Bloch-McConnell simulation and fit – BMsimfit

Documentation

Moritz Zaiss   
High Field Magnetic Resonance, Max Planck Institute for Biological Cybernetics  
Speemannstrasse 41  
Tübingen, Germany   
[moritz.zaiss@tuebingen.mpg.de](mailto:Moritz.zaiss@tuebingen.mpg.de)

# Overview

BMsimfit is a Matlab tool that allows to simulate the Bloch-McConnell equations for multiple CEST pools, including the water pool and a semisolid MT pool. This can be used to simulate Z-spectra, but also to fit multi-Z-spectrum data. For full quantification multi-B1 Z-spectrum data is required which can directly be fitted.

In this documentation, it is shown how multi-Z-spectrum data must be loaded to Matlab to use the simultaneous fit routines. In addition to Z-spectra, also measured or at least estimated parameters must be provided to the fit, e.g. T1. Further, for all parameters not provided but fitted, starting values and boundaries must be provided.

# Table of contents

1. **BM fitting**
   1. Load data and measurement parameters from xls
   2. Set up of all simulation parameters; (pool system parameters, MR system and pulse sequence parameters)
   3. find good starting values for the fit
   4. apply Multi-Z-spectrum fitting
   5. Single-offset methods
2. **BM simulation only**
   1. cw
   2. pulsed
   3. whole parameter space

# **BM fitting**

Running code for this section can be found in **“doc/1\_BMfit\_and\_load.m”** with the same section numbering.

#### **A) Loading Z-spectrum Data form XLS**

The data needs to be loaded in the form

w\_x % 1-dim offsetlist

Z\_x % 2-dim Z-spectrum stack Z\_x(w\_x,varyval)

P % struct including measurement parameters

Also the measurement parameters must be given in P.

P.FREQ = 300; % e.g. 7 \* 42.5764; [MHz] scanner static field strength

P.Trec= 3; % recovery time before saturation [s]

P.tp=10; % saturation pulse duration [s]

P.R1A=1/1.200; % R1 relaxation rate (water) [Hz]

P.M0=1; % fully relaxed M0 magnetization signal

P.Zi=1; % initial magnetization before saturation block

P.normalized=[]; % offset used for normalization

P.pulsed=0 % pulse train (1) or cw (0) ?

P.n=1 % number of saturation pulses

P.DC=0.5 % duty cycle

P.vary={'B1'}; % varied parameter (B1 for QUESP experiments)

P.varyval=[10 15 20 25 30 35]; % value of varied parameter (here B1 in µT)

To create these variables automatically form an xls file you can use the function (matlab section 1.1A)

[w\_x, Z\_x, P, rowname, Ztab]=LOAD\_xls\_qCEST\_2\_Ztab({'15degC'});

The xls file must be in the exact same structure than given in the example file “XLS\_qCEST\_name\_2017\_12\_31.xlsx” and the string in the argument is the sheet name.

If you run this section in Matlab, your data should be loaded and displayed directly.

If you just want to load a single dataset you can now go on with section 1.2.

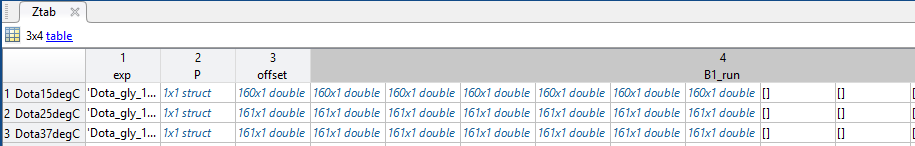
#### **B) Loading Z-spectrum Data from XLS to Ztab structure**

If you want to load all the sheets from a xls file, this can be loaded to a so called Matlab table, a table that includes all the different measurements including their parameters (matlab section 1.1B).

[w\_x, Z\_x, P, rowname, Ztab]=LOAD\_xls\_qCEST\_2\_Ztab();

The Ztab variable, which is a Matlab table is now created. Ztab contains a complete quantitative Z-spectrum in each row of the Matlab table.

The colums of the Ztab are given by ( just double click on the variable to see)



The row name is the short identifier, e.g. “Dota15degC”, used to call each row. “exp” is a string that describes the experiment, it appears in the title of plots later for example. “P” is the parameters struct containing the measurement parameters. “offset” is again the one dimensional offset list. It will be loaded to P.xZspec. “B1\_run” contains the multiple Z-spectra, e.g. for different saturation power. In “FIT” the fit setup and later the results are stored.

To extract the corresponding data for one data set and also check it at the same time the function plot\_tab can be used by giving the row name.

rowname='Dota15degC'; %

figure(2001), [w\_x, Z\_x, w\_xx, Z\_xx, P.varyval, P.vary, P]= plot\_tab(Ztab,rowname,'B1\_run');

before running the fits the Ztab data can also be manipulated (Normalization, B0 correction exclude offsets:

Ztab(rowname,:) = b0correct\_run(Ztab(rowname,:),'B1\_run') ;

[ Ztab(rowname,:), Mnorm] = norm\_run(Ztab(rowname,:),'B1\_run',P.normalized) ;

Ztab(rowname,:) = exclude\_run(Ztab(rowname,:),'B1\_run',-80)

if then loaded again using

[w\_x, Z\_x, w\_xx, Z\_xx, P.varyval, P.vary, P]= plot\_tab(Ztab,rowname,'B1\_run');

the manipulated data is loaded and you can now go on with section 1.2.

#### **Set up of all simulation parameters**

As described in Appendix A the BM-equations require all pool system parameters, MR system and pulse sequence parameters. All these parameters must be given in the simulation parameter struct **Sim**. Some parameters can also be given in the measurement parameters struct **P** which will overwrite the values in **Sim.**

Just adjust the code in section 1.2 of the file **“doc/1\_BMfit\_and\_load.m”** according to your parameters and model.

See also section 2. BM\_simulation.

#### **find good starting values for the fit**

This is actually the core of the fitting magic. We need to tell the with which are the depending variables, that are fitted and with which starting values and bounderies. This info is stored in the struct T. In Section 1.3 in the matlab file you will find blocks for each pool like this

% XXXXXXXXXXXXXXXXXXXXXXXXXXXX

% XXXXXXXXXX POOL A XXXXXXXXXX

T.varyA = [ 1 0 1 ];

T.dep\_varsA = { 'dwA', 'R1A', 'R2A' };

T.startA = [ Sim.dwA Sim.R1A Sim.R2A ];

T.lowerA = [ Sim.dwA-1 1/4\*Sim.R1A 1/10\*Sim.R2A ];

T.upperA = [ Sim.dwA+1 2\*Sim.R1A 20000\*Sim.R2A ];

The field vary activates a parameter for fitting (1) or deactivates it (0); in the latter case tha value in Sim or P of this parameter will be provided as foxed value to the fit. The field dep\_varsA just identifies the actual depending parameter name. The fields start, lower and upper give the staring values and boundaries for each parameter of the pools. In section 1.3 in the matlab file you will 4 blocks like this for pool A, B, C, D

However, block C is commented as it is always is reserved for ssMT which is not used here. As you see the start values and boundaries defined here depend on the Sim parameters, thus if choosing Sim wisely the satr vaues and boundaries might be set well, too. Thus block 1.2 and 1.3 can be repeated until a good match between data and simulation is achieved. The function

figure(2), GUESS=multiZplot(P,Sim,FIT.T,w\_x,Z\_x); % plot guess with data

The function multiZplot plots the simulation for the setup parameters together with the data. As soon as you have a rough match of data and simulation (peak positions, width and effect strength) you can go on with the actual fit in the next section.

#### **apply multi-Z-spectra fitting**

If this was performed the optimization step 1.4 can be started using multiZfit and finally plotting all relevant plots and parameters using multiZplot. If wanted all parameters and results can be stored in the Ztab column FIT

%% 1.4 multi-Z-BMfitting of the Z\_x data (wherever it comes from)

Sim.analytic=1; % set this to 1 if analytic fit should be used, (=0) numeric

Sim.n\_cest\_pool=2; % last minute change of simulation parameter

% fit-options

[FIT] =multiZfit(P,Sim,FIT.T,w\_x,Z\_x);

multiZplot(P,Sim,FIT.T,w\_x,Z\_x,FIT.popt,FIT.pci);

title([expname ' : ' rowname]);

FIT.P=P;

if exists('Ztab') % save fitresult in Ztable

Ztab(rowname,'FIT')={{FIT}};

end;

#### apply single-offset methods (QUESP and omega plots)

The single offset evaluations QUESP and Omega plot can be applied to the same data by section 1.5. The data is loaded at the offset given by single\_ofset. Here again start values and boundaries have to be given. Run the script to automatically create QUESP and Omega plot fits.

%% 1.5 single offset preparation

single\_offset=FIT.dwB(1);

figure(), subplot(2,1,1);

[Zlab Zref]=prep\_Zlab\_Zref(P,single\_offset,w\_x,Z\_x);

% boundaries for fb kb

opts.Lower = [0.0000135 0 ];

opts.StartPoint = [0.000135 4000 ];

opts.Upper = [0.0135 1500000];

%some evaluations need a value for R2B, the

P.R2B=0; P.B1cwpe\_quad=-1; P.dwB=single\_offset;

[que] =QUESP(Zlab,Zref,single\_offset,P,opts);

subplot(2,3,6);

ome=OmegaPlot(Zlab,Zref,single\_offset,P,opts);

Ztab(rowname,'QUESP')={{que}};

Ztab(rowname,'Omega')={{ome}};

clear que ome;

#### save all DATA fits and single offset reults back to the Ztab

%% 1.6 save Ztab

uisave('Ztab','Ztab\_DOTA')

#### reload a Ztab and run fits again.

%% 1.7 reaload Ztab (after this run 1.4 or 1.5 again)

% open Ztab and check for rowname and also column name of fit(FIT3p here)

rowname='Dota37degC';

Sim=Ztab.FIT3p{rowname}.Sim

FIT=Ztab.FIT3p{rowname}

expname=Ztab{rowname,'exp'}{1}; P=Ztab{rowname,'P'}{1};

figure(2001), [w\_x, Z\_x, ~, ~, varyval, P.vary, P]= plot\_tab(Ztab,rowname,'B1\_run');

# Bloch-McConnell simulation only

As described in Appendix A the BM-equations require all pool system parameters, MR system and pulse sequence parameters. All these parameters must be given in the simulation parameter struct **Sim**. Some parameters can also be given in the measurement parameters struct **P** which will overwrite the values in **Sim.**Running code for this section can be found in **“doc/2\_1\_simulation.m”.**

**Pool system**The pool system parameters are given in figure. For each pool the relative proton fraction fi, the chemical shift dwi, the relaxation rates R1i and R2i, and the exchange rate kiA must be given. All concentration fraction are always relative to water pool A which has fA=1. The back-exchange rates kAi are always calculated by kAi=fi\*kBA;

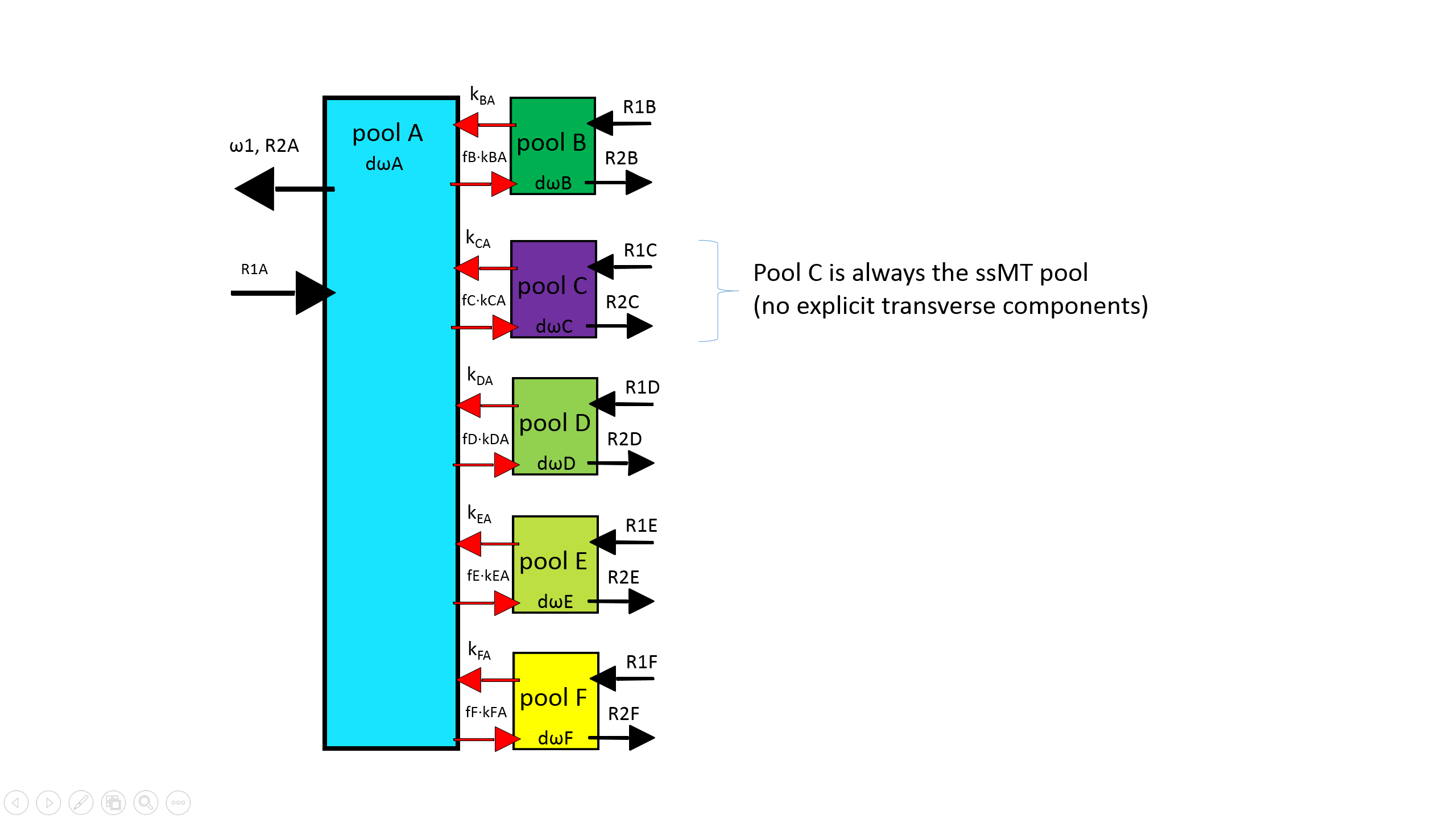


Figure : magnetization pool scheme for water pool (A), 4 CEST pools (B,D,E,F), and a semisolid MT pool (C).

The Sim struct for water with one CEST pool needs to be initialized similar to this:

**Sim.n\_cestpool=1;**

**Sim.dwA=0;  
Sim.R1A=1;  
Sim.R2A=10;**

**Sim.fB=0.01;  
Sim.kBA=100; % Hz or 1/s  
Sim.dwB=2; % ppm relative to dwA  
Sim.R1B=1;   
Sim.R2B=50;**

Until now, maximum 4 CEST pools can be simulated. As a shortcut and to save and load pool systems some exemplary systems are hardcoded in the function getSim().

**MR system and sequence**

As the simulated sequence is very simple (initial magnetization Mi decays/recovers during cw irradiation or an irradiation pulses train. The only scanner property is the field strength in MHz  
**Sim.FREQ=300;**For continues wave irradiation the saturation block is given by the B1 amplitude, the duration of the pulse tp; the shape of the pulse should be ‘block’ and number of pulses should be n=1. As the initial magnetization Zi=Mi/M0 must not be equal to 1 it has also to be given to the simulation.

**Sim.Zi=0.9;  
Sim.B1=1; % the saturation B1 in µT  
Sim.tp=5; % [s]  
Sim.n=1; Sim.DC=1;   
Sim.shape=’block’;   
Sim.pulsed=0;**

To tell the simulation that its cw the flag Sim.pulsed has to be set to 0. Otherwise the fast cw simulation is calculated in small time steps and performance is lost.

In case of pulsed saturation the flag needs to be set to 1 and a different pulse shape can be given.

**Sim.pulsed=0;   
Sim.Zi=0.9;  
Sim.B1=1; % the saturation B1 in µT  
Sim.tp=0.1; % [s]  
Sim.DC=0.5;   
Sim.n=15;   
Sim.shape=’seq\_gauss’;**

In all cases the offset frequencies of the saturation tat are the x-axis of the Z-spectrum must be given in ppm

**Sim.xZspec = -5:0.1:5;**

**Further required simulation parameters**

The simulation has some further flags that are required to set, or can be changed to get a more sophisticated setup. It can be further chosen which kind of simulation will run, numeric analytic or both.

**Sim.analytic = 1; % calculate analtical solution? 1=yes, 0=no**

**Sim.numeric = 1; % calculate numerical solution? 1=yes, 0=no**

**Sim.MT = 0; % 1 = with MT pool (pool C), 0 = no MT pool**

**Sim.Rex\_sol = 'Hyper'; % solution for Rex - cases: 'Hyper',**

**Sim.MT\_lineshape = 'Gaussian'; % MT lineshape -SuperLorentzian, Gaussian, Lorentzian**

**Sim.MT\_sol\_type = 'Rex\_MT'; % Rex\_MT solution type - cases: 'Rex\_MT'**

Having set up a BM system the simulation can just be run calling the functions

**NUMERIC\_SIM(Sim);**

**ANALYTIC\_SIM(Sim);**

Or just run the script in the doc folder “**1\_1\_simulation.m**”.  
You should also be able to understand now the script BM\_simulation.m that simulates a whole parameter space.

# Appendix: Implementation

### The Bloch–McConnell (BM) equations

We consider a 2–pool system, i.e., a system of two coupled spin populations, comprising pool a (the water pool) and pool b (the dilute solute pool), with forward (b 🠖 a) exchange rate kb and thermal equilibrium magnetizations M0,a and M0,b, respectively, in a static magnetic field B0 = (0, 0, B0). The ratio of these magnetizations, M0,b/M0,a = fb, is conserved by the backward (a 🠖 b) exchange rate ka = fbkb. Pool a is always the pool which is measured, here and in proton CEST it is the spin ensemble of water protons. In the case of HyperCEST, pool a is that of xenon in the solvent.

The temporal evolution of the magnetizations of the two pools during rf irradiation is well described by the BM equations for the case of exchange between pool a and b and by the BS equations for dipolar–coupled systems.

In the rotating frame of reference (x, y, z) defined by the frequency ωrf of the oscillating field B1(t), the BM equations read:

|  |  |
| --- | --- |
|  | (1) |
| with the 6–dimensional magnetization vector |  |
| . | (2) |

A is a block matrix

|  |  |
| --- | --- |
|  | (3) |

consisting of 3x3 submatrices Ki = ki∙Ι (see also eqs. (6)(8)(9)), and Li:

|  |  |
| --- | --- |
|  |  |
|  | (4) |
| where i = a, b. Finally, the constant vector C is |  |
|  | (5) |

The quantity Δω = Δωa = ωrf − ωa is the frequency offset relative to the Larmor frequency ωa of pool a (for 1H: ωa/B0 = γ = 267.5 rad/μTs). The offset of pool b: Δωb = ωrf − ωb = Δω − δbωa, is shifted by δb relative to the water proton resonance.

At field strengths of clinical MR imaging systems, longitudinal relaxation rates R1,a/b = 1/T1,a/b are in the order of s–1, while transverse relaxation rates R2,a/b = 1/T2,a/b are in the order of 10–100 s–1 for *in vivo* 1H systems. For macromolecules and bound water R2b can take values up to 106 s–1 (Stanisz et al., 2005). On resonance, the rf irradiation field B1 = (B1, 0, 0) with B1 ≈ μT, induces a precession of the magnetization with frequency ω1 = γ B1 around the x–axis in the rotating frame in the order of several 100 rad/s. In the case of off–resonant irradiation the magnetization rotates around the effective field = (ω1, 0, Δω)/γ = /γ which is tilted by the angle θ = tan-1(ω1/Δω) off the z-axis (Figure 5). The population fraction fb is assumed to be < 1 % for most MRZS experiments, hence ka is small compared to kb. In contrast, for MT fb can be as large as 19 % for cartilage, but is in the range of several percent for most tissues (Stanisz et al., 2005).

The Bloch–McConnell/–Solomon equations pick out one relaxation pathway explicitly. We therefore separate this process and define the relaxation rates excluding this specific pathway by R1’ = 1/T1’ and R2’ = 1/T2’ (Neuhaus and Williamson, 1989).

In the context of the BM equations the used relaxation rates already exclude exchange. Therefore, R1a = R1a’ and R2a = R2a’ and the same for pool b. The matrix Ka = fbKb reads

|  |  |
| --- | --- |
|  | (6) |

The proton fraction fb is

|  |  |
| --- | --- |
|  | (7) |

where [a] and [b] are the concentrations of pool a and b, respectively, and *na* and nb are the number of protons per molecule for the respective proton pools.

### Macromolecular semi-solid MT

For macromolecular MT one assumes that the transverse components are zero, which is justified since the T2 times are in the order of μs. Thus four magnetization components remain: those along the axes x, y, z for pool a and along z for the macromolecular pool . The matrix A (eq. (3)) and the vector C now read

|  |  |
| --- | --- |
|  | (11) |
|  | (12) |

The matrix contains a new term Rrfb which considers the lineshape of the macromolecular pool

|  |  |
| --- | --- |
| . | (13) |

Different line shape functions g() were employed (Henkelman et al., 2001, 1993): Lorentzian functions for liquid pools

|  |  |
| --- | --- |
|  | (14) |

or Gaussian for solid–like pools. For semi–solid structures in living tissue a super–Lorentzian lineshape defined by

|  |  |
| --- | --- |
|  | (15) |

was shown to fit the experimental data at best including an additional dipolar reservoir (Morrison and Mark Henkelman, 1995; Morrison et al., 1995).

If modeling focuses on CEST and the range close to the water peak, also a Lorentzian line can be fitted instead of a super–Lorentzian. This means that MT can be treated like a broad CEST effect (Liu et al., 2012; Zaiss et al., 2011).

For a detailed description of MT we recommend the review of Henkelman (Henkelman et al., 2001). Analytical solutions of the modified equations (11) are explored in Refs. (Adler and Yeung, 1993; Henkelman et al., 1993; Morrison et al., 1995; Yeung and Swanson, 1992; Yeung et al., 1994). Finally, the different two–pool models can be combined to a multi–pool system where all effects occur simultaneously (Desmond and Stanisz, 2012; Li et al., 2008; Liu et al., 2012; Trott and Palmer, 2004).

## Numerical solution of the Bloch–McConnell equations

The BM equations are first–order ordinary differential equations with an inhomogeneous term C (equation (1)). The formal solution of the BM equations is

|  |  |
| --- | --- |
|  | (16) |

The straightforward approach to solve a higher–dimensional system of ordinary differential equations is numerical integration, e.g. Runge–Kutta methods (Butcher, 1987; Murase and Tanki, 2011). When matrix inversion and exponentiation are performed numerically the magnetization can be calculated for constant A (Murase and Tanki, 2011; Tee et al., 2012; Woessner et al., 2005). Then stepwise constant solutions can be utilized to construct a full time–dependent solution of the BM equations (Sun et al., 2011; Zu et al., 2011b).

For n-pools the system of equations (1-3) just increases in the dimensionality. E.g. equation (39 transforms to



If we also allow intramolecular transfer between the individual CEST pools we just get more off-diagonal elements. For example, in the following matrix we allow a path from pool b to pool c:



### Shaped pulses

The model was modified for the case of pulsed saturation, which was achieved by discretization of the RF field according to the time course of the saturation pulse. In principle, this is also done at the MRI scanner by the analog sampling of the pulse. We use 200 steps to divide a shaped pulse. If the pulse has a high dynamic, this can lead to sidebands in simulated Z-spectra and thus discretization must be increased in the simulation.

### Pseudo Code

function Z = NUMERIC\_SIM(Sim)

%thermal Magnetization

y0 = [0 0 0 0 0 0 0 0 0 0 0 0 Sim.fA Sim.fB Sim.fC Sim.fD Sim.fE Sim.fF Sim.fG];

%parameters in x0

x0 = [ Sim.FREQ Sim.B1 Sim.MT Sim.n\_cest\_pool 0 0 0 0;

Sim.dwA 0 Sim.R1A Sim.R2A 0 0 0 0;

Sim.dwB Sim.kBA Sim.R1B Sim.R2B Sim.kBD Sim.kBE Sim.kBF Sim.kBG;

Sim.dwC Sim.kCA Sim.R1C Sim.R2C 0 0 0 0;

Sim.dwD Sim.kDA Sim.R1D Sim.R2D 0 Sim.kDE Sim.kDF Sim.kDG;

Sim.dwE Sim.kEA Sim.R1E Sim.R2E 0 0 Sim.kEF Sim.kEG;

Sim.dwF Sim.kFA Sim.R1F Sim.R2F 0 0 0 Sim.kFG;

Sim.dwG Sim.kGA Sim.R1G Sim.R2G 0 0 0 0

];

x0(1,2) = 0; % set B1 to zero

x00=x0; % copy parameter

%% MV is the magnetization vector at all offsets; initialization here

MV\_start = BMsolution(x0, xZspec, [0 1], z0, y0);

MV = MV\_start.\*Sim.Zi;

[w1 dphase theta B1q] = RF\_pulse(Sim, teile); % the pulse shape gets calculated

**% XXXXXXXXXXXXXXXXXXXXXX MAIN LOOP XXXXXXXXXXXXXXXXXXXXXX**

for dd=1:Sim.dummies

SIMULATE\_READOUT;

SIMULATE-RECOVERY\_AFTER\_READOUT

for jj=1:fix(Sin.n) % pulse train loop or LTM loop

% XXXXXXXXXXXXXXXXXXXXXX PAUSE / TRANSFER XXXXXXXXXXXXXXXXXXXXXX

% never start with a pause

if jj > 1

% relaxtion during pause

x0(1,2) = 0;

MV = BMsolution(x0, xZspec, [tpulse(numel(w1)) tpulse(numel(w1))+Sim.td], z0, y0, MV);

end;

% XXXXXXXXXXXXXXXXXX LABELING DURING PULSE XXXXXXXXXXXXXXXXXXXXXX

for ii = 1:numel(w1)-1 % discretization of the shaped pulse

% set correct B1 value in every pulse part/step

x0(1,2)=w1(ii);

[MV A\_temp Ainvb\_temp dia] = BMsolution(x0, xZspec, [tpulse(ii)

tpulse(ii+1)], z0, y0, MV);

end;

end

% XXXXXXXXXXXXXXXXXXXXXXXXXX SPOILING XXXXXXXXXXXXXXXXXXXXXXXXXXX

% relaxtion during spoiling

x0(1,2) = 0;

MV = BMsolution(x0, xZspec, [0 0.0045], z0, y0, MV);

MV(1:12,:) = Sim.spoilf\*MV(1:12,:); % spoiling

% hier ist nach einem LTM (label transfer module)

MV\_nth(jj,:,:) = MV;

end; % LTMs

end; % dummies

A.Sim = Sim;

A.x=xZspec';

A.zspec=squeeze(MV(13,:))';

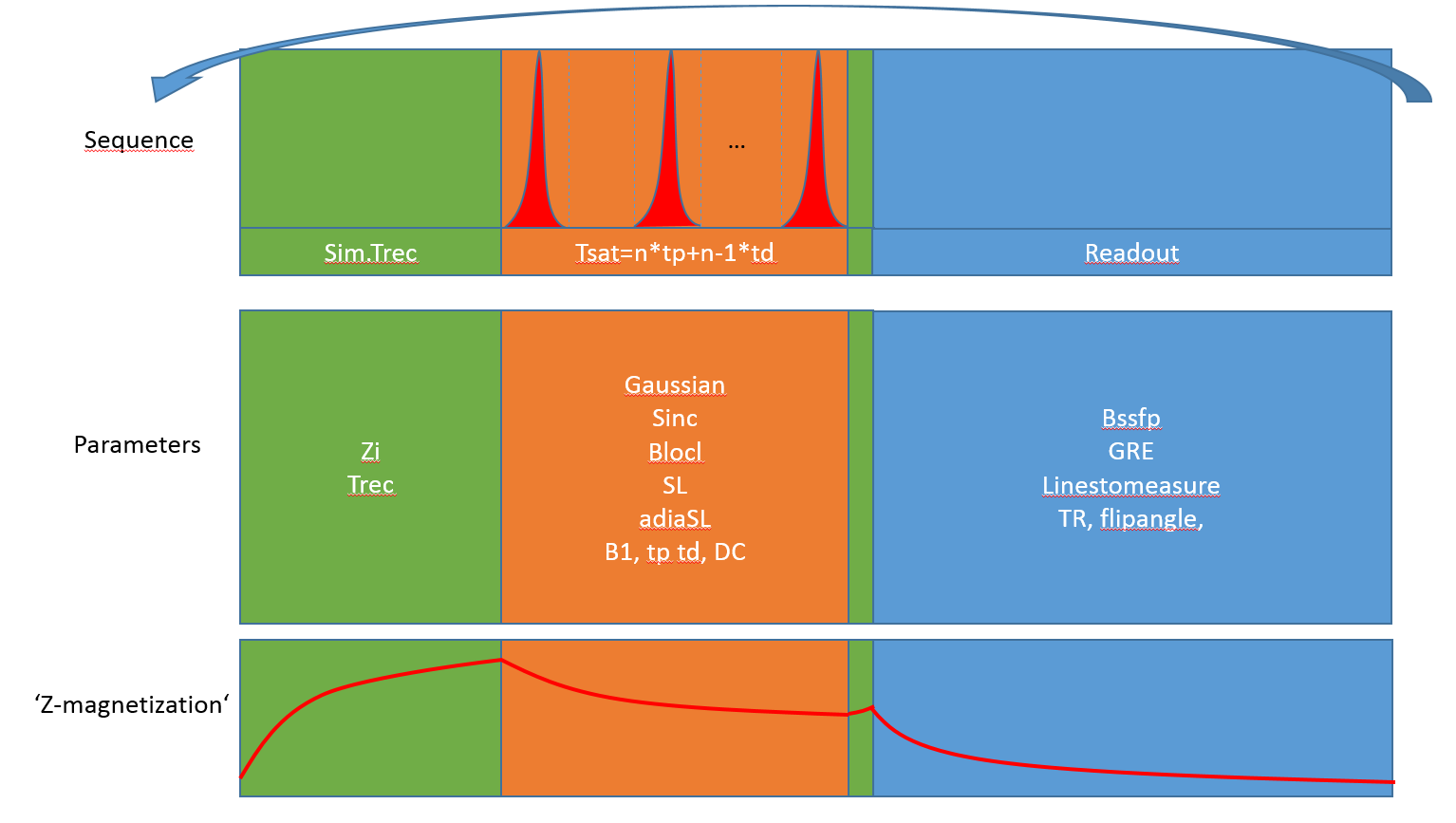


Figure 2: Visualization of Sequence, possible Parameters and dynamic of Z-magnetization