# FID-A: The FID Appliance

Version 1.0

User Manual

Jamie Near

10 December 2014

# TABLE OF CONTENTS

# 1. Introduction

The FID Appliance (FID-A) is an open-source software package for simulation of MRS experiments, design and analysis of radiofrequency (RF) pulses, and processing of MRS data. The software is freely available for download (www.github.com/CIC-methods/FID-A).

The FID-A software package consists of three separate toolboxes; the Simulation Toolbox, the RF-pulse toolbox, and the Processing toolbox. Each is described briefly below:

## 1.1. Simulation Toolbox

The FID-A NMR simulation toolbox is based on an implementation of the density matrix formalism, where the evolution of the spin system in a given NMR experiment is described by successive evolutions of the density matrix by time-independent Hamiltonian operators.

The simulation toolbox contains built-in functions for simulating the basic components of an MRS pulse sequence, including excitation (sim_excite.m), delays (sim_evolve.m), rotation using ideal (sim_rotate.m) and shaped (sim_shapedRF.m) radiofrequency pulses, and signal readout (sim_readout.m). Using these basic pulse sequence elements, common in-vivo MRS pulse sequences are also implemented, including the FID (sim_onepulse.m), PRESS (sim_press.m), STEAM (sim_steam.m) and spin-echo(sim_spinecho.m).

Simulation is performed on a given spin-system by specifying the pulse timings and the chemical shifts and coupling constants of the spin-system of interest. A full set of common metabolite spin-system definitions is provided based on values previously

published by Govindaraju et al (NMR Biomed 2000, 13:129-153). Spin system definitions can be found in …/FID-A/simulationTools/metabolites/. Each metabolite's spin system definition is contained within a structure with two fields: "shifts" and "J". Shifts is a vector of length N, which contains the chemical shift value (in ppm) for each of the N spins in the spin system. J is an N x N matrix specifying the coupling constant (in Hz) between each spin and every other spin in the system. For spin systems with more than 7 spins, the spin system is broken down in to multiple independent (uncoupled) components with fewer spins. This is because the computation time increases more than linearly with increasing number of spins, and so it is much more computationally efficient to simulate, say two groups of spins with 7 protons each rather than simulating one group of 14 protons. Finally, some spin systems have been reduced in size due to redundancy. For example, scyllo inositol consists of nine magnetically equivalent protons, and is therefore most easily simulated as a single proton. However, the user must then remember to scale the intensity of Scyllo Inositol Simulations by a factor of 9 using op_ampscale(sim_scyllo,9);.

Excitation and refocusing RF pulses can be modeled as ideal (instantaneous) rotations, or fully shaped RF waveforms, depending on the user requirements. In the case of shaped RF pulses, phase cycling is performed to remove unwanted coherences. Function names in the simulation toolbox begin with the prefix "sim_".

### 1.2. RF Pulse Toolbox

The FID-A RF toolbox enables the creation of basic RF pulse waveforms, and Bloch simulation to determine the resulting excitation/refocusing/inversion profiles, as well as frequency

shifting and resampling of rf waveforms. RF pulses are stored in MATLAB structure format, with fields corresponding to the RF waveform, the type of rf pulse (excitation, inversion, refocusing), the time-bandwidth product of the rf pulse, and the time-B1 product. Function names in the RF-pulse toolbox begin with the prefix ”rf_”.

### 1.3. Processing Toolbox

The FID-A processing toolbox contains “load” functions to accept MRS data in MRI vendor data formats (Siemens, Agilent, others in development), and store them in MATLAB.

The data can then be operated on using any of the over 50 different processing operations, including (but not limited to) filtering (op_filter.m), zeropadding (op_zeropad.m), time domain truncation (op_leftshift.m and op_zerotrim.m), frequency domain truncation (op_freqrange.m), eddy current correction (op_ecc.m), removal motion corrupted averages (op_rmbadaverages.m), retrospective frequency and phase drift correction (op_alignAverages.m and op_alignAverages_fd.m, Near et al, Magn Reson Med 2014, DOI: 10.1002/mrm.25094), combination of multi-element RF coil data (op_addrcvrs.m), and zero- and first-order phase corrections (op_addphase.m).

These functions can be nested within one another. For example, to load a spectrum, combine the receivers, combine the averages and then filter the result, can be done using the following single line of code:

```
out=
op_filter(op_averaging(op_addrcvrs(op_loadspec_twix('filename.dat'),1,'w'),5);
```

where the argument “5” represents a 5 Hz exponential filter, the argument “‘w’” specifies that a weighted coil recombination should be performed, and the argument “1” specifies that the phases and amplitudes of the rf coil channels should be determined using the first point in the time domain.

Finally, the FID-A processing toolbox contains “write” functions to output the data into the formats accepted by leading MRS data processing packages. Function names in the processing toolbox begin with the prefix “op_”, which stands for operator.

While most of the tools in the processing toolbox are single processing operations, there are a few examples of full MRS data processing pipelines. These include: op_specialproc.m, op_pressproc.m, and op_megapressproc.m. These “pipeline” operations begin with raw MRS data in Siemens .dat format and then process the data in a logical step-by-step fashion to generate fully processed data that is ready to be analyzed using one of the leading MRS analysis software packages (LCModel, jMRUI or Tarquin). Processing steps performed in these pipelines include combination of receive channels (op_addrcvrs.m), removal of motion corrupted averages (op_rmbadaverages.m), spectral registration of averages (op_alignAverages.m), and combination of averages (op_averaging.m). For more information on these pipeline operations, see section 4.

### 1.4. Data structure formatting

The FID-A software package is implemented in MATLAB (Natick MA, USA). Within FID-A, simulated or experimental MRS datasets, and RF pulses are stored in uniquely formatted data structures that encapsulate all of the data and relevant header information. Each dataset structure stores both time-domain and frequency-domain data arrays. When multiple averages, coil channels, or

subspectra of data are present, they are stored in separate dimensions of the arrays, and indexed within the header. Each RF pulse structure stores the rf waveform as well as information about the rf pulse type (excitation, refocusing or inversion), the time-bandwidth product, and the time-B1 product. By encapsulating data and header information in this way, processing operations have access to all relevant information and thus require as few input arguments as possible. The fields of the data structure are listed and briefly described below:

```
fids        - time domain MRS data.
specs       - frequency domain MRS data.
t           - vector of time values for plotting in the time domain [s]
ppm         - vector of frequency values for plotting in the frequency domain
              [ppm]
sz          - size of the fids and specs arrays
date        - date that the data was acquired or simulated
averages    - number of averages in the dataset (possibly altered by
              processing)
rawAverages - number of averages in the original dataset (not altered by
              processing).
subspecs    - number of subspectra (ISIS, edit on/off, etc) in the dataset
              (possibly altered by processing).
rawSubspecs - number of subspectra (ISIS, edit on/off, etc) in the original
              dataset (not altered by processing).
Bo          - magnetic field strength [Tesla]
txfrq       - Centre frequnecy [MHz];
linewidth   - linewidth of data (only used for simulated data) [Hz]
n           - number of spectral points
dwelltime   - dwell time of the data in the time domain [s] (dwelltime =
              1/spectralwidth)
sim         - type of simulation (ideal vs. shaped pulses), only used for
              simulated data.
te          - echo time of acquisition [ms], only used for simulated data
seq         - type of sequence used (only used for simulated data).
dims        - structure specifying which data dimensions are stored along
              which dimensions of the fids/specs arrays.  Fields include:
          t         - time/freqeunecy dimension (usually this is 1, the first
                      dimension of the fids/specs array).
```

coils - for multiple receiver array, this is the dimension of the arrayed receiver data (can be 2, 3 or 4).

averages - for multiple averages, this is the dimension of the averages (can be 2, 3 or 4).

subSpecs - in the case of subtraction data (ISIS, MEGA-PRESS), this is the dimension of the subSpectra (can be 2, 3 or 4).

flags - structure specifying what processing operations have already been done on the data. fields include:

writtentostruct — Has the dataset been written to a structure (1 or 0)

gotparams - Have the parameters been retrieved from the dataset (1 or 0)

filtered - Has the dataset been filtered (1 or 0)

zeropadded - Has the dataset been zeropadded (1 or 0)

freqcorrected - Has the dataset been frequency corrected (1 or 0)

phasecorrected - Has the dataset been phase corrected (1 or 0)

averaged - Have the averages been combined (1 or 0)

addedrcvrs - Have the receiver channels been combined (1 or 0).

Subtracted - Have the subspecs been subtracted (1 or 0)

Writtentotext - Has the data been writte to text file (1 or 0)

Downsampled - has the data been resampled to a different spectral resolution (1 or 0)

avgNormalized - Has the data been amplitude scaled following combination of the averages (1 or 0)

isISIS - Does the dataset contain ISIS subspectra (1 or 0)

The fields of the RF pulse structure are listed and briefly described below:

waveform — an n x 3 array, where n is the number of points in the rf pulse, the first column (:,1) contains the rf pulse phase, the second column (:,2) contains the absolute rf amplitude, and the third column (:,3) contains the duration of each time step (normally this is a vector of ones, but not necessarily).

type - The type of RF pulse. Options are 'exc' for an excitation pulse, 'inv' for an inversion pulse, and 'ref' for a refocusing pulse.

tw1 - The product of the duration of the rf pulse [s] and the w1max of

```
          the rf pulse [Hz].
tbw       - The product of the duration of the rf pulse [s] and the
          bandwidth of the rf pulse [Hz]
```

Below is a brief summary of all of the functions in the FID-A Software package. The same information can be obtained by typing 'help functionName" at the MATLAB command line:

# 2. Simulation Tools

## 2.1. sim_Hamiltonian.m

```
USAGE:
[H,d] = sim_Hamiltonian(sys,Bfield);

DESCRIPTION:
Creates the nxn Hamiltonian matrix for a spin system, which can then be used
in other functions to simulate NMR experiments.

INPUTS:
sys            = spin system definition structure.
Bfield         = magnetic field strength (Tesla).
```

## 2.2. sim_evolve.m

```
USAGE:
d_out = sim_evolve(d_in,H,t)

DESCRIPTION:
This function simulates free evolution of the spin system under the effects of
chemical shift and scalar coupling.

INPUTS:
d_in      = input density matrix structure.
H         = Hamiltonian operator structure.
t         = duration of evolution (s)
```

## 2.3. sim_excite.m

```
USAGE:
d_out = sim_excite(H,axis,angle)

DESCRIPTION:
This function simulates the effect of an ideal (instantaneous) excitation
pulse on the density matrix.  Used in simulation tools.

INPUTS:
d_in      = input density matrix structure.
H         = Hamiltonian operator structure.
axis      = Axis of rotation ('x' or 'y');
angle     = Flip angle of excitation (degrees).  Optional.  Default=90.  If
            angle is a scalar, then the same flip angle is applied to all
            spins in the spin system.  If angle is a vector, then the elements
            of the vector specify the flip angles to apply to each spin in the
            system.  In this case, the length of the vector must be the same
            as the number of spins in the spin system.
```

## 2.4. sim_excite_arbPh.m

```
USAGE:
d_out = sim_excite_arbPh(H,phase,angle)

DESCRIPTION:
This function simulates the effect of an ideal (instantaneous) excitation
pulse on the density matrix.  Used in simulation tools.  The phase of the
excitation pulse can be arbitrarily chosen.  To achieve an arbitrary phase,
this code executes a rotation about z by an angle of -phi (the rf pulse
phase), then executes a rotation about x by an angle of "angle"(the flip angle
of the excitation pulse), and then executes a rotationback about z by an angle
of phi.

INPUTS:
d_in      = input density matrix structure.
H         = Hamiltonian operator structure.
phase     = Phase of rotation in degrees (ie. 0='x', 90='y', etc);
angle     = Flip angle of excitation (degrees).  Optional.  Default=90.  If
            angle is a scalar, then the same flip angle is applied to all
            spins in the spin system.  If angle is a vector, then the elements
            of the vector specify the flip angles to apply to each spin in the
            system.  In this case, the length of the vector must be the same
            as the number of spins in the spin system.
```

## 2.5. sim_lcmrawbasis.m

```
USAGE:
[RF,out]=sim_lcmrawbasis(n,sw,Bfield,linewidth,metab,tau1,tau2,addref,makeraw,
seq)

DESCRIPTION:
Generate an LCModel .RAW file to be used as an individual metabolite basis
spectrum in an LCModel basis set.  The relevant characteristics of the
acquisition can be specified (pulse sequence, number of points, spectral
```

```
width, etc)

INPUTS:
n         = number of points in fid/spectrum
sw        = desired spectral width in [Hz]
Bfield    = main magnetic field strength in [T]
linewidth = linewidth in [Hz]
tau1      = first echo time in [s] (if seq='st', tau1 = TE)
tau2      = second echo time in [s].  (Used in Press, but not used in SE.  (If
            seq='st', tau2=TM).
addref    = add reference at 0ppm (for use in LCModel makebasis) ['y' or 'n']
makeraw   = make output file for lcmodel ['y' or 'n']
seq       = pulse sequence ['se' for Spin Echo or 'p' for Press]
metab     = one of the following choices
   'H2O'    = Water
   'Ala'    = Alanine
   'Asp'    = Aspartate
   'PCh'    = PhosphoCholine
   'Cr'     = Creatine
   'PCr'    = PhosphoCreatine
   'GABA'   = Gamma-aminobutyric acid (kaiser)
   'GABA3'  = Gamma-aminobutyric acid (de Graaf)
   'Gln'    = Glutamine
   'Glu'    = Glutamate
   'GSH'    = Glutathione
   'Gly'    = Glycine
   'Ins'    = Myo-inositol
   'Lac'    = Lactate
   'NAA'    = N-acetyl aspartate
   'Scyllo' = Scyllo-inositol
   'Tau'    = Taurine
   'Asc'    = Ascorbate (Vitamin C)
   'bHB'    = beta-Hydroxybutyrate
   'bHG'    = beta-Hydroxyglutarate
   'Glc'    = Glucose
   'NAAG'   = N-acetyl aspartyl glutamate
   'GPC'    = Glycero-phosphocholine
   'PE'     = Phosphoryl ethanolamine
   'Ser'    = Serine
```

## 2.6. sim_megapress.m

```
USAGE:
out=sim_megapress(n,sw,Bfield,linewidth,sys,taus,refoc1Flip,refoc2Flip,editFli
p)

DESCRIPTION:
Simulate the MEGA-PRESS sequence with instantaneous localization and editing
pulses.  Provides the ability to specify the flip angle of each refocusing
pulse and editing pulse on each spin in the spin system.

INPUTS:
n         = number of points in fid/spectrum
sw        = desired spectral width in [Hz]
Bfield    = main magnetic field strength in [T]
linewidth = linewidth in [Hz]
sys       = spin system definition structure
taus      = pulse sequence timing vector:
```

```
   taus(1)     = time in [ms] from 90 to 1st 180
   taus(2)     = time in [ms] from 1st 180 to 1st edit pulse
   taus(3)     = time in [ms] from 1st edit pulse to 2nd 180
   taus(4)     = time in [ms] from 2nd 180 to 2nd edit pulse
   taus(5)     = time in [ms] from 2nd edit pulse to ADC
refoc1Flip= array of refoc1 flip angles for each spin in system
refoc2Flip= array of refoc2 flip angles for each spin in system
editFlip  = array of editing flip angles for each spin in system
```

## 2.7. sim_megapress_shaped.m

```
USAGE:
sim_megapress_shaped(n,sw,Bfield,linewidth,taus,sys,editPulse,editTp,editPh1,e
ditPh2,refPulse,refTp,Gx,Gy,dx,dy,refPh1,refPh2)

DESCRIPTION:
This function simulates the MEGA-PRESS sequence with shaped localization
pulses and shaped editing pulses.  Enables choice of the timings of all of the
rf pulses as well as the choice of the phase of both the editing pulse and the
refocusing pulses.  This allows phase cycling of the editing and refocusing
pulses by repeating simulations with different editing pulse phases, which is
necessary to remove phase artefacts from the editing pulses.  For the editing
pulses, an eight step phase cycling scheme is typically sufficient, where by
the first editing pulse is cycled by 0 and 90 degrees, and the second editing
pulse is cycled by 0,90,180, and 270 degrees, and all phase cycles should be
added together to remove unwanted coherences.  For the refocusing pulses, a
four step phase cycling scheme is typically sufficient, where both refocusing
pulses are phase cycled by 0 and 90 degrees, and the phase are combined in the
following way:

signal = ([0 90] - [0 0]) + ([90 0] - [90 90]);

where, in [X Y], X is the phase of the first refocusing pulse and Y is the
phase of the second refocusing pulse

Note that this code only simulates one subspectrum at a time (edit-on or edit-
off).  The difference spectrum can be obtained by simulating one of each, and
then subtracting.

INPUTS:
n         = number of points in fid/spectrum
sw        = desired spectral width in [Hz]
Bfield    = main magnetic field strength in [T]
linewidth = linewidth in [Hz]
taus(1)     = time in [ms] from 90 to 1st 180
taus(2)     = time in [ms] from 1st 180 to 1st edit pulse
taus(3)     = time in [ms] from 1st edit pulse to 2nd 180
taus(4)     = time in [ms] from 2nd 180 to 2nd edit pulse
taus(5)     = time in [ms] from 2nd edit pulse to ADC
              FOR MEGA-PRESS on SIEMENS SYSTEM:
            taus=[4.545,12.7025,21.7975,12.7025,17.2526];
editPulse  = Editing pulse shape - [Nx3] array with [:,1]=phase,
             [:,2]=amplitude, [:,3]=duration
editTp     = duration of editing pulse in [ms];
sys        = Metabolite spin system definition structure;
editPh1    = the phase of the first editing pulse in [degrees];
editPh2    = the phase of the second editing pulse in [degrees];
refPulse   = Refocusing pulse shape - [Nx3] array with [:,1]=phase,
             [:,2]=amplitude, [:,3]=duration
```

```
refTp      = duration of refocusing pulse in [ms]
Gx         = gradient strength for first selective refocusing pulse [G/cm]
Gy         = gradient strength for second selective refocusing pulse [G/cm]
dx         = position offset in x-direction (corresponding to first refocusing
             pulse) [cm]
dy         = position offset in y-direction (corresponding to second
             refocusing pulse) [cm]
refPh1     = the phase of the first refocusing pulse in [degrees];
refPh2     = the phase of the second refocusing pulse in [degrees];
```

## 2.8. sim_megapress_shapedEdit.m

USAGE:

```
sim_megapress_shapedEdit(n,sw,Bfield,linewidth,taus,sys,editPulse,editTp,editP
h1,editPh2)
```

DESCRIPTION:
This function simulates the MEGA-PRESS sequence with instantaneous localization pulses and shaped editing pulses. Enables choice of the timings of all of the rf pulses as well as the choice of the phase of the editing pulse. This allows phase cycling of the editing pulses by repeating simulations with different editing pulse phases, which is necessary to remove phase artefacts from the editing pulses. For the editing pulses, an eight step phase cycling scheme is typically sufficient, where by the first editing pulse is cycled by 0 and 90 degrees, and the second editing pulse is cycled by 0,90,180, and 270 degrees, and all phase cycles should be added together to remove unwanted coherences.

Note that this code only simulates one subspectrum at a time (edit-on or edit-off). The difference spectrum can be obtained by simulating one of each, and then subtracting.

INPUTS:

```
n         = number of points in fid/spectrum
sw        = desired spectral width in [Hz]
Bfield    = main magnetic field strength in [T]
linewidth = linewidth in [Hz]
taus(1)     = time in [ms] from 90 to 1st 180
taus(2)     = time in [ms] from 1st 180 to 1st edit pulse
taus(3)     = time in [ms] from 1st edit pulse to 2nd 180
taus(4)     = time in [ms] from 2nd 180 to 2nd edit pulse
taus(5)     = time in [ms] from 2nd edit pulse to ADC
                 FOR MEGA-PRESS on SIEMENS SYSTEM:
              taus=[4.545,12.7025,21.7975,12.7025,17.2526];
editPulse  = Editing pulse shape - [Nx3] array with [:,1]=phase,
             [:,2]=amplitude, [:,3]=duration
editTp     = duration of editing pulse in [ms];
sys        = Metabolite spin system definition structure;
editPh1    = the phase of the first editing pulse in [degrees];
editPh2    = the phase of the second editing pulse in [degrees];
```

## 2.9. sim_megapress_shapedRefoc.m

USAGE:

```
sim_megapress_shapedRefoc(n,sw,Bfield,linewidth,taus,sys,editFlip,refPulse,ref
```

```
Tp,Gx,Gy,dx,dy,refPh1,refPh2)

DESCRIPTION:
This function simulates the MEGA-PRESS sequence with shaped localization
pulses and instantaneous editing pulses.  Enables choice of the timings of all
of the rf pulses as well as the choice of the phase of both the editing pulse
and the refocusing pulses.  This allows phase cycling of the editing and
refocusing pulses by repeating simulations with different editing pulse
phases, which is necessary to remove phase artefacts from the editing pulses.
For the editing pulses, an eight step phase cycling scheme is typically
sufficient, where by the first editing pulse is cycled by 0 and 90 degrees,
and the second editing pulse is cycled by 0,90,180, and 270 degrees, and all
phase cycles should be added together to remove unwanted coherences.  For the
refocusing pulses, a four step phase cycling scheme is typically sufficient,
where both refocusing pulses are phase cycled by 0 and 90 degrees, and the
phase are combined in the following way:

signal = ([0 90] - [0 0]) + ([90 0] - [90 90]);

where, in [X Y], X is the phase of the first refocusing pulse and Y is the
phase of the second refocusing pulse

Note that this code only simulates one subspectrum at a time (edit-on or edit-
off).  The difference spectrum can be obtained by simulating one of each, and
then subtracting.

INPUTS:
n         = number of points in fid/spectrum
sw        = desired spectral width in [Hz]
Bfield    = main magnetic field strength in [T]
linewidth = linewidth in [Hz]
taus(1)     = time in [ms] from 90 to 1st 180
taus(2)     = time in [ms] from 1st 180 to 1st edit pulse
taus(3)     = time in [ms] from 1st edit pulse to 2nd 180
taus(4)     = time in [ms] from 2nd 180 to 2nd edit pulse
taus(5)     = time in [ms] from 2nd edit pulse to ADC
                 FOR MEGA-PRESS on SIEMENS SYSTEM:
              taus=[4.545,12.7025,21.7975,12.7025,17.2526];
sys        = Metabolite spin system definition structure;
editFlip   = vector of editing flip angles in [degrees] at chemical shifts
             corresponding to 'shifts'.
refPulse   = Refocusing pulse shape - [Nx3] array with [:,1]=phase,
             [:,2]=amplitude, [:,3]=duration
refTp      = duration of refocusing pulse in [ms]
Gx         = gradient strength for first selective refocusing pulse [G/cm]
Gy         = gradient strength for second selective refocusing pulse [G/cm]
dx         = position offset in x-direction (corresponding to first refocusing
             pulse) [cm]
dy         = position offset in y-direction (corresponding to second
             refocusing pulse) [cm]
refPh1     = the phase of the first refocusing pulse in [degrees];
refPh2     = the phase of the second refocusing pulse in [degrees];
```

## 2.10. sim_onepulse.m

```
USAGE:
out=sim_onepulse(n,sw,Bfield,linewidth,sys)

DESCRIPTION:
```

```
This function simulates a pulse-acquire experiment with an ideal
(instantaneous) excitation pulse and an assumed lorentzian lineshape.  The
function calls the function 'sim_Hamiltonian' which produces the free
evolution Hamiltonian for the specified number of spins, J and shifts.

INPUTS:
n         = number of points in fid/spectrum
sw        = desired spectral width in [Hz]
Bfield    = main magnetic field strength in [T]
linewidth = linewidth in [Hz]
sys       = spin system definition structure
```

## 2.11. sim_onepulse_arbPh.m

```
USAGE:
out=sim_onepulse_arbPh(n,sw,Bfield,linewidth,sys,ph)

DESCRIPTION:
This function simulates a pulse-acquire experiment with an ideal
(instantaneous) excitation pulse and an assumed lorentzian lineshape.  The
function calls the function 'sim_Hamiltonian' which produces the free
evolution Hamiltonian for the specified number of spins, J and shifts.  This
function enables an excitation pulse with an arbitrary phase.

INPUTS:
n         = number of points in fid/spectrum
sw        = desired spectral width in [Hz]
Bfield    = main magnetic field strength in [T]
linewidth = linewidth in [Hz]
sys       = spin system definition structure
ph        = excitation pulse phase (in degrees)
```

## 2.12. sim_onepulse_shaped.m

```
USAGE:
out = sim_onepulse_shaped(n,sw,Bfield,linewidth,sys,RF,tp,dwdx,G)

DESCRIPTION:
This function simulates the effect of a frequency selective or slice selective
excitation, followed immediately by the acquisition window.  This is mainly an
excercise to see if I can get slice selective excitation working.

Note that when simulating a frequency selective pulse, it is okay to specify
only 8 arguments (no gradient needs to be specified).  If the 9th argument, G,
is specified and is non-zero, then a slice selective pulse is assumed.

INPUTS:
n         = number of points in fid/spectrum
sw        = desired spectral width in [Hz]
Bfield    = main magnetic field strength in [T]
linewidth = linewidth in [Hz]
sys       = spin system definition structure
RF        = radiofrequency pulse array [N x 3].  Phase, Amplitude, Duration.
tp        = RF pulse duration in [ms]
peakB1    = peak B1 amplitude in [kHz]
dwdx      = if simulating a frequency selective pulse, this argument should be
```

```
            the frequency offset [Hz].  If simulating a slice selective pulse,
            this argument should be the position offset [cm].
G         = gradient strength for slice-selective pulse [G/cm];
```

## 2.13. sim_press.m

```
USAGE:
out = sim_press(n,sw,Bfield,linewidth,sys,tau1,tau2)

DESCRIPTION:
This function simulates an ideal PRESS experiment.  The function calls the
function 'sim_Hamiltonian.m' which produces the free evolution Hamiltonian for
the specified number of spins, J and shifts.  This time the individual Iy and
Iz are needed as well.  This function simulates a spin-echo experiment with
echo time tau.

INPUTS:
n         = number of points in fid/spectrum
sw        = desired spectral width in [Hz]
Bfield    = main magnetic field strength in [T]
linewidth = linewidth in [Hz]
sys       = spin system definition structure
tau1      = Echo time in [s] of first press Spin Echo
tau2      = Echo time in [s] of second press Spin Echo
```

## 2.14. sim_press_shaped.m

```
USAGE:
out =
sim_press_shaped(n,sw,Bfield,linewidth,sys,tau1,tau2,RF,tp,dx,dy,Gx,Gy,phCyc1,
phCyc2)

DESCRIPTION:
This function simulates the PRESS experiment.  The excitation is simulated as
an instantaneous rotation, and the refocusing pulse is simulated as a shaped
rotation.

This code enables the choice of the phase of the refocusing pulses.  This
enables phase cycling of the refocusing pulses by repeating simulations with
different editing pulse phases, which is necessary to remove phase artefacts
from the editing pulses.  A four step phase cycling scheme is typically
sufficient, where both refocusing pulses are phase cycled by 0 and 90 degrees,
and the phase are combined in the following way:

signal = ([0 90] - [0 0]) + ([90 0] - [90 90]);

where, in [X Y], X is the phase of the first refocusing pulse and Y is the
phase of the second refocusing pulse.

Finally, this code simulates the spectrum at a given point in space (x,y),
given the values of the slice selection gradients (Gx, and Gy).  The pulse
waveform is assumed to be the same for both refocusing pulses.  In order to
fully simulate the MEGA-PRESS experiment, you have to run this simulation many
times at various points in space (x,y), and then add together the resulting
spectra.

INPUTS:
```

```
n         = number of points in fid/spectrum
sw        = desired spectral width in [Hz]
Bfield    = main magnetic field strength in [T]
linewidth = linewidth in [Hz]
sys       = spin system definition structure
tau1      = echo time 1 in [ms].
tau2      = echo time 2 in [ms].
RF        = radiofrequency pulse array [N x 3].  Phase, Amplitude, Duration.
tp        = RF pulse duration in [ms]
peakB1    = peak B1 amplitude in [kHz]
dx        = position offset in x-direction (corresponding to first refocusing
            pulse) [cm]
dy        = position offset in y-direction (corresponding to second refocusing
            pulse) [cm]
Gx        = gradient strength for first selective refocusing pulse [G/cm]
Gy        = gradient strength for second selective refocusing pulse [G/cm]
phCycl    = initial phase of the first refocusing pulse in [degrees];
phCycl2   = initial phase of the second refocusing pulse in [degrees];
```

## 2.15. sim_readout.m

```
USAGE:
d_out = sim_readout(d_in,H,n,sw,linewidth,rcvPhase,shape)

DESCRIPTION:
This function simulates an ADC readout of the transverse magnetization during
the free evolution of the spin system under the effects of chemical shift and
scalar coupling.

INPUTS:
d_in      = input density matrix structure.
H         = Hamiltonian operator structure.
n         = number of readout points
sw        = spectral width [Hz]
linewidth = full width at half maximum of spectral peaks [Hz]
rcvPhase  = receiver phase [degrees].  Optional.  Default = 0 (corresponds to
            x'-axis readout);
shape     = line broadening function.  Optional,
                'L' = lorentzian (default)
                'G' = gaussian
```

## 2.16. sim_rotate.m

```
USAGE:
d_out = sim_rotate(d_in,H,angle,axis)

DESCRIPTION:
This function simulates the effect of an ideal (instantaneous) rotation
on the density matrix.  Used in simulation tools.

INPUTS:
d_in      = input density matrix structure.
H         = Hamiltonian operator structure.
angle     = RF pulse flip angle (degrees).  If this value is a scalar, then
            the same flip angle will be applied to all spins in the system.
            To apply a different flip angle to the different spins in the
            system, the angle variable can be a vector of flip angles with
```

```
            length equal to the H.nspins.
 axis      = Axis of rotation ('x', 'y' or 'z'); (A z-rotation technically
             doesn't correspond to an rf pulse rotation, but it is included
             here anyway).
```

## 2.17. sim_rotate_arbPh.m

```
USAGE:
d_out = sim_rotate_arbPh(d_in,H,angle,ph)

DESCRIPTION:
This function simulates the effect of an ideal (instantaneous) rotation on the
density matrix.  Used in simulation tools.  The phase of the rf pulse can be
arbitrarily chosen.  To achieve an arbitrary phase, this code executes a
rotation about z by an angle of -phi (the rf pulse phase), then executes a
rotation about x by an angle of "angle" (the flip angle of the pulse), and
then executes a rotation back about z by an angle of phi.

INPUTS:
d_in      = input density matrix structure.
H         = Hamiltonian operator structure.
angle     = RF pulse flip angle (degrees).  If this value is a scalar, then
            the same flip angle will be applied to all spins in the system.
            To apply a different flip angle to the different spins in the
            system, the angle variable can be a vector of flip angles with
            length equal to the H.nspins.
ph        = Phase of rotation (in degrees; ie.  0='x', 90='y');
```

## 2.18. sim_runAllMetabs.m

```
USAGE:
This script is run simply by editing the input parameters and then clicking
"Run".

DESCRIPTION:
Script to generate simulated basis spectra for all metabolites of interest in
the human brain.  The script will generate an LCModel format .RAW file for
each metabolite basis spectrum, which can then be passed into LCModel's
"makebasis" function to generate a complete LCModel basis set.

INPUTS:
To run this script, edit the following parameters as desired and then click
run:
  lb       = linewidth (Hz)
  np       = Spectral points
  sw       = Spectral width (Hz)
  Bo       = Magnetic Field Strength (Tesla)
  te1      = First PRESS echo time, or SPECIAL echo time (s)
  te2      = Second PRESS echo time (if applicable) (s).
  seq      = Pulse sequence ('se'= SPECIAL, 'p'=press, 'st'=steam);
  ref      = Add reference peak at 0ppm (used in LCModel, y or n);
```

## 2.19. sim_runMegaPressShaped.m

```
USAGE:
This script is run simply by editing the input parameters and then clicking
"Run".

DESCRIPTION:
This script simulates a MEGA-PRESS experiment with fully shaped editing and
refocusing pulses.  Phase cycling of both the editing and refocusing pulses is
performed.  Furthermore, simulations are run at various locations in space to
account for the within-voxel spatial variation of the GABA signal.  Summation
across phase cycles and spatial positions is performed.  As a result of the
phase cycling and spatially resolved simulations,  this code takes a long time
to run.  Therefore, the MATLAB parallel computing toolbox (parfor loop) was
used to accelerate the siumulations.  Accelration  is currently performed in
the direction of the slice selective pulse along the x-direction, but this can
be changed.  Up to a factor of 12 acceleration can be achieved using this
approach.  To enable the use of the MATLAB parallel computing toolbox,
initialize the multiple worked nodes using "matlabpool size X" where "X" is
the number of available processing nodes.  If the parallel processing toolbox
is not available, then replace the "parfor" loop with a "for" loop.

INPUTS:
To run this script, edit the parameters below as desired and then click "run":
 refocWaveform     = name of refocusing pulse waveform.
 editWaveform      = name of editing pulse waveform.
 editOnFreq        = freqeucny of edit on pulse[ppm]
 editOffFreq       = frequency of edit off pulse[ppm]
 refTp             = duration of refocusing pulses[ms]
 editTp            = duration of editing pulses[ms]
 Bfield            = Magnetic field strength in [T]
 Npts              = number of spectral points
 sw                = spectral width [Hz]
 Bfield            = magnetic field strength [Tesla]
 lw                = linewidth of the output spectrum [Hz]
 thkX              = slice thickness of x refocusing pulse [cm]
 thkY              = slice thickness of y refocusing pulse [cm]
 x                 = vector of X positions to simulate [cm]
 y                 = vector of y positions to simulate [cm]
 taus              = vector of pulse sequence timings  [ms]
 spinSys           = spin system to simulate
 editPhCyc1        = vector of phase cycling steps for 1st editing pulse
[degrees]
 editPhCyc2        = vector of phase cycling steps for 2nd editing pulse
[degrees]
 refPhCyc1         = vector of phase cycling steps for 1st refocusing pulse
                     [degrees]
 refPhCyc2         = vector of phase cycling steps for 2nd refocusing pulse
                     [degrees]
```

## 2.20. sim_runMegaPressShapedEdit.m

```
USAGE:
This script is run simply by editing the input parameters and then clicking
"Run".

DESCRIPTION:
This script simulates a MEGA-PRESS experiment with fully shaped editing
pulses.  Phase cycling of editing pulses is performed, and summation  across
phase cycles is performed.
```

```
INPUTS:
To run this script, edit the parameters below as desired and then click "run":
 editWaveform          = name of editing pulse waveform.
 editOnFreq            = freqeucny of edit on pulse[ppm]
 editOffFreq           = frequency of edit off pulse[ppm]
 editTp                = duration of editing pulses[ms]
 Npts                  = number of spectral points
 sw                    = spectral width [Hz]
 Bfield                = magnetic field strength [Tesla]
 lw                    = linewidth of the output spectrum [Hz]
 taus                  = vector of pulse sequence timings  [ms]
 spinSys               = spin system to simulate
 editPhCyc1            = vector of phase cycling steps for 1st editing pulse
                         [degrees]
 editPhCyc2            = vector of phase cycling steps for 2nd editing pulse
                         [degrees]
```

## 2.21. sim_runMegaPressShapedRefoc.m

```
USAGE:
This script is run simply by editing the input parameters and then clicking
"Run".

DESCRIPTION:
This script simulates a MEGA-PRESS experiment with fully shaped refocusing
pulses.  Phase cycling of the refocusing pulses is performed and simulations
are run at various locations in space to account for the within-voxel spatial
variation of the GABA signal.  Summation across spatial positions is
performed.  As a result of the phase cycling and spatially resolved
simulations, this code takes a long time to run.  Therefore, the MATLAB
parallel computing toolbox (parfor loop) is used to accelerate the
siumulations.  Acceleration is currently performed in the direction of the
slice selective pulse along the x-direction, but this can be changed.  Up to a
factor of 12 acceleration can be achieved using this approach.  To enable the
use of the MATLAB parallel computing toolbox, initialize the multiple worked
nodes using "matlabpool size X" where "X" is the number of available
processing nodes.  If the parallel processing toolbox is not available, then
replace the "parfor" loop with a "for" loop.

INPUTS:
To run this script, edit the parameters below as desired and then click "run":
 refocWaveform         = name of refocusing pulse waveform.
 refTp                 = duration of refocusing pulses[ms]
 Npts                  = number of spectral points
 sw                    = spectral width [Hz]
 Bfield                = magnetic field strength [Tesla]
 lw                    = linewidth of the output spectrum [Hz]
 thkX                  = slice thickness of x refocusing pulse [cm]
 thkY                  = slice thickness of y refocusing pulse [cm]
 x                     = vector of X positions to simulate [cm]
 y                     = vector of y positions to simulate [cm]
 taus                  = vector of pulse sequence timings  [ms]
 spinSys               = spin system to simulate
 editFlipON            = vector of edit-ON pulse flip angles for each spin in
                         spin system.
 editFlipOFF           = vector of edit-OFF pulse flip angles for each spin in
                         spin system.
 refPhCyc1             = vector of phase cycling steps for 1st refocusing
                         pulse [degrees]
```

```
 refPhCyc2             = vector of phase cycling steps for 2nd refocusing
                         pulse [degrees]
```

## 2.22. sim_shapedRF.m

```
USAGE:
d_out = sim_shapedRF(d_in,H,RFstruct,flipAngle,phase,grad,pos)

DESCRIPTION:
This function simulates the effect of a shaped rf pulse on the density matrix.
The temporal shape of the refocussing pulses is modelled as a series of N
instantaneous rotations about the effective RF field, where N is the number of
time points in the RF waveform.  The instantaneous effective RF field can be
an arbitrary vector, and can be represented in polar coordinates as
B(Beff,alpha,zeta), where Beff is the magnitude field, alpha is the polar
angle (the angle between the transverse plane and the effective B field), and
zeta is the azimuthal angle, which is given by the phase of the RF).  Rotation
about the effective B-field is achieved by a composite rotation: Rotate about
Y by -alpha, rotate about Z by -zeta, then rotate about X by
2*pi*gamma*Beff*dt, then rotate back about Z by zeta and back about Y by
alpha.

INPUTS:
d_in      = input density matrix structure.
H         = Hamiltonian operator structure.
RF        = Radiofrequency pulse.  This can be the filename of a Siemens
             .pta file, or an RF pulse definition structure (obtained using
            rf_init.m)
Tp        = Pulse duration in [ms];
flipAngle = RF pulse flip angle [degrees].
phase     = Phase of RF pulse [degrees].  Optional.  Default = 0 (x'-axis.  90
            degress corresponds to +y' axis)
grad      = Gradient strength [G/cm]. Optional (for slice selective pulses
            only).
pos       = Position of spins relative to voxel centre [cm].  Optional (for
            slice selective pulses only).
```

## 2.23. sim_spinecho.m

```
USAGE:
out = sim_spinecho(n,sw,Bfield,linewidth,sys,tau)

DESCRIPTION:
This function simulates a spin-echo experiment with instantaneous RF pulses.

INPUTS:
n         = number of points in fid/spectrum
sw        = desired spectral width in [Hz]
Bfield    = main magnetic field strength in [T]
linewidth = linewidth in [Hz]
sys       = spin system definition structure
tau       = echo time in [s]
```

## 2.24. sim_spinecho_shaped.m

```
USAGE:
out = sim_spinecho_shaped(n,sw,Bfield,linewidth,sys,tau,RF,Tp,grad,pos,ph)

DESCRIPTION:
This function simulates a spin-echo experiment with instantaneous RF pulses.

INPUTS:
n         = number of points in fid/spectrum
sw        = desired spectral width in [Hz]
Bfield    = main magnetic field strength in [T]
linewidth = linewidth in [Hz]
sys       = spin system definition structure
tau       = echo time in [s]
```

## 2.25. sim_spoil.m

```
USAGE:
d_out = sim_spoil(d_in,H,angle)

DESCRIPTION:
This function simulates the effect of a rotation about the z-axis.

INPUTS:
d_in      = input density matrix structure.
H         = Hamiltonian operator structure.
angle     = Spoil angle (degrees).
```

## 2.26. sim_steam.m

```
USAGE:
out = sim_steam(n,sw,Bfield,linewidth,sys,te,tm)

DESCRIPTION:
Simulate the STEAM sequence using ideal (instantaneous) RF pulses.  To remove
unwanted coherences, a 4 step phase cycle is automatically performed, with the
first and third rf pulses being cycled by 0, 90 180, and 270 degrees.  THIS
CODE IS NOT TESTED.  RESULTS MAY NOT BE ACCURATE!!

INPUTS:
n         = number of points in fid/spectrum
sw        = desired spectral width in [Hz]
Bfield    = main magnetic field strength in [T]
linewidth = linewidth in [Hz]
sys       = spin system definition structure
te        = echo time in [s]
tm        = mixing time in [s]
```

## 2.27. sim_steam_gradSim.m

```
USAGE:
Script can be run by pressing "run".

DESCRIPTION:
```

```
This function runs the sim_steam_spoil function multiple times with different
spoiler gradient intensities.  The result is a spoiled STEAM sequence.

INPUTS:
Initialize the following variables and then click “run”:
spinsys       = Spin system.
TE            = Echo time [s].
TM            = Mixing time [s].
N             = Number of 'phase cycles'
```

### 2.28. writelcmraw.m

```
USAGE:
RF=writelcmraw(data_struct,outfile,metab);

DESCRIPTION:
Take a simulated metabolite basis spectrum in matlab structure format, and
output it into LCModel RAW format to be used in an LCModel basis spectrum.

INPUTS:
data_struct    = simulated metabolite basis spectrum in matlab structure
format.
outfile        = name of the output .RAW file.
metab          = Abbreviated name of the metabolite (ie. 'Cr', 'Glu', etc.)
```

## 3. RF Pulse Tools

### 3.1. rf_blochSim.m

```
USAGE:
[mv,sc]=rf_blochSim(RF,tp,peakB1,sc);

DESCRIPTION:
Perform a bloch simulation of an RF pulse.

INPUTS:
RF        = RF pulse definition structure
tp        = pulse duration in [ms]
sc        = Frequency span in [kHz] (optional)
peakB1    = Peak B1 amplitude in [kHz] (optional)
```

### 3.2. rf_dualBand.m

```
USAGE:
[rf,AMPINT]=rf_dualBand(tp,df,n,bw,ph,shft)

DESCRIPTION:
Creates an n-point dual banded gaussian inversion RF pulse with duration
```

```
tp(ms).  The first band will be at f=0Hz and the second band will be at df Hz.
Bw is the bandwidth of the two selection bands in Hz.

INPUTS:
tp          = pulse duration in ms.
df          = frequency of 2nd gaussian band [Hz].
n           = number of points in rf waveform.
bw          = bandwidth of both selection bands [Hz].
ph          = phase of the second gaussian.
shft        = frequency shift applied to both bands.
```

### 3.3. rf_freqshift.m

```
USAGE:
RF_shift=rf_freqshift(RF,Tp,f);

DESCRIPTION:
Apply a frequency shift to an RF pulse.

INPUTS:
RF          = RF pulse definition structure.
Tp          = duration of the rf pulse in [ms].
f           = amount that you would like to frequency shift the rf pulse in
              [Hz].
```

### 3.4. rf_gauss.m

```
USAGE:
[rf,AMPINT]=rf_gauss(tp,df,n,bw);

DESCRIPTION:
Create an n point gaussian rf waveform.  Waveform can be converted in to
siemens pta file using rf_writepta.m or into varian/agilent rf file using
rf_writerf.m.

INPUTS:
tp          = duration of rf pulse in ms.
df          = frequency of Gaussian pulse in Hz.  0 freqeuncy will correspond
              to the reference frequency of the rf transmitter.
n           = number of points in the rf waveform.
bw          = FWHM of the Gaussian inversion profile in the frequency domain
(Hz).
```

### 3.5. rf_hs.m

```
USAGE:
[rf,FM,scan,mvector]=rf_hs(outfile,N,n,tbw,Tp,trunc,thk)

DESCRIPTION:
This funciton creates any desired HS pulse.  N is the number of steps, n is
the order of the HS pulse, tbw is the time bandwidth product of the pulse, Tp
is the duration of the pulse and thk is the desired thickness of the pulse.
```

```
INPUTS:
outfile        = name of output rf file.
N              = Number of points in RF waveform.
n              = order of the HS pulse.
tbw            = Time bandwidth product.
Tp             = Duration of the RF pulse (ms).
trunc          = Truncation of the amplitude modulation function.
thk            = thickness of the slice selective pulse.
```

### 3.6. rf_loadwaveform.m

```
USAGE:
[RF_struct]=rf_loadwaveform(filename,type,f0);

DESCRIPTION:
Initialize an RF pulse structure to contain an RF Pulse waveform as well as
its accompanying header information.  This function finds the time-bandwidth
product (tbw) and the time-w1 product (tw1) of the pulse, and stores this
information in the header fields of the output RF structure.

INPUTS:
filename  = filename of RF pulse waveform text file.  Can be in Siemens format
            (.pta) or Varian/Agilent format (.RF).
type      = Excitation ('exc'), Refocusing ('ref') or Inversion ('inv')
f0        = centre frequency of the rf pulse [Hz].  Optional. Default=0.
```

### 3.7. rf_readRF.m

```
USAGE:
[rf,info]=rf_readRF(filename)

DESCRIPTION:
Read a Varian/Agilent .RF file into matlab.  The resulting RF matrix will have
3 columns specifying magnitude, phase, and duration.  This function simply
calls Martyn Klassen's readrfvnmr.m function.

INPUTS:
filename   = filename of the .RF file to read in.
```

### 3.8. rf_readpta.m

```
USAGE:
[rf,info]=rf_readpta(filename)

DESCRIPTION:
Read a Siemens .pta file into matlab.  The resulting RF matrix will have 2
columns specifying magnitude and phase.

INPUTS:
filename   = filename of the .pta file to read in.
```

### 3.9. rf_resample.m

```
USAGE:
RF_out=rf_resample(RF_in,N);

DESCRIPTION:
Resample the input RF pulse into a new waveform with N discrete points.

INPUTS:
RF        = RF pulse definition structure
N         = Number of points in new RF waveform
```

### 3.10. rf_writepta.m

```
USAGE:
RF=rf_writepta(rf,outfile);

DESCRIPTION:
Write a matlab RF pulse structure (containing 3 x N waveform array field with
rf.waveform(1,:)= phase, rf.waveform(2,:)=amplitude, and
rf.waveform(3,:)=timestep), to a siemens format .pta file.

INPUTS:
rf         = matlab RF pulse.
outfile    = name of the output .pta file to be written.
```

### 3.11. rf_writerf.m

```
USAGE:
RF=rf_writerf(rf,outfile);

DESCRIPTION:
Write a matlab RF pulse structure (containing 3 x N waveform array field with
rf.waveform(1,:)= phase, rf.waveform(2,:)=amplitude, and
rf.waveform(3,:)=timestep), to a varian/agilent format .RF file.

INPUTS:
rf         = matlab RF pulse.
outfile    = name of the output .RF file to be written.
```

## 4. Processing Tools

### 4.1. addphase.m

```
USAGE:
PhasedSpecs=addphase(specs,AddedPhase);

DESCRIPTION:
Add equal amount of complex phase to each point of a vector.
```

```
INPUTS:
specs          = Input vector.
AddedPhase     = Amount of phase (degrees) to add.
```

## 4.2. addphase1.m

```
USAGE:
PhasedSpecs=addphase1(specs,ppm,timeShift,ppm0,B0);

DESCRIPTION:
Add first order phase to a spectrum (added phase is linearly dependent on
frequency).

INPUTS:
specs      = input vector
ppm        = freqeuncy scale (ppm) corresponding to the specs vector
timeShift  = This defines the amount of 1st order phase shift by specifying
             the equivalent horizontal shift (in seconds) in the time domain.
ppm0       = The frequency "origin" (ppm) of the 1st order phase shift. (this
             frequency will undergo 0 phase shift).
B0         = The main magnetic field strength (needed since ppm depends on B0)
```

## 4.3. getLW.m

```
USAGE:
[FWHM]=getLW(in,zpfactor,Refppmmin,Refppmmax);

DESCRIPTION:
Estimates the linewidth of a reference peak in the spectrum.  Two methods are
used to estimate the linewidth:  1.  FWHM is measured by simply taking the
full width at half max of the reference peak.  2.  The FWHM is measured by
fitting the reference peak to a lorentzian lineshape and determine the FWHM of
the best fit.  The output FWHM is given by the average of these two measures.

INPUTS:
in         = input spectrum in structure format.
zpfactor   = zero-padding factor (used for method 1.)
Refppmmin  = Min of frequency range (ppm) in which to search for reference
peak.
Refppmmax  = Max of frequency range to (ppm) in which search for reference
peak
```

## 4.4. getLWandSNR.m

```
USAGE:
[FWHMmean,SNRmean] = getLWandSNR(in);

DESCRIPTION:
Calculate the linewidth and SNR of a spectrum.
```

```
INPUTS:
in    = input data in matlab structure format
```

### 4.5. getSNR.m

```
USAGE:
[SNR]=getSNR(in,NAAppmmin,NAAppmmax,noiseppmmin,noiseppmmax);

DESCRIPTION:
Find the SNR of the NAA peak in a spectrum.

INPUTS:
in             = input data in matlab structure format
NAAppmmin      = min of frequency range in which to search for NAA peak.
NAAppmmax      = max of frequency range in which to search for NAA peak.
noiseppmmin    = min of frequency range in which to measure noise.
noiseppmmax    = max of frequency range in which to measure noise.
```

### 4.6. op_ISIScombine.m

```
USAGE:
out=op_ISIScombine(in,addInd);

DESCRIPTION:
Combine dimensions corresponding to ISIS on/off acquisitions to produce fully
localized MRS volumes.  Mostly used for MEGA-SPECIAL data to combine the ISIS
subspectra but not the Edit-on/edit-off subspectra.

INPUTS:
in         = input data in matlab structure format.
addInd     = (optional) If add==1, then row indices [1 2] and [3 4] will be
             added.  Otherwise, row indices [1 2] and [3 4] will be
             subtracted.
```

### 4.7. op_addNoise.m

```
USAGE:
[out,noise]=op_addNoise(in,sdnoise,noise);

DESCRIPTION:
Add noise to a spectrum (useful for generating simulated data).  Normally
distributed random noise is added to both the real and imaginary parts of the
data.  Real and imaginary noise parts are uncorrelated.

INPUTS:
in         = Input data in matlab structure format.
sdnoise    = Standard deviation of the random noise to be added in the time
domain.
noise      = (optional)  Specific noise kernel to be added (if specified,
             sdnoise variable is ignored).
```

### 4.8. op_addScans.m

```
USAGE:
out=op_addScans(in1,in2,subtract);

DESCRIPTION:
Add or subtract two scans together.

INPUTS:
in1        = First spectrum to add (in matlab structure format)
in2        = Second spectrum to add (also in matlab structure format).
subtract   = (optional).  Add or subtract?  (0 = add, 1=subtract).  Default=0;
```

### 4.9. op_addphase.m

```
USAGE:
out=op_addphase(in,ph0,ph1,ppm0,suppressPlot);

DESCRIPTION:
add zero and first order phase to a spectrum.

INPUTS:
in             = input spectrum in matlab structure format
ph0            = zero order phase to add (degrees)
ph1            = 1st order phase to add (in seconds);
ppm0           = (optional) frequency reference point.  Default = 4.65;
suppressPlot   = (optional) Boolian to suppress plots.  Default = 0;
```

### 4.10. op_addphaseSubspec.m

```
USAGE:
out=op_addphaseSubspec(in,ph);

DESCRIPTION:
Add zero order phase to one of the subspectra in a dataset.  For example, the
edit-on spectrum of a mega-press acquisition.

INPUTS:
in    = Input spectrum in matlab structure format.
ph    = Phase (in degrees) to add to the second subspectrum.
```

### 4.11. op_addrcvrs.m

```
USAGE:
[out,fids_presum,specs_presum,ph,sig]=op_addrcvrs(in,point,mode,coilcombos);

DESCRIPTION:
Perform weighted coil recombination for MRS data acquired with a receiver coil
array.

INPUTS:
in            = input spectrum in matlab structure format.
point         = point of fid to use for phase estimation.
mode          = -'w' performs amplitude weighting of channels based on the
                maximum signal of each coil channel.
```

```
                -'h' performs amplitude weighting of channels based on the
                 maximum signal of each coil channel divided by the square of
                 the noise in each coil channel (as described by Hall et al.
                 Neuroimage 2014).
coilcombos    = (optional)  The predetermined coil phases and amplitudes as
                 generated by the op_getcoilcombos.m function.  If this
                 argument is provided, the 'point', and 'mode', arguments will
                 be ignored.
```

## 4.12. op_alignAverages.m

```
USAGE:
[out,fs,phs]=op_alignAverages(in,tmax,avg,initPars);

DESCRIPTION:
Perform spectral registration in the time domain to correct frequency and
phase drifts.  As described in Near et al.  Frequency and phase drift
correction of magnetic resonance spectroscopy data by spectral registration in
the time domain. Magn Reson Med. 2014 Jan 16. doi: 10.1002/mrm.25094. [Epub
ahead of print]

INPUTS:
in        = Input data structure.
tmax      = Maximum time (s) in time domain to use for alignment.
avg       = Align averages to the average of the averages? (y or n)
initPars  = (Optional) Initial fit parameters [freq(Hz), phase(degrees)].
            Default=[0,0];
```

## 4.13. op_alignAverages_fd.m

```
USAGE:
[out,fs,phs]=op_alignAverages_fd(in,minppm,maxppm,tmax,avg,initPars);

DESCRIPTION:
Perform time-domain spectral registration using a limited range of frequencies
to correct frequency and phase drifts.  As described in Near et al.  Frequency
and phase drift correction of magnetic resonance spectroscopy data by spectral
registration in the time domain. Magn Reson Med. 2014 Jan 16. doi:
10.1002/mrm.25094. [Epub ahead of print]

INPUTS:
in        = Input data structure.
Minppm    = Minimum of frequency range (ppm).
Maxppm    = Maximum of frequnecy range (ppm).
tmax      = Maximum time (s) in time domain to use for alignment.
avg       = Align averages to the average of the averages? (y or n)
initPars  = (Optional) Initial fit parameters [freq(Hz), phase(degrees)].
            Default=[0,0];
```

## 4.14. op_alignScans.m

```
USAGE:
[out,ph,frq]=op_alignScans(in,in1,tmax,mode);

DESCRIPTION:
```

```
Use spectral registration to align two separate scans (align in to in1);

INPUTS:
in        = input (spectrum to be registered)
in1       = base (spectrum that the input is to be registered to).
tmax      = Maximum time (s) in time domain to use for registration.
mode      = (optional)'f' - Frequency align only
                  'p' - Phase align only
                  'fp or pf' - Frequency and phase align (default)
```

## 4.15. op_alignScans_fd.m

```
USAGE:
[out,ph,frq]=op_alignScans_fd(in,in1,fmin,fmax,tmax,mode);

DESCRIPTION:
Use spectral registration on a limited frequnecy range to align two separate
MRS datasets (align in to in1);

INPUTS:
in    = input (spectrum to be registered to the base)
in1   = base (spectrum that the input is to be registered to).
fmin  = Minimum frequency for spectral alignment (ppm).
fmax  = Maximum frequency for spectral alignment (ppm).
tmax  = Maximum time (s) in time domain to use for registration.
mode  = (optional)'f' - Frequency align only
                  'p' - Phase align only
                  'fp or pf' - Frequency and phase align (default)
```

## 4.16. op_alignrcvrs.m

```
USAGE:
[out,ph,sig]=op_alignrcvrs(in,point,mode,coilcombos);

DESCRIPTION:
phase align the receiver channels without combining them.

INPTUS:
in            = input spectrum in matlab structure format.
 point         = Index of point in time domain to use for phase refernece.
 mode          = -'w' performs amplitude weighting of channels based on the
                  maximum signal of each coil channel.
                 -'h' performs amplitude weighting of channels based on the
                  maximum signal of each coil channel divided by the square of
                  the noise in each coil channel (as described by Hall et al.
                  Neuroimage 2014).
 Coilcombos    = (optional)  The predetermined coil phases and amplitudes as
                  generated by the op_getcoilcombos.m function.  If this
                  argument is provided, the 'point', and 'mode', arguments
                  will be ignored.
```

## 4.17. op_ampScale.m

```
USAGE:
out=op_ampScale(in,A);

DESCRIPTION:
Scale the amplitude of a spectrum by factor A.

INPUTS:
in    = input data in matlab structure format
A     = Amplitude scaling factor.
```

### 4.18. op_averaging.m

```
USAGE:
out=op_averaging(in);

DESCRIPTION:
Combine the averages in a scan by adding the averages together.  For
historical reasons, this function does not divide by the number of averages,
and therefore it is not really taking the average of the averages.  Dividing
by the number of averages is typically performed in a separate step using
op_avgNormalize.

INPUTS:
in    = input data in matlab structure format.
```

### 4.19. op_avgNormalize.m

```
USAGE:
out=op_avgNormalize(in,scaleFactor);

DESCRIPTION:
Divide by the number of averages.  This is typically performed after
op_averaging.m to obtain a truely "averaged spectrum", since op_averaging.m
adds the averages together but does not divide by the number of averages.

INPUTS:
in            = input data in matlab structure format.
scaleFactor   = (optional) Factor to divide by.  default = in.averages.
```

### 4.20. op_combinesubspecs.m

```
USAGE:
out=op_combinesubspecs(in,mode);

DESCRIPTION:
Combine the subspectra in an acquisition either by addition or subtraction.

INPUTS:
in        = input data in matlab structure format.
mode      = -"diff" adds the subspectra together (this is counter intuitive,
              but the reason is that many "difference editing" sequences use
```

phase cycling of the readout ADC to achieve "subtraction by addition".
-"summ" performs a subtraction of the subspectra.

## 4.21. op_complexConj.m

```
USAGE:
out=op_complexConj(in)

DESCRIPTION:
take the complex conjugate of the data;

INPUTS:
in    = Input data in matlab structure format.
```

## 4.22. op_concatAverages.m

```
USAGE:
out=op_concatAverages(in1,in2);

DESCRIPTION:
Concatenate two scans along the averages dimension.  Two scans with 50
averages each will now look like a single scan with 100 averages.

INPUTS:
in1    = first input in matlab structure format.
in2    = second input in matlab structure format.
```

## 4.23. op_creFit.m

```
USAGE:
parsFit=op_creFit(in,ph0,ph1);

DESCRIPTION:
Perform a Lorentzian lineshape fit to the creatine resonance in a brain proton
MRS dataset.

INPUTS:
in    = input data in matlab stucture format.
ph0   = zero order phase to add to the input data.
ph1   = 1st order phase to add to the input data.
```

## 4.24. op_dccorr.m

```
USAGE:
out=op_dccorr(in,mode,var);

DESCRIPTION:
Do a DC Correction on the data.  This method is a frequency domain operation.

INPUTS:
In    = input data in matlab structure format.
```

```
mode  = Point('p') or Value('v').  In point mode, the DC offset is calculated
        automatically at a specific point in the spectrum.  In value mode, the
        user has to provide the value of the desired DC offset.
Var   = If mode is 'p', then 'var' is the index of the spectral point that you
        wish to use to calculate the DC offset.  If mode is 'v', then 'var' is
        the value of the dc offset correction that you wish to employ.
```

### 4.25. op_downsamp.m

```
USAGE:
out=op_downsamp(in,dsFactor);

DESCRIPTION:  Change the time domain sampling rate of a spectrum by a factor
of 'dsFactor'.  Nearest neighbour interpolation is performed by default.

INPUTS:
in         = input data in matlab structure format.
dsFactor   = factor by which to divide the sampling rate of the fid.
```

### 4.26. op_ecc.m

```
USAGE:
[out,outw]=op_ecc(in,inw);

DESCRIPTION:
Perform an eddy current correction by estimating any non-linearity in the
phase of the water unsuppressed data in the time domain and applying the
appropriate correction to both the water suppressed and water unsuppressed
data.

INPUTS:
in     = water suppressed input data in matlab structure format.
in2    = water unsuppressed input data in matlab structure format.
```

### 4.27. op_fddccorr.m

```
USAGE:
out=op_fddccorr(in,npts);

DESCRIPTION:
Correct and DC offset in the frequency domain.  This is equivalent to a
vertical shift in the frequency domain.  The required vertical shift is
calculated by taking the average of the first and last "NPTS" points in the
frequency domain.  This requires that those points are in the noise floor.

INPUTS:
in     = input data in matlab structure format.
npts   = number of points at both edges of the freqeuncy domain that will be
         used for estimation of the DC offset of the spectrum.
```

### 4.28. op_filter.m

```
USAGE:
out=op_filter(in,lb);

DESCRIPTION:
Perform line broadening by multiplying the time domain signal by an
exponential decay function.

INPUTS:
in     = input data in matlab structure format.
lb     = line broadening factor in Hz.
```

### 4.29. op_freqAlignAverages.m

```
USAGE:
[out,fs]=op_freqAlignAverages(in,tmax,avg,initPars);

DESCRIPTION:
Perform spectral registration in the time domain using only frequency
adjustment (no phase adjustment).

INPUTS:
in         = Input data structure.
tmax       = Maximum time (s) in time domain to use for alignment.
avg        = Align averages to the average of the averages? (y or n)
initPars   = (Optional) Initial fit parameters [freq(Hz), phase(degrees)].
             Default=[0,0];
```

### 4.30. op_freqAlignAverages_fd.m

```
USAGE:
[out,fs,phs]=op_freqAlignAverages_fd(in,minppm,maxppm,tmax,avg,initPars);

DESCRIPTION:
Perform time-domain spectral registration using a limited range of frequencies
and using only frequency adjustment (no phase adjustment).

INPUTS:
in         = Input data structure.
minppm     = Minimum of frequency range (ppm).
maxppm     = Maximum of frequnecy range (ppm).
tmax       = Maximum time (s) in time domain to use for alignment.
avg        = Align averages to the average of the averages? (y or n)
initPars   = (Optional) Initial fit parameters [freq(Hz), phase(degrees)].
             Default=[0,0];
```

### 4.31. op_freqrange.m

```
USAGE:
out=op_freqrange(in,ppmmin,ppmmax);

DESCRIPTION:
Output only a specified frequency range of the input spectrum.
```

```
INPUTS:
in          = input data in matlab structure format.
ppmmin      = minimum extent of frequency range in ppm.
ppmmax      = maximum extent of frequency range in ppm.
```

### 4.32. op_freqshift.m

```
USAGE:
out=op_freqshift(in,f);

DESCRIPTION:
Apply a frequency shift to the input spectrum by 'f' Hz.

INPUTS:
in      = input data in matlab structure format.
f       = frequency shift to apply (in Hz).
```

### 4.33. op_freqshiftSubspec.m

```
USAGE:
out=op_freqshiftSubspec(in,f);

DESCRIPTION:
Apply a frequency shift to only one of the sub-spectra in a dataset.  This is
used to minimize subtraction artefacts from MEGA_PRESS data, for instance.

INPUTS:
in      = input data in matlab structure format.
f       = frequency shift to apply to subspectrum (in Hz).
```

### 4.34. op_gaussianPeak.m

```
USAGE:
out=op_gaussianPeak(n,sw,Bo,lw,ppm0,amp);

DESCRIPTION:
Generate a noiseless spectrum containing a single gaussian peak with desired
parameters (frequency, amplitude, linewidth, etc.).

INPUTS:
n       = Number of points in spectrum.
sw      = spectral width of spectrum (Hz).
Bo      = Magnetic field strength (Tesla).
lw      = Linewidth of gaussian peak (Hz).
ppm0    = Frequency of gaussian peak (ppm).
amp     = Amplitude of gaussian peak.
```

### 4.35. op_getcoilcombos.m

```
USAGE:
coilcombos=op_getcoilcombos(file_or_struct,point);
```

DESCRIPTION:
This funciton finds the relative coil phases and amplitudes. Coil phases are found by determining the phase and amplitude of the pointth point in the time domain.

INPUTS:
file_or_struct = this function will accept either a string filename or the name of a structure. If the input is a string, the program will read in the data corresponding to that filename. If the input is a structure, it will operate on that structure.
point = The index of the datapoint in the fid that is used for determination of signal intensity and phase.

## 4.36. op_getcoilcombos_specReg.m

USAGE:
coilcombos=op_getcoilcombos_specReg(file_or_struct,tmin,tmax,point);

DESCRIPTION:
This funciton finds the relative coil phases and amplitudes. Coil phases are found by fitting in the time domain. (In the original op_getcoilcombos, the phases were determined simply by observation of the pointth point in the time domain). The "Base" receiver channel is Determined as the one with the highest signal (all other coil channels will be registered to that channel.

INPUTS:
file_or_struct = this function will accept either a string filename or the name of a structure. If the input is a string, the program will read in the data corresponding to that filename. If the input is a structure, it will operate on that structure.
tmin = The earliest timepoint in the fid to be used for spectral registration.
tmax = The latest timepoint in the fid to be used for spectral registration.
point = The index of the datapoint in the fid that is used for determination of Signal intensity.

## 4.37. op_leftshift.m

USAGE:
out=op_leftshift(in,ls);

DESCRIPTION:
Remove leading datapoints from the fid to get rid of 1st order phase errors.

INPUTS:
in = input data in matlab strucure format.
ls = number of points to remove from the beginning of the fid.

## 4.38. op_loadjmrui.m

```
USAGE:
out=op_loadjmrui(filename);

DESCRIPTION:
Load a jMRUI text file into matlab structure format.

INPUTS:
filename   = filename of the jMRUI txt file.
```

### 4.39. op_loadlcmdetail.m

```
USAGE:
[metabs,corrMatrix]=op_loadlcmdetail(filename);

DESCRIPTION:
This function loads in the "detailed output" of LCModel and returns the matrix
of metabolite correlation coefficients.

INPUTS:
filename   = Filename of the lcmodel detailed output file.
```

### 4.40. op_loadspec_IMA.m

```
USAGE:
out=op_loadspec_IMA(filename,Bo,spectralwidth);

DESCRIPTION:
Loads a siemens .IMA file into matlab structure format.

INPUTS:
filename       = Filename of Siemens .IMA file to load.
Bo             = Field strength (Tesla).
spectralwidth  = spectral width of the input spectrum (Hz).
```

### 4.41. op_loadspec_twix.m

```
USAGE:
out=op_loadspec_twix(filename);

DESCRIPTION:
Reads in siemens twix raw data (.dat file) using the mapVBVD.m and
twix_map_obj.m functions from Philipp Ehses (philipp.ehses@tuebingen.mpg.de).

op_loadspec_twix outputs the data in structure format, with fields
corresponding to time scale, fids, frequency scale, spectra, and header fields
containing information about the acquisition.  The resulting matlab structure
can be operated on by the other functions in this MRS toolbox.

INPUTS:
filename   = filename of Siemens twix data to load.
```

### 4.42. op_loadspec_varian.m

```
USAGE:
out=op_loadspec_varian(filename);

DESCRIPTION:
Reads in varian .fid data using the readfid.m and readprocpar.m functions from
Martyn Klassen (mklassen@robarts.ca).

op_loadspec_varian outputs the data in structure format, with fields
corresponding to time scale, fids, frequency scale, spectra, and header fields
containing information about the acquisition.  The resulting matlab structure
can be operated on by the other functions in this MRS toolbox.

INPUTS:
filename   = filename of Varian .fid data to load.
```

## 4.43. op_lorentz.m

```
USAGE:
y=op_lorentz(pars,ppm);

DESCRIPTION:
Generate a parametrized lorentzian peak.  This function is fed into fitting
tools to enable fitting of peaks using lorentzian lineshapes.

INPUTS:
pars       = The parameters of the Lorentizan function.  This is a five
element vector consisting of the following fields:
                 Amplitude,
                 FWHM, (In Hz)
                 Centre Freq,  (In ppm)
                 baseline offset, (in Amplitude units)
                 Phase shift;  (in degrees)
ppm        = frequency axis vector (in ppm);
```

## 4.44. op_lorentz_linbas.m

```
USAGE:
y=op_lorentz_linbas(pars,ppm);

DESCRIPTION:
Generate a parametrized lorentzian peak with a linearly sloping baseline.
This function is fed into fitting tools to enable fitting of peaks using
lorentzian lineshapes.

INPUTS:
pars       = The parameters of the Lorentizan function.  This is a six
             element vector consisting of the following fields:
             [ Amplitude,
               FWHM, (In Hz)
               Centre Freq,  (In ppm)
               baseline slope, (in Amplitude units per ppm)
               baseline offset, (in Amplitude units)
               Phase shift];  (in degrees)
ppm        = frequency axis vector (in ppm);
```

### 4.45. op_lorentzianPeak.m

```
USAGE:
out=op_lorentzianPeak(n,sw,Bo,lw,ppm0,amp);

DESCRIPTION:
Generate a noiseless spectrum containing a single lorentzian peak with desired
parameters (frequency, amplitude, linewidth, etc.).

INPUTS:
n      = Number of points in spectrum.
sw     = spectral width of spectrum (Hz).
Bo     = Magnetic field strength (Tesla).
lw     = Linewidth of gaussian peak (Hz).
ppm0   = Frequency of gaussian peak (ppm).
amp    = Amplitude of gaussian peak.
```

### 4.46. op_makeFreqDrift.m

```
USAGE:
[out,fDrift]=op_makeFreqDrift(in,totalDrift,noise);

DESCRIPTION:
Add frequency drift to a dataset containing multiple averages.  This is
generally used to generate simulated datasets with phase drift.

INPUTS:
in         = input data in matlab structure format.
totalDrift = total amount of frequency drift (in Hz) to add over the whole
scan.
noise      = the standard deviation of noise to add to the frequency drift
function.
```

### 4.47. op_makePhaseDrift.m

```
USAGE:
[out,phDrift]=op_makePhaseDrift(in,totalDrift,noise);

DESCRIPTION:
Add phase drift to a dataset containing multiple averages.  This is generally
used to generate simulated datasets with phase drift.

INPUTS:
in         = input data in matlab structure format.
totalDrift = total amount of phase drift (in degrees) to add over the whole
scan.
noise      = the standard deviation of noise to add to the phase drift
function.
```

### 4.48. op_megapressproc.m

```
USAGE:
[out1_diff,out1_sum,outw]=op_megapressproc(filestring,avgAlignDomain,alignSS);
```

```
DESCRIPTION:
Processing script for Siemens MEGA-PRESS MRS data.  Includes combination of
reciever channels, removal of bad averages, freqeuncy drift correction, manual
alignment of edit-on and edit-off spectra, and leftshifting.

INPUTS:
filestring         = String variable for the name of the directory containing
                     the water suppressed .dat file.  Water unsuppressed .dat
                     file should be contained in [filestring '_w/'];
avgAlignDomain     = (Optional) Perform the spectral registration (drift
                     correction) using the full spectrum ('t'), or only a
                     limited frequency range ('f').  Default is 'f'.
alignSS            = 0 - Do not align the edit-on and edit-off subspectra.  2
                     -Perform manual alignment of edit-on and edit-off
                     subspectra.

OUTPUTS:
out1_diff          = Fully processed difference spectrum.
out1_sum           = Fully processed sum spectrum.
outw               = Fully processed water unsuppressed spectrum.
```

## 4.49. op_phaseAlignAverages.m

```
USAGE:
[out,phs]=op_phaseAlignAverages(in,Npts,avg,weighting)

DESCRIPTION:
Perform time-domain spectral registration using only phase adjustment (no
frequency adjustment).  This is rarely used.

INPUTS:
in         = Input data structure.
Npts       = Number of points in time domain to use for alignment.
avg        = Align averages to the average of the averages ('y'), or the first
             average in the series ('n');
weighting  = (Optional) Apply less weight to the later points of the fid?
```

## 4.50. op_phaseAlignAverages_fd.m

```
USAGE:
[out,phs]=op_phaseAlignAverages_fd(in,minppm,maxppm,Npts,avg,weighting)

DESCRIPTION:
Perform time-domain spectral registration using a limited range of frequencies
and using only phase adjustment (no frequency adjustment). This is rarely
used.

INPUTS:
in         = Input data structure.
minppm     = Minimum of frequency range (ppm).
maxppm     = Maximum of frequnecy range (ppm).
Npts       = Number of points in time domain to use for alignment.
avg        = Align averages to the average of the averages ('y'), or the first
             average in the series ('n');
weighting  = (Optional) Apply less weight to the later points of the fid?
```

## 4.51. op_pressproc.m

```
USAGE:
[out,out_w,out_noproc,out_w_noproc]=op_pressproc(filestring,aaDomain,tmaxin,it
erin);

DESCRIPTION:
Processing script for Siemens PRESS MRS data.  Includes combination of
receiver channels, removal of bad averages, freqeuncy drift correction, and
leftshifting.

INPUTS:
filestring:    String variable for the name of the directory containing the
               water suppressed .dat file.  Water unsuppressed .dat file
               should be contained in [filestring '_w/'];
aaDomain:      (Optional) Perform the spectral registration (drift correction)
               using the full spectrum ('t'), or only a limited frequency
               range ('f').  Default is 'f'.
tmaxin:        (Optional).  Duration (in sec.) of the time domain signal used
               in the spectral registration (drift correction). Default is
               0.2 sec.
iterin:        (Optional).  Maximum number of allowed iterations for the
               spectral registration to converge. Default is 20.

OUTPUTS:
out:           Fully processed, water suppressed output spectrum.
out_w:         Fully processed, water unsuppressed output spectrum.
out_noproc:    Water suppressed output spectrum without pre-processing (No
               bad-averages removal, no frequency drift correction).
out_w_noproc:  Water unsuppressed output spectrum without pre-processing.
```

## 4.52. op_readlcmraw.m

```
USAGE:
out=readlcmraw(filename,type);

DESCRIPTION:
Reads LCModel .RAW format into the FID-A data structure format in MATLAB.

INPUTS:
filename   = filename of LCModel raw file.
type       = type of LCModel raw file:
             'rda' - .raw file generated from Siemens RDA file
             'dat' - .raw file generated by FID-A using op_writelcm.
             'sim' - .raw file generated from FID-A simulated data.
             'raw' - not sure about this one.
```

## 4.53. op_readlcmraw_basis.m

```
USAGE:
out=readlcmraw_basis(filename);

DESCRIPTION:
Reads LCModel .raw model spectrum file into the FID-A data structure format in
```

```
MATLAB.

INPUTS:
filename   = filename of LCModel raw file.
```

### 4.54. op_rmNworstaverages.m

```
USAGE:
[out,metric,badAverages]=op_rmNworstaverages(in,n);

DESCRIPTION:
Removes motion corrupted averages from a dataset containing multiple averages.
The N most badly motion corrupted averages are discarded.

INPUTS:
in         = input data in matlab structure format
n          = number of bad averages to remove
```

### 4.55. op_rmbadaverages.m

```
USAGE:
[out,metric,badAverages]=op_rmbadaverages(in,nsd,domain);

DESCRIPTION:
Removes motion corrupted averages from a dataset containing multipleaverages.
Bad averages are identified by calculating a 'likeness' metric for each
average.  This is done by subtracting each average from the average of the
averages, and then calculating the root mean squared of this difference
spectrum.  Averages whose likeness metrics are greater than 'nsd' above the
mean are discarded.

INPUTS:
in         = input data in matlab structure format
nsd        = number of standard deviations to use a rejection threshold
domain     = domain in which to perform calculations ('t' or 'f')
```

### 4.56. op_rmworstaverage.m

```
USAGE:
[out,metric,badAverages]=op_rmworstaverage(in,n);

DESCRIPTION:
Removes motion corrupted averages from a dataset containing multiple averages.
The most badly motion corrupted average is discarded.

INPUTS:
in         = input data in matlab structure format
```

### 4.57. op_specialproc.m

```
USAGE:
[out,out_w,out_noproc,out_w_noproc]=op_specialproc(filestring,aaDomain,tmaxin,
```

iterin);

DESCRIPTION:
Processing script for Siemens SPECIAL MRS data. Includes combination of receiver channels, removal of bad averages, freqeuncy drift correction, and leftshifting.

INPUTS:
filestring: String variable for the name of the directory containing the water suppressed .dat file. Water unsuppressed .dat file should be contained in [filestring '_w/'];
aaDomain: (Optional) Perform the spectral registration (drift correction) using the full spectrum ('t'), or only a limited frequency range ('f'). Default is 'f'.
tmaxin: (Optional). Duration (in sec.) of the time domain signal used in the spectral registration (drift correction). Default is 0.2 sec.
iterin: (Optional). Maximum number of allowed iterations for the spectral registration to converge. Default is 20.

OUTPUTS:
out: Fully processed, water suppressed output spectrum.
out_w: Fully processed, water unsuppressed output spectrum.
out_noproc: Water suppressed output spectrum without pre-processing (No bad-averages removal, no frequency drift correction).
out_w_noproc: Water unsuppressed output spectrum without pre-processing.

## 4.58. op_specialproc_fmrs.m

USAGE:
[out_stimOFF,out_stimON,out_w]=op_specialproc_fmrs(filestring,blockDesign,leadingAvgsToRmv);

DESCRIPTION:
Processing script for functional MRS data acquired using the SPECIAL MRS sequence. Includes combination of reciever channels, removal of bad averages, freqeuncy drift correction, and leftshifting. Also includes prior knowledge of the stimulation paradigm (given by the 'blockDesign' vector) and returns the averaged stimulus OFF and simulus ON spectra.

INPUTS:
filestring: String variable for the name of the directory containing the water suppressed .dat file. Water unsuppressed .dat file should be contained in [filestring '_w/'];
blockDesign: This is a vector of positive and negative even integers that make up the ON/OFF block design. Each integer represents the number of sequential averages in a block. Positive integers refer to ON blocks, and negative integers refer to OFF blocks. For example, for a block design consisting of 30 OFF averages followed by 20 ON averages followed by 10 OFF averages, the blockDesign vector would be: [-30 20 -10];
leadingAvgsToRmv: The number of averages to omit from the beginning of each block. This is done to account for a lag in the neurochemical response to stimulus. Must be an even integer. (optional. Default=0);

OUTPUTS:

```
out_stimOFF:       Fully processed water suppressed spectrum from the sum of
                   the stimulus OFF periods.
out_stimON:        Fully processed water suppressed spectrum from the sum of
                   the stimulus ON periods.
out_w:             Fully processed, water unsuppressed output spectrum.
```

## 4.59. op_specialproc_fmrs_slidingWindow.m

```
USAGE:
[out1,out_w] = op_specialproc_fmrs_slidingWindow(filestring,windowSize);

DESCRIPTION:
Processing script for functional MRS data acquired using the SPECIAL MRS
sequence.  Includes combination of reciever channels, removal of bad averages,
freqeuncy drift correction, and leftshifting.  This code generates a 'sliding
window timecourse' of MR spectra by combining the averages within a small
window given by the windowSize argument, and then sliding the window by 1
average and combining again.  Each summed window is output as an LCModel text
file to be analyzed in LCModel.  As a result, this function generates many
text output files.

INPUTS:
filestring:        String variable for the name of the directory containing
                   the water suppressed .dat file.  Water unsuppressed .dat
                   file should be contained in [filestring '_w/'];
windowSize:        This is an integer that specifies the number of averages
                   that are stored within the sliding window.  It is
                   recommended to choose a window size that is divisible by
                   the number of phase cycles so that the window does not
                   contain any partial phase cycles.

OUTPUTS:
out1:             The first sliding window spectrum.
out_w:            Fully processed, water unsuppressed output spectrum.
```

## 4.60. op_subtractScans.m

```
USAGE:
out=op_subtractScans(in1,in2);

DESCRIPTION:
Subtract input 2 from input 1;

INPUTS:
in1        = 1st input data in matlab structure format.
in2        = 2nd input data in matlab structure format.
```

## 4.61. op_takeaverages.m

```
USAGE:
out=op_takeaverages(in,index);

DESCRIPTION:
Extract the averages with indices corresponding to the 'index' input array.
```

```
INPUTS:
in     = input data in matlab structure format.
index  = vector indicating the indices of the averages you would like to
extract.
```

### 4.62. op_takesubspec.m

```
USAGE:
out=op_takesubspec(in,index);

DESCRIPTION:
Extract the subspectra with indices corresponding to the 'index' input array.

INPUTS:
in     = input data in matlab structure format.
index  = vector indicating the indices of the subspectra you would like to
extract.
```

### 4.63. op_unfilter.m

```
USAGE:
out=op_unfilter(in,lb);

DESCRIPTION:
Multiply the fid by an inverted exponential decay function to undo the effects
of filtering.

INPUTS:
in     = input data in matlab structure format.
lb     = line narrowing factor (Hz).
```

### 4.64. op_writejmrui.m

```
USAGE:
RF=op_writejmrui(in,outfile);

DESCRIPTION:
Takes MRS data in matlab structure format and writes it to a text file that
can be read by jMRUI.

INPUTS:
in         = input data in matlab structure format.
outfile    = Desired filename of output text file.
```

### 4.65. op_writelcm.m

```
USAGE:
RF=op_writelcm(in,outfile,te);

DESCRIPTION:
Takes MRS data in matlab structure format and writes it to a text file that
```

```
can be read by LCModel.

INPUTS:
in         = input data in matlab structure format.
outfile    = Desired filename of output text file.
te         = Echo time of acquisition (in ms).
```

### 4.66. op_zeropad.m

```
USAGE:
out=op_zeropad(in,zpFactor);

DESCRIPTION:
Apply zeropadding (a.k.a. zero-filling) to MRS data.

INPUTS:
in         = input data in matlab structure format.
zpFactor   = the factor by which the number of points in the fid will be
             increased.  ie.  if zpFactor =2, then the number of zeros added
             to the end of the fid will be equal to the number of points in
             the original spectrum.
```

### 4.67. op_zerotrim.m

```
USAGE:
out=op_zerotrim(in,numPointsToTrim);

DESCRIPTION:
Remove zeros (or even non-zero data points) from the end of the fid.

INPUTS:
in                = input data in matlab structure format.
numPointsToTrim   = The number of points to trim from the end of the fid.
```

### 4.68. readlcmcoord.m

```
USAGE:
out = readlcmcoord(filename,metab)

DESCRIPTION:
Reads a LCModel .coord file and extracts the desired part.

INPUTS:
filename   = filename of the LCModel .coord file.
part       = Which metabolite fit to extract from the .coord file - The
             abbreviated metabolite name should be given (ie.
             'Cr','PCr','Glu','GABA',etc.)
```

### 4.69. readlcmcoord_getBackground.m

```
USAGE:
out = readlcmcoord_getBackground(filename,part)
```

```
DESCRIPTION:
Reads a LCModel .coord file and extracts the desired part.

INPUTS:
filename   = filename of the LCModel .coord file.
part       = Which part of the .coord file to extract - 'bg' extracts the
             LCModel baseline signal, 'sp' extracts the spectrum, and 'fit'
             extracts the fit.
```

## 4.70. readlcmtab.m

```
USAGE:
out = readlcmtab(filename)

DESCRIPTION:
Reads a LCModel .table output file and stores the metabolite concentrations
into a matlab structure array.

INPUTS:
filename   = filename of the LCModel .table file.
```