More details on the Code of "NOLIMS"

"NOLIMS.m" is the central place from which you run the simulation but also choose the settings you wish to have.

After some general settings:

```
%constants
settings.general.gamma = 2.675221874411*10^8;          % in rad/s/T; gyromagnetic ratio
settings.general.u0=1.26e-6;                            %u_0.
settings.general.T = 293.15;                            %Temperature in K: 20°C
settings.general.k = 1.380649e-23;                      %Boltzmann constant
settings.general.hbar = 1.05457182e-34;                 %hbar, reduced Planck's constant

%Field Setting
settings.general.B0 = 3;
settings.general.FreqField = settings.general.gamma / (2*pi)*10^-6 * settings.general.B0;
```

“more relevant” settings for the simulation can be selected.

```
%General settings for included effects
settings.general.noise = false;
settings.signal.SNR = 33;

settings.general.T2 = true;
settings.general.Suscept = true;
settings.general.bool_IVD = false;
settings.general.bool_IVD_reco = false;
settings.general.BlochSim = false;
settings.general.CoilSens = false;
settings.general.LowFieldw = false;
settings.general.T1Effect = false;
settings.general.B0MapImport = false;
settings.general.B0MapInReco = false;
settings.general.B1MapImport = false;

settings.general.RAMSavingReco = true;
settings.general.gpuComp = false;
settings.general.RAM_StepsPhaseEnc = 100;
```

Noise: noise & SNR

Add gaussian white noise to simulated signal with SNR you choose

Relaxation Effects: T2 & T1Effect

T2: If chosen T2 decay is added in the signal equation

T1Effect: Assumes conventional Bloch-Equation solution, homogeneous RF excitation and a magnetic field in z-direction, T1 map is not changed due to B1 variations
The effect is simulated by scaling the initial magnetization according to T1,TR, flip angle after the first RF excitation.

Susceptibility Effect: Suscept

If chosen, the additional magnetic field due to materials with different susceptibilities in a magnetic field is calculated and considered in the simulation. For the calculation of the additional magnetic field it is assumed that the external magnetic field is homogeneous and in z-direction.

Intravoxel dephasing (IVD): bool_IVD / bool_IVD_reco

```
settings.signal.matrixsize_signal = 8;

settings.reco.matrixsize_reco = 8;
```

Intravoxel dephasing, i.e. signal decay due to magnetic field gradients over voxels (in the file “ArbFieldsImport_MoreFlex_IVD.m”) can be simulated by either choosing “matrixsize_signal > matrixsize_reco” (might be hardware/time demanding) or more RAM efficient option “bool_IVD”: This means, that in the simulation each gradient (3 directions) over a voxel is calculated in order to simulate the effect on the signal according to eq. (13).
“bool_IVD_reco” incorporates IVD into the signal model for reconstruction.

Bloch Simulation: BlochSim

If true: The effect of an excitation pulse on an initial distribution of magnetization is calculated via a Bloch Simulation. There are already three different options for RF pulses: “block”, “sinc” and “gaussian”. Further details can be set as shown in the screenshot below.
Frequency modulated pulses can also be implemented, for that, the array “w” in “BlochSim3D_RAMEff_Vector.m” has to be adapted. Currently, “w”, the RF frequency is calculated as mean value of the underlying magnetic field during excitation (l.34/35).
The simulation is done in the lab frame considering all vector components of B0 and B1. The resulting magnetization vector is projected into the plane perpendicular to B0 since this is the detectable magnetization. Care must be taken by choosing the used time steps to also detect possibly fast oscillations and reduce numerical errors.

If false: Then, it is assumed that the initial magnetization is the detectable magnetization and lies now in the y-plane.

```
%Bloch Simulation of TX pulse
settings.TX.pulse_type = 'block';%Alternatives: block, sinc, gaussian10

settings.TX.PulseLength = 30.4*10^-6;%s
switch settings.TX.pulse_type   %Pulse integrals etc. from De Graaf, Robin A. In v
    case 'block'
        settings.TX.Rp = 1.37;
        settings.TX.pulse_integral = 1;
    case 'sinc'
        settings.TX.Rp = 5.95;
        settings.TX.pulse_integral = 0.1775;
    case 'gaussian10'
        settings.TX.Rp = 2.0;
        settings.TX.pulse_integral = 0.565;
    otherwise
        warning('Not implemented');
end

settings.TX.PulseBW = settings.TX.Rp/settings.TX.PulseLength; %Hz:  bandwidth of e
settings.TX.FlipAngle = 90*pi/180;   %flip angle in rad
settings.TX.PulseAmpl = settings.TX.FlipAngle/(settings.general.gamma*settings.TX.
settings.TX.ExcSensitivity=1;
settings.TX.AngleRF_Theta = 0; %rad; phase of excitation pulse

settings.TX.NumberSamplesBloch = 100;%numbers of time steps for Bloch Simulation
```

Coil sensitivity: CoilSens

If true: Simulation of a Receive array (coils on a kind of cylindrical surface) used for detection. The principle of reciprocity is used to calculate the coil sensitivity, i.e. the Biot-Savart law is used to calculate the magnetic field that is created by the setup. Then, the detectable magnetization is calculated by projection of B1 on the plane perpendicular to B0. To account for spatial sensitivity differences, in “CoilSensB1.m”, the Coil Sensitivity maps are calculated and normalized to the highest occurring value.

```
% RX Coils
settings.CoilSens.Ncoil_segments =550;
settings.CoilSens.Curr=1;
settings.CoilSens.NReceiveCoils=8;
settings.CoilSens.RadiusReceiveCoils = 0.1;
settings.CoilSens.RadiusCoilArray = (settings.reco.FOV - 0.005)/2;
settings.CoilSens.DistMaskWire = 0.001;
```

If false: A uniform Coil Sensitivity is assumed. B1's direction can be chosen, currently, it is in y-direction.

Additional ω(r) in Signal equation: LowFieldw

If true, the local resonance frequency ω(r) is not omitted in the signal equation as it is usually done for high field MRI (homogeneous magnetic field). It might be that other (not simulated) effects such as Q of the coil dominate, however, we wanted to keep the possibility to simulate the effect of this term.

B0 Map: B0MapImport & B0MapInReco

If desired, a B0-Map (amplitude & direction) can be imported and considered during the simulation. Currently, the magnetic field is added to the encoding fields (since it is a ZTE-like sequence) in "ArbFieldsImport_MoreFlex(_IVD).m". You should consider adding it also in "calcMeanB0.m" depending on the setup you want to simulate. In "calcMeanB0.m", the mean magnetic field is updated and with that also the mixing frequency of demodulation of the simulated signal.

With "B0MapInReco", the magnetic field map is also considered during reconstruction.

B1 Map: B1MapImport

If desired a B1-Map (for excitation) can be imported and considered during the Bloch-Simulation of the excitation pulse.

Otherwise, in "BlochSim3D_RAMEff_Vector.m" a B1-Map can be set up manually. Currently, a resonant B1 in the x-y-plane is implemented.

RAM reducing parameters: RAMSavingReco & gpuComp & RAM_StepsPhaseEncode

To reduce hardware demands, two options were implemented to trade RAM demands with runtime.

"RAMSavingReco": Instead of a preconditioned conjugate gradient reconstruction (PCG), an algebraic reonstruction technique (AART) can be chosen for reconstruction. The advantage of AART is, that per iteration step only one row of the encoding matrix is necessary reducing the matrix sizes required for reconstruction. If a NVIDIA gpu is available the AART reconstruction might be accelerated by choosing "gpuComp = true".

"RAM_StepsPhaseEncode": To reduce the matrix sizes during simulation of the MRI signal, not all "phase encoding"-steps / projections are considered at once, rather with this parameter there exists the possibility to reduce the projections per iteration by this factor. For example, if you have 10000 projections and choose RAM_StepsPhaseEncode=100, then, only 100 projections are calculated at once with the drawback that a for-loop is necessary which loops from 1 to 100. In the end, the signals are all concatenated and are equal to the case in which all projections are calculated at once.

Sample & Coordinate System:

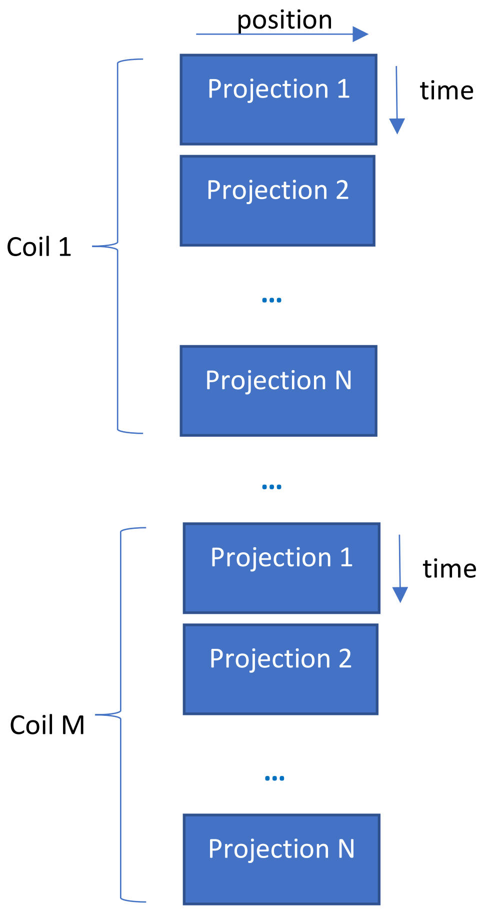
The coordinate system is defined as:

```
%Coordinate System
%Field is assumed to be in z-direction
%%%%%%%%%%%%%
% î
% x       ________
%       _|______  |
%     _|_______ | |
%    |        | | |
%    |        | |_|    z
%    |        |_|      ^
%    |________|       /
%
% -> y
% first dim: x: 1 =pos ->end neg
%second dim: y: 1= neg -> end pos
%third dim: z: 1=neg -> end pos

%Coordinate systems if matrix is shown with as()
%if x constant:       | -> z
%                   y v
%
%if y constant:      x
%                     î -> z
%
%if z constant:   x
%                   î -> y

%%%%%%%%%%%%%
```

As sample there exist multiple options. You can choose one of the following: “SheppLogan” (for 3D not really useful), “Sphere”, “Cylinder” or you can import a custom phantom – currently it is the brain phantom, which JEMRIS also uses.

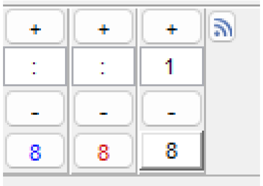
Encoding Matrix: Reconstruction

For reconstruction, the encoding matrix is structured as follows



Note to “ArrayShow”:

“ArrayShow” is a nice toolbox to visualize matrices. By default the first to dimensions are displayed, leaving the possibility to scroll in the last dimension starting from “1”. These buttons



enable to scroll through the matrix by using “+” or “-“ or by scrolling with the mouse wheel. Sometimes errors pop up, then, I would recommend to close the window which makes problems and reopen the array you want to have a look at.