



EPR Simulations with EasySpin

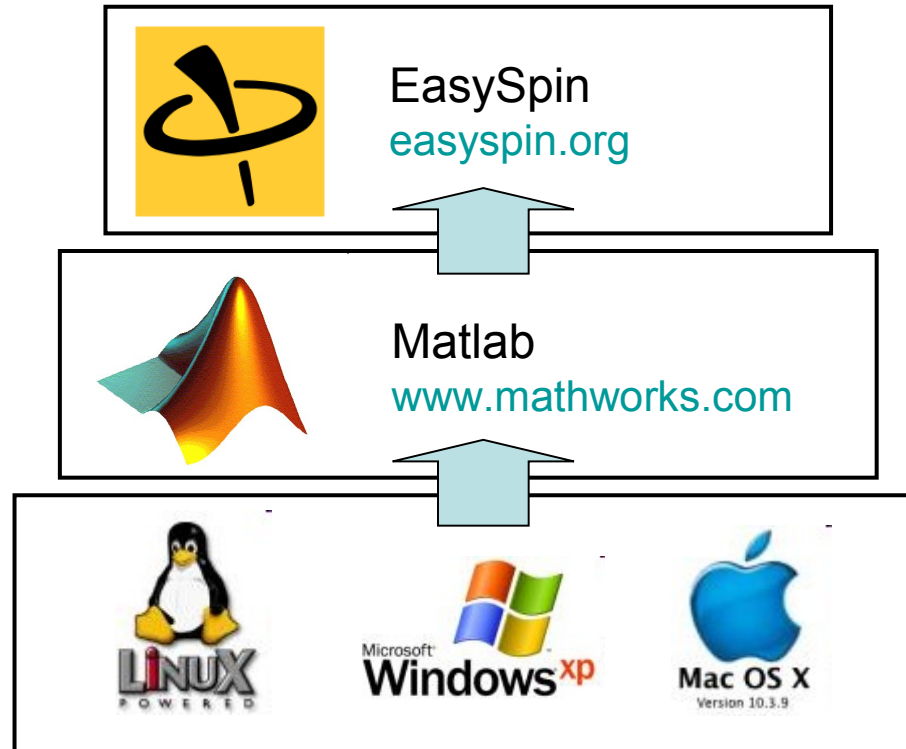
Stefan Stoll

Department of Chemistry
University of California, Davis

based on EasySpin 3.1
Oxford, 24 September 2009



Overview





Matlab and EasySpin documentation

Web resources

Matlab documentation

<http://www.mathworks.com/access/helpdesk/help/helpdesk.html>

EasySpin documentation

<http://easyspin.org>

Matlab newsgroup

<http://www.mathworks.com/matlabcentral/newsreader/>

<http://groups.google.com/group/comp.soft-sys.matlab/topics>

Matlab file exchange

<http://www.mathworks.com/matlabcentral/fileexchange>

Local Matlab and EasySpin documentation

- press <F1>
- go to *Start > Toolboxes > EasySpin > Help*
- type **doc** in command window
- point browser to <http://<EasySpinDir>/documentation/index.html>

- type **doc plot** in command window to get help on function **plot**



EasySpin references

S. Stoll, A. Schweiger,

EasySpin, a comprehensive software package for spectral simulation and analysis in EPR

J. Magn. Reson. 178, 42-55 (2006), <http://dx.doi.org/10.1016/j.jmr.2005.08.013>

The canonical publication. Please cite if you publish results obtained with the help of EasySpin.

A PDF version is included in every EasySpin installation.

S. Stoll, R. D. Britt,

General and efficient simulation of pulse EPR spectra

Phys. Chem. Chem. Phys. 11, 6614-6635 (2009),

Contains background to the current pulse EPR capabilities

S. Stoll, A. Schweiger,

Easyspin: simulating cw ESR spectra,

Biol. Magn. Reson. 27, 299-321 (2007)

An overview of EasySpin with special focus on the simulation of nitroxide spectra.

S. Stoll, A. Schweiger

An adaptive method for computing resonance fields for continuous-wave EPR spectra

Chem. Phys. Lett. 380, 464-470 (2003), <http://dx.doi.org/10.1016/j.cplett.2003.09.043>

S. Stoll

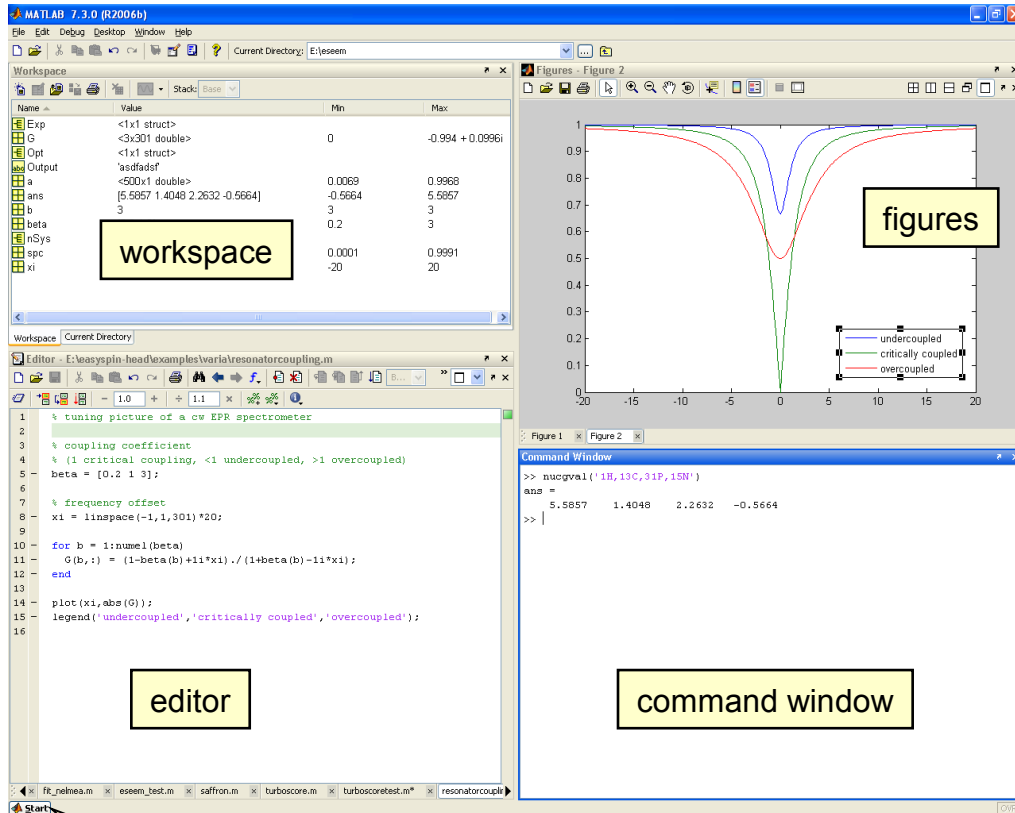
Spectral Simulations in Solid-State Electron Paramagnetic Resonance

PhD thesis, ETH Zurich, 2003, <http://e-collection.ethbib.ethz.ch/show?type=diss&nr=15059>

S. Stoll

EasySpin: a Software Package for the Computation of EPR and ENDOR spectra

EPR Newsletter 13(1-2), 24-26 (2003)



The screenshot displays the MATLAB 7.3.0 (R2004b) environment. The **workspace** window shows variables: Exp (1x1 struct), G (3x301 double), Opt (1x1 struct), Output (asdfsdf), a (500x1 double), ans (5x1 double), b (3), beta (3), nSys (3), spc (3), and xi (20). The **editor** window shows a script for tuning an EPR spectrometer, including comments for coupling coefficient, beta, and frequency offset, and code for plotting the magnitude of the transfer function G. The **figures** window shows a plot of |G| versus frequency offset xi, with three curves labeled 'undercoupled', 'critically coupled', and 'overcoupled'. The **command window** shows the execution of the command `nucgval('1H,13C,31P,15N')` resulting in the vector `ans = 5.5857 1.4048 2.2632 -0.5664`. The **start menu** is indicated at the bottom left.

Three ways to run your code from the editor

- press **<Ctrl>+<Enter>**, or
- select portion of code you want to run and press **<F9>**, or
- save as a file and press **<F5>**



Matlab: Vectors and arrays

Row vector (= 1xN array)

```
g = [2.21 2.21 2.03];
```

commas or spaces along rows
semicolons to separate rows

Column vector (= Mx1 array)

```
f = [1; 1; 2; 3; 5; 8];
```

Arrays (matrices)

```
M = [4 5; 6 7; 8 9];
```

Array functions

zeros

array with all 0

ones

array with all 1

rand

random numbers

Array manipulations

x(:)	make column vector
x.'	transpose
x'	adjoint
M(3,1) = pi;	set element
f(2) = [];	delete element
f(1:2) = 1;	set elements
f(3:end) = 0;	set elements
x(:,3) = 0;	set column
x(2,:) = [];	delete row
MM = [M; M]	concatenate

Mathematical operations

x*y	vector/matrix multiply
x.*y	elementwise multiply
f.^2	elementwise squared

Matlab is case-sensitive!
So **mwFreq** and **mwfreq** are treated as two separate variables!



Matlab: Strings

String enclosed between to single quotes

```
Algorithm = 'simplex';
Nucs = '63Cu,14N';
```

Concatenate two strings

```
y = 'Jerry';
x = ['Tom', ' and ', y];
```

Two single quotes to include a quote

```
x = 'King's Arms';
```

String = array of characters

String manipulations

<code>x(3)</code>	extract character
<code>x(1:4)</code>	extract sub-string
<code>x(1:4)=[]</code>	delete sub-string
<code>findstr(a,x)</code>	find one string in another
<code>strcmp(x,y)</code>	comparing strings
<code>strrep(x,a,b)</code>	search and replace

Converting to/from numbers

<code>str2num</code>	string to number
<code>num2str</code>	number to string
<code>mat2str</code>	matrix to string
<code>sprintf</code>	general conversion function (similar to C programming)



Matlab: Structures

Structure = collection of variables (called fields), with a name

An example

```
Software.Name = 'EasySpin';  
Software.YOB = 2000;  
Software.Version = [3 1 0];  
Software.Price = 0;  
Software.Users = 'many';  
Software.Developers = 1;  
Software.Functions = 97;  
Software.Bugs = NaN;  
Software.Tests = 286;  
Software.References = 210;
```

EasySpin spin system

```
Nitroxide.g = [2.0088 2.0061 2.004];  
Nitroxide.Nucs = '14N';  
Nitroxide.A = [16 16 86];  
Nitroxide.lwpp = 1;
```

EasySpin experimental parameters

```
Experiment.mwFreq = 94.9;  
Experiment.CenterSweep = [3462 50];
```



Matlab: Cell arrays

Cell arrays = arrays where each element can be **of arbitrary size and type**

```
A{1} = 'EPR';  
A{2} = [1 1 2 3 5 7];  
A{3} = sin(pi/5);  
A{4} = {'x',12};
```

$A(k)$ addresses the cell no k

$A\{k\}$ addresses the content of cell no k

```
A{3} = [];      puts an empty array into cell  $k$   
A(3) = [];     removes cell  $k$  altogether
```



Matlab: Comments

One-line comments: percentage sign

```
a = log(5);           % a sophisticated computation
```

Multi-line comments: %{ and %}

```
a = [5, 3i, 1.4];    % values to be squared
%{
b(1) = a(1)^2;       % element-by-element
b(2) = a(2)^2;
b(3) = a(3)^2;
%}
b = a.^2;           % all in one line
```



Matlab: Scripts and functions

Matlab files `*.m` are either scripts or functions, depending in the keyword `function`. Each callable function must reside in a separate file.

Functions can have sub-functions, which are not accessible from outside.

mysims.m Matlab script

```
Nx.Nucs = '14N';  
Nx.A = [16 16 86];  
Nx.tcorr = 1e-9;  
Ex.mwFreq = 9.5;  
[B,spc] = chili(Nx,Ex);  
save mysims Nx Ex B spc
```

affects variables on the workspace

addnoise.m Matlab function

```
function ynoisy = addnoise(y,snr)  
  
noise = rand(size(y)) - 0.5;  
noiselevel = snr*(max(y)-min(y));  
ynoisyy = y + noiselevel*noise;
```

has its own workspace

Calling the script

```
>> mysims
```

Calling the function

```
>> spcn = addnoise(sim,0.2)
```



EasySpin: Versions and installation

Software versions:

- most recent versions: Matlab R2009b, EasySpin 3.1
- EasySpin supports all Matlab versions starting from 6.5 (R13, from 2003)
- Matlab: new version every half year (e.g. R2008a, R2008b, R2009a, R2009b)
- EasySpin: major new version every year, with bug fix releases more often

Installation procedure

- Download zip file from EasySpin website easyspin.org

**Check easyspin.org
for any changes!**

- Unpack zip file to a directory *<mydir>*
e.g. to **C:/EasySpin** or **/usr/var/EasySpin**

- EasySpin folder structure:

main folder	<i><mydir>/easyspin-3.1.0</i>
toolbox functions	<i><mydir>/easyspin-3.1.0/easyspin</i>
documentation	<i><mydir>/easyspin-3.1.0/documentation</i>
examples	<i><mydir>/easyspin-3.1.0/examples</i>

- Launch Matlab
- Go to *File > Set path ...*
- Remove any old EasySpin entries from Matlab path
- Add the folder *<mydir>/easyspin-3.1.0/easyspin* to Matlab path
- Save and close the Set path window

- Type **easyspincompile** in command window
- Type **easyspininfo** in command window



Loading and saving data

Loading and importing

`eprload` reads data from Bruker spectrometers
ESP: .spc/.par, BES3T: .DTA/.DSC
also loads experimental parameters

`load` load data stored in Matlab format (.mat)

`textread` reads text (ASCII) data

`xlsread` reads Excel spreadsheets (.xls)

Import wizard: File > Import Data...
or through workspace toolbar

```
[B,spc] = eprload('myoglobin.spc');  
[B,spc] = eprload('catalase.DSC');  
[B,spc] = textread('data.txt', '%f %f', 'headerlines', 12)
```

Saving and exporting

`save` saves data in Matlab format (.mat)

`save -ASCII` saves text (ASCII) data

`xlswrite` saves as Excel spreadsheet (.xls)

```
data = [B spc];  
save -ASCII mysims.txt data  
xlswrite('mysims', data, 'simul', 'B34');
```



Data work-up

Baseline correction etc

<code>basecorr</code>	polynomial baseline correction
<code>exponfit</code>	fits exponentials to data
<code>mean</code>	computes the mean
<code>smooth</code>	data smoothing
<code>addnoise</code>	adds noise to data

Fourier transform etc

<code>fft</code>	Fast Fourier Transform (complex)
<code>fdaxis</code>	frequency-domain axis
<code>ctafft</code>	cross-term averaged FFT
<code>apowin</code>	apodization window (Hamming etc)

Complex data

<code>real</code>	real part
<code>imag</code>	imaginary part
<code>abs</code>	magnitude



Plotting data

Plotting functions

```
plot(spc)
plot(B,spc)
plot(B,spc,'r')
plot(B1,spc1,'r',B2,spc2,'g')
```

```
subplot(2,1,1)
subplot(2,1,2)
```

```
semilogx(x,y)
semilogy(x,y)
loglog(x,y)
```

```
stackplot(x,y)
```

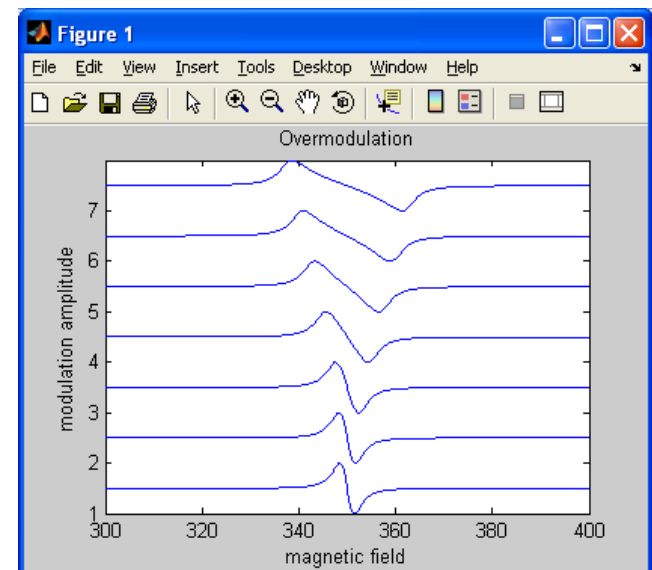
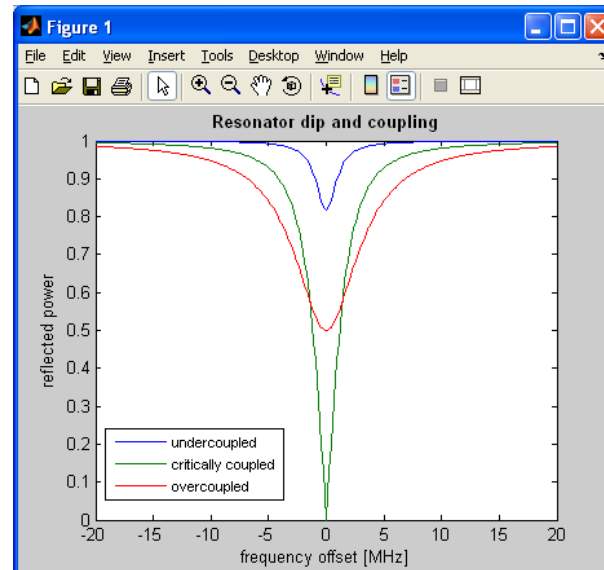
```
xlim([0 30])
ylim([0 2])
```

Changing the appearance of plots

```
xlabel('magnetic field [mT]');
ylabel('intensity');
title('EPR spectrum');
legend('first','second','third');
```

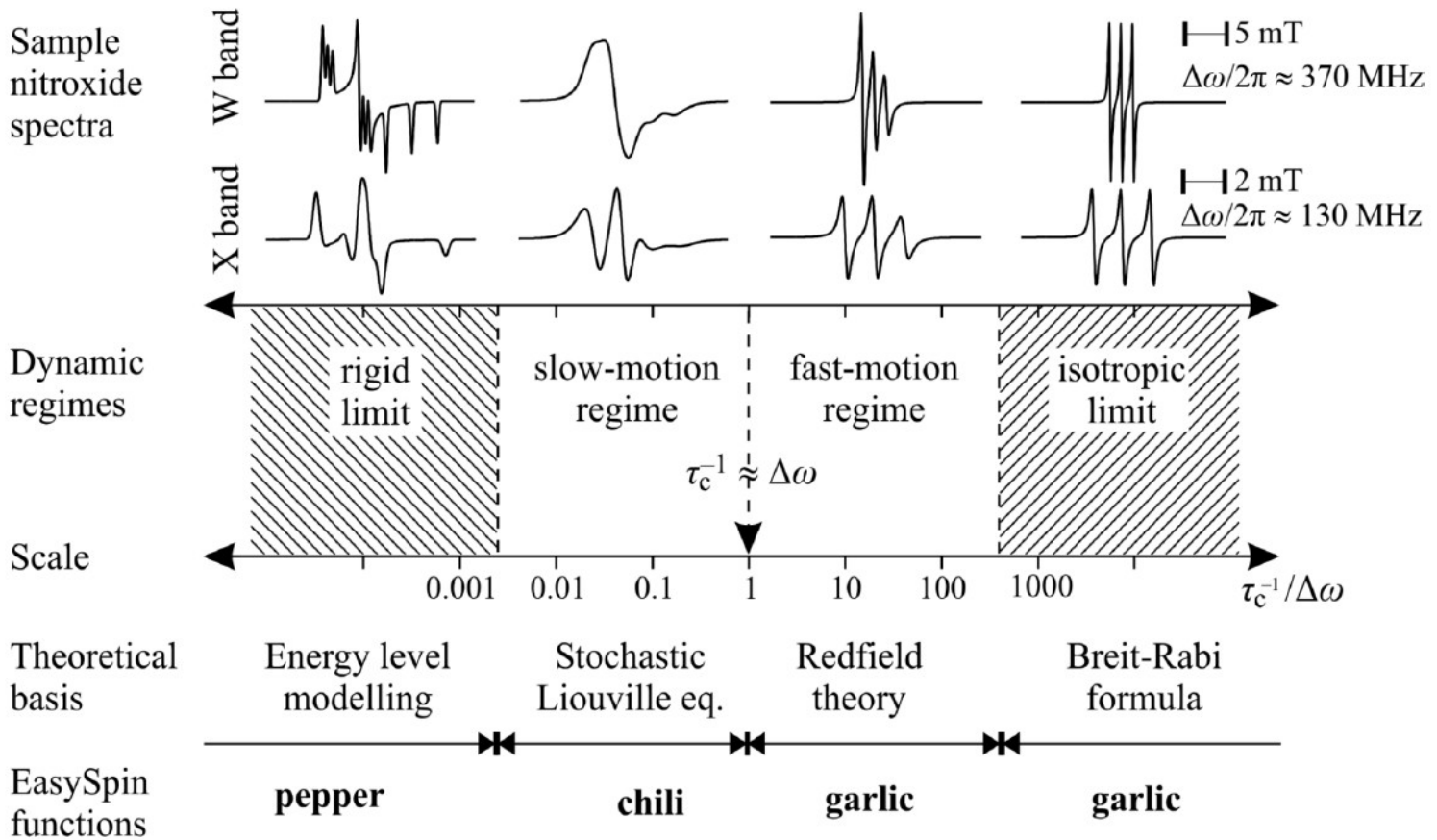
```
hold on
hold off
```

```
axis tight
```





Motional regimes





EasySpin simulation functions

<code>pepper</code>	rigid-limit cw EPR
<code>garlic</code>	isotropic-limit and fast-motion cw EPR
<code>chili</code>	slow-motion cw EPR
<code>salt</code>	ENDOR
<code>saffron</code>	pulse EPR (ESEEM and ENDOR)

Calling syntax:

`pepper(Sys, Exp)`

two input arguments

`pepper(Sys, Exp, Opt)`

three input arguments

`Sys`

spin system details (structure)

`Exp`

experimental details (structure)

`Opt`

simulation options (structure)

`pepper(...)`

plots the simulated spectrum

`spc = pepper(...)`

returns simulated spectrum

`[B, spc] = pepper(...)`

returns spectrum and field axis



Electron spins

Input syntax

<code>Sys.S = 1/2</code>	one electron spin $S = 1/2$ (default, no need to specify explicitly)
<code>Sys.S = 5/2</code>	one electron spin $S = 5/2$
<code>Sys.S = [1/2, 1/2]</code>	two electron spins $S_1 = S_2 = 1/2$

Possible values

<code>pepper</code>	solid-state cw EPR	arbitrary number, arbitrary spin
<code>garlic</code>	isotropic, fast-motion	one spin with $S = 1/2$
<code>chili</code>	slow-motion	one spin with $S = 1/2$
<code>salt</code>	ENDOR	arbitrary number, arbitrary spin
<code>saffron</code>	pulse EPR	one spin with arbitrary S



g tensors

$$(\mu_B / h) \mathbf{B} \mathbf{g} \mathbf{S} = (\mu_B / h) (B_x g_x S_x + B_y g_y S_y + B_z g_z S_z)$$

g values, one electron spin

`Sys.g = [1.9, 2.1, 2.2]`

`Sys.g = [1.9, 2.2]`

`Sys.g = 2`

rhombic *g*

axial *g*, = [1.9, 1.9, 2.2]

isotropic *g*, = [2, 2, 2]

g values, two electron spins

`Sys.g = [2,2,2; 2.1,2.1,2]`

`Sys.g = [2 2.1; 2.05 1.98]`

`Sys.g = [2.03; 2.01]`

one row per electron spin

rhombic *g*

axial *g*

isotropic *g*

Tensor orientation

`Sys.gpa = [0, pi/4, 0]`

Units: radians

one row per electron spin

orientation of *g* tensor frame in molecular frame

Frequency/field/*g* conversions

$$(\nu/\text{GHz}) = 13.9962 g (B_0/\text{T})$$

$$71.4477 (\nu/\text{GHz}) = g (B_0/\text{mT})$$

or use `eprconvert`

Frequency	Magnetic field
9800 MHz	318.267 mT
9.8 GHz	0.318267 T
0.326893 cm ⁻¹	3182.67 G
4.05295e-005 eV	3.18267 kG



Zero-field splittings

Hamiltonian (in frequency units)

$$S D S = D_x S_x^2 + D_y S_y^2 + D_z S_z^2 = D(S_z^2 - S^2 / 3) + E(S_x^2 - S_y^2)$$

$$D = \begin{pmatrix} D_x & 0 & 0 \\ 0 & D_y & 0 \\ 0 & 0 & D_z \end{pmatrix} = \begin{pmatrix} -\frac{1}{3}D + E & 0 & 0 \\ 0 & -\frac{1}{3}D - E & 0 \\ 0 & 0 & \frac{2}{3}D \end{pmatrix}$$

EasySpin input

Units: MHz

1 cm⁻¹ = 30 GHz

```
Sys.D = 100
```

```
Sys.D = [100 10]
```

```
Sys.D = [-40 -40 80]
```

$D = 100$ MHz, $E = 0$

$D = 100$ MHz, $E = 10$ MHz

$[D_x, D_y, D_z]$ in MHz

Tensor orientation

Units: radians

```
Sys.Dpa = [0;40;32]*pi/180;
```

EasySpin also supports higher-order zero field terms. See documentation.



Magnetic nuclei

Specifying nuclear spins

comma-separated list of isotopes

```
Sys.Nucs = '1H';  
Sys.Nucs = '63Cu,14N';  
Sys.Nucs = '13C';  
Sys.Nucs = '1H,1H,1H,1H';
```

Helper functions to add or remove nuclear spins from the spin system

```
Sys = nucspinadd(Sys, '1H', [2 8]);  
Sys = nucspinrmv(Sys, 2);
```



Magnetic nuclei

Nuclear constants

isotopes interactive PSE with nuclear isotope database

nucabund natural abundance
nucgval nuclear g factors
nucqmom nuclear electric quadrupole moments
nucspin nuclear spin quantum numbers

larmorfrq Larmor frequency

The screenshot shows a software window titled "Nuclear spins" with a periodic table. Elements are color-coded: blue for alkali and alkaline earth metals, yellow for non-metals, red for transition metals, green for lanthanides and actinides, and white for noble gases. A data table at the bottom lists isotopes of Sn:

115	Sn	0.5	+4.9028	13.0802 MHz	0.34%
117	Sn	0.5	+5.34139	14.2503 MHz	7.68%
119	Sn	0.5	-2.09456	5.58809 MHz	8.59%

At the bottom of the window, there is a text input field for "Magnetic field [mT]" with the value "350" entered.



Magnetic nuclei

Natural-abundance mixtures: omit mass number

`Sys.Nucs = 'Cu' ;`

69% ^{63}Cu , 31% ^{65}Cu

`Sys.Nucs = 'Cu,14N' ;`

`Sys.Nucs = 'Cu,Cl' ;`

53% $^{63}\text{Cu}/^{35}\text{Cl}$, 23% $^{65}\text{Cu}/^{35}\text{Cl}$, 17% $^{63}\text{Cu}/^{37}\text{Cl}$, etc

`Sys.Nucs = 'C' ;`

98.9% ^{12}C , 1.1% ^{13}C

"natural abundance" can vary!

Mixtures with custom abundances: separate field

`Sys.Nucs = '(14,15)N' ;`

`Sys.Abund = [50 50] ;`

`Sys.Nucs = 'Cu, (32,33)S' ;`

`Sys.Abund = [0.1 0.9] ;`



Hyperfine tensors

Hamiltonian (in frequency units)

$$SAI = S_x A_{xx} I_x + S_y A_{yy} I_y + S_z A_{zz} I_z$$

Units: MHz

Tensor principal values

one row per nucleus

1 mT = 28 MHz
10⁻⁴ cm⁻¹ = 3 MHz

`Sys.Nucs = '1H'`

one nuclear spin

`Sys.A = 10`

isotropic A, = [10 10 10]

`Sys.A = [4, 12]`

axial A (A_{xx}=A_{yy}), = [4, 4, 12]

`Sys.A = [3, 5, 12]`

rhombic A

`Sys.Nucs = '1H,14N'`

two nuclear spins

`Sys.A = [3 5 12; 1 2 3]`

two rhombic A

Isotropic, axial, rhombic components

one row per nucleus

`Sys.A_ = [4 2 0];`

[a_{iso}, T, ρ]

$$A = a_{\text{iso}} + \begin{pmatrix} -T - \rho & 0 & 0 \\ 0 & -T + \rho & 0 \\ 0 & 0 & 2T \end{pmatrix}$$

Tensor orientation

one row per nucleus

Units: radians

`Sys.Apa = [0, pi/4, 0]`

Euler angles of A eigenframe in molecular frame



Nuclear quadrupole tensors

Hamiltonian (in frequency units)

$$IPI = I_x P_{xx} I_x + I_y P_{yy} I_y + I_z P_{zz} I_z$$

$$P = \frac{e^2 q Q}{4I(2I-1)h} \begin{pmatrix} -(1-\eta) & 0 & 0 \\ 0 & -(1+\eta) & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

$$e^2 q Q / h = 2I(I-1)P_{zz}$$

$$\eta = \frac{P_{xx} - P_{yy}}{P_{zz}}$$

Tensor principal values

one row per nucleus

Units: MHz

$$\text{Sys.Q} = 0.7;$$

$$e^2 q Q / h = 0.7 \text{ MHz}$$

$$\text{Sys.Q} = [0.7, 0.1]$$

$$e^2 q Q / h = 0.7 \text{ MHz, and } \eta = 0.1 \text{ (between 0 and 1)}$$

$$\text{Sys.Q} = [-1.2, -0.8, 2]$$

3 principal values (in MHz)

Tensor orientation

one row per nucleus

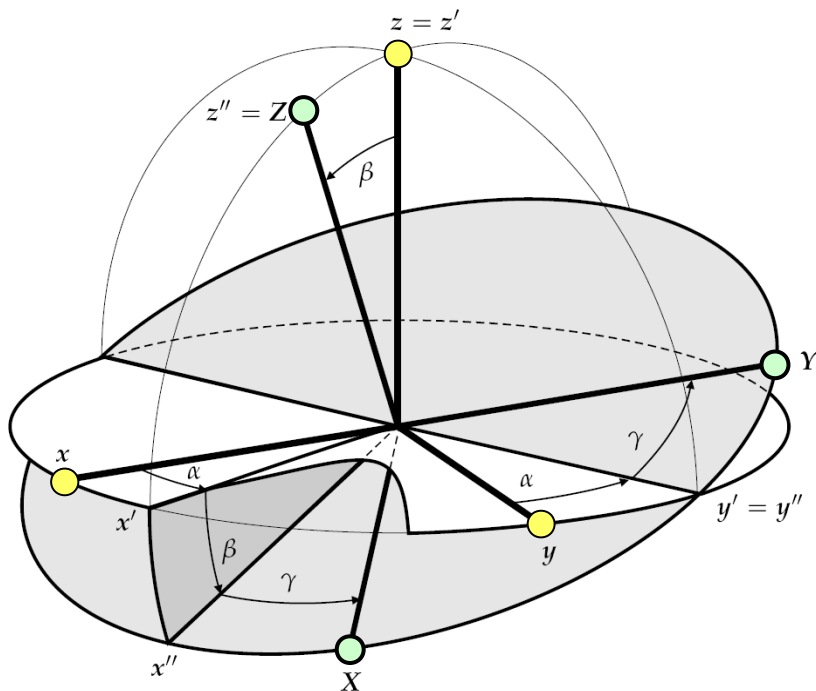
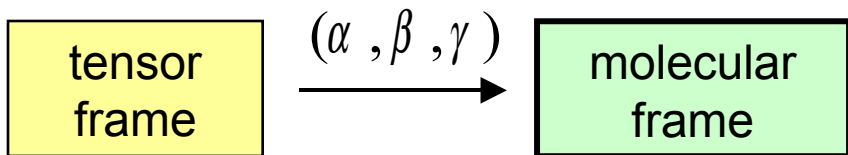
Units: radians

$$\text{Sys.Qpa} = [0, \pi/4, 0]$$

Euler angles of Q eigenframe in molecular frame



Orientations in EasySpin



```
% g tensor in eigenframe
g_eig = diag([1.9, 2.0, 2.1])
1.9000      0      0
0      2.0000      0
0      0      2.1000
```

```
% Euler angles alpha, beta, gamma (in radians)
abc = [10 45 30]*pi/180
0.1745      0.7854      0.5236
```

```
% Rotation matrix
R = erot(abc)
0.5162      0.5987      -0.6124
-0.4986      0.7915      0.3536
0.6964      0.1228      0.7071
rows of R: molecular axes in g frame
columns of R: g axes in molecular frame
```

```
% g tensor in molecular frame
g_mol = R*g_eig*R'
2.0108      0.0041      -0.0793
0.0041      1.9876      0.0597
-0.0793      0.0597      2.0015
```

$$\begin{aligned}
 R &= R_{z''}(\gamma) \cdot R_{y'}(\beta) \cdot R_z(\alpha) \\
 &= \begin{pmatrix} c\gamma & s\gamma & 0 \\ -s\gamma & c\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c\beta & 0 & -s\beta \\ 0 & 1 & 0 \\ s\beta & 0 & c\beta \end{pmatrix} \cdot \begin{pmatrix} c\alpha & s\alpha & 0 \\ -s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
 &= \begin{pmatrix} c\gamma c\beta c\alpha - s\gamma s\alpha & c\gamma c\beta s\alpha + s\gamma c\alpha & -c\gamma s\beta \\ -s\gamma c\beta c\alpha - c\gamma s\alpha & -s\gamma c\beta s\alpha + c\gamma c\alpha & s\gamma s\beta \\ s\beta c\alpha & s\beta s\alpha & c\beta \end{pmatrix}
 \end{aligned}$$



Static broadenings

Convolutional broadenings

isotropic only

Units: mT

phenomenologically account for line broadening

```
Sys.lwpp = [0.2 0.3];
Sys.lwpp = 0.3;
Sys.lw = [0 0.1];
```

1st element Gaussian
2nd element Lorentzian (optional)
lwpp peak-to-peak linewidth
lw full width at half maximum (FWHM)

```
Sys.lwEndor = 0.3;
```

(only **salt** and **saffron**), in MHz

Strain broadenings (only **pepper**)

isotropic or anisotropic

Units: MHz

line broadening due to unresolved hf splittings and site-to-site disorder

Sys.HStrain unresolved hyperfine splittings (MHz)

Sys.gStrain distribution of g values

Sys.AStrain distribution of A values (correlated with g) (MHz)

Sys.DStrain distribution of D and E (MHz)



Rotational diffusion

Rotational correlation time

(**garlic** and **chili**)

Units: s

```
Sys.tcorr = 1e-9;  
Sys.logtcorr = -9;
```

$$\tau_c = \frac{1}{6R}$$

Diffusion tensor

(**chili** only)

Units: 1/s

```
Sys.Diff = 1e9;  
Sys.logDiff = 9;
```

isotropic

```
Sys.Diff = [1 2]*1e8;  
Sys.logDiff = [8 8.3];
```

axial

```
Sys.Diff = [2,2.5,3]*1e8;  
Sys.logDiff = [8.3,8.4,8.5];
```

rhombic

```
Sys.Diffpa = [0 20 0]*pi/180;
```

tilt angles for
tensor orientation



Static orientational distributions

Orientation of a paramagnet in the sample:

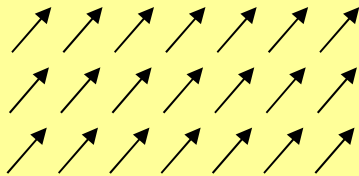
$$\Omega = (\phi, \theta, \chi)$$

Distribution of orientations of the paramagnets:

$$P(\Omega)$$

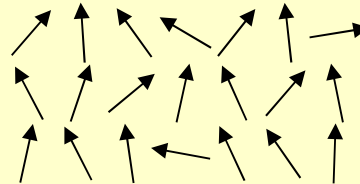
Crystals

one or several distinct orientations with equal probabilities



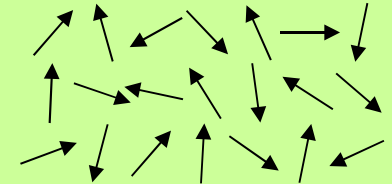
Partially oriented systems

probability is varying function of orientation



Powders

all orientations with equal probability



Exp. Orientations
Exp. CrystalSymmetry

Exp. Ordering
partial MOMD

default
MOMD



Single crystals

Specifying site splittings via the crystal symmetry

space group → number and orientation of sites in unit cell → site splitting

<code>Exp.CrystalSymmetry = 'P21/m';</code>	space group symbol
<code>Exp.CrystalSymmetry = 11;</code>	space group number (between 1 and 230)
<code>Exp.CrystalSymmetry = 'C2h';</code>	point group, Schönflies notation
<code>Exp.CrystalSymmetry = '2/m';</code>	point group, Hermann-Mauguin notation

Crystal orientation in the spectrometer

<code>Exp.Orientations = [0;0;0];</code>	orientation of the crystal
<code>Exp.Orientations = [0 0;0 pi/2;0 0];</code>	orientation of domains of a twin crystal

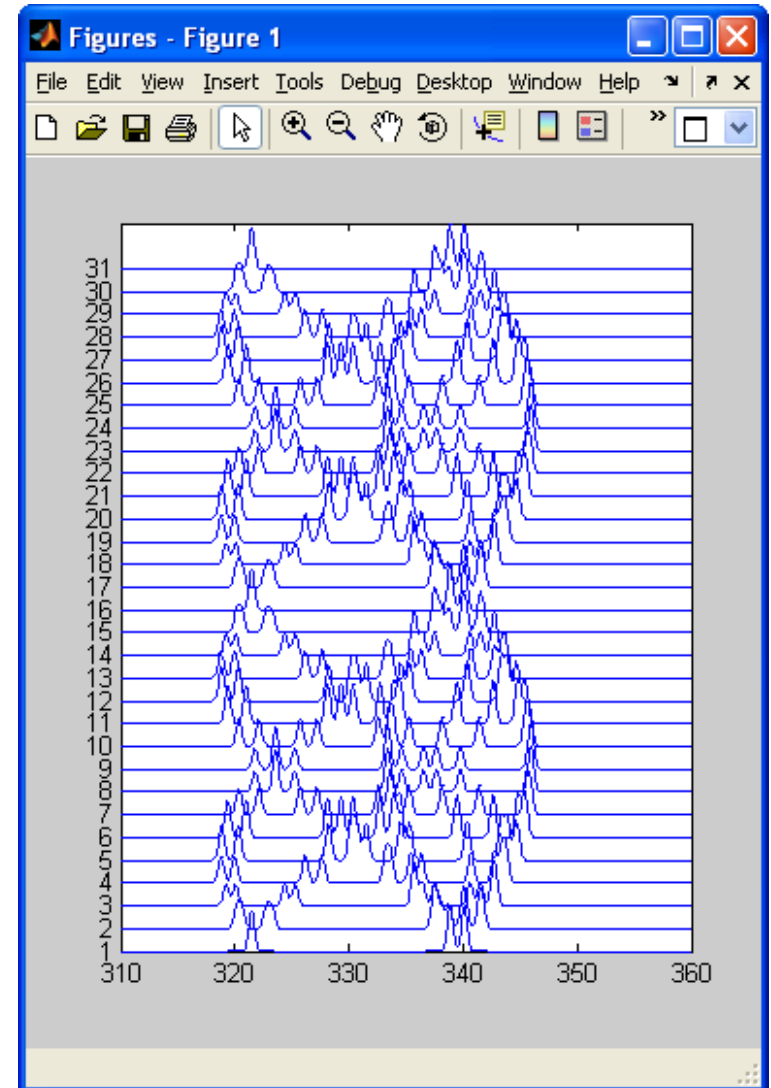


Single crystals

Crystal rotations

- specify rotation axis
- generate rotated orientations
- specify crystal symmetry

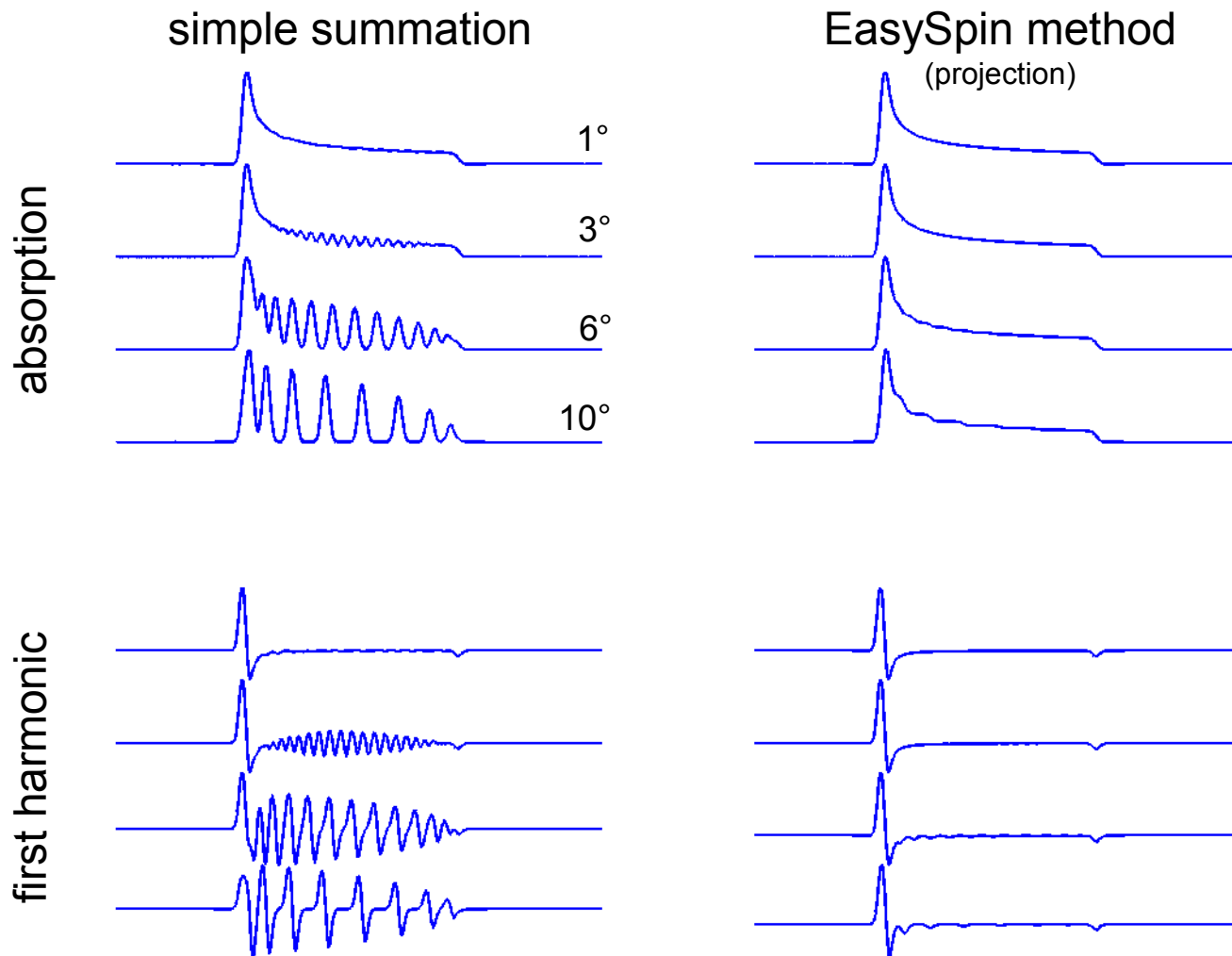
```
n = [1 0 0];  
[phi,theta] = rotplane(n,[0 pi],31);  
Exp.Orientations = [phi;theta];  
Exp.CrystalSymmetry = 'P21212';  
  
Opt.Output = 'separate';  
  
[B,spc] = pepper(Sys,Exp,Opt);  
  
stackplot(B,spc);
```





Powder simulations: orientational averaging

"Simulation grass"





Partially oriented systems

Orientational distribution

$$P(\boldsymbol{\Omega}) \propto \exp[-U(\boldsymbol{\Omega})/k_B T]$$

Orientational potential

$$U(\boldsymbol{\Omega}) = -k_B T \sum_{L,K} \lambda_K^L Y_K^L(\boldsymbol{\Omega})$$

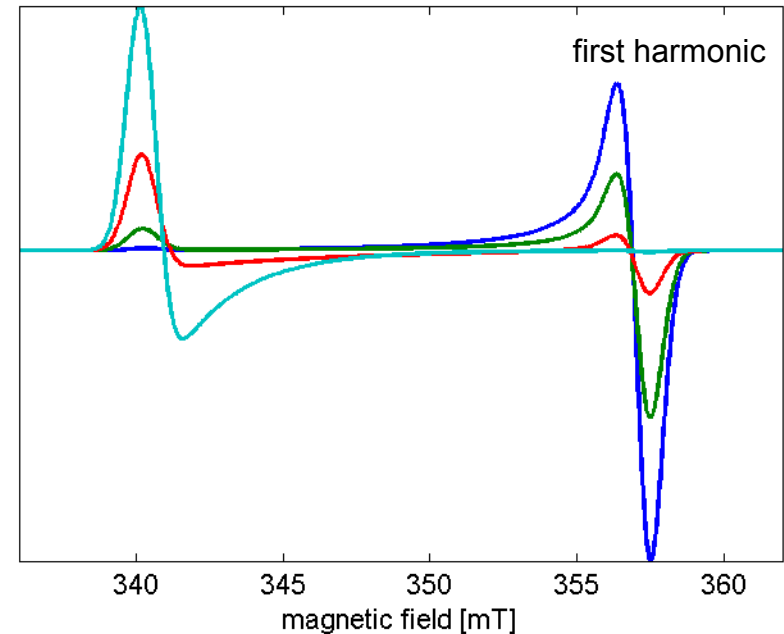
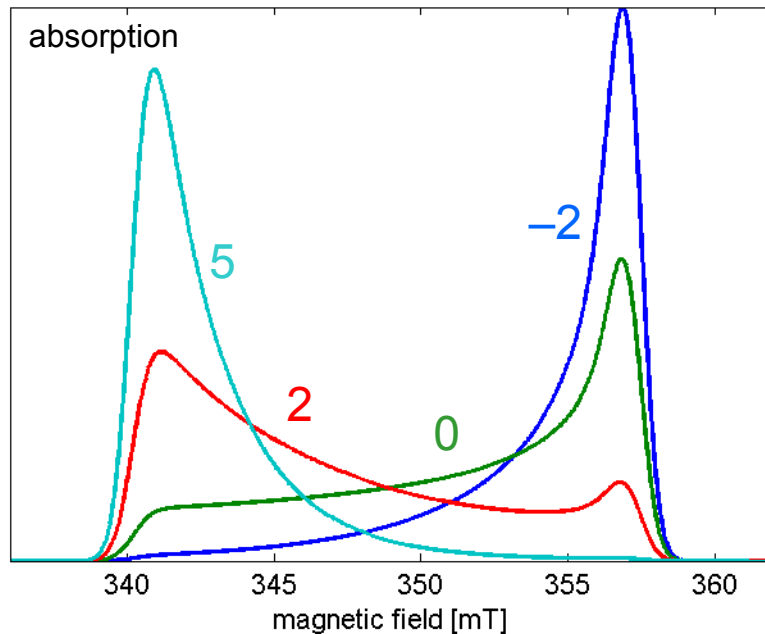
Predefined potential in EasySpin:

`Exp.Ordering = 2;`

$$U(\boldsymbol{\Omega}) = -k_B T \lambda' \frac{3 \cos^2 \theta - 1}{2}$$

Custom potential in EasySpin:

`Exp.Ordering = @(ph,th) cos(3*th).^2;`



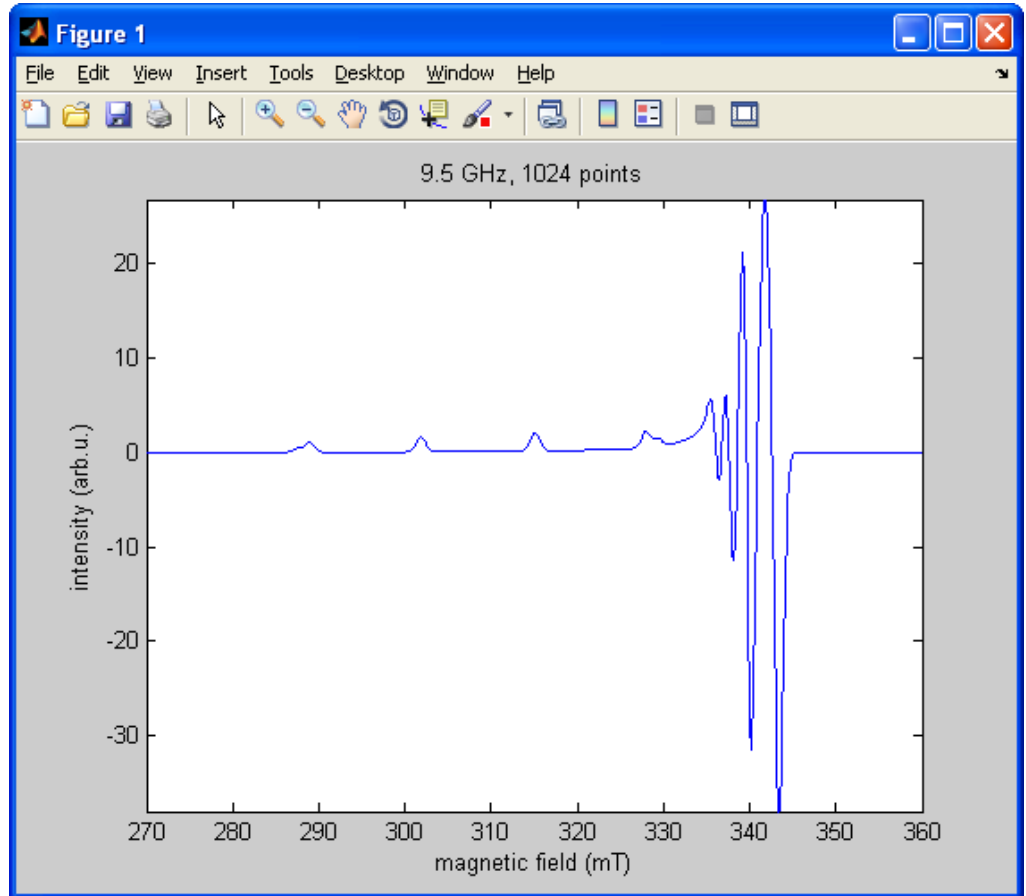


cw EPR: a first example

```
SysCu.g = [2 2.2];  
SysCu.lwpp = 1;  
SysCu.Nucs = 'Cu';  
SysCu.A = [50 400];
```

```
ExpX.mwFreq = 9.5;  
ExpX.Range = [270 360];
```

```
pepper (SysCu, ExpX);
```





Matrix diagonalization vs. perturbation theory

Matrix diagonalization

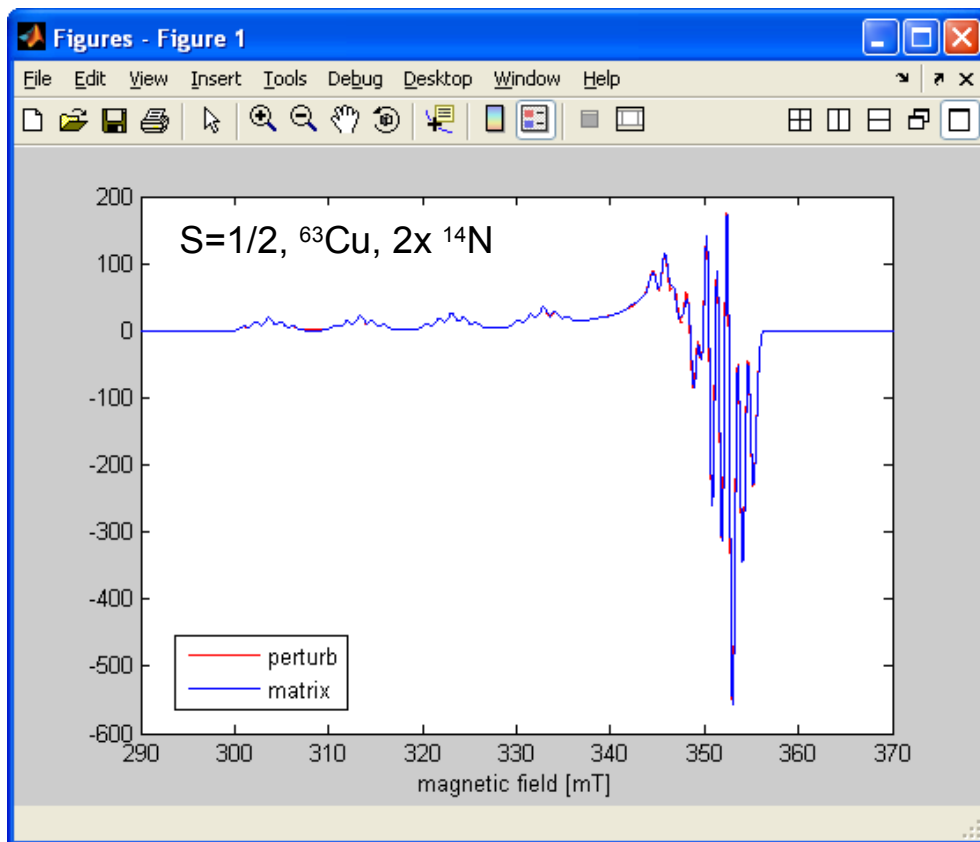
- + arbitrary number of electron and nuclear spins
- + general spin Hamiltonian
- + all transitions
- + always exact
- slow

always ok!

2nd-order perturbation theory

- for one electron spin only ($S=1/2$ and $S>1/2$)
- neglects some Hamiltonian terms
- only allowed transitions
- accurate only for small A and D
- + many nuclei possible
- + fast

use with care!

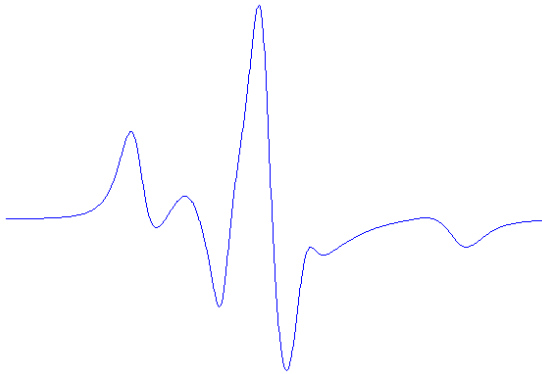
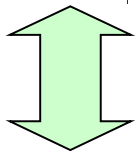
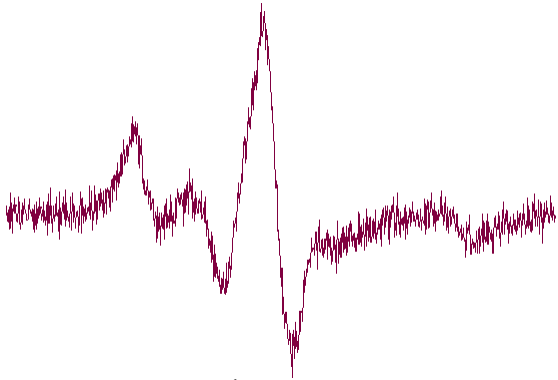


Opt.Method = 'matrix'	8.4 s
Opt.Method = 'perturb'	0.08 s



Least-squares fitting

experimental spectrum y_{exp}



model simulation, y_{sim}

Residuals

$$r_i = f(y_{i,\text{exp}}) - f(y_{i,\text{sim}}(\mathbf{p}))$$

Sum of squares

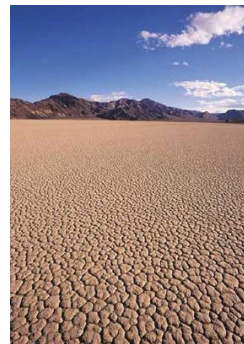
$$\chi^2 \propto \sum_{i=1}^n r_i^2$$

rmsd

$$\varepsilon = \sqrt{\chi^2 / N}$$

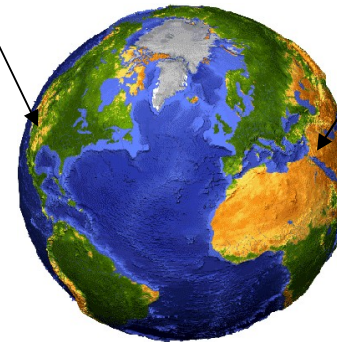
The global minimum is very difficult to locate!

a local minimum



Death Valley

the global minimum



Dead Sea shore



Least-squares fitting

EasySpin does not use analytical derivatives

Global methods

(search all over the place)

- Monte Carlo
- genetic algorithms
- simulated annealing
- tree and grid searches

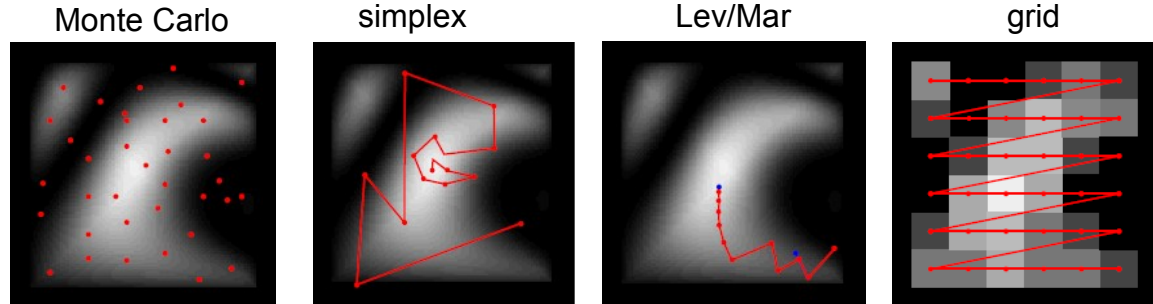
Local methods

(downhill into valley from a starting point)

- Nelder-Mead simplex
- Levenberg-Marquardt
- NL2SOL, NL2SNO
- Powell

Hybrid methods

- 1) global method to locate promising region(s)
- 2) local method to zero in on minimum



Available in EasySpin:

global methods:

- grid search
- genetic algorithm
- Monte Carlo

local methods:

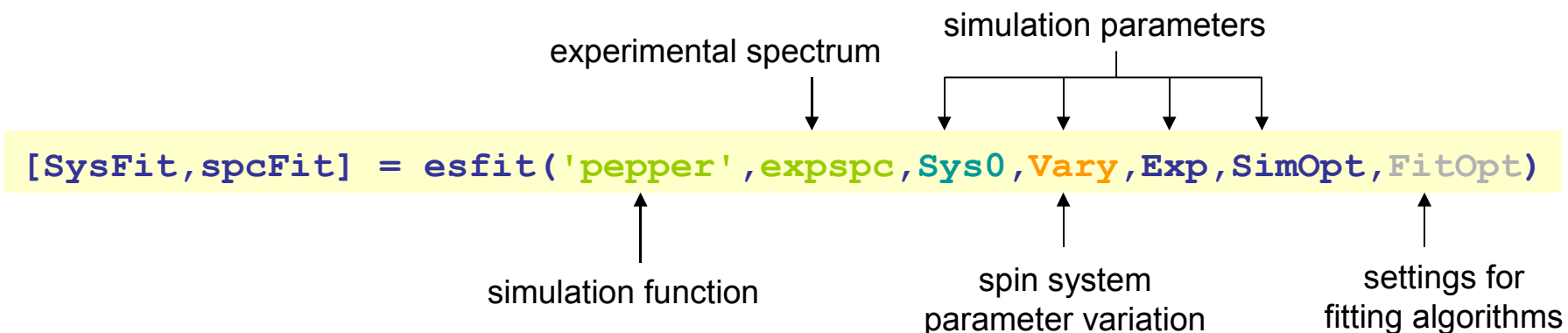
- Levenberg-Marquardt
- Nelder-Mead simplex

hybrid methods



Least-squares fitting

One interface for all simulation functions.



(1) Central values of parameters

```
Sys0.g = [2.008 2.006 2.003];  
Sys0.Nucs = '14N';  
Sys0.A = [16 16 86];  
Sys0.logtcorr = -8.5;
```

(2) Allowed variation of parameters

one parameter, for 1D search

```
Vary.logtcorr = 1;
```

two parameters, for 2D search

```
Vary.logtcorr = 0.2;
```

```
Vary.g = [0, 0, 0.0005];
```

No limit on number of parameters, but the more parameters the slower the fitting.



Least-squares fitting

Choosing the algorithm:

`FitOpt.Method`

- 'simplex' = Nelder/Mead simplex (default)
- 'levmar' = Levenberg/Marquardt
- 'montecarlo' = Monte Carlo
- 'genetic' = genetic algorithm + simplex
- 'grid' = grid search + simplex

'fcn' = use spectrum
'int' = use integral

Example:

```
FitOpt.Method = 'simplex int'
```

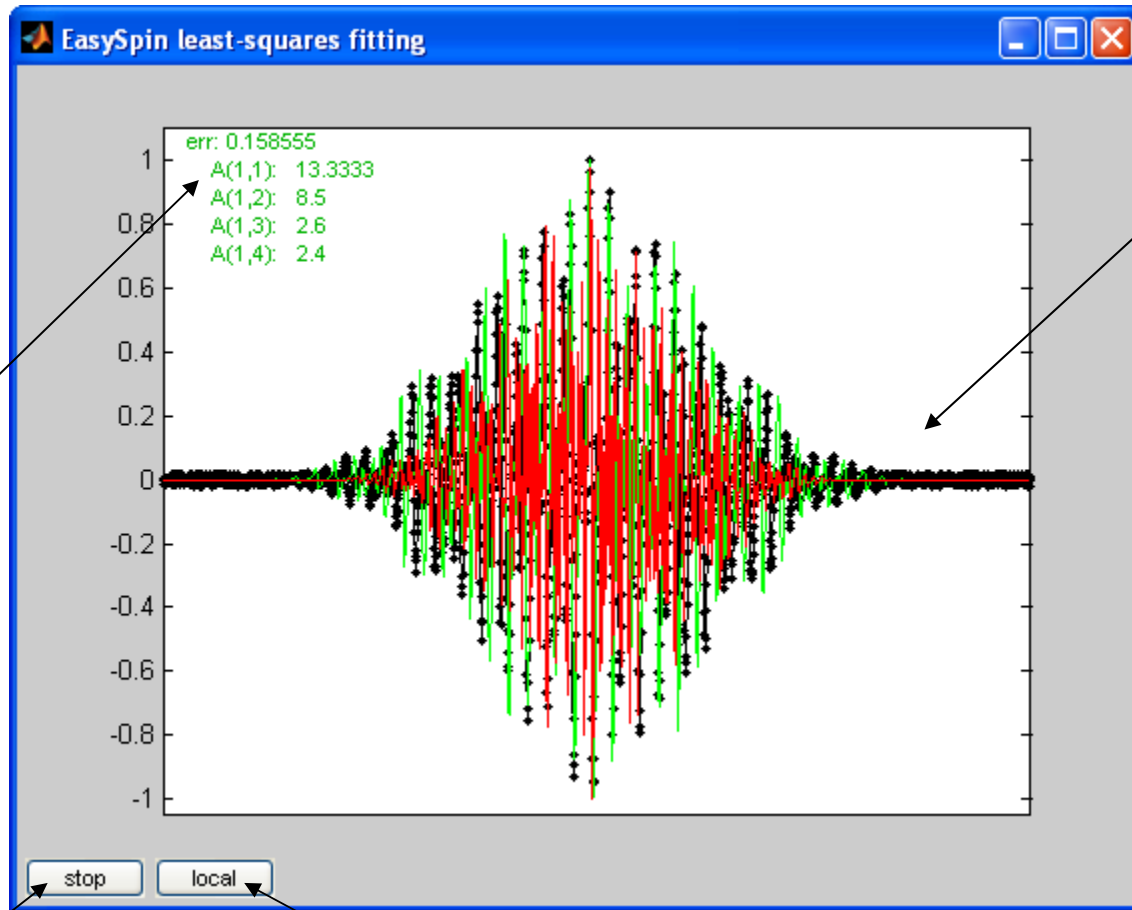
Knowing when to stop:

`FitOpt.TolFun` termination tolerance for chi-squared
`FitOpt.TolX` termination tolerance for parameter step
`FitOpt.maxTime` maximum runtime

each algorithm has different additional parameters



Least-squares fitting



current best
parameter
set and its
fitting error

black: data
red: current
simulation
green: best
simulation
so far

stops fitting

launches local search
from current best parameter set



Least-squares fitting

Quality of fit: How deep is the minimum?

indicators:

- maximum/mean/standard deviation of residuals
- residual index R
- correlation coefficient of experimental and simulated spectrum
- eye + expertise

compare to noise level σ :

- max/mean residual $< \sigma$
accurate fit, only random errors in experimental data
- max/mean residual $\gg \sigma$
inaccurate fit, systematic errors in simulated spectrum
no convergence or wrong model

Error of parameters: How steep are the the walls?

sensitivity of chi-square to change of parameter values



Least-squares fitting

Before you start fitting

- Baseline correct the spectrum; smooth the spectrum if too noisy.
- Choose a physically reasonable model (number of spins, tensors, etc).
- Keep the number of fitting parameters as small as possible.
- Find good starting parameters by (1) **analyzing your spectrum thoroughly**, (2) **studying literature of similar systems**, and (3) **using theoretical estimates**.
- Narrow down the parameter ranges as much as possible.

EasySpin (or any other program) cannot find the minimum

- if you are too far off initially.
- if the parameter ranges are too large or too small.
- if there are too many parameters.
- if the model is wrong.
- if there is too much noise or a non-flat baseline in the spectrum.

After convergence

- Estimate the goodness-of-fit at the global minimum. It might still be bad.
- Look how sensitive the goodness-of-fit is to changes in parameters.
- Even the best minimum you find might be physically incorrect.
- The global minimum might not be the real minimum due to noise effects.



Pulse EPR: A first example

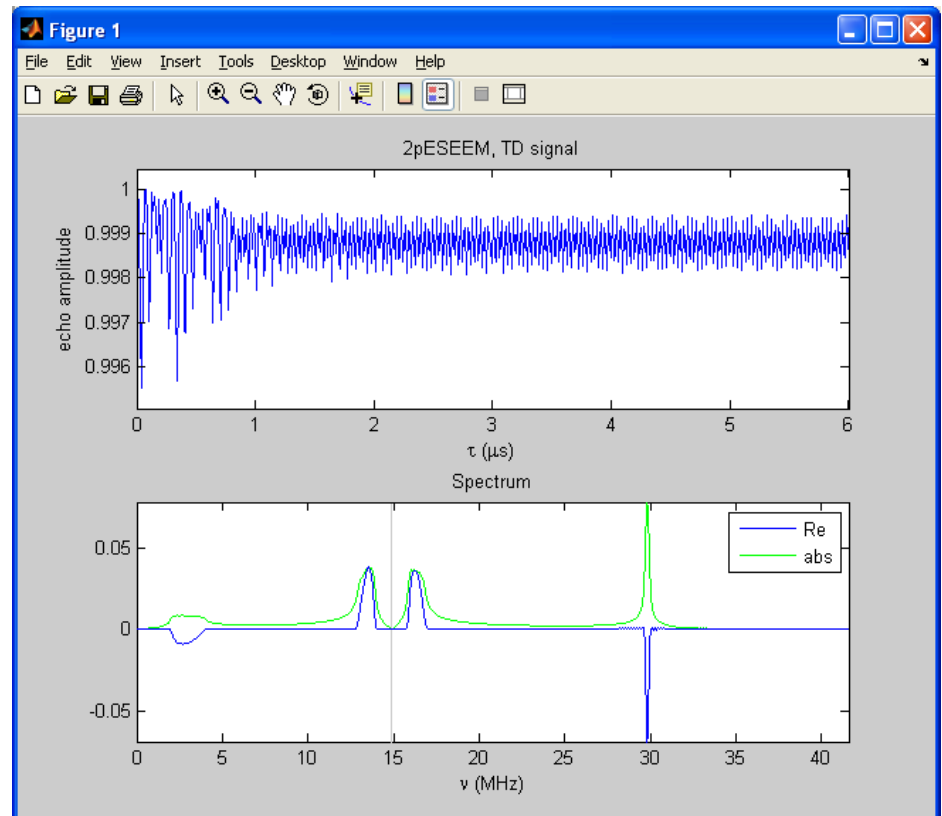
```
clear
```

```
Sys.Nucs = '1H';  
Sys.A = [2 4];
```

```
Exp.Field = 350;
```

```
Exp.Sequence = '2pESEEM';  
Exp.tau = 0.0001;  
Exp.dt = 0.012;  
Exp.nPoints = 501;
```

```
saffron(Sys,Exp);
```





Pulse EPR: Overview

Simulation function: [saffron](#)

Pre-defined pulse sequence

```
Exp.Sequence = '2pESEEM';  
Exp.Sequence = '3pESEEM';  
Exp.Sequence = 'HYSCORE';  
Exp.Sequence = 'MimsENDOR';
```

Timings

```
Exp.tau = 0.120;  
Exp.dt = 0.008;
```

Units: μs

Echo-detected field sweep?
Simulate cw EPR spectrum with
[pepper](#), using `Exp.Harmonic = 0`.

What it supports

- hard and soft pulses
- several nuclear spins
- high electron spin
- user-defined pulse sequences
- orientation selection built-in
- data processing built-in



Pulse EPR: Basic experimental settings

Magnetic field (in mT)

```
Exp.Field = 350;
```

Number of points

```
Exp.nPoints = 201;           1D and 2D
```

```
Exp.nPoints = [64 201]; 2D
```

If not given, EasySpin uses default values.

Dwell time (in μs)

```
Exp.dt = 0.008;           1D and 2D
```

```
Exp.dt = [0.008 0.012]; 2D
```

This has to be given for ESEEM experiments.

Frequency range (in MHz)

```
Exp.Range = [0 30];
```

This has to be given for ENDOR experiments.

Spectrometer frequency (in GHz)

```
Exp.mwFreq = 9.5;
```

Only needed for orientation selection.

Excitation bandwidth (in MHz)

```
Exp.ExciteWidth = 100;
```

Only needed for orientation selection.

Pulse experiment

```
Exp.Sequence
```

for pre-defined pulse experiments

```
Exp.Flip, Exp.Inc, Exp.t, Exp.tp
```

for user-defined pulse experiments



Pulse EPR: Pre-defined sequences

all times and delays
in microseconds

Two-pulse ESEEM

```
Exp.Sequence = '2pESEEM';  
Exp.tau = 0.080;
```

Three-pulse ESEEM

```
Exp.Sequence = '3pESEEM';  
Exp.tau = 0.100;  
Exp.T = 0.080;
```

HYSCORE

```
Exp.Sequence = 'HYSCORE';  
Exp.tau = 0.172;  
Exp.t1 = 0.08;  
Exp.t2 = 0.08;
```

Mims ENDOR

```
Exp.Sequence = 'MimsENDOR';  
Exp.tau = 0.100;  
Exp.Range = [0 30];
```



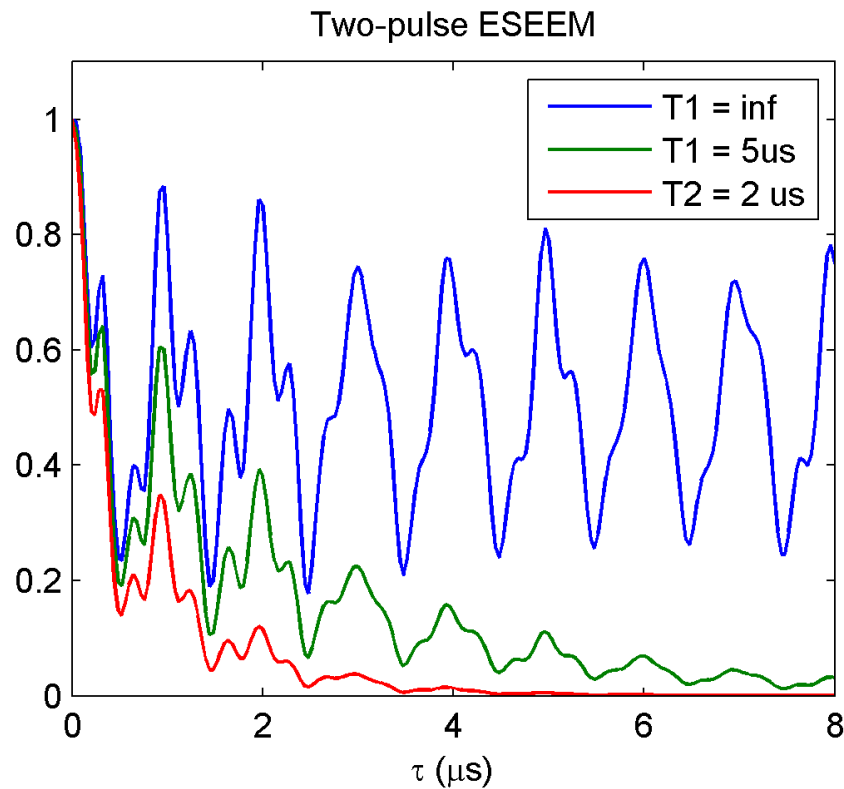
Pulse EPR: T_1 and T_2 relaxation

Relaxation time constants (time for fall-off to 36.8%)

`Exp.T1T2 = [10 2];`

first element: T_1

second element: T_2

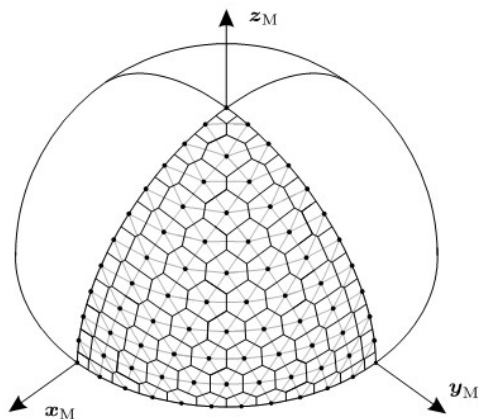




Powder spectra

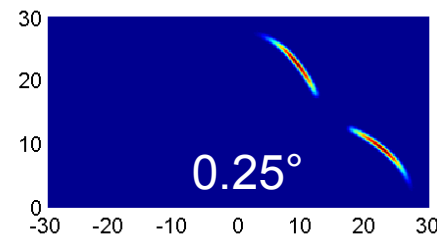
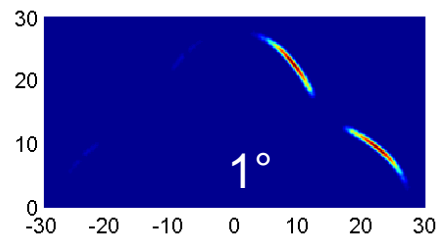
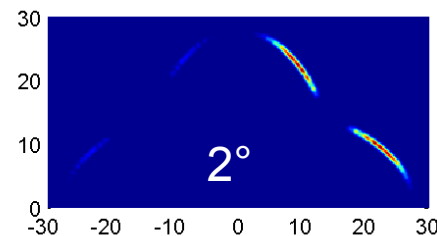
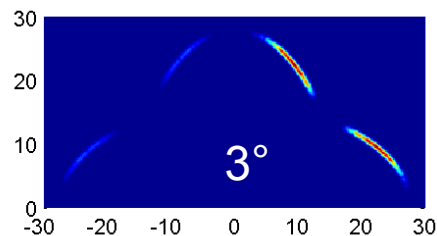
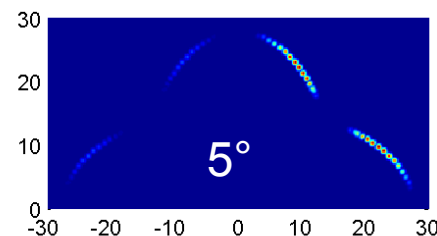
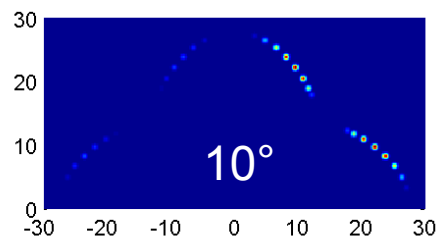
EasySpin computes powder spectra by default.

Number of orientations



Opt.nKnots gives the orientational resolution in terms of number of orientations between $\theta = 0^\circ$ (north pole) and $\theta = 90^\circ$ (equator)

Opt.nKnots = 10; 10°
Opt.nKnots = 31; 3°
Opt.nKnots = 91; 1°
Opt.nKnots = 361; 0.25°
Opt.nKnots = 901; 0.1°



The more orientations, the
- smoother the spectrum
- longer the computation time



Pulse EPR, ENDOR: Orientation selection

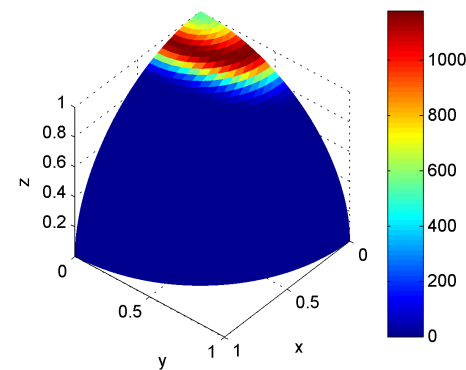
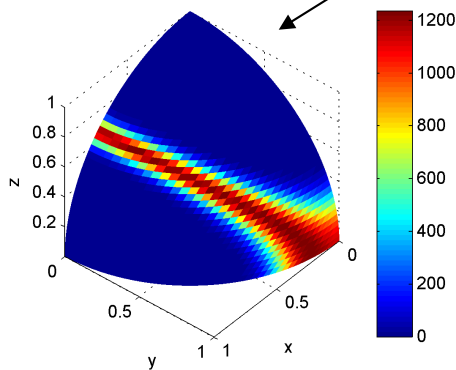
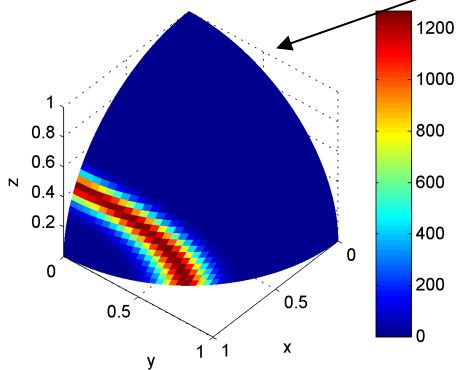
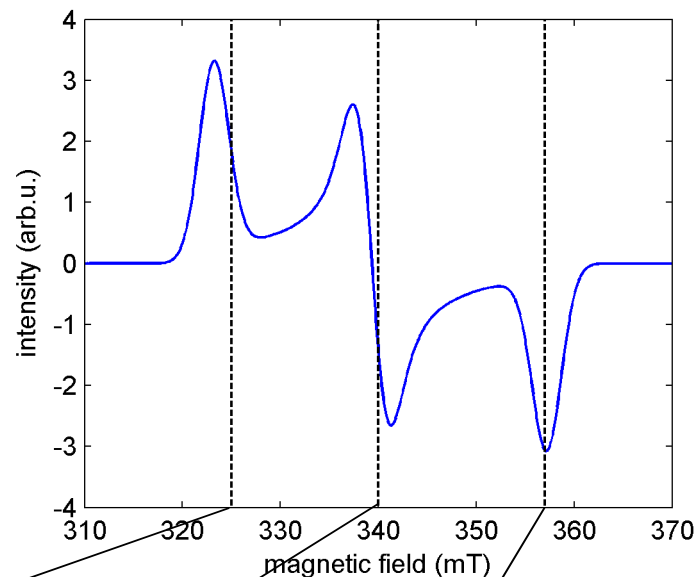
Orientation/transition selection

1. large anisotropies, wide EPR spectrum
2. limited excitation bandwidth of mw pulses

Parameters necessary

`Exp.ExciteWidth = 100;`
(Gaussian FWHM, in MHz)

`Exp.mwFreq = 9.5;`
(has to be given, in GHz)





Pulse EPR: $S > 1/2$

Set electron spin

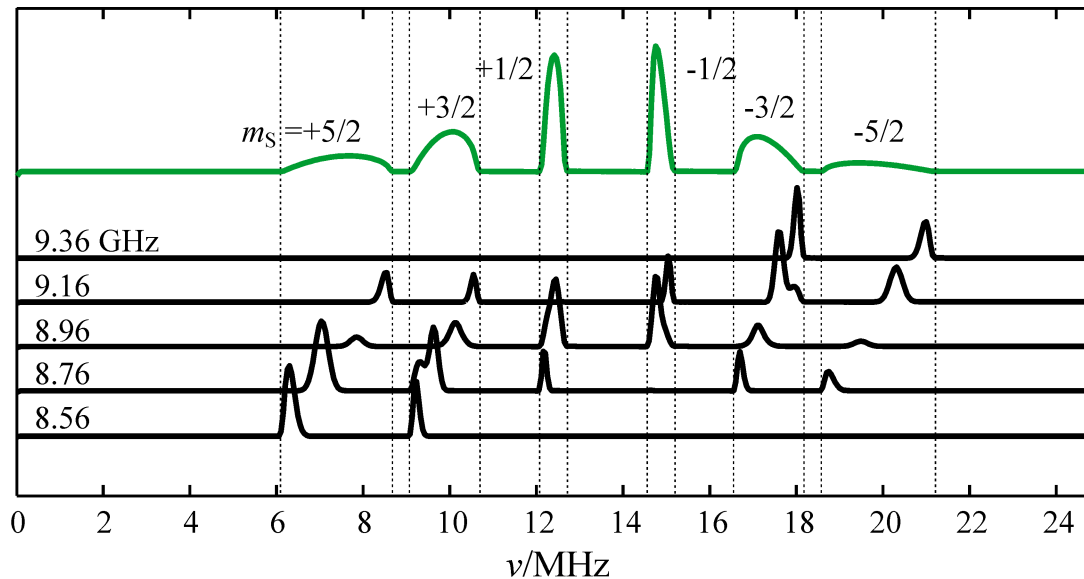
```
Sys.S = 1;  
Sys.S = 5/2;  
Sys.S = 7/2;
```

Set zero-field interaction

```
Sys.D = -200;  
Sys.D = [-130 -80 210];
```

What to remember

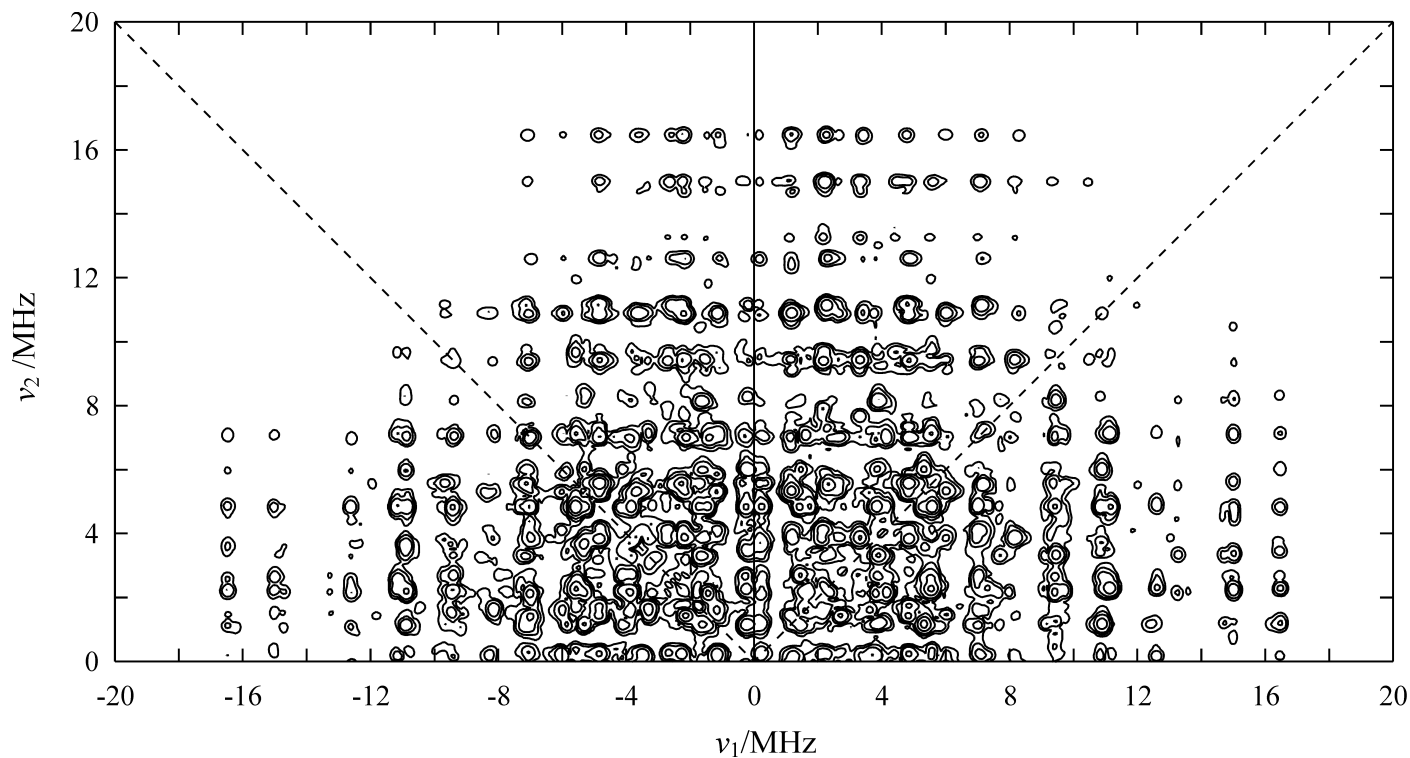
- flip angles are different for different transitions
- optimized echo amplitude might not correspond to 90° flips for any transition
- often strong orientation selection
- nuclear frequencies strongly depend on m_S





Pulse EPR: $S > 1/2$, many nuclei

Mn(II)(Im)₆ $S = 5/2$, $I = 5/2$, and six $I = 1$ (26244 spin states)



simulated single-crystal HYSCORE, ideal pulses (6 seconds)



Pulse EPR: User-defined pulse experiments

Fields needed for a user-defined experiment

`Exp.Flip` flip angle in multiples of 90°
`Exp.Inc` incrementation scheme
`Exp.t` initial delays, in microseconds

Example: Two-pulse ESEEM

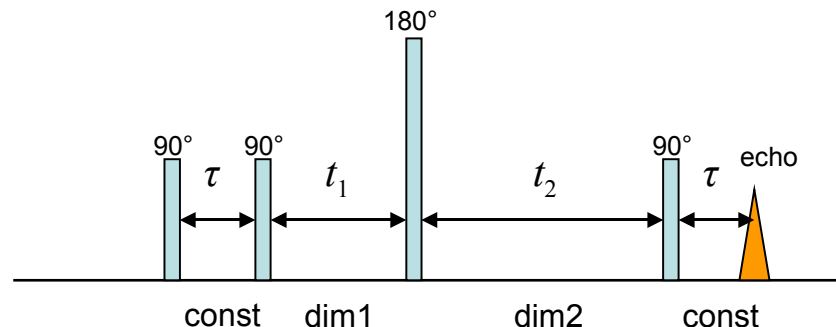
```
Exp.Flip = [1 2];  
Exp.Inc = [1 1];  
Exp.t = [0 0];
```

Example: 2D three-pulse ESEEM

```
Exp.Flip = [1 1 1];  
Exp.Inc = [1 2 1];  
Exp.t = [0.112 0 0.112];
```

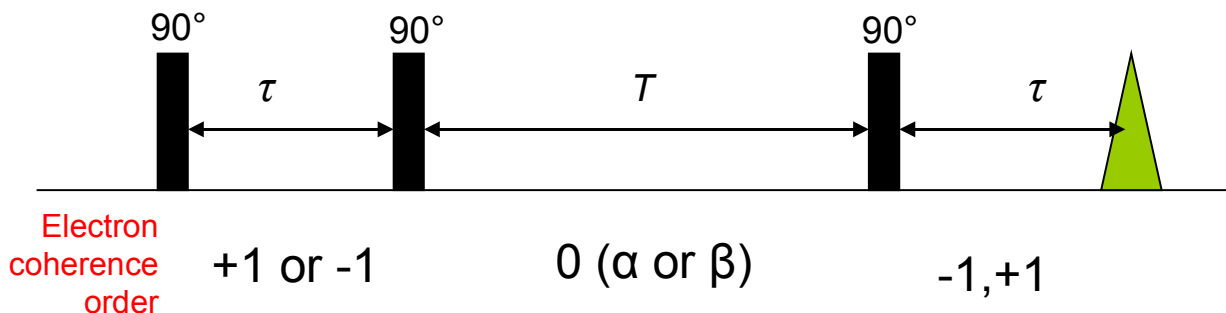
Example: HYSCORE

```
Exp.Flip = [1 1 2 1];  
Exp.Inc = [0 1 2 0];  
Exp.t = [0.1 0 0 0.1];
```





Pulse EPR: Transfer pathways and filters



Echo pathways:

($+\alpha-$), ($+\beta-$)
 ($-\alpha+$), ($-\beta+$)

Pathways with detectable echo

- equal times in + and in -
- ending in -

Examples:

two-pulse ESEEM ($+ -$)
 three-pulse ESEEM ($+\alpha-$), ($+\beta-$)
 HYSCORE ($+\alpha\beta-$), ($+\beta\alpha-$)
 DONUT-HYSCORE ($+\alpha\beta\alpha-$), ($+\beta\alpha\beta-$)
 refocused primary ($-+-$)

Pathway filter in EasySpin:

Exp.Filter one character per delay

'+' +1
 '-' -1
 'a' 0 (α)
 'b' 0 (β)
 '.' anything
 '0' 0 (α or β)
 '1' + or -

Pathway filter in EasySpin:

three-pulse ESEEM: `Exp.Filter = 'a.'`
 five-pulse ESEEM: `Exp.Filter = '+....'`;

`Exp.Filter = '.a.'`
`Exp.Filter = '+....'`;



Pulse EPR: Soft (real, non-ideal) pulses

Not possible with pre-defined sequences!

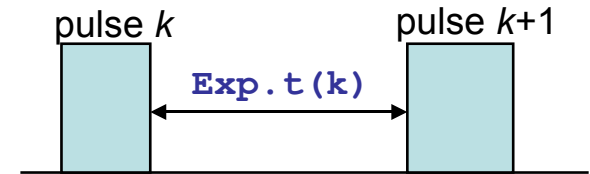
Soft pulses in user-defined sequences

Units: μs

`Exp.tp` length of pulses

Example: HYSORE

```
Exp.Flip = [1 1 2 1];  
Exp.Inc = [0 1 2 0];  
Exp.t = [100 0 0 100]*1e-3;  
Exp.tp = [10 10 20 10]*1e-3;
```



Delay values in `Exp.t`:
between end of one pulse
and beginning of next

Soft pulses: Offset integration needed

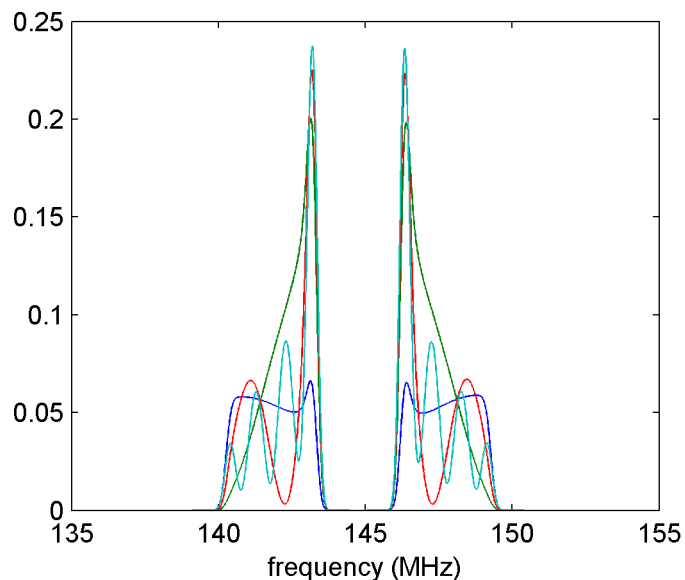
`Opt.lwOffset` width of offset distribution, in MHz
(about $1/tp$ of first pulse; typically 100 MHz)
`Opt.nOffsets` number of integration points (typically 10-30)



Pulse EPR: ENDOR

Specifying ENDOR

```
Exp.Sequence = 'MimsENDOR';  
Exp.Range = [135 155]; frequency range (in MHz)  
Exp.tau = 0.100;
```



What to think about

- **saffron** does not (yet) take hyperfine enhancement into account (in contrast to **salt**)
- ENDOR spectrum is simulated as sum over nuclei

ENDOR linewidth (in MHz) stick spectrum is convoluted with lineshape

```
Sys.lwEndor = 0.1;
```

Gaussian FWHM

```
Sys.lwEndor = [0.1 0.2];
```

[Gaussian FWHM, Lorentzian FWHM]



Pulse EPR: some options

Product rule

`Opt.ProductRule = 0;` product rule off (default)
`Opt.ProductRule = 1;` product rule on, faster for 3+ nuclei

HYSCORE log plot

`Opt.logplot = 0;` linear coloring of intensities (default)
`Opt.logplot = 1;` logarithmic coloring of intensities, gives more detail

Processing options

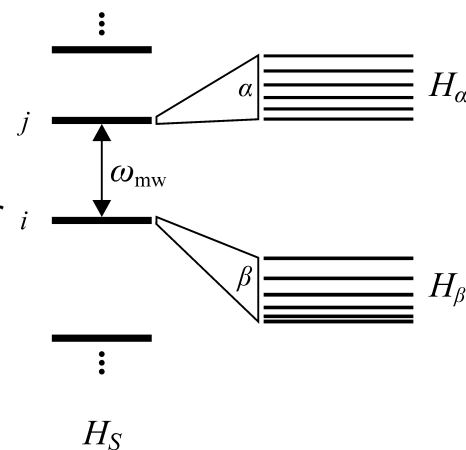
`Opt.Window = 'ham';` apodization window, applied before FFT
`Opt.ZeroFillFactor = 2;` factor for zero filling before FFT



Pulse EPR: theoretical background

1. Hamiltonians

- Set up Hamiltonian H_S for spins other than the ESEEM/ENDOR nuclei
- Compute energy levels and finds EPR transitions resonant with microwave frequency
- For each transition, compute nuclear sub-Hamiltonians for upper (α) and lower (β) manifold
- Compute energies and eigenstates for each nuclear sub-Hamiltonian



2. Pulse sequence

- Analyze the pulse sequence to determine pathways contributing to the echo

3. Peaks

- Use a submatrix formalism to compute the peak amplitudes and frequencies from the nuclear sub-Hamiltonians for each EPR transition and pathway
- Can use product rule to speed up calculations with many nuclei

4. Spectrum

- Construct a high-resolution stick spectrum
- Compute the time domain signal by inverse Fourier transform

For a powder, run this for a set of orientations.



Rotations, orientations and vectors

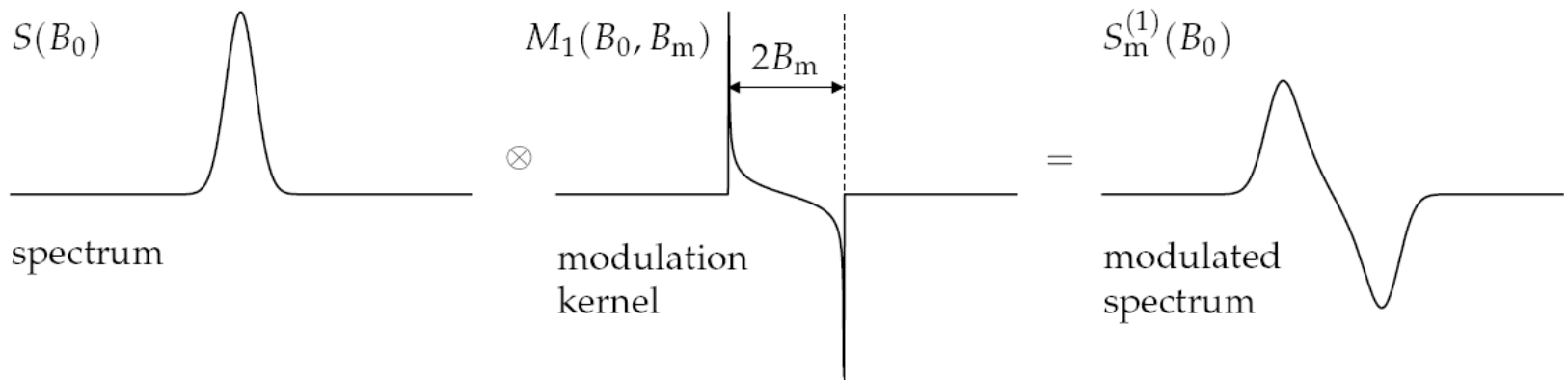
EasySpin functions for dealing with angles, orientations and rotations

<code>degree</code>	for angle unit conversion, $\text{pi} = 180 * \text{degree}$
<code>vec2ang</code> <code>ang2vec</code>	convert vector(s) to polar angles convert polar angles to vector(s)
<code>erot</code> <code>eulang</code>	computer rotation matrix from Euler angles computer Euler angles from rotation matrix
<code>rotaxi2mat</code> <code>rotmat2axi</code>	convert rotation axis plus angle to rotation matrix convert rotation matrix to rotation axis plus angle
<code>rotplane</code>	generate vectors in a plane
<code>sphgrid</code> <code>sphrand</code>	generate uniformly distributed orientations generate randomly distributed orientations



Field modulation

Convolution of absorption spectrum with modulation kernel



```
Exp.Harmonic = 0;
```

```
[B,spc] = pepper(Sys,Exp);
```

```
spcmmod = fieldmod(B,spc,ppAmpl);
```



Natural constants

2006 CODATA recommended values in S.I. units

gfree	g value of the free electron <i>latest value:</i> 2.0023193043617(15)
bmagn	Bohr magneton
nmagn	Nuclear magneton
planck	Planck constant
amu	Atomic unit of mass
avogadro	Avogadro constant
bohrrad	Bohr radius
boltzm	Boltzmann constant
clight	Vacuum speed of light
echarge	Elementary electric charge
emass	Mass of electron
faraday	Faraday constant
hartree	Atomic unit of energy
molgas	Molar gas constant
nmass	Mass of neutron
pmass	Mass of proton
rydberg	Rydberg constant
angstrom	Molecular-scale length unit
barn	Conventional unit of nuclear quadupole moments



Units and conversions

EasySpin uses

mT	for all magnetic fields
mT	for convolutional line widths
GHz	for EPR spectrometer frequencies (microwave)
MHz	for all other frequencies (ENDOR)
MHz	for energy levels, nuclear frequencies, etc
MHz	for hyperfine and quadrupole tensors
MHz	for zero-field tensors etc
s	for correlation times
1/s	for diffusion tensors
Hz	for Heisenberg exchange frequency
rad	for angles (a full turn is 2π)

$$1 \text{ mT} \rightarrow 10 \text{ G}$$

$$1 \text{ mT} \rightarrow 28.0 \text{ MHz } (g = 2)$$

$$1 \text{ mT} \rightarrow 14.0 \text{ MHz } \times g$$

$$1 \text{ mT} \rightarrow 42.6 \text{ MHz } (^1\text{H})$$

$$1 \text{ mT} \rightarrow 6.54 \text{ MHz } (^2\text{H})$$

$$1 \text{ mT} \rightarrow 10.7 \text{ MHz } (^{13}\text{C})$$

$$1 \text{ mT} \rightarrow 3.08 \text{ MHz } (^{14}\text{N})$$

$$1 \text{ mT} \rightarrow 4.32 \text{ MHz } (^{15}\text{N})$$

$$10^{-4} \text{ cm}^{-1} \rightarrow 3.00 \text{ MHz}$$

Conversion functions: `mt2mhz`, `mhz2mt`, `larmorfrq`, `degree`



Spin operators

`sop` cartesian and ladder operators
`stev` higher-order operators

`sop(1/2, 'x')`

0	0.5
0.5	0

`sop([1/2 1/2], 'z+')`

0	0.5	0	0
0	0	0	0
0	0	0	-0.5
0	0	0	0

`[Sx, Sy, Sz] = sop(1, 'x', 'y', 'z')`

`Sx =`

0	0.7071	0
0.7071	0	0.7071
0	0.7071	0

`Sy =`

0	0 - 0.7071i	0
0 + 0.7071i	0	0 - 0.7071i
0	0 + 0.7071i	0

`Sz =`

1	0	0
0	0	0
0	0	-1



Angular momenta

EasySpin includes basic support for general angular momenta computations

`clebschgordan`

Clebsch-Gordan coefficients $\langle j_1 j_2 m_1 m_2 | j_1 j_2 j m \rangle$

`wigner3j`

Wigner 3- j symbols $\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$

`wigner6j`

Wigner 6- j symbols $\begin{Bmatrix} j_1 & j_2 & j_3 \\ j_4 & j_5 & j_6 \end{Bmatrix}$

`plegendre`

Associated Legendre polynomials $P_l^m(x)$

`spherharm`

Spherical harmonics $Y_l^m(\theta, \phi)$



Line shapes

Gaussian

```
y = gaussian(B,B0,FWHM,Derivative)
```

Lorentzian

```
y = lorentzian(B,B0,FWHM,Derivative)
```

Voigtian (= convolution of Gaussian and Lorentzian)

```
y = voigtian(B,B0,[FWHM_Gau FWHM_Lor],Derivative)
```

Pseudo-Voigtian (= weighted sum of Gaussian and Lorentzian)

```
y = lshape(B,B0,FWHM,Derivative,weightGau)
```