Parallel density matrix propagation in spin dynamics simulations

Luke J. Edwards and Ilya Kuprov[a)]
Oxford e-Research Centre, University of Oxford, 7 Keble Road, Oxford OX1 3QG, United Kingdom

(Received 21 September 2011; accepted 6 January 2012; published online 31 January 2012)

Several methods for density matrix propagation in parallel computing environments are proposed and evaluated. It is demonstrated that the large communication overhead associated with each propagation step (two-sided multiplication of the density matrix by an exponential propagator and its conjugate) may be avoided and the simulation recast in a form that requires virtually no inter-thread communication. Good scaling is demonstrated on a 128-core (16 nodes, 8 cores each) cluster. © *2012 American Institute of Physics*. [doi:10.1063/1.3679656]

I. INTRODUCTION

In common with much of quantum theory, the theoretical formalism of spin dynamics is not easily adaptable to parallel computing architectures – matrix operations in both the frequency domain (Hamiltonian diagonalization) and the time domain (density matrix propagation) have large communication overheads, resulting in poor scaling with respect to the number of CPUs on both clusters and shared-memory systems. Situations where a simulation contains multiple independent blocks (e.g., different orientations in a powder average or different slices in a multi-dimensional NMR experiment) are straightforward and have already been treated in the literature[1,2] as well as implemented in mainstream simulation software packages.[3–5] However, a parallel simulation of a single large spin system is considerably harder and requires alterations to the formalism as well as the simulation algorithms to expose the parallel stages and minimize the thread communication overhead.

Spin dynamics simulations are usually performed using the density operator formalism.[6] This is preferred to a method using the Schrödinger equation because systems encountered in NMR and ESR spectroscopy are ensembles of molecules in which some degrees of freedom must be averaged over – density operator description provides a natural way of performing such averaging. For small spin systems, the simulations can be carried out either in the frequency domain (by diagonalizing the Hamiltonian), or in the time domain by propagating the system forward from the initial condition and projecting out the observables. Simulations are less straightforward for large spin systems, where close attention must be paid to CPU and memory utilization as well as to achieving maximum possible efficiency from the computational point of view.

Frequency domain simulations of large spin systems face insurmountable difficulties regardless of computer architecture – while the Hamiltonian itself is often sparse, its eigenvectors are nearly always dense, meaning that a terabyte of storage (a generous allowance) would only accommodate 20 spins and any improvement would be logarithmic. Full diagonalization of Hamiltonians with dimension in excess of 10^5 (i.e., with more than about 17 spins) is not at present realistic,[7] although some progress has recently been made with simulations that only seek a few specific eigenvalue-eigenvector pairs.[8]

Time domain spin dynamics simulations, which involve propagating the density operator $\hat{\rho}(t)$ forward in time under the Liouville-von Neumann equation[6]

$$\frac{d}{dt}\hat{\rho}(t) = -i[\hat{H}, \hat{\rho}(t)]$$

$$\hat{\rho}(t + \Delta t) = e^{-i\hat{H}\Delta t}\hat{\rho}(t)e^{i\hat{H}\Delta t} \qquad (1)$$

are less expensive (particularly if restricted state spaces can be used[9]) because the small step propagator $e^{-i\hat{H}\Delta t}$ inherits the sparsity of the Hamiltonian.[7] The parallelization problem in the time domain is therefore reduced to finding a parallel algorithm for density matrix propagation. This issue has been investigated for optimal control problems,[10] symplectic propagation of large regular spin lattices,[11] and observable dynamics in spin systems.[12] The primary obstacle is that propagation steps under the Liouville-von Neumann equation in Hilbert space involve double-sided matrix multiplication, and so every element of $\hat{\rho}(t + \Delta t)$ depends on every element of $\hat{\rho}(t)$. This means that, even if elements of $\hat{\rho}(t + \Delta t)$ are evaluated in parallel, the entire $\hat{\rho}(t)$ matrix has to be communicated to every node at each time step. This is an unacceptably large amount of communication, particularly on clusters, where the network bandwidth presents a bottleneck. Alternatively, the propagation step may be split using matrix factorizations (diagonalization[12] and singular value decomposition (SVD)[11] have been suggested in the literature), but the factorizations are themselves expensive and difficult to parallelize. Another alternative, using matrix-vector multiplications in Liouville space[6]

$$\frac{d}{dt}\hat{\rho}(t) = -i\hat{\hat{H}}\hat{\rho}(t) \Rightarrow \hat{\rho}(t + \Delta t) = e^{-i\hat{\hat{H}}\Delta t}\hat{\rho}(t), \qquad (2)$$

in which $\hat{\hat{H}}$ is the Hamiltonian commutation superoperator, suffers in practice from astronomical matrix dimensions (equal to the square of Hilbert space dimension), and load-balancing issues[13] caused by the unpredictable distribution of non-zeros in the superoperator matrices, as well as large communication overhead – the state vector is redistributed at each time step.

a)Electronic mail: ilya.kuprov@oerc.ox.ac.uk. Fax: +44 1865 610612.

In the present paper we propose and evaluate several methods for Hilbert space density matrix propagation in parallel computing environments. It is demonstrated that the large communication overhead associated with each propagation step may be avoided and the simulation recast, without any approximations, in a form that requires no inter-thread communication beyond distributing the initial condition and retrieving the final results from the worker nodes. Good scaling is demonstrated for NMR simulations on a 128-core (16 nodes, 8 cores each) cluster.

II. COMPUTATION, STORAGE, AND COMMUNICATION OVERHEADS

The physical nature of spin interactions requires the Hamiltonian of an n-spin system to have at most $(n^2 + 3n)/2$ interactions: $(n^2 - n)/2$ bilinear couplings between different spins, n quadratic couplings of a spin to itself, and n couplings to the external magnetic field. In the most general case, each bilinear coupling involves six linearly independent spin operators (corresponding to the isotropic part and five irreducible components of the anisotropic part), each quadratic coupling involves five operators (quadrupolar interaction and zero-field splitting are traceless, but not necessarily axial) and each Zeeman coupling involves three spin operators ($\hat{L}_X$, $\hat{L}_Y$, $\hat{L}_Z$). The upper bound for the number of linearly independent operators in the physically meaningful Hamiltonian of an n-spin system is therefore $3n^2 + 5n$. The fact that this number grows polynomially with the number of spins has profound algebraic and physical consequences elsewhere,[9,14] but in our current context it may be used to get an upper bound on the density of the matrices (the ratio of the number of non-zeros to the total number of elements) involved in numerical spin dynamics simulations. The resulting bounds may then be used to obtain asymptotic estimates on the storage, communication, and computation requirements.

It may be seen by direct inspection that matrix representations of Cartesian spin operators ($\hat{L}_X$, $\hat{L}_Y$, $\hat{L}_Z$) and their binary direct products ($\hat{L}_X\hat{S}_X$, $\hat{L}_X\hat{S}_Y$, etc.) in the Pauli basis have at most one non-zero element per row, with the number of rows for an n-spin system being equal to $\prod_{k=1}^{n}(2S_k + 1)$, where S_k is the total spin quantum number of the k-th spin. Of their possible combinations, the isotropic coupling operator $\hat{L}_X\hat{S}_X + \hat{L}_Y\hat{S}_Y + \hat{L}_Z\hat{S}_Z$ has at most two, and each of the five second-rank irreducible spherical tensor operators have at most one (for $\hat{T}_{2,\pm 2}$ and $\hat{T}_{2,\pm 1}$) or two (for $\hat{T}_{2,0}$) non-zeros per row. Taken together this yields, an upper bound of $4n^2 + 5n$ on the number of non-zeros per row (or column) of a spin Hamiltonian and the following upper bounds on the total number of non-zeros N_{NZ} and matrix density $d_{\hat{H}}$

$$N_{NZ} \leq (4n^2 + 5n)\prod_{k=1}^{n}(2S_k + 1),$$

$$d_{\hat{H}} \leq \frac{N_{NZ}}{\left[\prod_{k=1}^{n}(2S_k + 1)\right]^2} = \frac{4n^2 + 5n}{\prod_{k=1}^{n}(2S_k + 1)}. \quad (3)$$

This bound on $d_{\hat{H}}$ formally validates the common knowledge that spin operators are very sparse,[15] but adds the important statement that the sparsity of the Hamiltonian matrix increases *exponentially* with the number of spins. In a 20-spin system with $S_k = 1/2$ and everything coupled to everything, $d_{\hat{H}} \leq 0.00143$ – any possible Hamiltonian would be mostly zeros.

Importantly, the sparsity analysis presented above does not apply to the density matrix. In a typical solid state NMR spin system, for instance, $\hat{\rho}$ becomes dense in the first few milliseconds of time evolution. The storage, computation, and communication costs associated with the density matrix are therefore the same as for a general dense matrix. However, the cost of multiplication of $\hat{\rho}$ by the Hamiltonian scales as $O(d_{\hat{H}}N^3)$ with representation dimension N, which is considerably better than the $O(N^3)$ cost of the dense matrix multiplication.

It is also easy to demonstrate that, for a well-chosen time step, the cost of multiplication by an exponential propagator has the same asymptotic scaling as the cost of multiplication by a Hamiltonian. Indeed, the multiplication by the exponential propagator may be expressed as

$$e^{-i\hat{H}\Delta t}\hat{\rho} = \sum_{k=0}^{\infty}\frac{(-i\hat{H}\Delta t)^k}{k}\hat{\rho}$$

$$= \sum_{k=0}^{\infty}\frac{(-i\Delta t)^k}{k!}(\hat{H}(\hat{H}\ldots(\hat{H}\hat{\rho}))). \quad (4)$$

If the time step is chosen as $\Delta t = \|\hat{H}\|_\infty^{-1}$, this series converges to machine precision in about 15 iterations meaning that only a fixed number of multiplications by the Hamiltonian are in practice required. This is often useful for simulation of systems with time-dependent Hamiltonians because matrix exponentiation may be avoided entirely. The cost of the Hamiltonian infinity-norm calculation is $O(N_{NZ})$ additions, which is negligible. The cost of forming and manipulating the Hamiltonian is similarly small compared to the cost of the same operations on the density matrix so long as the sparsity of the Hamiltonian is preserved by those operations.

From the reasoning above, we can conclude that the primary computation and communication cost of a time-domain spin dynamics simulation is associated with time propagation of the density matrix – the housekeeping costs associated with operators and propagators are negligible. In practice this means that Hamiltonians and propagators may be replicated without loss of efficiency on distributed computing systems with up to about $1/d_{\hat{H}}$ worker nodes. The density matrix, however, clearly requires distributed storage and distributed operations; any inter-thread communication involving the density matrix must be kept to a minimum.

III. EXPECTATION VALUE DYNAMICS FOR A SPECIFIC OBSERVABLE

Parallel propagation was recently explored by Skinner and Glaser,[12] who suggested that the density matrix could be expanded as

$$\hat{\rho} = \sum_k p_k|v_k\rangle\langle v_k|, \quad (5)$$

where $|v_k\rangle$ is the k-th eigenvector of the density matrix and p_k is the corresponding eigenvalue. With this decomposition the calculation of expectation values is easy to parallelize with a relatively modest communication overhead:

$$\begin{aligned}\langle\hat{A}(t)\rangle &= \mathrm{Tr}[\hat{A}e^{-i\hat{H}t}\hat{\rho}(0)e^{i\hat{H}t}]\\ &= \sum_k p_k \mathrm{Tr}[\hat{A}e^{-i\hat{H}t}|v_k(0)\rangle\langle v_k(0)|e^{i\hat{H}t}]\\ &= \sum_k p_k[\langle v_k(0)|e^{i\hat{H}t}]\hat{A}[e^{-i\hat{H}t}|v_k(0)\rangle]\\ &= \sum_k p_k\langle v_k(t)|\hat{A}|v_k(t)\rangle, \end{aligned} \tag{6}$$

where the individual trajectories $e^{-i\hat{H}t}|v_k(0)\rangle$ corresponding to different values of the index k are independent for the entire duration of the simulation and may be computed on different nodes. The elementary spin operators are always very sparse and the cost of supplying each node with $\hat{H}$ and $\hat{A}$ matrices is therefore acceptable. At each propagation step only the expectation values of $\hat{A}$ need to be gathered and summed on the head node, with weights specified by $\{p_k\}$. Matrix exponentiation is not required because the elementary propagation step,

$$|v_k(t+\Delta t)\rangle = e^{-i\hat{H}\Delta t}|v_k(t)\rangle, \tag{7}$$

may be computed directly from $\hat{H}$ and $|v_k(t)\rangle$ using Krylov techniques that only use sparse matrix-vector multiplications.[16] The computational scaling of Eq. (6) is therefore expected to be excellent. Situations where the Hamiltonian is time-dependent differ only in the number of operators to be distributed to the nodes at the problem set-up stage – time-dependent Hamiltonians in magnetic resonance always have the following form:

$$\hat{H}(t) = \hat{H}_0 + \sum_{k=1}^{N} a_k(t)\hat{H}_k, \tag{8}$$

where $\hat{H}_0$ is the static component of the Hamiltonian and $a_k(t)$ are scalar functions modulating the dynamic components $\hat{H}_k$. The number of time-dependent terms is always small: low single digits for radiofrequency irradiation and at most 25 in the presence of the rotational modulation, such as magic angle spinning.[17] The basis operators $\hat{H}_k$ and their modulation coefficients can therefore be distributed to the nodes at the beginning of the calculation and their sparsity ensures that the communication requirements stay modest.

The expansion postulated in Eq. (5) is only computationally affordable if the initial density matrix is diagonal in the current basis. If $\hat{\rho}$ is not diagonal, two kinds of problems can potentially arise: some density matrices (for example $\hat{\rho} = \hat{L}_+$) cannot be diagonalized because they are singular or near-singular, and some cannot be diagonalized because they are too large. The first problem has a simple solution – a singular value decomposition (SVD),

$$\hat{\rho} = \sum_k \sigma_k |u_k\rangle\langle v_k|, \tag{9}$$

exists for any matrix and may be used instead of diagonalization in Eq. (5) with the corresponding modifications applied to Eq. (6):

$$\begin{aligned}\langle\hat{A}(t)\rangle &= \mathrm{Tr}[\hat{A}e^{-i\hat{H}t}\hat{\rho}(0)e^{i\hat{H}t}]\\ &= \sum_k \sigma_k \mathrm{Tr}[\hat{A}e^{-i\hat{H}t}|u_k(0)\rangle\langle v_k(0)|e^{i\hat{H}t}]\\ &= \sum_k \sigma_k[\langle v_k(0)|e^{i\hat{H}t}]\hat{A}[e^{-i\hat{H}t}|u_k(0)\rangle]\\ &= \sum_k \sigma_k\langle v_k(t)|\hat{A}|u_k(t)\rangle. \end{aligned} \tag{10}$$

Because the left-side trajectory, $\langle u_k(0)|e^{i\hat{H}t}$, is no longer the conjugate of the right-side trajectory, $e^{-i\hat{H}t}|v_k(0)\rangle$, this doubles the amount of work compared to the formulation given by Skinner and Glaser[12] but avoids the problem of singular density matrices. In practical simulations, the overall amount of work ends up being smaller because non-Hermitian density matrices are used to replace phase cycles, which are typically 8–16 simulations long. It is also often the case (particularly in weakly coupled spin systems) that the density matrix has many small singular values, which may be ignored altogether, thus reducing the amount of work in Eq. (10).

While Eq. (9) is in some ways an improvement on Eq. (5), the matrix dimension problem still remains unsolved. For a sparse density matrix requiring $O(N)$ doubles for storage, both diagonalization and SVD need $O(N^3)$ doubles to store the dense eigenvector arrays and $O(N^3)$ multiplications to obtain them.[18] Both diagonalization and SVD are also notorious for their poor parallelization, large communication overheads, and numerical accuracy issues with the degenerate eigenvalue sets that are often encountered in magnetic resonance simulations. A sparse Cholesky factorization[18] for a suitably preconditioned density matrix,

$$\begin{aligned}\hat{\rho} + 1\|\hat{\rho}\| = LL^\dagger \Rightarrow \hat{\rho}(t) &= e^{-i\hat{H}t}LL^\dagger e^{i\hat{H}t} - 1\|\hat{\rho}\|\\ &= (e^{-i\hat{H}t}L)(e^{-i\hat{H}t}L)^\dagger - 1\|\hat{\rho}\|, \end{aligned} \tag{11}$$

is similarly inefficient because it requires $O(N^3)$ multiplications and increases matrix density (even when one reorders for sparsity[19]), thus making communications more expensive.

In view of these difficulties, we propose an alternative decomposition which does not require any kind of matrix factorization and preserves the neat parallel structure of Eq. (10):

$$\hat{\rho} = \sum_k |\rho_k\rangle\langle\delta_k|, \tag{12}$$

where $|\rho_k\rangle$ is the k-th column of the density matrix and $|\delta_k\rangle$ is a vector with 1 in position k and zeros elsewhere. This formulation does not require any operations on the initial density matrix beyond sending the columns to their allocated nodes at the start of the calculation. A similar argument to Eq. (10) then gives the following expression for the observable:

$$\begin{aligned}\langle\hat{A}(t)\rangle &= \mathrm{Tr}[\hat{A}e^{-i\hat{H}t}\hat{\rho}(0)e^{i\hat{H}t}]\\ &= \sum_k \mathrm{Tr}[\hat{A}e^{-i\hat{H}t}|\rho_k(0)\rangle\langle\delta_k(0)|e^{i\hat{H}t}]\end{aligned}$$

$$= \sum_k [\langle \delta_k(0)|e^{i\hat{H}t}]\hat{A}[e^{-i\hat{H}t}|\rho_k(0)\rangle]$$

$$= \sum_k \langle \delta_k(t)|\hat{A}|\rho_k(t)\rangle. \quad (13)$$

The computational complexity of this formulation is identical to Eq. (10) – two sets of vectors must be propagated at the worker nodes with no inter-thread communication apart from the negligible cost of sending the resulting observable $\langle \delta_k(t)|\hat{A}|\rho_k(t)\rangle$ back to the head node.

Algorithm A: Expectation value dynamics for a specific observable

Step 1: Distribute the columns of the initial density matrix $|\rho_k\rangle$, the columns of the unit matrix $|\delta_k\rangle$, the Hamiltonian $\hat{H}$, and the observable operator $\hat{A}$ to the worker nodes. If the step propagator $\hat{P} = \exp(-i\hat{H}\Delta t)$ is available at the start of the calculation it may be supplied to the nodes instead of the Hamiltonian.

Step 2: On each worker node k pre-allocate a dense array of zeros $A^{(k)}$ for the storage of the local contribution to the observable dynamics trace.

Step 3: On each worker node k propagate the local vectors and co-vectors through the prescribed number of time points and record their contribution to the observable dynamics:

$$A^{(k)}(t_n) = \langle \delta_k(t_n)|\hat{A}|\rho_k(t_n)\rangle,$$

$$|\rho_k(t_{n+1})\rangle = \exp(-i\hat{H}\Delta t)|\rho_k(t_n)\rangle,$$

$$|\delta_k(t_{n+1})\rangle = \exp(-i\hat{H}\Delta t)|\delta_k(t_n)\rangle,$$

using matrix-vector multiplications if the exponential propagator had been supplied and Krylov propagation[16] if $\exp(-i\hat{H}\Delta t)$ is not directly available or the Hamiltonian is time-dependent.

Step 4: Collect the observable traces $A^{(k)}$ from every worker node and add them up element-by-element to obtain the final observable dynamics trace.

The resulting algorithm inherits the primary advantage of the method proposed by Skinner and Glaser[12] – no inter-thread communication at the propagation stage – and also removes the need to perform any kind of matrix factorization at the problem set-up stage. The head node only needs to compute the sum of the observable traces returned by the worker nodes.

IV. FINAL STATE CALCULATION

The situation where the final density matrix is required after a given evolution period is similar to the calculation of observable evolution discussed in Sec. III:

$$\hat{\rho}(t) = e^{-i\hat{H}t}\hat{\rho}(0)e^{i\hat{H}t} = \sum_k [e^{-i\hat{H}t}|\rho_k(0)\rangle][\langle \delta_k(0)|e^{i\hat{H}t}]$$

$$= \sum_k |\rho_k(t)\rangle\langle \delta_k(t)|. \quad (14)$$

The individual vectors $|\rho_k\rangle$ and $|\delta_k\rangle$ may be distributed to different worker nodes for processing and sent back to the head node at the end of the propagation – this is illustrated schematically in Figure 1.

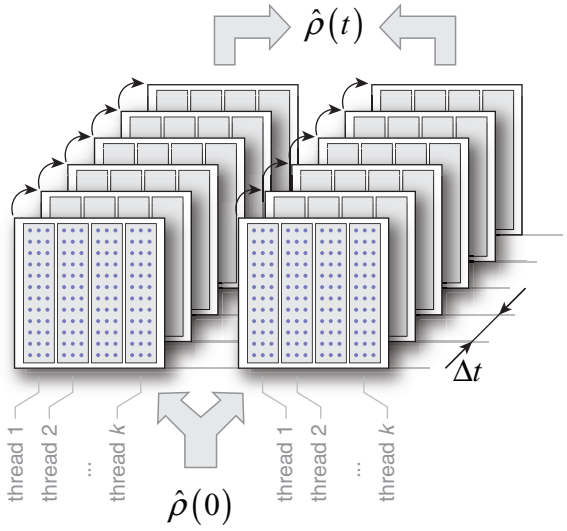

FIG. 1. Schematic illustration of the split propagation method (Algorithm B1) for the evaluation of the final state. The density matrix is factorized according to Eqs. (5), (9) or (12); the two factors are then sliced and propagated independently. The matrix is reassembled at the head node at the end of the calculation. The best performance is in practice achieved with Eq. (12), which avoids computationally expensive factorizations.

Algorithm B1: Final state calculation

Step 1: Distribute the columns of the initial density matrix $|\rho_k\rangle$, the columns of the unit matrix $|\delta_k\rangle$, and the Hamiltonian $\hat{H}$ to the worker nodes. If the step propagator $\hat{P} = \exp(-i\hat{H}\Delta t)$ is available at the start of the calculation, it may be supplied to the worker nodes instead of the Hamiltonian.

Step 2: On each worker node k propagate the local vectors and co-vectors through the prescribed number of time points

$$|\rho_k(t_{n+1})\rangle = \exp(-i\hat{H}\Delta t)|\rho_k(t_n)\rangle,$$

$$|\delta_k(t_{n+1})\rangle = \exp(-i\hat{H}\Delta t)|\delta_k(t_n)\rangle,$$

using matrix-vector multiplications if the exponential propagator had been supplied and Krylov propagation[16] if $\exp(-i\hat{H}\Delta t)$ is not directly available or the Hamiltonian is time-dependent.

Step 3: Collect the final vectors $|\rho_k\rangle$ and $|\delta_k\rangle$ from every worker node and re-assemble the density matrix on the head node

$$\hat{\rho} = \sum_k |\rho_k\rangle\langle \delta_k|.$$

There is no inter-thread communication at the propagation stage, but some head node processing is required at Stage 3 – the cost of re-assembling the density matrix is $O(N^2)$ multiplications. Similarly to the expectation value algorithm above, this approach doubles the amount of work compared to the diagonalization-based procedure,[12] but avoids numerical matrix factorizations and is therefore expected to show better scaling. It will also work for singular density matrices.

For systems where the available communication bandwidth is large (e.g., shared-memory supercomputers), we

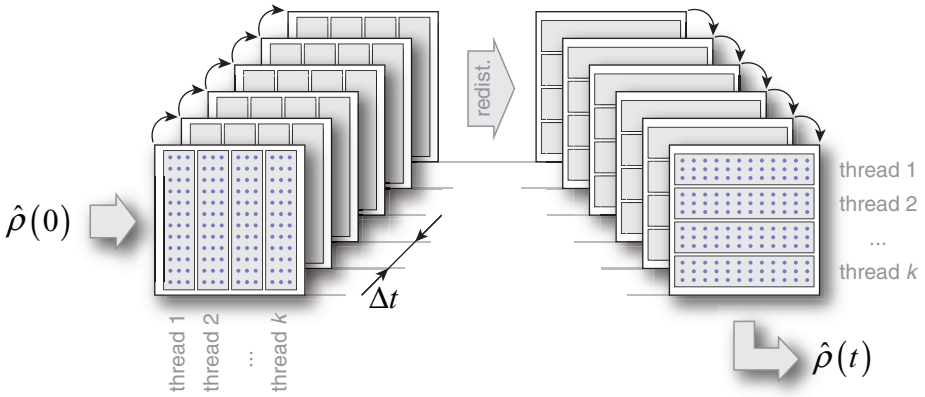

FIG. 2. Schematic illustration of the double transpose method (Algorithm B2) for the evaluation of the final state. The density matrix is sliced and propagated under the left side propagator, then transposed, redistributed and propagated under the right side propagator. Compared to the split propagation method, this algorithm requires less processing the head node, but has greater communication requirements.

would suggest a different algorithm which involves more communication but less head node processing. The basic idea is to re-order the multiplication operations in the repeated application of Eq. (1) during time evolution calculations:

$$\hat{\rho}(t_n) = \underbrace{\hat{P}\ldots\hat{P}}_{n}\,\hat{\rho}(0)\,\underbrace{\hat{P}^{\dagger}\ldots\hat{P}^{\dagger}}_{n} = \left[\underbrace{\hat{P}\ldots\hat{P}}_{n}\left[\underbrace{\hat{P}\ldots\hat{P}}_{n}\,\hat{\rho}(0)\right]^{\dagger}\right]^{\dagger},$$
$$\hat{P} = e^{-i\hat{H}\Delta t}. \tag{15}$$

Because the density matrix is always multiplied from the left, the individual columns of $\hat{\rho}(0)$ may be distributed to the worker nodes and there is no inter-thread communication during the evaluation of the inner square bracket in Eq. (15). The Hermitian conjugate, however, presents a significant communication hurdle – the density matrix that was distributed column-wise between the worker nodes, and is likely to no longer be sparse, has to be transposed and re-distributed. After that, an identical propagation stage is carried out and the rows of the final density matrix are sent back to the head node. This process is shown schematically in Figure 2.

Algorithm B2: Final state calculation

Step 1: Distribute the columns of the initial density matrix $|\rho_k\rangle$ and the Hamiltonian $\hat{H}$ to the worker nodes. If the step propagator $\hat{P} = \exp(-i\hat{H}\Delta t)$ is available at the start of the calculation, it may be supplied to the worker nodes instead of the Hamiltonian.

Step 2: On each node k propagate the local columns of the density matrix through the prescribed number of time points:

$$|\rho_k(t_{n+1})\rangle = \exp(-i\hat{H}\Delta t)|\rho_k(t_n)\rangle,$$

using matrix-vector multiplications if the exponential propagator had been supplied and Krylov propagation[16] if $\exp(-i\hat{H}\Delta t)$ is not directly available or the Hamiltonian is time-dependent.

Step 3: Re-distribute the density matrix row-wise between the worker nodes and conjugate-transpose each row on receipt to make a column.

Step 4: on each node k propagate the local columns of the density matrix through the prescribed number of time points:

$$|\rho_k(t_{n+1})\rangle = \exp(-i\hat{H}\Delta t)|\rho_k(t_n)\rangle,$$

using matrix-vector multiplications if the exponential propagator had been supplied and Krylov propagation[16] if $\exp(-i\hat{H}\Delta t)$ is not directly available or the Hamiltonian is time-dependent.

Step 5: collect the columns of the density matrix on the head node and conjugate-transpose the result to obtain the final density matrix.

Steps 3 and 5 in this algorithm are communication-intensive and have non-sequential memory access patterns at the matrix transpose stages, but there is no head-node arithmetic and less memory is used on the worker nodes. On shared-memory systems and clusters with large interconnect bandwidth this approach may therefore be preferable to Algorithm B1.

V. PERFORMANCE DATA AND SUMMARY

The performance of the three parallelization methods presented in this paper, measured as the number of time steps per wall clock second, is compared in Table I. All simulations were performed on a MATLAB cluster with 128 Intel Nehalem cores (16 nodes, 8 cores each) and 1.09 TB of RAM,

TABLE I. Scaling behaviour of the parallel propagation algorithms.

	Time steps per wall clock second		
Number of CPU cores	Algorithm A (Observable)	Algorithm B1 (Final state)	Algorithm B2 (Final state)
1	1.2	3.1	1.9
2	2.5	6.2	3.7
4	4.9	12.5	7.4
8	9.9	25.1	14.8
16	18.9	49.7	29.8
32	29.4	72.7	48.1
64	48.4	112.8	78.6
128	68.0	151.7	110.9

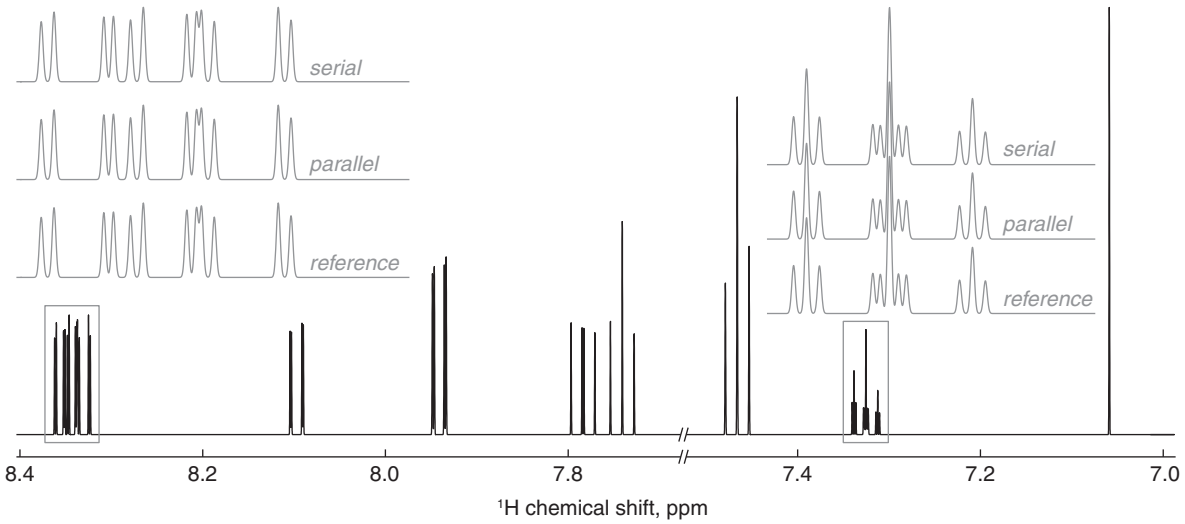

FIG. 3. An illustration of the fact that Eqs. (12)–(14) are exact transformations of the standard density operator propagation procedure in Eq. (1). The figure shows a pulse-acquire ^{1}H NMR spectrum of 3-phenylmethylene-1H, 3H-naphtho-[1, 8-c, d]-pyran-1-one[20] at 14.1 T. The results of a parallel calculation using Algorithm A, a single-threaded calculation using the same algorithm and the reference calculation using conventional propagation techniques are the same to machine precision.

provisioned from the Amazon EC2 cloud service, using version 1.0.959 of the *Spinach* library.[7] The figures in Table I refer to the simulation of the detection period in a pulse-acquire NMR experiment (exact calculation with a 4096 × 4096 density matrix) on the 12-spin system of 3-phenylmethylene-1H, 3H-naphtho-[1, 8-c, d]-pyran-1-one[20] at 14.1 T. In each case 35 378 propagation steps were taken with the time step chosen to be equal to the reciprocal infinity-norm of the rotating frame Hamiltonian – these conditions correspond to those of a typical liquid-state NMR simulation. The resulting spectrum is shown in Figure 3 – the spectra computed using the serial and the parallel propagation algorithms are equal to machine precision, and the simulations match the experimental spectrum[20] to within the accuracy of the measurement instrument.

As may be seen from Table I, the parallel propagation algorithms show satisfactory scaling all the way to 128 cores. The scaling is perfectly linear up to the node size (8 cores), but starts slowing down from 16 cores onwards due to the network communication overhead, with a factor of 2 increase in the node count yielding approximately a factor of 1.5 increase in the simulation speed. Because the node interconnect on Amazon EC2 cloud is 100BASE-TX Ethernet, these numbers should be viewed as pessimistic – typical HPC interconnects are much faster.

Of the two final state evaluation methods, the split propagation method (Algorithm B1) outperforms the double transpose method (Algorithm B2) by about 40%. This is likely a consequence of its smaller communication requirements and sequential memory access patterns – the split propagation method does not have the intermediate synchronization and matrix transpose steps. The calculation of observables (Algorithm A) is slower because it involves an extra matrix multiplication per step compared to the final state calculation, but shows similar performance scaling with the core count because its communication requirements are minimal. All attempts to use density matrix factorizations (diagonalization from Eq. (5), SVD from Eq. (9), and Cholesky factorization with and without reordering for sparsity) as a way of splitting the propagation problem have led to much slower algorithms. It is therefore clear that all forms of density matrix factorizations must be avoided and Eq. (12) is the best way forward.

Magnetic resonance simulations have another frequently occurring calculation type – trajectory calculation, where the full density matrix is stored at each time step. Despite putting considerable effort into the matter, we were unable to find a way of parallelizing trajectory calculation in a way that would exhibit acceptable scaling on either clusters or shared-memory systems – the number of density matrices to be communicated over the interconnect equals the number of time steps, meaning that communication becomes a major bottleneck. Another problem is memory: calculation of the full system trajectory for the example simulation described above would require a terabyte of storage. Significant compression may be achieved by using difference encoding (the difference from the previous density matrix may be stored, rather than the full density matrix, at each step), but the communication overhead of a trajectory calculation is still unacceptably large and further work is certainly required in this direction.

In summary, we found that it is possible to cast the density matrix propagation problem in a way that avoids inter-thread communication and thus enables large-scale parallel processing. To achieve satisfactory scaling on massively parallel systems, density matrix factorizations (diagonalization, SVD and Cholesky factorization were attempted in this work) must be avoided. This is possible (Eqs. (12) and (13) demonstrate the procedure) and enables efficient parallel calculation of final states and observable dynamics on supercomputers with up to 128 processor cores in our test calculations.

ACKNOWLEDGMENTS

The authors are grateful to Steffen Glaser, Stef Salvini, Tom Skinner, and Jeyan Thiyagalingam for helpful dis-

cussions. The project is funded by the Engineering and Physical Sciences Research Council (EPSRC) (Grant Nos. EP/F065205/1 and EP/H003789/1) and supported by the Oxford e-Research Centre.

[1]V. Y. Orekhov and V. A. Jaravine, Prog. Nucl. Magn. Reson. Spectrosc. **59**, 271 (2011).
[2]J. H. Kristensen and I. Farnan, J. Magn. Reson. **161**, 183 (2003).
[3]M. Bak, J. T. Rasmussen, and N. C. Nielsen, J. Magn. Reson. **147**, 296 (2000).
[4]Z. Tosner, R. Andersen, N. C. Nielsen, and T. Vosegaard, simpson (version 3.1), 2011.
[5]M. Veshtort and R. G. Griffin, J. Magn. Reson. **178**, 248 (2006).
[6]R. R. Ernst, G. Bodenhausen, and A. Wokaun, *Principles of Nuclear Magnetic Resonance in One and Two Dimensions* (Clarendon, Oxford, 1987).
[7]I. Kuprov, H. J. Hogben, M. Krzystyniak, G. T. P. Charnock, and P. J. Hore, J. Magn. Reson. **208**, 179 (2011). The source code for version 1.0.959 of the *Spinach* library, including the simulations discussed in this paper, is available at http://spindynamics.org.
[8]A. Weiße and H. Fehske, in *Computational Many-Particle Physics* (Springer, New York, 2008), Vol. 739, pp. 529–l544.
[9]A. Karabanov, I. Kuprov, G. T. P. Charnock, A. v. d. Drift, L. J. Edwards, and W. Köckenberger, J. Chem. Phys. **135**, 084106 (2011).
[10]T. Gradl, A. Sporl, T. Huckle, S. J. Glaser, and T. Schulte-Herbruggen, Lect. Notes Comput. Sci. **4128**, 751 (2006).
[11]C. H. Woo and P. W. Ma, Phys. Rev. E **79**, 046703 (2009).
[12]T. Skinner and S. Glaser, Phys. Rev. A **66**, 032112 (2002).
[13]A. Buluc and J. R. Gilbert, in *Proceedings of the 37th International Conference on Parallel Processing* (IEEE, New York, 2008), p. 503.
[14]I. Kuprov, N. Wagner-Rundell, and P. J. Hore, J. Magn. Reson. **189**, 241 (2007).
[15]R. S. Dumont, S. Jain, and A. Bain, J. Chem. Phys. **106**, 5928 (1997).
[16]M. Hochbruck and C. Lubich, SIAM J. Numer. Anal. **34**, 1911 (1997).
[17]I. Kuprov, J. Magn. Reson. **209**, 31 (2011).
[18]G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. (Johns Hopkins University Press, Baltimore, 1996).
[19]J. R. Gilbert, C. Moler, and R. Schreiber, SIAM J. Matrix Anal. Appl. **13**, 333 (1992).
[20]P. N. Penchev, N. M. Stoyanov, and M. N. Marinov, Spectrochim. Acta, Part A **78**, 559 (2011).