Computer practical: time domain electron spin resonance in *Spinach*

This practical is about setting up DEER simulations in the presence of analytically intractable complications: orientation selection, more than two electron spins, exchange coupling, zero-field splitting, significant non-secular dipolar terms, *etc.* You will find a large number of other examples in *examples/esr_solids* directory, and a number of pre-programmed pulse sequences in *experiments* directory.

In the examples below, feel free to modify the parameters to bring the simulations closer to your own practical usage scenarios.

1. Installation

If *Spinach 2.8* is not installed already, download the latest version from https://spindynamics.org, unpack the zip file and follow the installation instructions. It is important to set *Matlab* path correctly. Alternatively, log in to NMRBox where *Spinach* is installed and configured already.

2. Frequency domain ESR spectra and pulse holes

At the first stage of a DEER simulation, it is necessary to examine the frequency swept ESR spectrum and to choose pump and probe frequency, as well as to calibrate soft pulses. We start by creating a matlab function file (don't forget the `function ... end` keywords), specifying the magnet field (in pulsed ESR it normally stays constant), and spin system parameters:

```
% Isotopes
sys.isotopes={'E','14N'};                                  % COPY AND PASTE
                                                           % CODE FROM HERE,
% Magnet field                                             % DO NOT RE-TYPE
sys.magnet=3.35;

% Interactions
inter.zeeman.matrix=cell(1,2);
inter.zeeman.matrix{1}=[2.01045  0.00000  0.00000
                        0.00000  2.00641  0.00000
                        0.00000  0.00000  2.00211];
inter.coupling.matrix=cell(2,2);
inter.coupling.matrix{1,2}=[1.2356  0.0000  0.6322
                            0.0000  1.1266  0.0000
                            0.6322  0.0000  8.2230]*1e7;
```

Here, a magnet (W-band, 3.35 Tesla), a common nitroxide *g*-tensor from the literature, and a common ^{14}N hyperfine coupling (in Hz) were specified. Quadrupolar interaction on ^{14}N may also be added if necessary – it usually does not influence DEER dynamics. A full description of how spin systems are specified in *Spinach* is available in the online manual (https://spindynamics.org/wiki).

Next we specify the formalism and the basis set. Spinach runs most efficiently in Liouville space when the basis operators are irreducible spherical tensors (particular combinations of spin operators with elegant rotation and Zeeman evolution properties):

```
% Basis set
bas.formalism='sphten-liouv';
bas.approximation='none';
```

We then call housekeeping functions that perform input data analysis and pre-processing:

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

Run the file. Housekeeping functions will print diagnostic output to *Matlab* console. Read it carefully, it contains a detailed summary of what *Spinach* thinks the user has supplied. If the input is found to be internally inconsistent, an informative error message will be printed to the console in red.

At this point, we have supplied spin system information to *Spinach*. It now needs to know the experiment parameters. We will use the *holeburn* experiment (which simulates the hole that a particular soft pulse burns in the powder pattern) within the *powder* context (which handles powder averages and rotating frames). Take a good look at the online manual pages for *holeburn.m* and *powder.m* functions. They support a large number of parameters, of which we will only need a few:

```
% Sequence parameters
parameters.spins={'E'};                          % Working spins
parameters.rho0=state(spin_system,'Lz','E');     % Initial condition
parameters.coil=state(spin_system,'L+','E');     % Detection state
parameters.decouple={};                          % No decoupling
parameters.offset=-2e8;                          % Transmitter offset, Hz
parameters.sweep=8e8;                            % Sweep width, Hz
parameters.npoints=128;                          % FID point count
parameters.zerofill=512;                         % Zerofilling
parameters.axis_units='MHz';                     % Axis units
parameters.grid='rep_2ang_6400pts_sph';          % Spherical grid
parameters.derivative=0;                         % Plot zeroth derivative
parameters.invert_axis=0;                        % Plot from left to right
```

Parameter names are all self-explanatory. Note that signals that end up outside the sweep width will get reflected back – a simulation cannot impose a frequency bandpass filter of the same kind that the spectrometer does. The offset is with respect to the free electron Zeeman frequency. Note also the huge powder averaging grid – time domain ESR simulations are expensive.

The next stage is to specify the parameters of the soft pulse. We will run a 100 nanosecond pulse at the frequency offset of –300 MHz from the free electron Zeeman frequency, with the initial microwave phase of $\pi/2$, and $2r + 1 = 5$ points in the microwave phase grid:

```
% Soft pulse parameters
parameters.pulse_rnk=2;
parameters.pulse_phi=pi/2;
parameters.method='expm';
parameters.pulse_dur=100e-9;
parameters.pulse_pwr=2*pi*10e6;
parameters.pulse_frq=-300e6;
```

With all parameters now in place, we can call the *holeburn* pulse sequence from the *powder* context, with the rotating frame assumptions set to 'esr'. We also need a reference spectrum where the amplitude of the soft pulse is set to zero:

```
% Soft pulse simulation
fid_a=powder(spin_system,@holeburn,parameters,'esr');

% Simulation with soft pulse off
parameters.pulse_pwr=0;
fid_b=powder(spin_system,@holeburn,parameters,'esr');
```

This is a time-domain simulation: first the soft pulse, then the ideal 90-degrees excitation pulse, then free induction decay detection of the kind that is done in NMR. We will need to apply apodisation to the free induction decays, to Fourier transform them, and then to plot the resulting spectra:

```
% Apodization
fid_a=apodization(fid_a,'exp-1d',6);
fid_b=apodization(fid_b,'exp-1d',6);
```

```
% Fourier transform
spectrum_a=fftshift(fft(fid_a,parameters.zerofill));
spectrum_b=fftshift(fft(fid_b,parameters.zerofill));

% Plotting
figure(); hold on;
plot_1d(spin_system,real(spectrum_a),parameters,'r-');
plot_1d(spin_system,real(spectrum_b),parameters,'b-');
klegend({'soft pulse','reference'});
```

The output, for two different offset frequencies (it will take a few minutes, use a smaller spherical grid if you want a quick and dirty result), is shown in Figure 1. Note different excitation efficiencies at different frequencies even though pulse duration and amplitude are the same.

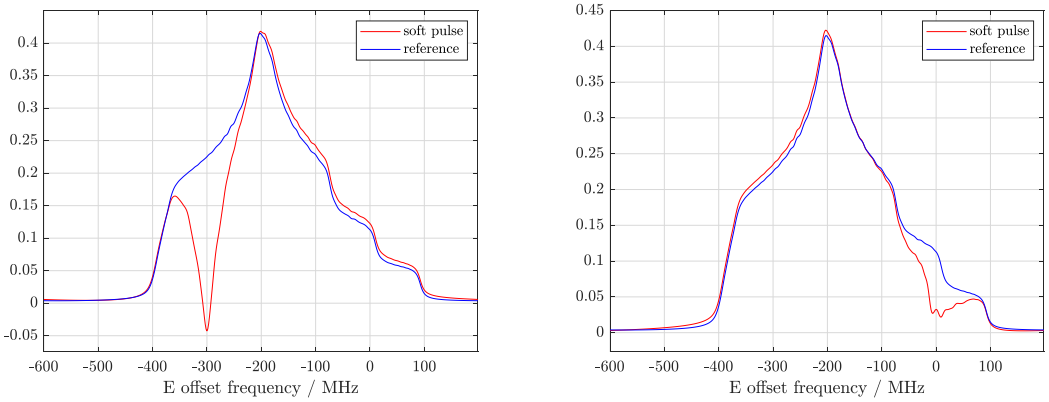

Figure 1. *Holes burned in the nitroxide powder pattern by the pulses with the parameters specified above at the offset frequencies of –300 MHz (left) and 0 MHz (right) relative to the electron Zeeman frequency.*

Try different pulse frequencies, amplitudes, initial phases, and complications (for example, zero-field splitting). For quick estimates, the number of points in the powder averaging grid may be reduced, see the manual for the list of spherical integration grids that are available in *Spinach*.

3. Orientation-selective DEER simulation

Consider a simple system at X-band with two electrons at a distance of 20 Angstrom with non-overlapping signals and *g*-tensors at a specific orientation relative to one another:

```
% Magnet field
sys.magnet=0.3451805;

% Isotopes
sys.isotopes={'E','E'};

% Zeeman interactions
inter.zeeman.eigs=cell(1,2);
inter.zeeman.euler=cell(1,2);
inter.zeeman.eigs{1}=[2.284 2.123 2.075];
inter.zeeman.euler{1}=[135 90 45]*(pi/180);
inter.zeeman.eigs{2}=[2.035 2.013 1.975];
inter.zeeman.euler{2}=[30 60 120]*(pi/180);

% Coordinates (Angstrom)
inter.coordinates=cell(2,1);
inter.coordinates{1}=[0.00  0.00 0.00];
inter.coordinates{2}=[20.00 0.00 0.00];
```

The usual formalism selection and housekeeping commands go next:

```
% Basis set
```

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

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

Spinach has an experiment script (*deer_3p_soft_hole.m*) that performs pulse hole diagnostics. After looking through the online manual, we find that the following parameters are necessary:

```
% Sequence parameters
parameters.spins={'E'};                          % Working spins
parameters.rho0=state(spin_system,'Lz','E');     % Initial condition
parameters.coil=state(spin_system,'L+','E');     % Detection state
parameters.grid='rep_2ang_6400pts_sph';          % Powder grid
parameters.method='expm';                        % Propagation method

% EPR spectrum parameters
parameters.offset=-4e8;                          % Rotating frame offset
parameters.sweep=3e9;                            % Sweep width
parameters.npoints=256;                          % FID point count
parameters.zerofill=2048;                        % Zerofilling
parameters.axis_units='GHz-labframe';            % Axis units
parameters.derivative=0;                         % Plotting options
parameters.invert_axis=1;                        % Plotting options

% 3-pulse DEER sequence parameters
parameters.pulse_rnk=[2 2 2];                       % Phase grid ranks
parameters.pulse_dur=[20e-9 50e-9 40e-9];           % Durations
parameters.pulse_phi=[pi/2 pi/2 pi/2];              % Phases
parameters.pulse_pwr=2*pi*[8e6 8e6 8e6];            % Powers (rad/s)
parameters.pulse_frq=[9.720e9 10.255e9 9.720e9]; % Frequencies
```

Frequencies, amplitudes, and durations of the microwave pulses are obtained from the same kind of analysis as was described in the previous section. We can now launch the pulse hole diagnostics function from the *powder* context:

```
% ESR hole burning simulations to locate the holes
fids=powder(spin_system,@deer_3p_soft_hole,parameters,'deer');
```

The call takes a few minutes and returns four FIDs as columns of a matrix. We need to apodise and Fourier transform them before plotting:

```
% Apodisation and Fourier transform
fids=apodization(fids,'crisp-1d');
specs=fftshift(fft(fids,parameters.zerofill,1),1);

% Plotting
figure(); plot_1d(spin_system,real(specs(:,1)),parameters,'r-');
figure(); plot_1d(spin_system,real(specs(:,2)),parameters,'r-');
figure(); plot_1d(spin_system,real(specs(:,3)),parameters,'r-');
figure(); plot_1d(spin_system,real(specs(:,4)),parameters,'r-');
```

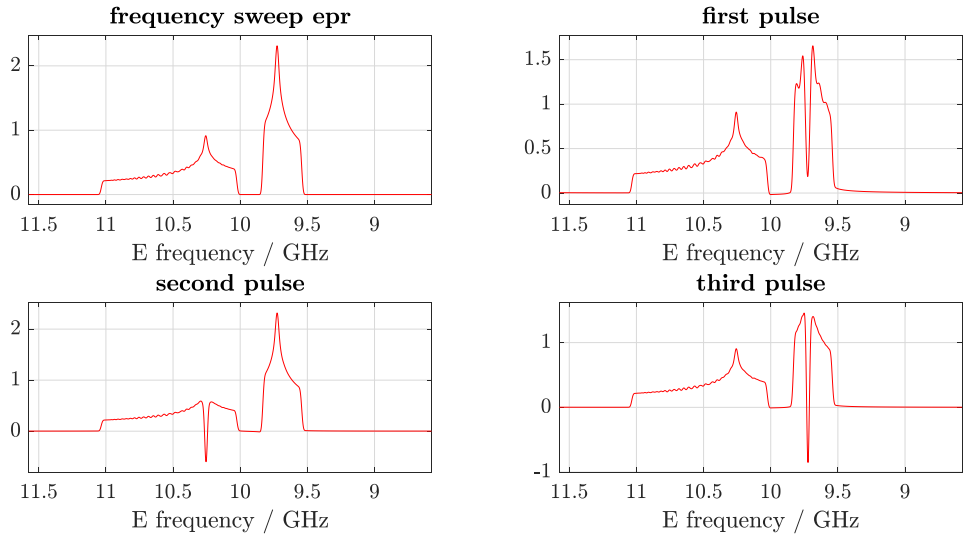

Figure 2. *Pulse hole diagnostics plots for orientation-selective DEER simulations.* ***Top left:*** *frequency-swept ESR spectrum.* ***Top right:*** *the effect of the first probe pulse in the DEER sequence.* ***Bottom left:*** *the effect of the pump pulse in the DEER sequence.* ***Top right:*** *the effect of the second probe pulse in the DEER sequence.*

The next step is to perform the DEER simulation and to obtain a stack of spin echoes. *Spinach* experiment file is called *deer_3p_soft_deer.m*; open it and take a look at the source code. You will see straightforward pulse and evolution commands, as well as code comments and documentation.

We proceed to specify DEER sequence timing parameters:

```
% DEER echo timing parameters
parameters.p1_p3_gap=1e-6;       % Gap between P1 and P3
parameters.p2_nsteps=100;        % Number of steps in P2 position
parameters.echo_time=100e-9;     % Time to sample around the echo
parameters.echo_npts=100;        % Number of points in the echo
```

The sequence is also called from the *powder* context; it returns a stack of echo signals:

```
% DEER simulation
echo_stack=powder(spin_system,@deer_3p_soft_deer,parameters,'deer');

% Time axis for the echo window
echo_axis=1e9*linspace(-parameters.echo_time/2,...
                        parameters.echo_time/2,parameters.echo_npts+1);

% Time axis for the DEER trace
deer_axis=1e6*linspace(0,parameters.p1_p3_gap,parameters.p2_nsteps+1);
```

The easiest way to analyse the echo stack is by singular value decomposition, which returns the dominant echoes in left singular vectors and their modulation in the right singular vectors:

```
% DEER echo stack analysis
[deer_echoes,deer_sigmas,deer_traces]=svd(echo_stack);
deer_echoes=deer_echoes*deer_sigmas/deer_traces(1);
deer_traces=deer_traces*deer_sigmas/deer_traces(1);
```

There is usually only one principal echo, but we can plot the first three singular pairs just in case:

```
% Plot echo components
figure(); plot(echo_axis,real(deer_echoes(:,1:3)));
kylabel('echo, real channel'); kgrid;
kxlabel('echo window, ns'); axis tight;
klegend({'u_1\cdot\sigma_1',...
         'u_2\cdot\sigma_2',...
         'u_3\cdot\sigma_3'});
```

```
ktitle('principal components of the echo stack');

% Plot DEER components
figure(); plot(deer_axis,real(deer_traces(:,1:3)));
kylabel('echo, real channel'); axis tight;
kxlabel('2nd pulse insertion point, \mus');
klegend({'v_1\cdot\sigma_1',...
         'v_2\cdot\sigma_2',...
         'v_3\cdot\sigma_3'}); kgrid;
ktitle('principal components of the echo stack');
```

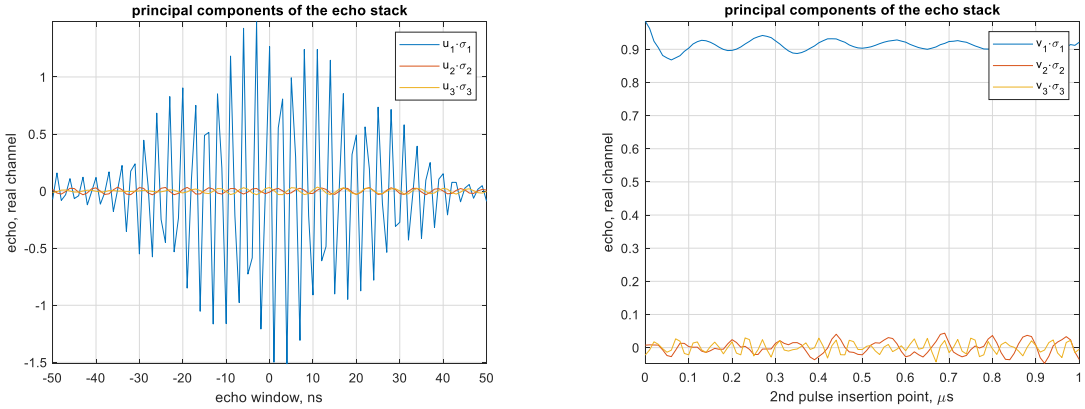

Figure 3. *First three principal components of the DEER echo modulation stack. Because many spin system parameter distributions are not modelled in this simplified simulation, the echo is long and intense, with many modulation periods visible.*

4. Exploration

(a) Move the pump and the probe frequencies close together and observe the interference caused by the flip-flop terms in the dipolar operator. The effect may be suppressed artificially by setting the assumptions to 'deer-zz'.

(b) Reduce the spherical averaging grid size and observe the effect on the echo stack.

(c) Add a third electron and observe the effect on the echo stack and the simulation time.