SpinTool Getting Started Guide

By Assaf Tal, 27/12/2016

Getting Started

Installation of SpinTool should work on pretty much any version of MATLAB. All you have to do is unzip and add all directories and subdirectories to the MATLAB path (right click the directory in MATLAB and select "Add To Path → Add Folders and Subfolders").

Recompiling Binaries

SpinTool uses MEX files, which are compiled c files, to speed up simulations. Precompiled binaries are included for Windows 64 bit platforms. Those of you running on other platforms, such as OSX or Linux or Win 32 bits, will need to recompile. To do so, you will need to install an appropriate C++ compiler and link it to MATLAB. It's not as bad as it sounds. In MATLAB, first type

```
mex -setup
```

This will tell you what you have installed. If you already have a C++ compiler installed, great! Go to the

```
Core64\Relaxation
```

subdirectory within SpinTool and type in

```
mex ApplyZRotation.c
mex ApplyPulseRelaxDiagnostics.c
mex ApplyPulseRelax.c
mex AcquireRelax.c
mex AcquirePulseRelax.c
mex ApplyPulsePerfect.c
```

Basically, all files that end with .c. This should work flawlessly.

If you don't have a compiler installed, typing mex –setup will provide a link to the MATLAB site where you will be advised on what (freely available) compilers you can download and install. It's fairly painless but might take a bit of work to complete. After you install, type

```
mex –setup C++
```

and MATLAB should automatically find the installed compiler (assuming you followed their instructions on what to download!).

Example: Create and Test a Hyperbolic Secant Inversion Pulse

In this example we create a 180 hyperbolic secant pulse with max B1 of 1 kHz and 30% adiabaticity (so it inverts perfectly already at 0.7 kHz).

Create the pulse:

- p = PulseCreateSechSiemens(1.0, 0.7);

Plot its frequency response between ±1 kHz using 501 points with initial magnetization along z (M=[0;0;1])

- PlotPulseFreqResponse(p, [0;0;1], -1, 1, 501, 'mz')

Export to Siemens .h include format, using the filename 'MyPulse.h'. The plot above reveals the bandwidth is about 1.5 kHz. The flip angle is set to 180 degrees:

- ExportPulseToIncludeFile(p, 'MyPulse', 1.5, 180, 'This is a comment');

Hint: use ExportPulseToSiemens to export the pulse to a PTA file.

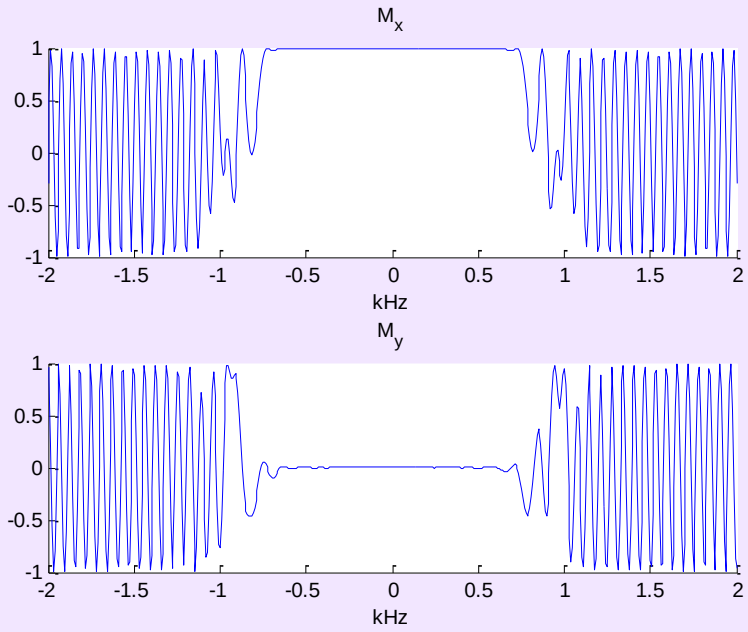
Let's create a sequence of two back-to-back hyperbolic secant pulses:

- mySeq = {p, p};

Now let's plot its frequency response for spins initially along the x-axis. This would be a "LASER-like" effect: two back-to-back adiabatic frequency sweeps will undo each other's non-linear phases and should result in a linear phase in the pass-band. We're going to plot Mx and My:

- PlotSeqFreqResponse(mySeq, [1;0;0], -2, 2, 501, {'mx', 'my'})

This is what will pop up on the screen:



Example: a Very Simple Excite-and-Return Lipid Suppression Scheme

Two pi/2 pulses can be used to "suppress" lipid signals: the spins are excited with a 90 degree pulse, a delay is inserted (say, 1 ms) and then a second pi/2 pulse with a 90-degree phase increment is used to store those spins which have precessed enough so as to shift by 90° phase in the transverse plane. Let's show this using simulation:

Create the sequence:

- watSupSeq = {{'hard', 90, 0}, {'delay', 1}, {'hard', 90, 90}};

Plot |Mxy| as a function of offset (no gradients) between -2 kHz and +2 kHz, using 501 points, for spins starting along the +z axis:

- PlotSeqFreqResponse(watSupSeq, [0;0;1], -2, 2, 501, 'mxy')

Example: A Frequency Encoding Imaging Sequence

In this example we simulate frequency encoding for a 1D boxcar sample. Our excitation is non-selective because we don't need to worry about slice selection (we have no slice!). Our pulse is a simple 90° (hard) – rewinder gradient – acquire (w/ gradient). Our acquisition "dwell time" in k-space is:

$$\delta k = \not\gamma G \delta t$$
$$\Delta k = \not\gamma G T_{acq}$$
$$\not\gamma = \gamma / 2\pi$$

where δt is the actual dwell time, and T_{acq} is the total acquisition time, linked via $T_{acq}=N\cdot\delta t$. According to Fourier rules, these are related to the field of view (FOV) via

$$FOV = \frac{1}{\delta k} = \frac{1}{\not\gamma \cdot G \cdot \delta t}.$$

First, let's define our sample, which will be a constant 20 mm distribution of spins starting out from the z-axis having 0 chemical shift. We'll have 200 spins total with an "infinite" T_1 and T_2:

- spins = InitSpinsRelax(0, 200, 20, [0;0;1], 1e6, 1e6);

Now let's define our gradient, assuming we have a FOV of 100 mm and a spatial resolution of 1 mm, and a dwell time of 10μs (note our system of units: kHz and mm):

- FOV = 100; % mm
- N = 100;
- dt = 0.01; % ms
- gm = 42.576; % kHz/mT
- G = 1000/(FOV*dt*gm); % in mT/m
- Tacq = dt*N;

Now let's define our sequence:

- seq = {{'hard', 90, 270}, {'purge', 0, 0, -G, Tacq/2}, {'acquire', N, 1/dt, 0, 0, G}};

Note the 270° phase on the excitation pulse, which makes the spins tilt to the x-axis (so they have no phase). Now we apply:

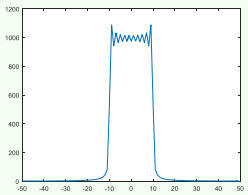
- [spinsOut, fidCellArray] = ApplySequence(spins, seq);

The fidCellArray is a cell array of FIDs acquired throughout the sequence. There was only one acquisition done so we have only one element in the cell array. We extract it, and perform a FT to obtain the image:

- fid = fidCellArray{1};
- img = fftshift(fft(fid));

Finally, we create a suitable x-axis for display ranging from –FOV/2 to FOV/2 in steps of FOV/N and plot the image. Double check it really ranges from -10 to +10 mm!

- xAxis = [-FOV/2:FOV/N:FOV/2-FOV/N];
- plot(xAxis, abs(img))



Spins

Spins are stored in an array of structures:

spins(i).r	mm	3x1 real vector	position
spins(i).M	a.u.	3x1 real vector	magnetization
spins(i).cs	kHz	real number	chemical shift
spins(i).T1	ms	real number	T1 relaxation
spins(i).T2	ms	real number	T2 relaxation
spins(i).M0	a.u.	real number	Equilibrium magn.
spins(i).B1	scaling	complex number	B1 inhomogeneity
spins(i).B0	kHz	real number	B0 constant offset
spins(i).RS	[scaling]	complex number	Receiver inhomogeneity

Spin index

Hint: set B1, RS to 1 for perfect coils. Set B0 to 0 for no B0 inhomogeneity.

Relevant commands:

InitSpinsRelax initializes a 1D array of spins and is probably the "go-to" command to create a structure of spins quickly.

I suggest any (more complicated) configuration of spins in 2D or 3D be created specifically by populating the spin array using a custom built loop.

Pulses

A pulse is a structure housing both arrays and numbers.

pulse.tp	ms	real number	Pulse duration
pulse.RFamp	kHz	1xN real vector	Pulse amplitude
pulse.RFphase	radians	1xN real vector	Pulse phase
pulse.Gx	kHz/mm	1xN real vector	x-gradient
pulse.Gy	kHz/mm	1xN real vector	y-gradient
pulse.Gz	kHz/mm	1xN real vector	z-gradient

There is an implicit assumption that pulse points are equi-spaced.

Relevant commands:

Pulse creation routines start with PulseCreate[...]. Some useful examples are:

1. PulseCreateHard – create a hard pulse
2. PulseCreateBIR4 – create a BIR4 pulse
3. PulseCreateSechSiemens – creates an adiabatic hyperbolic secant inversion pulse. Very useful since you only need to specify the maximal B1 (in kHz) and the threshold B1 (in kHz, < B1Max) from which the pulse will function as desired.
4. PulseCreateSinc – creates a simple & crude sinc pulse.

Sequence

A sequence is a cell array {element 1, element 2, element 3, ... } of "sequence elements" consisting of pulses, delays, hard pulses and so forth. Possible pulse elements are:

1. A pulse object
2. {'delay', d} - applies a delay for d milliseconds
3. {'hard', ang, ph} - applies a hard pulse (0.1 microsecs) with a flip angle ang (deg) and phase ph (deg)
4. {'rect', ang, ph, d} - applies a rect pulse with a flip angle ang (deg) phase ph (deg) and duration d (ms)
5. {'purge', Gx, Gy, Gz, d} - applies a delay d (ms) with gradients Gx, Gy, Gz along the x, y and z axes (in mT/m)
6. {'purgemoment', kx, ky, kz, d} - applies a delay d (ms) with gradient moments kx, ky, kz along the x, y, z axes (in m^(-1))
7. {'killmxy'} – sets the transverse magnetization of the spins to 0.
8. {'acquire', Nt, SW, Gx, Gy, Gz} - acquires an FID with Nt points, SW spectral width (kHz) and gradients Gx, Gy and Gz along the x, y, z axes (in mT/m)

Applying Pulses and Sequences