Supplementary Material for

Simulation of NMR spectra at zero- and ultra-low field from A to Z – a tribute to Prof. Konstantin L'vovich Ivanov

Quentin Stern[1], Kirill Sheberstov[2]

[1]Univ Lyon, CNRS, ENS Lyon, UCBL, Université de Lyon, CRMN UMR 5280, 69100 Villeurbanne, France
[2]Laboratoire des biomolécules, LBM, Département de chimie, École normale supérieure, PSL University, Sorbonne Université, CNRS, 75005 Paris, France

Correspondence to: Quentin Stern (quentin.stern@protonmail.com)

Contents

The codes presented here as PDF files can be downloaded from this link:
https://doi.org/10.5281/zenodo.7271319

A. Explicit calculation of the eigenbasis for A_2X spin system

Below we calculate explicitly the states of the eigenbasis for the A_2X spin system. An arbitrary spin state of the A_2X spin system can be represented as a linear combination of states obtained by direct products of the individual Zeeman states: $\mathcal{B}_Z^8 = \{\alpha,\beta\} \otimes \{\alpha,\beta\} \otimes \{\alpha,\beta\}$; here, the superscript 8 represents the dimensionality of the Hilbert space. The goal is to explicitly express the eigenbasis of the *J*-coupling Hamiltonian that governs ZULF evolution in terms of the Zeeman basis $\mathcal{B}_Z^8$. This is achieved by adding up angular momenta of individual spins taking into account Clebsch-Gordan coefficients to construct the state (see Eq. 46 of the main text). Figure 7 in the main text illustrates two stages of adding up angular momenta for the A_2X case: A + A and A_2 + X. There are four states with total spin F = 3/2 and four more states with total spin F = 1/2. The full specification of an eigenfunction at zero field requires three quantum numbers: the total spin of all three spins F, its projection m_F, and the total spin of some of the two spins. Observable transitions in ZULF conserve total spin of the A_2 subsystem (see Eq. 59 in the main text), so it makes sense to consider eigenfunctions that are characterised by the total spin of A_2 subsystem, F_{A2}. Therefore, three quantum numbers $|F, m, F_{A2}\rangle$ are used to specify all eight eigenstates.

Eq. 46 of the main text gives the expression of the eigenfunctions of the *J*-coupling Hamiltonian in terms of the Zeeman states and the Clebsch-Gordan coefficients for a pair of spins. Because we are considering three spins, we need to use Eq. 46 twice (for each addition operation) yielding the expression:

$$|F, m, F_{A2}\rangle \;=\; \sum_{m_I, m_S} C^{F_{A2}, m_{A2}}_{I_{a1}, m_{a1}; I_{a2}, m_{a2}} C^{F, m}_{F_{A2}, m_{A2}; S, m_S} |I_{a1}, m_{a1}, I_{a2}, m_{a2}\, S, m_S\,\rangle, \qquad \text{Eq S1}$$

Here, I_{a1} and m_{a1} are the total spin and its projection of the first spin A, respectively, likewise I_{a2} and m_{a2} are those for the second spin A, and S and m_S are the total spin and its projection for spin X, respectively. As an example, let us consider the $|3/2, 1/2, 1\rangle$ state. There are two nonzero Clebsch-Gordan coefficients $C^{F,m}_{F_{A2}, m_{A2}; S, m_S}$ coupling the two A spins:

$$C^{F=3/2, m=1/2}_{F_{A2}=1, m_{A2}=1; S=1/2, m_S=-1/2} = \left\langle 1, 1; \tfrac{1}{2}, -\tfrac{1}{2} \middle| \tfrac{3}{2}, \tfrac{1}{2} \right\rangle = \sqrt{\tfrac{1}{3}},$$

$$C^{F=3/2, m=1/2}_{F_{A2}=1, m_{A2}=0; S=1/2, m_S=1/2} = \left\langle 1, 0; \frac{1}{2}, \frac{1}{2} \middle| \frac{3}{2}, \frac{1}{2} \right\rangle = \sqrt{\frac{2}{3}} \qquad \text{Eq S2}$$

Each of them is multiplied by the corresponding Clebsch-Gordan coefficient coupling the A_2 pair with spin X:

$$C^{F_{A2}=1, m_{A2}=1}_{I_{a1}=1/2, m_{a1}=1/2; I_{a2}=1/2, m_{a2}=1/2} = \left\langle \tfrac{1}{2}, \tfrac{1}{2}; \tfrac{1}{2}, \tfrac{1}{2} \middle| 1,1 \right\rangle = 1,$$

$$C^{F_{A2}=1, m_{A2}=0}_{I_{a1}=1/2, m_{a1}=\pm 1/2; I_{a2}=1/2, m_{a2}=\mp 1/2} = \left\langle \frac{1}{2}, -\frac{1}{2}; \frac{1}{2}, \frac{1}{2} \middle| 1,0 \right\rangle = \left\langle \frac{1}{2}, \frac{1}{2}; \frac{1}{2}, -\frac{1}{2} \middle| 1,0 \right\rangle = \sqrt{\frac{1}{2}} \qquad \text{Eq S3}$$

So that the resulted eigenstate is:

$$|F = 3/2, m = 1/2, F_{A2} = 1\rangle \;=\; \sqrt{\tfrac{1}{3}} |\alpha\alpha\beta\rangle + \sqrt{\tfrac{2}{3}} \sqrt{\tfrac{1}{2}} (|\alpha\beta\alpha\rangle + |\beta\alpha\alpha\rangle). \qquad \text{Eq S4}$$

Analogously, one can calculate all the eigenstates that are shown in Table S1. The reader is encouraged to check this table. Wolfram Mathematica can be used to compute the value of the Clebsch-Gordan coefficients with the syntax $\mathrm{ClebschGordan}[\{\mathrm{Fa1}, \mathrm{ma1}\}, \{\mathrm{Fa2}, \mathrm{ma2}\}, \{\mathrm{FA2}, \mathrm{mA2}\}]$.

Finally, we note that the resulting eigenbasis is also the eigenbasis for the A_3 spin system. One may therefore use it to compute the eigenbasis of the *J*-coupling Hamiltonian for the XA_3 spin system using Eq. 46 one more time.

Table S1. Eigenstates of XA_2 spin system sorted according to $|F, m, F_{A2}\rangle$ quantum numbers, their explicit forms in terms of Zeeman basis and all the Clebsch-Gordan coefficients that are used to calculate the states. The Clebsch-Gordan coefficients are written using a braket notation with the correspondence $C^{F,m_F}_{I,m_I,S,m_S} = \langle I, S; m_I, m_S | F, m_F \rangle$

Total spin states $(|F, m, F_{A2}\rangle)$	**Explicit form**	**Clebsch-Gordan coefficients used to calculate the eigenstate**
$\|3/2, 3/2, 1\rangle$	$\|\alpha\alpha\alpha\rangle$	**A + A:** $\left\langle \frac{1}{2}, \frac{1}{2}; \frac{1}{2}, \frac{1}{2} \middle| 1,1 \right\rangle = 1$ **A_2 + X:** $\left\langle 1, 1; \frac{1}{2}, \frac{1}{2} \middle| \frac{3}{2}, \frac{3}{2} \right\rangle = 1$
$\|3/2, 1/2, 1\rangle$	$\sqrt{\frac{1}{3}}\|\alpha\alpha\beta\rangle + \sqrt{\frac{2}{3}}\sqrt{\frac{1}{2}}(\|\alpha\beta\alpha\rangle + \|\beta\alpha\alpha\rangle)$	**A + A:** $\left\langle \frac{1}{2}, \frac{1}{2}; \frac{1}{2}, \frac{1}{2} \middle| 1,1 \right\rangle = 1$ $\left\langle \frac{1}{2}, -\frac{1}{2}; \frac{1}{2}, \frac{1}{2} \middle| 1,0 \right\rangle = \left\langle \frac{1}{2}, \frac{1}{2}; \frac{1}{2}, -\frac{1}{2} \middle| 1,0 \right\rangle = = \sqrt{\frac{1}{2}}$ **A_2 + X:** $\left\langle 1, 1; \frac{1}{2}, -\frac{1}{2} \middle| \frac{3}{2}, \frac{1}{2} \right\rangle = \sqrt{\frac{1}{3}}$ $\left\langle 1, 0; \frac{1}{2}, \frac{1}{2} \middle| \frac{3}{2}, \frac{1}{2} \right\rangle = \sqrt{\frac{2}{3}}$
$\|3/2, -1/2, 1\rangle$	$\sqrt{\frac{1}{3}}\|\beta\beta\alpha\rangle + \sqrt{\frac{2}{3}}\sqrt{\frac{1}{2}}(\|\alpha\beta\beta\rangle + \|\beta\alpha\beta\rangle)$	**A + A:** $\left\langle \frac{1}{2}, -\frac{1}{2}; \frac{1}{2}, -\frac{1}{2} \middle| 1, -1 \right\rangle = 1$ $\left\langle \frac{1}{2}, -\frac{1}{2}; \frac{1}{2}, \frac{1}{2} \middle| 1,0 \right\rangle = \left\langle \frac{1}{2}, \frac{1}{2}; \frac{1}{2}, -\frac{1}{2} \middle| 1,0 \right\rangle = = \sqrt{\frac{1}{2}}$ **A_2 + X:** $\left\langle 1, -1; \frac{1}{2}, \frac{1}{2} \middle| \frac{3}{2}, -\frac{1}{2} \right\rangle = \sqrt{\frac{1}{3}}$ $\left\langle 1, 0; \frac{1}{2}, -\frac{1}{2} \middle| \frac{3}{2}, -\frac{1}{2} \right\rangle = \sqrt{\frac{2}{3}}$
$\|3/2, -3/2, 1\rangle$	$\|\beta\beta\beta\rangle$	**A + A:** $\left\langle \frac{1}{2}, -\frac{1}{2}; \frac{1}{2}, -\frac{1}{2} \middle| 1, -1 \right\rangle = 1$ **A_2 + X:** $\left\langle 1, -1; \frac{1}{2}, -\frac{1}{2} \middle| \frac{3}{2}, -\frac{3}{2} \right\rangle = 1$
$\|1/2, 1/2, 1\rangle$	$\sqrt{\frac{2}{3}}\|\alpha\alpha\beta\rangle - \sqrt{\frac{1}{3}}\sqrt{\frac{1}{2}}(\|\alpha\beta\alpha\rangle + \|\beta\alpha\alpha\rangle)$	**A + A:** $\left\langle \frac{1}{2}, \frac{1}{2}; \frac{1}{2}, \frac{1}{2} \middle| 1,1 \right\rangle = 1$ $\left\langle \frac{1}{2}, -\frac{1}{2}; \frac{1}{2}, \frac{1}{2} \middle| 1,0 \right\rangle = \left\langle \frac{1}{2}, \frac{1}{2}; \frac{1}{2}, -\frac{1}{2} \middle| 1,0 \right\rangle = = \sqrt{\frac{1}{2}}$

		A_2 + X: $\left\langle 1,1;\frac{1}{2},-\frac{1}{2}\middle\vert\frac{1}{2},\frac{1}{2}\right\rangle=\sqrt{\frac{2}{3}}$ $\left\langle 1,0;\frac{1}{2},\frac{1}{2}\middle\vert\frac{1}{2},\frac{1}{2}\right\rangle=-\sqrt{\frac{1}{3}}$
$\vert 1/2,-1/2,1\rangle$	$\sqrt{\frac{2}{3}}\vert\beta\beta\alpha\rangle-\sqrt{\frac{1}{3}}\sqrt{\frac{1}{2}}(\vert\alpha\beta\beta\rangle+\vert\beta\alpha\beta\rangle)$	**A + A:** $\left\langle\frac{1}{2},-\frac{1}{2};\frac{1}{2},-\frac{1}{2}\middle\vert 1,-1\right\rangle=1$ $\left\langle\frac{1}{2},-\frac{1}{2};\frac{1}{2},\frac{1}{2}\middle\vert 1,0\right\rangle==\left\langle\frac{1}{2},\frac{1}{2};\frac{1}{2},-\frac{1}{2}\middle\vert 1,0\right\rangle$ $=\sqrt{\frac{1}{2}}$ **A_2 + X:** $\left\langle 1,-1;\frac{1}{2},-\frac{1}{2}\middle\vert\frac{3}{2},-\frac{1}{2}\right\rangle=\sqrt{\frac{2}{3}}$ $\left\langle 1,0;\frac{1}{2},-\frac{1}{2}\middle\vert\frac{3}{2},-\frac{1}{2}\right\rangle=-\sqrt{\frac{1}{3}}$
$\vert 1/2,1/2,0\rangle$	$\sqrt{\frac{1}{2}}(\vert\alpha\beta\alpha\rangle-\vert\beta\alpha\alpha\rangle)$	**A + A:** $\left\langle\frac{1}{2},\frac{1}{2};\frac{1}{2},-\frac{1}{2}\middle\vert 0,0\right\rangle=\sqrt{\frac{1}{2}}$ $\left\langle\frac{1}{2},-\frac{1}{2};\frac{1}{2},\frac{1}{2}\middle\vert 0,0\right\rangle=-\sqrt{\frac{1}{2}}$ **A_2 + X:** $\left\langle 0,0;\frac{1}{2},\frac{1}{2}\middle\vert\frac{1}{2},\frac{1}{2}\right\rangle=1$
$\vert 1/2,-1/2,0\rangle$	$\sqrt{\frac{1}{2}}(\vert\alpha\beta\beta\rangle-\vert\beta\alpha\beta\rangle)$	**A + A:** $\left\langle\frac{1}{2},\frac{1}{2};\frac{1}{2},-\frac{1}{2}\middle\vert 0,0\right\rangle=-\sqrt{\frac{1}{2}}$ $\left\langle\frac{1}{2},-\frac{1}{2};\frac{1}{2},\frac{1}{2}\middle\vert 0,0\right\rangle=-\sqrt{\frac{1}{2}}$ **A_2 + X:** $\left\langle 0,0;\frac{1}{2},-\frac{1}{2}\middle\vert\frac{1}{2},-\frac{1}{2}\right\rangle=1$

Tests on ideal signals to show the efficiency of the Fourier transform algorithm

This document aims at showing that the Fourier transform is accurately computed. It shows:

- That the frequency axis is accurately computed regardless of the number of points in the FID
- That a window function must be applied to the FID to avoid a baseline shift in the Fourier transform and consequent bias in the signal integration

Effect of zero filling and verification of the frequency axis

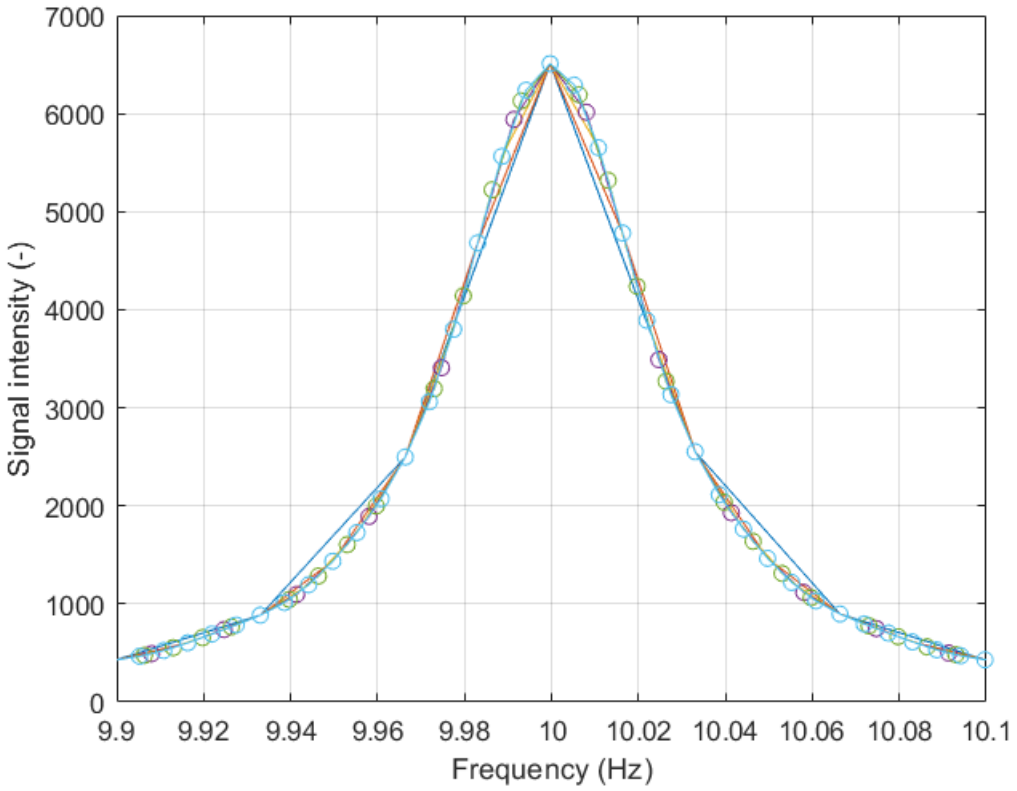
This section first computes the time domain signal *St* of an oscillating signal with frequency *nu0* and a decoherence time constant *T2*. The signal is sampled with *N* points from 0 to *tmax*.

```
nu0=10;
T2=6; tmax=30;
x=0.1;
N=2^15;
t=linspace(0,tmax,N)'; St=zeros(N,1);
St=St+cos(t*2*pi*nu0).*exp(-t/T2);
```

The signal is Fourier transformed using function *FFT* (defined at the bottom of the code), with different values of zero filling *N0*. The real parts of the resulting curves are ploted on the same graph.

The intensity of the signal decreases linearly with the number of points in the Fourier transform (here, *N0*). So that all curves superimpose, each curve is multiplied by the number of points of the Fourier transform *N0*.

```
figure
for i=1:6
    N0=N*i;
    [nu,Snu_1]=FFT(t,St,N0);
    plot(nu,real(Snu_1)*N0,'o-'), hold on
end
hold off
ylabel('Signal intensity (-)')
xlabel('Frequency (Hz)')
xlim(nu0+[-x +x]), grid on
```



The plot shows that all the curves superimpose perfectly and that the frequency of the signal is precisely *nu0*, as expected. The user may play with the frequency *nu0* and choosing arbitrarily long *T2* values, the signal will always appear at *nu0* (provided *tmax* is chosen to be at least 5*T2).

This shows that the way the computation of the Fourier transform and of the frequency axis are precise.

Verification of the consistency of the integration

This first section computes the time domain signal *St* of oscillating signals with a frequency list *nu0* with a decoherence time constant *T2*. The signal is sampled with *N* points from 0 to *tmax*.

```
nu0=[5 25 50];
T2=6; tmax=80;
N=2^15; N0=N*16;
t=linspace(0,tmax,N)'; St=zeros(N,1);
for i=1:length(nu0)
    St=St+cos(t*2*pi*nu0(i)).*exp(-t/T2);
end
```

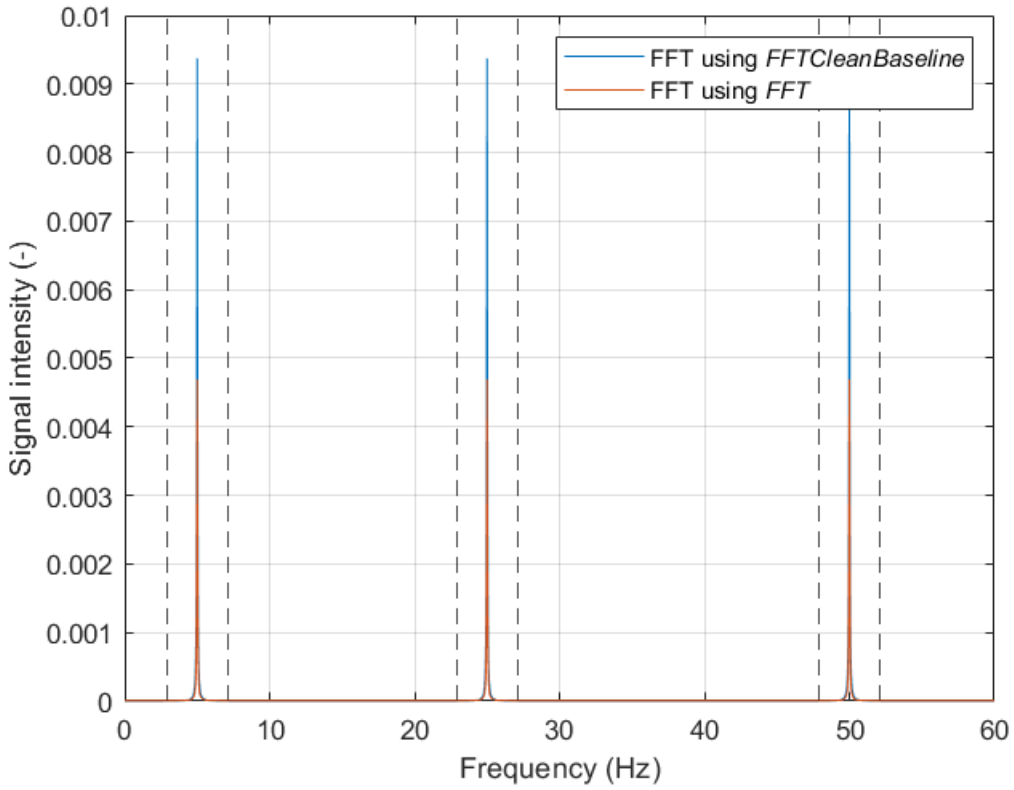
The signal is Fourier transformed using function *FFT* and *FFT_CleanBaseline* (defined at the bottom of the code) with *N0* points.

```
[nu,Snu_1]=FFT_CleanBaseline(t,St,N0);
[nu,Snu_2]=FFT(t,St,N0);
Sreal_1=real(Snu_1);
```

```
Sreal_2=real(Snu_2);
```

The real part of the Fourier transform is plotted. The veritical dashed lines show the area of integration of each signal which will be used later (going from -W*FWHM to +W*FWHM, FWHM=Full width at half maximum).

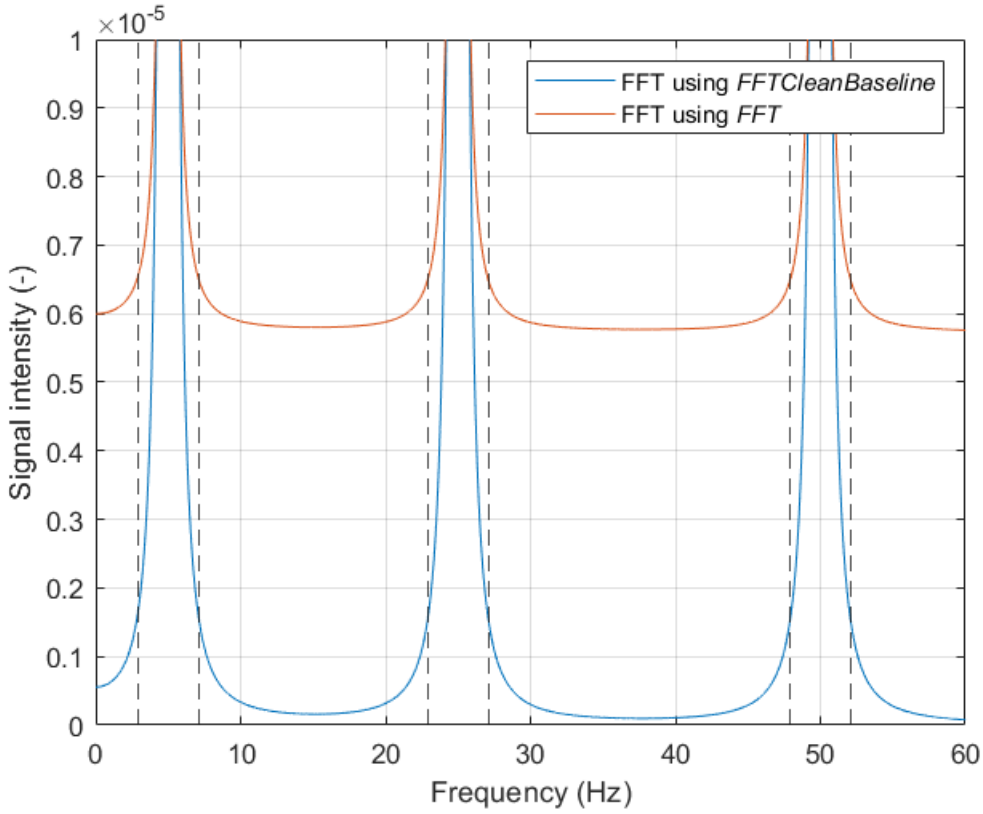
```
W=40;
figure
plot(nu,Sreal_1), hold on
plot(nu,Sreal_2), hold off
grid on
xlim([0 60])
ylabel('Signal intensity (-)'), xlabel('Frequency (Hz)')
for i=1:length(nu0)
    xline(nu0(i)-W/pi/T2,'k--')
    xline(nu0(i)+W/pi/T2,'k--')
end
legend({'FFT using {\itFFTCleanBaseline}','FFT using {\itFFT}'})
```



*FFT_CleanBaseline yield*s a signal twice as intense as *FFT*. However, without zooming at the foot of the signals, there are not reason to think that one algorithm is better than the other one.

```
figure
plot(nu,Sreal_1), hold on
plot(nu,Sreal_2), hold off
grid on
xlim([0 60]), ylim([0 1e-5])
ylabel('Signal intensity (-)'), xlabel('Frequency (Hz)')
```

```
for i=1:length(nu0)
    xline(nu0(i)-W/pi/T2,'k--')
    xline(nu0(i)+W/pi/T2,'k--')
end
legend({'FFT using {\itFFTCleanBaseline}','FFT using {\itFFT}'})
```



The zoom above reveals that the Fourier transform computed using *FFT* has a baseline shift while that computed using *FFT_CleanBaseline* appropriately tends towards 0 away form the spectral components.

This difference is reflected in the signal integrals that are computed using both Fourier transform algorithms.

Because the time domain signal is the sum of three cosine waves with unit amplitude, the first point of hte FID is equal to 3:

```
St(1)
```

```
ans = 3
```

For both *FFT* and *FFT_CleanBaseline*, the total sum of the spectrum is equal to the first point of the FID.

In the case of *FFT_CleanBaseline*:

```
sum(Sreal_1)
```

```
ans = 3.0000
```

while in the case of FFT:

```
sum(Sreal_2)
```

```
ans = 3.0000
```

When the individual resonances of the frequency domain signal computed with *FFT_CleanBaseline* are integrated, they tend towards 1, as they should:

```
SumOfTheIntegrals=0;
for i=1:length(nu0)
    Integral=sum(Sreal_1(nu>nu0(i)-W/pi/T2 & nu<nu0(i)+W/pi/T2));
    disp(['Integral of signal ' num2str(i) ': ' num2str(Integral)])
    SumOfTheIntegrals=SumOfTheIntegrals+Integral;
end
```

```
Integral of signal 1: 0.99258
Integral of signal 2: 0.99226
Integral of signal 3: 0.99214
```

and the sum of the three integrals tends towards 3:

```
SumOfTheIntegrals
```

```
SumOfTheIntegrals = 2.9770
```

To the contrary, when the individual resonances of the frequency domain signal computed with *FFT* are integrated, they tend towards significantly lower values than 1, because part of the integral moved into the baseline:

```
SumOfTheIntegrals=0;
for i=1:length(nu0)
    Integral=sum(Sreal_2(nu>nu0(i)-W/pi/T2 & nu<nu0(i)+W/pi/T2));
    disp(['Integral of signal ' num2str(i) ': ' num2str(Integral)])
    SumOfTheIntegrals=SumOfTheIntegrals+Integral;
end
```

```
Integral of signal 1: 0.52738
Integral of signal 2: 0.52722
Integral of signal 3: 0.52716
```

And the sum of the three integrals is barely higher than half what it should be:

```
SumOfTheIntegrals
```

```
SumOfTheIntegrals = 1.5818
```

```
function [nu,S]=FFT_CleanBaseline(t,I,N)
    % Converts to column vector
    I=I(:);

    % Window function to avoid baseline shift
    I=[I(1); 2*I(2:end)];

    % Sample length (s)
    T = (max(t)-min(t))/(length(I)-1);
```

```
    % Frequency axis (Hz)
    nu = (0:N/2-1)/T/N;

    % Fourier transform
    S = fft(I,N)/N;

    % Remove negative frequencies
    S=2*S(1:N/2);
end
function [nu,S]=FFT(t,I,N)
    % Converts to column vector
    I=I(:);

    % Sample length (s)
    T = (max(t)-min(t))/(length(I)-1);

    % Frequency axis (Hz)
    nu = (0:N/2-1)/T/N;

    % Fourier transform
    S = fft(I,N)/N;

    % Remove negative frequencies
    S=2*S(1:N/2);
end
```

Simulation of NMR spectra at zero and ultra-low field - Case of XA spin system

Figure path and options

```
clear all
FigurePath='C:\Users\quent\Documents\Projects\Tutorial paper on ZULF\Figures\';
FigureWidth=11;             % Figure width in cm
FigureHeight=3;             % Figure height in cm
FigureWidth2=12;            % Nutation figure width in cm
FigureHeight2=3;            % Nutation figure height in cm
c1=[0    0.4470    0.7410]; % Figure color 1
c2=[1 1 1]*0.5;             % Figure color 2
lw=1;                       % Figure linewidth
YlimSpectra=[0 0.4];        % Figure spectra y-limits in pT
YlimFID=[-60 60];           % Figure FID y-limits in pT
Xlim=[0 160];               % Figure spectra x-limits
```

General paramaters and operators

Spin system paramaters

```
JIS=140;            % J-coupling between I and S in Hz
gI= 67.262e6;       % Gyromagnetic ration of spin I (13C) in rad.s-1.T-1
gS=276.513e6;       % Gyromagnetic ration of spin S (1H) in rad.s-1.T-1
```

Experimental conditions

```
Bpol=2;             % Field for prepolarization in T
T=298;              % Temperature of prepolarization in K
Bx=0.5e-6;          % Magnetic field along x for ultra-low field experiments in T
Bpulse=50e-6;       % Magnetic field along x during DC pulses in T
NA=6.02e23;         % Avogadro's number in mol-1
C=27;               % Molecule concentration in mol.L-1
V=100e-6;           % Sample volume in L
r=0.01;             % Distance between the center of the sample and the center of the detector
```

Physical constants

```
hbar=1.05457e-34;   % Reduced Plank constant in J.s/rad
kB=1.3806485e-23;   % Boltzmann constant J.K-1
mu0=4*pi*1e-7;      % Permeability of free space dived in T.m.A-1
```

Parameters for propagation

```
taq=5;              % Propagation/acquisition time in s
K=2^12;             % Number of points in the FID
L=2^16;             % Number of points in the FID with zero filling
T2=1;               % Coherence time constant in s
dt=taq/K;           % Time interval for propagation in s
t=(0:K-1)'/K*taq;   % Time axis in s
```

Frequency axis

```
f=(K-1)/taq;        % Sampling frequency in Hz
nu=(0:L/2-1)/L*f;   % Frequency axis in Hz (only positive frequencies)
```

Integration range

```
numin=138;          % Lower boundary for signal integration in Hz
numax=142;          % Higher boundary for signal integration in Hz
```

Definition of operators

Pauli matrices

```
e =[1 0;  0 1];
sx=[0 1;  1 0];
sy=[0 -1i;1i 0];
sz=[1 0;  0 -1];
```

Operators for spin I

```
Ix=kron(sx/2,e);
Iy=kron(sy/2,e);
Iz=kron(sz/2,e);
```

Operators for spin S

```
Sx=kron(e,sx/2);
Sy=kron(e,sy/2);
Sz=kron(e,sz/2);
```

Unity operator

```
E=kron(e,e);
```

Initial density matrix

Polarization of spin I at equilibrium in the prepolarizing magnet calculated using Boltzman distribution

```
PI=tanh(hbar*Bpol*gI/kB/T/2);
```

Polarization of spin S at equilibrium in the prepolarizing magnet calculated using Boltzman distribution

```
PS=tanh(hbar*Bpol*gS/kB/T/2);
```

Initial density matrix of spin I and S at equilibrium in the prepolarizing magnet

```
Iz_2by2=sz/2;
rhoI=e/2+PI*Iz_2by2;
rhoS=e/2+PS*Iz_2by2;
rho_eq=kron(rhoI,rhoS);
```

Modification of the density matrix removing the identity

```
NumberOfSpins=2;
rho_eq=rho_eq-eye(2^NumberOfSpins)/2^NumberOfSpins;
```

Hamiltonian and propagators

Zeeman Hamiltonian as a function of field along x, y and z

```
HZ=@(Bx, By, Bz) -gI*(Bx*Ix+By*Iy+Bz*Iz) -gS*(Bx*Sx+By*Sy+Bz*Sz);
```

J Hamiltonian

```
HJ=2*pi*JIS*(Ix*Sx+Iy*Sy+Iz*Sz);
```

Total Hamiltonian

```
H=@(Bx, By, Bz) HJ+HZ(Bx, By, Bz);
```

Propagators at zero and ultra-low field

```
U_ZF =expm(-1i*dt*H(0,0,0));
U_ULF=expm(-1i*dt*H(Bx,0,0));
```

Observable operator: total field along z in pT at distance r from the sample (in the point dipole approximation)

```
N=NA*C*V;
O=(gI*Iz+gS*Sz)*N*hbar*1e12*mu0/2/pi/r^3;
```

Sudden drop experiments

Zero field case

```
% Vector for the time domain signal
TDS_sudden_ZF=zeros(K,1);

% Initialization of the density matrix
rho=rho_eq;
for k=1:K
    % Computes the expectation value of the field along z
    TDS_sudden_ZF(k)=real(trace(O*rho));

    % Propagates during dt
    rho=U_ZF*rho*U_ZF';
end

% Apodization - line broadening
TDS_sudden_ZF=TDS_sudden_ZF.*exp(-t/T2);

% Fourier transform
FDS_sudden_ZF = FourierTransform(TDS_sudden_ZF,L);
```

```
% Intensity of the sudden drop signal
I_sudden_drop=sum(FDS_sudden_ZF(nu>numin & nu<numax))
```

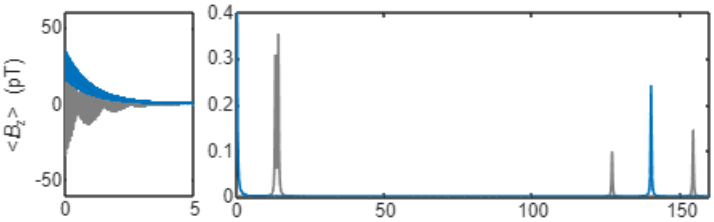
```
I_sudden_drop = 9.1326
```

Ultra-low field case

```
% Vector for the time domain signal
TDS_sudden_ULF=zeros(K,1);

% Initialization of the density matrix
rho=rho_eq;
for k=1:K
    % Computes the expectation value of the field along z
    TDS_sudden_ULF(k)=real(trace(O*rho));

    % Propagates during dt
    rho=U_ULF*rho*U_ULF';
end

% Relaxation
TDS_sudden_ULF=TDS_sudden_ULF.*exp(-t/T2);

% Fourier transform
FDS_sudden_ULF = FourierTransform(TDS_sudden_ULF,L);
```

Plot the results

```
figure("Units","centimeters","Position",[0 0 FigureWidth FigureHeight])
subplot(1,4,1)
plot(t,TDS_sudden_ULF,'LineWidth',lw,'color',c2), hold on
plot(t,TDS_sudden_ZF,'LineWidth',lw,'color',c1), hold off
xlim([0 taq]), %xlabel('Time (s)')
ylabel('<{\itB}_z> (pT)')
ylim(YlimFID)
set(gca,'fontsize',7)

subplot(1,4,[2 4])
plot(nu,FDS_sudden_ULF,'LineWidth',lw,'color',c2), hold on
plot(nu,FDS_sudden_ZF,'LineWidth',lw,'color',c1), hold off
xlim(Xlim), %xlabel('Frequency (Hz)')
set(gca,'fontsize',7)
ylim(YlimSpectra)

exportgraphics(gcf,fullfile(FigurePath, 'XA_system_sudden_drop.pdf'),'ContentType','vector')
```



Adiabatic field drop

Propagation of the adiabatic field drop

```
B_start=200e-6; % Magnetic field at the start of the decay in T
t_adia=0.5;     % Time of the adiabatic field drop in s
T_adia=0.05;    % Decay time constant of the adiabatic field drop in s
M=5000;         % Number of discrete points for the adiabatic field drop
```

Time intervals for adiabatic field drop

```
dt_adia=t_adia/M;
```

Field profile of the adiabatic field drop: exponential decay generated by a Helmholtz coil.

```
B_adia=B_start*(exp(-(0:M-1)/M*t_adia/T_adia)-exp(-t_adia/T_adia))/(1-exp(-t_adia/T_adia));
```

Propagation

```
rho_adia=rho_eq;
for k=1:M
    U=expm(-1i*dt_adia*H(0,0,B_adia(k)));
    rho_adia=U*rho_adia*U';
end
```

Zero field case

```
% Vector for the time domain signal
TDS_AFD_ZF=zeros(K,1);

% Initialization of the density mat
rho=rho_adia;
for k=1:K
    % Computes the expectation value of the field along z
    TDS_AFD_ZF(k)=real(trace(O*rho));

    % Propagates during dt
    rho=U_ZF*rho*U_ZF';
end

% Apodization - line broadening
TDS_AFD_ZF=TDS_AFD_ZF.*exp(-t/T2);
```

```
% Fourier transform
FDS_AFD_ZF = FourierTransform(TDS_AFD_ZF,L);
```

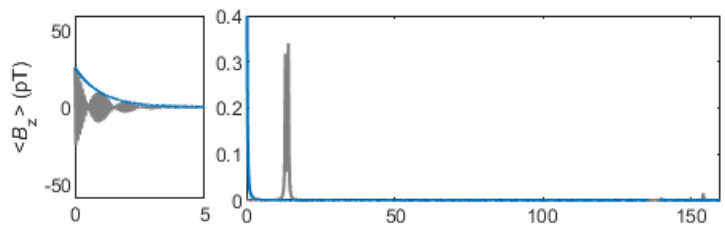
Ultra-low field case

Propagation

```
% Vector for the time domain signal
TDS_AFD_ULF=zeros(K,1);

% Initialization of the density mat
rho=rho_adia;
for k=1:K
    % Computes the expectation value of the field along z
    TDS_AFD_ULF(k)=real(trace(O*rho));

    % Propagates during dt
    rho=U_ULF*rho*U_ULF';
end

% Relaxation
TDS_AFD_ULF=TDS_AFD_ULF.*exp(-t/T2);

% Fourier transform
FDS_AFD_ULF = FourierTransform(TDS_AFD_ULF,L);
```

Plot the results

```
figure("Units","centimeters","Position",[0 0 FigureWidth FigureHeight])
subplot(1,4,1)
plot(t,TDS_AFD_ULF,'LineWidth',lw,'color',c2), hold on
plot(t,TDS_AFD_ZF,'LineWidth',lw,'color',c1), hold off
xlim([0 taq]), %xlabel('Time (s)')
ylim(YlimFID), ylabel('<{\itB}_z> (pT)')
set(gca,'fontsize',7)

subplot(1,4,[2 4])
plot(nu,FDS_AFD_ULF,'LineWidth',lw,'color',c2), hold on
plot(nu,FDS_AFD_ZF,'LineWidth',lw,'color',c1), hold off
xlim(Xlim), %xlabel('Frequency (Hz)')
set(gca,'fontsize',7)
ylim(YlimSpectra)

exportgraphics(gcf,fullfile(FigurePath, 'XA_system_adiabatic.pdf'),'ContentType','vector')
```



Propgation of the pulse along Z

Pulse length in s

```
tpulse=150e-6;
```

Pulse propagator and density operator after the pulse

```
Upulse=expm(-1i*tpulse*HZ(0,0,Bpulse));
rho_afterpuse=Upulse*rho_adia*Upulse';
```

Zero field case

```
% Vector for the time domain signal
TDS_Z_pulse_ZF=zeros(K,1);

% Initialization of the density mat
rho=rho_afterpuse;
for k=1:K
    % Computes the expectation value of the field along z
    TDS_Z_pulse_ZF(k)=real(trace(O*rho));

    % Propagates during dt
    rho=U_ZF*rho*U_ZF';
end

% Apodization - line broadening
TDS_Z_pulse_ZF=TDS_Z_pulse_ZF.*exp(-t/T2);

% Fourier transform
phase=pi/2;
FDS_Z_pulse_ZF = FourierTransform(TDS_Z_pulse_ZF*exp(1i*phase),L);
```

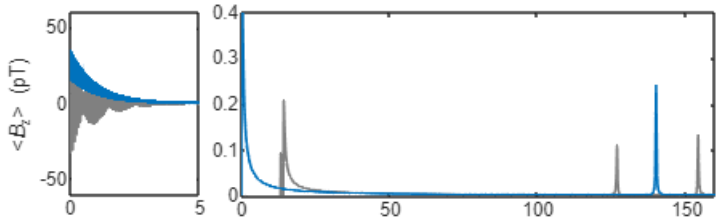
Ultra-low field case

```
% Vector for the time domain signal
TDS_Z_pulse_ULF=zeros(K,1);

% Initialization of the density mat
rho=rho_afterpuse;
for k=1:K
    % Computes the expectation value of the field along z
    TDS_Z_pulse_ULF(k)=real(trace(O*rho));

    % Propagates during dt
    rho=U_ULF*rho*U_ULF';
end

% Relaxation
TDS_Z_pulse_ULF=TDS_Z_pulse_ULF.*exp(-t/T2);
```

```
% Fourier transform
phase=pi/2;
FDS_Z_pulse_ULF = FourierTransform(TDS_Z_pulse_ULF*exp(1i*phase),L);
```

Plot the results

```
figure("Units","centimeters","Position",[0 0 FigureWidth FigureHeight])
subplot(1,4,1)
plot(t,TDS_Z_pulse_ULF,'LineWidth',lw,'Color',c2), hold on
plot(t,TDS_Z_pulse_ZF,'LineWidth',lw,'Color',c1), hold off
xlim([0 taq]), %xlabel('Time (s)')
ylim(YlimFID), ylabel('<{\itB}_z> (pT)')
set(gca,'fontsize',7)

subplot(1,4,[2 4])
plot(nu,FDS_Z_pulse_ULF,'LineWidth',lw,'Color',c2), hold on
plot(nu,FDS_Z_pulse_ZF,'LineWidth',lw,'Color',c1), hold off
xlim(Xlim), %xlabel('Frequency (Hz)')
set(gca,'fontsize',7)
ylim(YlimSpectra)

exportgraphics(gcf,fullfile(FigurePath, 'XA_system_adiabatic_and_pulse_Z.pdf'),'ContentType','v
```



Pulse along X

Pulse length in s

```
tpulse=910e-6;
```

Pulse propagator and density operator after the pulse

```
Upulse=expm(-1i*tpulse*HZ(Bpulse,0,0));
rho_afterpuse=Upulse*rho_adia*Upulse';
```

Zero field case

```
% Vector for the time domain signal
TDS_X_pulse_ZF=zeros(K,1);

% Initialization of the density mat
rho=rho_afterpuse;
for k=1:K
    % Computes the expectation value of the field along z
    TDS_X_pulse_ZF(k)=real(trace(O*rho));
```

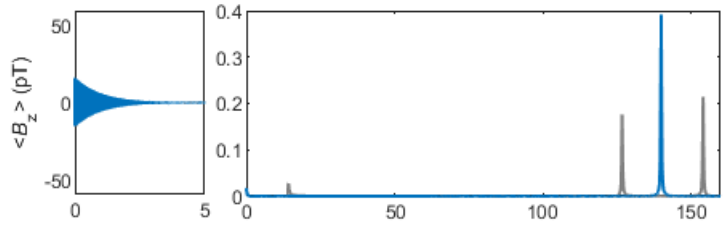
```
    % Propagates during dt
    rho=U_ZF*rho*U_ZF';
end

% Apodization - line broadening
TDS_X_pulse_ZF=TDS_X_pulse_ZF.*exp(-t/T2);

% Fourier transform
FDS_X_pulse_ZF = FourierTransform(TDS_X_pulse_ZF,L);
```

Ultra-low field case

```
% Vector for the time domain signal
TDS_X_pulse_ULF=zeros(K,1);

% Initialization of the density mat
rho=rho_afterpuse;
for k=1:K
    % Computes the expectation value of the field along z
    TDS_X_pulse_ULF(k)=real(trace(O*rho));

    % Propagates during dt
    rho=U_ULF*rho*U_ULF';
end

% Relaxation
TDS_X_pulse_ULF=TDS_X_pulse_ULF.*exp(-t/T2);

% Fourier transform
FDS_X_pulse_ULF = FourierTransform(TDS_X_pulse_ULF,L);
```

Plot the results

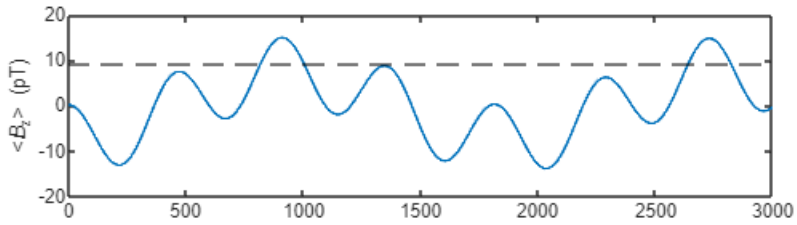
```
figure("Units","centimeters","Position",[0 0 FigureWidth FigureHeight])
subplot(1,4,1)
plot(t,TDS_X_pulse_ULF,'LineWidth',lw,'color',c2), hold on
plot(t,TDS_X_pulse_ZF,'LineWidth',lw,'color',c1), hold off
xlim([0 taq]), %xlabel('Time (s)')
ylim(YlimFID), ylabel('<{\itB}_z> (pT)')
set(gca,'fontsize',7)

subplot(1,4,[2 4])
plot(nu,FDS_X_pulse_ULF,'LineWidth',lw,'color',c2), hold on
plot(nu,FDS_X_pulse_ZF,'LineWidth',lw,'color',c1), hold off
xlim(Xlim), %xlabel('Frequency (Hz)')
set(gca,'fontsize',7)
ylim(YlimSpectra)

exportgraphics(gcf,fullfile(FigurePath, 'XA_system_adiabatic_and_pulse_X.pdf'),'ContentType','v
```



Nutation curve with pulse along x

```
% Vector containing the pulse lengths in s
tpulse=linspace(0,3000e-6,3000);

% Vector for signal as a function of the pulse length
I_X_pulse_nutation=zeros(length(tpulse),1);

parfor i=1:length(tpulse)
    % Vector for the time domain signal
    TDS_X_pulse_nutation=zeros(K,1);

    % Propagator of the pulse Hamiltonian
    Upulse=expm(-1i*tpulse(i)*HZ(Bpulse,0,0));

    % Initialization of the density mat
    rho=Upulse*rho_adia*Upulse';
    for k=1:K
        % Computes the expectation value of the field along z
        TDS_X_pulse_nutation(k)=real(trace(O*rho));

        % Propagates during dt
        rho=U_ZF*rho*U_ZF';
    end

    % Apodization - line broadening
    TDS_X_pulse_nutation=TDS_X_pulse_nutation.*exp(-t/T2);

    % Fourier transform
    FDS_X_pulse_nutation=FourierTransform(TDS_X_pulse_nutation,L);

    % Chops the relevant portion of the spectrum
    S_=FDS_X_pulse_nutation(nu>numin & nu<numax);

    % Integrates the portion of the spectrum
    I_X_pulse_nutation(i)=sum(S_);
end
figure("Units","centimeters","Position",[0 0 FigureWidth2 FigureHeight2])
plot(tpulse*1e6,I_X_pulse_nutation,'LineWidth',lw)
yline(I_sudden_drop,'k--')
ylabel('<{\itB}_z> (pT)'), %xlabel('Pulse length (\mus)')
ylim([-20 20])
```

```
set(gca,'fontsize',7)

exportgraphics(gcf,fullfile(FigurePath, 'XA_ZULF_nutation_curve_X.pdf'),'ContentType','vector')
```



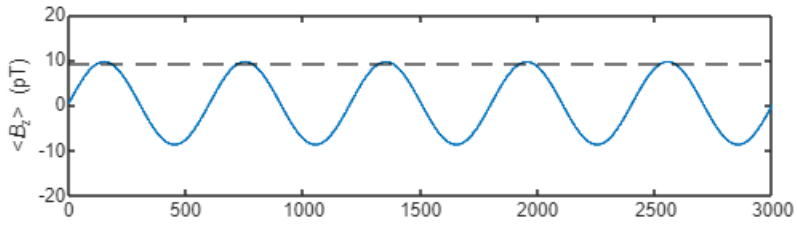

```
disp(['Signal ratio with respect to sudden drop: ' num2str(max(I_X_pulse_nutation)/I_sudden_dro
```

```
Signal ratio with respect to sudden drop: 1.64
```

Nutation curve with pulse along z

```
% Vector containing the pulse lengths in s
tpulse=linspace(0,3000e-6,3000);

% Vector for signal as a function of the pulse length
I_Z_pulse_nutation=zeros(length(tpulse),1);

parfor i=1:length(tpulse)
    % Vector for the time domain signal
    TDS_Z_pulse_nutation=zeros(K,1);

    % Propagator of the pulse Hamiltonian
    Upulse=expm(-1i*tpulse(i)*HZ(0,0,Bpulse));

    % Initialization of the density mat
    rho=Upulse*rho_adia*Upulse';
    for k=1:K
        % Computes the expectation value of the field along z
        TDS_Z_pulse_nutation(k)=real(trace(O*rho));

        % Propagates during dt
        rho=U_ZF*rho*U_ZF';
    end

    % Apodization - line broadening
    TDS_Z_pulse_nutation=TDS_Z_pulse_nutation.*exp(-t/T2);

    % Fourier transform
    phase=pi/2;
    FDS_Z_pulse_nutation=FourierTransform(TDS_Z_pulse_nutation*exp(1i*phase),L);

    % Chops the relevant portion of the spectrum
    S_=FDS_Z_pulse_nutation(nu>numin & nu<numax);
```

```
    % Integrates the portion of the spectrum
    I_Z_pulse_nutation(i)=sum(S_);
end

figure("Units","centimeters","Position",[0 0 FigureWidth2 FigureHeight2])
plot(tpulse*1e6,I_Z_pulse_nutation,'LineWidth',lw)
yline(I_sudden_drop,'k--')
ylabel('<{\itB}_z> (pT)')
%xlabel('Pulse length (\mus)')
ylim([-20 20])
set(gca,'fontsize',7)

exportgraphics(gcf,fullfile(FigurePath, 'XA_ZULF_nutation_curve_Z.pdf'),'ContentType','vector')
```



Functions

```
function S=FourierTransform(I,N)
    % Converts to column vector
    I=I(:);

    % Window function to avoid baseline shift
    I=[I(1); 2*I(2:end)];

    % Fourier transform
    S = real(fft(I,N))/N;

    % Remove negative frequencies
    S=2*S(1:N/2);
end
```

Simulation of NMR spectra at zero and ultra-low field - Case of XAn spin system

Figure path

```
clear all
FigurePath='C:\Users\quent\Documents\Projects\Tutorial paper on ZULF\Figures\';
c1=[0    0.4470    0.7410];
c2=[1 1 1]*0.5;
lw=1;
```

General paramaters and operators

Spin system paramaters

```
JIS=140;            % J-coupling between I and S in Hz
JSS=10;             % J-coupling between S spins in Hz
gI= 67.262e6;       % Gyromagnetic ration of spin I (13C) in rad.s-1.T-1
gS=276.513e6;       % Gyromagnetic ration of spin S (1H) in rad.s-1.T-1
```

Experimental conditions

```
Bpol=2;             % Field for prepolarization in T
T=298;              % Temperature of prepolarization in K
Bx=0.5e-6;          % Magnetic field along x for ultra-low field experiments in T
Bpulse=50e-6;       % Magnetic field along x during DC pulses in T
NA=6.02e23;         % Avogadro's number in mol-1
C=27;               % Molecule concentration in mol.L-1
V=100e-6;           % Sample volume in L
r=0.01;             % Distance between the center of the sample and the center of the detector
```

Physical constants

```
hbar=1.05457e-34;   % Reduced Plank constant in J.s
kB=1.3806485e-23;   % Boltzmann constant J.K-1
mu0=4*pi*1e-7;      % Permeability of free space dived in T.m.A-1
```

Parameters for propagation

```
taq=5;              % Propagation/acquisition time in s
K=2^13;             % Number of points in the FID
L=2^16;             % Number of points in the FID with zero filling
T2=1;               % Coherence time constant in s
dt=taq/K;           % Time interval for propagation in s
t=(0:K-1)'/K*taq;   % Time axis in s
```

Frequency axis

```
f=(K-1)/taq;        % Sampling frequency in Hz
nu=(0:L/2-1)/L*f;   % Frequency axis in Hz
```

Initial polarization

Polarization of spin I at equilibrium in the prepolarizing magnet calculated using Boltzman distribution

```
PI=tanh(hbar*Bpol*gI/kB/T/2);
```

Polarization of spin S at equilibrium in the prepolarizing magnet calculated using Boltzman distribution

```
PS=tanh(hbar*Bpol*gS/kB/T/2);
```

FID and spectra structure

Maximum number of X spins

```
N=5;
```

Cells to stores the time domain signals

```
TDS_ZF=cell(N,1);
TDS_ULF=cell(N,1);
```

Cells to store the frequency domain signals

```
FDS_ZF=cell(N,1);
FDS_ULF=cell(N,1);
```

Basis operators in single-spin Hilbert space

Pauli matrices

```
e =[1 0;  0 1];
sx=[0 1;  1 0];
sy=[0 -1i;1i 0];
sz=[1 0;  0 -1];
```

Angular momentum operators in single spin space

```
Ix_=sx/2;
Iy_=sy/2;
Iz_=sz/2;
E=eye(2);
```

Number of molecules in the sample

```
N_molecules=NA*C*V;
```

Loops over the number of X spins

```
parfor n=1:N
    % Total number of spins of the AXn spins system
    m=1+n;
```

```
% Constructs the operators for m-dimensional Hilbert space
Ix=cell(m,1);
Iy=cell(m,1);
Iz=cell(m,1);
for i=1:m
    Ix{i}=1; Iy{i}=1; Iz{i}=1;
    for j=1:m
        if i==j
            Ix{i}=kron(Ix{i},Ix_);
            Iy{i}=kron(Iy{i},Iy_);
            Iz{i}=kron(Iz{i},Iz_);
        else
            Ix{i}=kron(Ix{i},E);
            Iy{i}=kron(Iy{i},E);
            Iz{i}=kron(Iz{i},E);
        end
    end
end

% Constructs the Zeeman Hamiltonian
HZ=@(Bx, By, Bz) -gI*(Bx*Ix{1}+By*Iy{1}+Bz*Iz{1});
for i=2:m
    HZ=@(Bx, By, Bz) HZ(Bx, By, Bz) -gS*(Bx*Ix{i}+By*Iy{i}+Bz*Iz{i});
end

% Constructs the J-Hamiltonian
HJ=zeros(2^m);
for i=2:m
    % AX couplings
    HJ=HJ +2*pi*JIS*(Ix{1}*Ix{i}+Iy{1}*Iy{i}+Iz{1}*Iz{i});

    % XX couplings
    for j=i+1:m
        HJ=HJ +2*pi*JSS*(Ix{i}*Ix{j}+Iy{i}*Iy{j}+Iz{i}*Iz{j});
    end
end

%  Constructs the density matrix
rho0=eye(2)/2+PI*Iz_;
for i=2:m
    rho0=kron(rho0,eye(2)/2+PS*Iz_);
end

% Removes the identity
rho0=rho0-eye(2^m)/2^m;

% Constructs the propagators at zero and ultra-low field
U_ZF =expm(-1i*dt*HJ);
U_ULF=expm(-1i*dt*(HJ+HZ(Bx,0,0)));
```

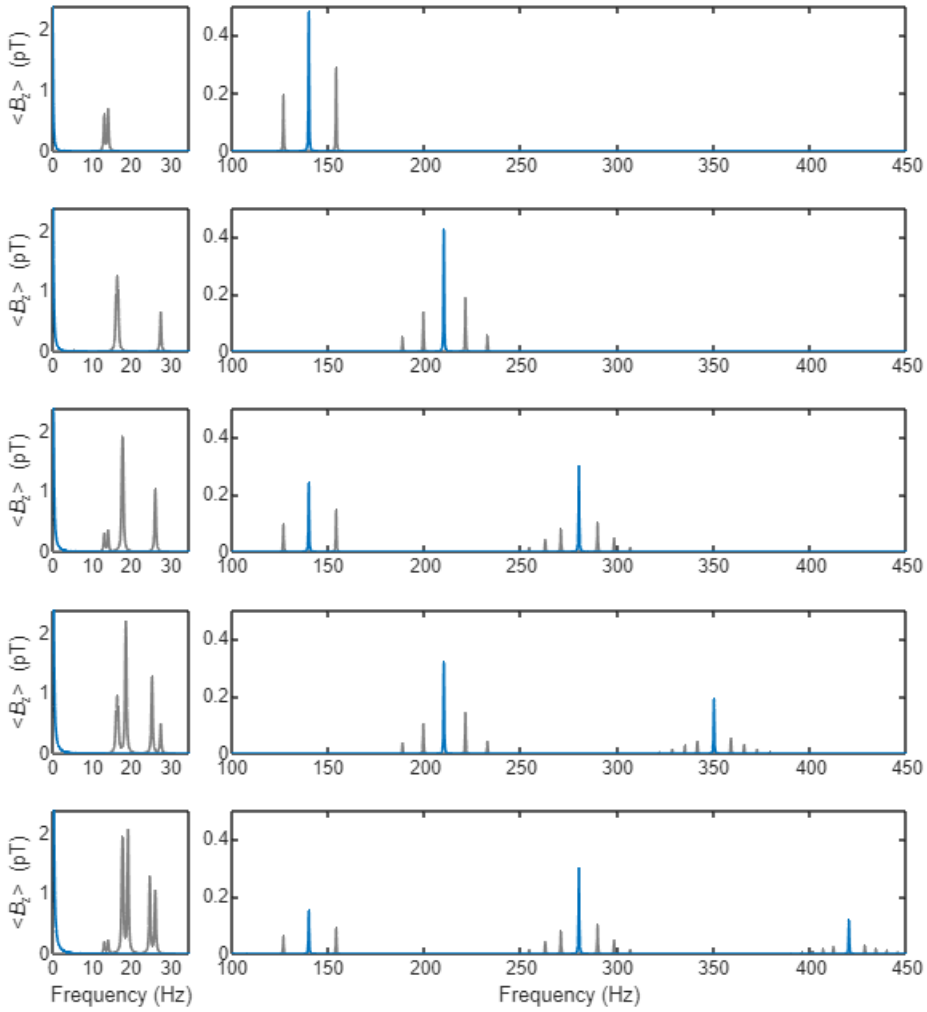
```
    % Constructs the observable operator in pT
    O=gI*Iz{1};
    for i=2:m
        O=O+gS*Iz{i};
    end

    % Conversion to pT
    O=N_molecules*hbar*1e12*mu0/2/pi/r^3*O;

    % Creates empty vectors to store the time domain signals
    TDS_ZF{n}= zeros(length(t),1);
    TDS_ULF{n}=zeros(length(t),1);

    % Propagation loop for the ZF case
    % Initialization of the density matrix
    rho=rho0;
    for k=1:K
        % Computes the expectation value of the field along z
        TDS_ZF{n}(k)=TDS_ZF{n}(k)+real(trace(O*rho));

        % Propagates during dt
        rho=U_ZF*rho*U_ZF';
    end

    % Propagation loop for the ULF case
    rho=rho0;
    for k=1:K
        % Computes the expectation value of the field along z
        TDS_ULF{n}(k)=TDS_ULF{n}(k)+real(trace(O*rho));

        % Propagates during dt
        rho=U_ULF*rho*U_ULF';
    end

    % Apodization function - line broadening
    TDS_ZF{n} =TDS_ZF{n} .*exp(-t/T2);
    TDS_ULF{n}=TDS_ULF{n}.*exp(-t/T2);

    % Fourier transform
    FDS_ZF{n}  = FourierTransform(TDS_ZF{n},L);
    FDS_ULF{n} = FourierTransform(TDS_ULF{n},L);
end
```

Plots the results

```
figure("Units","centimeters","Position",[0 0 18 19])
for i=1:N
    subplot(N,5,1+(i-1)*5)
    plot(nu, real(FDS_ULF{i}),'linewidth',lw,'color',c2), hold on
    plot(nu, real(FDS_ZF{i}), 'linewidth',lw,'color',c1), hold off
```

```
    xlim([0 35]), ylim([0 2.4])
    ylabel('<{\itB}_z> (pT)')
    xticks([0 10 20 30])
    set(gca,'fontsize',7)

    if i==N
        xlabel('Frequency (Hz)')
    end

    subplot(N,5,(i-1)*5+(2:5))
    plot(nu, real(FDS_ULF{i}),'linewidth',lw,'color',c2), hold on
    plot(nu, real(FDS_ZF{i}), 'linewidth',lw,'color',c1), hold off
    xlim([100 450]), ylim([0 0.5])
    set(gca,'fontsize',7)

    if i==N
        xlabel('Frequency (Hz)')
    end
end

exportgraphics(gcf,fullfile(FigurePath, 'XAn_system_spectra.pdf'),'ContentType','vector')
```

```
Warning: Vectorized content might take a long time to create, or it might contain unexpected results. Set
'ContentType' to 'image' for better performance. Click here to not see this message again.
```



Functions

```
function S=FourierTransform(I,N)
    % Converts to column vector
    I=I(:);

    % Window function to avoid baseline shift
    I=[I(1); 2*I(2:end)];

    % Fourier transform
    S = real(fft(I,N))/N;

    % Remove negative frequencies
    S=2*S(1:N/2);
end
```