Parallel time-dependent variational principle algorithm for matrix product states (Supplemental Material)

Paul Secular,[1,*] Nikita Gourianov,[2] Michael Lubasch,[2] Sergey Dolgov,[3] Stephen R. Clark,[4,5] and Dieter Jaksch[2,6]

[1] *Department of Physics, University of Bath, Claverton Down, Bath BA2 7AY, UK*
[2] *Clarendon Laboratory, Department of Physics, University of Oxford, Oxford OX1 3PU, UK*
[3] *Department of Mathematical Sciences, University of Bath, Claverton Down, Bath BA2 7AY, UK*
[4] *H.H. Wills Physics Laboratory, University of Bristol, Bristol BS8 1TL, UK*
[5] *Max Planck Institute for the Structure and Dynamics of Matter, CFEL, 22761 Hamburg, Germany*
[6] *Centre for Quantum Technologies, National University of Singapore, 3 Science Drive 2, Singapore 117543, Singapore*

(Dated: June 12, 2020)

In this Supplemental Material we present technical information aimed at practitioners wishing to use the parallel two-site time dependent variational principle algorithm. We compare two different methods of approximating power laws by sums of exponentials, which should prove useful for calculations on larger systems. We also provide details of the software, settings, and parameters used to carry out our numerical experiments, in order to aid reproducibility.

S1. IMPLEMENTATION

The parallel two-site time-dependent variational principle (p2TDVP) code was built on top of the Tensor Network Theory (TNT) Library [S1–S4] – a C code that implements OpenMP [S5] shared memory parallelism [S2], and which supports multithreaded linear algebra libraries. p2TDVP was implemented using the Message Passing Interface (MPI) standard [S6]. This allowed us to target distributed memory architectures, hence increasing the total amount of random-access memory (RAM) available for simulations. For our benchmark calculations we employed a hybrid parallel approach in which the MPI network-level parallelism was combined with existing shared memory parallelism.

S2. APPROXIMATING POWER LAWS

The Hamiltonian matrix product operators (MPOs) used in our calculations were defined using a MATLAB [S8] interface written by Coulthard [S4]. This employs the method described by Pirvu *et al.* in the Appendix of Ref. [S7] to approximate power laws by sums of exponentials. Henceforth, when we refer to the MPO error, we mean the error in this approximation.

Although the approach suggested by Pirvu *et al.* is particularly fast and stable, it does not always give optimal results. In some cases it is preferable to employ a nonlinear least-squares fit. In Fig. S1 we compare the method from Ref. [S7] to the Levenberg-Marquardt nonlinear least-squares algorithm [S9, S10], by approximating $1/r^2$ as a sum of nine exponentials.

The Levenberg-Marquardt fit was calculated in MATLAB using the `lsqnonlin()` function [S11] with the options shown in Table S1. We find that the Levenberg-Marquardt method is slower but gives better results, comparable to those in Ref. [S12]. Using this method should thus allow for a smaller MPO bond dimension, and hence a slight speedup. More importantly, the fit holds over a longer distance, making it valuable for simulations with larger sized systems.

Option	Value
`Algorithm`	`levenberg-marquardt`
`MaxFunctionEvaluations`	`10000`
`MaxIterations`	`1000`
`StepTolerance`	`1E-6`
`FunctionTolerance`	`1E-14`

TABLE S1. Options used for the MATLAB `lsqnonlin()` function when calculating the fit shown in Fig. S1.

S3. TEST PLATFORM

We carried out our benchmarks on the Balena high performance computing (HPC) cluster [S13] at the University of Bath. We had access to a maximum of 32 compute nodes, with a maximum runtime per job of 5 days. All simulations were run on Dell PowerEdge C8220 nodes, which have two Intel E5-2650 v2 CPUs (20 MB Cache, 2.60 GHz base frequency), giving a total of 16 cores per node. Each node has a memory of 64 GB (8 GB × 8) DDR3 (1866 MHz).

S4. SIMULATION DETAILS

We linked the TNT Library to ARPACK-NG [S14], and the multithreaded Intel Math Kernel Library (MKL) [S15]. We used the same version of the Intel MPI Library [S16] and compiler, compiling with the `-O2` and `-xHost` optimization flags. We set the OpenMP/MKL environment variables shown in Table S2 to allow dynamic adjustment of the number of threads used (up to a maximum of 16), whilst also disabling nested threading.

* paul@secular.me.uk; http://secular.me.uk/

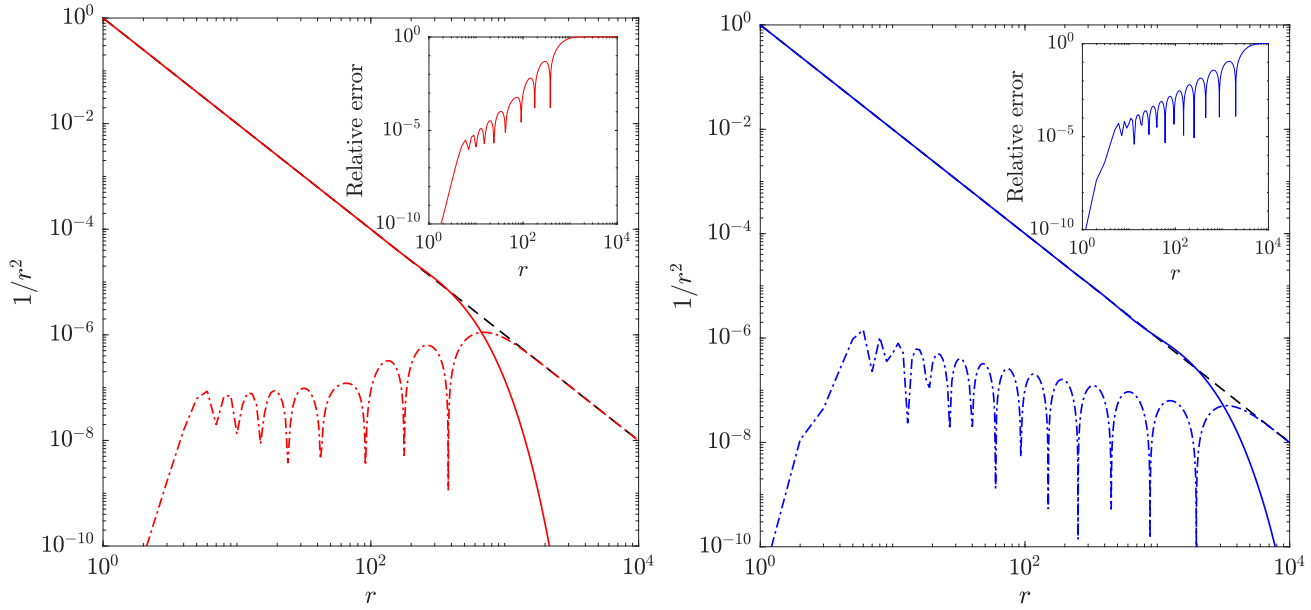

FIG. S1. Approximation of $1/r^2$ (dashed lines) by a sum of 9 exponentials (solid lines) using (left panel) the method described by Pirvu *et al.* in Ref. [S7], and (right panel) the Levenberg-Marquardt nonlinear least-squares algorithm discussed in the text. The broken lines show the absolute error in the approximations, and the insets show the relative error.

Environment variable	Value
`OMP_NUM_THREADS`	`16`
`MKL_NUM_THREADS`	`16`
`OMP_NESTED`	`FALSE`
`OMP_DYNAMIC`	`TRUE`
`MKL_DYNAMIC`	`TRUE`
`KMP_AFFINITY`	`compact,1,0,granularity=fine`

TABLE S2. Environment variables used to control OpenMP threading in the TNT Library and Intel MKL.

p	Number of sites owned by first process	Number of sites owned by central processes	Number of sites owned by last process
8	17	16	16
16	9	8	8
24	10	5	9
32	5	4	4

TABLE S3. Partitioning of the 129-site MPS in the long-range Ising model simulations for p parallel processes.

The linear algebra settings used for all calculations were as follows. We used the TNT Library default zero tolerance of 10^{-14} for the automatic blocking of matrices [S2]. We set a relative truncation tolerance of $\varepsilon = 10^{-12}$ for singular value decompositions (SVDs), and used the LAPACK [S17] dense matrix divide-and-conquer [S18–S20] routine (as implemented in Intel MKL). For the Lanczos exponentiation in p2TDVP, we created the Krylov subspace using ARPACK-NG with a maximum of 8 basis vectors, and a convergence tolerance of 10^{-6}. Density matrix renormalization group (DMRG) calculations used the ARPACK-NG sparse eigenvalue solver [S14, S21] with these same settings.

A. Long-range Ising model

All Hamiltonian MPOs had a maximum absolute error $\lesssim 10^{-8}$. The other parameters are as described in the main text. The ground state matrix product state (MPS) was calculated using two-site DMRG, and was found to have a maximum bond dimension of $\chi = 22$. For the p2TDVP calculations, the 129-site MPS was partitioned as described in Table S3.

B. Long-range XY model

The ground state of the antiferromagnetic XY Hamiltonian is twofold degenerate when there are an odd number of lattice sites. To break this degeneracy we added a small perturbation to the Hamiltonian, so that we actually considered the ground state of

$$H = \frac{1}{2}\sum_{i<j}^{L} \frac{1}{|i-j|^{\alpha}} \left(\sigma_i^x \sigma_j^x + \sigma_i^y \sigma_j^y\right) + \frac{\delta B}{2}\sum_{k}^{L} \sigma_k^x, \quad \text{(S1)}$$

with $\delta B = 10^{-6}$. We calculated this ground state using parallel two-site DMRG [S22] with 16 processes. As in Ref. [S23], we used a maximum MPS bond dimension of $\chi = 128$ [S24], and a Hamiltonian MPO with maximum absolute error $< 10^{-8}$.

p	Number of sites owned by first process	Number of sites owned by central processes	Number of sites owned by last process
8	15	12	14
16	9	6	8
24	7	4	6
32	6	3	5

TABLE S4. Partitioning of the 101-site MPS in the long-range XY model simulations.

p	Number of sites owned by first process	Number of sites owned by central processes	Number of sites owned by last process
32	11	6	10

TABLE S5. Partitioning of the 201-site MPS in the long-range XXX model simulation.

For the time evolution we used $\chi_{\max} = 256$ [S24] as in Ref. [S23]. No truncation error tolerance was set. The 101-site MPS was partitioned as described in Table S4.

C. Long-range XXX model

To calculate the thermodynamic limit results, we numerically integrated the exact expression for $C_\infty(x,t)$ (given in the main text) using the `integral2()` function in MATLAB with the `iterated` option [S25].

For the 201-site p2TDVP calculation we approximated the $1/r^2$ power law by a sum of 12 exponentials, giving an MPO bond dimension of 38 with a maximum absolute error of 1.5×10^{-7} and a maximum relative error of 5.3×10^{-3}. As shown in Fig. S1, it is possible to use the Levenberg-Marquardt nonlinear least-squares algorithm to approximate the power law more efficiently using just 9 exponentials, albeit with a slightly larger error. This would give an MPO bond dimension of 29, and hence a potential speedup of ≈ 1.3.

The ground state of the model $|\psi_0\rangle$ was found using two-site DMRG with a U(1) symmetric MPS of maximum bond dimension $\chi = 512$. The energy per site converged to

$$E_0/N = -0.410611165931, \tag{S2}$$

with a total discarded weight of 2.1×10^{-9}. In Fig. S2 we show the magnitude of the ground state correlation function $\langle \sigma_r^z \sigma_k^z \rangle$ for $k = 101$. This appears to follow a power law with exponent equal to 1 (dashed line), except towards the edges where the correlations decay exponentially due to the open boundaries.

The initial state for our p2TDVP calculation was $|\psi\rangle = \sigma_{101}^z |\psi_0\rangle$. We time evolved $|\psi\rangle$ on 32 processes, using a truncation error tolerance of $w_{\max} = 10^{-16}$, and a maximum bond dimension of $\chi_{\max} = 1024$. The MPS was partitioned as shown in Table S5. As in Ref. [S26],

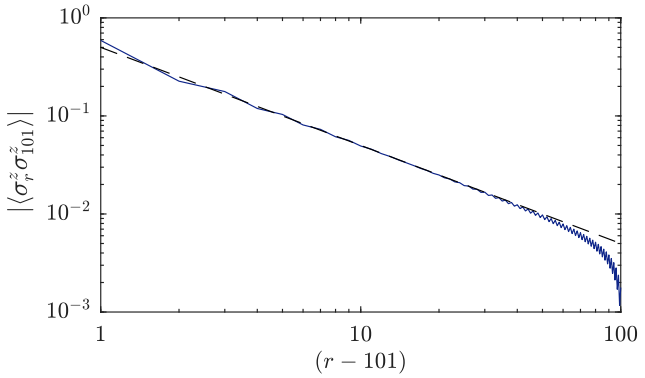

FIG. S2. Ground state correlations in the 201-site long-range ($\alpha = 2$) XXX model. The dashed line is proportional to $(r - 101)^{-1}$.

we used a timestep of $\delta t = 0.025$ and computed the dynamical spin-spin correlation function,

$$C(r-k,t) = e^{-iE_0 t} \langle \psi_0 \,|\, \sigma_r^z \,|\, \psi(t) \rangle , \tag{S3}$$

every eight timesteps.

Although a full scaling analysis was impractical for this simulation, we used different numbers of processes to time evolve the final state at the end of the simulation for an additional timestep. With one process, this took 71.6 minutes; with 32 processes, it took 7.8 minutes – a speedup of 9.1 (in comparison, the 65-site simulation gave a speedup of 11.0 on 32 processes). This suggests that our partitioning of the MPS was not optimal. By looking at the final bond dimensions we were able to devise a better partitioning scheme, which gave a speedup of 15.4 with 32 processes. The scaling results are shown in Fig. S3, with the corresponding partitions described in Table S6. We see close to ideal scaling up to 4 processes, with reasonable scaling up to 16 processes. A speedup of 15.1 was achieved with 24 processes, after which it tails off due to load imbalance. The reason for this is that the dynamics of the system are fairly localized, so the

Process ID(s)	0	1				
No. of sites owned	101	100				
Process ID(s)	0	1–2	3			
No. of sites owned	85	16	84			
Process ID(s)	0	1–6	7			
No. of sites owned	77	8	76			
Process ID(s)	0	1	2–13	14	15	
No. of sites owned	65	12	4	12	64	
Process ID(s)	0	1	2–21	22	23	
No. of sites owned	60	11	3	11	59	
Process ID(s)	0	1	2–3	4–27	28–29	30–31
No. of sites owned	33	32	6	2	6	32

TABLE S6. (top to botttom) MPS partitions for 2, 4, 8, 16, 24, and 32 processes, corresponding to the single timestep scaling shown in Fig. S3.

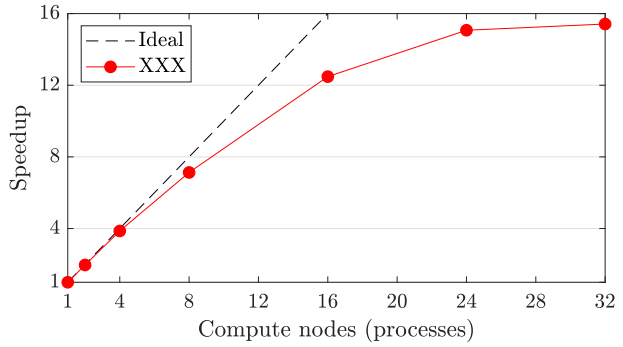

FIG. S3. Scaling plot for the extra timestep at the end of the 201-site XXX model p2TDVP simulation.

relatively large bond dimension is only saturated by the central tensors.

This example highlights the fact that the choice of partitioning scheme and number of processes is nontrivial for simulations in which the MPS bond dimensions χ_j grow inhomogeneously. Unless necessitated by memory requirements, using "too many" compute nodes is a waste of resources. On the other hand, sub-optimal partitioning is a performance issue. It should be possible to address this using a dynamic load balancer, as has previously been done for the parallel time-evolving block decimation algorithm [S27].

[S1] S. Al-Assam, S. R. Clark, D. Jaksch, and TNT Development team, Tensor Network Theory Library beta version 1.2.1, https://gitlab.physics.ox.ac.uk/tntlibrary (2016).

[S2] C. Goodyer, TNT Library : Tensor Manipulation and Storage, http://www.hector.ac.uk/cse/distributedcse/reports/UniTNT/UniTNT/index.html (accessed 6 December 2019) (2013).

[S3] S. Al-Assam, S. R. Clark, and D. Jaksch, The tensor network theory library, J. Stat. Mech. **2017**, 093102 (2017).

[S4] J. Coulthard, *Engineering quantum states of fermionic many-body systems*, DPhil Thesis, University of Oxford, 2018, https://ora.ox.ac.uk/objects/uuid:c2ad8834-7202-45cf-8348-4f9b07e942d1.

[S5] The OpenMP Architecture Review Board, OpenMP, https://www.openmp.org/.

[S6] MPI Forum, Message Passing Interface (MPI), https://www.mpi-forum.org/.

[S7] B. Pirvu, V. Murg, J. I. Cirac, and F. Verstraete, Matrix product operator representations, New J. Phys. **12**, 025012 (2010).

[S8] The Mathworks, Inc., MATLAB version 9.3.0.713579 (R2017b), https://www.mathworks.com/products/matlab.html.

[S9] K. Levenberg, A method for the solution of certain nonlinear problems in least squares, Quart. Appl. Math. **2**, 164 (1944).

[S10] D. W. Marquardt, An Algorithm for Least-Squares Estimation of Nonlinear Parameters, J. Soc. Ind. Appl. Math. **11**, 431 (1963).

[S11] The MathWorks, Inc., Least-Squares (Model Fitting) Algorithms - MATLAB & Simulink, https://www.mathworks.com/help/optim/ug/least-squares-model-fitting-algorithms.html (accessed 29 October 2019).

[S12] G. M. Crosswhite, A. C. Doherty, and G. Vidal, Applying matrix product operators to model systems with long-range interactions, Phys. Rev. B **78**, 035116 (2008).

[S13] The University of Bath, Balena HPC cluster, https://www.bath.ac.uk/corporate-information/balena-hpc-cluster/ (accessed 29 October 2019).

[S14] opencollab, ARPACK-NG version 3.4.0, https://github.com/opencollab/arpack-ng.

[S15] Intel® Math Kernel Library (Intel® MKL) version 64/18.0.128, https://software.intel.com/mkl.

[S16] Intel® MPI Library version 64/18.0.128, https://software.intel.com/mpi-library.

[S17] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, LAPACK *Users' Guide*, 3rd ed. (SIAM, Philadelphia, PA, 1987), https://www.netlib.org/lapack/lug/.

[S18] J. J. M. Cuppen, A divide and conquer method for the symmetric tridiagonal eigenproblem, Numer. Math. **36**, 177 (1980).

[S19] M. Gu and S. C. Eisenstat, A Stable and Efficient Algorithm for the Rank-One Modification of the Symmetric Eigenproblem, SIAM J. Matrix Anal. Appl. **15**, 1266 (1994).

[S20] J. D. Rutter, *A serial implementation of Cuppen's divide and conquer algorithm for the symmetric eigenvalue problem*, Tech. Rep. UCB/CSD-94-799 EECS Department, University of California, Berkeley, 1994, http://www2.eecs.berkeley.edu/Pubs/TechRpts/1994/6315.html.

[S21] R. Lehoucq, K. Maschhoff, D. Sorensen, and C. Yang, ARPACK - Arnoldi Package, https://www.caam.rice.edu/software/ARPACK/.

[S22] E. M. Stoudenmire and S. R. White, Real-space parallel density matrix renormalization group, Phys. Rev. B **87**, 155137 (2013).

[S23] J. Haegeman, C. Lubich, I. Oseledets, B. Vandereycken, and F. Verstraete, Unifying time evolution and optimization with matrix product states, Phys. Rev. B **94**, 165116 (2016).

[S24] J. Haegeman, (private communication).

[S25] The MathWorks, Inc., Numerically evaluate double integral - MATLAB integral2, https://www.mathworks.com/help/matlab/ref/integral2.html (accessed 29 October 2019).

[S26] M. P. Zaletel, R. S. K. Mong, C. Karrasch, J. E. Moore, and F. Pollmann, Time-evolving a matrix product state with long-ranged interactions, Phys. Rev. B **91**, 165112 (2015).

[S27] M. Urbanek and P. Soldán, Parallel implementation of the time-evolving block decimation algorithm for the Bose–Hubbard model, Comput. Phys. Commun. **199**, 170 (2016).