Large-scale magnetic resonance simulations: A tutorial

Maria Grazia Concilio

Abstract
Computational modeling is becoming an essential tool in magnetic resonance to design and optimize experiments, test the performance of theoretical models, and interpret experimental data. Recent theoretical research and software development made possible simulations of large spin systems, for example, proteins with thousands of spins, in reasonable time. In the last few years, the Fokker–Planck formalism also re-emerged due to its ability to handle spatial dynamics. The purpose of this tutorial is to describe advantages and disadvantages of the most common formalisms, the latest developments and strategies to improve the computational efficiency, and to guide users in the setting up of a simulation using the Spinach software.

1 | INTRODUCTION

A simulation software may be seen as a tool complementary to the spectrometer that grants a better understanding of the dynamical properties of the system, as well as for the optimization of experiments. Although the theories describing magnetic resonance experiments have been available since the 1960s, computations were inhibited by the lack of computer power and the exponential complexity scaling. Simulations involving more than 10 spins were not feasible. Calculations of this scale now take just a few seconds due to improved algorithms, an increase in computing power, and the simplicity of modern programming languages. The choice of the theoretical approach depends on the field of application. For example, the Bloch–Torrey equations\(^1,2\) can be used for simulations of relaxation, diffusion, and flow, on isolated spin-1/2 nuclei. The density operator formalism, reliant on the Liouville–von Neumann equation\(^3\), can used in simulations of experiments on both single systems and ensembles\(^4,5\); and allows the treatment of kinetics and sophisticated relaxation models, for example, the Redfield relaxation.\(^6\) Models based on the Fokker–Planck formalism\(^7\) are used in problems where spin dynamics coexists with classical dynamics (e.g., diffusion, flow, and sample rotation). Another time domain approach is the product operator formalism, which is used for calculating the evolution under free precession, chemical shift, and radio frequency (RF) pulses of the density operator (described in Section 3) in systems of weakly coupled spins.\(^8,9\) Other methods use analytical descriptions,\(^10–12\) but it is always convenient to use numerical simulations in the case of large spin systems and powders with orientational effects or spatial dynamics. The Spinach software\(^13\) supports different formalisms in a single package, and due to the implementation of tensor structured objects\(^14,15\) and polynomially scaling algorithms,\(^16,17\) it enables quantum mechanical simulations previously considered unfeasible.
2 | CLASSICAL VECTOR MODEL BASED ON BLOCH EQUATIONS

The dynamics of isolated spins can be described in terms of the motion of a classical magnetization vector using the Bloch equations.1,18–21 Considering an ensemble of spins with quantum number equal to $\frac{1}{2}$, in the presence of an external magnetic field, $B_0$, along the Z direction; each spin occupies one of two distinct energy levels, aligned either parallel (the low energy, $\alpha$ state) or anti-parallel (the high energy, $\beta$ state) to the magnetic field. At the thermal equilibrium, there is slight excess of spins in the low-energy state that leads to net magnetization. Bloch equations connect the applied magnetic field vector, $B = (B_x, B_y, B_z)$, to the bulk nuclear magnetization, $M = (M_x, M_y, M_z)$. In the laboratory frame of reference, the equation of motion describing the time dynamics of the magnetization vector is:

$$M(t) = \gamma M(t) \times B(t) + R[M(t) - M_0],$$

where $R = \begin{pmatrix} -1/T_2 & 0 & 0 \\ 0 & -1/T_2 & 0 \\ 0 & 0 & -1/T_1 \end{pmatrix}$, (1)

where $\gamma$ is the gyromagnetic ratio, $R$ is the relaxation matrix, $T_1$ and $T_2$ are the longitudinal and transverse relaxation times, $M_0$ is the magnetization vector at the thermal equilibrium, and $B(t) = B_0 + B_r(t)$ is the external magnetic field vector composed by a static part $B_0$ and an RF field $B_r(t)$ that is time dependent. The $B_r(t)$ is applied in the form of a linearly oscillating field with amplitude $B_1$ that can be decomposed into two counter-rotating components with frequency $\pm \omega_r$. Only the component rotating in the same sense as the spins is retained:

$$B_{r\ell}(t) = 2B_1 \left[ \cos(\omega_r t + \varphi) e_x + \sin(\omega_r t + \varphi) e_y \right],$$

where $\varphi$ is the phase acquired by the RF field and $e_x$ and $e_y$ are unit vectors in the x and y directions. The time dependence of Equation 2 can be eliminated through a transformation from the laboratory frame to a new frame, called rotating frame, where most of the simulations are performed (described in Section 3.4). In the rotating frame, the three components of the effective magnetic field vector correspond to:

$$B_{r\ell}^x = B_1 \cos\varphi, B_{r\ell}^y = B_1 \sin\varphi, B_{r\ell}^z = B_0 + \omega_r t / \gamma = -\Omega / \gamma,$$

where the superscript $\ell$ indicates rotating frame, $\Omega = \omega_0 - \omega_{r\ell}$ is the effective precession frequency in the rotating frame, $\omega_0 = -\gamma B_0$ is the Larmor frequency in the laboratory frame. Notice that the time dependence has disappeared from Equation 3, but the x and y components might reacquire time dependence for pulses with variable phase and amplitude. In the rotating frame, the equation of motion can be written as:

$$M_{r\ell}^x(t) = \gamma \left[ M_{r\ell}^x B_{r\ell}^z - M_{r\ell}^z B_{r\ell}^x \right] - M_{r\ell}^x / T_2,$$

$$M_{r\ell}^y(t) = \gamma \left[ M_{r\ell}^y B_{r\ell}^z - M_{r\ell}^z B_{r\ell}^y \right] - M_{r\ell}^y / T_2,$$

$$M_{r\ell}^z(t) = \gamma \left[ M_{r\ell}^z B_{r\ell}^x - M_{r\ell}^x B_{r\ell}^z \right] - (M_{r\ell}^z - M_0) / T_1.$$ (4)

The free induction decay (FID) can then be written in terms of complex transverse magnetization according to:

$$M_r(t) = M_x(t) + iM_y(t),$$

If pulse field gradients are present, the Larmor frequency of the spins becomes position dependent, and the field vector can be written as:

$$B(r, t) = (0, 0, r \cdot G(t) + B_0)^T,$$

where $r$ is the displacement vector and $G(t) = (G_x(t), G_y(t), G_z(t))$ is the time-dependent gradient. The effect of molecular self-diffusion can be included in Equation 1 in a generalized model proposed by Torrey in 1956. The complex transverse magnetization in Equation 5 becomes a function of both of $r$ and $t$; the resulting Bloch–Torrey equation can be written in terms of transport of transverse magnetization (assuming $G_x$ and $G_y$ equal to zero), in the rotating frame:

$$\frac{\partial M_r}{\partial t} = -i \omega_0 M_r - \frac{M_r}{T_2} + i \gamma r \cdot G(t) M_r + \nabla \cdot (D \nabla M_r),$$

where $D$ is the diffusion tensor and the symbol $\nabla$ represents the gradient. Further generalizations were made by Stejskal and Tanner in 1965 to incorporate anisotropic diffusion, restricted diffusion and flow; and by Henkelman et al. in 1993 to describe chemical exchange. Bloch–Torrey equations can be used to describe magnetic resonance experiments involving isolated spins, for example, Fourier spectroscopy, $T_1$ measurements, and composite pulses. They are the pillar of several packages for magnetic resonance imaging (MRI) simulations and can often be used to describe intravoxel dephasing effects due to RF pulses, field inhomogeneities, spin echoes, and stimulated echo sequences. Bloch–Torrey equations cannot be used to simulate pulse sequences that involve high-spin orders including multiple quantum coherence.
and coherence transfer processes. In Equations 1 and 7, there are no terms to describe the coupling between spins, these are included in the density operator formalism described in Section 3. In the context of MRI, the Bloch–Torrey equations fail to describe quantum mechanical effects (e.g., J-coupling evolutions) coexisting with spatial dynamics (e.g., diffusion, flow, and sample rotation). These can be simulated using the Fokker–Plank formalism described in Section 4.

3 | DENSITY OPERATOR FORMALISM

3.1 | Density operator

The state of a spin system is represented by a state function or wave function. In the Dirac notation, the kth state function can be written as ket, |k⟩, and its transpose as a bra, ⟨k|. A state function can be expanded in terms of orthonormal basis sets, {φk}, k = 1, 2, ..., n. The definition is given in Section A.2 in the supporting information:

$$|\psi\rangle = \sum_{k=1}^{H} c_k|\phi_k\rangle,$$

where c_k are potentially time-dependent complex numbers, called expansion coefficients, and H is the overall dimension of vector space. A vector space where a norm and a scalar product (Equations A.1 and A.2 in the supporting information) are defined is called the Hilbert space. For a system of N spins, the dimensionality of the Hilbert space is:

$$H = \prod_{n=1}^{N} (2S_n + 1),$$

where S is the quantum number of the spin n. The state of an ensemble of spin systems is defined by the density operator, \( \hat{\rho} \), operators are specified with a single hat, ∇. The density operator is given by the direct product (described in Section A.3 in the supporting information) of a wave function and its conjugate transpose:

$$\hat{\rho} = |\psi\rangle\langle\psi| = \sum_{k} \sum_{j} c_k c_j^* |\phi_k\rangle \langle \phi_j|,$$

The physical meaning of the density operator can be understood from the expression of its matrix elements—the diagonal elements represent the populations of the states, and the off-diagonal elements represent the correlation coefficients. In Spinach, the density operator defining the state of a single spin or a spin system is built using irreducible spherical tensor operators (described in Section 3.3.2) and can be specified by calling the state.m function. For example, an initial state \( \rho_0 \) corresponding to longitudinal magnetization, represented by the operator \( \hat{L}_z \), and a detection state \( \rho_{coil} \), corresponding to transverse magnetization, represented by the operator \( \hat{L}_x \), can be specified as:

```matlab
rho0 = state (spin_system,'Lz','all');
```

and

```matlab
rho_coil = state (spin_system,'Lx','all');
```

where the flag all represents the sum of a specified state on all the spins in the system. In the case of a spin system in the thermal equilibrium, where the energy levels are populated according to the Boltzmann law, the density operator at a given temperature T in Kelvin is given by:

$$\hat{\rho}_{eq} = \frac{1}{\text{Tr} \left( \exp \left( -\frac{H}{k_B T} \right) \right)} \exp \left( -\frac{H}{k_B T} \right),$$

where \( k_B \) is the Boltzmann constant, the symbol Tr is the trace of the matrix (corresponding to the sum of the diagonal elements of the matrix), and H is the spin Hamiltonian (defined in Section 3.3). A simulations where the initial condition is the thermal equilibrium can be set by calling the equilibrium.m function:

```matlab
rho_eq = equilibrium (spin_system,H0,Q,[\alpha \beta \gamma]),
```

where the spin_system object contains the spin system information (described in Section 3.2.1), \( H_0 \) is the isotropic part of the spin Hamiltonian, Q is the irreducible component of the anisotropic part (described in Section 3.3), and \( \alpha, \beta, \gamma \) are Euler angles defining the orientation of the system in specified powder grid. In the case of liquid state simulations, where the anisotropic part of the Hamiltonian is averaged out, the input parameters Q and \( [\alpha \beta \gamma] \) do not need to be provided.

3.2 | Spin interactions

Most simulations require the couplings between spins to be considered, as their time dependence can provide valuable information about the system. Physical systems
are composed of many interactions; but from an algebraic perspective, spin interactions can only be of three types: linear, bilinear, and quadratic. Linear interactions arise from the interaction between spins and the external magnetic field $B_0$ or a RF field (Equation 2); these are called Zeeman interactions. Bilinear interactions arise from homonuclear and heteronuclear couplings between spins, for example, $J$-coupling, hyperfine, dipolar, and exchange couplings. Finally, quadratic interactions, for example the nuclear quadrupolar interaction (NQI), that arise from the coupling between a spin and “itself”. The total spin Hamiltonian can be built summing together individual interaction Hamiltonians calculated from the scalar product of spin vectors and an interaction matrix, for example, as:

$$\hat{H} = \hat{H}_Z + \hat{H}_J + \hat{H}_Q = \hat{I} \cdot \hat{A} \cdot \hat{B} + \hat{I} \cdot \hat{A} \cdot \hat{L} + \hat{I} \cdot \hat{A} \cdot \hat{L},$$

(12)

where $\hat{H}_Z$ is the Zeeman Hamiltonian, $\hat{H}_J$ is the $J$-coupling Hamiltonian, and $\hat{H}_Q$ is the quadrupolar Hamiltonian; $\hat{I}$ and $\hat{L}$ denote spin operator vectors corresponding to the three Cartesian projections of the spin angular momentum operators in the space; $\hat{B}$ is the magnetic induction vector, and $\hat{A}$ represents an interaction tensor (a $3 \times 3$ matrix representing the orientation of the system in the space), that corresponds to the chemical shift tensor, hyperfine coupling tensor, and the quadrupolar coupling tensor.

### 3.2.1 Setting up of spin interactions

Most simulation problems require the couplings between spins to be considered, as their time dependence can provide valuable information about the system. The structure array $\text{sys}$ specifies the spin system and instrument configuration, whereas the structure $\text{inter}$ specifies the interactions present in the system. For example, a spin system of two protons can be initialized as:

```python
sys.isotopes={'1H','1H'};
```

Spin–field interactions require the external magnetic field strength to be supplied in units of Tesla:

```python
sys.magnet=14.1;
```

The full chemical shift tensors of each spin may be supplied as matrices with units of parts per million:

```python
inter.zeeman.matrix[1]=[14.89 34.35 0.00 34.35 145.74 0.00 0.00 0.00 150.51];
```

or as scalars in calculations that do not require the anisotropic part of the chemical shift interaction:

```python
inter.zeeman.scalar={5.0 0.0};
```

Bilinear interactions can be specified providing full coupling tensors as matrices in Hertz, for example:

```python
inter.coupling.matrix[1,2]=[1.2356 0.0000 0.6322 0.0000 1.1266 0.0000 0.6322 0.0000 8.2230]*1e6;
```

where the indices $\{1,2\}$ represents a pair of spins specified in $\text{sys.isotopes}$. The anisotropy of the $J$-coupling between two nuclear spins is usually ignored, and the corresponding orientational average is known as scalar coupling, which can be provided in Hertz:

```python
inter.coupling.scalar[1,2]=10;
```

If Cartesian coordinates of every spin (in Angstroms) are provided, Spinach calculates interspin dipolar interactions internally:

```python
inter.coordinates = [[0.00.00.00][0.00.06.00]];
```

If the spin system contains nuclei with spin quantum numbers greater than $\frac{1}{2}$, the total spin Hamiltonian includes also quadratic terms arising from the interaction between a nuclear electric quadrupole moment and an electric field gradient (EFG). In Spinach, the quadrupole Hamiltonian in its principal axis frame (where the interaction tensor is diagonal) corresponds to:

$$\hat{H}_Q = \frac{e^2 q Q}{4I(2I-1)\hbar} \begin{pmatrix} -1 + \eta & -1 - \eta \\ -1 - \eta & -1 + \eta \end{pmatrix},$$

(13)

where $C_Q = \frac{e^2 q Q}{4I(2I-1)\hbar}$ is the is the quadrupolar coupling constant in Hertz; and $\eta$ is the quadrupolar tensor asymmetry parameter (calculated from the eigenvalues of the electric field gradient tensor), $I$ spin quantum number.
of the quadrupolar nucleus and \( h \) is the Planck constant. The nuclear quadrupolar couplings matrix according to Equation 13 can be obtained by calling the `eeqq2nqi.m` function, for example as:

\[
\text{inter.coupling.matrix}[1,1]=\text{eeqq2nqi}([C_\gamma,\eta,I,\text{eulers}]);
\]

where indices \([1,1]\) indicates an interaction between a spin and itself; \( \text{eulers} \) is a vector containing three euler angles in radians, giving the orientation of the principal axis frame relative to the laboratory frame. The data structures \( \text{sys} \) and \( \text{inter} \) should be then supplied to the `create.m` function that builds the spin system object containing the spin system information— isotopes, spin coordinates, coupling, and so forth. The command is:

\[
\text{spin_system} = \text{create} ([\text{sys}, \text{inter}]);
\]

This tutorial provides an example set in the supporting information, where interactions are set in accordance to the above description. Other options for the setting up of spin interactions can be found in Kuprov.\(^25\)

### 3.3 Spin Hamiltonian

Molecules rotate in solution and spin in magic angle spinning (MAS) experiments (described in Section 4). Each interaction Hamiltonian contains a rotational invariant part (“isotropic”) that is time independent and a part that changes with molecular orientation (“anisotropic”) that is time dependent. The rotational modulation of a spin interaction tensor using Cartesian operators corresponds to:

\[
\hat{R} \left[ \vec{S} \cdot \vec{A} \cdot \vec{L} \right] = \vec{S} \cdot (\text{RAR}^{-1}) \cdot \vec{L}, \quad (14)
\]

where \( \hat{R} \) is a rotation matrix and \( \text{RAR}^{-1} \) is its inverse. The rotation matrix can be parameterized in various ways (using angle-axis parameterization, quaternions parameterization, and directional cosine matrix parameterization). The result of Equation 14 corresponds to a complicated trigonometric expression. A more practical representation for the description of three-dimensional rotations is based on irreducible spherical tensor operators (ISTs), indicated as \( \hat{T}_{l,m} \), where \( l \) is the rank and \( m = -l, -l+1, \ldots, l \) represents its \( 2l+1 \) components. Considering that the anisotropic part of all the spin interactions transforms virtually in accordance to a second-rank irreducible representation of the rotation group, the rotational modulation of these interactions may be treated in a general way using the IST operators.\(^26\)–\(^30\) A general rotation \( \hat{R}(\alpha,\beta,\gamma) \) with Euler angles \( \alpha, \beta, \gamma \) can be expressed as:

\[
\hat{R}(\alpha,\beta,\gamma) \hat{T}_{l,m} = \sum_{m' = -l}^{l} \hat{T}_{lm} \hat{\Xi}^{(l)}_{m',m}(\alpha,\beta,\gamma), \quad (15)
\]

where \( \hat{\Xi}^{(l)}_{m',m}(\alpha,\beta,\gamma) \) are elements of the Wigner rotation matrix of the order \( l \).\(^31\)\(^32\) The total Hamiltonian can then be expanded as a linear combination of IST operators corresponding to spin interactions according to:

\[
\hat{\mathcal{H}} = \hat{\mathcal{H}}_{\text{iso}} + \sum_{l = 1}^{2} \sum_{m,k = -l}^{l} \hat{\Xi}^{(l)}_{km} \hat{Q}^{(l)}_{km}, \quad (16)
\]

where \( \hat{\mathcal{H}}_{\text{iso}} \) represents the isotropic part of the Hamiltonian, \( \hat{\Xi}^{(l)}_{km} \) are the Wigner functions specifying the molecular orientation, and \( \hat{Q}^{(l)}_{km} \) contains rotational basis operators and represents the anisotropic part of the Hamiltonian. For a spin system of two spins, where one of them is quadrupolar, the rotational basis operators correspond to:

\[
\begin{align*}
\hat{Q}^{(2)}_{km} &= \sum_{L} \Phi_{m}(B,L) \hat{T}^{(2)}_{l}(B,L) + \sum_{S} \Phi_{m}(B,S) \hat{T}^{(2)}_{l}(B,S) \\
&= \sum_{L,S} \Phi_{m}(L,S) \hat{T}^{(2)}_{l}(L,S) + \sum_{S} \Phi_{m}(S,S) \hat{T}^{(2)}_{l}(S,S),
\end{align*}
\]

where \( B \) indicates the static magnetic field, \( L \) and \( S \) indicate the spins, \( \Phi_{m}(B,L) \), \( \Phi_{m}(B,S) \), \( \Phi_{m}(L,S) \) and \( \Phi_{m}(S,S) \) are irreducible spherical tensor parameters for linear, bilinear, and quadratic interactions, respectively. The isotropic part of the spin Hamiltonian in Equation 16 corresponds to zero-rank terms that are invariant under rotation. The anisotropic part of the spin interactions corresponds to second-rank terms that are the only terms that have to be rotated in practical calculations. First-rank terms in Equation 16 are very small and are often ignored. In Spinach, the spin Hamiltonian operator in the Hilbert space and superoperator in the Liouville
space (described in Section 3.5), along with the rotational decomposition, can be obtained by calling the `hamiltonian.m` function:

\[
[H_0, Q] = \text{hamiltonian} \{\text{spin\_system}\};
\]

where \(H_0\) represents the isotropic part of the spin Hamiltonian and \(Q\) is the irreducible basis of the anisotropic part. The spin Hamiltonian in Equation 16 can then be built with the following command:

\[
H = H_0 + \text{orientation}\{Q, [\alpha \beta \gamma]\};
\]

The `orientation.m` function builds the anisotropic part for a specified set of Euler angles \([\alpha \beta \gamma]\).

### 3.3.1 The IST parameters

A total Hamiltonian in Equation 16 can be then written in terms of all the interaction tensor elements. In Equation 17, zero-rank IST parameters are given by:

\[
\Phi_{0,0} = -\frac{a_{zz} + a_{xx} + a_{yy}}{\sqrt{3}},
\]

second-rank IST parameters are given by:

\[
\begin{align*}
\Phi_{2,0} &= \frac{2a_{zz} - (a_{xx} + a_{yy})}{\sqrt{6}}, \\
\Phi_{2,\pm 1} &= \mp \frac{(a_{zz} + a_{xx}) \mp i(a_{xx} - a_{yy})}{2}, \\
\Phi_{2,\pm 2} &= \frac{(a_{xx} - a_{yy}) \mp i(a_{yy} + a_{xx})}{2},
\end{align*}
\]

Second-rank IST parameters are given by:

\[
T_{2,0}(B, L) = 0, T_{2,\pm 1}(B, L) = \mp \frac{1}{2} \hat{L}_\pm, T_{2,0}(B, L) = + \sqrt{2} \hat{L}_z.
\]  

Second-rank ITS operators, describing bilinear and quadratic interactions between spins, correspond to combinations of two-spin product operator basis given by:

\[
\begin{align*}
\hat{T}_{2,2}(L, S) &= + \frac{1}{2} \hat{S}_z \hat{L}_+ + \hat{S}_- \hat{L}_+, \\
\hat{T}_{2,2}(L, S) &= + \frac{1}{3} (\hat{S}_z \hat{L}_+ - \frac{1}{2} (\hat{S}_+ \hat{L}_- + \hat{S}_- \hat{L}_+)), \\
\hat{T}_{2,-2}(L, S) &= + \frac{1}{2} \hat{S}_- \hat{L}_- + \hat{T}_{2,-1} = + \frac{1}{2} (\hat{S}_z \hat{L}_- + \hat{S}_- \hat{L}_z),
\end{align*}
\]

Ladder operators appearing in Equations 20–21 are given by \(\hat{L}_\pm = \hat{L}_x \pm i\hat{L}_y\), where the spin operator basis \(\hat{L}_x\), \(\hat{L}_y\), and \(\hat{L}_z\) that can be built from the direct product of Pauli matrices. Single-spin IST operator \(\hat{T}_{k,m}\) with rank \(k\) and multiplicity \(m\) can be obtained calling the `irr_sph_ten.m` function:

\[
T = \text{irr\_sph\_ten}(m, k);
\]

For example, considering a system with two \(\frac{1}{2}\)-spins, the spin operator for each spin can be obtained from:

\[
\begin{align*}
\hat{L}_x &= (\hat{\sigma}_x \otimes \hat{E}), \hat{L}_y &= (\hat{\sigma}_y \otimes \hat{E}), \hat{L}_z &= (\hat{\sigma}_z \otimes \hat{E}) \\
\hat{S}_x &= (\hat{E} \otimes \hat{\sigma}_x), \hat{S}_y &= (\hat{E} \otimes \hat{\sigma}_y), \hat{S}_z &= (\hat{E} \otimes \hat{\sigma}_z),
\end{align*}
\]

where \(\hat{\sigma}_x\), \(\hat{\sigma}_y\), and \(\hat{\sigma}_z\) represent the Pauli matrices and \(\hat{E}\) is the unit matrix, corresponding to:

\[
\begin{align*}
\hat{E} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \sigma_x &= \begin{pmatrix} 0 & 1/2 \\ 1/2 & 0 \end{pmatrix}, \\
\sigma_y &= \begin{pmatrix} 0 & -i/2 \\ i/2 & 0 \end{pmatrix}, \sigma_z &= \begin{pmatrix} 1/2 & 0 \\ 0 & -1/2 \end{pmatrix},
\end{align*}
\]

In Spinach, the `operator.m` function that builds an operator in the Hilbert space and a commutation superoperator in the Liouville space (described in Section 3.5). For example, spin operators corresponding to the sum of \(\hat{L}_x\) and \(\hat{L}_y\) on all the protons of a spin system are computed as:

\[
L_p = \text{operator}\{\text{spin\_system}, 'Lx', 'Ly'\};
\]

The pulse operators can then be built:
Lx = \langle H_p + H_p' \rangle / 2; \text{ and } Ly = \langle H_p - H_p' \rangle / 12; \\

Pulse operators must be supplied in any Spinach calculations, so see the example set for details.

### 3.4 Laboratory and rotating frame Hamiltonians

The spin Hamiltonian given in Equation 16 contains all the couplings present in the system. Considering a spin system of two spins L and S connected by J-coupling, the laboratory frame Hamiltonian used in liquid and solid state simulations corresponds to:

\[ \hat{H}_{\text{lab}} = \frac{1}{3} \text{Tr}(Z_L) \hat{L}_Z + \frac{1}{3} \text{Tr}(Z_S) \hat{S}_Z, \]  

(25)

where \( Z_L \) is the Zeeman interaction matrix of the spin L and \( Z_S \) is the Zeeman interaction matrix of the spin S. The term \( \hat{H}_{\text{J,iso}} \) is the isotropic coupling term corresponding to:

\[ \hat{H}_{\text{J,iso}} = \frac{1}{3} \text{Tr}(J_{LS})(L_X \hat{S}_X + L_Y \hat{S}_Y + L_Z \hat{S}_Z), \]  

(26)

where \( J_{LS} \) is the J-coupling interaction matrix. The laboratory frame Hamiltonian contains also Zeeman carrier frequencies given by the product of the isotropic free-particle gyromagnetic ratio and the magnetic field, corresponding to \( \omega_{L,0} = -\gamma_L B_0 \) and \( \omega_{S,0} = -\gamma_S B_0 \) for the spin L and S, respectively. The Zeeman carrier frequency does not carry any relevant information and dominates the cost of the calculation. For example, in a magnetic field of 14 Tesla, the Zeeman Larmor frequency is of the order of MHz and the time step is of the order of microseconds; processes occurring on time scales of seconds would require millions of points, leading to extremely long wall-clock times. Thus, it should be removed from the Hamiltonian. The procedure to remove Zeeman terms from the spin Hamiltonian is called interaction representation transformation or rotating frame transformations that yield the secular Hamiltonian (the derivation is omitted):

\[ \hat{H}_{\text{rot,frame}} = \hat{H}_{\text{Z,iso}} + \hat{H}_{\text{J,iso}} + \frac{\Phi_{2,0}(B,L) \hat{T}_{2,0}(B,L)}{Z \text{ part of linear couplings}} + \frac{\Phi_{2,0}(L,S) \hat{T}_{2,0}(L,S)}{\text{Secular part of bilinear couplings}} + \frac{\Phi_{2,0}(S,S) \hat{T}_{2,0}(S,S)}{\text{Secular part of quadratic couplings}} \]  

(27)

In rotating frame simulations, Zeeman carrier frequencies are subtracted from the coefficient in front of the \( \hat{L}_Z \) and \( \hat{S}_Z \) operator in the isotropic Zeeman terms:

\[ \hat{H}_{\text{Z,iso}} = [\text{Tr}(Z_L) - \omega_{L,0}] \hat{L}_Z + [\text{Tr}(Z_S) - \omega_{S,0}] \hat{S}_Z, \]  

(28)

In the case of couplings between spins of different types, the IST operator in the secular part of bilinear couplings, in Equation 27, simply corresponds to \( \hat{T}_{2,0}(L,S) = \sqrt{\frac{2}{\gamma_S \gamma_L}} \hat{S}_Z \hat{L}_Z \). The latter is called weak coupling term. The secular Hamiltonian is used in NMR simulations performed at high-field (\( \geq 1 \) Tesla). Spinach also allows more complicated higher-order corrections to the rotating frame transformation. In rotating frame simulations, the offset with respect to the carrier Zeeman frequency can be specified as an input parameter:

\[ \text{parameters.offset} \]

and can be added to the Hamiltonian by calling the \text{frgoffset.m} function. In Spinach, laboratory frame or rotating frame Hamiltonians can be set automatically using the \text{assume.m} function that builds the spin Hamiltonian in accordance with Equations 24 and 27:

\[ \text{spin_system} = \text{assume}(\text{spin_system, 'assumption'}); \]

or using a kernel \text{context.m} function:

\[ \text{answer} = \text{context}(\text{spin_system, @pulse_sequence, parameters, 'assumption'}); \]

where the flag assumption can be set equal to labframe to build a laboratory frame Hamiltonian, or equal to NMR to build a secular Hamiltonian. The MATLAB command window will display the following outputs:
Summary of Zeeman tensors and their anisotropies (ppm for nuclei, g-tensor for electrons)

| Spin | 2S+1 Matrix | Tr/3 | norm[rank1] | norm[rank2] |
|------|-------------|------|-------------|-------------|
| 1    | +1.489000e+01 +1.435000e+01 +0.000000e+00 | +1.0371e+02 +0.0000e+00 +9.7293e+01 |
|      | +3.435000e+01 +1.457400e+02 +0.000000e+00 | +3.435000e+01 +1.505100e+02 |
|      | +0.000000e+00 +0.000000e+00 +1.505100e+02 |

| Spin | 2S+1 Matrix | Tr/3 | norm[rank1] | norm[rank2] |
|------|-------------|------|-------------|-------------|
| 2    | +3.075000e+01 +3.100000e+00 +0.000000e+00 | +2.1733e+01 +0.0000e+00 +9.9333e+00 |
|      | +1.310000e+00 +2.265000e+01 +0.000000e+00 | -5.659298e+03 |
|      | +0.000000e+00 +0.000000e+00 +1.180000e+01 |

Summary of the tensors and their anisotropies in Hertz:

| Spin A | Spin B | Matrix | Tr/3 | norm[rank1] | norm[rank2] |
|--------|--------|--------|------|-------------|-------------|
| 1      | 2      | +1.131244e+04 | -6.000931e+02 | +0.000000e+00 | +0.000000e+00 |
|        |        | -5.996165e+02 | -5.638144e+03 | +0.000000e+00 | +5.0000e+00 |
|        |        | +0.000000e+00 | +0.000000e+00 | -5.659298e+03 | +1.1329e+04 |

where $\text{Tr}/3$ is the isotropic part of the Hamiltonian and norm[rank1] and norm[rank2] represent norms of the first- and second-rank terms, respectively.

Context functions are responsible for running miscellaneous housekeeping (liquid state calculation, powder averages, spatial dynamics, etc.), creating spin operators or superoperators, kinetic and relaxation superoperators, and passing them to a pulse_sequence.m function. Available context functions are liquid.m (for simulation in liquid state), powder.m (for static simulations in solid state), imaging.m (for MRI and spatiotemporal NMR simulations), and singlerot.m (for single axis magic spin angle simulations). The pulse_sequence.m function must contain the following command:

```
answer=pulse_sequence(spin_system, parameters, H,R,K);
```

A spin Hamiltonian operator in the Hilbert space or a Liouvillian superoperator in the Liouville space (described in Section 3.5) can then be built with the following syntax:

```
L=H+1i*R+1i*K;
```

where

```
R=relaxation (spin_system, [α β γ]);
```

and

```
K=kinetics (spin_system);
```

where $H$ is the Hamiltonian, $R$ is the relaxation superoperator, and $K$ is the kinetic super operator that can be obtained by calling the functions relaxation.m and kinetics.m, respectively.

3.5 | Equations of motion

The density operator formalism is derived from the solution of the time-dependent Schrödinger equation. The equation of the motion for the density operator is called the Liouville–von Neumann equation:

$$
\frac{\partial}{\partial t} \hat{\rho}(t) = \frac{\partial}{\partial t} (\langle \psi(t) | \psi(t) \rangle ) = \left( \frac{\partial}{\partial t} | \psi(t) \rangle \right) \langle \psi(t) | + | \psi(t) \rangle \left( \frac{\partial}{\partial t} | \psi(t) \rangle \right) = -i \hat{H}(t) | \psi(t) \rangle \langle \psi(t) | + i | \psi(t) \rangle \langle \psi(t) | \hat{H}(t) = -i \hat{H}(t) \hat{\rho}(t) - \hat{\rho}(t) \hat{H}(t) = -i [\hat{H}(t), \hat{\rho}(t)],
$$

where $\hat{H}(t)$ is a time-dependent Hamiltonian and $| \psi(t) \rangle$ is a time-dependent wave function. The Liouville–von Neumann equation is the pillar of the most magnetic resonance simulation software and mostly used for large-scale liquid and solid-state simulations.

3.5.1 | Simulations in the Hilbert space

The Hilbert space can be used for frequency domain simulations (not described in this tutorial), for determining energy level diagrams and transition probabilities between energy levels, but they do not support time
domain process. In the Hilbert space, Equation 29 has a solution corresponding to:

$$\dot{\rho}(t) = \hat{U}(t,0)\rho(0)\hat{U}(t,0)^\dagger, \hat{U}(t,0) = \hat{T}\exp\left\{-i \int_0^t \hat{H}(t')dt'\right\},$$

(30)

where $\rho(0)$ is the initial density matrix, $\hat{U}(t)$ is the propagator from the time zero to the time $t$, and $\hat{T}$ is the Dyson time ordering operator that evaluates the exponential functions and reorders the Hamiltonians in a way that preserves the descending time sequence to ensure that the Hamiltonian commutes with itself at different time steps.\(^5\) In the rotating frame, the Hamiltonian is made time independent within finite segments of time. The density matrix propagated in time intervals under a time-independent Hamiltonian corresponds to

$$\dot{\rho}(t + \Delta t_1 + \cdots + \Delta t_k) = e^{-i\hat{H}_t \Delta t_1} \cdots e^{-i\hat{H}_t \Delta t_k} \dot{\rho}(t) e^{i\hat{H}_t \Delta t_1} \cdots e^{i\hat{H}_t \Delta t_k},$$

(31)

where $\Delta t$ is the time step and $e^{\pm i\hat{H}\Delta t}$ is a matrix exponential called propagator. For example, considering a spin system with two $\frac{1}{2}$-spins, the matrix dimension for operators, propagators, and states in the Hilbert space is equal to 4 by 4, Equation 31 has the following structure:

$$\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix} \cdot \begin{bmatrix} \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot \\ \cdot \end{bmatrix} \cdot \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}.$$

Equation 31 involves a double-sided product that is inconvenient in the case of multiple nested products, so a single-sided simplification can be made by moving into Liouville space.

### 3.5.2 Simulations in the Liouville space

Most of the simulations are performed in the Liouville space, because it supports sophisticated relaxation theories (e.g., the Redfield theory) and kinetics. The Liouville space is the adjoint representation of the Hilbert space. Adjoint operators (called superoperators) are denoted as $\hat{H} = ad_{\hat{H}} = [\hat{H},]^\dagger$.\(^5,34-36\).

$$\frac{\partial}{\partial t} |\rho(t)\rangle = -i\hat{H}[\rho(t)],$$

(32)

where $|\rho(t)\rangle$ is the state vector and $\hat{H}$ is Hamiltonian commutation superoperator that corresponds to a commutator with the Hamiltonian:

$$\hat{H}[\rho] = [\hat{H},\rho] = [\hat{E} \otimes \hat{H} - \hat{H}^T \otimes \hat{E}]|\rho\rangle,$$

(33)

where $\hat{H}$ is the spin Hamiltonian operator and $\hat{H}^T$ is its transpose, $\hat{E}$ is the unit matrix of the same size of the spin Hamiltonian. In the Liouville space, the elements of the density matrix are arranged in a column vector, called state vector. In Spinach, Hilbert space operators can be converted into Liouville space superoperators by calling the `hilb2liouv.m` function:

```matlab
L = hilb2liouv(H, 'comm');
```

where the flag `comm` indicates a commutation superoperator. In the Liouville space, the evolution can be expressed as:

$$|\dot{\rho}(t + \Delta t_1 + \cdots + \Delta t_k)\rangle = e^{-i\hat{E} \Delta t_k} \cdots e^{-i\hat{E} \Delta t_1} |\rho(t)\rangle,$$

(34)

The matrix dimension for propagators in the case of a spin system with two $\frac{1}{2}$-spins is equal to 16 by 16, and the dimension of the state vector is 16 by 1, Equation 34 corresponds to:

$$[|\rho(t)\rangle] = [(16 \times 16) \cdot |\rho(0)\rangle],$$

Although the matrix dimension is larger in the Liouville space, it allows polynomial scaling with the size of the system (described in Section 5). An example of setting up of simulation in Liouville space is shown below. Figure 1 depicts a simulation of a 1D NMR spectrum of a glycine system where the relaxation behavior is controlled by both the chemical shift anisotropy (CSA) and dipole–dipole (DD) coupling.

The DD-CSA interference causes the double members of the spectra to relax at different rates. Simulation parameters and the code (explained step by step) used to obtain Figure 1 are given in Example 1 in Section B.1 of the supporting information.

### 3.5.3 Setting up of the Liouville and Hilbert space

Spinach supports both the Hilbert and the Liouville space simulation formalisms. The formalism can be specified using `bas.formalism` structure. Simulations in the Hilbert space use Pauli matrices as basis sets and can be specified with the command:

```matlab
bas.formalism='zeeman-hilb';
```
Simulations in the Liouville space use IST operators as basis and can be specified with:

```matlab
bas.formalism='sphten-liouv';
```

Information about the formalism must be supplied to the `basis.m` function using the following command:

```matlab
spin_system=basis (spin_system,bas);
```

Spinach supports **complete and incomplete basis sets**, as described in Section 5.

### 3.5.4 Setting up of the propagation method

In Spinach, long-range propagations under a static Hamiltonian and Liouvillian in accordance to Equations 31 and 34, respectively, can be performed by calling the `evolution.m` function:

```matlab
rho_final=evolution (spin_system,L,coil,rho_zero,...
    time_step,nsteps,output,destination);
```

where the `rho_final` corresponds to the final state vector (or density matrix), depending on the options specified in the function call, `L` is a spin Hamiltonian or a Liouvillian, the `coil` (optional argument) represents the detection state, `rho_zero` is the initial state vector or density matrix to be propagated, `nsteps` represents the number of simulation steps characterized by duration given in `time_step`, and the output specifies the form of `rho_final` that can be a single state vector (or density matrix) or a simulation trajectory, the destination flag is an optional to be used for destination screening (described in Section 5.1.7). This function should be used for trajectory calculations, detection periods of time domain experiments, and computation of the time integrals of observables. In Spinach, the propagation can also be performed using the Taylor series:

\[
  e^{-i\Delta t} \rho = \sum_{n=0}^{\infty} \left( \frac{-i\Delta t}{n!} \right)^n \rho = \sum_{n=0}^{\infty} \left( \frac{-i\Delta t}{n!} \right)^n \left( L \left( \left( \left( \left( \left( \left( \left( \left( \left( \left( \left( \left( L \rho \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right),
\]

Equation 35 involves only matrix–vector multiplications and calculates the action by a matrix exponential on a vector without computing the matrix exponential. The scaling of the Taylor series is \(O(N^2)\), where \(N\) is the matrix dimension. This approach is particularly well suited for propagation under single hard pulses and time-dependent Liouvillians or Hamiltonians. The time propagation with the Taylor series can be performed by calling the `step.m` function with the following command:

```matlab
rho_final=step (spin_system,L,rho_zero,time_step);
```

where `L` can be a pulse operator (described in Section 3.3.2), a spin Hamiltonian operator or a Liouvillian and `time_step` is the time step to take. Considering that the Taylor series is polynomial, its convergence to the machine precision will depend on the product of the norm of the spin Hamiltonian operator...
(or the Liouvillian) and the time step. When the norm is smaller than one, the Taylor series converges in less than 20 steps, and when the norm is higher than one, the propagation is performed using scaling and squaring techniques\textsuperscript{37} that require matrix exponentialization. The matrix exponential is computed in \( n \) squaring steps that involve \( n \) matrix–matrix multiplications according to:

\[
\exp[-i\hat{L}\Delta t^n] = \exp[-i\hat{L}\Delta t]^{2^n} = \left(\left(\exp[-i\hat{L}\Delta t]^{2}\right)^{2} \ldots\right)^{2},
\]

\text{(36)}

The scaling of the matrix exponentialization is \( \mathcal{O}(N^3) \), where \( N \) is the matrix dimension. The propagation of a spin system with more than 200 spins becomes problematic. An alternative is the Krylov propagation that computes the product between the matrix exponential propagator and the density matrix directly, thus avoiding matrix exponentialization.\textsuperscript{38} In Spinach, the size of the Liouvillian is evaluated internally and the switch to Krylov propagation is made automatically in the case of a very large Liouvillians, like those found in protein spin system. The Krylov propagation requires less memory than the scaling and squaring technique but is slower and hence should only be used when exponential of the Liouvillian does not fit into memory, but the Liouvillian itself does. If memory is available, the switch to Kyrlov propagation can be disabled with the command:

\texttt{sys.disable=\{'krylov\'};}

The krylov propagation can be also used within a pulse sequence by calling the \texttt{krylov.m} function:

\texttt{answer=krylov(spin_system,L,coil,rho_zero, ... time_step,nsteps,output);}

This function uses the same input parameters as \texttt{evolution.m}.

### 3.6 Expectation value and trajectory analysis

In our simulations, we are often interested in the evolution of on observable property that can be extracted from the density operator. The term observable corresponds to properties of an individual molecule rather than ensembles. Operators corresponding to observables are indicated with the symbol \( \hat{O} \). The average outcome of many repeated experiments of an observable property of a quantum system is called the expectation property of a quantum system is called the expectation value and is indicated with the angular brackets \( \langle \cdot \rangle \):

\[
\langle \hat{O} \rangle = \text{Tr}\{\hat{O}\hat{\rho}\},
\]

\text{(37)}

The expectation value can be found by evaluating the trace of the product of an observable operator and the density operator. In practice, the expectation value related to an observable operator can be computed as:

\[
\text{Hx=real(Hx'*trajectory);};
\]

where \( \text{Hx} \) represents an observable spin operator and \( \text{trajectory} \) is a simulation trajectory obtained with the \texttt{evolution.m} function by setting the option with the flag \texttt{trajectory}. Considering a spin system of two coupled spins, \(^1\text{H}\) and \(^{13}\text{C}\), an example of evolution of observable single spin operators over a simulation trajectory is shown in Figure 2.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2.png}
\caption{Evolution of single spin operators as a function of the time during a spin lock}
\end{figure}

The simulation starts with transverse magnetization on the proton, then it is transferred to the carbon during the spin lock. Simulation parameters and the code (explained step by step) used to obtain Figure 2 are given in Example 2 in Section B.2 of the supporting information. In Spinach, a trajectory analysis can be also performed by calling the \texttt{trajan.m} function:

\texttt{trajan(spin_system,trajectory,property,time_axis);};

where \( \text{property} \) corresponds to an observable property of the system, such as a correlation order, coherence order, or total state space population.
4 | FOKKER–PLANK FORMALISM

In the Fokker–Planck picture, the state of the system is not represented by the density matrix $\rho(x, t)$ that describes the state of a specific particle at the point $x$ at the time $t$. It considers the probability $p(x, t)$ of having a particle in the point $x$ in the spin state $\rho$ at the time $t$ in a very high dimensional space (three dimensions with respect to the spatial coordinates and $4^n$ dimensions with respect to spin). This approach can be used in simulations where there is the simultaneous presence of quantum mechanical dynamics (i.e., spin–spin coupling, spin relaxation, and chemical reactions) and spatial dynamics (e.g., diffusion, flow, or sample rotation), for example, singlet and metabolomics imaging,\textsuperscript{39,40} pure-shift NMR,\textsuperscript{41} and ultrafast NMR.\textsuperscript{42} The Fokker–Planck equation can be derived from the continuity equation that states that the probability must be conserved and the local change in probability as a function of space and time is equal to the divergence of its flux, $j(x, t)$:

$$\frac{\partial}{\partial t} p(x, t) = -\nabla \cdot j(x, t). \tag{38}$$

where $\nabla$ is the gradient containing the spatial derivatives $\nabla = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right]^T$. For a given probability profile, the flux is given by the product of the probability and the velocity field:

$$j(x, t) = v(x, t) p(x, t). \tag{39}$$

Considering that the velocity field in space can be described by the classical equation of the motion and the velocity field in quantum mechanics can be described by the Liouville–von Neuman equation, it is possible to write separately the divergence with respect to the spatial $\nabla_x$ and spin $\nabla_{\rho}$ coordinates. The former is given by the velocity in the space $v_x(x, \rho, t)$ multiplied by the probability $p(x, \rho, t)$, whereas the latter is given by the velocity in the spin space $v_{\rho}(x, \rho, t)$ multiplied by the probability:

$$\frac{\partial p(x, \rho, t)}{\partial t} = -\nabla_x \cdot [v_x(x, \rho, t)p(x, \rho, t)] - \nabla_{\rho} \cdot [v_{\rho}(x, \rho, t)p(x, \rho, t)]. \tag{40}$$

The divergence represents the sum of all the derivatives with respect to all the coordinates:

$$\frac{\partial}{\partial t} p(x, t) = -\text{div} \left[ j(x, t) \right] = [\nabla^T \cdot v(x, t) + v^T(x, t) \cdot \nabla + \nabla^T \cdot D(x, t) \cdot \nabla] p(x, t). \tag{41}$$

The average of any property can be obtained from the integral over the entire configurational space of the product between that property and its probability. The ensemble average density matrix corresponds to:

$$\bar{\rho}(x, t) = \int \rho p(x, \rho, t) dV_{\rho}. \tag{42}$$

where $dV_{\rho}$ is the volume of the spin space. The equation of the motion can be obtained from the time derivative of the average density matrix and is obtained by differentiating Equation 42 with respect to the time and then using the expression for the derivative from Equation 40, to obtain:

$$\frac{\partial p(x, t)}{\partial t} = -iL(x, t)p(x, t) + M(x, t)p(x, t). \tag{43}$$

Equation 43 is the Fokker–Planck equation, in which $L(x, t)$ is the Liouvillian and $M(x, t)$ is the spatial dynamics generator obtainable from classical mechanics. Time evolution generators corresponding to infinitesimal translation on a spatial or a phase grid are represented by

$$\frac{\partial p(x, t)}{\partial t} = \left[ v(t) \frac{\partial}{\partial x} \right] p(x, t), \tag{44}$$

translational flow

$$\frac{\partial p(x, t)}{\partial t} = \left[ \omega(t) \frac{\partial}{\partial \rho} \right] p(x, t), \tag{45}$$

rotational flow

$$\frac{\partial p(x, t)}{\partial t} = \left[ D \frac{\partial^2}{\partial x^2} \right] p(x, t), \tag{46}$$

rotational diffusion

where $v(t)$ is the flow velocity, $\omega(t)$ is the angular velocity, and $D$ is the diffusion coefficient. In cases where $M(x, t)$ corresponds to the diffusion operator, Equation 43 is known as the stochastic Liouville equation (SLE), which was used for the first time in magnetic resonance in 1970 for simulations of EPR spectra.\textsuperscript{43–47} The SLE could describe the rotational diffusion at all magnetic field strengths and correlation times, going from extreme narrowing to solid limits, and has recently been applied to MAS simulations.\textsuperscript{48,49} The probability density can be moved forward in the time by applying an exponential with respect to a generator according to

$$p(t + a) = \exp \left[ a \frac{\partial}{\partial t} \right] p(t), \tag{47}$$

propagator

where $a$ corresponds to linear velocity in the case of flow or angular velocity in the case of phase increment, $p(t)$ is
the initial probability, and $\partial/\partial t$ corresponds to one of the derivatives shown in Equations 44–46. An operator of infinitesimal translation squared is a generator of infinitesimal diffusion, Equation 46. Its solution is equivalent to Green’s function of the heat equation describing the dynamics of a particle undertaking Brownian motion starting from $x_0$ at the time zero $t_0$ to the position $x$; the time $t$ can be described by the conditional probability:

$$p(x,t \mid x_0,t_0) = \frac{1}{\sqrt{4\pi D(t-t_0)}} \exp \left( -\frac{(x-x_0)^2}{4D(t-t_0)} \right).$$ (48)

In Spinach, spatial dynamics generators have a matrix representation. Flow and diffusion generators (Equations 44 and 46) are represented by central finite difference matrices, and spatial coordinates are discretized on a finite grid,\textsuperscript{50} for example, as

$$f'(x) \approx \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} + O(\Delta x^2)$$ $$\Rightarrow \left[ \frac{\partial}{\partial x} \right] = \begin{pmatrix} -1 & 0 & 1 & \cdots \\ 0 & -1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$ (49)

where $f'(x)$ is the derivative of a function defined on a finite grid, $\Delta x$ is the spacing between discretisation points, and $O(\Delta x^2)$ is the residual error determining the accuracy of the calculation. For example, if $\Delta x$ is twice finer, the grid will become twice finer and the error will be reduced by a factor equal to $\Delta x^2$. In Spinach, finite-difference derivative matrices can be set as

```plaintext
parameters.deriv={"period",n};
```

where the flag period indicates finite difference approximations with periodic boundary conditions and $n$ is the stencil size made by a point on the spatial grid with two, four, or six neighbors and can be set equal to 3, 5, and 7. The stencil size and the number of spatial points determine the accuracy of the calculation. Accurate simulations require proper digitization of the spatial dimension for the correct treatment of diffusion and flow. All the frequencies along the spatial dimension have to be below the Nyquist condition of the grid:

$$\Omega_{\text{max}} \leq \frac{L}{2B_0} \|H_z\|_2 \int |g(t)| dt,$$ (50)

where $L$ is the sample size, $\|H_z\|_2$ is the 2-norm corresponding to the largest eigenvalue of the Zeeman Hamiltonian, $g(t)$ is the gradient amplitude in the chosen dimension as a function of the time, and $B_0$ is the external magnetic field. The main source of high frequencies along the spatial dimension are gradient fields, so the highest frequency in the strongest gradient present in the sequence has to be correctly digitized. In Spinach, the ngridpts.m function can be used to determine the minimum number of points required to satisfy the Nyquist condition of the grid:

```plaintext
number_of_points = ngridpts
(gradient_amplitude,gradient_duration,...
isotope,max_coherence_order,sample_size);
```

Accurate simulations require a number of points six or seven times greater than the minimum number of points given by Nyquist condition, or as many as required for convergence.\textsuperscript{14} In the case of RF and rotor phase increment generators (Equation 45) used in simulations of shaped, frequency swept pulses and MAS, Spinach uses Fourier differentiation matrices\textsuperscript{51} on a grid with $N$ regularly spaced points:

$$\left[ \frac{\partial}{\partial \phi} \right]_{nk} = \begin{cases} (-1)^{n+k} & 2 \cot \left(\frac{(n-k)\pi}{N}\right) \\ 0 & n \neq k \\ \frac{n}{2} & n = k \end{cases}.$$ (51)

Fourier derivatives are calculated in the Spinach’s internal kernels and do not require specification as input parameter. In the Fokker–Planck formalism, the time propagation occurs in the direct product between spin space and spatial degrees of freedom:

$$\text{Fokker-Planck} = \text{Liouville space} \times \text{Spatial degree of freedom}.$$ (52)

Considering a 10-spin system, the dimension of the Liouville space is of the order of $4^{10} \sim 10^6$, and in the case of three-dimensional spatial dynamics that requires at least 100 points along the three dimensions, the result is a matrix of the order of $10^{12}$. Matrices of this dimension cannot be stored without the polyadic decomposition,\textsuperscript{14} which computes matrix–vector multiplication in Equation 52 without opening Kronecker product. The polyadic can be enabled with the following command:

```plaintext
sys.enable={"polyadic"};
```

At the time of writing, referring to the Spinach release 2.5, the polyadic is supported only within the imaging.m
context (described later in the text) and should be set for three-dimensional MRI simulations.

4.1 Fokker–Planck equation in imaging simulations

When we have multiple motions with the corresponding generators, then the overall dynamics is given by the sum of all the generators of individual dynamics. For example, the unidimensional Fokker–Planck equation that describes frequency swept pulses and shaped pulses in amplitude–frequency coordinates in the presence of magnetic field gradients, diffusion, and flow is given by

$$\frac{\partial}{\partial t} \rho(\phi, z, t) = -iL(\phi, z, t) + D_T \frac{\partial^2}{\partial z^2} + v(t) \frac{\partial}{\partial z} \rho(\phi, z, t),$$

where $L$ is the Liouvillian, $\partial/\partial \phi$ is the generator of the RF phase increment that increments the phase with rate $\omega_{RF}(t)$, $\partial^2/\partial z^2$ and $\partial/\partial z$ are the generators of spatial dynamics, $D_T$ is the translational diffusion coefficient in case of isotropic diffusion, and $v(t)$ is the flow velocity. In Fokker–Planck simulations, state vectors $\rho(\phi^{(n)}, z^{(k)}, t)$ are stacked vertically in order of increasing index of the spatial coordinates grid points, followed by the increasing index of the RF phase grid points. The matrix representation of Equation 53 is given in Figure 3.

Each element of the Liouvillian corresponds to

$$L_{\phi_n, z_k}(t) = H_0 + a(t)[S_X \cos(\phi_n + \phi_0) + S_Y \sin(\phi_n + \phi_0)]$$

radiofrequency term

$$+ g_z(t) z_k \sum_m \gamma_m S_Z^{(m)} + iR + iK,$$

gradient term

where $H_0$ is the time-independent part of the Hamiltonian and contains the static interactions; $a(t)$ is the time-dependent amplitude of the RF pulse, $\phi_0$ is the initial phase of the RF pulse, $\phi_n$ is the phase of the RF pulse at the point $n$, $\{S_X, S_Y\}$ are Cartesian radiofrequency operators, $g(t)$ is the time dependent gradient amplitude, $z_k$ is the spatial coordinate in the point $k$, $\gamma_m$ is the gyromagnetic ratio of the $m^{th}$ spin in the system, and $S_Z$ is the longitudinal spin commutation superoperator for each spin $m$, $R$ is the relaxation superoperator and $K$ is the kinetics super operator, $D_z$ and $D_{z^2}$ are finite-difference derivative matrices on the spatial grid, and $D_{\phi}$ is a Fourier derivative matrix on the phase grid. The digitalization of the RF phase and spatial dynamics on a grid makes the Liouvillian a constant matrix, and generators of time-dependent process

---

**Figure 3** Matrix representation of the unidimensional Fokker–Planck equation. The Liouvillian is represented as a $M \times N$ matrix, where $M$ is the total number of points in the phase grid and $N$ is the total number of points in the space grid.
RF phase, diffusion, and flow) become time independent. Equation 53 can be computed by calling the \texttt{shaped\_pulse\_af.m} function:

```matlab
final\_rho=shaped\_pulse\_af(spin\_system,L,Sx,\ldots
Sy,initial\_rho,parameters\_rf\_freq\_list,\ldots
parameters\_rf\_amp\_list,parameters\_duration\_list,\ldots
parameters\_rf\_phi0,parameters\_max\_rank,\ldots
parameters\_method);
```

The parameter $L$ is the Liouvillian, $S_x$ and $S_y$ are the pulse operators. Simulation parameters can be provided for the irradiation frequency, a value for the amplitude, a value for the phase of the pulse, a value for the overall pulse duration, and a value for the number of steps representing the pulse discretization points:

```matlab
parameters\_rf\_frequency
parameters\_rf\_amplitude
parameters\_rf\_phi0
parameters\_rf\_phase
parameters\_rf\_duration
parameters\_rf\_steps
```

The number of points in the RF phase grid is equal to $2k + 1$, where $k$ is termed the maximum rank and usually takes a value of 2 or 3. It can be set as:

```matlab
parameters\_max\_rank
```

The function \texttt{shaped\_pulse\_af.m} is used to simulate soft off-resonance and slice-selective pulses. Figure 4 shows an example of a pulse acquire experiment using a Gaussian soft pulse, applied on a system of 39 coupled protons, to select a signal at a specific irradiation frequency.

In this example, the simulation starts with longitudinal magnetization then a 90° hard or soft pulse is applied at 0 Hz before the signal acquisition. The selective pulse is simulated with the Fokker–Planck equation (Equation 53). Diffusion, flow, and pulse field gradients are not included in this calculation. Simulation parameters and the code (explained step by step) used to obtain Figure 4 are given in Examples 3a and 3b in Section B.3 of the supporting information. In the case of an MRI simulation involving slice selection pulses, the simulation must include pulse field gradients. The gradient operator is computed by calling the \texttt{g2fplanck.m} function, with the syntax

```matlab
G=g2fplanck(spin\_system,parameters);
```

where parameters provide the number of points in the spatial grid and the sample dimension. A three-dimensional MRI simulation can be set for example:

```matlab
parameters\_npts=[x\_points\ y\_points\ z\_points];
parameters\_dims=[x\_dimension\ y\_dimension\ z\_dimension];
```

An initial condition corresponding to longitudinal magnetization and a detection state corresponding to transverse magnetization in each point of the grid can be set, for example, as

```matlab
parameters\_rho\_0\_ph={ones (prod (parameters\_npts,1))};
parameters\_rho\_0\_st={state (spin\_system,'Lz','1H','cheap')};
parameters\_coil\_ph={ones (prod (parameters\_npts,1))};
parameters\_coil\_st={state (spin\_system,'Lz','1H','cheap')};
```

**FIGURE 4** Comparison between simulations of pulse acquire experiments: (a) using a hard 90°; (b) using a 90° selective Gaussian applied at 0 Hz
In the case the simulation includes spatial dynamics, the Fokker–Planck evolution generator can be computed by calling the v2fplanck.m function:

\[
F = \text{v2fplanck}(\text{spin\_system}, \text{parameters});
\]

where parameters correspond to flow velocities and diffusion that can be set on a spatial grid by providing a vector of values describing the evolution of the velocity in time. For example, constant flow velocity along each dimension can be set by:

\[
\text{parameters.u} = \text{flow\_velocity\_z} * \text{ones}(\text{parameters.npts});
\]

\[
\text{parameters.v} = \text{flow\_velocity\_y} * \text{ones}(\text{parameters.npts});
\]

\[
\text{parameters.w} = \text{flow\_velocity\_x} * \text{ones}(\text{parameters.npts});
\]

Isotropic diffusion at each point of the spatial grid can be set with the parameter:

\[
\text{parameters.diff};
\]

The Liouvillian, the relaxation and the kinetics superoperators, the Fokker–Planck spatial dynamics generator (including diffusion and flow), and the gradient operators can be generated internally by calling the imaging.m context function:

\[
\text{answer}=\text{imaging}(\text{spin\_system}, @\text{pulse\_sequence}, \ldots \text{parameters});
\]

that passes the operators to a pulse_sequence.m function:

\[
\text{answer}=\text{pulse\_sequence}(\text{spin\_system}, \text{parameters}, \text{H,R,K,G,F});
\]

The Liouvillian is built according to

\[
L = H + F + i R + i K;
\]

The gradient operator can then be added to the Liouvillian with the following command:

\[
L = L + \text{parameters.gradient\_amplitude}*\text{G(1)};
\]

where \(G(1)\) indicates a unidimensional gradient. Figure 5 shows an example of a three-dimensional MRI simulation where a slice selection pulse is applied to a brain phantom to select a single slice at a specified frequency.

In this example, the simulation starts with longitudinal magnetization on all the protons, and then, a 90° slice selection pulse is applied at given frequency. Finally, the observable operator \(\hat{L}_+\) can be computed using the fpl2phan.m function:

\[
\text{mri\_slice}=\text{fpl2phan}(\text{mri}, \text{state}(\text{spin\_system,'L+'}, \ldots \text{'1H'}, \text{'cheap'}, \text{parameters.npts}));
\]

Simulation parameters and the code (explained step by step) used to obtain Figure 5 are given in Example 4 in Section B.4 of the supporting information. An alternative approach for computing shaped pulses is the time slicing method,\(^52,53\) made available in Spinach by calling the shaped_pulse_xy.m function, which computes the shaped pulse in Cartesian coordinates. In this approach, shaped pulses are simulated explicitly using a time-dependent Liouvillian, whereas in Fokker–Planck

**Figure 5** Simulation of a magnetic resonance imaging slice selection pulse using the Fokker–Planck formalism. On the left: An initial phantom representing a human brain. On the right: A slice of the human brain after the slice selection pulse.
formalism, the time dependence is eliminated. In the time slice calculation, the total duration of the pulse is divided into \( n \) slices of duration \( \Delta t \); the density matrix is then propagated in each time slice in accordance with Equations 30–31. The total Liouvillian that contains time independent and RF terms is diagonalized to yield an eigenvalue matrix \( D \) and an eigenvector matrix, \( V \), which are used to create a propagator \( P_n \) in each time instance \( n \) according to

\[
P_n = V_n \exp[-iD_n \Delta t_n] V_n^\dagger.
\]

The overall propagator \( P \) is built from the product of all the propagators \( P_n \) and then multiplied by the initial state vector. The time discretization is also performed in the Fokker–Plank formalism, but algorithms implemented in Spinach avoid the diagonalization in Equation 55 because it scales according to \( O(N^3) \) with the matrix dimension \( N \). A frequency-swept pulse in the presence of a gradient can be simulated, for example as:

```python
final_rho=shaped_pulse_xy(spin_system,L,...
{Sx,Sy,G{1}},{Cx,Cy,gradient_amplitude_list},...
duration_list,initial_rho,parameters.method);
```

where the parameter \( L \) is the Liouvillian, \( Sx \) and \( Sy \) are the pulse operators, \( G{1} \) indicates a unidimensional gradient (optional), \( \text{gradient\_amplitude\_list} \) is a vector representing the variation of the gradient in the time, and \( Cx \) and \( Cy \) are the Cartesian amplitudes in X and Y coordinates at each time slice. Chirp pulses in the presence of a gradient are used in ultrafast NMR for spatial encoding; however, not all the full pulse is active for excitation and inversion. Simulations can show the active region in a chirp profile used to apply spatial encoding to a quantum coherence in a spin system of three spins, see Figure 6.

In this simulation, a spin echo block generates multiple quantum coherence, then the coherence \(+1\) is selected, and the chirp pulse inverts the coherence order from \(+1\) to \(-1\). The profile is linear in a region of about 10 mm, but the diffusion adds a distortion. Simulation parameters and the code (explained step by step) used to obtain Figure 6 are given in Example 5 in Section B.5 of the supporting information.

### 4.1.1 Comparison between propagation methods

In Spinach, the propagation of the Fokker–Planck equation can be performed either with the exponential propagation method or with the Taylor series\(^\text{13}\) (Equation 35). The former approach requires the calculation of a matrix exponential propagator that involves matrix–matrix multiplications that scale by \( O(N^3) \) with the matrix dimension. A shaped pulse in amplitude–frequency or Cartesian coordinates can be simulated using this approach with the following setting:

```python
parameters.method='expm';
```

The Taylor series involves only matrix–vector multiplications that scale by \( O(N^2) \). A shaped pulse in amplitude–frequency or Cartesian coordinates can be simulated using this approach with the following setting:

```python
parameters.method='expv';
```

**FIGURE 6** Chirp profile in the absence (a) and in the presence of diffusion (b)
Table 1 shows a comparison between the exponential propagator method and Taylor series method for the simulation of a shaped pulse in frequency–amplitude using the Fokker–Plank formalism (with the shaped_pulse_af.m function) and Cartesian coordinates (with the shaped_pulse_xy.m function) using the time slicing approach. The memory requirement for the storage of the Liouvillian superoperator in computations using the Taylor series is smaller than the requirement for the storage of the matrix exponential propagator in computations using the exponential propagation method, details are given in Table S1.

Table 1 shows an improvement in efficiency when calculations are performed using the Taylor method. Computations using the time slicing method are faster than calculations performed using the Fokker–Planck formalism. However, a pulse in Cartesian coordinates requires thousands of points to correctly digitize its rapidly oscillating x and y components, whereas fewer points (typically 100) are required to digitize a pulse in amplitude–frequency coordinates. Therefore, calculations become faster using the Fokker–Planck formalism when the propagation is computed with the Taylor series.

### 4.2 | Fokker–Planck equation in MAS simulations

The Fokker–Planck formalism can be also used in the context of NMR MAS. Its advantage relies in the disappearance of the time dependence from the MAS Hamiltonian, leading to an improvement in efficiency. The time dependence is also eliminated in the Floquet theory, and previous work has shown comparable performance between the two theories.

A lot of work has been invested in improving the efficiency of the MAS calculations through the optimization of two- and three-angle spherical grids. Considering that the gamma angles in the powder grid and the sample rotation angle both correspond to a rotation about the rotor axis, the gamma angles can be averaged over the rotor phase, and only a two-angle grid is required. This is a feature of several algorithms including the γ-COMPUTE algorithm. The Fokker–Planck equation in MAS simulations corresponds to:

\[
\frac{\partial}{\partial t} \rho(\varphi, t) = \left[ -iL(n, \varphi) + \omega_{\text{MAS}} \frac{\partial}{\partial \varphi_{\text{MAS}}} \right] \rho(\varphi, t),
\]  

where \( L \) is the Liouvillian, \( n = [111]/\sqrt{3} \) is the spinning axis direction, and \( \omega_{\text{MAS}} \) and \( \varphi_{\text{MAS}} \) are the MAS spinning rate and phase, respectively. In the Fokker–Planck formalism, the generator of time-dependent processes becomes time independent. The derivative \( \partial/\partial \varphi \) is the phase of the rotor incremented by a time-independent Fourier derivative (Equation 51). The rotor grid corresponds to a circular grid with \( 2k + 1 \) points, where \( k \) is termed maximum rank of the Fokker–Planck theory, usually set equal to 2 or 3. This parameter can be set as:

\[ \text{parameters.max_rank} \]

The minimum number of points in the rotor grid for these simulations is given by the ratio between the largest interaction present in the system divided by the spinning rate. The MAS Liouvillian becomes time independent through the digitalization the rotor phase on a periodic two-angle spherical grid \( \{\varphi_j\} \):
where $Q_{lm}$ is the rotational basis operators and $R$ and $K$ are the relaxation and kinetics superoperators, respectively. The matrix form of Equation 57 is given below:

$$\frac{\partial}{\partial t} \begin{pmatrix} \rho(\phi_1,t) \\ \rho(\phi_2,t) \\ \vdots \\ \rho(\phi_N,t) \end{pmatrix} = -i \begin{pmatrix} L(\phi_1) & 0 & \cdots & 0 \\ 0 & L(\phi_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & L(\phi_N) \end{pmatrix} \begin{pmatrix} \rho(\phi_1,t) \\ \rho(\phi_2,t) \\ \vdots \\ \rho(\phi_N,t) \end{pmatrix} + \omega_{\text{MAS}} D_{\phi} \begin{pmatrix} \rho(\phi_1,t) \\ \rho(\phi_2,t) \\ \vdots \\ \rho(\phi_N,t) \end{pmatrix}. \tag{58}$$

The Wigner D-matrices in Equation 58 accomplish the following transformations:

1. $D_{\text{crystal}}(\alpha_{PC}, \beta_{PC}, \gamma_{PC})$ rotates the molecule from the principal axis frame, in which the interactions are defined, into the crystallite-fixed frame, where $\alpha_{PC}$, $\beta_{PC}$, and $\gamma_{PC}$ represent the Euler angles between the two frames. In Spinach, a single crystallite orientation can be specified as:

   ```
   parameters.orientation
   ```

A set of crystallite orientations on a spherical averaging grid can be specified as:

```parameters.grid```

2. $D_{\text{rotor}}(\varphi_j)$ rotates the crystallite-fixed frame into the rotor-fixed frame defined by an orientation with a specified rotor phase. The rotor axis direction can be specified as:

   ```
   parameters.axis=[sqrt(2/3) 0 sqrt(1/3)];
   ```

The spinning rate $\omega_{\text{MAS}}$ is defined by:

```parameters.rate```

3. $D_{\text{laboratory}}(\alpha_{RL}, \beta_{RL}, \gamma_{RL})$ represents the transformation between the rotor-fixed frame and the laboratory frame corresponding to the rotor axis direction $n$, and $\alpha_{RL}$, $\beta_{RL}$, and $\gamma_{RL}$ represent the Euler angles describing the transformation between the rotor-fixed frame and the laboratory frame where the simulation is performed.

The transformations described above are performed internally by the `singlerot.m` context function. Figure 7 shows an example of simulation of a cross-polarization experiment between a proton and a $^{15}$N nucleus with and without MAS.

In this simulation, the polarization is transferred from the spin with higher $\gamma$ to the spin with smaller $\gamma$ through a dipolar coupling. The transfer occurs at the Hartmann–Hahn matching condition, which is fulfilled when the RF power at the nuclei is equalized, in according to $\gamma_{1H} B_{1,1H} = \gamma_{15N} B_{1,15N}$. The Hartmann–Hahn matching condition under MAS breaks into $\gamma_{1H} B_{1,1H} = \gamma_{15N} B_{1,15N} \pm n$ side bands appearing at integers

![FIGURE 7](image.png) Hartmann–Hahn matching condition as a function of the $^1$H radio frequency power, the matching condition occurs at (a) 50 kHz in a static sample and (b) at different integers of the spinning speed when the sample is under magic angle spinning with a rate equal to 10 kHz.
of the spinning speed. Simulation parameters and the code (explained step by step) used to obtain Figure 7 are given in Examples 6a and 6b in Section B.6 of the supporting information.

4.3 | Fokker–Planck equation in overtone MAS simulations

Overtone simulations are usually performed using a laboratory frame Hamiltonian because there is no convenient rotating frame due to the large difference between frequencies involved in the spin evolution process (order ~ Hz for $^1$H and order ~ MHz for $^{14}$N overtone), and the spinning frequency is of the order of kHz; and makes the spin Hamiltonian time dependent. Laboratory frame simulations require two points per period of the fast oscillation present in the system. Therefore, millions of time steps have to be averaged over a three-angle spherical grid, and the second rank Wigner D-functions make the spin Hamiltonian time dependent. Laboratory frame simulations require two points per period of the fast oscillation present in the system. Therefore, millions of time steps have to be averaged over a three-angle spherical grid. The convergence in an overtone pattern requires a number of points in the spherical grid much higher than those required to digitize single quantum transitions with the same number of spinning sidebands. Consider a spin system composed of a nitrogen $^{14}$N and a proton, the Fokker–Planck equation in overtone MAS simulations is:

$$\frac{\partial}{\partial t} \rho(\alpha, \beta, \gamma, \varphi_{\text{MAS}}^{N}, \varphi_{\text{RF}}^{N}, t) = \left[ -i \mathbf{L}(\alpha, \beta, \gamma, \varphi_{\text{MAS}}^{N}, \varphi_{\text{RF}}^{N}) + \omega_{\text{MAS}} \frac{\partial}{\partial \varphi_{\text{MAS}}^{N}} + \omega_{\text{RF}}^{N} \frac{\partial}{\partial \varphi_{\text{RF}}^{N}} \right] \rho(\alpha, \beta, \gamma, \varphi_{\text{MAS}}^{N}, \varphi_{\text{RF}}^{N}, t),$$

(59)

where $\mathbf{L}$ is the Liouvillian:

$$\mathbf{L}(\alpha, \beta, \gamma, \varphi_{\text{MAS}}^{N}, \varphi_{\text{RF}}^{N}) = \mathbf{H}_0 + \sum_{k,n,m=-\infty}^{\infty} \mathbf{Z}^{(2)}_{km}(\varphi_{\text{MAS}}^{N}) \mathbf{Z}^{(2)}_{nm}(\alpha, \beta, \gamma) \mathbf{Q}_{km}$$

$$+ \alpha_{\text{RF}}^{N} [S_{X}^{N} \cos \theta + S_{Y}^{N} \sin \theta \sin \varphi_{\text{RF}}^{N}] + \alpha_{\text{mas}}^{N} [S_{Z}^{N} \sin \theta + S_{X}^{N} \cos \theta] + \mathbf{R} + i \mathbf{K},$$

(60)

where $\mathbf{H}_0$ is the isotropic part of the spin Hamiltonian; $\mathbf{Q}_{km}$ is the rotational basis operators; $\mathbf{R}$ and $\mathbf{K}$ are the relaxation and kinetics superoperators, respectively; $\alpha$, $\beta$, and $\gamma$ are the three Euler angles in the powder grid; $\theta$ is spinning axis at the magic spin angle; and $\omega_{\text{MAS}}$ and $\omega_{\text{RF}}^{N}$ are the MAS spinning rate and overtone frequency, respectively, specified by:

and

```
parameters.overtone_frequency
```

The terms $\varphi_{\text{MAS}}$ and $\varphi_{\text{RF}}^{N}$ correspond to the rotor and nitrogen RF phases, respectively, specified by:

```
parameters.max_rank
```

The terms $\alpha_{\text{RF}}^{H}$ and $\alpha_{\text{RF}}^{N}$ are the RF power amplitudes of the proton and nitrogen specified by:

```
parameters.rf_pwr
```

This parameter requires a vector of two values where the first value corresponds to the RF power on the $^{14}$N and the second values correspond to the RF power on the proton; $\{S_{X}^{N}, S_{Y}^{N}, S_{Z}^{N}\}$ are the nitrogen Zeeman commutation super operators, and $\{S_{X}^{H}, S_{Y}^{H}, S_{Z}^{H}\}$ are the proton Zeeman commutation super operators that can be set at the MAS respectively, for example:

```
parameters.Nx=cos(atan(sqrt(2)))*
operator(spin_system,'Lz','14N')+
sin(atan(sqrt(2)))*
operator(spin_system,'L+', '14N')+
operator(spin_system,'L-','14N'))/2;
```

and

```
parameters.Hx=cos(atan(sqrt(2)))*
operator(spin_system,'Lz','1H')+
sin(atan(sqrt(2)))*
operator(spin_system,'L+','1H')+
operator(spin_system,'L-','1H'))/2;
```

The second rank Wigner D-functions $\mathbf{Z}^{(2)}_{nm}(\alpha, \beta, \gamma)$ specify the crystallite orientation within the powder grid, and the second rank Wigner D-functions $\mathbf{Z}^{(2)}_{kn}(\varphi_{\text{MAS}}^{N}, \varphi_{\text{RF}}^{N})$ define the rotor orientation and phase using an angle-axis parameterization. The effective Hamiltonian is computed using a matrix logarithm technique:

$$\left< \hat{H}(\mathbf{x}) \right> = \frac{i}{T} \ln \left[ \exp_{(0)} \left( -i \int_{0}^{T} \hat{H}(\mathbf{x}, t) \, dt \right) \right],$$

(61)

where $\exp_{(0)}$ is a time-ordered exponential that can be computed by slicing the RF period $T$ into 16 slices and then multiplying the slice propagator using the propagator squaring approach. The Hamiltonian in Equation 61
becomes time-independent, the time propagation for the duration of the cross-polarization period can be computed according to:

\[
\dot{\rho}(t_{CP}) = \exp[-i\hat{H}t_{CP}]\rho_0,
\]

where \(\rho_0\) and \(\dot{\rho}(t_{CP})\) are the initial density matrix and final density matrix after the cross-polarization period, respectively; and \(t_{CP}\) is the duration of the cross-polarization period. This parameter can be set, for example, as:

```plaintext
parameters.rf_dur
```

Considering that only few hundred points occupied by the overtone spectrum have to be acquired, the acquisition is performed in the frequency domain detection that is much faster than the time domain detection\(^{67,68}\). The Fourier transform of the free induction decay is performed on \(^{14}\)N and corresponds to:

\[
f(\omega) = -i\left(\hat{S}_N^+ \left(\hat{H} + \omega\hat{E} + i\hat{r}\right)^{-1}\right)\rho_0,
\]

where \(\hat{S}_N^+\) is the detection state on \(^{14}\)N, \(\hat{E}\) is the unit matrix, and \(r\) is the damping rate parameter. The latter parameter is required to avoid singularities in the complex plane. The matrix-inverse-times-vector operation can be performed with an iterative solver called ILU preconditioned generalized minimal residual (GMRES).\(^{70}\) A frequency domain acquisition in accordance to Equation 63 can be performed by calling the slowpass.m function:

```plaintext
spectrum=slowpass(spin_system,parameters,H,R,K);
```

The initial condition and the detection state corresponding to the proton and nitrogen magnetization at the MAS can be set, for example, as:

```plaintext
parameters.rho0=cos(atan(sqrt(2)))*...
state(spin_system,'Lz','1H')+
sin(atan(sqrt(2)))*...
(state(spin_system,'L+','1H') +
state(spin_system,'L−','1H'))/2;
```

and

```plaintext
parameters.coil=cos(atan(sqrt(2)))*...
state(spin_system,'Lz','14N')+
sin(atan(sqrt(2)))*...
(state(spin_system,'L+','14N') +
state(spin_system,'L−','14N'))/2;
```

Figure 8 shows an example of a simulation of a cross-polarization experiment between a proton and a \(^{14}\)N overtone transition in glycine under MAS.

The cross-polarization process has already been described above in this section, Figure 8 shows the \(^{14}\)N overtone excitation at 48 kHz. Simulation parameters and the code (explained step by step) used to obtain Figure 8 are given in Example 7 in Section B.7 of the supporting information.

5 | EFFICIENCY OPPORTUNITIES

The computational efficiency can be improved reducing the dimension of the state vector/Liouvillean including only spin states that practically contribute to the spin dynamics. Alternatively, parallel computing architectures, such as graphics processing units (GPU) or computer clusters can be used.

5.1 | State space restriction

As compared with the Hilbert space, simulations in the Liouville space are more time-consuming since matrices have larger dimension. However, the dimension of the state vector can be reduced, using restricted state space and low correlation order techniques.\(^{16,17,71,72}\) In Section 3, we saw that the spin space is given by the direct product of Pauli matrices, \(P_1 \otimes P_2 \otimes \ldots \otimes P_m\), where \(P_m\) is
the IST operator or a Pauli matrix of the spin \( m \) or the identity operator. In the case of 1/2-spins, the matrix dimension is equal to \( 2^N \) in the Hilbert space and equal to \( 4^N \) in the Liouville space, where \( N \) is the number of spins. The matrix dimension therefore grows exponentially with the number of spins in the system. However, the dimension of the space spanned by a realistic spin system trajectory is usually much smaller than the dimension of the full Liouville space.\(^{16,17,71,73} \) The initial state vector is usually composed of the sum of single spin operators corresponding to the \( L_Z \) or \( L_\pm \) states. For example, for a spin system composed of three spins, the state vector representing the longitudinal magnetization corresponds to \( \rho = L_Z^{(1)} + L_Z^{(2)} + L_Z^{(3)} \). After the action of pulses and evolutions, the single spin operators evolve into multispin operator (or high-spin order) corresponding to product states equal to \( L_+^{(1)} \otimes L_-^{(2)} \) (two-spin order), \( L_+^{(1)} \otimes L_+^{(2)} \otimes L_-^{(3)} \) (three-spin order), and so on. In liquid state simulations, the presence of relaxation prevents high-order coherences and high levels of spin correlation from being populated to a significant extent.\(^{73} \) Figure 9 shows an example of dynamics from one to seven-spin order during a pulse-acquire experiment on a strychnine spin system.

Figure 9 shows that spin order higher than four do not contribute to the dynamics and can be neglected. Simulation parameters and the code (explained step by step) used to obtain Figure 9 are given in Example 8 in Section B.8 of the supporting information. If we exclude correlations of more than \( k < n \) spins, the operators would get the form \( E \otimes P_1 \otimes \ldots \otimes P_2 \otimes E \otimes \ldots \otimes P_k \), corresponding to a number of IST operators or Pauli matrices scattered among the unit matrix operators, \( E \). The dimension of the new state space is

\[
N = 4^k \frac{n!}{k!(n-k)!} = 4^k \frac{1}{k!} n(n-1)...(n-k+1) = O\left(4^n k^k\right),
\]

(64)

The cost of the system propagation in this new restricted space scales polynomially with respect to the total number of spins, and the simulation time is reduced. There are methods implemented in Spinach that can be applied a priori in order to decide which state in practice would contribute to the dynamics. This analysis is performed internally, these methods include:

5.1.1 Interaction graph and connected subgraph analysis

The interaction graph is based on the local coupling density and on the fact that high-spin orders relax faster than low-spin orders and therefore do not accumulate in large quantity.\(^{16} \) Even complicated magnetic resonance experiments do not often include spin order higher than four and correlations between remote spins so they can be dropped from the basis. For a given spin, the relevant states and coherences are those that involve its close neighbors on an interaction graph that can be expanded in a complete set of connected subgraphs (spin clusters) of a specified size composed by a
spin and its nearest neighbors. The size of the interaction matrices is specified by the users in the spin system specification, these can include the product states between directly coupled spins or between spins closer than a given cutoff. For example, a protein system in liquid state has billions of $J$-couplings and dipolar interactions, but only couplings larger than 5 Hz and interactions within 5 Å between three or four closest spins are important. Hence, setting the interaction cutoff up to 5 Hz and the proximity cutoff up to 5 Å, the number of $J$-couplings and dipolar interactions is reduced to less than hundreds of thousands. In this way, the important states are included and the entanglements higher than the size of biggest cluster are excluded, and the simulation time gets dramatically reduced. In Spinach, the option to request complete basis set, which is only practical up to about 20 spins in Hilbert space and 10 spins in the Liouville space, can be specified as:

```matlab
bas.approximation='none';
```

The option to use incomplete basis sets can be specified as:

```matlab
bas.approximation='IK-0' or 'IK-1' or 'IK-2';
```

where the option 'IK − 0' includes all product states—correlations between indirectly coupled spins up to a number of spins specified by a cutoff level:

```matlab
bas.level
```

This basis set is used to drop correlations between very remote spins. The option 'IK − 1' includes all product states between directly coupled spins up to a number specified by the cutoff bas.level and correlation between indirectly coupled spins up to a number specified by a cutoff space level:

```matlab
bas.space_level
```

that are closer together than a user-specified tolerance, representing the proximity cutoff radius in Angstroms:

```matlab
sys.tols.prox_cutoff
```

The option 'IK − 2' includes, for every spin, all correlations with all directly coupled spins and correlations with up to a number of spins specified by a cutoff space level:

```matlab
bas.space_level
```

that are closer together than the user-specified proximity cutoff. Fastest simulations use incomplete basis sets that are only available within the Liouville space.

### 5.1.2 Accuracy of the state space restriction approximation

The applicability for the state space restriction approximation requires that the fraction of the magnetization leaking outside the restricted state space does not exceed the user-specified tolerances. This can be fulfilled when

$$k > 2\sqrt{\frac{r}{h}} \text{erfc}^{-1}\left(\xi \text{erfc}\left(\frac{1}{2}\sqrt{\frac{r}{h}}\right)\right), \quad (65)$$

where $k$ is the cutoff level specified by the users—the maximum spin correlation order to be included in the basis, $\xi$ is the user-specified tolerance, $r$ is the average relaxation rate in the system, and $h$ is the average spin–spin coupling. The critical parameter is the $h/r$ ratio of the largest spin–spin coupling to the smaller relaxation rate. Spin systems characterized by small $h/r$ ratio, where the time scale of the relaxation process is smaller or comparable with the time scale of the dynamics, can be accurately described by low correlation order and require small basis sets (e.g., in liquid-state NMR systems). Spin systems characterized by large $h/r$ ratio, where the time scale of the relaxation process is slower than the time scale of the dynamics, require larger basis sets (e.g., in solid-state NMR systems). Consequences of the state space restrictions on the simulation time were already shown in a previous work. It is advisable to run trajan.m function (described in Section 3.6) to confirm that the basis set is sufficiently large, as shown in Figure 9.

### 5.1.3 Trajectory level pruning and Krylov subspace analysis

After the state space restriction using the interaction graph analysis, if operators are still sparse, the state space can be further reduced by considering only the states that contribute to the system evolution under a pulse sequence. The idea of the trajectory level pruning in restricted state spaces consists of finding the minimal basis sets that contribute to the simulation trajectory. All simulation trajectories have a finite duration and finite step length. The space spanned by
trajectory vectors is called the Krylov subspace. Many spin states are not populated in a typical magnetic resonance simulation. Therefore, the dimension of the Krylov subspace is smaller than the dimension of the full state space, and the size of the simulation can be reduced by finding a way to map these subspaces. Once a trajectory under a given Liouvillian has been obtained, the minimal basis sets spanning this trajectory are obtained using orthogonalization techniques, for example, Gram–Schmidt,\textsuperscript{74} single value decomposition (SVD),\textsuperscript{75} and Lanczos.\textsuperscript{46} In Spinach, the minimal basis set construction is considered complete when the next trajectory vector is entirely contained in the previous, or within a user-specified tolerance. This option is switched on by default. Systems involving fewer than five spins do not require sophisticated trajectory-level state space analysis, and the option should be disabled with the command:

```matlab
sys.disable={'trajlevel'};
```

An example in this setting is given in Example 7 in the supporting information.

### 5.1.4 Zero track elimination

The problem described in Section 5.1.3 can be reformulated in a way that instead of considering states that appear in the Krylov subspace, and then projecting them into that subspace, we look at states that do not appear in the Krylov subspace and project them out. Liouvillians and the typical initial states (i.e., $\hat{S}_z$ and $\hat{S}_\pm$) are sparse matrices, in a typical pulse sequence, many elements of the state vector will remain zero, and dimensions corresponding to those states can be removed. The zero track elimination (ZTE) techniques\textsuperscript{17} detect states that do not appear in the Krylov subspace and prunes them out. In Spinach, the ZTE is switched on by default and works by inspecting the first few steps in the system trajectory, end drops states that did not get populated. The ZTE is equivalent to the Krylov subspace analysis, but is more efficient. State space restrictions in addition to ZTE are useful in simulations of large systems of strongly coupled spins.

### 5.1.5 Symmetry pruning

The symmetry pruning is based on symmetries and connectivities in the Liouville space and considers that only fully symmetric irreducible representations of a symmetry group contributing to the simulation.\textsuperscript{76} Symmetric systems, like magnetically equivalent nuclei, are represented by symmetry groups that commute with the Hamiltonian. A simulation that starts in a state with a particular symmetry and evolves under a Hamiltonian that conserves that symmetry would never populate states of different symmetries. The Spinach kernels detect these states and generate a block-diagonal Liouvillian in symmetry-adapted basis sets. Each block can be simulated separately, leading to improvement in the simulation efficiency.

### 5.1.6 Liouvillian path tracing

In simulations involving spins that do not interact with each other, the coherence order of each spin is conserved independently. The Liouvillian path tracing (PT) technique is used to identify disconnected subspaces in the state space, introduced by accidental symmetries and conservation laws, and propagates them individually.\textsuperscript{13,76} This option is switched on by default in Spinach, and it can be disabled with following command:

```matlab
sys.disable={'pt'};
```

It is used in Liouville space simulations of large densely coupled spin systems encountered in electron paramagnetic resonance spectroscopy and spin chemistry.

### 5.1.7 Destination state screening

Further improvements can be made to the previous methods considering only spin system trajectories that evolve to or affect the detection state. The destination state screening\textsuperscript{77} identifies system trajectories containing contribution from both the initial and the detection states and retain them. It can be set with the destination flag in the evolution.m function (described in Section 3.5.4).

### 5.2 Parallelization in multicore systems

It is believed that parallel computing architectures, including GPUs, multicore central processing units (CPUs), and computer clusters, can improve the efficiency of simulations. This is not always true. The equation of the motion are integrated into software packages in a matrix-vector representation; parallel
architectures can accelerate matrix-vector multiplications per second. A single NVidia Titan V card can perform $10^{13}$ multiplications per second, and a modern CPU with 16 cores and 4-GHz clock frequency and 4 multiplications per cycle can perform 256 multiplication per second; these are indeed very good numbers. However, the key to the efficiency relies on latency and bandwidth. The former is the time taken for the data requested by the CPU to start arriving from the different memory banks, and the latter is the rate at which that data arrives. Every bite of memory is always 1 ns away from the CPU, and during that time, a CPU would wait rather than calculate. Parallel computing systems can improve the efficiency of calculations that involve linear memory access (e.g. fast Fourier transform), matrices larger than $200 \times 200$ (smaller matrices would spend more time in transit than multiplying), simulations involving billions of identical calculations. Few examples include: different orientations in powder average simulations, different points of on a grid, and different time slices in multidimensional NMR simulations. Several functions in Spinach that take advantage of parallel architectures are switched off by default. The GPU arithmetic on NVidia GPU cards can be switched on with the command:

```matlab
sys.enable={'gpu'};
```

Situations where the state spaces are dominated by a single large subgraph, for example, multidimensional liquid state NMR simulations, can be accelerated using the greedy parallelization that can be switched on with the command

```matlab
sys.enable={'greedy'};
```

Situations where there are repeated requests of expensive functions of the same argument, for example, matrix exponential in the propagator function, can be solved using hashing and caching algorithms. These can be switched on with the command

```matlab
sys.enable={'chaching'};
```

In this procedure, a matrix is first hashed, meaning that a record is kept into the disk and then fetched when needed, rather than recomputed. This approach can be used when sufficient storage is available, in cases where the matrix exponentialization dominates the numerical cost of the simulation (e.g., optimal control, or in MAS simulations, where each rotor period contains identical Hamiltonian).

## 6 CONCLUSION

In summary, it is possible to conclude that compared with the Bloch equations, the density operator and the Fokker–Planck formalisms represent a more general way to simulate the spin dynamics process like relaxation and kinetics. The Fokker–Planck theory allows simulations of complicated magnetic resonance experiments (SPEN-NMR, MRI, and MAS NMR,) where spin dynamics coexist with spatial dynamics. In the Liouville space, efficiency improvements can be obtained with state space restriction and low correlation orders methods, or through the utilization of parallel computer architectures. Current challenges in spin dynamics remain related to the matrix dimension, particularly in the solid state where the relaxation is slow, and in MRI involving metabolomics (composed by more than six spins) diffusing in a three-dimensional space. The problem in solid state may be solved considering that the powder averaging has the same effect as relaxation in liquid state. In MRI simulations, there are some correlations between spin and spatial degrees of freedom that may be redundant in the presence of relaxation and diffusion, and may be removed from the Fokker-Planck space. This is a work in progress in our team. Algorithms implemented in Spinach allow large-scale magnetic resonance simulations, including MRI, powder average and relaxation theories. It also provides a software environment for pulse sequence development with the possibility of using the optimal control theory, and supports parallel processing on GPU and computer clusters. Another advantage of Spinach is that it is an open source and allows direct access to an intuitive and well-documented code. Instructions on installation and practical information can be found in Kuprov or on the website http://spindynamics.org/. Information on each function described in this tutorial can be found on the Spinach wiki page http://spindynamics.org/wiki/index.php?title=Main_Page.

### ACKNOWLEDGEMENTS

I acknowledge Dr Giuseppe Pileio and Professor Marina Carravetta (University of Southampton) for useful comments about the manuscript. A special thanks to Professor Ilya Kuprov (University of Southampton) for the useful comments about the manuscript and for the very good supervision and Mr J. Keeley (University of Southampton) for having thoroughly proofread the manuscript. The financial support from Engineering and Physical Science Research Council (EPSRC) through an Impact Acceleration Grant is also acknowledged.
REFERENCES
[1] F. Bloch, Phys Ther Rev 1946, 70, 460.
[2] H. C. Torrey, Phys Ther Rev 1956, 104, 563.
[3] J. V. Neumann, Göttinger Nachrichten 1927, I, 245.
[4] A. Abragam, The principles of nuclear magnetism, Clarendon Press, Oxford 1961.
[5] R. R. Ernst, G. B. A. Wokaun, Principles of nuclear magnetic resonance in one and two dimensions, Clarendon Press, Oxford 1987.
[6] A. G. Redfield, IBM J Res Dev 1957, 1, 19.
[7] A. D. Fokker, Ann Phys 1914, 348, 810.
[8] H. Benoit-Cattin, G. Collewet, B. Belaroussi, H. Saint-Jalmes, C. Odet, J Magn Reson 2005, 173(1), 97.
[9] Z. Tosner, R. Andersen, B. Stevensson, M. Edén, N. C. Nielsen, T. Vosgeaard, J Magn Reson 2014, 246, 79.
[10] M. Veshtort, R. G. Griffin, J Magn Reson 2006, 178(2), 248.
[11] C. Bengs, M. H. Levitt, Magn Reson Chem 2018, 56(6), 374.
[12] F. S. Deoburegas, J. S. Waugh, J Magn Reson 1992, 96(2), 280.
[13] H. J. Hogben, M. Krzyzyniak, G. T. F. Charnock, P. J. Hore, I. Kuprov, J Magn Reson 2011, 208(2), 179.
[14] A. J. Allami, M. G. Concilio, P. Lalby, I. Kuprov, Sci Adv 2019, 5(7), eaaw962.
[15] L. J. Edwards, D. V. Savostyanov, Z. T. Welderufael, D. Lee, I. Kuprov, J Magn Reson 2014, 243, 107.
[16] I. Kuprov, N. Wagner-Rundell, P. J. Hore, J Magn Reson 2007, 189(2), 241.
[17] I. Kuprov, J Magn Reson 2008, 195(1), 45.
[18] D. Van Nguyen, J.-R. Li, D. Grebenkov, D. Le Bihan, J Comput Phys 2014, 263, 283.
[19] T. H. Jochimsen, A. Schäfer, R. Bammer, M. E. Moseley, J Magn Reson 2006, 180(1), 29.
[20] D. Sinnaeve, Concept Magn Reson Part A 2012, 40a(2), 39.
[21] L. Beltrachini, Z. A. Taylor, A. F. Frangi, J Magn Reson 2015, 259, 126.
[22] J. E. Tanner, E. O. Stejskal, J Chem Phys 1965, 42, 288.
[23] R. M. Henkelman, X. Huang, Q. S. Xiang, G. J. Stanis, S. D. Swanson, M. J. Bronskill, Magn Reson Med 1993, 29(6), 759.
[24] M. L. Congbo Cai, Z. Chen, x. Chen, S. Cai, J. Zhong, CR Physique 2008, 9, 119.
[25] I. Kuprov, Magn Reson Chem 2018, 56(6), 415.
[26] B. C. Sanctuary, J Chem Phys 1976, 64(11), 4352.
[27] B. C. Sanctuary, Mol Phys 1983, 48(6), 1155.
[28] J. Gendell, J. H. Freed, G. K. Fraenkel, J Chem Phys, 1964, 41, 949.
[29] J. H. Freed, G. K. Fraenkel, J Chem Phys 1963, 39, 326.
[30] B. C. Sanctuary, T. K. Halstead, Adv Magn Reson 1990, 15, 79.
[31] D. M. Brink, G. R. Satchler, Angular Momentum, Clarendon Press, Oxford 1993.
[32] D. A. Varshalovich, A. N. Moskalev, V. K. Khersonski, Quantum Theory of Angular Momentum: Irreducible Tensors, Spherical Harmonics, Vector Coupling Coefficients, 3nf Symbols, World Scientific, Singapore 1988.
[33] S. A. Smith, T. O. Levante, B. H. Meier, R. R. Ernst, J Magn Reson, Ser A 1994, 106(1), 75.
[34] M. H. Levitt, Spin dynamics: Basics of nuclear magnetic resonance, John Wiley & Sons, Falkreath 2001.
[35] P. Hore, J. Jones, S. Wimperis, NMR: The toolkit: How pulse sequences work, Oxford University Press, USA 2015.
[36] J. Keeler, Understanding NMR spectroscopy, John Wiley & Sons, Falkreath 2011.
[37] C. Moler, C. Van Loan, SIAM Rev 2003, 45(1), 3.
[38] M. Hochbruck, C. Lubich, SIAM J Numer Anal 1997, 34(5), 1911.
[39] G. Pileio, J. N. Dumez, I. A. Pop, J. T. Hill-Cousins, R. C. D. Brown, J Magn Reson 2015, 252, 130.
[40] G. Pileio, S. Bowen, C. Laustsen, M. C. D. Taylor, J. T. Hill-Cousins, L. J. Brown, R. C. D. Brown, J. H. Ardenkjaer-Larsen, M. H. Levitt, J Am Chem Soc 2013, 135(13), 5084.
[41] M. G. Concilio, P. Kiraly, G. A. Morris, J Magn Reson 2019, 301, 85.
[42] L. Guduff, A. J. Allami, C. van Heijenoort, J. N. Dumez, I. Kuprov, Phys Chem Chem Phys 2017, 19(27), 17577.
[43] R. P. Mason, J. H. Freed, J Phys Chem 1974, 78(13), 1321.
[44] R. P. Mason, C. F. Polnasze, J. H. Freed, Fed Proc 1974a, 33(5), 1371.
[45] R. P. Mason, C. F. Polnaszek, J. H. Freed, J Phys Chem 1974b, 78(13), 1324.
[46] G. Moro, J. H. Freed, J Chem Phys 1981, 75(6), 3157.
[47] G. V. Bruno, J. H. Freed, Chem Phys Lett 1974, 25(3), 328.
[48] I. Kuprov, J Magn Reson 2016, 270, 124.
[49] L. J. Edwards, D. V. Savostyanov, A. A. Nevzorov, M. Concistré, G. Pileio, I. Kuprov, J Magn Reson 2013, 235, 121.
[50] J. P. S. Bizarro, P. Rodrigues, Phys Plasmas 2001, 8(5), 1903.
[51] L. N. Trefethen, SIAM, Philadelphia 2000.
[52] P. Hodgkinson, D. Sakellariou, L. Emsley, Chem Phys Lett 2000, 326(5–6), 515.
[53] I. Frantsuzov, M. Ernst, S. P. Brown, P. Hodgkinson, Solid State Nucl Mag Reson 2015, 70, 28.
[54] I. Scholz, J. D. van Beek, M. Ernst, Solid State Nucl Mag Reson 2010, 37(3–4), 39.
[55] M. Leskes, P. K. Madhhu, S. Vega, Prog Nucl Mag Reson Spectrosc 2010, 57(4), 345.
[56] A. Schmidt, S. Vega, J Chem Phys 1992, 96(4), 2655.
[57] B. Stevenson, M. Eden, J Magn Reson 2006, 181(1), 162.
[58] M. Eden, Concept Magn Reson Part A 2003, 18a(1), 24.
[59] A. Ponti, J Magn Reson 1999, 138(2), 288.
[60] M. Eden, M. H. Levitt, J Magn Reson 1998, 132(2), 220.
[61] M. Bak, N. C. Nielsen, J Magn Reson 1997, 125(1), 132.
[62] S. J. Varner, R. L. Vold, G. L. Hoatson, J Magn Reson, Ser A 1996, 123(1), 72.
[63] D. M. Wang, G. R. Hanson, J Magn Reson, Ser A 1995, 117(1), 1.
[64] M. J. Mombourquette, J. A. Weil, J Magn Reson 1992, 99(1), 37.
[65] P. Hodgkinson, L. Emsley, Prog Nucl Mag Reson Spectrosc 2000, 36(3), 201.
[66] M. Hohwy, B. Bildsoe, H. J. Jakobsen, N. C. Nielsen, J Magn Reson 1999, 136(1), 6.
[67] M. Concistré, I. Kuprov, I. M. Haies, P. T. Williamson, M. Carravetta, J Magn Reson 2019, 298, 1.
[68] I. M. Haies, J. A. Jarvis, L. J. Brown, I. Kuprov, P. T. Williamson, M. Carravetta, *Phys Chem Chem Phys* 2015, 17(37), 23748.

[69] W. B. Blanton, J. W. Logan, A. Pines, *J Magn Reson* 2004, 166(2), 174.

[70] Y. Saad, M. H. Schultz, *SIAM J Sci Stat Comput* 1986, 7(3), 856.

[71] M. C. Butler, J. N. Dumez, L. Emsley, *Chem Phys Lett* 2009, 477(4–6), 377.

[72] J. N. Dumez, M. C. Butler, L. Emsley, *J Chem Phys* 2010, 133(22), 224501.

[73] A. Karabanov, I. Kuprov, G. T. P. Charnock, A. van der Drift, L. J. Edwards, W. Köckenberger, *J Chem Phys* 2011, 135(8), 084106.

[74] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia 1996.

[75] S. Gugercin, *Linear Algebra Appl* 2008, 428(8–9), 1964.

[76] H. J. Hogben, P. J. Hore, I. Kuprov, *J Chem Phys* 2010, 132(17), 174101.

[77] M. Krzystyniak, L. J. Edwards, I. Kuprov, *J Magn Reson* 2011, 210(2), 228.

[78] D. L. Goodwin, I. Kuprov, *J Chem Phys* 2016, 144(20), 204107.

[79] E. A. Suturina, I. Kuprov, *Phys Chem Chem Phys* 2016, 18(38), 26412.

[80] D. L. Goodwin, I. Kuprov, *J Chem Phys* 2015, 143, 084113.

[81] P. de Fouquieresa, S. G. Schirmera, S. J. Glaserb, I. Kuprov, *J Magn Reson*, 2011, 212, 412.

**SUPPORTING INFORMATION**

Additional supporting information may be found online in the Supporting Information section at the end of this article.

---

**How to cite this article:** Concilio MG. Large-scale magnetic resonance simulations: A tutorial. *Magn Reson Chem*. 2020;58:691–717. [https://doi.org/10.1002/mrc.5018](https://doi.org/10.1002/mrc.5018)