Getting started with NMR simulations

Describing a single quantum system

When the Hamiltonian is constant, Schrödinger's equation has a simple solution:

$$\frac{\partial}{\partial t}\left|\psi\right\rangle = -i\mathbf{H}\left|\psi\right\rangle, \qquad \left|\psi(t)\right\rangle = \exp(-i\mathbf{H}t)\left|\psi(0)\right\rangle \tag{1}$$

Here $\left|\psi\right\rangle$ is the system state vector (*aka* wavefunction), $\mathbf{H}$ is the Hamiltonian matrix, and the exponential of a matrix is defined using the Taylor series:

$$\exp(-i\mathbf{H}t) = \sum_{n=1}^{\infty}\frac{(-it)^n}{n!}\mathbf{H}^n \tag{2}$$

For a system with fewer than ten spins, modern computers run this in milliseconds. Once the wavefunction is known, observables are calculated in the standard way, for example:

$$L_{\mathrm{X}}(t) = \left\langle\psi(t)\right|\mathbf{L}_{\mathrm{X}}\left|\psi(t)\right\rangle \tag{3}$$

When the Hamiltonian does depend on time, it may still be approximated as piecewise constant, in which case the outcome of the previous time slice is the initial condition for the next one. Up to numerical technicalities, Equations (1)-(3) are the summary of time-domain quantum mechanics.

Describing an ensemble of quantum systems

An NMR sample is never a single molecule – it is an ensemble of a large number of them. For quantum ensembles, the fundamental quantity is not the wavefunction, but the density matrix:

$$\boldsymbol{\rho} = \left|\psi\right\rangle\left\langle\psi\right| \tag{4}$$

The equation of motion and its solution follow from Schrödinger's equation:

$$\frac{\partial\boldsymbol{\rho}}{\partial t} = -i[\mathbf{H},\boldsymbol{\rho}], \qquad \boldsymbol{\rho}(t) = \exp[-i\mathbf{H}t]\boldsymbol{\rho}(0)\exp[+i\mathbf{H}t] \tag{5}$$

The same applies to the calculation of observables. For an observable quantity O:

$$\left\langle O\right\rangle = \left\langle\psi\right|\mathbf{O}\left|\psi\right\rangle = \mathrm{Tr}\left(\left\langle\psi\right|\mathbf{O}\left|\psi\right\rangle\right) = \mathrm{Tr}\left(\left|\psi\right\rangle\left\langle\psi\right|\mathbf{O}\right) = \mathrm{Tr}(\boldsymbol{\rho}\mathbf{O}) \tag{6}$$

where we have used the fact that cyclic permutations of products are allowed under the trace:

$$\mathrm{Tr}(\mathbf{ABC}) = \mathrm{Tr}(\mathbf{CAB}) = \mathrm{Tr}(\mathbf{BCA}) \tag{7}$$

For a single quantum system, Equations (4)-(6) are equivalent to Schrödinger's equation. There are two reasons why density matrix is needed to describe ensembles:

1. Wavefunction phase does not influence observables; it can vary randomly across an ensemble. As a result, $\left|\psi\right\rangle$ does not survive the ensemble average, whereas $\left|\psi\right\rangle\left\langle\psi\right|$ does:

$$\overline{\left|\psi\right\rangle} = \frac{1}{2\pi}\int_0^{2\pi} e^{i\varphi}\left|\psi\right\rangle d\varphi = 0, \qquad \overline{\left|\psi\right\rangle\left\langle\psi\right|} = \frac{1}{2\pi}\int_0^{2\pi} e^{i\varphi}\left|\psi\right\rangle\left\langle\psi\right|e^{-i\varphi}d\varphi = \left|\psi\right\rangle\left\langle\psi\right| \tag{8}$$

2. $\left|\psi\right\rangle$ becomes unacceptably large for ~10^{23} systems, but $\left|\psi\right\rangle\left\langle\psi\right|$ can simply be averaged.

From the practical perspective, $\boldsymbol{\rho}$ is a more sophisticated form of the absolute square of the wavefunction – it is a table of probabilities (diagonal terms) and correlation coefficients (all other terms).

Thermal equilibrium

In an ensemble of identical systems in thermal equilibrium, there are no correlations and the probability of finding a system in the energy level n is given by Boltzmann's law:

$$\hat{H}\left|\psi_n\right\rangle = E_n\left|\psi_n\right\rangle, \qquad p_n = \frac{\exp\left(-E_n/kT\right)}{\sum_m \exp\left(-E_m/kT\right)} \tag{9}$$

These probabilities are the diagonal terms of the density matrix, therefore:

$$\boldsymbol{\rho}_{\text{eq}} = \frac{\sum_n \left|\psi_n\right\rangle \exp\left(-E_n/kT\right)\left\langle\psi_n\right|}{\sum_n \exp\left(-E_n/kT\right)} = \frac{\exp\left(-\mathbf{H}/kT\right)}{\text{Tr}\left[\exp\left(-\mathbf{H}/kT\right)\right]} \approx \frac{\mathbf{1}-\mathbf{H}/kT}{\mathbf{Tr}[\mathbf{1}]} \tag{10}$$

The high temperature approximation is valid when $\left\|\hat{H}\right\| \ll kT$, where the norm of the Hamiltonian is defined as its largest absolute eigenvalue. The unit matrix is inconsequential: it commutes with the Hamiltonian and does not influence the observables. In practical calculations it is commonly ignored.

Making Hamiltonians

To make a spin Hamiltonian, all magnetic interactions should be written down as classical physics expressions, and then magnetic moments replaced with Pauli matrices:

$$\mu_{\text{X}} \rightarrow \gamma\boldsymbol{\sigma}_{\text{X}} \qquad \mu_{\text{Y}} \rightarrow \gamma\boldsymbol{\sigma}_{\text{Y}} \qquad \mu_{\text{Z}} \rightarrow \gamma\boldsymbol{\sigma}_{\text{Z}} \tag{11}$$

where γ is the magnetogyric ratio, and (for spin-1/2 particles):

$$\boldsymbol{\sigma}_{\text{X}} = \begin{pmatrix} 0 & 1/2 \\ 1/2 & 0 \end{pmatrix} \qquad \boldsymbol{\sigma}_{\text{Y}} = \begin{pmatrix} 0 & -i/2 \\ i/2 & 0 \end{pmatrix} \qquad \boldsymbol{\sigma}_{\text{Z}} = \begin{pmatrix} 1/2 & 0 \\ 0 & -1/2 \end{pmatrix} \tag{12}$$

When multiple spins are present, the operators for individual spins are Kronecker products with a unit matrix on all other spins. For a two-spin system:

$$\mathbf{S}_{\text{Z}}^{(1)} = \boldsymbol{\sigma}_{\text{Z}} \otimes \mathbf{1} \qquad \mathbf{S}_{\text{Z}}^{(2)} = \mathbf{1} \otimes \boldsymbol{\sigma}_{\text{Z}} \tag{13}$$

Because the classical expressions for all magnetic interactions are known, the process is simple, *e.g.*:

$$E = -\mu_{\text{Z}}^{(1)}\mu_{\text{Z}}^{(2)} \qquad \Rightarrow \qquad \mathbf{H} = -\gamma_1\gamma_2\mathbf{S}_{\text{Z}}^{(1)}\mathbf{S}_{\text{Z}}^{(2)} \tag{14}$$

All NMR Hamiltonians are tabulated in the literature. In practice, the process is performed automatically by a computer and we simply get a matrix in the end.

Writing your own simulations

Since about year 2000, brain time has been much more expensive than CPU time. The programming language that takes the smallest amount of human effort to get an NMR simulation going is *Matlab*. Our first stage is to define Pauli matrices:

```
% Define Pauli matrices
sigma_x=[0,   1/2;  1/2, 0];
sigma_y=[0, -1i/2; 1i/2, 0];
sigma_z=[1/2,   0; 0, -1/2];
unit=[1 0; 0 1];
```

It is a good idea to check the commutation relations to make sure the matrices are entered correctly:

$$[\boldsymbol{\sigma}_{\text{X}},\boldsymbol{\sigma}_{\text{Y}}] = i\boldsymbol{\sigma}_{\text{Z}}; \quad [\boldsymbol{\sigma}_{\text{Z}},\boldsymbol{\sigma}_{\text{X}}] = i\boldsymbol{\sigma}_{\text{Y}}; \quad [\boldsymbol{\sigma}_{\text{Y}},\boldsymbol{\sigma}_{\text{Z}}] = i\boldsymbol{\sigma}_{\text{X}} \tag{15}$$

The second stage is to use Kronecker products to generate two-spin operators:

```
% Build two-spin operators
Lx=kron(sigma_x,unit); Sx=kron(unit,sigma_x);
Ly=kron(sigma_y,unit); Sy=kron(unit,sigma_y);
Lz=kron(sigma_z,unit); Sz=kron(unit,sigma_z);
```

The third stage is to build the Hamiltonian. Let us specify a strongly coupled two-spin system:

$$H = \omega_{\mathrm{L}}\mathbf{L}_{\mathrm{Z}} + \omega_{\mathrm{S}}\mathbf{S}_{\mathrm{Z}} + 2\pi J\left(\mathbf{L}_{\mathrm{X}}\mathbf{S}_{\mathrm{X}} + \mathbf{L}_{\mathrm{Y}}\mathbf{S}_{\mathrm{Y}} + \mathbf{L}_{\mathrm{Z}}\mathbf{S}_{\mathrm{Z}}\right) \tag{16}$$

All interaction amplitudes must be in radians per second:

```
% Build the Hamiltonian
omega_L=2*pi*200; % 200 Hz
omega_S=2*pi*400; % 400 Hz
omega_J=2*pi*40;  % 40 Hz
H=omega_L*Lz+omega_S*Sz+omega_J*(Lx*Sx+Ly*Sy+Lz*Sz);
```

Let us take the initial condition to be:

$$\boldsymbol{\rho}(0) = \mathbf{L}_{\mathrm{X}} + \mathbf{S}_{\mathrm{X}} \tag{17}$$

This corresponds to both spins sitting on the X axis, as they would after a 90-degree radiofrequency pulse. For the observable operator, we will pick the total X magnetization for the real part and the total Y magnetization for the imaginary part:

$$\mathbf{O} = \left(\mathbf{L}_{\mathrm{X}} + \mathbf{S}_{\mathrm{X}}\right) + i\left(\mathbf{L}_{\mathrm{Y}} + \mathbf{S}_{\mathrm{Y}}\right) \tag{18}$$

In *Matlab* language, this would be:

```
% Initial and detection state
rho=Lx+Sx; coil=(Lx+Sx)+1i*(Ly+Sy);
```

where the term "coil" was used for the observable operator.

Before computing the exponentials in Equation (5), we need to decide the time step of the simulation. All frequencies must be adequately digitised: to observe the Nyquist condition, we need two points per period of the fastest oscillation. We therefore need to calculate the highest frequency present in the system. That is actually the definition of 2-norm, and therefore:

```
% Decide the time step
time_step=0.5/norm(H,2);
```

We can now compute the matrix exponentials; they are called *propagators*:

```
% Build the step propagator
P=expm(-1i*H*time_step);
```

This operation is slow, but only one propagator needs to be calculated because the much faster conjugate-transpose operation has the effect of replacing every instance of "i" with "$-i$".

$$\left(e^{-i\mathbf{H}\Delta t}\right)^{\dagger} = e^{+i\mathbf{H}\Delta t} \tag{19}$$

The system will now be propagated forward step by step, and the observable calculated and stored for each step. This is accomplished by the use of the "for" loop:

```
% Time evolution with for 2048 steps
fid=zeros(1,2048);
for n=1:2048
    fid(n)=trace(rho*coil);
    rho=P*rho*P';
end
```

Our free induction decay is now recorded. Everything from this point on is essentially NMR data processing. The signal needs to be multiplied by a decaying function (this is called apodisation):

```
% Apodization
window_function=exp(-5*linspace(0,1,2048));
fid=fid.*window_function;
```

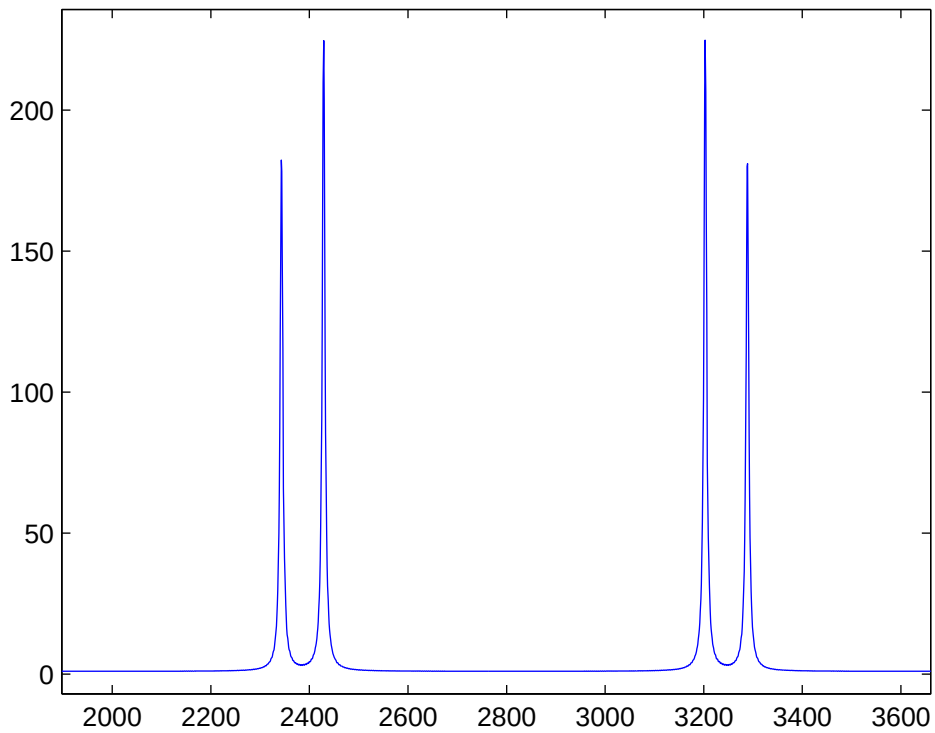
The `linspace` function returns an array of 2048 numbers spaced equally between 0 and 1. Calling an exponential of this array computes the exponential of every element. The element-by-element multiplication operation (denoted by dot-star) then multiplies every element of the fid array by the corresponding element of the window function array. This compact notation avoids quite a few loops.

Finally, we need to run a Fourier transform (with zerofilling to 8196 points) and plot the result

```
% Fourier transform with zerofill
spectrum=fftshift(fft(fid,8196));

% Plotting
plot(real(spectrum));
```

The `fftshift` function moves the zero frequency from the edge (engineering convention) to the centre (physics convention) of the spectrum. The following picture is produced, with the instantly recognizable two-spin NMR spectrum:



The default X axis in Matlab is point count; a proper frequency axis may be generated by noting that the largest positive and negative frequency present in the trajectory is $1/\Delta t$ where Δt is the time step.