets,³² namely, we ordered the modes according to their frequency. Similar to the findings of Rakhuba and Oseledets, we could not find particularly improved results for different orderings.

B. Methodology

To obtain the lowest 84 eigenstates, we iterated the following procedure, both for TTNS and MPS:

1. Do a (loose) state average calculation with 20 states for maximal seven sweeps (as shown in Fig. 10). This gives a rough estimate of the eigenstates.
2. Cluster the 20 states in groups whose energy differ by maximal 2 cm^{-1} . Discard the last cluster as this may overlap with higher lying states.

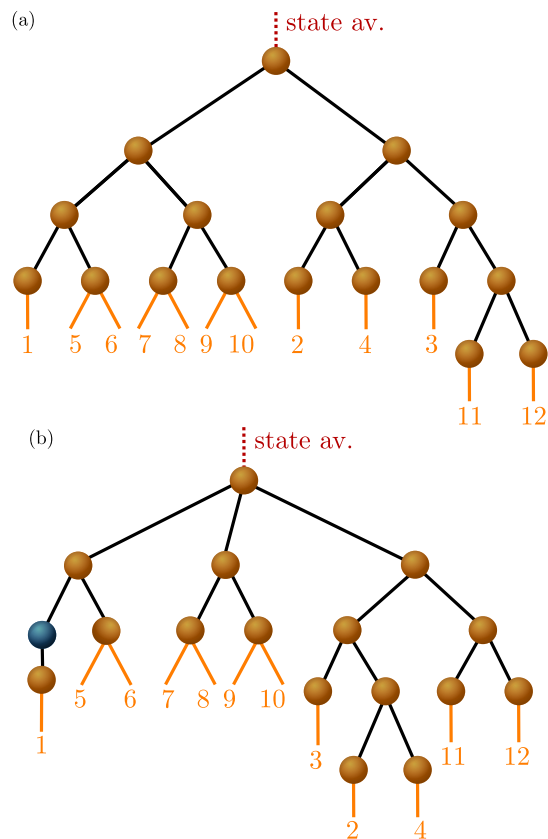


FIG. 13. Used tree tensor network states employed for the acetonitrile calculations. The numbers denote the particular physical dimension. (a) is the primarily used tree and (b) the optimized tree. The blue node denotes a redundant node/tensor.

3. For each cluster, do an additional (refined) state average calculation for maximal ten sweeps. This gives the converged states.

The convergence criterion for the sweep was a relative difference of 10^{-9} in the energy, compared to the energy of the previous sweep. Throughout, the Hamiltonian was shifted by the previously computed states (see Eq. (14)). We used a shift of $S = 5000 \text{ cm}^{-1}$. In each calculation, we adapted the bond dimension every fourth sweep (see Section II E). Besides using a singular value threshold ϵ for the bond adaption, we fixed the maximally allowed bond dimension n_{max} and also set a minimal bond dimension of $n_{\text{min}} = 3$. We performed calculations with a bond dimension threshold of either $\epsilon = 10^{-4}$ or 10^{-5} and a maximal bond dimension of either $n_{\text{max}} = 25$, 40 or 50. For the state average calculations, the accuracy of the eigenstates depends on which node they are evaluated. Hence, after convergence of the loose state average calculation (with 20 states), we selected the root node that gave the lowest energies.

In principle, one could also avoid the state average calculations at all and perform optimizations state by state. However, state averaging accelerates the optimiza-

tion procedure as fewer optimizations are necessary, even though larger bond dimensions are required in state average calculations. Another way to compute the spectrum would be to perform just one big state average calculation. This has been done for CH_3CN for computing the lowest 69 states with ML-MCTDH.⁹¹ However, this approach requires very big bond dimensions as all eigenstates need to be described by the same tree. The more eigenstates needed to be computed, the tighter the demands on the bond dimension.

As proof of concept, after having obtained eigenstates, we optimized the tree as explained in Section II F. For that, we arbitrarily chose the 10th excited state and optimized the tree using the “greedy” algorithm (only accepting new tensor configurations if the bond dimension is reduced). We did an overly exhaustive optimization and performed 4000 macroiterations (randomly selecting a tensor and a neighbor) and within each macroiteration we performed 190 microiterations (randomly permuting dimensions).

We also compared to ML-MCTDH-based optimizations, both for the ground state and for a state-averaged computation minimizing the lowest 13 states. For that, we used a fixed number of single-particle functions in each layer of $n^{1,\kappa} = \{20, 20\}$, $n^{2,\kappa} = \{8, 20, 6, 25\}$, $n^{3,\kappa} = \{4, 6, 10, 8, 5, 6, 6, 10\}$ and $n^{4,\kappa} = \{6, 6\}$. The ML-MCTDH optimizations used a regularization parameter of the equations of motions of $\epsilon_r = 10^{-10}$. In improved relaxation, after each diagonalization of the root node, the SPFs were propagated until the norm of their time-derivative was below 10^{-9} . The accuracy of the propagators were set to 10^{-9} as well.

The program we used is implemented in Python and makes use of NumPy¹⁰¹ and SciPy.¹⁰² The routines for computing a matrix-vector product with matrices decomposed in SOP form are implemented in C++ and are taken from Ref. 103 (see appendix therein). As diagonalizer we use Davidson’s method,¹⁰⁴ as implemented in PySCF.¹⁰⁵

C. Results and Discussion

In the following, we present the results and compare them to results based on Smolyak quadrature, performed by Avila and Carrington,⁹⁸ and to results based on tensor trains (TT), performed by Rakhuba and Oseledets.³²

Rakhuba and Oseledets obtained a zero point vibrational energy (ZPVE) of $9837.4063 \text{ cm}^{-1}$ and Avila and Carrington a ZPVE of $9837.4073 \text{ cm}^{-1}$ (as given by Rakhuba and Oseledets). Despite using the same basis, we were not able to reproduce the ZPVE from Rakhuba and Oseledets. The best (i.e. smallest) value we could obtain with our code is $9837.4069 \text{ cm}^{-1}$. We confirmed this value with an independent standard MCTDH calculation (using the Heidelberg package⁹⁹) and an exhaustive number of single particle functions (all natural weights were less than 10^{-11}). We speculate that the (very minor) dis-

crepancies between our results and those from Rakhuba and Oseledets exist because they fitted the potential to a tensor train (matrix product operator) format. For this reason, in order to get very accurate reference values for the spectrum to compare to, we performed an additional calculation without the approximation of a fit. For that, we used an MPS with bond dimension of $n_{\text{max}} = 100$ without bond adaption and without state averaging.

The performance of the TTNS-based optimization, compared to that of ML-MCTDH is shown in Fig. 14, both for the ground state and the lowest 13 states (state averaged). Already after three TTNS iterations is the ground state converged to about 0.01 cm^{-1} . In contrast, ML-MCTDH improved relaxation requires 17 iterations and an imaginary time propagation 34 fs in order to reach the same accuracy. State-average optimizations show a similar behavior. There, the TTNS optimization is already converged after just one iteration whereas the ML-MCTDH optimization requires 8 iterations to reach a similar accuracy ($\sim 1 \text{ cm}^{-1}$, using the same tensor sizes as for the ground state minimization). However, due to the approximate canonicalization for state-averaged root nodes (see Section II D 3) in TTNS, the ML-MCTDH calculations can give slightly higher accuracies for a given bond dimension. Nevertheless, the state-average TTNS calculation can always be followed by additional state-specific calculations using, e.g., shifted Hamiltonians (see Section II D 2).

It would be useful to compare runtimes instead of iterations. However, our ML-MCTDH code is not fully optimized and the performance of the propagation in ML-MCTDH is very sensitive the choice of propagator. Nonetheless, one TTNS iteration is approximately comparable to one ML-MCTDH iteration with respect to runtime¹⁰⁶ and the TTNS optimization converges much faster such that it should also be faster in runtime.

We now discuss the accurate state-average TTNS optimizations for the whole spectrum. Also in this case, we noticed a very rapid convergence of the minimization. Similar to the previous observations, for the loose calculations with 20 averaged states, starting from a random initial state, already after just one sweep were the first 14 eigenstates converged to within a few cm^{-1} . Thus, no two-site algorithm is necessary in this case.

The energy levels and errors are shown in Table II. Note that negative errors for the TT calculations do not mean that the results are more accurate (because a fit of the potential is used; see above). The absolute errors are also plotted in Fig. 15.

The obtained accuracy is very good. For the TTNS with $n_{\text{max}} = 50$, the maximal absolute error in the levels is 0.015 cm^{-1} . For the TTNS with $n_{\text{max}} = 40$, the maximal error is 0.036 cm^{-1} . The errors of the MPS are comparable, although slightly larger for $n_{\text{max}} = 25$, compared to the same TTNS. The TT calculations from Rakhuba and Oseledets with fixed bond dimension give very similar errors, compared to our MPS calculations with variable bond dimension.

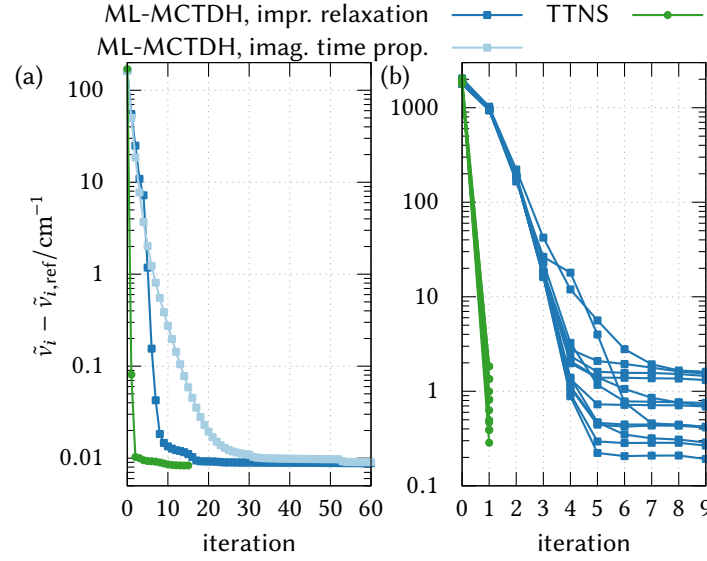


FIG. 14. Absolute errors of the ground state (left panel) and the lowest 13 states (state averaged, right panel) of acetonitrile, comparing optimization based on tree tensor network state (TTNS) optimization (green, showing the error after each sweep), the multilayer multi-configuration time-dependent Hartree method (ML-MCTDH) with improved relaxation (dark blue, showing the error after each diagonalization) and ML-MCTH with imaginary time propagation (pale blue, showing the error after each time step of 1 fs). All calculations used the same tensor sizes (see text). Thus, the ground state calculation is more accurate.

TABLE II: Vibrational levels $\tilde{\nu}_i$ and absolute errors (in cm^{-1}) of acetonitrile. The error is represented with respect to calculations from a matrix product state (MPS) with (fixed) bond dimension of $n = 100$. The tree tensor network states (TTNS) and MPS calculations were performed with an adaptive bond dimension with parameter ϵ (Section II E) and maximal bond dimension of n_{\max} . The results from the Smolyak quadrature are taken from Ref. 98, The results from the tensor trains (TT) with fixed bond dimension are taken from Ref. 32. The assignment is according to Ref. 98.

$n_{\max} =$	Ref. MPS	Smolyak	TT	TT	TTNS	TTNS	TTNS	MPS	MPS
$\epsilon =$	100		40	25	25	40	50	25	40
Level i	$\tilde{\nu}_{i,\text{ref}}/\text{cm}^{-1}$	$(\tilde{\nu}_i - \tilde{\nu}_{i,\text{ref}})/\text{cm}^{-1}$							
ZPVE	9837.4069	0.0004	-0.0006	0.0004	0.0013	0.0001	0.0000	0.0015	0.0001
ν_{11}	360.990	0.001	0.000	0.000	0.015	0.001	0.000	0.007	0.001
	360.990	0.001	0.000	0.000	0.016	0.001	0.000	0.007	0.001
$2\nu_{11}$	723.179	0.002	0.001	0.000	0.040	0.001	0.001	0.024	0.002
	723.179	0.002	0.001	0.000	0.040	0.001	0.001	0.024	0.002
$2\nu_{11}$	723.825	0.002	0.001	0.001	0.044	0.001	0.001	0.031	0.003
ν_4	900.659	0.003	-0.001	-0.003	0.001	0.000	0.000	0.003	0.001
ν_9	1034.124	0.002	0.000	0.001	0.017	0.001	0.000	0.009	0.001
	1034.124	0.002	0.000	0.001	0.020	0.001	0.000	0.010	0.001
$3\nu_{11}$	1086.552	0.002	0.000	0.000	0.061	0.005	0.002	0.042	0.004
$3\nu_{11}$	1086.552	0.002	0.001	0.001	0.061	0.004	0.002	0.041	0.004
$3\nu_{11}$	1087.774	0.002	0.001	0.001	0.069	0.005	0.002	0.058	0.008
	1087.774	0.002	0.001	0.001	0.069	0.005	0.002	0.058	0.007
$\nu_4 + \nu_{11}$	1259.809	0.073	0.000	-0.071	0.024	0.001	0.001	0.036	0.004
	1259.809	0.073	0.000	-0.071	0.024	0.001	0.001	0.036	0.004
ν_3	1388.967	0.006	0.004	0.038	0.027	0.002	0.000	0.029	0.002
$\nu_9 + \nu_{11}$	1394.679	0.010	0.003	0.015	0.046	0.005	0.002	0.025	0.003
	1394.679	0.010	0.003	0.023	0.047	0.005	0.002	0.026	0.003
$\nu_9 + \nu_{11}$	1394.898	0.009	0.002	0.011	0.046	0.004	0.001	0.023	0.003
$\nu_9 + \nu_{11}$	1397.680	0.007	0.004	0.044	0.032	0.002	0.001	0.035	0.003
$4\nu_{11}$	1451.093	0.008	0.000	-0.006	0.065	0.012	0.002	0.063	0.010
	1451.093	0.008	0.000	-0.006	0.065	0.012	0.002	0.063	0.010
$4\nu_{11}$	1452.818	0.009	0.001	-0.005	0.068	0.012	0.003	0.084	0.014
	1452.818	0.009	0.001	-0.005	0.068	0.012	0.003	0.086	0.014
$4\nu_{11}$	1453.394	0.009	0.001	-0.005	0.006	0.000	0.000	0.012	0.002

ν_7	1483.219	0.010	0.001	-0.003	0.026	0.002	0.001	0.032	0.002
	1483.220	0.009	0.001	-0.002	0.030	0.002	0.001	0.032	0.002
$\nu_4 + 2\nu_{11}$	1620.199	0.023	-0.001	-0.021	0.040	0.008	0.001	0.100	0.010
	1620.199	0.023	-0.001	-0.021	0.040	0.008	0.001	0.100	0.010
$\nu_4 + 2\nu_{11}$	1620.744	0.023	-0.001	-0.020	0.042	0.008	0.002	0.125	0.015
$\nu_3 + \nu_{11}$	1749.519	0.011	0.006	0.067	0.201	0.011	0.003	0.108	0.010
	1749.519	0.011	0.008	0.090	0.202	0.011	0.003	0.109	0.011
$\nu_9 + 2\nu_{11}$	1756.412	0.014	0.007	0.056	0.123	0.009	0.005	0.061	0.010
$\nu_9 + 2\nu_{11}$	1756.412	0.014	0.007	0.057	0.124	0.010	0.005	0.061	0.010
$\nu_9 + 2\nu_{11}$	1757.119	0.014	0.004	0.019	0.122	0.009	0.005	0.070	0.012
	1757.119	0.014	0.005	0.026	0.122	0.009	0.005	0.073	0.012
$\nu_9 + 2\nu_{11}$	1759.760	0.012	0.008	0.084	0.121	0.012	0.004	0.122	0.016
	1759.760	0.012	0.010	0.097	0.129	0.013	0.004	0.131	0.017
$2\nu_4$	1785.177	0.030	-0.057	-0.136	0.016	0.000	0.000	0.019	0.001
$5\nu_{11}$	1816.786	0.013	0.001	-0.009	0.090	0.004	0.001	0.032	0.005
	1816.786	0.013	0.001	-0.009	0.090	0.004	0.001	0.032	0.005
$5\nu_{11}$	1818.938	0.014	0.002	-0.007	0.078	0.008	0.002	0.092	0.018
$5\nu_{11}$	1818.939	0.013	0.002	-0.006	0.078	0.008	0.002	0.092	0.018
$5\nu_{11}$	1820.016	0.015	0.001	-0.009	0.083	0.008	0.002	0.109	0.021
	1820.016	0.015	0.001	-0.009	0.083	0.009	0.002	0.110	0.021
$\nu_7 + \nu_{11}$	1844.245	0.013	0.005	0.059	0.100	0.007	0.003	0.065	0.010
$\nu_7 + \nu_{11}$	1844.316	0.014	0.006	0.063	0.100	0.007	0.004	0.065	0.010
	1844.317	0.013	0.005	0.064	0.101	0.007	0.004	0.066	0.010
$\nu_7 + \nu_{11}$	1844.676	0.014	0.005	0.060	0.108	0.007	0.004	0.068	0.011
$\nu_4 + \nu_9$	1931.514	0.033	0.000	-0.024	0.050	0.004	0.001	0.041	0.005
	1931.514	0.033	0.001	-0.023	0.060	0.004	0.001	0.043	0.006
$\nu_4 + 3\nu_{11}$	1981.815	0.034	0.000	-0.028	0.100	0.013	0.004	0.188	0.023
$\nu_4 + 3\nu_{11}$	1981.815	0.035	0.000	-0.029	0.100	0.013	0.004	0.188	0.023
$\nu_4 + 3\nu_{11}$	1982.818	0.039	-0.002	-0.036	0.113	0.014	0.005	0.230	0.042
	1982.818	0.039	-0.002	-0.036	0.113	0.015	0.005	0.230	0.043
$2\nu_9$	2057.044	0.024	0.004	0.007	0.026	0.001	0.001	0.013	0.002
$2\nu_9$	2065.265	0.021	0.002	-0.012	0.137	0.005	0.003	0.036	0.004
	2065.265	0.021	0.003	0.014	0.137	0.006	0.003	0.038	0.005
$\nu_3 + 2\nu_{11}$	2111.364	0.016	0.013	0.144	0.160	0.020	0.011	0.307	0.024
	2111.364	0.016	0.015	0.200	0.160	0.020	0.011	0.310	0.024
$\nu_3 + 2\nu_{11}$	2112.281	0.016	0.013	0.183	0.181	0.022	0.012	0.340	0.035
$\nu_9 + 3\nu_{11}$	2119.307	0.020	0.010	0.079	0.218	0.035	0.014	0.122	0.019
	2119.307	0.020	0.010	0.105	0.218	0.035	0.014	0.122	0.019
$\nu_9 + 3\nu_{11}$	2120.521	0.020	0.008	0.050	0.229	0.036	0.015	0.152	0.033
	2120.521	0.020	0.009	0.054	0.231	0.036	0.015	0.159	0.034
$\nu_9 + 3\nu_{11}$	2120.889	0.021	0.008	0.041	0.223	0.034	0.014	0.154	0.034
$\nu_9 + 3\nu_{11}$	2122.816	0.018	0.019	0.030	0.208	0.033	0.014	0.359	0.043
	2122.816	0.018	0.022	0.185	0.209	0.033	0.014	0.362	0.043
$\nu_9 + 3\nu_{11}$	2123.282	0.019	0.018	0.136	0.229	0.036	0.015	0.423	0.055
$2\nu_4 + \nu_{11}$	2142.444	0.170	-0.065	-0.289	0.094	0.004	0.001	0.090	0.008
	2142.444	0.170	-0.065	-0.289	0.094	0.004	0.001	0.092	0.008
$6\nu_{11}$	2183.617	0.018	0.002	-0.011	0.110	0.005	0.004	0.034	0.006
	2183.617	0.018	0.002	-0.011	0.110	0.005	0.004	0.034	0.006
$6\nu_{11}$	2186.117	0.021	0.002	-0.012	0.084	0.011	0.004	0.111	0.022
	2186.117	0.021	0.002	-0.012	0.084	0.011	0.004	0.111	0.022
$6\nu_{11}$	2187.618	0.024	0.003	-0.013	0.093	0.005	0.003	0.125	0.015
	2187.618	0.024	0.003	-0.013	0.093	0.005	0.003	0.125	0.015
$6\nu_{11}$	2188.119	0.025	0.003	-0.014	0.019	0.005	0.004	0.136	0.016
$\nu_7 + 2\nu_{11}$	2206.608	0.018	0.007	0.121	0.127	0.026	0.004	0.171	0.015
$\nu_7 + 2\nu_{11}$	2206.615	0.018	0.009	0.148	0.125	0.026	0.004	0.168	0.014
$\nu_7 + 2\nu_{11}$	2206.757	0.009	0.010	0.054	0.131	0.027	0.005	0.173	0.018
	2206.758	0.008	0.010	0.120	0.136	0.027	0.005	0.176	0.018
$\nu_7 + 2\nu_{11}$	2207.541	0.018	0.008	0.053	0.132	0.027	0.005	0.190	0.023
	2207.541	0.018	0.009	0.065	0.140	0.027	0.005	0.199	0.026

Fig. 16 shows the number of parameters for each eigenstate. This quantity roughly gives an estimate on the required effort of the calculations as it depends on the

bond dimensions. The adaptive bond dimension gives significant improvement, compared to a fixed bond dimension. For $n_{\max} = 40$, the average number of pa-

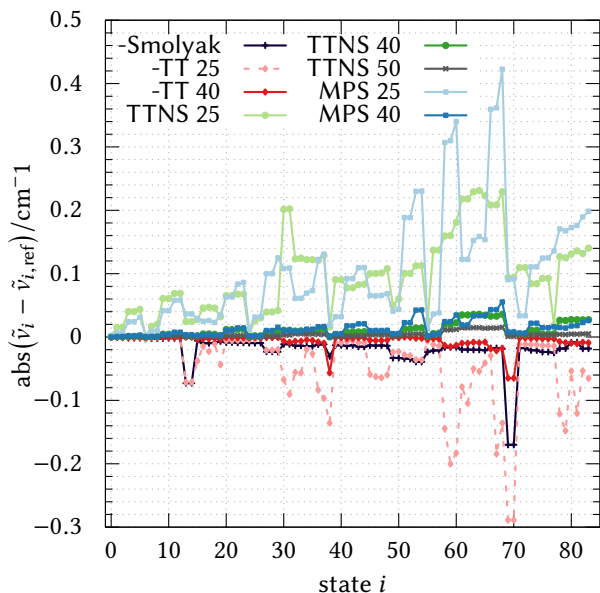


FIG. 15. Absolute errors of the energy levels for Smolyak,⁹⁸ tensor train (TT),³² tree tensor network state (TTNS) and matrix product state (MPS) calculations. The number behind the names represent the maximally allowed bond dimension. See Table II for more details. For clarity, the Smolyak and TT results are plotted on the negative part of the ordinate, even though absolute errors are shown.

rameters is around $8 \cdot 10^4$. A fixed bond dimension of $n = 40$ would give around $1.4 \cdot 10^5$ number of parameters whereas a fixed bond dimension of $n = 20$ would give $6 \cdot 10^4$ number of parameters. Thus, an adaptive calculation with $n_{\max} = 40$ requires only marginally more parameters than a calculation with fixed $n = 25$.

We note that it is not necessarily the case that the required bond dimension grows with the energy of the eigenstate. Instead, the number of parameters is roughly the same for all computed eigenstates and only oscillates significantly for some states (often to a lower number of parameters).

We now discuss the results of the proof-of-concept optimization of the tree. We note that, even though we did an overly exhaustive optimization with about 4000×190 SVD calculations, the optimization was extremely fast and only took about one minute on a standard computer.

The optimized tree is shown in panel (b) in Fig. 13. Compared to the non-optimized tree (panel (a)), the physical dimension 3 is now closer in the tree to dimensions 2 and 4. Instead of dimensions 11 and 12, the nodes with physical dimensions 2 and 4 are now in the deepest layer. Note the blue tensor in Fig. 13 above dimension 1. This tensor is redundant and can be removed from the tree.¹⁰⁷ Otherwise, there are no changes. This simply means that the initially used tree already is very good. More exhaustive global optimization procedures may result in different trees. We point out that using an MPS as initial tree (not shown) also leads to similar “clusters”

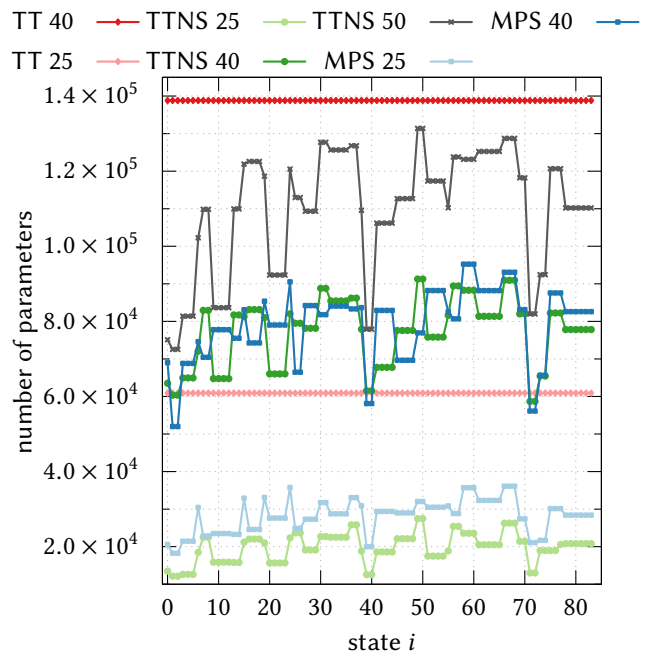


FIG. 16. Total number of parameters for each state for the tensor train (TT),³² tree tensor network state (TTNS) and matrix product state (MPS) calculations. The number behind the names represent the maximally allowed bond dimension. The TT calculations use constant bond dimensions.

of tensors, including mode combination, that is, several physical dimensions on one tensor. However, the optimized trees that are based on an MPS still consists of larger linear chains and only some tensors cluster in branches of this chain. At any rate, the tree optimization procedure also improves very bad initial trees.

Throughout, including tree optimizations for other states and based on other initial trees, we found that three-dimensional tensors in the tree are almost exclusively selected. This empirically confirms the fact that three-dimensional tensors have the lowest scaling of number of entries with respect to dimensionality and are still possible to use within TTNS and MPS.^{20,46,48}

Compared to the initial tree, the optimized tree results in a similar, slightly lower accuracy for the eigenstates. For example, for the setup with maximal bond dimension of 25, the maximal absolute error of the initial tree is 0.23 cm^{-1} whereas it is 0.21 cm^{-1} in the optimized tree. The ratios of number of parameters for the initial tree, compared to the optimized tree, are shown in Fig. 17. Even though the tree optimization is based on only the 10th eigenstate, almost all eigenstates slightly reduce the total number of parameters, which is proportional to the bond dimension, by around 10 to 5%. Only for the higher lying eigenstates is the ratio worse. While this result is not overly impressive, we again point out that the initial tree is already very good and that this proof-of-concept optimization can be improved by using global optimization procedures.

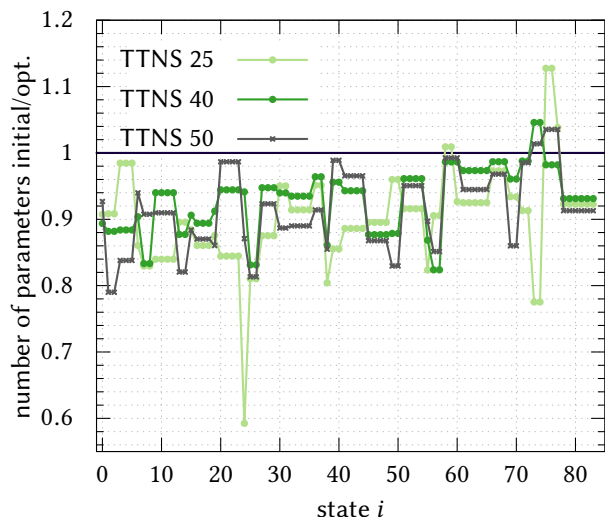


FIG. 17. Ratios of number of parameters for the initial tree (see Fig. 13 (a)), compared to the optimized tree (Fig. 13 (b)). The number behind the names represent the maximally allowed bond dimension; compare with the labeling in Table II.

Interestingly, compared to MPS, TTNS need a similar amount of parameters. Only for $n_{\max} = 25$, the number of parameters of the TTNS is slightly lower and the accuracy slightly better, compared to MPS. For $n_{\max} = 25$, the optimized tree requires around 70 % of the number of parameters of the MPS. However for higher accuracies (larger maximal bond dimensions), it seems that, for this test case, MPS perform similarly or equally well than TTNS. For other systems, in particular for much higher-dimensional systems, TTNS may show a more significant advantage over MPS.

IV. CONCLUSIONS AND OUTLOOK

In this work, we have transferred the main algorithm and the diagrammatic language from the context of the density matrix renormalization group (DMRG) to tree tensor network states (TTNS) for computing vibrational eigenstates. This complements the well-established methodology of the multilayer multi-configuration time-dependent Hartree (ML-MCTDH) approach. In particular, we showed the advantages of using the sweep algorithm from the DMRG for vibrational calculations.

We tested TTNS and matrix product states (MPS, the *ansatz* behind the DMRG) for the 12-dimensional benchmark system acetonitrile (CH_3CN). We could obtain very accurate energy levels with absolute errors less than 0.04 cm^{-1} for moderate tensor sizes (bond dimensions). The TTNS calculations converge much faster than ML-MCTDH-based calculations. Further, we showed that using adaptive tensor sizes leads to a significant reduction of the number of parameters without jeopardizing

accuracy.

Using the same code and algorithms both for MPS and TTNS allow for a direct comparison between these two tensor networks. For CH_3CN , TTNS give slightly improved accuracies with lower numbers of parameters than MPS. Compared to MPS, an optimized tree reduces the number of parameters to about 70 % for medium accuracies. However, for higher accuracies, the difference in the number of parameters is not significant and MPS perform very well and are (slightly) simpler to implement. Nevertheless, TTNS with only three-dimensional tensors have the same computational scaling as MPS for a Hamiltonian in sum-of-product form such that it seems unlikely that TTNS will show any disadvantages, compared to MPS, besides a slightly more complicated implementation. Thus, a TTNS could mostly be favored over an MPS. In any case, more tests with other systems (including more complex potential energy surfaces and higher dimensionalities) need to be performed in order to pinpoint the advantages (and disadvantages) of using TTNS over MPS.

Additionally, we presented a simple procedure how to optimize (“disentangle”) a tree by randomly swapping and moving dimensions between neighboring tensors. As proof of concept, we showed how to optimize the tree for CH_3CN , reducing the number of required parameters by around 5 % to 10 %, compared to a tree based on a *very good* initial guess. We remark that this optimization could also be performed separately for each eigenstate or even *during* a time propagation, where, after some time steps, the tree changes in order to reduce the required bond dimensions.

Here, we presented only the basic methodology of TTNS and there are many possible improvements. In particular, almost all sophisticated advances in the DMRG can straightforwardly be implemented for TTNS. For example, there has been much work on improving excited state calculation. Another way of improvement is the usage of dynamical or adaptive pruning (which is similar to selected configuration interaction) to further decrease the required number of parameters. Previous studies of dynamical pruning in combination with the MCTDH algorithm are very promising.^{91,92,108} At any rate, we plan to apply TTNS and MPS to larger systems with more complicated potential energy surfaces.

Furthermore, DMRG-based algorithms for time evolution^{35,36,38,109,110} can straightforwardly be applied to TTNS.^{39,111} It will be very interesting to see how they compare to the MCTDH-based algorithms. Also, we believe that the diagrammatic notation used in the DMRG community and in this work will highlight new facets of established MCTDH methodology. This will lead to better understanding and maybe even lead to further improvements that enable the computation of challenging systems.

ACKNOWLEDGMENTS

The author is thankful to G. K. Chan, K. Gunst and R. Haghsheenas for helpful discussions. He acknowledges support from the German Research Foundation (DFG) via grant LA 4442/1-1.

- ¹H.-D. Meyer, U. Manthe, and L. Cederbaum, *Chem. Phys. Lett.* **165**, 73 (1990).
- ²U. Manthe, H. Meyer, and L. S. Cederbaum, *J. Chem. Phys.* **97**, 3199 (1992).
- ³M. H. Beck, A. Jäckle, G. A. Worth, and H.-D. Meyer, *Phys. Rep.* **324**, 1 (2000).
- ⁴H.-D. Meyer, F. Gatti, and G. A. Worth, eds., *Multidimensional Quantum Dynamics: MCTDH Theory and Applications*, 1st ed. (Wiley-VCH, 2009).
- ⁵T. G. Kolda and B. W. Bader, *SIAM Rev.* **51**, 455 (2009).
- ⁶F. Huarte-Larrañaga and U. Manthe, *J. Phys. Chem. A* **105**, 2522 (2001).
- ⁷G. A. Worth, H.-D. Meyer, and L. S. Cederbaum, *J. Chem. Phys.* **109**, 3518 (1998).
- ⁸H. Wang, *J. Chem. Phys.* **113**, 9948 (2000).
- ⁹M. Nest and H.-D. Meyer, *J. Chem. Phys.* **119**, 24 (2003).
- ¹⁰H. Wang and M. Thoss, *J. Chem. Phys.* **119**, 1289 (2003).
- ¹¹U. Manthe, *J. Chem. Phys.* **128**, 164116 (2008).
- ¹²O. Vendrell and H.-D. Meyer, *J. Chem. Phys.* **134**, 044135 (2011).
- ¹³H. Wang, *J. Phys. Chem. A* **119**, 7951 (2015).
- ¹⁴U. Manthe, *J. Phys. Condens. Matter* **29**, 253001 (2017).
- ¹⁵W. Hackbusch and S. Kühn, *J. Fourier Anal. Appl.* **15**, 706 (2009).
- ¹⁶L. Grasedyck, *SIAM J. Matrix Anal. and Appl.* **31**, 2029 (2010).
- ¹⁷W. Hackbusch, *Tensor Spaces and Numerical Tensor Calculus* (Springer, 2012).
- ¹⁸Y.-Y. Shi, L.-M. Duan, and G. Vidal, *Phys. Rev. A* **74**, 022320 (2006).
- ¹⁹L. Tagliacozzo, G. Evenbly, and G. Vidal, *Phys. Rev. B* **80**, 235127 (2009).
- ²⁰V. Murg, F. Verstraete, Ö. Legeza, and R. M. Noack, *Phys. Rev. B* **82**, 205105 (2010).
- ²¹G. K.-L. Chan, *WIREs Comput. Mol. Sci.* **2**, 907 (2012).
- ²²R. Orús, *Ann. Phys.* **349**, 117 (2014).
- ²³Y. Xie, J. Zheng, and Z. Lan, *J. Chem. Phys.* **142**, 084706 (2015).
- ²⁴J. Schulze, M. F. Shibli, M. J. Al-Marri, and O. Kühn, *J. Chem. Phys.* **144**, 185101 (2016).
- ²⁵D. Mendive-Tapia, E. Mangaud, T. Firmino, A. de la Lande, M. Desouter-Lecomte, H.-D. Meyer, and F. Gatti, *J. Phys. Chem. B* **122**, 126 (2018).
- ²⁶R. Orús, *Eur. Phys. J. B* **87**, 280 (2014).
- ²⁷U. Schollwöck, *Ann. Phys.* **326**, 96 (2011).
- ²⁸S. R. White, *Phys. Rev. Lett.* **69**, 2863 (1992).
- ²⁹S. R. White, *Phys. Rev. B* **48**, 10345 (1993).
- ³⁰I. V. Oseledets, *SIAM J. Sci. Comput.* **33**, 2295 (2011).
- ³¹G. K.-L. Chan, *Phys. Chem. Chem. Phys.* **10**, 3454 (2008).
- ³²M. Rakhuba and I. Oseledets, *J. Chem. Phys.* **145**, 124101 (2016).
- ³³S. M. Greene and V. S. Batista, *J. Chem. Theory Comput.* **13**, 4034 (2017).
- ³⁴A. Baiardi, C. J. Stein, V. Barone, and M. Reiher, *J. Chem. Theory Comput.* **13**, 3764 (2017).
- ³⁵J. Ren, Z. Shuai, and G. Kin-Lic Chan, *J. Chem. Theory Comput.* **14**, 5027 (2018).
- ³⁶Y. Kurashige, *J. Chem. Phys.* **149**, 194114 (2018).
- ³⁷A. Baiardi, C. J. Stein, V. Barone, and M. Reiher, *J. Chem. Phys.* **150**, 094113 (2019).
- ³⁸A. Baiardi and M. Reiher, *J. Chem. Theory Comput.* **15**, 3481 (2019).
- ³⁹F. A. Y. N. Schröder, D. H. P. Turban, A. J. Musser, N. D. M. Hine, and A. W. Chin, *Nat. Commun.* **10**, 10:1062 (2019).
- ⁴⁰D. Iouchtchenko and P.-N. Roy, *J. Chem. Phys.* **148**, 134115 (2018).
- ⁴¹P. S. Thomas and T. Carrington, *J. Phys. Chem. A* **119**, 13074 (2015).
- ⁴²T. Hammer and U. Manthe, *J. Chem. Phys.* **136**, 054105 (2012).
- ⁴³R. Wodraszka and U. Manthe, *J. Chem. Phys.* **136**, 124119 (2012).
- ⁴⁴H. Wang, *J. Phys. Chem. A* **118**, 9253 (2014).
- ⁴⁵H. J. Changlani, S. Ghosh, C. L. Henley, and A. M. Läuchli, *Phys. Rev. B* **87**, 085107 (2013).
- ⁴⁶N. Nakatani and G. K.-L. Chan, *J. Chem. Phys.* **138**, 134113 (2013).
- ⁴⁷M. Gerster, P. Silvi, M. Rizzi, R. Fazio, T. Calarco, and S. Montangero, *Phys. Rev. B* **90**, 125154 (2014).
- ⁴⁸K. Gunst, F. Verstraete, S. Wouters, Ö. Legeza, and D. Van Neck, *J. Chem. Theory Comput.* **14**, 2026 (2018).
- ⁴⁹A. Leclerc and T. Carrington, *J. Chem. Phys.* **140**, 174111 (2014).
- ⁵⁰P. S. Thomas and T. Carrington, *J. Chem. Phys.* **146**, 204110 (2017).
- ⁵¹P. S. Thomas, T. Carrington, J. Agarwal, and H. F. Schaefer, *J. Chem. Phys.* **149**, 064108 (2018).
- ⁵²F. Otto, *J. Chem. Phys.* **140**, 014106 (2014).
- ⁵³O. Vendrell, F. Gatti, and H.-D. Meyer, *Angew. Chem. Int. Ed.* **46**, 6918 (2007).
- ⁵⁴Not decomposing the tensor \mathbf{B} leads to the so-called mode-combination approach in standard MCTDH.⁷ It allows for simulating larger systems but is still inherently limited with respect to dimensionality, even though improvements are possible.⁹²
- ⁵⁵V. Murg, F. Verstraete, R. Schneider, P. R. Nagy, and Ö. Legeza, *J. Chem. Theory Comput.* **11**, 1027 (2015).
- ⁵⁶While here we only use real-valued tensors, we hint at complex conjugation in order to be general and to distinguish whether $|\Psi\rangle$ or its dual $\langle\Psi|$ is represented in a particular diagram.
- ⁵⁷Another frequently used notation is to use arrows or triangles to specify the matricization.^{22,109}
- ⁵⁸G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. (Johns Hopkins University Press, 2013).
- ⁵⁹N. K. Madsen, M. B. Hansen, A. Zocante, K. Monrad, M. B. Hansen, and O. Christiansen, *J. Chem. Phys.* **149**, 134110 (2018).
- ⁶⁰L. J. Doriol, F. Gatti, C. Iung, and H.-D. Meyer, *J. Chem. Phys.* **129**, 224109 (2008).
- ⁶¹A. Jäckle and H.-D. Meyer, *J. Chem. Phys.* **104**, 7974 (1996).
- ⁶²D. Peláez and H.-D. Meyer, *J. Chem. Phys.* **138**, 014108 (2013).
- ⁶³M. Schröder and H.-D. Meyer, *J. Chem. Phys.* **147**, 064105 (2017).
- ⁶⁴S. Manzhos and T. Carrington, *J. Chem. Phys.* **125**, 194105 (2006).
- ⁶⁵W. Koch and D. H. Zhang, *J. Chem. Phys.* **141**, 021101 (2014).
- ⁶⁶The kinetic energy operator very often is already in SOP form.
- ⁶⁷R. Wodraszka and T. Carrington, *J. Chem. Phys.* **148**, 044115 (2018).
- ⁶⁸T. Xiang, *Phys. Rev. B* **53**, R10445 (1996).
- ⁶⁹H.-D. Meyer, F. L. Quéré, C. Léonard, and F. Gatti, *Chem. Phys.* **329**, 179 (2006).
- ⁷⁰F. Culot and J. Liévin, *Theor. Chim. Acta* **89**, 227 (1994).
- ⁷¹K. Drukker and S. Hammes-Schiffer, *J. Chem. Phys.* **107**, 363 (1997).
- ⁷²S. Heislbetz and G. Rauhut, *J. Chem. Phys.* **132**, 124102 (2010).
- ⁷³S. R. White, *Phys. Rev. B* **72**, 180403(R) (2005).
- ⁷⁴C. Hubig, I. P. McCulloch, U. Schollwöck, and F. A. Wolf, *Phys. Rev. B* **91**, 155115 (2015).
- ⁷⁵C. Lubich, *Appl. Math. Res. Express*, 311 (2015).
- ⁷⁶B. Kloss, I. Burghardt, and C. Lubich, *J. Chem. Phys.* **146**, 174107 (2017).
- ⁷⁷K.-S. Lee and U. R. Fischer, *Int. J. Mod. Phys. B* **28**, 1550021 (2014).

- ⁷⁸U. Manthe, *J. Chem. Phys.* **142**, 244109 (2015).
- ⁷⁹H.-D. Meyer and H. Wang, *J. Chem. Phys.* **148**, 124105 (2018).
- ⁸⁰H. Wang and H.-D. Meyer, *J. Chem. Phys.* **149**, 044119 (2018).
- ⁸¹J. J. Dorando, J. Hachmann, and G. K.-L. Chan, *J. Chem. Phys.* **127**, 084109 (2007).
- ⁸²W. Hu and G. K.-L. Chan, *J. Chem. Theory Comput.* **11**, 3000 (2015).
- ⁸³L. Zhao and E. Neuscamman, *J. Chem. Theory Comput.* **12**, 3436 (2016).
- ⁸⁴L. N. Tran, J. A. R. Shea, and E. Neuscamman, *J. Chem. Theory Comput.* **15**, 4790 (2019).
- ⁸⁵R. Kosloff and H. Tal-Ezer, *Chem. Phys. Lett.* **127**, 223 (1986).
- ⁸⁶S. Wouters, W. Poelmans, P. W. Ayers, and D. Van Neck, *Comput. Phys. Commun.* **185**, 1501 (2014).
- ⁸⁷U. Manthe, *J. Chem. Phys.* **128**, 064108 (2008).
- ⁸⁸E. Ronca, Z. Li, C. A. Jimenez-Hoyos, and G. K.-L. Chan, *J. Chem. Theory Comput.* **13**, 5560 (2017).
- ⁸⁹Ö. Legeza, J. Röder, and B. A. Hess, *Phys. Rev. B* **67**, 125114 (2003).
- ⁹⁰D. Mendive-Tapia, T. Firmino, H.-D. Meyer, and F. Gatti, *Chem. Phys.* **482**, 113 (2017).
- ⁹¹R. Wodraszka and T. Carrington, *J. Chem. Phys.* **146**, 194105 (2017).
- ⁹²H. R. Larsson and D. J. Tannor, *J. Chem. Phys.* **147**, 044103 (2017).
- ⁹³In practice, the bond dimension is increased by one after every adaptation. This avoids oscillations of the bond dimension.
- ⁹⁴G. K.-L. Chan and M. Head-Gordon, *J. Chem. Phys.* **116**, 4462 (2002).
- ⁹⁵R. Olivares-Amaya, W. Hu, N. Nakatani, S. Sharma, J. Yang, and G. K.-L. Chan, *J. Chem. Phys.* **142**, 034102 (2015).
- ⁹⁶T. Weise, *Global Optimization Algorithms*, 2nd ed. (www.it-weise.de, 2009).
- ⁹⁷D. Begue, P. Carbonniere, and C. Pouchan, *J. Phys. Chem. A* **109**, 4611 (2005).
- ⁹⁸G. Avila and T. Carrington, *J. Chem. Phys.* **134**, 054126 (2011).
- ⁹⁹G. A. Worth, M. H. Beck, A. Jäckle, and H.-D. Meyer, *The MCTDH Package*, Version 8.4.17 (2019). See <http://mctdh.uni-hd.de>.
- ¹⁰⁰D. J. Tannor, *Introduction to Quantum Mechanics: A Time-Dependent Perspective*, 1st ed. (University Science Books, 2007).
- ¹⁰¹S. van der Walt, S. C. Colbert, and G. Varoquaux, *Comput. Sci. Eng.* **13**, 22 (2011).
- ¹⁰²E. Jones, T. Oliphant, P. Peterson, *et al.*, “SciPy: Open source scientific tools for Python,” (2001), <http://www.scipy.org/>.
- ¹⁰³H. R. Larsson, B. Hartke, and D. J. Tannor, *J. Chem. Phys.* **145**, 204108 (2016).
- ¹⁰⁴E. R. Davidson, *J. Comp. Phys.* **17**, 87 (1975).
- ¹⁰⁵Q. Sun, T. C. Berkelbach, N. S. Blunt, G. H. Booth, S. Guo, Z. Li, J. Liu, J. D. McClain, E. R. Sayfutyarova, S. Sharma, S. Wouters, and G. K.-L. Chan, *WIREs Comput. Mol. Sci.* **8**, 1340 (2017).
- ¹⁰⁶For TTNS, each iteration requires the diagonalization of each tensor and local (from one node to a neighboring node) transformation of the Hamiltonian whereas for ML-MCTDH with improved relaxation, each iteration requires imaginary time propagation of the single particle functions, followed by a global transformations of the Hamiltonian (creation of the mean-fields), computation and inversion of density matrices and diagonalization of the root node.
- ¹⁰⁷Tensor removal and addition is not possible with the current optimization procedure but could be performed manually.
- ¹⁰⁸F. Köhler, K. Keiler, S. I. Mistakidis, H.-D. Meyer, and P. Schmelcher, *J. Chem. Phys.* **151**, 054108 (2019).
- ¹⁰⁹J. Haegeman, C. Lubich, I. Oseledets, B. Vandereycken, and F. Verstraete, *Phys. Rev. B* **94**, 165116 (2016).
- ¹¹⁰S. Paeckel, T. Köhler, A. Swoboda, S. R. Manmana, U. Schollwöck, and C. Hubig, *Ann. Phys.* **411**, 167998 (2019).
- ¹¹¹D. Bauernfeind and M. Aichhorn, “Time dependent variational principle for tree tensor networks,” (2019), arXiv:1908.03090.