Magnetic Resonance Spectrum Simulator (MARSS) User Manual

Introduction

MARSS[1] is a software package designed to simulate the output time domain signal (free induction decay) for any arbitrary metabolite and pulse sequence. MARSS can be used to simulate basis sets for quantification, optimization studies or to test novel sequences. **When using MARSS please reference the associated manuscript[1] and if you have any questions please do not hesitate to contact me**. The input required is a .mat file, and requires the mandatory inputs, but has additional options as well that are useful for some certain sequences. MARSS was written in MATLAB and compiled with the 2013b runtime environment. I cannot guarantee forwards or backwards compatibility, but if it does not work with your version please contact me and we will work to resolve it.

We provide the MATLAB executable "MARSS.exe", as well as the README which details how to ensure that the MATLAB executable can be run, and a sample .mat file of input (MARSSinput.mat). After you have filled out the parameters for the input file save the workspace as "MARSSinput.mat" and simply run the executable. Some output will be printed to the command prompt, or errors as appropriate. Some spin systems can take a very long amount of time (days), whereas some can be done in a few seconds or minutes. This relies upon whether the 1D projection method and the propagator method can be used for the particular pulse sequence, as explained in the manuscript.

MARSS will output three folders: 1) "IndividualSpins_for_yourfilename", where "yourfilename" is what is input in the MARSSinput.mat. Inside this folder will be a .mat for each specified metabolite, and within each .mat is a struct called exptDat. Exptdat has the fields sf (center frequency in MHz), sw_h (receive frequency in Hz), nspecC (number of acquired points) and fid (the simulated FID). The size of fid will be nspecC by Nspins, where Nspins is the number of spins for this metabolite (for example 6 for GABA). This format could be useful because in principle each moiety can have different T1 or T2 values, and individual spins could thus be scaled accordingly. The second folder MARSS outputs is 2) "SummedSpins_for_yourfilename". Within this folder contains a .mat file for each metabolite which are identical to "IndividualSpins_for_yourfilename", except that the size of FID will be 1 by nspecC (i.e., all spins for a metabolite are added together). This is what is read in by INSPECTOR to make a basis set. The last folder output is "RawBasis_for_yourfilename". Within each one of these is a .raw file which can be used by LCModel to generate a basis set. In addition to this MARSS outputs a single file called "generatedSignals.mat". Inside this .mat there is a single call array called "signal", which is a cell that contains all the matrices for all the specified metabolites. For example, if you have 4 metabolites signal will be a cell array of length 4, where each index contains the FID for each specified metabolite. Note that all data output is in the time domain, thus to display spectra the Fourier transform must be used.

A quick tip to understanding MARSS is that crusher gradients (i.e., those gradients played in-between RF pulses) and slice-selective gradients are handled differently. Crushers, for the basic user, can simply be ignored and the problem of unwanted coherence pathways is handled through a filter. Slice-selective gradients, however, cannot. Also the rephasing lobes associated with slice-selective gradients must also be input.

Mandatory inputs in the input .mat file:

1) The physical RF pulse, input as a cell of N RF pulses called “rfPulses”. The amplitude unit of the RF pulses must all be the same, and can either be ‘Gauss’, ‘Hz’ or radians (‘rad’). To specify which amplitude type the variable “amplitudeUnit” must be the corresponding string. For example, if your RF pulses have a physical amplitude of Hz then write

```
>>amplitudeUnit = ‘Hz’.
```

For example, for a 3-pulse experiment where the three RF pulses are variables of name individualRFPulse1, individualRFPulse2 and individualRFPulse3, respectively, one would write:

```
>>rfPulses{1} = individualRFPulse1;

>>rfPulses{2} = individualRFPulse2;

>>rfPulses{3} = individualRFPulse3;
```

And individualRFPulse1/2/3 can either be a matrix of size (number of points by 1) or (number of points by 2), where the second column corresponds to the phase of the RF pulse if the pulses have phase (must be input in units of radians).

2) Durations of the RF pulses, input as a vector of size N in seconds. For example, if you had a 3-pulse experiment where the first pulse is 1 ms in duration and the second duration is 2 ms and the third duration is 3 ms you would write

```
>>durations(1) = 1E-3;

>>durations(2) = 2E-3;

>>durations(3) = 3E-3;
```

3) Delays between RF pulses, input as a vector of size N in seconds. The first value is the delay (i.e., dead time) between the end of the 1st RF pulse and the beginning of the 2nd RF pulse, the second value is between the 2nd and 3rd rf pulse, and the Nth entry is between the Nth RF pulse and the start of acquisition. **Note this is not the time from**

the peak of one RF pulse to the next, it is the time from the end of one pulse to the beginning of the next. For example, if two pulses were played back-to-back the delay corresponding to value would be zero, regardless of the duration of the pulses. If you had a three-pulse experiment where the delays between the 1st and 2nd pulse is 1 ms, the delay between the 2nd and 3rd RF pulse is 2 ms and the delay between the 3rd RF pulse and the start of acquisition is 3 ms you would write

```
>>delays(1) = 1E-3;

>>delays(2) = 2E-3;

>>delays(3) = 3E-3;
```

4) The slice-select gradients which is a matrix of size N x 3, input in units of mT/m. For example, if you had a 3-pulse experiment and the slice-select gradient amplitudes were 1 mT/m, 2 mT/m, and 3 mT/m, and played out in the x, y and z axes, respectively then you would write:

```
>>G = zeros(3,3);

>>G(1,1) = 1;

>>G(2,2) = 2;

>>G(3,3) = 3;
```

Note that the notation used for all axes is column 1 corresponds to x, column 2 corresponds to y and column 3 corresponds to z.

5) The rephasing areas / crusher gradients, input in units of mTs/m which is a matrix of size N x 3. If you do not input the crusher gradients because you are using the coherence pathway filter (recommended, see optional inputs) then you still must put in the rephasing lobes (typically one just one following the slice-select RF pulses for spin-echo based sequences like PRESS, sLASER, LASER, etc, and 3 for STEAM). If you have a self-refocusing pulse then and do not want to simulate crushers then this matrix would need to set to be all zeros. For the most common example where you have a single rephasing lobe directly after the slice-selective RF pulse (i.e., PRESS, sLASER, LASER and SPECIAL) and the rephasing crusher is of area 0.002 mTs/m (determined by the specifics of the excitation pulse and slice-selective gradient) in the x direction directly after the first pulse in a 3-pulse experiment you would input this as:

```
>>rephaseAreas = zeros(N,3);
>>rephaseAreas(1,1) = 0.002;
```

If you also use rephaseAreas for crusher gradients remember that the phase accrued during these periods is linearly proportional to the time-integral of the gradient waveform. Thus if you have two crushers that play out in a single period, simply add their areas (mTs/m) together, then input this value into rephaseAreas. Similarly, if you have a rephasing lobe and a crusher (which exists, for example, directly after the 1st RF pulse for PRESS) simply add the rephasing lobe area to the crusher lobe area. To calculate the rephasing area, R_a for an excitation pulse the following equation can be used:

$$R_a = \tau_{iso} G,$$

where τ_{iso} is the "iso-delay" time, in seconds, which is the time from when the magnetization has effectively tipped into the transverse plane until the end of the pulse (equal to ½ the duration of the RF pulse for symmetric pulses, whereas for min-phase pulses it is a value smaller than half the duration of the RF pulse) and G is the slice-select gradient amplitude, in mT/m.

6) Field strength (in Tesla), for example if you wanted to simulate for 3 Tesla you would write

```
>>B0 = 3;
```

7) Metabolites input as a cell. Most chemical shift and J-coupling constants are obtained from Govindaraju *et al*.[2], with GABA taken from Near *et al*.[3] and 2HG from Choi *et al*.[4] The following metabolites can be specified:

Name of metabolite	MARSS metabolite string
Acetate	Ace
Ascorbate	Asc
Aspartate	Asp
Alanine	Ala
N-Acetylaspartic Acid	NAA
N-Acetylaspartylglutamic Acid	NAAG
γ-aminobutyric acid	GABA
Choline	Ch
Creatine	Cr
Ethanolamine	Eth
Glucose [0.36*GlcA + 0.64*GlcB (biological ratio of the two glucose anomers)]	Glc
α-glucose	GlcA
β-glucose	GlcB
Glutamate	Glu
Glutamine	Gln
Glutathione	GSH
Glycerol	Glycerol
Glycero-phosphocholine	GPC
Glycine	Gly

Homocarnosine	Hom
Myo-inositol	mI
Scyllo-inositol	sI
Lactate	Lac
Phenylanine	PA
Phosphocreatine	PCr
Phosphorylcholine	PCh
Phosphorylethanolamine	PE
Pyruvate	Pyr
Serine	Ser
Succinate	Suc
Taurine	Tau
Threonine	Thr
Tryptophan	Try
Tyrosine	Tyr
Valine	Val
Water	Water
2-Hydroxyglutarate	2HG

Note that water always appears at the reference peak (i.e., a default of 4.65, but can be specified to other values, since the center frequency is tuned to water). You can see how the metabolite information (chemical shift, J-couplings, number of protons) are stored in the included matrix file “metabList.mat”. If you would like to add your own metabolite you can add it to this file – for more information contact me at kl2968@columbia.edu.

The variable “metabolite” is a cell with an entry containing the MARSS Metabolite String for each metabolite that MARSS will simulate. For example, if you wanted GABA, NAA and Cr in your outputted folder write:

```
>>metabolites{1} = ‘GABA’;

>>metabolites{2} = ‘NAA’;

>>metabolites{3} = ‘Cr’;
```

8) Spatial locations to be simulated over, which is input as a vector. Must have values for x, y and z in mm. For example, if you wanted to simulate over 40 mm (centered about 0) with 128 points in each dimension then you would write:

```
>>x = linspace(-20, 20,128);

>>y = linspace(-20, 20,128);

>>z = linspace(-20, 20,128);
```

If you are not interested in one dimension you could just set one of them equal to a constant value, but make sure that the value you specify is within your voxel. If you choose a value outside of your voxel you will still get spectra but it will have a very weird shape. If the amplitude of the spectra is very small (less than 1 at the peak) this is likely indicative of a problem. All 3 of x, y and z must be set.

9) The coherence pathway filter is a way to filter the density matrix after each RF pulse as explained in the MARSS manuscript. For example, if you want to use it for PRESS then you would write:

```
>> coherencePathway = [-1 1 -1];
```

Note that this input isn’t strictly necessary, but you do need to handle unwanted coherence pathways in some way for most sequences. The input coherencePathway is a convenient and fast way to remove unwanted coherence pathways. Alternatively phase cycling (see 2 below) or crusher gradients could be used. Crusher gradients would then have to be input in rephaseAreas. For information about which coherence pathway is selected for many common MRS sequences see Landheer *et al*.[5]

10) The receiver bandwidth in Hertz that matches your experimental acquisition. For example if you wanted 5000 Hertz receiver bandwidth you would input

```
>> bw = 5000;
```

11) The number of points in the FID. For example if you wanted 2048 points you would input:

```
>> Npoints = 2048;
```

12) The output directory where the basis sets will be simulated. For example if you wanted it in ‘C:\Users\Karl’ you would input:

```
>>outputDirectory = ‘C:\Users\Karl’;
```

Optional inputs not necessary for all experiments

1) referencePeak is the frequency (in ppm) water is placed at. This is because all metabolites are referenced to TMS in simulation (0 ppm), however the center frequency for scanners is 0 ppm = water, not TMS. So we shift all spectra by this value. Default is 4.65 ppm (frequency of water at 37 °C). If you wanted it to be another value (say 4.75 ppm for water at 20 °C), then you would write

```
>>referencePeak = 4.75;
```

2) Phase cycling is a way to filter the density matrix to exclude unwanted coherence pathways. If you want to use phase cycling them simply have the variable phaseShifts in the .mat file. phaseShifts should be a matrix of size $(N+1,N_c)$, where the first column corresponds to the phase shifts applied to the 1st RF pulse, the second column corresponding to the phase shifts applied to the 2nd RF pulse, etc, and the last column corresponds to the phase shift applied to the receiver. N_c is the number of phase step cycles and all phase shifts should be input in radians. For example if you wanted to use

the following 8 step phase cycling scheme for PRESS you would write:

```
>>phaseShifts = [0 0 0 0; 0 pi/2 0 pi; 0 pi 0 0; 0 3*pi/2 0 pi; pi 0 0 pi; pi pi/2 0 0; …
                   pi pi 0 pi; pi 3*pi/2 0 0];
```

3) rfOffsets are the offset frequencies of your N pulses, the default is all zeros. This is necessary for J-difference editing experiments, whereas for slice-selects it will just move the position of the voxel. Note that the units are in Hz from tetramethylsilane (0 ppm), **not from water**, so the value here is likely different than the value you put in the scanner. For example, if you wanted to edit GABA which places the editing pulse at 1.89 ppm this would correspond to an offset of 1.89*B0*gyromagneticRatio, which for 3T, is 241 Hz. If your second pulse was the editing pulse in a three-pulse experiment, and the other two had offsets of 0 Hz, you would write

```
>>rfOffsets(1) = 0;

>>rfOffsets(2) = 241;

>>rfOffsets(3) =0;
```

4) autoMean (default is ‘true’). This variable does an automatic averaging over all spatial positions of the density matrix. If you write

```
>>autoMean = false;
```

Then the resulting signals in generatedSignals.mat will be of size (length(x), length(y), length(z), numberOfSpins, numberOfPoints), which allows you to visualize the traces at different spatial locations. Do not use this for generating basis sets as both LCModel and INSPECTOR will be unable to read this. This is only for testing and visualization. Note that this greatly reduces computational efficiency so large numbers of points (20^3+) will take very long for large spin systems. Note that you may run out of memory when doing this for large spin systems and number of spatial points (just due to the maximum matrix size MATLAB can hold), in which case you could fix this by arraying over one (or more) spatial positions (i.e., hold one position constant and repeat the simulation numerous times incrementing that position), or do a single spatial dimension at a time. I use this to ensure that my rephasing lobes are of proper area by only arraying over a single spatial dimension at a time.

5) If you have gradient waveforms (i.e., time varying gradients for GOIA pulses, for example) which are played out during the RF pulses then you can specify them using the variable G, but in this case it must be a cell, not a matrix. The cell must be of size N (number of RF pulses), with each element containing a matrix of N_G x 3, where N_G is the number of points in the gradient waveform, and the first column corresponds to the x values, the second column corresponds to the y values, and the third column corresponds to the z values. The values should be input in mT/m, and the durations of these gradient waveforms is equal to the duration of the RF pulses. If you have both

time-varying and non time-varying gradient waveforms then an entry of G can be a vector. For example, if you have say time-varying gradient accompanying the first RF pulse for a three-pulse experiment (which, in this case, is a sinusoid along the x axis), but the second and third gradients are just 5 mT/m in the y axis and 10 mT/m in the z axis, respectively, then you would input

```
>>G{1} = [10*sin(1:100)' zeros(100,1) zeros(100,1)];

>>G{2} = [0 5 0];

>>G{3} = [0 0 10];
```

The same rules regarding 1D projection method and the propagators applies with gradient waveforms, so once again for some sequences this may take a long time. Note that it is assumed these time-varying gradients are the same duration as the accompanying RF pulses.

6) If you want to include relaxation directly into your simulated basis set (i.e., relaxation during delays and during RF pulses) then you must set the variable relaxationTimes, which is a matrix of size numberOfMetabolites x 2, where the two columns correspond to the T1 and T2 values of your metabolites, respectively. The order of the relaxation times must match the order of the metabolite cell. For example say you have NAA (T1 = 1s, T2 = 100 ms) and Cr (T1 = 2s, T2 = 200 ms), then you would input the following:

```
>>metabolites{1} = ‘NAA’;

>>metabolites{2} = ‘Cr’;

>>relaxationTimes = [1 0.1; 2 0.2];
```

For most experiments simply scaling the basis set by exp(-TE/T2) is sufficient (as shown in the manuscript), but for some sequences it may important so I have included it (spin-locking experiments, CP experiments). Note that the 1D projection method and propagators cannot be used with relaxation, so MARSS will take a very long time for large number of spatial points (few hours to few days depending on which metabolite). Note that when relaxation is specified there is no relaxation applied during the FID – only during the period prior to acquisition. Note that this will only scale it by relaxation during the pulse sequence, so for steady-state correction effects (i.e., incomplete relaxation between TRs) this must still be scaled accordingly for absolute quantification.

Tips & Notes

There are 6 MARSS templates provided in the folder: PRESSGETE30ms.mat, PRESSPhilipsTE30ms.mat, PRESSSiemensTE30ms.mat, STEAMGETE20msTM10ms.mat, STEAMPhilipsTE20msTM16ms.mat, STEAMSiemensTE20msTM10ms.mat, corresponding to PRESS and STEAM for three vendors (GE, Siemens and Philips). The TE and TM are given in the name of the file, if you want a different TE or TM you can go into the files and change the variable “delays” to meet your experimental details. If you need help doing this, please contact me. These are also useful as templates because they have all variables that must be input. If you want to run MARSS with one of these templates just simply copy the file and rename it to "MARSSinput.mat" and run MARSS.exe.

For J-difference editing just experiments simulate two basis sets (one with the editing pulse on, one with the editing pulse off) and then write a MATLAB script that cycles through the outputted metabolites and subtracts them from one another. If you need help with this, please contact me.

All outputted spectra are linebroadened by a 1 Hz exponential filter. This means when you fit your spectra to the basis set and you need to linebroaden it by, say, 5 Hz, then the actual linewidth of your spectra is 5 + 1 (6) Hz.

Notes for LCModel users

Ensure that NOSHIF is true, i.e., add the following to all $NMEACH sections."

NOSHIF=T.

For any questions, bugs, critiques or suggestions please email me at kl2968@columbia.edu.

References

1. Landheer, K., Swanberg, K. M. & Juchem, C. *Magnetic Resonance Spectrum Simulator (MARSS), A Novel Software Package for Fast and Computationally Efficient Basis Set Simulation. NMR Biomed, in press.*

2. Govindaraju, V., Young, K. & Maudsley, A. A. Proton NMR chemical shifts and coupling constants for brain metabolites. *NMR Biomed* **13,** 129–153 (2000).

3. Near, J., Evans, C. J., Puts, N. A. J., Barker, P. B. & Edden, R. A. E. J-difference editing of GABA: simulated and experimental multiplet patterns. *Magn Reson Med* **70,** 1–17 (2013).

4. Choi, C. *et al.* 2-hydroxyglutarate detection by magnetic resonance spectroscopy in IDH-mutated patients with gliomas. *Nat Med* **18,** 624–629 (2012).

5. Landheer, K. & Juchem, C. Dephasing optimization through coherence order pathway selection (DOTCOPS) for improved crusher schemes in MR spectroscopy. *Magn Reson Med* **81,** 2209–2222 (2019).