Computer practical 1 – pulse sequence benchmarking

<u>Instructions</u>: below is a detailed walkthrough that shows examples of code that performs specific tasks. Read the text, take a look at the code, look at the Spinach manual pages of the functions involved, and then write and run your own version of that code. More spins? Different nuclei? You decide.

A ^{19}F-^{1}H heteronuclear Overhauser effect experiment (HOESY) may be run in two superficially equivalent ways. We may start with ^{1}H magnetisation, evolve it in the indirect dimension, transfer it to ^{19}F, and then detect the free induction decay on ^{19}F. Alternatively, we could start with ^{19}F magnetisation and eventually detect ^{1}H. The objective of this practical is to save some time to the experimental team – as we shall see, one of the two ways is preferable in protein systems.

1. **Start *Matlab* and type 'edit hello_world.m'. This will open the code editor and create an empty file with 'hello_world.m' as the name. At the top of the file type 'function hello_world()', make a few empty lines and put 'end' somewhere underneath.**

Never run *Spinach* simulations without this 'function – end' pair. Its purpose is to make sure that variables are cleared when the simulation finishes. Having variables linger in the workspace is not a good idea because of the inevitable confusion it causes when forgotten variables have some effect.

2. **Specify the magnet field, a proton-fluorine pair, and some random chemical shifts:**

```
% Magnet field
sys.magnet=14.1;

% Isotopes
sys.isotopes={'1H','19F'};

% Isotropic chemical shifts
inter.zeeman.scalar={2.0 20.0};
```

Zero chemical shifts are not recommended because many pulse sequence artefacts arise from resonance offset errors; we would like to be able see such artefacts.

HOESY is a relaxation-driven experiment. We must therefore specify all interactions that may be responsible for spin relaxation in this system. In proton-fluorine pairs at high field, the dominant relaxation mechanisms are dipolar and CSA. The latter is particularly prominent for fluorine in aromatic systems because ^{19}F CSA can be in the hundreds of ppm. Typical proton CSA is 10 ppm, and a typical distance (for example, in 3-fluorotyrosine) is around 2.5 Angstrom. *Spinach* automatically accounts for cross-correlations, in this case DD-CSA cross-correlation.

3. **Specify CSA tensors and atomic coordinates:**

```
% Specify coordinates and CSAs
inter.coordinates{1}=[0.00 0.00 0.00];
inter.coordinates{2}=[2.50 0.00 0.00];
inter.zeeman.eigs{1}=[  5.0   5.0  -10.0];
inter.zeeman.eigs{2}=[100.0 100.0 -200.0];
inter.zeeman.euler{1}=[0   pi/2  0];
inter.zeeman.euler{2}=[0   0     0];
```

There are many ways to specify interactions in Spinach (see the manual). In this case, it will obtain the dipolar interactions from the atomic coordinates (Angstroms). Chemical shift tensors are specified using eigenvalues (ppm) and Euler angles (radians). Note that the tensors are traceless – we had already specified isotropic chemical shifts in Item 2.

4. **Specify relaxation theory parameters:**

```
% Relaxation theory
inter.relaxation={'redfield'};
inter.rlx_keep='secular';
inter.equilibrium='levante';
inter.temperature=298;
inter.tau_c={5e-9};
```

We must use Redfield relaxation theory. Alternatives do exist (for example, SLE), but they are less easy to use. Because a rotating frame simulation is performed, only the secular part of the relaxation superoperator must be kept. We must also update the relaxation superoperator to return the system to the thermal equilibrium at 298 Kelvin (inhomogeneous master equation formalism is used). Finally, a rotational correlation time typical of a large protein (5 ns) is specified. Every line in the code block above is a separate lecture in our *Spin Dynamics* course – this is deep water.

5. **Specify the formalism, the approximation level, and call Spinach housekeeping:**

```
% Formalism and approximation
bas.formalism='sphten-liouv';
bas.approximation='none';

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

Spinach is most efficient when it uses irreducible spherical tensors (particular combinations of Pauli matrices) rather than Pauli matrices as the basis set. The reason is that these combinations transform in particularly simple ways under rotations, which are ubiquitous in spin dynamics. Because this spin system is small, no approximations are needed.

6. **Run the file. The two housekeeping functions will produce copious console output; read it carefully. If there are any errors or omissions in the input, *Spinach* will point them out.**

7. **Request the basic dynamics generators from *Spinach*. These include Hamiltonian superoperator, relaxation superoperator, and pulse operators on both spins:**

```
 % Get Hamiltonian
H=hamiltonian(assume(spin_system,'nmr'));

% Get relaxation superoperator
R=relaxation(spin_system);

% Get pulse operators
Hp=operator(spin_system,'L+','1H');
Hx=(Hp+Hp')/2; Hy=(Hp-Hp')/2i;
Fp=operator(spin_system,'L+','19F');
Fx=(Hp+Hp')/2; Fy=(Fp-Fp')/2i;
```

Note the 'nmr' assumptions in the Hamiltonian call. Spinach supports all types of magnetic resonance, and most of them operate in different rotating frame settings from NMR – the 'assume' command applies high-field nuclear magnetic resonance assumptions.

Spinach kernel does not use Cartesian spin operators internally – it uses {+,–,z} basis. Therefore X and Y Cartesian spin operators are assembled from raising and lowering operators.

Further console output will be produced by 'hamiltonian' and 'relaxation' functions. Read that output carefully, particularly the sections detailing the various assumptions. You will see from the console output that the 'relaxation' function makes its own calls to the 'hamiltonian' function because it needs the laboratory frame Hamiltonian for its internal purposes.

8. Request the thermal equilibrium state:

```
% Get the equilibrium state
H0=hamiltonian(assume(spin_system,'labframe'),'left');
rho_eq=equilibrium(spin_system,H0);
```

The Hamiltonian used to build the thermal equilibrium must be the laboratory frame Hamiltonian. The 'left' parameter tells Spinach that we want a left side product, rather than a commutation, superoperator – this is dictated by the mathematics of the Boltzmann equilibrium.

9. Inspect the thermal equilibrium state:

```
% Inspect thermal equilibrium state
stateinfo(spin_system,rho_eq,5);
```

The last parameter refers to the number of the most populated states to report on. The first three lines of the output should be as follows:

```
 ....     ....       +1.000e+00    1
(1,+0)    ....       +4.834e-05    9
 ....    (1,+0)      +4.550e-05    3
```

where the first two columns refer to the product state components on the two spins, dots indicate a unit operator, the brackets are (L,M) notation for irreducible spherical tensors (L=1 and M=0 correspond to Lz operator), the floating point numbers are populations, and the integers are positions of the corresponding state in the basis set. We can see a unit population in the unit state (always the case because the trace of the density matrix is always 1) and small populations in Hz and Fz as befits the room temperature, with slightly less on fluorine.

10. Request detection states:

```
% Get detection states
coil_h=state(spin_system,'L+','1H','exact');
coil_f=state(spin_system,'L+','19F','exact');
```

The last parameter refers to state normalisation (specifying 'cheap' makes large-scale calculations faster, but consistent norms would not be guaranteed).

11. Compare pulse-acquire FIDs on the two nuclei:

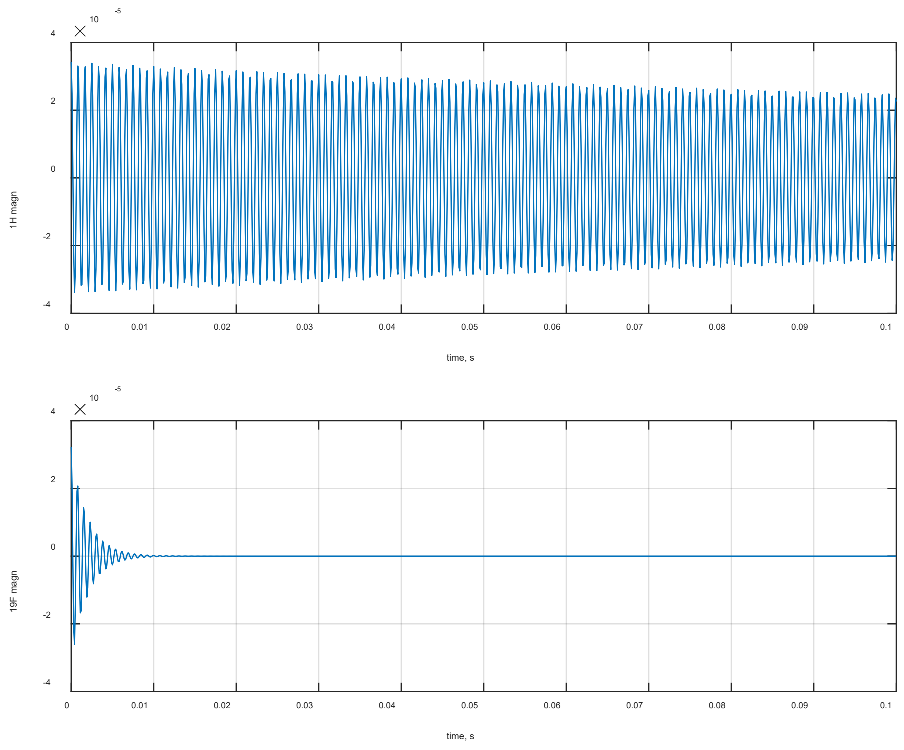
```
% Run pulse-acquire on 1H
rho=step(spin_system,Hy,rho_eq,pi/2);
fid_h=evolution(spin_system,H+1i*R,coil_h,rho,1e-4,1e3,'observable');

% Run pulse-acquire on 19F
rho=step(spin_system,Fy,rho_eq,pi/2);
fid_f=evolution(spin_system,H+1i*R,coil_f,rho,1e-4,1e3,'observable');
```

Here the step command executes one-off events, such as hard pulses. The evolution command returns the results of various types of time evolution – in this case, the observables corresponding to the detection states provided. In this case, 1000 steps is taken, 100 microseconds each. See the manual pages for 'step' and 'evolution' for further information.

12. Plot the two FIDs:

```
% Compare the fids
figure(); t_axis=linspace(0,0.1,1001)';
subplot(2,1,1); plot(t_axis,real(fid_h));
ylabel('1H magn'); xlabel('time, s'); grid on;
subplot(2,1,2); plot(t_axis,real(fid_f));
ylabel('19F magn'); xlabel('time, s'); grid on;
```

This highlights the difference in the relaxation behaviour of protons and fluorine – the transverse fluorine magnetisation is gone in ten milliseconds.



13. Run relaxation theory analysis of the spin system:

```
% Print relaxation theory report
relaxan(spin_system);
```

The console printout confirms very rapid transverse relaxation of ^{19}F nuclei:

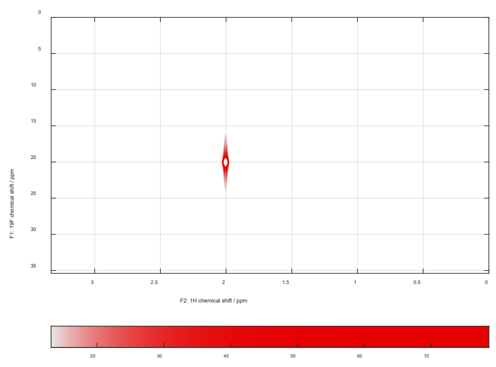
```
=================================================================
Number  Isotope  R1(Hz)       R2(Hz)       T1(s)        T2(s)
=================================================================
1       1H       4.836e-01    3.742e+00    2.068e+00    2.672e-01
2       19F      2.871e+00    5.078e+02    3.483e-01    1.969e-03
=================================================================
```

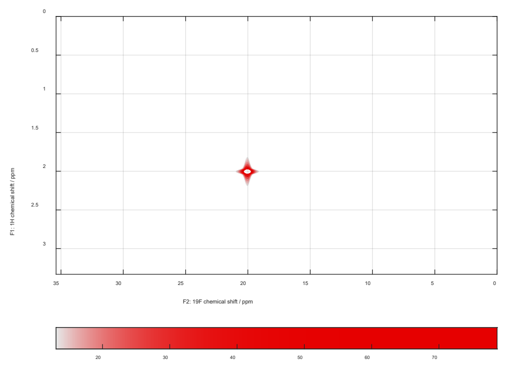
14. **Open the HOESY pulse sequence ('edit hoesy.m') and inspect the source code, particularly the initial condition. Is that a quantitative simulation? Why did the authors not bother making that simulation quantitative? How would you go about making it quantitative?**

15. **Run the 1H-19F version and 19F-1H version of the HOESY sequence. The 1H-19F version of the code is given below:**

```
% Set 1H-19F HOESY parameters
parameters.tmix=0.5;
parameters.sweep=[2000 20000];
parameters.offset=[1000 10000];
parameters.npoints=[128 128];
parameters.zerofill=[512 512];
parameters.spins={'1H','19F'};
parameters.decouple_f1={};
parameters.decouple_f2={};
parameters.axis_units='ppm';

% Run 1H-19F HOESY simulation and processing
fid=liquid(spin_system,@hoesy,parameters,'nmr');
fid.cos=apodization(fid.cos,'sqcosbell-2d');
fid.sin=apodization(fid.sin,'sqcosbell-2d');
f1_cos=real(fftshift(fft(fid.cos,parameters.zerofill(2),1),1));
f1_sin=real(fftshift(fft(fid.sin,parameters.zerofill(2),1),1));
spec=fftshift(fft(f1_cos-1i*f1_sin,parameters.zerofill(1),2),2);

% Plotting
figure(); plot_2d(spin_system,abs(spec),parameters,...
                  20,[0.05 0.25 0.05 0.25],2,256,6,'positive');
```

How would you modify this to make a 19F-1H version? The parameters needed here are all listed in the manual page for HOESY. The sequence uses hypercomplex processing.





Is the difference in signal intensity significant? Why? What would have changed if the pulse sequence had a coherence transfer or a multiplet refocusing stage for a 10 Hz *J*-coupling? Introduce a 0.025 second free evolution delay after the first pulse in HOESY (use the step function to evolve under H+1i*R) and observe the effect.

16. **Find out (empirically, by changing B_0 field) how transverse ^{19}F relaxation rate depends on B_0. How would you expect it to depend on the B_0 field?**