Journal of Magnetic Resonance 208 (2011) 179–194

Contents lists available at ScienceDirect

Journal of Magnetic Resonance

journal homepage: www.elsevier.com/locate/jmr

Spinach – A software library for simulation of spin dynamics in large spin systems

H.J. Hogben [a], M. Krzystyniak [b], G.T.P. Charnock [b], P.J. Hore [a], Ilya Kuprov [b,*]

[a] Physical and Theoretical Chemistry Laboratory, Chemistry Department, University of Oxford, South Parks Road, Oxford OX1 3QZ, UK
[b] Oxford e-Research Centre, University of Oxford, 7 Keble Road, Oxford OX1 3QG, UK

ARTICLE INFO

Article history:
Received 1 October 2010
Revised 7 November 2010
Available online 17 November 2010

Keywords:
NMR
ESR
Simulation
Spin dynamics
Spinach

ABSTRACT

We introduce a software library incorporating our recent research into efficient simulation algorithms for large spin systems. Liouville space simulations (including symmetry, relaxation and chemical kinetics) of most liquid-state NMR experiments on 40+ spin systems can now be performed without effort on a desktop workstation. Much progress has also been made with improving the efficiency of ESR, solid state NMR and Spin Chemistry simulations.

Spinach is available for download at http://spindynamics.org.

1. Introduction

It is a truth universally acknowledged that a brute force solution of the Liouville–von Neumann equation for a general system with over 20 spins is not practically feasible because the dimension of the matrices involved scales exponentially with the number of spins. The magnetic resonance community (unlike our colleagues in theoretical chemistry [1–4]) has thus far accepted this fact and evolved in the direction of software functionality rather than efficiency, taking full advantage of the emerging sparse storage [5,6], parallelization [7], propagation [8–10], symmetry factorization [11–16], model order reduction [17,18] and symbolic processing [19–21] techniques, but fundamentally still using algorithms that scale exponentially with the system size, meaning that the computing power (or patience) usually runs out at around 5–10 spins.

Still, the functionality available is impressive: existing packages can, in principle, simulate almost anything in NMR or ESR. They may be classified broadly into two categories – the analytical and semi-analytical tools [19–21], written mostly in *Mathematica*, and the numerical codes [5,7,22–25], written in *Fortran*, *C* or *Matlab*. Several packages are designed as function libraries [19,21,24,25], which makes them particularly useful for exploratory simulations. A powerful program with a convenient graphical user interface has recently appeared [23]. The most advanced codes, such as *SIMPSON* [22] and *SPINEVOLUTION* [5] include dedicated scripting languages that make them very flexible. That having been said, a software package has yet to emerge that would not grind to a halt with a 20-spin system – the computational scaling problem remains unsolved.

The present paper gives a summary of recent research into large-scale spin dynamics simulations – it briefly reviews a family of algorithms [26–29] and introduces a software package (*Spinach*) that aims to make spin dynamics simulations computationally efficient while preserving the existing software infrastructure. We can reasonably conclude that most liquid-state NMR experiments can now be simulated for 40+ spins with relative ease, and much progress has been made with solid state NMR [30,31], ESR and Spin Chemistry systems.

The primary objective of the *Spinach* library, which is described in Sections 3 and 4, is to generate accurate low-dimensional matrix representations for Liouville space operators and state vectors of large spin systems. The resulting matrices are often orders of magnitude smaller than those obtained with Kronecker products [28,29], and may either be used directly or imported into any of the existing software packages – these matrices are an alternative adjoint representation of the $\mathfrak{su}(2^N)$ algebra of spin [26], and therefore nothing changes in any of the usual simulation algorithms except for the matrix dimension.

2. Efficient spin dynamics simulation algorithms

In common with much of Computational Quantum Theory, the fundamental objective of an "efficient" spin dynamics simulation algorithm is to reduce the problem dimension to a minimum, to bring as many simulation stages as possible down from exponential into polynomial scaling and to provide estimates of the accuracy of any approximations involved. The methods reviewed

* Corresponding author. Fax: +44 1865 610612.
E-mail address: ilya.kuprov@oerc.ox.ac.uk (I. Kuprov).

1090-7807/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jmr.2010.11.008

below serve to accomplish these goals – some exactly, some approximately and some by recasting existing methods and theories into a more CPU- and memory-friendly form.

2.1. Basis indexing

By far the most versatile complete basis set for spin dynamics simulations is direct products of irreducible spherical tensors (ISTs) [26,32–35]. If the state space of each individual spin is indexed by enumerating all operators in its (necessarily finite) state space, *e.g.*

$$\hat{T}_{lm} \Longleftrightarrow \begin{pmatrix} l \\ m \end{pmatrix} \tag{1}$$

where $\hat{T}_{lm}$ is an IST, the matrix representations of basis operators for a multi-spin system may be avoided entirely because the direct product structure of any basis operator is completely determined by the index list:

$$\hat{T}_{l_1m_1} \otimes \hat{T}_{l_2m_2} \otimes \cdots \otimes \hat{T}_{l_nm_n} \Longleftrightarrow \begin{pmatrix} l_1 & l_2 & \dots & l_n \\ m_1 & m_2 & \dots & m_n \end{pmatrix} \tag{2}$$

which may be further transformed to require only a single integer per spin:

$$\hat{T}_{l_1m_1} \otimes \hat{T}_{l_2m_2} \otimes \cdots \otimes \hat{T}_{l_nm_n} \Longleftrightarrow \begin{pmatrix} l_1^2+l_1-m_1 & l_2^2+l_2-m_2 & \dots & l_n^2+l_n-m_n \end{pmatrix} \tag{3}$$

where the ISTs are now indexed by ascending rank l and within ranks by ascending projection number m – the position of the operator within this flattened list is given by $l^2 + l - m$. Such a representation has the benefit of requiring exponentially less memory than the explicit matrix representation – just N integers instead of at least $O(2^N)$ complex double-precision floating point numbers. The algebraic properties of ISTs are well researched [34,36–44], and many fundamental operations may be carried out directly in the indexed notation by remapping the indices. For multiplication of suitably normalized single-spin operators [32,35]:

$$\hat{T}_{l_1m_1}\hat{T}_{l_2m_2} = \sum_{L=|l_1-l_2|}^{l_1+l_2} C_{l_1m_1l_2m_2}^{LM}\hat{T}_{LM}, \quad M = m_1 + m_2 \tag{4}$$

where $C_{l_1m_1l_2m_2}^{LM}$ are related to Clebsch–Gordan coefficients. The structure coefficients c_{ijk} of the $\mathfrak{su}(2^N)$ Lie algebra of an N-spin system

$$\left(\bigotimes_{n=1}^{N}\hat{T}_{l_n^{(i)}m_n^{(i)}}\right)\left(\bigotimes_{n=1}^{N}\hat{T}_{l_n^{(j)}m_n^{(j)}}\right) = \sum_k c_{ijk}\left(\bigotimes_{n=1}^{N}\hat{T}_{l_n^{(k)}m_n^{(k)}}\right) \tag{5}$$

may also be computed without resort to matrix representations:

$$\begin{aligned} c_{ijk} &= \mathrm{Tr}\left[\left(\bigotimes_{n=1}^{N}\hat{T}_{l_n^{(i)}m_n^{(i)}}\right)\left(\bigotimes_{n=1}^{N}\hat{T}_{l_n^{(j)}m_n^{(j)}}\right)\left(\bigotimes_{n=1}^{N}\hat{T}_{l_n^{(k)}m_n^{(k)}}\right)^{\dagger}\right] \\ &= \mathrm{Tr}\left[\bigotimes_{n=1}^{N}\left(\hat{T}_{l_n^{(i)}m_n^{(i)}}\hat{T}_{l_n^{(j)}m_n^{(j)}}\hat{T}_{l_n^{(k)}m_n^{(k)}}^{\dagger}\right)\right] \\ &= \prod_{n=1}^{N}\mathrm{Tr}\left(\hat{T}_{l_n^{(i)}m_n^{(i)}}\hat{T}_{l_n^{(j)}m_n^{(j)}}\hat{T}_{l_n^{(k)}m_n^{(k)}}^{\dagger}\right) = \prod_{n=1}^{N} f_{ijk}^{(n)} \end{aligned} \tag{6}$$

in terms of the structure coefficients f_{ijk} of $\mathfrak{su}(2)$ algebras of individual spins, which are known and tabulated. In Eq. (6), the *ijk* indices run across the basis operators of $\mathfrak{su}(2^N)$, chosen to be direct products of single-spin ISTs, and the n index enumerates the spins in the system. Evaluation of a single structure coefficient using Eq. (6) is exponentially faster than the calculation using explicit matrix representations: N multiplications instead of at least $O(8^N)$.

2.2. Incomplete basis sets

The CPU time savings noted in the previous section do not circumvent the fact that there are exponentially many states in the complete basis set of $\mathfrak{su}(2^N)$ and the structure coefficient array c_{ijk} has exponential dimensions. This is a fundamental fact of nature for many-body systems, and any way around it must involve physical approximations.

The basis set need not be complete [26,28,29], but it must be closed with respect to temporal propagation – if a restricted space K is chosen for simulation, the density operator must not "leak" outside K:

$$\hat{\rho} \in K \Rightarrow e^{-i\hat{\hat{L}}t}\hat{\rho} \in K \quad \forall t \in [0, t_{\max}] \tag{7}$$

at least for the duration of the simulation, $t_{\max}$. Formally exact examples of K being considerably smaller than the full state space are well known and include systems with conservation laws [45], symmetry [11,14,16] and non-interacting subsystems [26]. Approximate restricted basis sets would exclude states that can in principle get populated, but are unlikely to be on the time scale of the simulation [28,29].

From the algebraic point of view, for a given effective Liouvillian superoperator $\hat{\hat{L}}$, the set G of all time propagators is a Lie group [46]:

$$\begin{aligned} &e^{-i\hat{\hat{L}}t_1}, e^{-i\hat{\hat{L}}t_2} \in G \;\Rightarrow\; e^{-i\hat{\hat{L}}t_1}e^{-i\hat{\hat{L}}t_2} = e^{-i\hat{\hat{L}}(t_1+t_2)} \in G \\ &\forall e^{-i\hat{\hat{L}}t_1} \in G \;\exists e^{i\hat{\hat{L}}t_1} \in G \quad \text{s.t. } e^{-i\hat{\hat{L}}t_1}e^{i\hat{\hat{L}}t_1} = \hat{\hat{E}} \end{aligned} \tag{8}$$

where t_1 and t_2 are arbitrary real numbers and $\hat{\hat{E}}$ is the identity superoperator. The system trajectory under the action of this group

$$G(\hat{\rho}_0) = \{e^{-i\hat{\hat{L}}t}\hat{\rho}_0, t \in [0, t_{\max}]\} \tag{9}$$

is a group orbit [28,29] of the initial state operator $\hat{\rho}_0$. The minimal set of operators spanning $G(\hat{\rho}_0)$ is the minimal basis required to describe the spin system evolution exactly. Reduced state space techniques should therefore be aimed at finding or approximating this minimal basis.

2.3. State space restriction using interaction topology

Our somewhat naive initial formulation of the restricted state space (RSS) approximation [29] was essentially using heuristics – it is often possible to decide *a priori* which states are unlikely to contribute to the spin system evolution. One could take advantage of three practical observations here:

1. All time-domain experiments in magnetic resonance have finite (often short) signal decay time – the system might not have enough time to evolve into certain states.
2. All spin interactions are at most binary, meaning that, even in densely coupled solids, the magnetization transport network in Liouville space is very sparse – access to remote areas of the state space could be slow.
3. The initial state and the detection state in all magnetic resonance experiments are simple collections of single-spin or two-spin (in exotic experiments such as PHIP [47] and CIDNP [48–50]) orders – the magnetization that strays too far from these narrow regions of the vast state space might never find its way back.

On these empirical grounds we could conjecture that a large enough spin system might not have the time or the opportunity to get into certain states. This is schematically illustrated in the

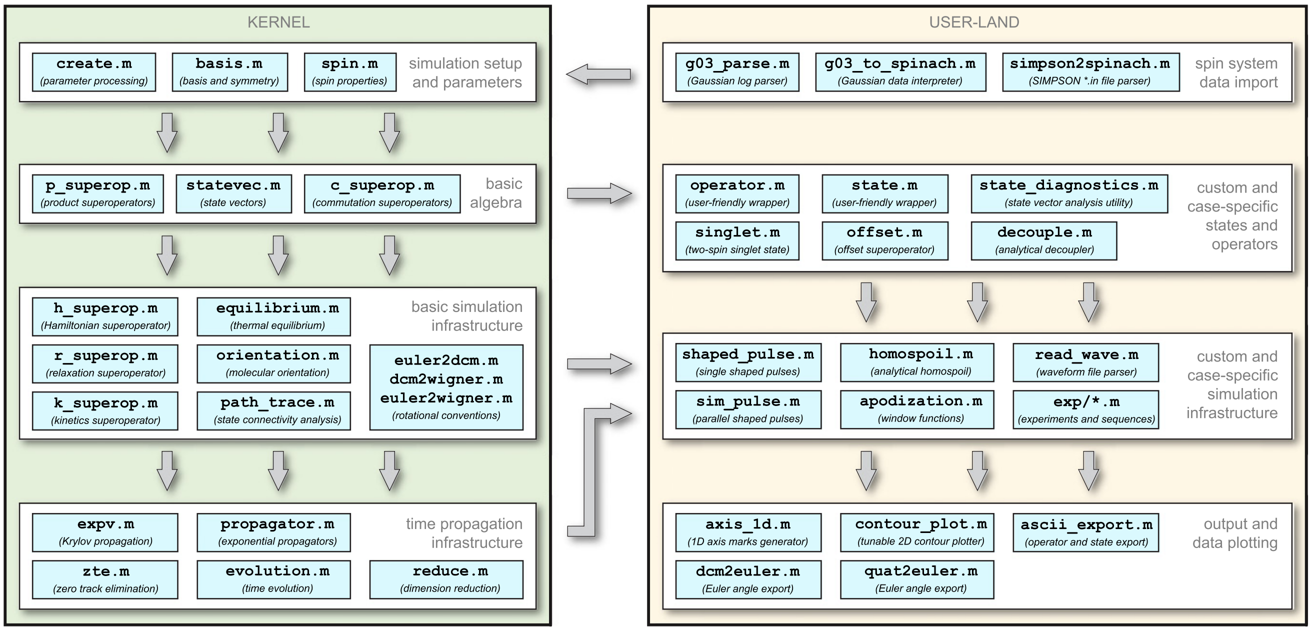

Fig. 1. *Spinach* code architecture. The kernel is a highly abstract Lie algebra and quantum theory engine, providing the core simulation infrastructure that does not depend on the exact nature of the spin system. The user-land contains wrappers, translators, building blocks and case-specific functions interfacing the kernel to the physical and methodological reality.

Graphical Abstract: clearly, the restricted state-space approximation should be particularly relevant for the very sparsely connected spin systems encountered in liquid-state NMR.

The validity and accuracy of these assumptions depend on the details of each individual spin system and experiment. While RSS has demonstrated impressive performance with liquid-state NMR [28,29] and Spin Chemistry [26] simulations, the situation with solid state NMR in particular is more complicated [30,31]. In practice, accurate results are expected for solid state NMR in the following situations:

A. In static powders, where the lines of each individual crystallite may be sharp, but the powder average lines are broad. An inverse Fourier transform of the powder spectrum would decay very quickly – that is, irrespective of the actual dynamics in specific crystallites, the dynamics seen by the coil is short-time. For short-time dynamics, as per Observation 1 above, state space restriction is a good approximation.
B. In spinning powders, where the dipolar coupling density and strength are significantly reduced by magic angle spinning [51]. For sparse networks of small-amplitude couplings, as per Observations 2 and 3 above, restricted state space is a good approximation.

The observations above suggest that the state space should be restricted to low-order spin states connecting nearby (in the coupling sense) spins. This is quite easy to accomplish using graph theory and this was the method proposed in our initial paper on the subject [29]. The truncation level and the proximity tolerance are variable parameters left to user discretion – in the limit of no truncation the simulation is exact.

2.4. State space restriction using Krylov subspace analysis

An alternative formulation of the restricted state space formalism [17,18,26,28] is based on rigorous algebraic analysis of the subspaces that can or cannot be populated during spin-system evolution. The subject is quite rich, but one very general statement stands out:

Irrespective of the size and complexity of the spin system, the minimum matrix dimension required to simulate accurately any NMR, EPR or Spin Chemistry experiment is less than or equal to the number of digitization points in the free induction decay or its equivalent.

The proof is quite simple and may be formulated in two alternative ways:

A. From the digital signal-processing perspective, an n-point time-domain signal can only encode n distinct frequencies and can therefore be generated as a solution to a system of n linear differential equations. If the spectrum is not crowded, even fewer equations may be necessary.
B. From the algebraic perspective, in finite-step simulations the system trajectory under a given Liouvillian (Eq. (9)) up to the nth point is contained in the Krylov subspace K_n generated by the action of the propagator $\hat{\hat{P}}$ on the initial state vector $\hat{\rho}_0$:

$$K_n = \text{span}\left\{\hat{\rho}_0, \hat{\hat{P}}\hat{\rho}_0, \hat{\hat{P}}^2\hat{\rho}_0, \ldots, \hat{\hat{P}}^{n-1}\hat{\rho}_0\right\}; \quad \hat{\hat{P}} = e^{-i\hat{\hat{L}}\Delta t} \tag{10}$$

where Δt is the time step. Because there are n vectors in that list, the dimension of the space they span cannot exceed n, and is likely to be smaller than that. The same argument applies in the case where the Liouvillian is time-dependent, because the time dependence does not affect the number of vectors in the trajectory in Eq. (10).

The generality of the statement above is slightly shocking – it appears that the consequences of all spin dynamics simulations being finite-step and finite-time have so far been underestimated. Note that the proofs are algebraic and do not touch upon the "physical meaning" of the minimal system of equations. It is also true that these are pure existence proofs – they do not bring us any closer to actually finding the minimal basis: the first proof is not constructive and the second one would require the exact simulation to be carried out before the basis of the minimal space containing K_n could be obtained.

In the spin dynamics context, the question of finding or approximating K_n has received attention as early as 1980 – Moro and Freed [17,18,52] investigated the application of the Lanczos algorithm, which approximates the space spanned by matrix exponentials with the space spanned by matrix powers (which occur in the Taylor expansion of the exponential):

$$\begin{aligned}&\mathrm{span}\left\{\hat{\rho}_0, \hat{\hat{P}}\hat{\rho}_0, \hat{\hat{P}}^2\hat{\rho}_0, \ldots, \hat{\hat{P}}^{n-1}\hat{\rho}_0\right\} \\ &\quad\approx \mathrm{span}\left\{\hat{\rho}_0, \hat{\hat{L}}\hat{\rho}_0, \hat{\hat{L}}^2\hat{\rho}_0, \ldots, \hat{\hat{L}}^{m}\hat{\rho}_0\right\}\end{aligned} \tag{11}$$

The resulting algorithm proved very useful in highly general formulations of spin relaxation theory based on the stochastic Liouville equation [52], which often suffer from astronomical matrix dimensions. Essentially, the Lanczos algorithm identifies the subspace in which the spin system is confined (to a user-specified accuracy) and projects the entire simulation into that subspace.

For large reduced state spaces ($m > 500$) the straight Lanczos pruning can be expensive, and we recently proposed a large-scale implementation (called zero track elimination, ZTE [28]), which effectively reverses the logic outlined above – it identifies the vectors that *do not* appear in the Krylov subspace during the spin-system evolution and deletes them from the basis.

2.5. Krylov propagation

It is well known that the product of a matrix exponential and a vector, such as $\exp(-\mathrm{i}\hat{\hat{L}}\Delta t)\hat{\rho}$ can be computed much faster than the matrix exponential $\exp(-\mathrm{i}\hat{\hat{L}}\Delta t)$ itself [53], particularly if the matrix $\hat{\hat{L}}$ is sparse and Δt is small. All available avenues towards $\exp(-\mathrm{i}\hat{\hat{L}}\Delta t)$ involve matrix–matrix multiplications [54,55] and therefore have an $O(n^3)$ scaling with the matrix dimension, but $\exp(-\mathrm{i}\hat{\hat{L}}\Delta t)\hat{\rho}$ can be computed at $O(n^2)$ cost by re-ordering the multiplication operations in the corresponding Taylor expansion:

$$\begin{aligned}\exp\left(-\mathrm{i}\hat{\hat{L}}\Delta t\right)\hat{\rho} &= \left[\sum_{n=0}^{\infty}\frac{(-\mathrm{i}\Delta t)^n}{n!}\hat{\hat{L}}^n\right]\hat{\rho} \\ &= \sum_{n=0}^{\infty}\frac{(-\mathrm{i}\Delta t)^n}{n!}\left(\hat{\hat{L}}\left(\hat{\hat{L}}\left(\ldots\left(\hat{\hat{L}}\hat{\rho}\right)\right)\right)\right)\end{aligned} \tag{12}$$

so that only matrix–vector multiplications need to be performed. This approach is particularly well suited for propagation under time-dependent Liouvillians, where previously computed matrix exponentials cannot be re-used, and in low-memory situations, because no extra matrices need to be stored.

Although Eq. (12) is a good illustration, it is not the optimal way of computing $\exp(-\mathrm{i}\hat{\hat{L}}\Delta t)\hat{\rho}$. Much better accuracy and convergence rate are achieved if a generalized polynomial approximation to the exponential function is used instead [53]. All degree n polynomial approximations to $\exp(-\mathrm{i}\hat{\hat{L}}\Delta t)\hat{\rho}$ are elements of the Krylov subspace K_n defined as:

$$K_n = \mathrm{span}\left\{\hat{\rho}, (-\mathrm{i}\hat{\hat{L}}\Delta t)\hat{\rho}, (-\mathrm{i}\hat{\hat{L}}\Delta t)^2\hat{\rho}, \ldots, (-\mathrm{i}\hat{\hat{L}}\Delta t)^n\hat{\rho}\right\} \tag{13}$$

and the optimal way of computing $\exp(-\mathrm{i}\hat{\hat{L}}\Delta t)\hat{\rho}$ for a given degree n therefore is to project both $\hat{\hat{L}}$ and $\hat{\rho}$ into the basis of K_n, compute the product $\exp(-\mathrm{i}\hat{\hat{L}}\Delta t)\hat{\rho}$ there using standard matrix exponentiation techniques [54,55] and project the result back into the original space. This is known as the *Krylov method*; for well scaled $\hat{\hat{L}}\Delta t$ matrices $n \approx 30$ is generally sufficient [53]. Because of its small memory footprint and $O(n^2)$ scaling, the Krylov method enables direct time propagation in simulations with state space dimensions exceeding 10^6 [28].

2.6. Sparse array clean-up

An issue that is separate from matrix dimension is matrix sparsity. The use of sparse algebra in magnetic resonance simulations is well documented [5,6] and carries great advantages, because the binary nature of spin–spin couplings makes Hamiltonian and Liouvillian matrices very sparse. Unfortunately, the same cannot be said about powers of those matrices – if a Hamiltonian $\hat{H}$ is sparse, its density increases quickly when powers are taken; $\hat{H}^{-1}$ is almost always dense. This means that exponential propagators, if evaluated directly using Taylor, Chebyshev [10] or Padé [54,55] approximations, would also be dense – a serious problem for matrix dimensions in excess of 10^3 and a show-stopper for dimensions above 10^4 because of memory overflow.

This situation can be avoided if we restrict ourselves to only ever compute powers of $\mathrm{i}\hat{\hat{L}}\Delta t$ where $\Delta t < \|\hat{\hat{L}}\|^{-1}$. This is a reasonable constraint to impose – it is the definition of the Larmor time step, which is required anyway to avoid signal aliasing. The eigenvalues of $(\mathrm{i}\hat{\hat{L}}\Delta t)^n$ then drop off exponentially with n and many new non-zeros appearing after matrix multiplication are so small as to be inconsequential and can be dropped from the index, largely preserving (or even improving) matrix sparsity:

$$|L_{ij}| < \varepsilon \Rightarrow L_{ij} \to 0 \tag{14}$$

where ε is a user-supplied tolerance. A reasonable value for ε is a couple of orders of magnitude lower than the reciprocal experiment duration (*Spinach* defaults to $\varepsilon = 10^{-7}$ rad/s).

The clean-up procedure is applied every time a matrix is generated or multiplied in *Spinach*. The relatively small overhead of examining the non-zero index for small elements is compensated by the significant reduction of memory footprint and acceleration of matrix multiplication operations. It should be noted that sparse array clean-up is only effective for matrices that have been scaled inside the unit norm as described above – a propagator that takes the system forward by an hour would of course be dense.

2.7. SO(3) rotations

It is a general group-theoretical result that, in a rigid spin system undergoing overall rotation in three-dimensional space, the total Hamiltonian always admits the following expansion:

$$\hat{H} = \hat{H}_{\mathrm{iso}} + \sum_{l=1}^{\infty}\sum_{m=-l}^{l}\sum_{k=-l}^{l}\mathfrak{D}_{km}^{(l)}\hat{Q}_{km}^{(l)} \tag{15}$$

where $\hat{H}_{\mathrm{iso}}$ is the isotropic part of the Hamiltonian, $\mathfrak{D}_{km}^{(l)}$ are Wigner functions (of Euler angles or any other rotation parameters) specifying molecular orientation and $\hat{Q}_{km}^{(l)}$ are static spin operators. The situations in which Eq. (15) contains l ranks other than 2 are rare [56,57], and the expressions used in the rotations module of *Spinach* explicitly assume second rank interactions. The rotational basis operators $\hat{Q}_{km}$ are then related to second-rank irreducible spherical tensor operators:

$$\begin{aligned}\hat{Q}_{km} = &\sum_{L}\Phi_m(B,L)\hat{T}_k^{(2)}(B,L) + \sum_{LS}\Phi_m(L,S)\hat{T}_k^{(2)}(L,S) \\ &+ \sum_{S}\Phi_m(S,S)\hat{T}_k^{(2)}(S,S)\end{aligned} \tag{16}$$

where L and S indices enumerate the spins and the reduced orientation functions $\Phi_m(B,L)$, $\Phi_m(L,S)$ and $\Phi_m(S,S)$ depend on the eigenvalues and molecular frame orientations (given by another set of Wigner functions) of linear, bilinear and quadratic interaction tensors respectively:

$$\Phi_m = \frac{a_{XX} - a_{YY}}{2}\left(\mathfrak{D}^{(2)}_{m,-2} + \mathfrak{D}^{(2)}_{m,2}\right) + \frac{2a_{ZZ} - (a_{XX} + a_{YY})}{\sqrt{6}}\mathfrak{D}^{(2)}_{m,0} \tag{17}$$

If the full interaction tensor is known the interaction Hamiltonian can be written in terms of all the interaction tensor elements. The reduced orientation functions may then be written as [35]:

$$\begin{aligned}\Phi_{\pm 2} &= \frac{1}{2}(\boldsymbol{A}_{xx} - \boldsymbol{A}_{yy} \pm 2\mathrm{i}\boldsymbol{A}_{xy})\\ \Phi_{\pm 1} &= (\mp\boldsymbol{A}_{xz} - \mathrm{i}\boldsymbol{A}_{yz})\\ \Phi_0 &= \sqrt{\frac{3}{2}}\boldsymbol{A}_{zz}\end{aligned} \tag{18}$$

where $\boldsymbol{A}$ is a symmetric traceless interaction tensor. The irreducible spherical tensors in Eq. (16) are defined in the usual way [58]. An in depth derivation of Eqs. (15)–(18), as used in the *Spinach* rotations module, is given in [63].

2.8. Diagonalization-free relaxation theory

The standard way of computing the integral encountered in the Liouville space formulation of Bloch–Redfield–Wangsness relaxation theory [59–61]

$$\begin{aligned}\hat{\hat{R}} &= -\sum_{kmpq}\int_0^\infty G_{kmpq}(\tau)\hat{\hat{Q}}_{km}e^{-\mathrm{i}\hat{\hat{H}}_0\tau}\hat{\hat{Q}}^\dagger_{pq}e^{\mathrm{i}\hat{\hat{H}}_0\tau}d\tau,\\ G_{kmpq}(\tau) &= \langle\mathfrak{D}^{(2)}_{km}(0)\mathfrak{D}^{(2)*}_{pq}(\tau)\rangle\end{aligned} \tag{19}$$

(angular brackets denote ensemble average) is to diagonalize $\hat{\hat{H}}_0$ and expand $\hat{\hat{Q}}_{km}$ in its eigenstates, at which point the integral collapses into a collection of analytical Fourier transforms of the correlation functions $G_{kmpq}(\tau)$. In small systems, as well as those where $\hat{\hat{H}}_0$ is dominated by Zeeman interactions, this is easy, but as soon as the problem dimension exceeds about 10^4, the diagonalization is no longer feasible. The small-step propagator

$$\exp[-i\hat{\hat{H}}_0\Delta\tau], \quad \|\hat{\hat{H}}_0\|\Delta\tau \leqslant 1 \tag{20}$$

however, is still easily computed (*e.g.* using Taylor expansion with matrix clean-up described in Section 2.6) for $\hat{\hat{H}}_0$ dimensions in excess of 10^5, and therefore the fastest practical way of evaluating the integrals making up the sum in Eq. (19)

$$\int_0^\infty G(\tau)e^{-\mathrm{i}\hat{\hat{H}}_0\tau}\hat{\hat{Q}}e^{\mathrm{i}\hat{\hat{H}}_0\tau}d\tau \tag{21}$$

is by a fixed-step numerical quadrature, such as Boole's $O(h^7)$ rule [62]. A detailed account of this numerical integration method, as implemented in *Spinach*, is provided in Ref [63]. The general nature of the rotational factorization described in the previous section means that all cross-correlations are included automatically.

2.9. Symmetry factorization in Liouville space

In systems with magnetically equivalent spins, the state space can be reduced by exploiting the associated permutation symmetry. The symmetry adapted linear combinations $\hat{O}^{(\Gamma)}_k$ of the initial basis operators $\hat{O}_k$ belonging to the irreducible representation Γ of a group G can be calculated using the character formula [64–66]:

$$\hat{O}^{(\Gamma)}_k = \frac{1}{N}\sum_{g\in G}\chi^{(\Gamma)}_g g(\hat{O}_k) \tag{22}$$

in which $\chi^{(\Gamma)}_g$ is the character of the group element g, N is the order of the group, $g(\hat{O}_k)$ is the result of g acting on $\hat{O}_k$ – this action permutes the order of the direct product components in $\hat{O}_k$ and therefore indices in their descriptors in Eq. (2). The summation is carried over the individual elements of the symmetry group.

We have recently shown that Liouville space trajectories would normally stay in the fully symmetric irreducible representation [26]. This simplifies Eq. (22) – the basis simply needs to be fully symmetrised with respect to all group operations. Details of this formalism and its extension to multiple groups of equivalent spins are presented in our recent paper [26] and earlier papers by other authors [16,67–70]. Eq. (22) as well as the auxiliary group direct product procedures [64–66] are used by the *Spinach* symmetry module.

2.10. State space connectivity tracing

The Liouvillian matrix, even after state space restriction, ZTE and symmetry factorization, is still very sparse – each state is only directly connected to a few other states. It is often the case that there are disconnected sub-networks within this connectivity network. To take advantage of this fact, the Liouvillian may be treated as the adjacency matrix of a graph: diagonal elements correspond to nodes and off-diagonal elements to edges of the graph. The non-interacting subspaces then correspond to disjoint subgraphs and may be found in $O(n)$ time with respect to the number of non-zeros in the Liouvillian [26]. To this end *Spinach* employs Tarjan's graph partitioning algorithm [71,72]. The input parameter is a binary form of the Liouvillian:

$$P_{nk} = \begin{cases}1 & \text{if } |L_{nk}| > \varepsilon\\ 0 & \text{otherwise}\end{cases} \tag{23}$$

where ε, a user-supplied tolerance, is set to zero for an exact procedure, but may be set higher to increase the sparsity of the Liouvillian. Tarjan's algorithm returns node indices for each disjoint subgraph, which correspond to state indices for each non-interacting subspace.

If the state space can be written as a direct sum of N non-interacting subspaces, the dynamics of an observable $\hat{a}$ in the full space is a sum over such subspaces:

$$\begin{aligned}\langle\hat{a}\rangle(t) &= \langle\hat{a}|\hat{\rho}(t)\rangle = \left\langle\hat{a}|e^{-\mathrm{i}\hat{\hat{L}}t}\hat{\rho}(0)\right\rangle = \left\langle\bigoplus_{n=1}^N\hat{a}^{(n)}\middle|e^{-\mathrm{i}\left(\bigoplus_{n=1}^N\hat{\hat{L}}\right)t}\bigoplus_{n=1}^N\hat{\rho}(0)\right\rangle\\ &= \left\langle\bigoplus_{n=1}^N\hat{a}^{(n)}\middle|\bigoplus_{n=1}^N e^{-\mathrm{i}\hat{\hat{L}}^{(n)}t}\bigoplus_{n=1}^N\hat{\rho}(0)\right\rangle = \sum_{n=1}^N\left\langle\hat{a}^{(n)}|e^{-\mathrm{i}\hat{\hat{L}}^{(n)}t}\hat{\rho}^{(n)}(0)\right\rangle\\ &= \sum_{n=1}^N\langle\hat{a}^{(n)}\rangle(t)\end{aligned} \tag{24}$$

Taking advantage of this block structure significantly accelerates the simulations – for example, over 500 independent subspaces are identified in the HSQC simulation of sucrose (`hsqc_test_1.m` file in the *Spinach* examples directory), with the result that the biggest subspace encountered in the fairly sophisticated simulation of a 30-spin system (!) has the dimension of 238. Some of the blocks often turn out to be empty. The *Spinach* connectivity tracing module runs a zero check on all subspaces; any subspace that is not populated is dropped.

2.11. Conservation law screening

In the rare cases where an analytical criterion for subspaces is known, it is possible to perform the connectivity tracing analytically on the state list before the operators are built. If an operator $\hat{A}$ commutes with the system Hamiltonian its corresponding observable must be a constant of motion. Any state with an expectation value $\langle\hat{A}\rangle$ that does not coincide with that of the initial state violates the conservation law, will never be populated and may be eliminated. The active subspace is therefore the intersection of $\langle\hat{A}\rangle = const$ subspaces for every linearly independent operator $\hat{A}$ in the null space of $\hat{\hat{H}}$ [26]. Additionally, if $\hat{\rho}(0)$ is an eigenstate of the commutation superoperator $\hat{\hat{A}}$, then the eigenvalue a must also be conserved. Almost inevitably $\hat{\hat{A}}$ has more than one possible eigenvalue, and states may be grouped according to common values of a. These groups are non-interacting subspaces as described in the section above.

Simple quantities often conserved during the evolution of spin systems are the total spin $\hat{S}^2$ and the total Z-component of spin $\hat{S}_Z$:

$$\hat{S}^2 = \sum_{n,k}\left(\hat{S}_Z^{(n)}\hat{S}_Z^{(k)} + \frac{1}{2}\left(\hat{S}_+^{(n)}\hat{S}_-^{(k)} + \hat{S}_-^{(n)}\hat{S}_+^{(k)}\right)\right); \quad \hat{S}_Z = \sum_k \hat{S}_Z^{(k)} \tag{25}$$

As the ISTs are eigenstates of the corresponding superoperators, with eigenvalues given by $l(l+1)$ and m respectively, a rapid screening procedure is possible using indexed notation from Eq. (2).

Examples of conservation-screened basis sets in *Spinach* are *ESR-1*, which is adapted for high-field ESR spectroscopy, and *MARY-1*, intended for MARY (magnetically altered reaction yield) [93,94] simulations. *ESR-1* includes the complete state space for all electrons (which are pulsed and observed), but only the identity state and $\hat{T}_{l0}$ for the nuclei, taking into account the fact that, in simple pulsed ESR experiments at high field, the transverse nuclear states are never populated [73]. The *MARY-1* basis set only includes the "zero-quantum subspace" – states with $\langle\hat{S}_Z\rangle = 0$, taking advantage of the corresponding conservation law.

3. *Spinach* kernel

The code base of *Spinach* has two parts: the kernel, containing very general functions applicable to any spin system, and the user-land – a collection of case-specific functions, experiment models and pulse sequences. A schematic of this arrangement, showing the flow of data between major functions in the kernel and the user-land, is given in Fig. 1.

At the start of the simulation the spin system information is either supplied directly to the kernel, using the syntax documented in Table S1 of the Supplementary Information, or read in from a third-party data file using one of the import filters in the user-land (*Gaussian03* [74] logs and *Simpson* [7,22] `*.in` files at the time of writing). The kernel oversees the utilization of this information in the most efficient way possible and supplies the user-land with directly usable objects, such as superoperator matrices (Hamiltonian, relaxation, kinetics, *etc.*), state vectors and propagators. These can either be used directly using the established spin dynamics simulation techniques, or supplied to the various time evolution modules also provided by the kernel (Fig. 1). Importantly, the sophisticated state space restriction technology reviewed in the previous section is handled transparently – all procedures described in Section 2 are performed automatically (but may also be applied manually using the syntax documented in Section 4). The kernel guarantees that the matrices and vectors it provides may be used directly, meaning that any existing Liouville-space spin dynamics simulation code may be ported to *Spinach* (and considerably accelerated for large spin systems) by simply using the superoperators and state vectors supplied by the *Spinach* kernel instead of the usual direct product matrices.

The flow of superoperators and state vectors through the kernel is illustrated in the left panel of Fig. 1. The product superoperators, generated by `p_superop`, are used by `k_superop` to generate the kinetics superoperators and by `c_superop` to generate commutation superoperators. The latter are used by `h_superop` to generate the isotropic part of the Hamiltonian superoperator and the rotational basis for its anisotropic part. Those are then used in the rotation (`orientation`) and relaxation (`r_superop`) modules to generate specific orientation Hamiltonians and relaxation superoperators respectively.

The kernel provides basic time propagation functionality and trajectory-level state space restriction tools (Fig. 1, bottom left), such as connectivity tracing [26] and zero track elimination [28]. These are also performed transparently and automatically by the `evolution` function.

3.1. Kernel data structure

The kernel keeps all simulation information in a single structured array called `spin_system`, which is generated by the `create.m` function and updated by the `basis.m` function. Its first level sub-fields are listed in Fig. 2 and each subfield is further elaborated upon *via* complete schematics given in the Supplementary Information, Figs. S1–S4. The user-land can use this structure to retrieve information about any aspect of the simulation. The `spin_system` structure is also used throughout the kernel as a source of information used to generate various infrastructure objects (*e.g.* superoperators) and perform infrastructure operations (*e.g.* propagation) in the most efficient way possible.

3.2. Spin system creation and basis specification

These functions must be the first kernel call. They control the parameters of the simulation, the structure of the spin system, tolerances on all approximations, output level and algorithm selection. After all parameters are processed, these functions create the `spin_system` data structure. The call syntax is

```
spin_system=create(sys,inter);
spin_system=basis(spin_system,bas);
```

where the `sys` and `inter` structures are described in Table S1 and the `bas` structure is described in Table S2 in the Supplementary Information. It is often convenient to generate `sys` and `inter` structures using one of the import filters available in the user-land (Section 5). Both functions produce extensive diagnostic output about the structure and interactions found within the spin system. All approximations and assumptions are also explicitly printed.

Spinach supports complete, restricted and connectivity-adaptive basis sets. The greater the coupling density in the spin system and the longer the simulation, the bigger basis is required. Liquid-state NMR simulations (pulse-acquire, DQF-COSY, HSQC, *etc.*) are accurately simulated with *IK-1* and *IK-2* basis sets (Table S2); reducing the basis causes a gradual loss of multiplicity detail, as shown in Fig. 3. This basis size dependence is similar to the one observed in the electronic structure theory – the bigger the basis, the more accurate the result.

Every approximation in *Spinach* has an associated set of tolerances that may be altered by setting `sys.tols.*` sub-fields (the complete list is given in the Supplementary Information Table S3). The defaults are very conservative and guarantee accurate results in a large variety of simulations. Relaxing these tolerances from their default values would in many cases significantly accelerate the simulation.

3.3. Product superoperators, commutation superoperators and state vectors

The basic building blocks for all Liouville-space superoperators are product and commutation superoperators generated by direct products of irreducible spherical tensors:

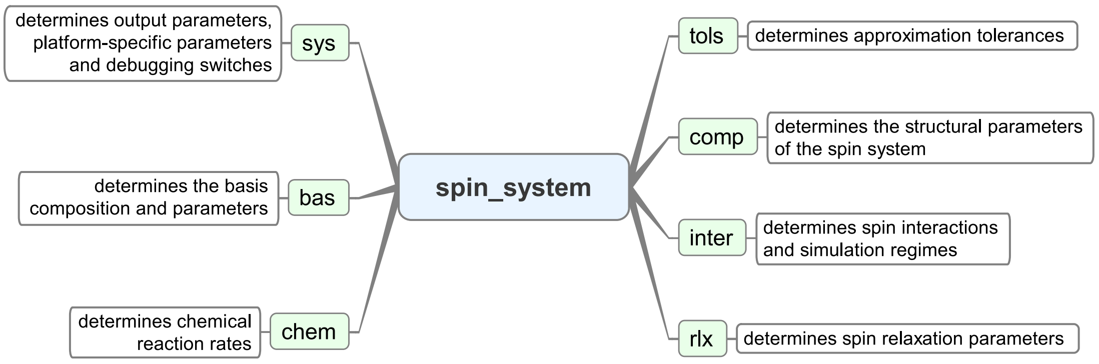
spin_system
sys
determines output parameters, platform-specific parameters and debugging switches
bas
determines the basis composition and parameters
chem
determines chemical reaction rates
tols
determines approximation tolerances
comp
determines the structural parameters of the spin system
inter
determines spin interactions and simulation regimes
rlx
determines spin relaxation parameters

Fig. 2. Top level sub-fields of the `spin_system` data structure, which controls the behavior of all algorithms and functions in *Spinach* kernel. This structure is constructed by `create` (Table S1 in the Supplementary Information) and updated by `basis` (Table S2 in the Supplementary Information).

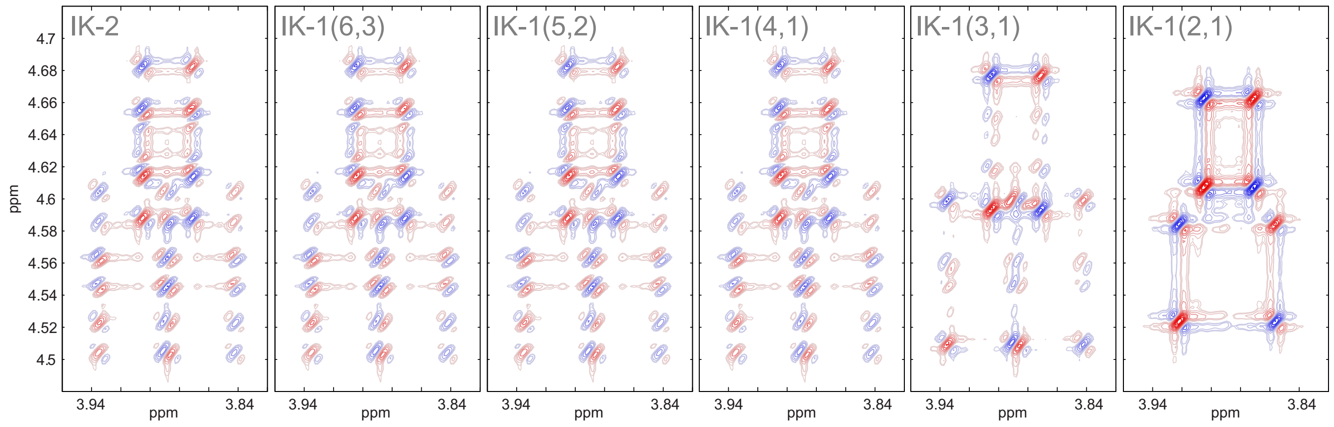

Fig. 3. A cross-peak from the DQF-COSY spectrum of sucrose (22 ^{1}H nuclei, `dqf_cosy_test_2.m` file in the *Spinach* example set) computed using explicit time propagation in Liouville space as a function of the basis set (Table S2 in the Supplementary Information) used for the simulation. The size of the largest matrix encountered in the simulation is 206 for *IK-2*, 4906 for *IK-1(6,3)*, 1367 for *IK-1(5,2)*, 418 for *IK-1(4,1)*, 123 for *IK-1(3,1)* and 21 for *IK-1(2,1)*.

$$\hat{Q} = \bigotimes_{n=1}^{N} \hat{T}_{l_n m_n}; \quad \hat{\hat{Q}}_- = [\hat{Q}, \cdot]_- = \hat{\hat{Q}}_L - \hat{\hat{Q}}_R;$$
$$\hat{\hat{Q}}_+ = [\hat{Q}, \cdot]_+ = \hat{\hat{Q}}_L + \hat{\hat{Q}}_R; \quad \hat{\hat{Q}}_L \hat{\rho} = \hat{Q}\hat{\rho}; \quad \hat{\hat{Q}}_R \hat{\rho} = \hat{\rho}\hat{Q} \tag{26}$$

where $\hat{\hat{Q}}_L$ and $\hat{\hat{Q}}_R$ are left and right product superoperators, $\hat{\hat{Q}}_-$ is the commutation superoperator corresponding to $\hat{Q}$, and $\hat{\hat{Q}}_+$ is the anti-commutation superoperator. Product, commutation and anticommutation superoperators are available by using the following syntax:

```
L=c_superop(spin_system,opspec);
L=a_superop(spin_system,opspec);
L=p_superop(spin_system,opspec,side);
```

where `L` is the superoperator in question, `side` can be `'left'` or `'right'` and `opspec` is the *Spinach* operator specification – a compact internal notation used to represent spin operators and states. An `opspec` is a string of integers giving the states of each individual spin in the order of their appearance in the `sys.isotopes` variable. The state indexing convention follows Eq. (3). The first nine ISTs are listed explicitly in Table S4 in the Supplementary Information, along with their associated spherical harmonics. The sequence of integers in `opspec` is mapped into the sequence of operators in the direct product, for example:

$$[0203102100] \Rightarrow [\hat{E} \otimes \hat{L}_Z \otimes \hat{E} \otimes \hat{L}_- \otimes \hat{L}_+ \otimes \hat{E} \otimes \hat{L}_Z \otimes \hat{L}_+ \otimes \hat{E} \otimes \hat{E}, \cdot]$$

Similar syntax is used by the `statevec.m` function that returns the state vectors:

```
rho=statevec(spin_system,opspec);
```

It is not always convenient to supply the superoperator and state specification in the form described above, and the user-land has wrapper functions providing a more eye-friendly syntax:

```
rho=state(spin_system,oper,spins)
L=operator(spin_system,oper,spins)
```

where `spins` may be set to `'all'`, `'1H'`, `'13C'`, *etc.* and `oper` to `'Lz'`, `'L+'` or `'L-'`. These functions parse the user-friendly input, convert it into `opspec` and call `statevec` and `c_superop` respectively.

3.4. Hamiltonian commutation superoperator

The kernel function returning the isotropic Hamiltonian commutation superoperator has no adjustable parameters:

```
H_iso=h_superop(spin_system);
```

except for the option to ignore the secularity status of all interactions and return the full non-secular laboratory frame Hamiltonian superoperator:

```
H_iso=h_superop(spin_system,'full');
```

This second option is most commonly invoked by the relaxation superoperator module and may be useful in the rare cases where a laboratory frame simulation needs to be performed. A call with two output parameters

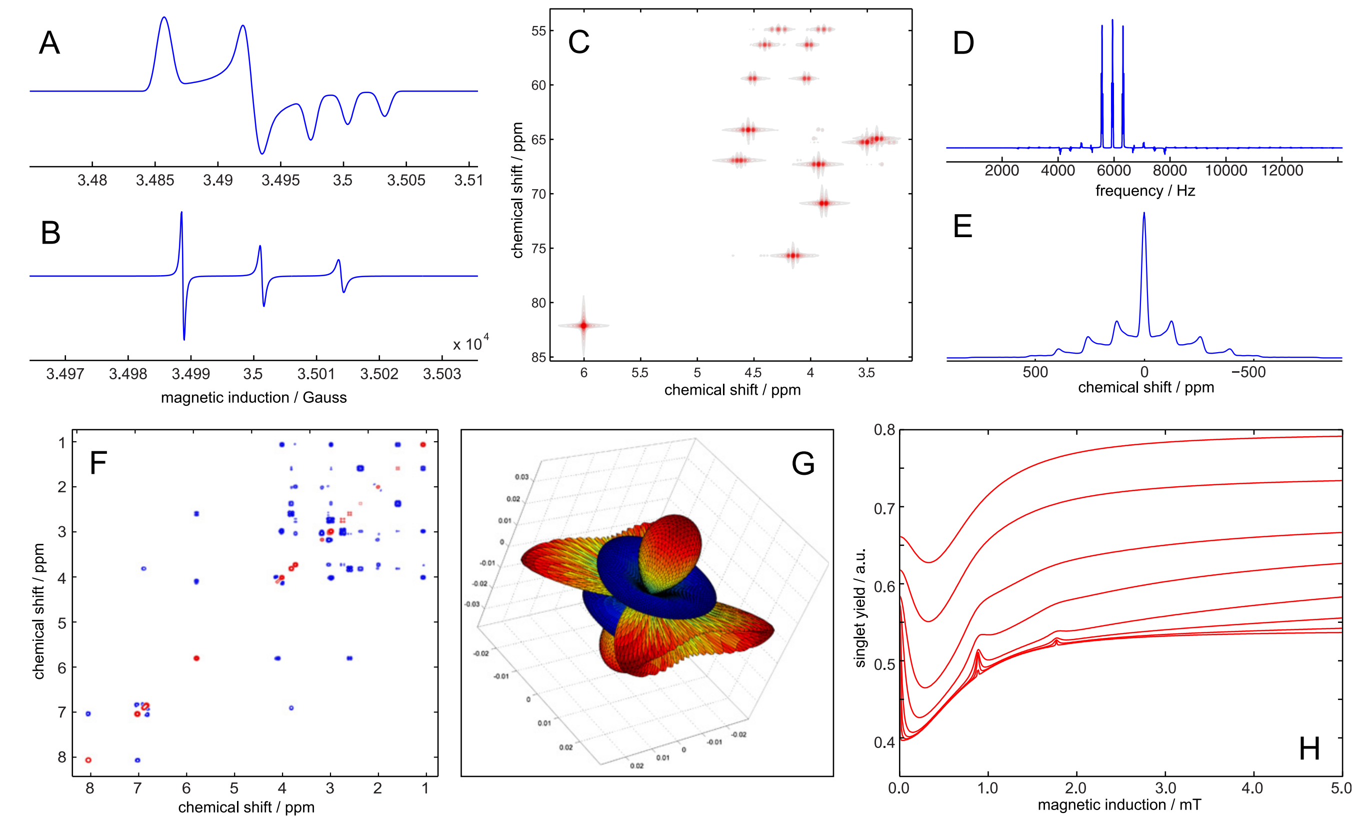

Fig. 4. Screenshots of the results produced by some of the example simulation files supplied with *Spinach*. Full simulation details are available within the files specified. (A) Solid-state ESR spectrum of a nitroxide radical (`esr/solids_test_1.m`) – two-spin system; state space reduction: 36 → 12. (B) Liquid state ESR spectrum of the same radical with BRW relaxation theory enabled (`esr/relaxation_test_2.m`) – seven spin system; state space reduction: 36 → 3. (C) HETCOR spectrum of sucrose computed using explicit time propagation in Liouville space, magnetic parameters imported from a DFT simulation (`nmr/hetcor_test_1.m`) – 34 spin system; state space reduction: 10^{20} → 158 subspaces with ⩽200 states each. (D) The excitation profile of a Q5 pulse applied to a linear chain of 30 strongly coupled (nearest neighbor) protons (`nmr/shaped_pulse_test_1.m`) – 31 spin system; state space reduction: 10^{18} → 323. (E) Solid state ^{235}U NMR spectrum of a uranium nucleus with an axial quadrupolar interaction (`nmr/solids_test_4.m`), complete basis set. (F) NOESY spectrum of strychnine, computed with full Redfield superoperator using explicit time propagation in Liouville space, magnetic parameters and coordinates imported from a DFT simulation (`nmr/noesy_test_1.m`) – 21 spin system; state space reduction: 10^{13} → 246 subspaces with ⩽1510 states each. (G) Singlet yield anisotropy for a radical pair with a single nucleus with an axial hyperfine coupling (`spin_chemistry/maryan_test_1.m`) – three spin system; state space reduction: 64 → 4 subspaces with ⩽15 states each. (H) Magnetic field dependence of the singlet yield for a radical pair (`spin_chemistry/mary_test_2.m`) – eight spin system; state space reduction: 65,536 → 4 subspaces with ⩽290 states each. Sub-fields of the spin_system data structure controlling the composition of the spin system (comp) and the basis set used during the simulation (bas).

```
[H_iso,Q]=h_superop(spin_system);
```

returns the rotational basis $\{\hat{\hat{Q}}_{km}\}$, computed using Eq. (16), for the anisotropic part of the Hamiltonian commutation superoperator. The anisotropic part for a specific spin system orientation may be obtained by supplying the resulting `Q` variable, along with the three Euler angles, to the `orientation` function:

```
H_aniso=orientation(spin_system,Q,...
[alpha beta gamma]);
```

This function is used, in particular, during the calculation of powder averages using pre-computed orientation sets corresponding to Lebedev spherical integration grid points [75]. Powder averaging in general is a user-land problem – the kernel only provides the grids.

Unless the `sys.lowfield` switch is set or triggered by setting `sys.magnet` to zero, *Spinach* would run time propagation in the multiply rotating frame with respect to all Zeeman interactions in the system. The decisions on the secularity status of all interactions are made during the call to `secularity.m` kernel function. For Zeeman interactions: all terms $(\hat{L}_Z, \hat{L}_\pm)$ at low field, secular terms ($\hat{L}_Z$ only) at high field. For bilinear couplings: all terms $(\hat{T}_0^{(2)}, \hat{T}_{\pm1}^{(2)}, \hat{T}_{\pm2}^{(2)})$ at low field, secular terms ($\hat{T}_0^{(2)}$ only) between spins of the same type at high field and weak coupling terms ($\hat{L}_Z\hat{S}_Z$ only) between spins of different type at high field. For quadratic couplings: all terms $(\hat{T}_0^{(2)}, \hat{T}_{\pm1}^{(2)}, \hat{T}_{\pm2}^{(2)})$ at low field, secular terms ($\hat{T}_0^{(2)}$ only) at high field. Partially rotating frames (such as those used in ENDOR and ESEEM simulations) can also be specified.

The primary danger associated with the common practice of running simulations in the multiply rotating frame is in the assumption that the non-secular interaction terms are negligible. While this is certainly true for mainstream NMR and ESR simulations, care must be taken when running at magnetic fields that do not greatly exceed the magnitude of couplings in the system. The internal heuristic in *Spinach* is likely to make the right choice, but we would still encourage users to inspect the diagnostic output produced by `h_superop` to ensure that the secularity assumptions are correctly set for the system in question. The secularity status of all interactions may be specified manually by setting the `inter.zeeman.strength` and `inter.coupling.strength` variables and passing them to `create` (Section 3.2).

3.5. Relaxation superoperator

The function itself has no adjustable parameters:

```
R=r_superop(spin_system);
```

and uses the information (theory, correlation time, *etc.*) that the user has supplied to the `create` function. While the 'none' and the 'damp' options (Table S1 in the Supplementary Information) are obvious, some further notes are in order for Redfield theory [59–61].

Spinach implements a very general case of Bloch–Redfield–Wangsness relaxation theory (see Sections 2.6 and 2.7 as well as our recent papers [20,63] on the subject), which includes the "difficult" contributions, such as interaction rhombicities, cross-correlations [76,77] and dynamic frequency shifts [78,79]. Contributions from all couplings present in the system (including quadrupolar and zero-field splitting) are included, and the theory is not restricted to high-field systems – the role of $\hat{L}_0$ can be played, for example, by a particularly large isotropic hyperfine coupling. The relaxation superoperator evaluation procedure, detailed in Section 2.7, is specifically designed to allow very large matrix dimensions (up to $\sim 10^6$ on a desktop workstation) to be processed [63].

Relaxation to thermal equilibrium, if requested by setting `inter.equilibrium` to *'thermal'*, proceeds using the method outlined by Levitt and Di Bari [80], whereby the relaxation to the equilibrium state vector $\hat{\rho}^{(0)}$

$$\frac{d}{dt}\begin{pmatrix}\rho_1\\ \vdots\\ \rho_n\end{pmatrix} = -\mathrm{i}\begin{pmatrix}L_{11} & \cdots & L_{1n}\\ \vdots & \ddots & \vdots\\ L_{n1} & \cdots & L_{nn}\end{pmatrix}\begin{pmatrix}\rho_1\\ \vdots\\ \rho_n\end{pmatrix} + \begin{pmatrix}R_{11} & \cdots & R_{1n}\\ \vdots & \ddots & \vdots\\ R_{n1} & \cdots & R_{nn}\end{pmatrix}\begin{pmatrix}\rho_1-\rho_1^{(0)}\\ \vdots\\ \rho_n-\rho_n^{(0)}\end{pmatrix} \tag{27}$$

is introduced as an extra row and column in the Liouvillian:

$$\frac{d}{dt}\begin{pmatrix}1\\ \rho_1\\ \vdots\\ \rho_n\end{pmatrix} = -\mathrm{i}\begin{pmatrix}0 & 0 & \cdots & 0\\ -\mathrm{i}[\hat{\hat{R}}\rho^{(0)}]_1 & L_{11}+\mathrm{i}R_{11} & \cdots & L_{1n}+\mathrm{i}R_{1n}\\ \vdots & \vdots & \ddots & \vdots\\ -\mathrm{i}[\hat{\hat{R}}\rho^{(0)}]_n & L_{n1}+\mathrm{i}R_{n1} & \cdots & L_{nn}+\mathrm{i}R_{nn}\end{pmatrix}\begin{pmatrix}1\\ \rho_1\\ \vdots\\ \rho_n\end{pmatrix} \tag{28}$$

Because the first element of any *Spinach* basis is the unit operator, this change of dimension is not required and the procedure amounts to introducing a one-way cross-term to the unit operator. The time propagation functions supplied by the kernel (`evolution` and `step`) are aware of Eq. (28) and will automatically relax the system towards the equilibrium requested.

3.6. Kinetics superoperator

The chemical kinetics superoperator is requested by:

```
K=k_superop(spin_system);
```

and uses the information from `inter.chem` variable that the user supplied to `create`. *Spinach* takes advantage of the fact that bosons and fermions of the same type are fundamentally identical and assumes that chemical exchange processes amount to the *transport and flux of magnetization* within a topologically fixed spin system rather than to moving spins. For chemical exchange, networks of first-order chemical processes are supported:

$$\frac{d}{dt}[A_j] = \sum_{m=1}^{n} k_{jm}[A_m] \;\Rightarrow\; \frac{d}{dt}\vec{A}(t) = \mathbf{K}\vec{A}(t) \tag{29}$$

where $[A_j]$ is the concentration of spin j and k_{jm} is the rate of magnetization transport from spin m to spin j. An important condition on the elements of the kinetics matrix $\mathbf{K}$ is that the diagonal elements must balance out the off-diagonal elements so that the law of the conservation of matter is observed.

From the simulation perspective, when magnetization is pumped over from site n to site m in a spin system, the state populations are pumped across between the following states:

$$\ldots\otimes\hat{L}_\alpha^{(n)}\otimes\ldots \;\to\; \ldots\otimes\hat{L}_\alpha^{(m)}\otimes\ldots \tag{30}$$

where $\alpha = \{z, +, -, \ldots\}$ is an index running over the elements of the spin state space. The task of constructing a superoperator that would perform this action amounts to subtracting a slice of the population from $\langle\ldots\otimes\hat{L}_\alpha^{(n)}\otimes\ldots\rangle$ and forwarding it to $\langle\ldots\otimes\hat{L}_\alpha^{(m)}\otimes\ldots\rangle$:

$$\hat{\hat{K}} = k_{mn}\left(\left|\hat{L}_\alpha^{(m)}\right\rangle\left\langle\hat{L}_\alpha^{(m)}\right| - \left|\hat{L}_\alpha^{(n)}\right\rangle\left\langle\hat{L}_\alpha^{(n)}\right|\right) \tag{31}$$

The superoperator in brackets shifts the population from one state to another [45].

The policy on chemical transport of multi-spin orders is case-dependent. In principle, the coherence with any observer spins must be preserved during chemical exchange, that is:

$$\ldots\otimes S\otimes\ldots\otimes\hat{L}_\alpha^{(n)}\otimes\ldots \;\to\; \ldots\otimes S\otimes\ldots\otimes\hat{L}_\alpha^{(m)}\otimes\ldots \tag{32}$$

and, so long as we are treating a single system (or an ensemble of systems, which have chemical exchange happening inside them), this should be taken into account. In a typical solution, however, this would lead to the emergence of inter-molecular coherences between identical molecules – a situation that density matrix formalism is ill-equipped to accommodate. Because such coherences are non-observable and the spin is very unlikely to jump back to the exact molecule that it originally came from, the coherences in question are counted as lost:

$$\hat{\hat{K}} = -k_{mn}\left|\hat{S}\otimes\hat{L}_\alpha^{(n)}\right\rangle\left\langle\hat{S}\otimes\hat{L}_\alpha^{(n)}\right| \tag{33}$$

and the corresponding contribution to the kinetics superoperator effectively causes relaxation.

For reactive radical pair simulations of the kind often encountered in Spin Chemistry [81–83], the singlet and triplet recombination superoperators may be included into $\hat{\hat{K}}$ by setting the `inter.chem.rp` variables (Table S1 in the Supplementary Information). The theories available at the time of writing are Haberkorn [81] and Jones–Hore [83]:

$$\hat{\hat{K}}_{\mathrm{H}}\hat{\rho} = -\frac{1}{2}\left(k_{\mathrm{T}}[\hat{Q}_{\mathrm{T}},\hat{\rho}]_+ + k_{\mathrm{S}}[\hat{Q}_{\mathrm{S}},\hat{\rho}]_+\right) \tag{34}$$

$$\hat{\hat{K}}_{\mathrm{JH}}\hat{\rho} = -(k_{\mathrm{S}}+k_{\mathrm{T}})\hat{\rho} - k_{\mathrm{S}}\hat{Q}_{\mathrm{T}}\hat{\rho}\hat{Q}_{\mathrm{T}} - k_{\mathrm{T}}\hat{Q}_{\mathrm{S}}\hat{\rho}\hat{Q}_{\mathrm{S}} \tag{35}$$

where $k_{\mathrm{S,T}}$ are singlet and triplet radical pair recombination rate constants, $[]_+$ denotes an anticommutator, and $\hat{Q}_{\mathrm{S,T}}$ are singlet and triplet projection operators.

3.7. Exponential propagator

For a given Liouvillian superoperator $\hat{\hat{L}}$ and a given time step Δt, the exponential propagator $\exp(-\mathrm{i}\hat{\hat{L}}\Delta t)$ can be requested as:

```
P=propagator(spin_system,L,delta_t);
```

The seemingly unsophisticated choice of matrix exponentiation algorithm in *Spinach* – Taylor series with scaling and squaring [54,55]

$$\exp(-\mathrm{i}\hat{\hat{L}}\Delta t) = \sum_{n=1}^{\infty}\frac{(-\mathrm{i}\Delta t)^n}{n!}\hat{\hat{L}}^n \quad \exp(-\mathrm{i}\hat{\hat{L}}\Delta t) = \left[\exp\left(\frac{-\mathrm{i}\hat{\hat{L}}\Delta t}{k}\right)\right]^k \tag{36}$$

is a consequence of the need to handle very large sparse matrices, which must stay sparse and for which only approximate norms are known. All other methods either involve dense matrices (*e.g.* the division operation in the Padé method and the diagonalization in the eigenvalue method [55]) or place strict constraints on the matrix norm (*e.g.* Chebyshev exponentiation [8,10]). Taken together with the requirement to maintain matrix sparsity at all times (see Section 2.5 above), and the need to work gracefully with inexperienced users, this leaves us with Taylor approximation, with its infinite convergence radius and retention of sparsity, as the optimal choice. In practice, the scaled Taylor series converges in 10–15 iterations. The use of Chebyshev approximation can be requested by setting the `sys.tols.exponentiation` switch (Table S3) to `'chebyshev'`.

The propagator function monitors the matrix density and switches over to dense algebra if it cannot avoid exceeding the matrix density threshold, which is set at 25% by default. For small time steps, the number of non-zeros in the propagator is similar to that of the Liouvillian. It increases rapidly as soon as the time step crosses the $\|\hat{\hat{L}}\|\Delta t = 1$ threshold.

Derivatives of the exponential propagator

$$\begin{aligned}\frac{\partial}{\partial\alpha}\exp(-\mathrm{i}\hat{\hat{L}}\Delta t) &= \sum_{n=1}^{\infty}\frac{(-\mathrm{i}\Delta t)^n}{n!}\sum_{k=0}^{n-1}\hat{\hat{L}}^k\hat{\hat{L}}'_\alpha\hat{\hat{L}}^{n-k-1}\\ &= \exp(-\mathrm{i}\hat{\hat{L}}\Delta t)\sum_{n=1}^{\infty}\frac{(-\mathrm{i}\Delta t)^n}{n!}[\hat{\hat{L}},[\hat{\hat{L}},\ldots[\hat{\hat{L}},\hat{\hat{L}}'_\alpha]\ldots]] \qquad (37)\end{aligned}$$

with respect to an arbitrary linear parameter α of an arbitrary operator $\hat{\hat{L}}'_\alpha$ occurring in $\hat{\hat{L}}$ are available by the use of the following syntax

```
P=propagator(spin_system,L,delta_t,derivatives);
```

where `derivatives` is a cell array of matrices corresponding to the $\hat{\hat{L}}'_\alpha$ operators for all the necessary parameters α. For the same reasons (sparsity, convergence and robustness) as the Taylor series above, the commutator series given in Eq. (37) is used for the evaluation of the propagator derivatives. They are used, in particular, by the GRAPE gradient [84,85] module described in Section 3.11 below.

3.8. Thermal equilibrium

The thermal equilibrium state vector may be requested by calling:

```
rho=equilibrium(spin_system);
```

and setting the `inter.temperature` variable (Table S1 in the Supplementary Information) before calling `create`. The general problem of finding the thermal equilibrium state of an ensemble of coupled spin systems has exponential complexity, and *Spinach* only returns the equilibrium state with respect to the Zeeman Hamiltonian under the assumption that other interactions in the system have a negligible effect on the equilibrium distribution of spin polarization. A warning is printed to this effect every time `equilibrium` is called.

Because longitudinal spin states in Liouville space correspond to *polarizations* rather than *populations*, the state vector returned by `equilibrium` would in some cases contain small numbers (*e.g.* for ^{15}N at room temperature). It is therefore advisable, when running with accurate thermal equilibria at high temperatures, to inspect the trajectory-level state space reduction tolerances (Table S3) and make sure that important states are not dropped automatically because of their low occupancies. The default tolerances are in most cases tight enough.

It should also be noted that setting the `inter.temperature` variable to be *identically* equal to zero does *not* collapse the system into the lowest possible collective energy level, but causes `equilibrium` to return the simplified equilibrium state that is often used in basic NMR and ESR simulations: $\hat{\rho}_0 = \sum_k \hat{L}_Z^{(k)}$; this is the default.

Relaxation to thermal equilibrium may be requested by setting `inter.equilibrium` to *'thermal'*. Further details are given in Section 3.5 above.

3.9. Trajectory-level state space reduction

Even the reduced basis sets, such as IK-2 and ESR-1 (Table S2 in the Supplementary Information), often turn out to be excessive and contain unpopulated dimensions [28] which are specific to the experiments being simulated. They can be pruned using the zero track elimination procedure (Section 2.4) by calling

```
P=zte(spin_system,L,rho);
```

where `L` is the Liouvillian superoperator, `rho` is the initial state vector and `P` is a matrix projecting the system from the current to the reduced basis set and back:

$$\begin{aligned}&\hat{\hat{L}}_{\mathrm{ZTE}} = P^{\mathrm{T}}\hat{\hat{L}}P \quad \hat{\rho}_{\mathrm{ZTE}} = P^{\mathrm{T}}\hat{\rho}\\ &\hat{\hat{L}} = P\hat{\hat{L}}_{\mathrm{ZTE}}P^{\mathrm{T}} \quad \hat{\rho} = P\hat{\rho}_{\mathrm{ZTE}}\end{aligned} \qquad (38)$$

The reverse transformation amounts to placing zero tracks back to their original locations. The detailed analysis of the ZTE technique is given in our recent paper [28] and summarized in Section 2.4 above.

While diagonalization is in most cases unfeasible even for reduced Liouvillians because of its $O(n^3)$ cost in time and memory, a sparsity-preserving block-diagonalization can be achieved at $O(n)$ cost using the connectivity tracing procedure proposed in our recent paper [26] and summarized in Section 2.9. Projectors into the subspaces that are disconnected in the current basis may be requested by calling

```
P=path_trace(spin_system,L);
```

where `P` is a cell array of projectors into disconnected subspaces. The projectors are applied in the same way as the ZTE projector in Eq. (38). The efficiency of the path tracing procedure depends on the choice of the basis set. For the IST basis sets used in *Spinach*, there are always at least two (*e.g.* `mary_test_1.m`) and sometimes over a hundred (*e.g.* `hsqc_test_1.m`) independent subspaces, depending on the calculation type and spin interactions present.

ZTE, path tracing and symmetry factorization can be applied sequentially (symmetry, then ZTE, then path tracing) using a wrapper function that returns the projectors into the resulting set of disconnected minimal subspaces

```
P=reduce(spin_system,L,rho);
```

where `P` is a cell array of projectors that may be used as prescribed by Eq. (38). Individual algorithms may be disabled by setting the `sys.disable` switch before the call to `create` (Table S1 in the Supplementary Information). Unless they are specifically disabled, trajectory-level pruning algorithms are applied automatically and transparently every time a call is made to the time evolution function.

3.10. Time evolution

Two functions are available for moving the state vector forward in time under a given Liouvillian. A single propagation step Δt under a Liouvillian $\hat{\hat{L}}$ (*e.g.* a hard pulse with $\hat{\hat{L}} = \hat{\hat{S}}_{\mathrm{X}}$ and $\Delta t = \pi/2$) is best performed with Krylov algorithm using

```
rho=step(spin_system,L,rho,delta_t);
```

where `rho` is the state vector. For a single propagation step, the Krylov algorithm is faster than matrix exponentiation because it computes $\exp(-i\hat{\hat{L}}\Delta t)\hat{\rho}$ directly from $\hat{\hat{L}}$ and $\hat{\rho}$ using only matrix–vector multiplications (details are given in Section 2.5). The `step` function is also optimal in the case when the Liouvillian is time-dependent (shaped pulses, *etc.*) and many steps need to be taken with different values of $\hat{\hat{L}}$.

For long trajectories under time-independent Liouvillians, the kernel provides a fairly sophisticated wrapper function (`evolution.m`) that, which automatically takes advantage of the trajectory-level state space reduction functions described in Section 3.9.

- The final state vector after a period of evolution can be requested by:

```
rho=evolution(spin_system,...
L,[],rho,timestep,nsteps,'final');
```

A horizontal stack of state vectors can be supplied, in which case every vector in the stack is taken forward by the same time interval.

- System trajectory for a given number of steps under a given Liouvillian can be requested by:

```
rho_stack=evolution(spin_system,L,[],rho,...
          timestep,nsteps,'trajectory');
```

The output variable `rho-stack` contains the state vectors for every point in the trajectory concatenated horizontally (a total of `nsteps+1` columns). The first point is the initial state. A specific situation often encountered in NMR and ESR spectroscopy is a 180° "refocusing" pulse in the middle of an incremented evolution period. The stack of final states for each increment in the duration of the evolution period can be requested using the *'refocused_trajectory'* option:

```
rho_stack=evolution(spin_system,L,[],rho,...
          timestep,nsteps,'refocused_trajectory',R);
```

where `R` is the cell array of generators of the refocusing pulse (*e.g.* $\pi\hat{\hat{S}}_X$ in the case of a 180° pulse on the *X*-axis), which will be applied in the order of appearance in `R`.

- The dynamics of the observable corresponding to a given state vector can be requested by:

```
observable=evolution(spin_system,L,coil,...
           rho,timestep,nsteps,'observable');
```

where `coil` (called so for obvious reasons) is the state vector corresponding to the detection state (*e.g.* $\hat{S}_+$ in the case of quadrature detection). If a horizontal stack of state vectors is supplied as the initial condition, a vertical stack of observable traces is returned, individual lines corresponding to the observable traces starting from each of the initial conditions supplied.

The best illustration of the practical use of these options is the source code of the 2D NMR and DNP pulse sequences in the user-land (Section 4 and `exp` directory of the *Spinach* distribution).

3.11. Optimal control waveform optimization

Spinach implements the GRAPE (gradient ascent pulse engineering) procedure [84,85] for optimal control based waveform optimization. The calling syntax is:

```
[objective,gradient]=grape(spin_system,drift,...
waveform,time_step,nsteps,...
starting_state,target_state);
```

in which `drift` is the static "drift" Liouvillian, `controls` is a cell array of control superoperators, `waveform` is a row vector giving the control amplitudes at each step (for multiple control operators, the waveforms should be concatenated horizontally in the order in which the control operators are listed in `controls`), `starting_state` is the initial condition state vector and `target_state` is the destination state vector, which should be populated to the maximum possible extent by the optimized waveform. The `objective` output is the real part of the scalar product between the final state and the destination state, and the `gradient` output is the gradient of the objective function with respect to the waveform parameters, evaluated at the current value of `waveform`.

The utilization of the resulting objective function and gradient is a subject of active current research; from the kernel perspective, this is a user-land problem. Examples of using *Matlab*'s built-in L-BFGS [86] optimization module (a part of the *Optimization Toolbox*) are given in the `examples` directory. It should be noted that the first-order approximation to the GRAPE gradient is often unsuitable for BFGS runs [87] due to its limited accuracy, an exact gradient option is provided for this purpose. This option is chosen by default, unless the user manually specifies the level of accuracy in the `sys.tols.grape_gradient` subfield when calling `create` (Table S3 in the Supplementary Information).

4. *Spinach* user-land

The user-land (Fig. 1) is a diverse and growing collection of functions written to facilitate the real-world calculations using the kernel as the simulation back-end. The functions described in this section are by no means the complete list – they only serve to illustrate the kind of functionality that the *Spinach* kernel makes it easy to implement. The user-land interfaces the kernel to the outside world and much of it is left to the discretion of the end user – *Spinach* is designed to be a re-usable library of basic building blocks rather than a collection of pre-made simulation scripts.

4.1. Standard NMR pulse sequences

At the time of writing, *Spinach* user-land provides COSY, COSY-α, DQF-COSY, HETCOR, HMQC, HSQC, NOESY and pulse-acquire NMR sequences [45,88,89], the chief limitation on the number of sequences being the time available to the authors for implementing them. The sequence code resides in the `exp` directory and runs comfortably with 40+ spin systems on a desktop workstation. Over 30 examples are provided in the `examples` directory. The calling syntax is common for all sequences:

```
fid=sequence(spin_system,parameters);
```

where `fid` is the free induction decay without apodization and `parameters` is a structure with the fields listed in Table S5 in the Supplementary Information. Not all sub-fields of `parameters` are applicable to every sequence, the applicable parameters are listed in the headers of the sequence files.

By default, the pulse sequence functions would assemble their own Liouvillians. This can be overridden and a user-specified Liouvillian supplied by using the following syntax:

```
fid=sequence(spin_system,parameters,L);
```

where `L` is the Liouvillian in question.

The pulse sequences that are intended, by design, for liquid-state NMR, ignore the `sys.regime` switch (Table S1 in the Supplementary Information) and always perform a liquid-state simulation. An exception is `pulse_acquire`, which would honor the switch and run a single-crystal or a powder-average simulation depending on the value of `sys.regime`.

An important advantage of simulating an NMR experiment over running it is the ability to perform coherence selection algebraically, thus avoiding the expensive phase cycles and gradients entirely – all pulse sequence functions supplied with *Spinach* make full use of this fact.

4.2. Shaped pulses

Shaped pulse functions, including a function performing multiple shaped pulses in parallel on different channels, are available, along with a library of waveforms (Gaussian [90], Q3 and Q5 [91], REBURP [92], *etc.*, courtesy of Prof. K. Pervushin). Matrix exponentiation is avoided during the propagation under a shaped pulse – Krylov propagation (Section 2.5) [53] is used instead. A waveform can be read from the library with the following command:

```
[amplitudes,phases,integral]=...
read_wave(filename,npoints)
```

where `npoints` is the number of points to which the waveform is to be resampled (harmonic resampling) and `integral` is the integral (Simpson quadrature [62]) of the radiofrequency amplitude across the pulse. Shaped pulses have no universal calibration criteria – the `integral` variable may be used to generate the initial estimate, but ultimately the pulse amplitude must be calibrated on a test system "the spectrometer way" – by scaling the overall multiplier. For a given waveform, a single shaped pulse is executed with:

```
rho=shaped_pulse(spin_system,L,rho,...
spins,offset,phases,amplitudes,duration);
```

where `rho` is a state vector, `L` is the drift Liouvillian, `offset` is a shift in the rotating frame frequency to be performed for the duration of the pulse, `spins` is a string indicating the nucleus on which the pulse is to be performed, `phases` is a row vector giving the phase of each waveform point, `amplitudes` is a row vector giving the amplitude of each waveform point and `duration` is the pulse duration. Multiple shaped pulses can be performed with:

```
rho=sim_pulse(spin_system,L,rho,...
spins,offsets,phases,amplitudes,durations)
```

where `rho` is a state vector, `L` is the drift Liouvillian, `offsets` is a row vector of rotating frame frequency shifts to be performed at each channel for the duration of the pulse, `phases` is a cell array of row vectors giving the phase of each waveform point at each channel, `amplitudes` is a cell array of row vectors giving the amplitude of each waveform point at each channel, `durations` is a row vector of pulse durations at each channel and `spins` is a cell array of strings specifying the channels on which each pulse is to be performed. The `sim_pulse` function resamples the grids of the input waveforms onto a common time grid, generates a unified Liouvillian stack on the resulting common grid and applies the resulting propagators to the state vector. The pulse centres are aligned in the resulting combined waveform.

4.3. Spin Chemistry experiments

Spinach provides isotropic and anisotropic MARY [93,94] experiments as well as a template singlet state dynamics experiment, which may be used to build a large variety of case-specific Spin Chemistry simulations. An option is available to use Redfield relaxation theory (Section 2.7) where the role of $\hat{\hat{L}}_0$ is played by the isotropic hyperfine interaction. Accurate quantum mechanical models of radical pair recombination (Section 3.6) are also implemented. The calling syntax for MARY is:

```
M=mary(spin_system,fields,kinetics);
```

where `fields` is a vector of magnetic fields in Tesla, `kinetics` is a vector of exponential model recombination rates in Hz and `M` is a matrix of singlet yields evaluated for each combination of magnetic field and recombination rate. The `MARY-1` basis (Table S2 in the Supplementary Information) is specifically designed for MARY experiments.

The anisotropic MARY syntax is:

```
M=maryan(spin_system,field,kinetics);
```

where `field` is the magnetic field in Tesla and `kinetics` is the exponential model recombination rate. The function evaluates the full orientation profile of the singlet yield on a 10,242-point (default) icosahedral spherical grid and returns the array of singlet yields as a function of spherical angles (there are three columns in the output matrix `M`: θ, φ and singlet yield).

The template magnetic field effect experiment returning the singlet state dynamics curve is invoked as:

```
ssd=mfe(spin_system,field,timestep,nsteps);
```

where `field` is the magnetic field in Tesla, `timestep` is the simulation time step in seconds, `nsteps` is the number of time steps and `ssd` is a vector giving the singlet state population at every step in the simulation. The relaxation and kinetics parameters for `mfe` are set in the call to `create` (Table S1 in the Supplementary Information).

4.4. Standard ESR experiments

The following pulsed ESR experiments are available in *Spinach* user-land at the time of writing: pulse-acquire, ESEEM [95] and Mims ENDOR [96]. Electron shells of any multiplicity can be requested (Table S1 in the Supplementary Information). Redfield relaxation theory (Section 2.7) and kinetics superoperators (Section 3.6) are available.

Two specialized basis sets (Table S2 in the Supplementary Information) are available for ESR calculations. The *ESR-1* basis set is intended for simple high-field ESR experiments with isotropic hyperfine couplings and includes the complete state space for all electrons (which are pulsed and observed), but only the identity state and $\hat{T}_{l0}$ for the nuclei (conservation law screening, see Section 2.11). *ESR-2* is intended for most high-field ESR simulations (possibly involving relaxation, chemical kinetics and anisotropic couplings) and includes a complete basis set on all electrons and anisotropically coupled nuclei, but only $\hat{E}$ and $\hat{T}_{l0}$ for the isotropically coupled nuclei. This, when used together with the non-interacting subspace separation procedure (Section 2.10), is effectively a generalization of current wisdom about the state spaces encountered in high-field ESR dynamics [73].

4.5. Spin system import filters

At the time of writing, *Spinach* provides import filters for *Gaussian03* [74] magnetic property calculation logs and *Simpson* [7,22] `*.in` files. The parse command

```
properties=g03_parse(filename);
```

will read the following properties if they are found in the Gaussian log: atomic coordinates in the standard orientation, SCF energy, hyperfine coupling tensors, *g*-tensor, shielding tensors and scalar coupling constants. The properties are placed into the sub-fields of the `properties` structure. All coupling tensors are symmetrized automatically – we are not convinced that the interaction tensor asymmetries [56,57] that come out of Gaussian calculations are physically real. The selection of properties for import into *Spinach* is handled by the `g03_to_spinach` function:

```
[sys,inter]=g03_to_spinach(properties,...spins,
           references,options);
```

which generates the `sys` and `inter` fields that may be passed to `create` (Section 3.2). The `spins` option is a cell array listing the spins and isotopes to be imported, *e.g.*

```
{{'H','1H'},{'N','14N'},{'E','E3'}}
```

instructs the function to import hydrogens as ^{1}H, nitrogens as ^{14}N and assume the electron shell to be in a triplet state. Because Gaussian03 reports chemical shielding relative to the bare nucleus in vacuum, a set of references is required to transform it into IUPAC conventions [97]. The `references` variable is a row vector listing such reference chemical shielding (*e.g.* ^{1}H shielding computed for TMS at the same level of theory) for all spins in the order of their appearance in the `spins` variable. Case-specific import criteria can be supplied using the options variable (Table S6 in the Supplementary Information). Because the size of the restricted basis set often depends on the interaction topology, these options should be considered carefully.

The `spinsys` section of a *Simpson* [7,22] input file can be read and converted into *Spinach* `sys` and `inter` structures using:

```
[sys,inter]=simpson2spinach(filename);
```

this function is a part of a two-way interface to *Simpson*, which is presently under development.

4.6. Apodization and plotting

Unless relaxation theory, powder averaging or chemical kinetics are enabled in the simulation, the FIDs produced by *Spinach* do not decay and require manual apodization. Several basic methods can be applied by calling

```
fid=apodization(fid,window_type,parameters);
```

Because of the way the Fourier transform is implemented in *Matlab*, the first point (1D) or the corner point (2D) of the free induction decay must be halved prior to performing the FT; this correctly places the baseline at the zero level and is silently done when `apodization` is called.

The 1D NMR/ESR plotter function in *Spinach* is unremarkable; it simply calls *Matlab*'s plotting routine and labels the axes:

```
plot_1d(spin_system,spectrum,parameters);
```

accepting the same `parameters` structure as was used to invoke the pulse sequence (Section 4.1). The 2D contour plotting function is considerably more sophisticated and deserves some attention. The full call:

```
contour_plot(spin_system,spectrum,parameters,...
ncont,delta,k,ncol,m);
```

where the `parameters` is the same structure as was used to run the pulse sequence, the number of contours (`ncont`), the minimum and maximum contour elevation as a fraction of highest absolute intensity in the spectrum (`delta`), the curvature parameter for the contour spacing function (`k=1` corresponds to linear contour spacing, `k>1` bends the spacing to increase contour density towards the baseline), number of colors in the colormap (`ncol`) and the curvature of the colormap (`m=1` corresponds to a linear color ramp into the red for positive contours and into the blue for negative contours, `m>1` makes colors saturate faster further away from the baseline). The basic three-parameter call:

```
contour_plot(spin_system,spectrum,parameters);
```

assumes the (generally reasonable) values of `ncont=20`, `delta=[0.02 1.0]`, `k=2`, `ncol=256`, `m=6`. The baseline color is automatically set to 10% gray to prevent it from blending with the white background.

4.7. Rotations

Few things in elementary physics look as easy and are as deadly as three-dimensional rotations – Leonard Euler's original error of selecting an invalid parameterization for SO(3) is so profound and has such far-reaching consequences [98] that the internal policy of the *Spinach* kernel specifically forbids any interaction specification other than the 3×3 matrix and any orientation specification other than the Wigner matrix. All other conventions are converted at the kernel entry point and all internal functions are written accordingly.

Still, the need for conversion between rotational conventions does occasionally arise in practice, and *Spinach* provides, without warranty, the following functions:

- `S=euler2dcm(angles);`

takes Euler angles (Varshalovich B convention) and returns a directional cosine matrix (DCM), which is well defined for $S^{-1}DS$ transformations that recover the 3×3 matrix for tensors supplied as eigenvalues and Euler angles.

- `W=euler2wigner(angles);`

takes Euler angles (Varshalovich B convention) and returns a second-rank Wigner function matrix $W_{mk} = \mathfrak{D}^{(2)}_{mk}(\alpha,\beta,\gamma)$, which is sorted in descending order with respect to both indices.

- `W=dcm2wigner(S);`

converts a directional cosine matrix into Wigner matrix. It should be noted that matrices coming out of the interaction tensor diagonalization procedure are often a reflection away from the DCM, because of the randomness associated with eigenvector phase.

- `[alpha,beta,gamma]=dcm2euler(S);`

converts a directional cosine matrix into Euler angles. This is not a well-defined procedure from the Lie algebraic point of view [98], and it should be avoided if possible. The above mentioned issue of eigenvector matrix not always being equal to DCM also applies.

- `[alpha,beta,gamma]=quat2euler(q);`

converts a quaternion rotation specification into Euler angles. All the above-listed health warnings associated with Euler angles apply.

5. Examples

The `examples` directory of the *Spinach* distribution contains over 50 well-commented simulation scripts covering most of the program functionality. Expert users would probably find it helpful to examine the source code of the pulse sequence simulation functions in the `exp` directory. A representative collection of screenshots is given in Fig. 4 and a very detailed walk-through of several example codes is given in the Supplementary Information. These include an accurate DQF-COSY spectrum of rotanone (22 spins with a sparse network of scalar couplings), a pulsed ESR spectrum of the TEMPO radical and the Redfield relaxation superoperator for a simple model system. The kinetics module neatly reproduces the Anderson statistical theory line shapes [99] (Fig. S5 in the Supplementary Information) and even the "impossible" double-quantum coherence peaks to the solvent [100] described by Warren et al. are easily simulated (Fig. S6).

The kernel of *Spinach* is flexible, efficient and well commented – there is no reason for an advanced user to stay constrained to the functions already provided in the user-land. A good illustration of kernel calls being used in a practical context is the HMQC sequence code (`hmqc.m` in the `exp` directory). We will give a commented walkthrough below.

After printing diagnostic messages to the console, `hmqc.m` proceeds to request the Liouvillian (including relaxation and kinetics superoperators) from the kernel:

```
H=h_superop(spin_system);
R=r_superop(spin_system);
K=k_superop(spin_system);
L=H+1i*R+1i*K;
```

After some obvious steps (offsets are applied and time steps calculated), the program requests the control operators, to be used for pulses later:

```
Lp_H=operator(spin_system,'L+',...
              parameters.spins_f2);
Lm_H=operator(spin_system,'L-',...
              parameters.spins_f2);
Lp_X=operator(spin_system,'L+',...
              parameters.spins_f1);
Lm_X=operator(spin_system,'L-',...
              parameters.spins_f1);
Lx_H=(Lp_H+Lm_H)/2; Ly_H=(Lp_H-Lm_H)/2i;
Lx_X=(Lp_X+Lm_X)/2; Ly_X=(Lp_X-Lm_X)/2i;
```

The initial state and the quadrature detection state are requested in a similar way:

```
rho=state(spin_system,'Lz',parameters.spins_f2);
coil=state(spin_system,'L+',parameters.spins_f2);
```

When the sequence starts, a pulse is applied on the F2 nuclei:

```
rho=step(spin_system,Lx_H,rho,pi/2);
```

The coherence transfer evolution period is then run (all trajectory-level state space restriction techniques are applied transparently inside the `evolution` function, and diagnostic messages are printed to the console):

```
rho=evolution(spin_system,L,[],rho,delta,1,
    'final');
```

A phase-cycled second pulse is applied on the F1 nuclei:

```
rho=step(spin_system,Lx_X,rho,pi/2)-
    step(spin_system,Lx_X,rho,-pi/2)-...
    1i*step(spin_system,Ly_X,rho,pi/2)+...
    1i*step(spin_system,Ly_X,rho,-pi/2);
```

The F1 evolution period of HMQC has a refocusing 180° pulse in the middle, and the second call to `evolution` informs the kernel about that fact:

```
rho_stack=evolution(spin_system,L,[],rho,...
timestep_f1, parameters.npoints_f1-1,'...
refocused_trajectory',Lx_H*pi);
```

The output of this call contains a horizontal *stack* of state vectors, each specific vector corresponding to a time point in the F1 evolution period. The next pulse and the coherence transfer evolution period are applied to the entire stack:

```
rho_stack=step(spin_system,Lx_X,rho_stack,pi/2);
rho_stack=evolution(spin_system,L,[],...rho_stack,
          delta,1,'final');
```

Finally, detection is performed under algebraic decoupling:

```
[L,rho_stack]=decouple(spin_system,L,...rho_stack,
parameters.spins_f1);
fid=evolution(spin_system,L,coil,rho_stack,...
timestep_f2, parameters.npoints_f2-1,...
'observable');
```

The resulting FID is returned to the user. The syntax above is, arguably, as simple and general as a spin dynamics simulation API can get – all the complexities of relaxation theory, operator representations in truncated basis sets, trajectory-level state space restriction, symmetry factorization and propagation are handled internally by the kernel, and the user is only required to spell out the pulse sequence diagram.

6. Kernel programming and developer notes

Given the diversity of the user-land (NMR, ESR, Dynamic Nuclear Polarization, Optical Magnetometry, Spin Chemistry, Optimal Control, Quantum Computing, *etc.*), it is essential that the kernel maintains a high level of generality and flexibility. This is difficult and for this reason the authors reserve the authority over kernel architecture and recommend placing all new functions into user-land – whenever possible and appropriate, we will generalize and promote them into the kernel. Researchers willing to modify the kernel are, of course, free to do so but the official *Spinach* development repository will only accept kernel changes that are universal and preserve the above noted generality.

From the overall development perspective, *Spinach* is an open-source package and we do not feel protective about releasing the source code – we view our primary contribution to be in the realm of ideas and algorithms rather than coding. Accordingly, it is not our intention to compete, in particular, with the established solids NMR and ESR codes, such as *SIMPSON* [7,22], *SPINEVOLUTION* [5], *mPackages* [21] and *EasySpin* [25] – we would much rather assist the authors of those packages in the adaptation of the state space restriction techniques reviewed above. *Spinach* has been written with readability and portability in mind – almost every line is commented and all variables have descriptive names. Complete maps of the data structure are provided in Figs. 1 and 2 and Figs. S1–

S4. The graphical user interface, when it arrives, will also support all major packages.

While we do plan to eventually outsource, *EasySpin* style [25], the inner loops to a compiled language and parallelize the propagation and powder average stages (as recently pioneered in *SIMPSON* [7]), *Spinach* will continue to use *Matlab* for the foreseeable future. We have two primary reasons for this choice:

A. Code acceleration achieved by employing more efficient algorithms [26,28,29,63] is dramatically greater than the effect of rewriting (or linking to) the existing algorithms in a platform-specific compiled language. Accordingly, porting to *C* or *Fortran* is not, at the moment, a high priority.

B. The cost of brain time (£25+per hour) is considerably greater than the cost of CPU time (£0.06per hour) – *Matlab*'s compact and elegant syntax saves the former at the (slight) expense of the latter and is therefore optimal for our purposes.

The current ongoing implementation effort includes spatial degrees of freedom, stochastic Liouville equation, solid state relaxation theories and an extended library of standard pulse sequences.

7. Conclusion

This paper should be viewed as an introduction to Version 1.0 of *Spinach*. The development will, of course, continue to the best of our knowledge and ability. In its present version *Spinach* offers a mechanism for the Liouville space simulation of spin systems that were previously too large, and significantly accelerates simulations of previously tractable systems. The most general case of a spin system with a dense coupling network evolving for an infinite time remains unsolved, but this situation is rarely encountered in practice. Liouville space simulations (including symmetry, relaxation and chemical kinetics) of most liquid-state NMR experiments on 40+ spin systems can now be performed without effort on a desktop workstation. Much progress has also been made with improving the efficiency of ESR, solid state NMR and Spin Chemistry simulations.

Acknowledgments

The authors are very grateful to Mark Butler, Jean-Nicolas Dumez, Lyndon Emsley, Jack Freed, Steffen Glaser, Jeff Harmer, Paul Hodgkinson, Alexej Jerschow, Walter Köckenberger, Malcolm Levitt, Niels Chr. Nielsen, Konstantin Pervushin, Chris Rodgers, Thomas Schulte-Herbrüggen and Zdenek Tosner for many useful discussions and feedback on the early program prototypes. The project is funded by the EPSRC (EP/F065205/1, EP/H003789/1) and supported by the Oxford e-Research Centre. The Lebedev grids included with *Spinach* have been kindly provided by the Levitt group (University of Southampton). The shaped pulse waveforms are courtesy of Konstantin Pervushin (Nanyang Technological University).

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.jmr.2010.11.008.

References

[1] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G.L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A.P. Seitsonen, A. Smogunov, P. Umari, R.M. Wentzcovitch, QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials, J. Phys.: Condens. Matter 21 (2009) 395502.
[2] M.F. Guest, I.J. Bush, H.J.J. Van Dam, P. Sherwood, J.M.H. Thomas, J.H. Van Lenthe, R.W.A. Havenith, J. Kendrick, The GAMESS-UK electronic structure package: algorithms, developments and applications, Mol. Phys. 103 (2005) 719–747.
[3] G.T. Velde, F.M. Bickelhaupt, E.J. Baerends, C.F. Guerra, S.J.A. Van Gisbergen, J.G. Snijders, T. Ziegler, Chemistry with ADF, J. Comput. Chem. 22 (2001) 931–967.
[4] U. Becker, D. Ganyushin, A. Hansen, D.G. Liakos, C. Kollmar, S. Kossmann, T. Petrenko, C. Reimann, C. Riplinger, K. Sivalingam, E. Valeev, B. Wezisla, F. Wennmohs, F. Neese, ORCA 2.8.0, 2010.
[5] M. Veshtort, R.G. Griffin, SPINEVOLUTION: a powerful tool for the simulation of solid and liquid state NMR experiments, J. Magn. Reson. 178 (2006) 248–282.
[6] R.S. Dumont, S. Jain, A. Bain, Simulation of many-spin system dynamics via sparse matrix methodology, J. Chem. Phys. 106 (1997) 5928–5936.
[7] Z. Tosner, R. Andersen, N.C. Nielsen, T. Vosegaard, SIMPSON 3.0.1, 2010.
[8] M. Ndong, H. Tal-Ezer, R. Kosloff, C.P. Koch, A Chebychev propagator for inhomogeneous Schrodinger equations, J. Chem. Phys. 130 (2009) 124108.
[9] M. Ndong, H. Tal-Ezer, R. Kosloff, C.P. Koch, A Chebychev propagator with iterative time ordering for explicitly time-dependent Hamiltonians, J. Chem. Phys. 132 (2010) 064105.
[10] H. Tal-Ezer, R. Kosloff, An accurate and efficient scheme for propagating the time-dependent Schrodinger-equation, J. Chem. Phys. 81 (1984) 3967–3971.
[11] P. Hodgkinson, D. Sakellariou, L. Emsley, Simulation of extended periodic systems of nuclear spins, Chem. Phys. Lett. 326 (2000) 515–522.
[12] M. Hohwy, H. Bildsoe, H.J. Jakobsen, N.C. Nielsen, Efficient spectral simulations in NMR of rotating solids. The gamma-COMPUTE algorithm, J. Magn. Reson. 136 (1999) 6–14.
[13] J.I. Musher, Equivalence of nuclear spins, J. Chem. Phys. 46 (1967) 1537.
[14] J.I. Musher, Magnetic equivalence of nuclear spins, J. Chem. Phys. 47 (1967) 5460.
[15] S.M. Nokhrin, D.F. Howarth, J.A. Weil, Magnetic resonance in systems with equivalent spin-1/2 nuclides. Part 2: energy values and spin states, J. Magn. Reson. 193 (2008) 1–9.
[16] S.M. Nokhrin, J.A. Weil, D.F. Howarth, Magnetic resonance in systems with equivalent spin-1/2 nuclides. Part 1, J. Magn. Reson. 174 (2005) 209–218.
[17] G. Moro, J.H. Freed, Calculation of ESR-spectra and related Fokker–Planck forms by the use of the Lanczos-algorithm, J. Chem. Phys. 74 (1981) 3757–3773.
[18] K.V. Vasavada, D.J. Schneider, J.H. Freed, Calculation of electron-spin-resonance spectra and related Fokker–Planck forms by the use of the Lanczos-algorithm. 2. Criteria for truncation of basis-sets and recursive steps utilizing conjugate gradients, J. Chem. Phys. 86 (1987) 647–661.
[19] A. Jerschow, MathNMR: spin and spatial tensor manipulations in mathematica, J. Magn. Reson. 176 (2005) 7–14.
[20] I. Kuprov, N. Wagner-Rundell, P.J. Hore, Bloch–Redfield–Wangsness theory engine implementation using symbolic processing software, J. Magn. Reson. 184 (2007) 196–206.
[21] M. Levitt, A. Brinkmann, mPackages 5.21, 2010.
[22] M. Bak, J.T. Rasmussen, N.C. Nielsen, SIMPSON: a general simulation program for solid-state NMR spectroscopy, J. Magn. Reson. 147 (2000) 296–330.
[23] M. Helgstrand, P. Allard, QSim, a program for NMR simulations, J. Biomol. NMR 30 (2004) 71–80.
[24] S.A. Smith, T.O. Levante, B.H. Meier, R.R. Ernst, Computer-simulations in magnetic-resonance – an object-oriented programming approach, J. Magn. Reson. 106 (1994) 75–105.
[25] S. Stoll, A. Schweiger, EasySpin, a comprehensive software package for spectral simulation and analysis in EPR, J. Magn. Reson. 178 (2006) 42–55.
[26] H.J. Hogben, P.J. Hore, I. Kuprov, Strategies for state space restriction in densely coupled spin systems with applications to spin chemistry, J. Chem. Phys. 132 (2010) 174101.
[27] I. Kuprov, C.T. Rodgers, Derivatives of spin dynamics simulations, J. Chem. Phys. 131 (2009) 234108.
[28] I. Kuprov, Polynomially scaling spin dynamics II: further state-space compression using Krylov subspace techniques and zero track elimination, J. Magn. Reson. 195 (2008) 45–51.
[29] I. Kuprov, N. Wagner-Rundell, P.J. Hore, Polynomially scaling spin dynamics simulation algorithm based on adaptive state-space restriction, J. Magn. Reson. 189 (2007) 241–250.
[30] M.C. Butler, J.N. Dumez, L. Emsley, Dynamics of large nuclear-spin systems from low-order correlations in Liouville space, Chem. Phys. Lett. 477 (2009) 377–381.
[31] J.N. Dumez, M.C. Butler, E. Salager, B. Elena-Herrmann, L. Emsley, Ab initio simulation of proton spin diffusion, Phys. Chem. Chem. Phys. 12 (2010) 9172–9175.
[32] D.M. Brink, G.R. Satchler, Angular Momentum, third ed., Clarendon, 1993.
[33] J.H. Freed, G.K. Fraenkel, Theory of linewidths in electron spin resonance spectra, J. Chem. Phys. 39 (1963) 326.
[34] B.C. Sanctuary, F.P. Temme, Multipole NMR 13. Multispin interactions and symmetry in Liouville space, Mol. Phys. 55 (1985) 1049–1062.

[35] D.A. Varshalovich, A.N. Moskalev, V.K. Khersonskii, Quantum Theory of Angular Momentum, World Scientific, 1988.
[36] G. Campolieti, N. Lee, B.C. Sanctuary, Multipole NMR 12. Theory of spin decoupling, Mol. Phys. 55 (1985) 1033–1047.
[37] G. Campolieti, B.C. Sanctuary, H.B.R. Cole, Multipole theory of soft pulses in NMR of quadrupolar solids, J. Magn. Reson. 88 (1990) 457–472.
[38] B.C. Sanctuary, Multipole operators for an arbitrary number of spins, J. Chem. Phys. 64 (1976) 4352–4361.
[39] B.C. Sanctuary, Multipole NMR 3. Multiplet spin theory, Mol. Phys. 48 (1983) 1155–1176.
[40] B.C. Sanctuary, Multipole NMR .11. Scalar spin coupling, Mol. Phys. 55 (1985) 1017–1031.
[41] B.C. Sanctuary, Multipole NMR .10. Multispin, multiquantum, multilinear operator bases, J. Magn. Reson. 61 (1985) 116–129.
[42] B.C. Sanctuary, H.B.R. Cole, Multipole theory of composite pulses, J. Magn. Reson. 71 (1987) 106–115.
[43] B.C. Sanctuary, T.K. Halstead, P.A. Osment, Multipole NMR .4. Dynamics of single spins, Mol. Phys. 49 (1983) 753–784.
[44] B.C. Sanctuary, M.S. Krishnan, Rotating and principal-axis frames in multipole NMR, J. Magn. Reson. 69 (1986) 210–217.
[45] R.R. Ernst, G. Bodenhausen, A. Wokaun, Principles of Nuclear Magnetic Resonance in One and Two Dimensions, Clarendon, 1987.
[46] J.H. Kristensen, H. Bildsøe, H.J. Jakobsen, N.C. Nielsen, Application of Lie algebra to NMR spectroscopy, Progr. NMR Spectrosc. 34 (1999) 1–69.
[47] J. Bargon, J. Kandels, K. Woelk, Orthohydrogen and parahydrogen induced nuclear-spin polarization, Z. Phys. Chem. 180 (1993) 65–93.
[48] P.J. Hore, R.W. Broadhurst, Photo-CIDNP of biopolymers, Progr. NMR Spectrosc. 25 (1993) 345–402.
[49] I. Kuprov, M. Goez, P.A. Abbott, P.J. Hore, Design and performance of a microsecond time-resolved photo-chemically induced dynamic nuclear polarization add-on for a high-field nuclear magnetic resonance spectrometer, Rev. Sci. Instrum. 76 (2005) 084103.
[50] I. Kuprov, P.J. Hore, Uniform illumination of optically dense NMR samples, J. Magn. Reson. 171 (2004) 171–175.
[51] M.J. Duer, Introduction to Solid-State NMR Spectroscopy, Blackwell, 2004.
[52] G. Moro, J.H. Freed, Efficient computation of magnetic-resonance spectra and related correlation-functions from stochastic Liouville equations, J. Phys. Chem. 84 (1980) 2837–2840.
[53] R.B. Sidje, Expokit: a software package for computing matrix exponentials, ACM Trans. Math. Soft. 24 (1998) 130–156.
[54] C. Moler, C. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later, SIAM Rev. 45 (2003) 3–49.
[55] C. Moler, C. Vanloan, Nineteen dubious ways to compute exponential of a matrix, SIAM Rev. 20 (1978) 801–836.
[56] K.J. Harris, D.L. Bryce, R.E. Wasylishen, NMR line shapes from AB spin systems in solids – the role of antisymmetric spin–spin coupling, Can. J. Chem. 87 (2009) 1338–1351.
[57] S. Wi, L. Frydman, Quadrupolar-shielding cross-correlations in solid state nuclear magnetic resonance. Detecting antisymmetric components in chemical shift tensors, J. Chem. Phys. 116 (2002) 1551–1561.
[58] M. Mehring, V.A. Weberruss, Object-Oriented Magnetic Resonance: Classes and Objects, Calculations and Computations, Academic Press, 2001.
[59] M. Goldman, Formal theory of spin-lattice relaxation, J. Magn. Reson. 149 (2001) 160–187.
[60] A.G. Redfield, On the theory of relaxation processes, IBM J. Res. Dev. 1 (1957) 19–31.
[61] R.K. Wangsness, F. Bloch, The dynamical theory of nuclear induction, Phys. Rev. 89 (1953) 728–739.
[62] M. Abramowitz, I.A. Stegun, Handbook of Mathematical Functions: With Formulas Graphs and Mathematical Tables, Wiley, 1972.
[63] I. Kuprov, Diagonalization-free implementation of spin relaxation theory for large spin systems, ArXiV, submit/0155413.
[64] G. James, M. Liebeck, Representations and Characters of Groups, second ed., Cambridge University Press, Cambridge, 2001.
[65] H. Weyl, The Theory of Groups and Quantum Mechanics, Methuen, 1941.
[66] E.P. Wigner, J.J. Griffin, Group Theory and Its Application to the Quantum Mechanics of Atomic Spectra, Academic Press, 1959.
[67] P.L. Corio, Structure of High-Resolution NMR Spectra, Academic Press, 1966.
[68] N.C. Nielsen, T. Schulte-Herbruggen, O.W. Sorensen, Bounds on spin dynamics tightened by permutation symmetry – application to coherence transfer in I2s and I3s spin systems, Mol. Phys. 85 (1995) 1205–1216.
[69] J.A. Weil, D.F. Howarth, Magnetic resonance in systems with equivalent spin-1/2 nuclides. Part 3: Ket analysis and spectral intensities, J. Magn. Reson. 197 (2009) 28–35.
[70] A. Wokaun, R.R. Ernst, Use of multiple quantum transitions for relaxation studies in coupled spin systems, Mol. Phys. 36 (1978) 317–341.
[71] H.N. Gabow, R.E. Tarjan, A linear-time algorithm for a special case of disjoint set union, J. Comput. Syst. Sci. 30 (1985) 209–221.
[72] R. Tarjan, Depth-first search and linear graph algorithms, SIAM J. Comput. 1 (1972) 146–160.
[73] S. Stoll, R.D. Britt, General and efficient simulation of pulse EPR spectra, Phys. Chem. Chem. Phys. 11 (2009) 6614–6625.
[74] M.J. Frisch, G.W. Trucks, H.B. Schlegel, G.E. Scuseria, M.A. Robb, J.R. Cheeseman, J.J.A. Montgomery, T. Vreven, K.N. Kudin, J.C. Burant, J.M. Millam, S.S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G.A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J.E. Knox, H.P. Hratchian, J.B. Cross, C. Adamo, J. Jaramillo, R. Gomperts, R.E. Stratmann, O. Yazyev, A.J. Austin, R. Cammi, C. Pomelli, J.W. Ochterski, P.Y. Ayala, K. Morokuma, G.A. Voth, P. Salvador, J.J. Dannenberg, V.G. Zakrzewski, S. Dapprich, A.D. Daniels, M.C. Strain, O. Farkas, D.K. Malick, A.D. Rabuck, K. Raghavachari, J.B. Foresman, J.V. Ortiz, Q. Cui, A.G. Baboul, S. Clifford, J. Cioslowski, B.B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R.L. Martin, D.J. Fox, T. Keith, M.A. Al-Laham, C.Y. Peng, A. Nanayakkara, M. Challacombe, P.M.W. Gill, B. Johnson, W. Chen, M.W. Wong, C. Gonzalez, J.A. Pople, Gaussian 03, Revision E.02, 2009.
[75] M. Eden, M.H. Levitt, Computation of orientational averages in solid-state NMR by Gaussian spherical quadrature, J. Magn. Reson. 132 (1998) 220–239.
[76] M. Goldman, Interference effects in the relaxation of a pair of unlike spin-1/2 nuclei, J. Magn. Reson. 60 (1984) 437–452.
[77] A. Kumar, P.K. Madhu, Cross-correlations in multispin relaxation, Conc. Magn. Reson. 8 (1996) 139–160.
[78] H. Desvaux, R. Kummerle, J. Kowalewski, C. Luchinat, I. Bertini, Direct measurement of dynamic frequency shift induced by cross-correlations in N-15-enriched proteins, Phys. Chem. Chem. Phys. 5 (2004) 959–965.
[79] L. Werbelow, R.E. London, Dynamic frequency shift, Conc. Magn. Reson. 8 (1996) 325–338.
[80] M.H. Levitt, L. Di Bari, Steady state in magnetic resonance pulse experiments, Phys. Rev. Lett. 69 (1992) 3124.
[81] R. Haberkorn, Density matrix description of spin-selective radical pair reactions, Mol. Phys. 32 (1976) 1491–1493.
[82] K.L. Ivanov, M.V. Petrova, N.N. Lukzen, K. Maeda, Consistent treatment of spin-selective recombination of a radical pair confirms the Haberkorn approach, J. Phys. Chem. A 114 (2010) 9447–9455.
[83] J.A. Jones, P.J. Hore, Spin-selective reactions of radical pairs act as quantum measurements, Chem. Phys. Lett. 488 (2010) 90–93.
[84] Z. Tosner, T. Vosegaard, C. Kehlet, N. Khaneja, S.J. Glaser, N.C. Nielsen, Optimal control in NMR spectroscopy: numerical implementation in SIMPSON, J. Magn. Reson. 197 (2009) 120–134.
[85] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbruggen, S.J. Glaser, Optimal control of coupled spin dynamics: design of NMR pulse sequences by gradient ascent algorithms, J. Magn. Reson. 172 (2005) 296–305.
[86] C.Y. Zhu, R.H. Byrd, P.H. Lu, J. Nocedal, Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization, ACM Trans. Math. Soft. 23 (1997) 550–560.
[87] D. Byatt, I.D. Coope, C.J. Price, Performance of various BFGS implementations with limited precision second-order information, Anziam J. 45 (2004) 511–522.
[88] T.D.W. Claridge, High-Resolution NMR Techniques in Organic Chemistry, second ed., Elsevier, 2009.
[89] M.H. Levitt, Spin Dynamics: Basics of Nuclear Magnetic Resonance, second ed., Wiley, 2008.
[90] C. Bauer, R. Freeman, T. Frenkiel, J. Keeler, A.J. Shaka, Gaussian pulses, J. Magn. Reson. 58 (1984) 442–457.
[91] L. Emsley, G. Bodenhausen, Optimization of shaped selective pulses for NMR using a quaternion description of their overall propagators, J. Magn. Reson. 97 (1992) 135–148.
[92] H. Geen, R. Freeman, Band-selective radiofrequency pulses, J. Magn. Reson. 93 (1991) 93–141.
[93] C.R. Timmel, F. Cintolesi, B. Brocklehurst, P.J. Hore, Model calculations of magnetic field effects on the recombination reactions of radicals with anisotropic hyperfine interactions, Chem. Phys. Lett. 334 (2001) 387–395.
[94] C.R. Timmel, U. Till, B. Brocklehurst, K.A. McLauchlan, P.J. Hore, Effects of weak magnetic fields on free radical recombination reactions, Mol. Phys. 95 (1998) 71–89.
[95] W.B. Mims, Envelope modulation in spin-echo experiments, Phys. Rev. B 5 (1972). 2409–&.
[96] W.B. Mims, Endor spectroscopy by Fourier transformation of the electron-spin echo envelope, Abstr. Pap. Am. Chem. Soc. 180 (1980) 89.
[97] R.K. Harris, Conventions for tensor quantities used in NMR NQR and ESR, Solid State NMR 10 (1998) 177–178.
[98] M. Siemens, J. Hancock, D. Siminovitch, Beyond Euler angles: exploiting the angle-axis parametrization in a multipole expansion of the rotation operator, Solid State NMR 31 (2007) 35–54.
[99] P.W. Anderson, A mathematical model for the narrowing of spectral lines by exchange or motion, J. Phys. Soc. Jpn. 9 (1954) 316.
[100] W.S. Warren, W. Richter, A.H. Andreotti, B.T. Farmer, Generation of impossible cross-peaks between bulk water and biomolecules in solution NMR, Science 262 (1993) 2005–2009.