



EasySpin

EasySpin: Software for Electron Paramagnetic Resonance

Stefan Stoll

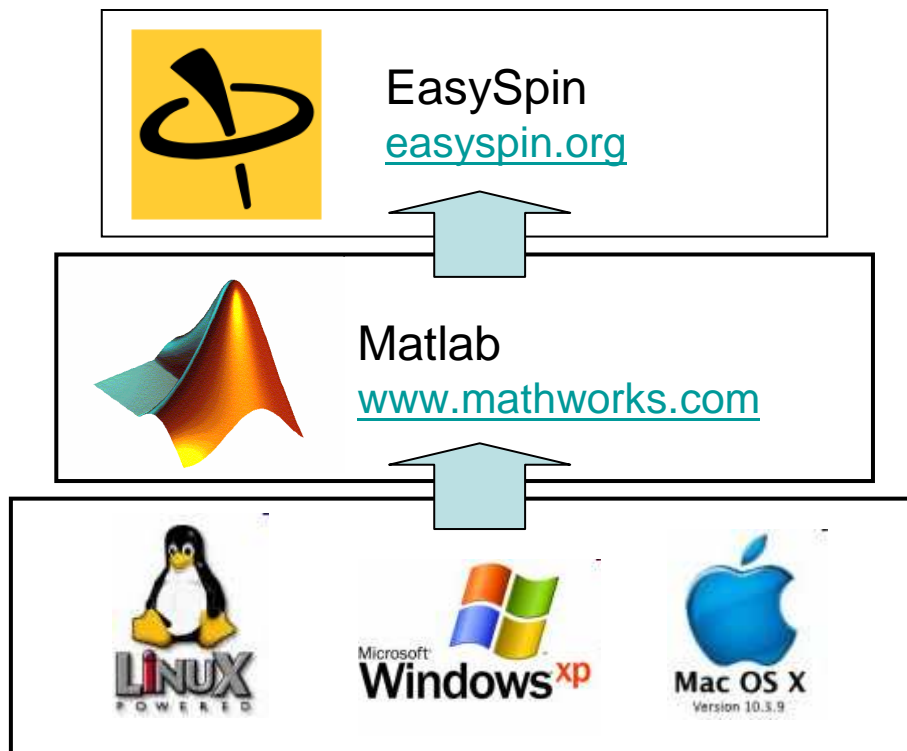
Department of Chemistry
University of California, Davis

based on
EasySpin 3.0 pre-release for EPR summer school
St. Andrews, Scotland, August 2008



Overview

EasySpin is a Matlab toolbox for spectral simulations and data analysis in EPR spectroscopy



Features

- ▶ spectral simulations
 - isotropic cw EPR spectra
 - fast-motion cw EPR spectra
 - slow-motion cw EPR spectra
 - solid-state cw EPR, ENDOR spectra
 - pulse EPR spectra
- ▶ least-squares fitting
- ▶ loading and saving of data
- ▶ rotations, Euler angles
- ▶ line shapes
- ▶ spin operators and matrices
- ▶ data processing
 - RC filtering, denoising
 - pseudomodulation
 - apodization windows
 - baseline correction



Help and 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**



Matlab: Desktop

The screenshot shows the MATLAB 7.3.0 (R2006b) desktop environment. The interface includes a menu bar (File, Edit, Debug, Desktop, Window, Help), a toolbar, and several windows:

- Workspace:** A table showing variables and their values. A callout labeled "workspace" points to this window.
- Editor:** A window showing the code for `resonatorcoupling.m`. A callout labeled "editor" points to this window.
- Figures:** A window titled "Figures - Figure 2" displaying a plot of the magnitude of the transfer function $|G|$ versus frequency ω . The plot shows three curves: "undercoupled" (blue), "critically coupled" (green), and "overcoupled" (red). A callout labeled "figures" points to this window.
- Command Window:** A window showing the execution of the command `nucgval('1H,13C,31P,15N')` and its output. A callout labeled "command window" points to this window.

The MATLAB Start menu is visible at the bottom left, with a callout labeled "start menu".

Name	Value	Min	Max
Exp	<1x1 struct>		
G	<3x301 double>	0	-0.994 + 0.0996i
Opt	<1x1 struct>		
Output	'asdfsdf'		
a	<500x1 double>	0.0069	0.9968
	[5.5857 1.4048 2.2632 -0.5664]	-0.5664	5.5857
	3	3	3
	[0.2 1 3]	0.2	3
nSys	<1x1 struct>		
spc	<1x1024 double>	0.0001	0.9991
xi	<1x301 double>	-20	20

```
1 % tuning picture of a cw EPR spectrometer
2
3 % coupling coefficient
4 % (1 critical coupling, <1 undercoupled, >1 overcoupled)
5 beta = [0.2 1 3];
6
7 % frequency offset
8 = linspace(-1,1,301)*20;
9
10 b = 1:numel(beta)
11 G(b,:) = (1-beta(b)+1i*xi) ./ (1+beta(b)-1i*xi);
12 end
13
14 plot(xi,abs(G));
15 legend('undercoupled','critically coupled','overcoupled');
16
```

```
>> nucgval('1H,13C,31P,15N')
ans =
    5.5857    1.4048    2.2632   -0.5664
>>
```



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;`

delete 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 = 'Jake''s Crawfish';
```

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 0 0];  
Software.Price = 0;  
Software.Downloads = 800;  
Software.Users = 'many';  
Software.Developers = 1;  
Software.Functions = 97;  
Software.Bugs = NaN;  
Software.Tests = 286;  
Software.References = 111;
```

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];  
  
Experiment = struct('mwFreq',...  
    94.9, '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 R2008a, EasySpin 3.0 (summer school)
- EasySpin supports all Matlab versions starting from 6.5 (R13, from 2003)
- Matlab: new version every half year (e.g. R2008a, R2008b)
- EasySpin: new version every year, with bug fix releases in between

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-3.0.0</i>
toolbox functions	<i><mydir>/easyspin-3.0.0/easyspin</i>
documentation	<i><mydir>/easyspin-3.0.0/documentation</i>
examples	<i><mydir>/easyspin-3.0.0/examples</i>

- Launch Matlab
- Go to *File > Set path ...*
- Remove any old EasySpin entries from Matlab path
- Add the folder *<mydir>/easyspin-3.0.0/easyspin* to Matlab path
- Save and close the Set path window

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



EasySpin: Files

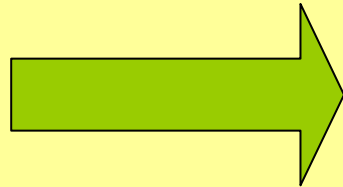
Independent of Matlab version and operating system:

<code>easyspin/*.m</code>	Matlab source and help files
<code>easyspin/*.p</code>	Matlab prepared files
<code>easyspin/private/*.c</code>	C source files

Dependent on Matlab version and on operating system:

`easyspin/private/*.mex*` mex file (C file compiled for Matlab)
- generated initially by **easyspininfo**
- to explicitly recompile, type **easyspincompile**

`afunction.c`



<code>afunction.dll</code>	Windows (old)
<code>afunction.mexw32</code>	Windows, 32bit
<code>afunction.mexw64</code>	Windows, 64bit
<code>afunction.mexglx</code>	Linux, 32bit
<code>afunction.mexa64</code>	Linux, 64bit
<code>afunction.mexmac</code>	Mac



Loading and saving data

Loading and importing

<code>eprload</code>	reads data from Bruker spectrometers ESP: .spc/.par, BES3T: .DTA/.DSC also loads experimental parameters
<code>load</code>	load data stored in Matlab format (.mat)
<code>textread</code>	reads text (ASCII) data
<code>xlsread</code>	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

<code>save</code>	saves data in Matlab format (.mat)
<code>save -ASCII</code>	saves text (ASCII) data
<code>xlswrite</code>	saves as Excel spreadsheet (.xls)

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



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)
```

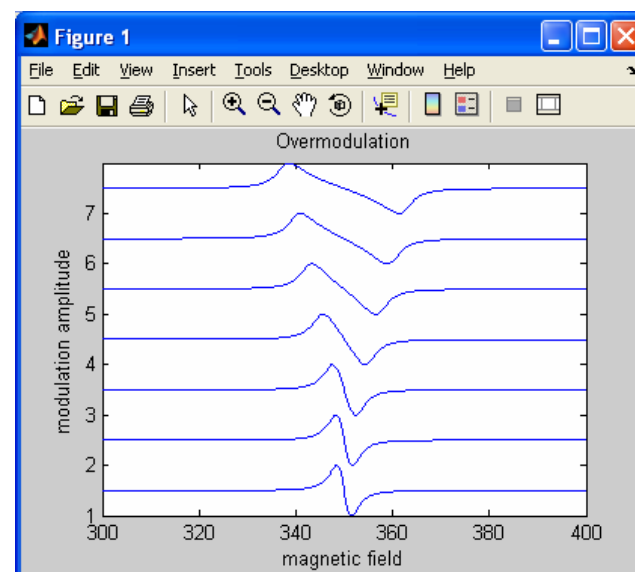
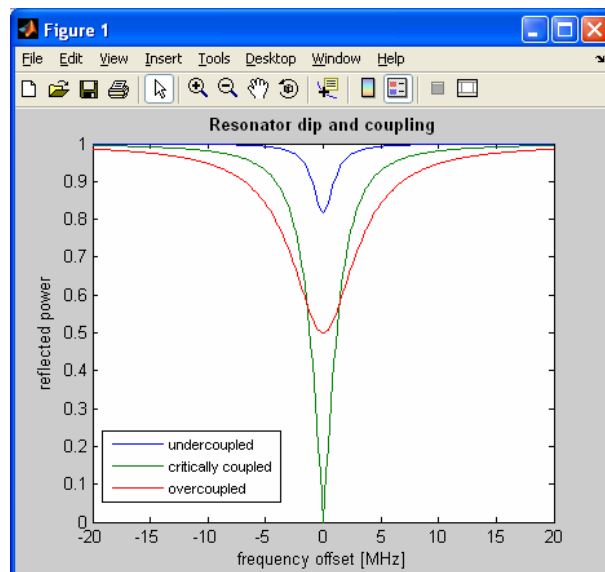
```
stackplot(x,y)
```

Changing the appearance of plots

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

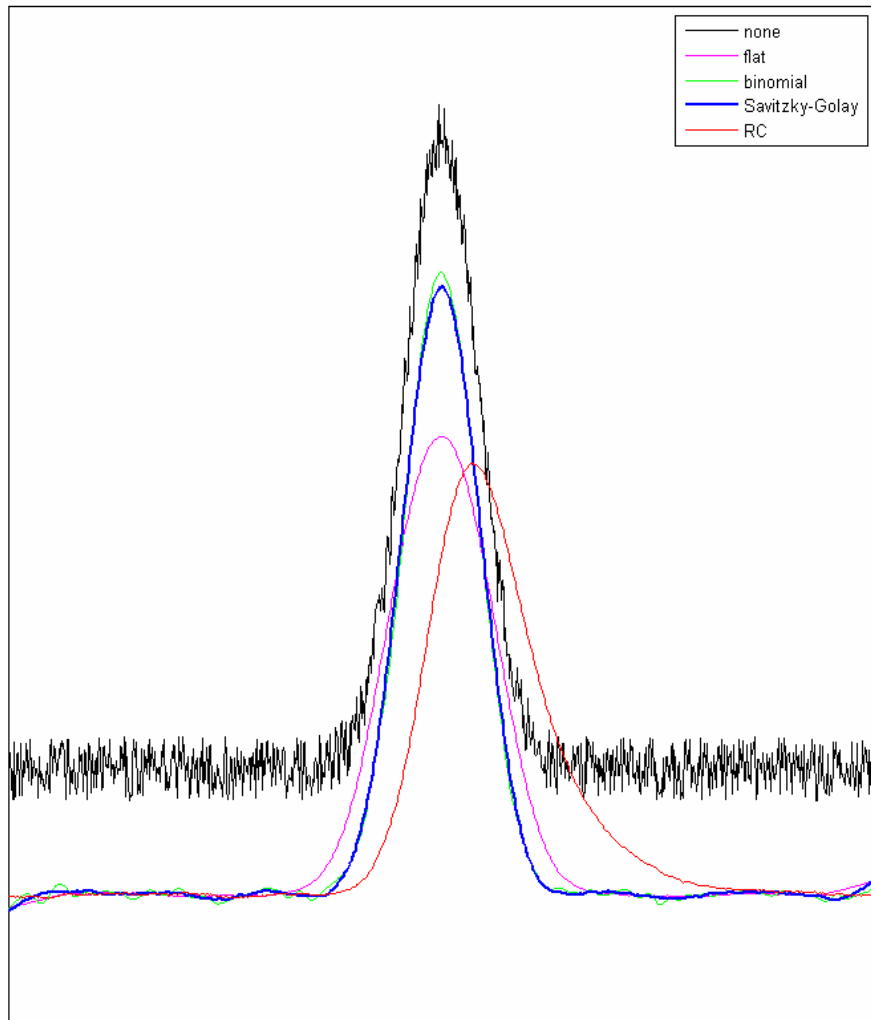
```
hold on
hold off
```

```
axis tight
```





Smoothing of noisy data



`rcfilt(y, SampTime, TimeConst)`

- exponential moving average
- distorts shape, shifts and broadens peak
- use only to mimic time constant in spectrometers

`smooth(B, spc, n)`

- binomial or flat moving average
- Savitzky-Golay (recommended: no peak shift, no line distortion, derivatives built-in)

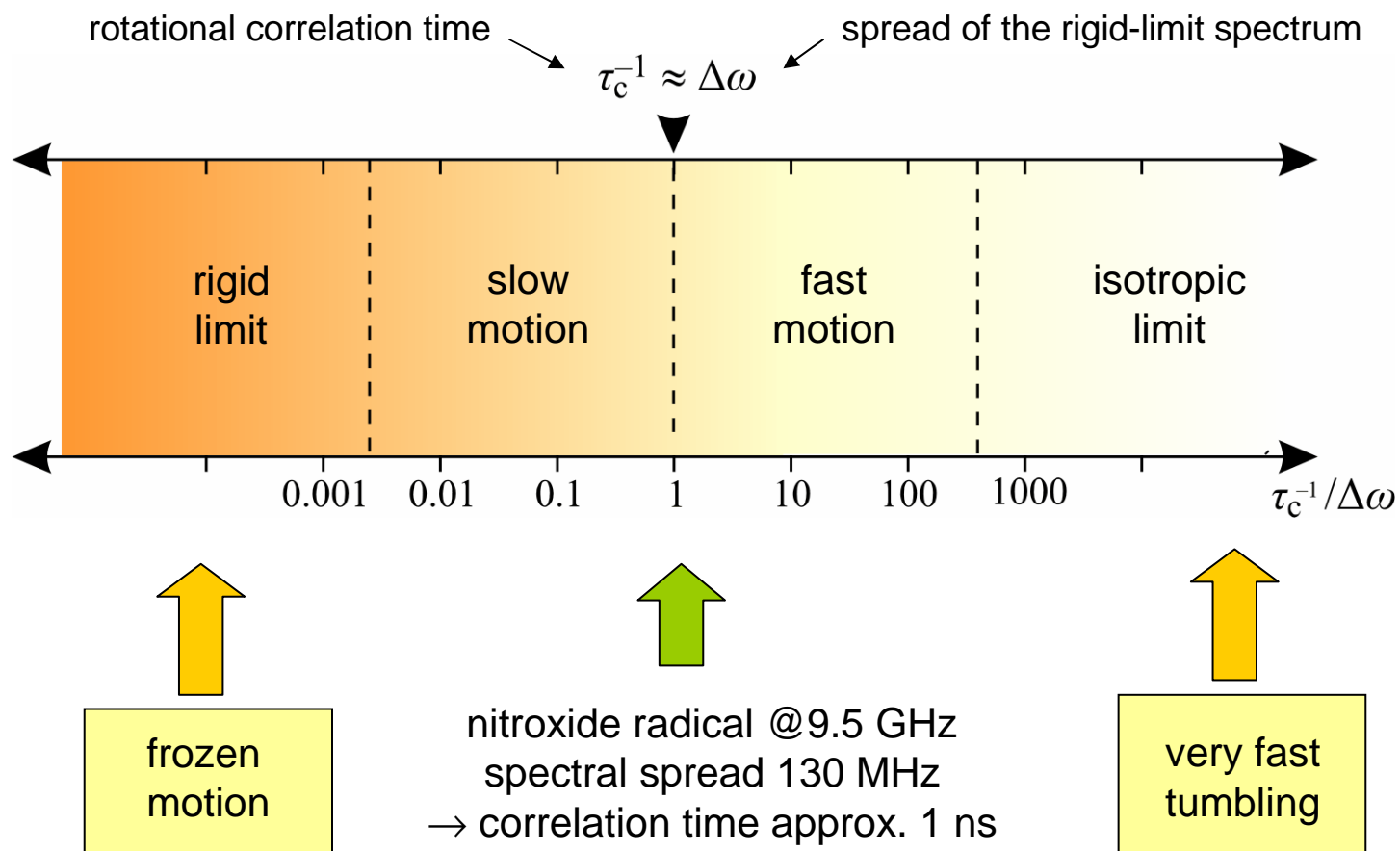
`addnoise(spc, SNR, NoiseType)`

- adds noise to spc so that the final signal-to-noise is SNR.
- noise types: uniform and Gaussian

"I never make simulations without noise."
(G. Denninger)

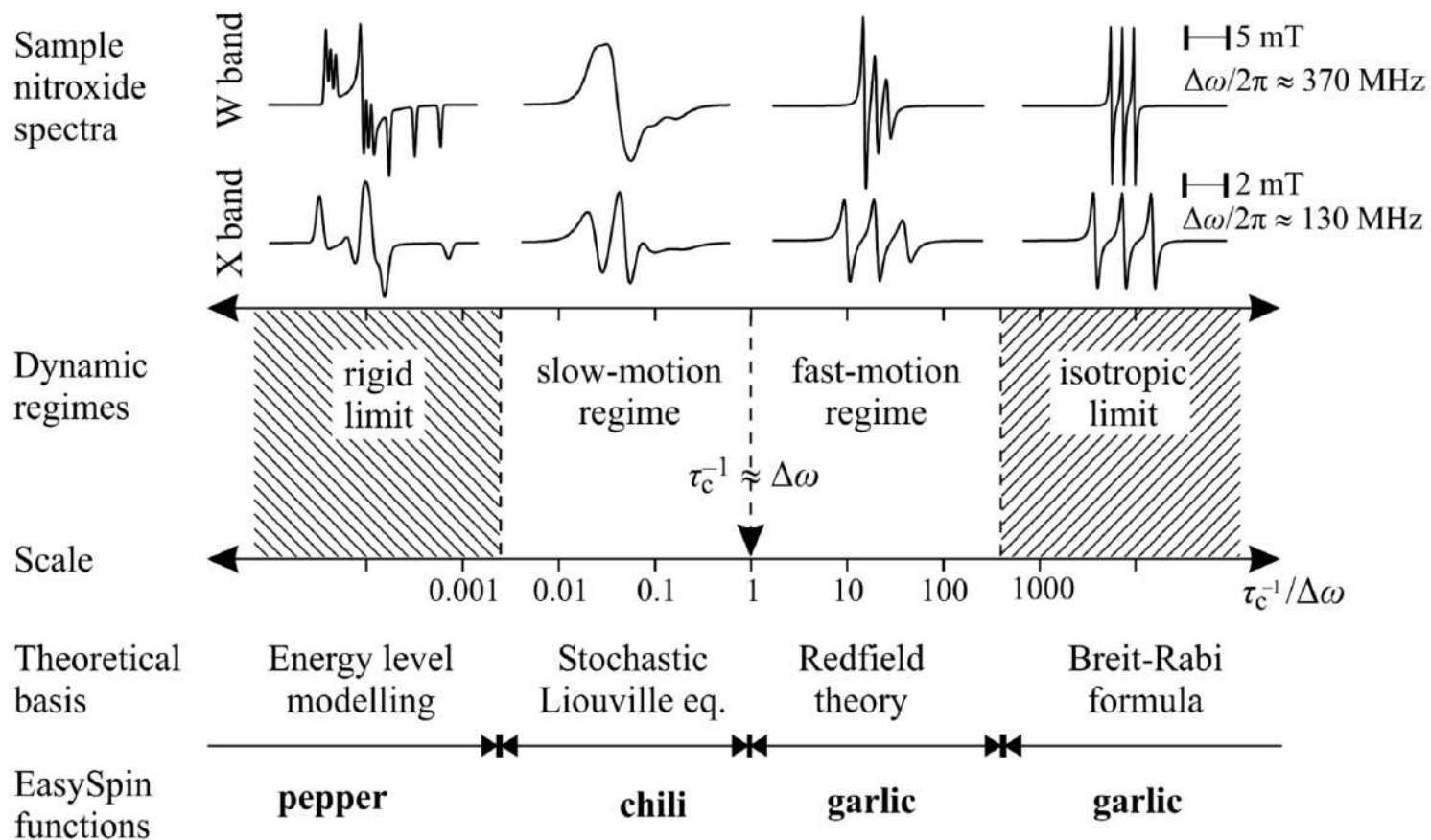


Motional regimes





Motional regimes





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)

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
`[B , spc] = pepper (...)` returns field and spectrum (arrays)



Electron spins

Input syntax

`Sys.S = 1/2`

one electron spin $S = 1/2$
(default, no need to specify explicitly)

`Sys.S = 5/2`

one electron spin $S = 5/2$

`Sys.S = [1/2, 1/2]`

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 $S = 1/2$



g tensors

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

g values, one electron spin

`Sys.g = 2`

`Sys.g = [1.9, 2.2]`

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

isotropic g , = [2 2 2]

axial g , = [1.9, 1.9, 2.2]

rhombic g

g values, two electron spins

`Sys.g = [2.03; 2.01]`

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

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

one row per electron spin

isotropic g

axial g

rhombic g

Tensor orientation

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

Units: radians

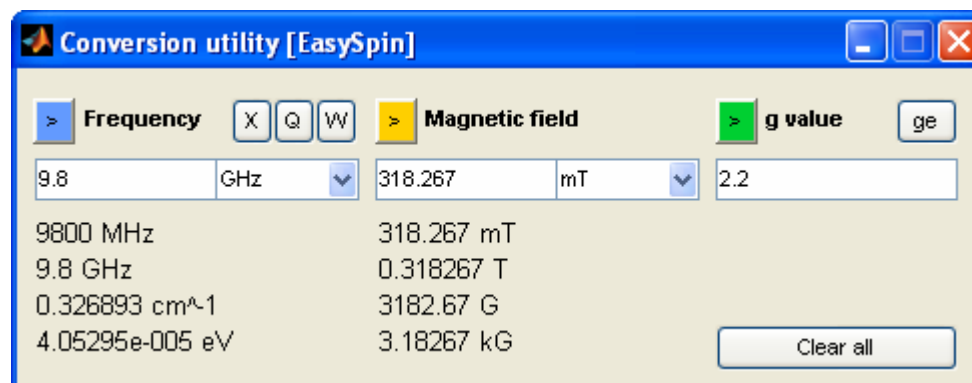
one row per electron spin

orientation of g tensor frame in molecular frame

Frequency/field/g conversions

$$\nu/\text{GHz} = 13.996 g B_0/T$$

or use `eprconvert`





Zero-field splittings

Hamiltonian (in frequency units)

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

EasySpin input

Units: MHz

1 cm⁻¹ = 30 GHz

`Sys.D = 100`

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

`Sys.D = [100 10]`

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

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

$[D_x, D_y, D_z]$ in MHz

Tensor orientation

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

Units: radians



Magnetic nuclei

Specifying nuclear isotopes

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

comma-separated list of isotopes

Equivalent nuclei (only for `garlic`)

```
sys.Nucs = '1H,13C';  
sys.n = [3 1];
```

Helper functions

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



Magnetic nuclei

Other functions

- nucabund** natural abundance
- nucgval** nuclear g factors
- nucqmom** nuclear electric quadrupole moments
- nucspin** nuclear spin quantum numbers
- larmorfrq** Larmor frequency
- isotopes** interactive PSE with nuclear isotope database

The screenshot shows the 'Nuclear spins' application window. It features a periodic table where elements are color-coded: blue for H, Li, Na, K, Rb, Cs, Fr, Ra; yellow for B, C, N, O, F, Ne, Al, Si, P, S, Cl, Ar, Ga, Ge, As, Se, Br, Kr, In, Sn, Sb, Te, I, Xe, Tl, Pb, Bi, Po, At, Rn; red for Sc, Ti, V, Cr, Mn, Fe, Co, Ni, Cu, Zn, Y, Zr, Nb, Mo, Tc, Ru, Rh, Pd, Ag, Cd, Lu, Hf, Ta, W, Re, Os, Ir, Pt, Au, Hg; green for La, Ce, Pr, Nd, Pm, Sm, Eu, Gd, Tb, Dy, Ho, Er, Tm, Yb; and white for all other elements. Below the table is a data table for Sn isotopes:

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, there is a text input field for 'Magnetic field [mT]' with the value '350' entered.



Magnetic nuclei

new in 3.0

Natural-abundance mixtures: omit mass number

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

69% ⁶³Cu, 31% ⁶⁵Cu

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

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

53% ⁶³Cu/³⁵Cl, 23% ⁶⁵Cu/³⁵Cl, 17% ⁶³Cu/³⁷Cl, etc

```
Sys.Nucs = 'C';
```

98.9% ¹²C, 1.1% ¹³C

"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)

$$S A I = S_x A_{xx} I_x + S_y A_{yy} I_y + S_z A_{zz} I_z$$

Units: MHz

Tensor principal values

one row per nucleus

`Sys.Nucs = '1H'`

`Sys.A = 10`

`Sys.A = [4, 12]`

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

one nuclear spin

isotropic A , = [10 10 10]

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

rhombic A

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

`Sys.A = [10, 2]`

`Sys.A = [4 12; 1 3]`

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

two nuclear spins

two isotropic A

two axial A

two rhombic A

1 mT = 28 MHz

$10^{-4} \text{ cm}^{-1} = 3 \text{ MHz}$

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

`Sys.Q = 0.7;`

$e^2 q Q / h = 0.7$ MHz

`Sys.Q = [0.7, 0.1]`

$e^2 q Q / h = 0.7$ MHz, and $\eta = 0.1$ (between 0 and 1/3)

`Sys.Q = [-1.2, -0.8, 2]`

3 principal values (in MHz)

Tensor orientation

one row per nucleus

Units: radians

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

Euler angles of Q eigenframe in molecular frame

Q averages out in isotropic limit, used in fast-motion regime and rigid limit, neglected in slow-motion regime

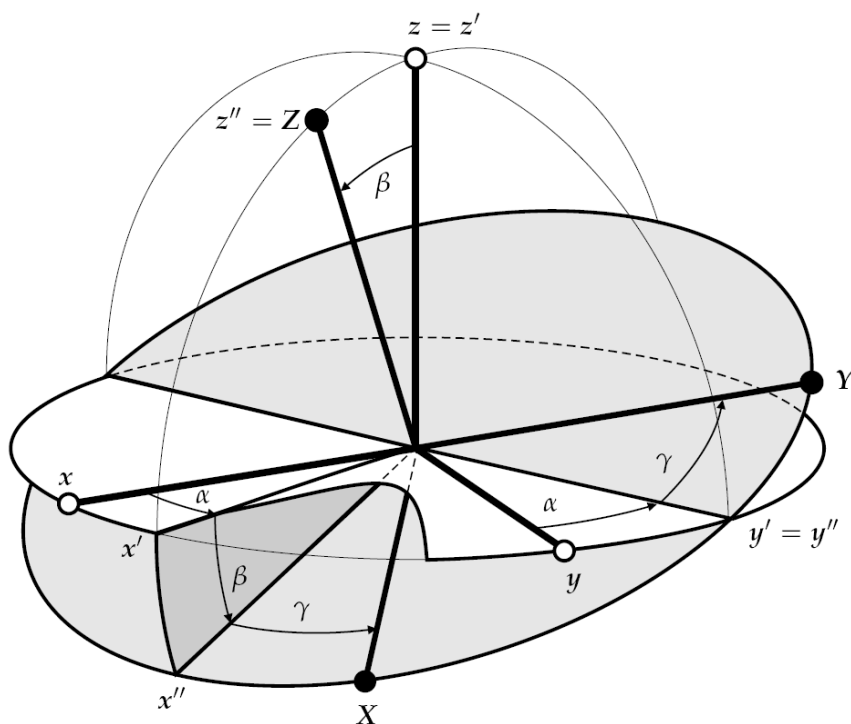


Orientations in EasySpin

Relative orientation between two coordinate systems xyz and XYZ described a 3x3 rotation matrix which can be parameterized by three Euler angles.

$xyz \longrightarrow XYZ$

$$\mathbf{\Omega} \equiv (\alpha, \beta, \gamma)$$



see EasySpin documentation

Rotation matrix $zy'z''$ convention, passive rotation

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

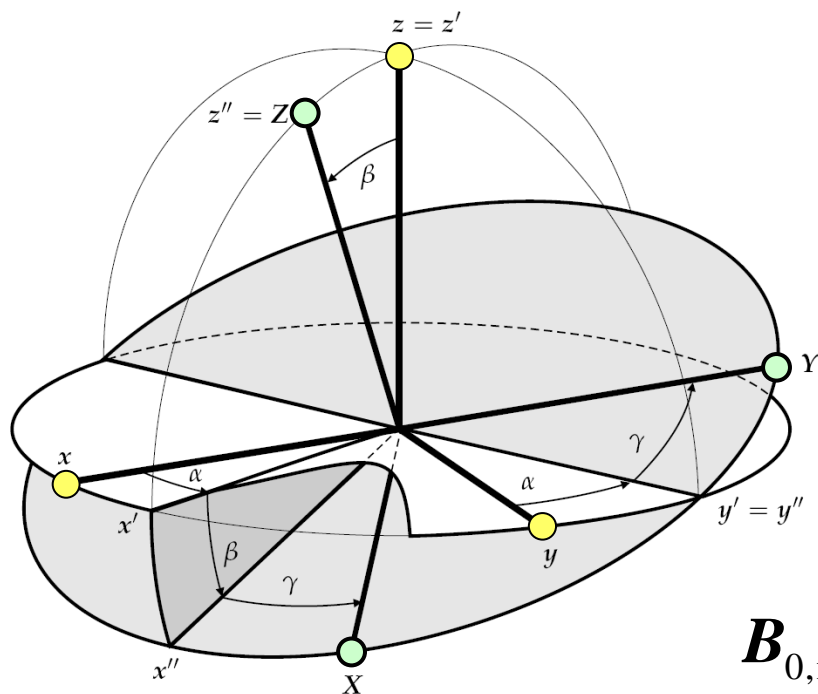
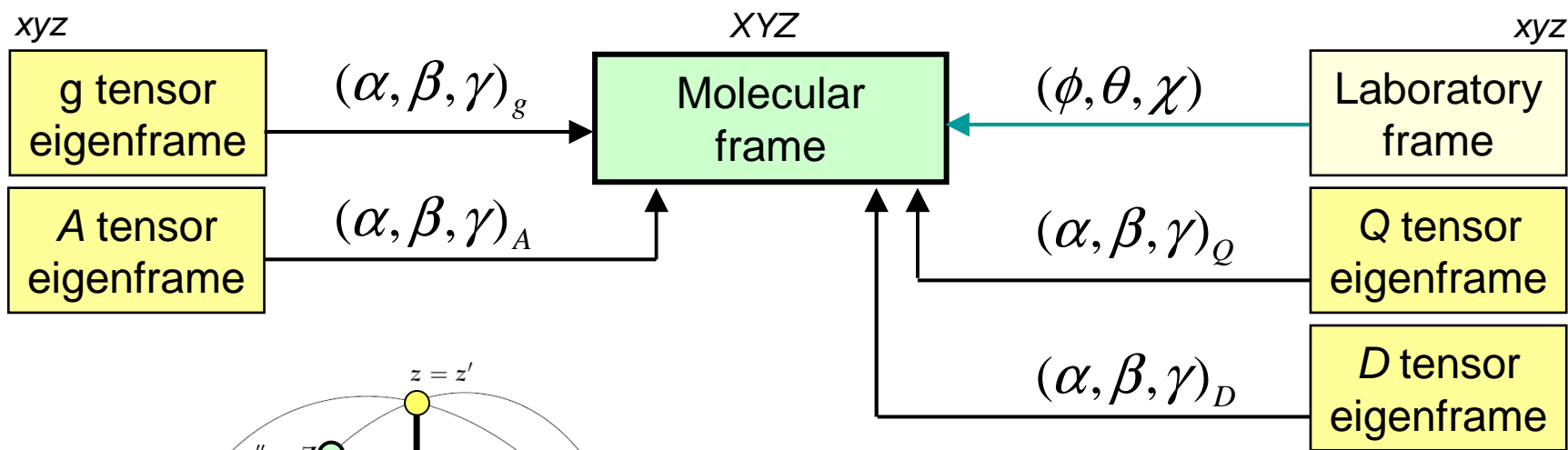
A passive rotation does not change the orientation of the object (vector, tensor), but the orientation of the coordinate system it is represented in.

$$v_{XYZ} = R(\mathbf{\Omega}) v_{xyz}$$

$$T_{XYZ} = R(\mathbf{\Omega}) T_{xyz} R^T(\mathbf{\Omega})$$



Orientations in EasySpin



$$\mathbf{g}_{\text{mol}} = R(\boldsymbol{\Omega}) \mathbf{g}_{\text{diag}} R^T(\boldsymbol{\Omega})$$

$$\mathbf{B}_{0,\text{mol}} = R(\boldsymbol{\Omega}) \mathbf{B}_{0,\text{lab}} = R(\boldsymbol{\Omega}) (0, 0, B_0)^T$$



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`)

Strain broadenings (only `pepper`)

isotropic or anisotropic

Units: MHz

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

<code>Sys.HStrain</code>	unresolved hyperfine splittings (MHz)
<code>Sys.gStrain</code>	distribution of g values
<code>Sys.AStrain</code>	distribution of A values (correlated with g) (MHz)
<code>Sys.DEstain</code>	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



Experimental settings

Spectrometer frequency (in GHz)

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

Number of points default 1024

```
Exp.nPoints = 2001;
```

Sample temperature (in K)

```
Exp.Temperature = 77;
```

omit if possible, as it can slow down simulations considerably

Detection harmonic default 1

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

```
Exp.Harmonic = 1;
```

```
Exp.Harmonic = 2;
```

Field range (in mT)

a) automatic determination

no field necessary, works only for systems with $S=1/2$ and small hyperfine couplings

b) start and end field (in mT)

```
Exp.Range = [320 380];
```

c) center field and sweep width (in mT)

```
Exp.CenterSweep = [350 60];
```

Detection mode default perpendicular

```
Exp.Detection = 'perpendicular';
```

```
Exp.Detection = 'parallel';
```



Static orientational distributions

Orientation of a paramagnet in the sample:

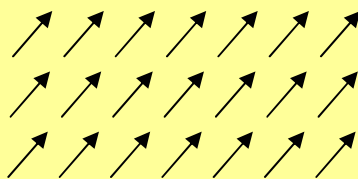
$$\Omega = (\phi, \theta, \chi)$$

Distribution of orientations of the paramagnets:

$$P(\Omega)$$

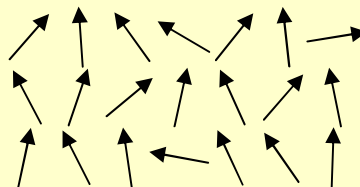
Crystals

one or several distinct orientations with equal probabilities



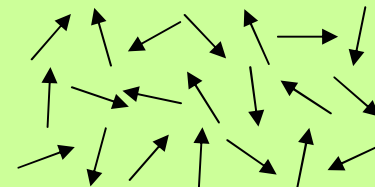
Partially oriented systems

probability is varying function of orientation



Powders

all orientations with equal probability



Exp.Orientations
Exp.CrystalSymmetry

Exp.Ordering
partial MOMD

default
MOMD



Single crystals

new in 3.0

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

new in 3.0

Crystal rotations

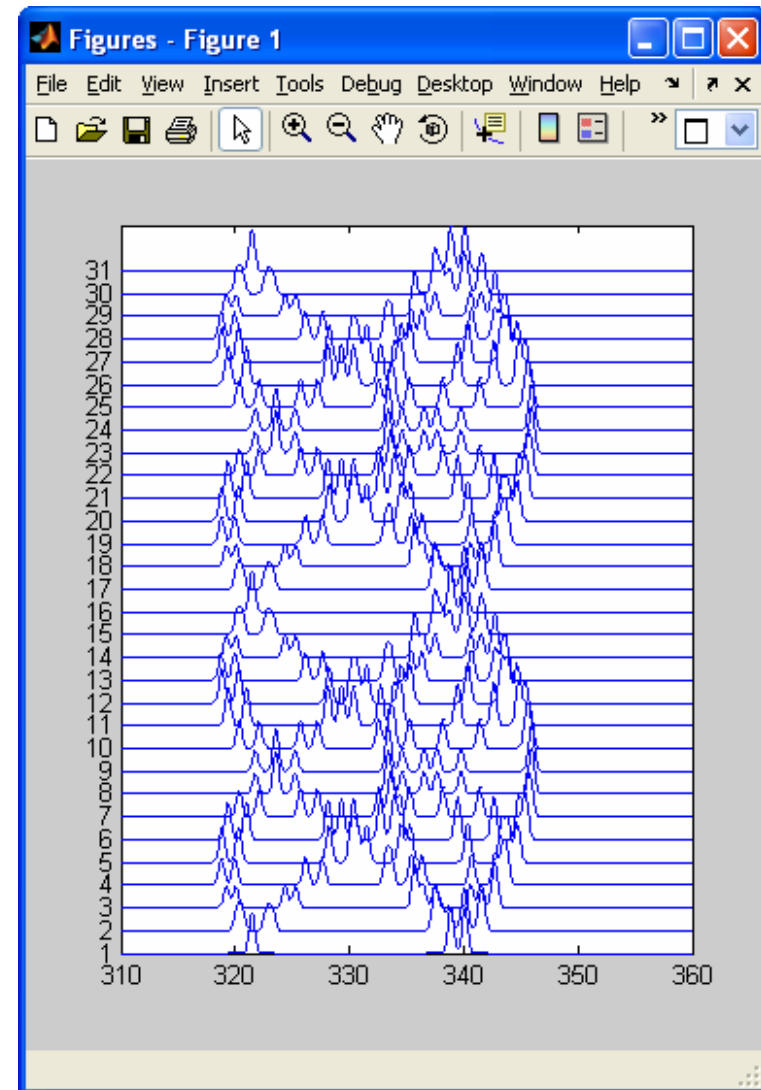
- specify rotation axis
- generate rotated orientations
- specify crystal symmetry

```
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);
```

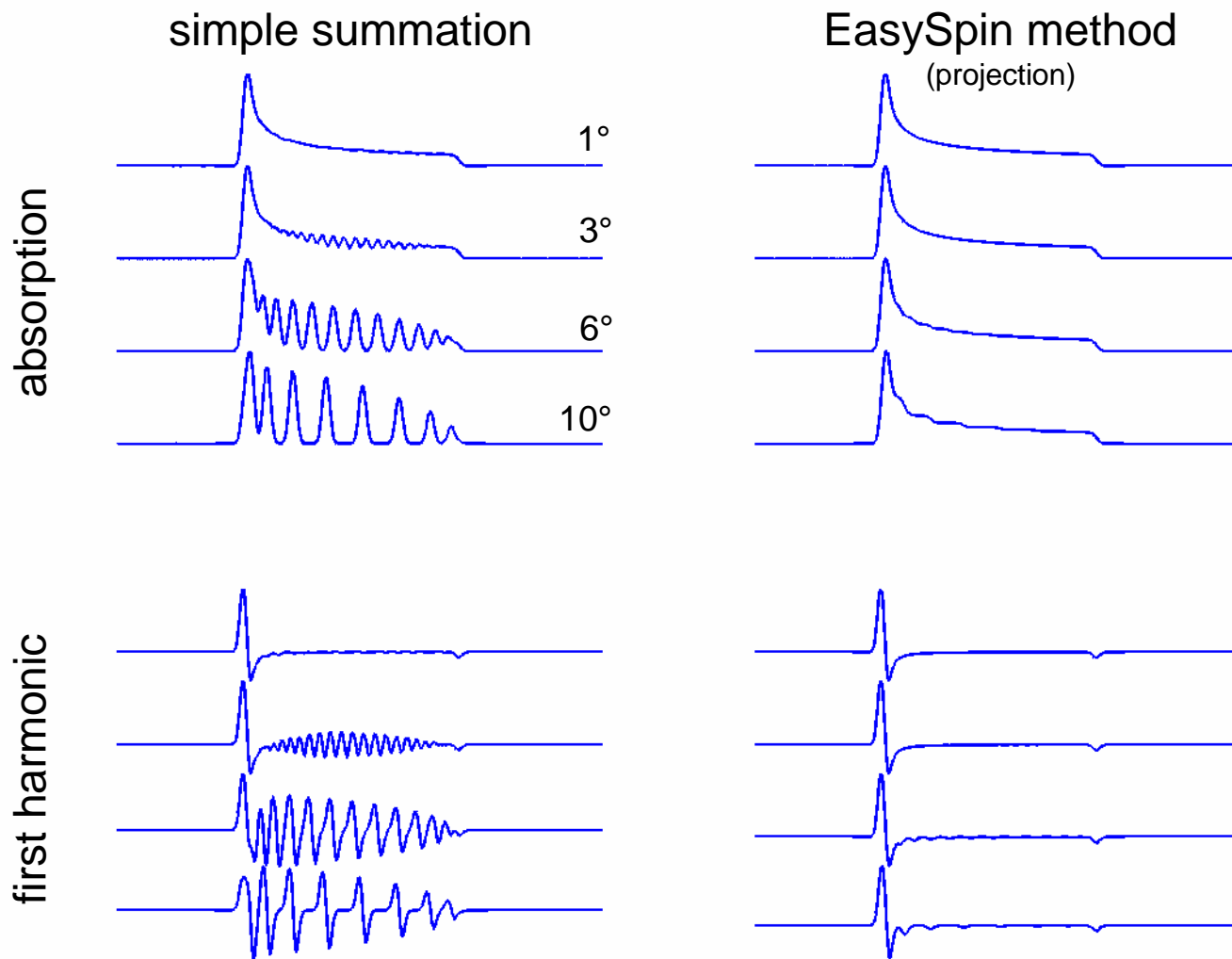
```
stackplot(B,spc);
```





Powder simulations: orientational averaging

"Simulation grass"





Partially oriented systems

Orientational distribution

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

Orientational potential

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

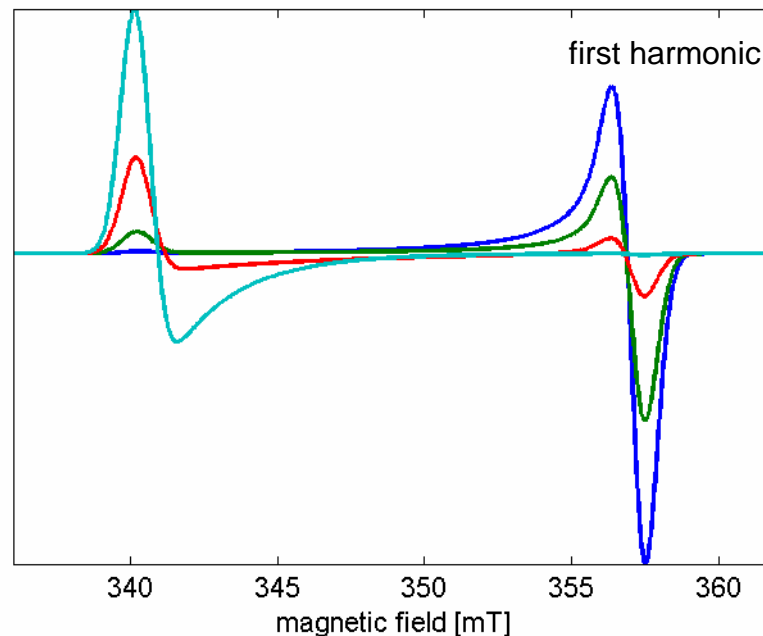
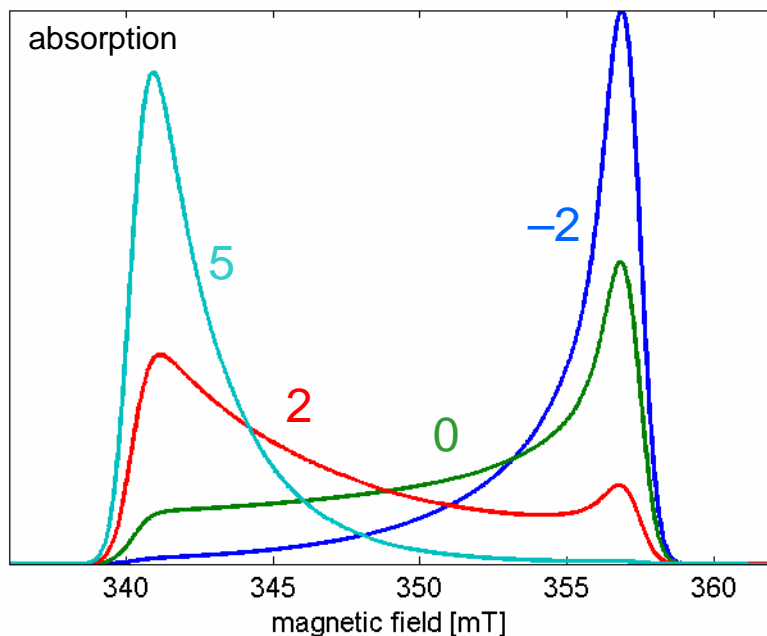
Predefined potential in EasySpin:

`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;`





Simulation algorithms

pepper: solid-state cw EPR spectra

- full matrix diagonalization
- matrix diagonalization + first-order perturbation for small splittings
- second-order perturbation theory (approximate)

new in 3.0

garlic: liquid/fast-motion cw EPR

- Breit-Rabi equations (exact solution) for line positions
- Redfield theory for fast motion regime line widths

chili: slow-motion cw EPR

- stochastic Liouville equation (Freed)

salt: solid-state ENDOR

- full matrix diagonalization

saffron: solid-state pulse EPR

- matrix diagonalization of nuclear Hamiltonians in rotating frame (high-field approximation for electron spins)

new in 3.0



Matrix diagonalization vs. perturbation theory

new in 3.0

Matrix diagonalization

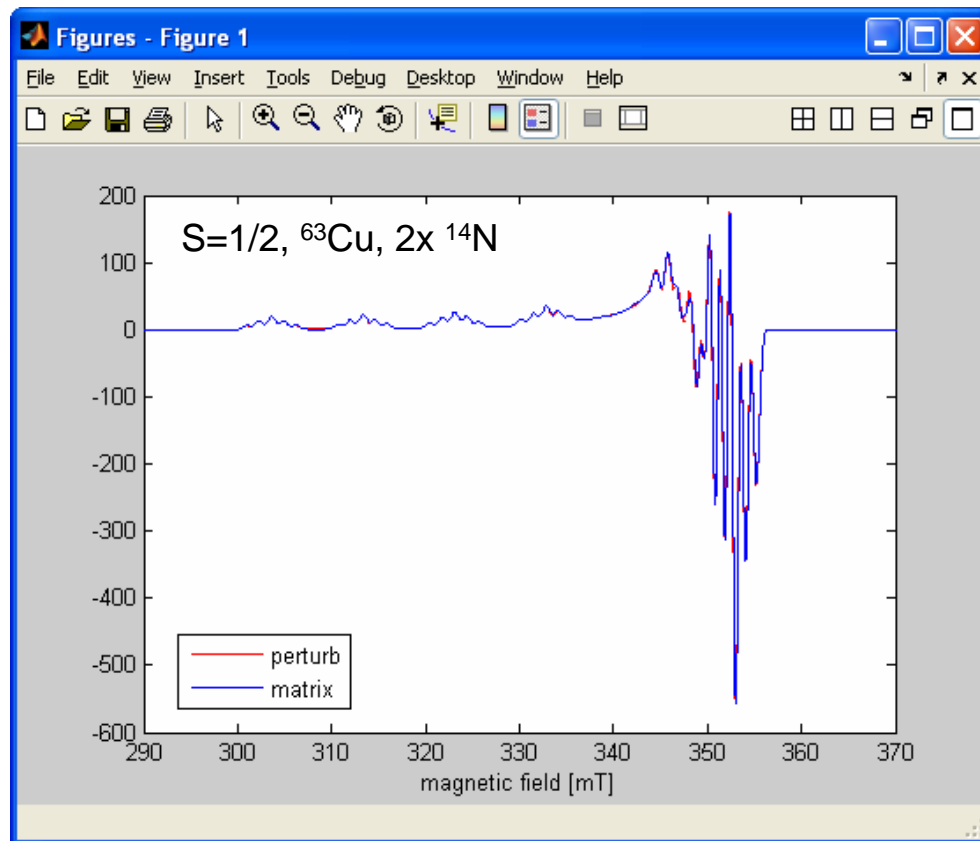
- + arbitrary number of electron and nuclear spins
- + general spin Hamiltonian
- + all transitions
- + always exact
- slow

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



Pulse EPR

new in 3.0

Simulation function: `saffron`

Type of pulse sequence

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

```
Exp.Sequence = '3pESEEM';
```

```
Exp.Sequence = 'HYSCORE';
```

Timings

```
Exp.tau = 0.120;
```

```
Exp.dt = 0.008;
```

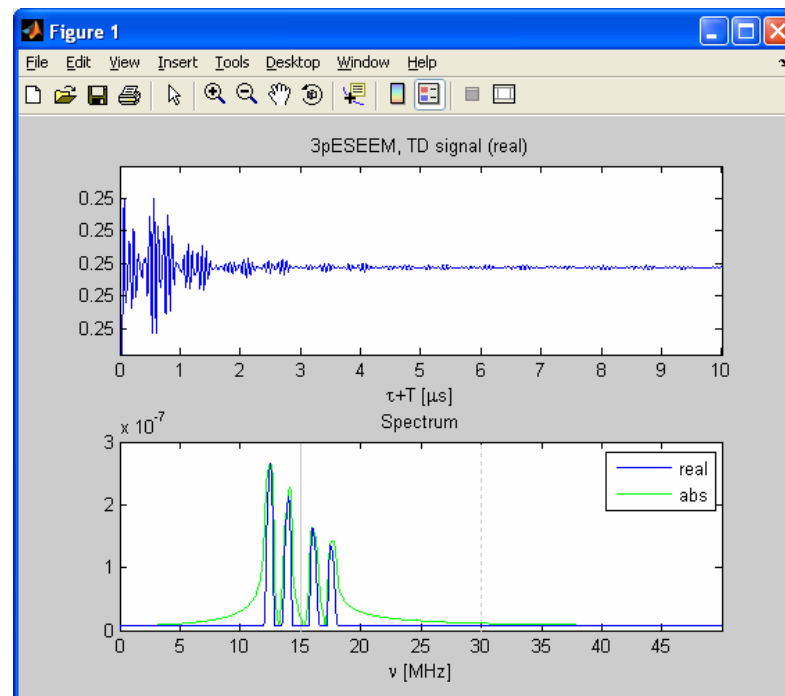
Units: μs

What it supports

- hard pulses
- several nuclear spins
- orientation selection built-in
- data processing built-in

What it doesn't support

- selective pulses
- DEER and pulsed ENDOR



total execution time: 0.3 seconds

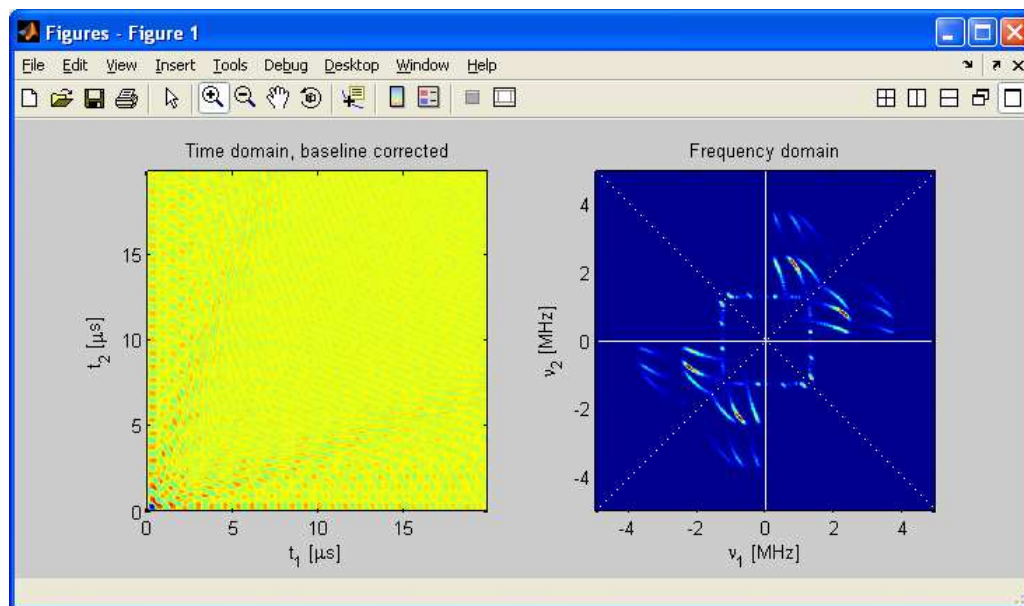


Pulse EPR

new in 3.0

Example of a HYSCORE simulation

```
Sys.Nucs = '14N';  
Sys.A = [-1 -1 2]*0.5+0.8;  
Sys.Q = [-1 -1 2]*0.2;  
  
Exp.Sequence = 'HYSCORE';  
Exp.Field = 330;  
Exp.tau = 0.001;  
Exp.dt = 0.1;  
Exp.nPoints = 200;  
  
saffron(Sys, Exp);
```

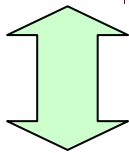
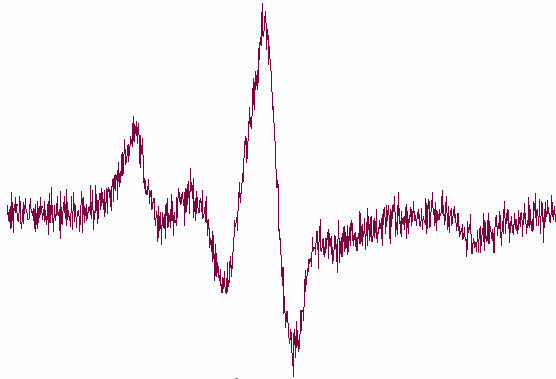


total execution time: 2.1 seconds
(including FT etc)



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

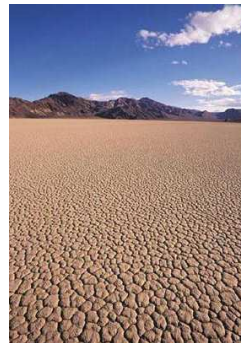
$$\chi^2 \propto \sum_{i=1}^n r_i^2$$

rmsd

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

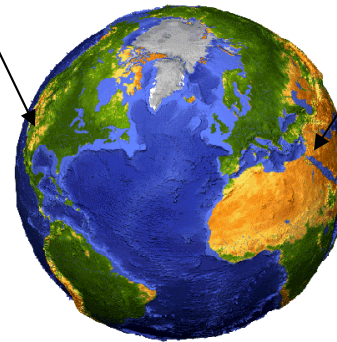
The global minimum is very difficult to locate!

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

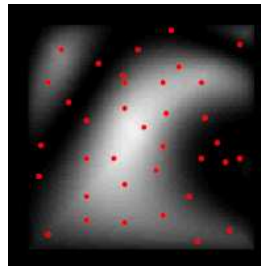
(downhill into valley from a starting point)

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

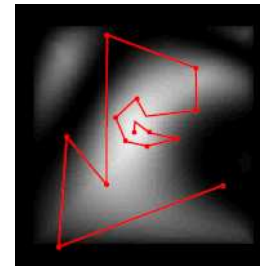
Hybrid methods

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

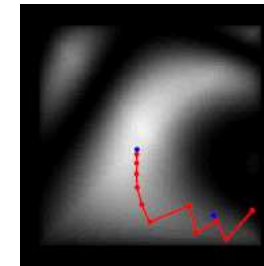
Monte Carlo



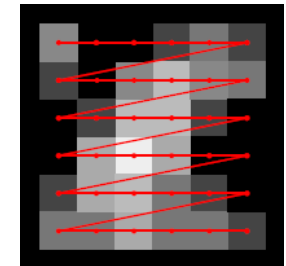
simplex



Lev/Mar



grid



Available in EasySpin:

global methods:

- grid search
- genetic algorithm
- Monte Carlo

local methods:

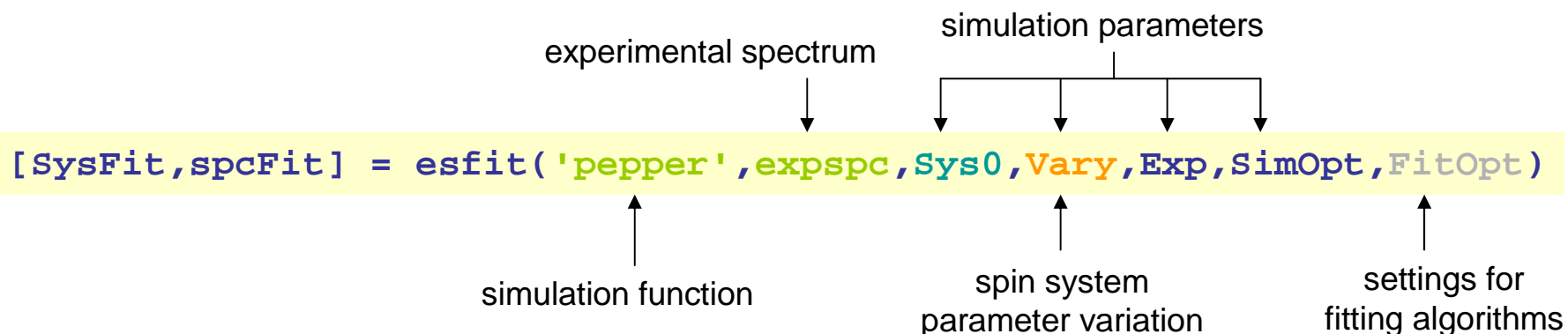
- Levenberg-Marquardt
- Nelder-Mead simplex

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.logtcrr = -8.5;
```

(2) Allowed variation of parameters

one parameter, for 1D search

```
Vary.logtcrr = 1;
```

two parameters, for 2D search

```
Vary.logtcrr = 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:

new in 3.0

`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

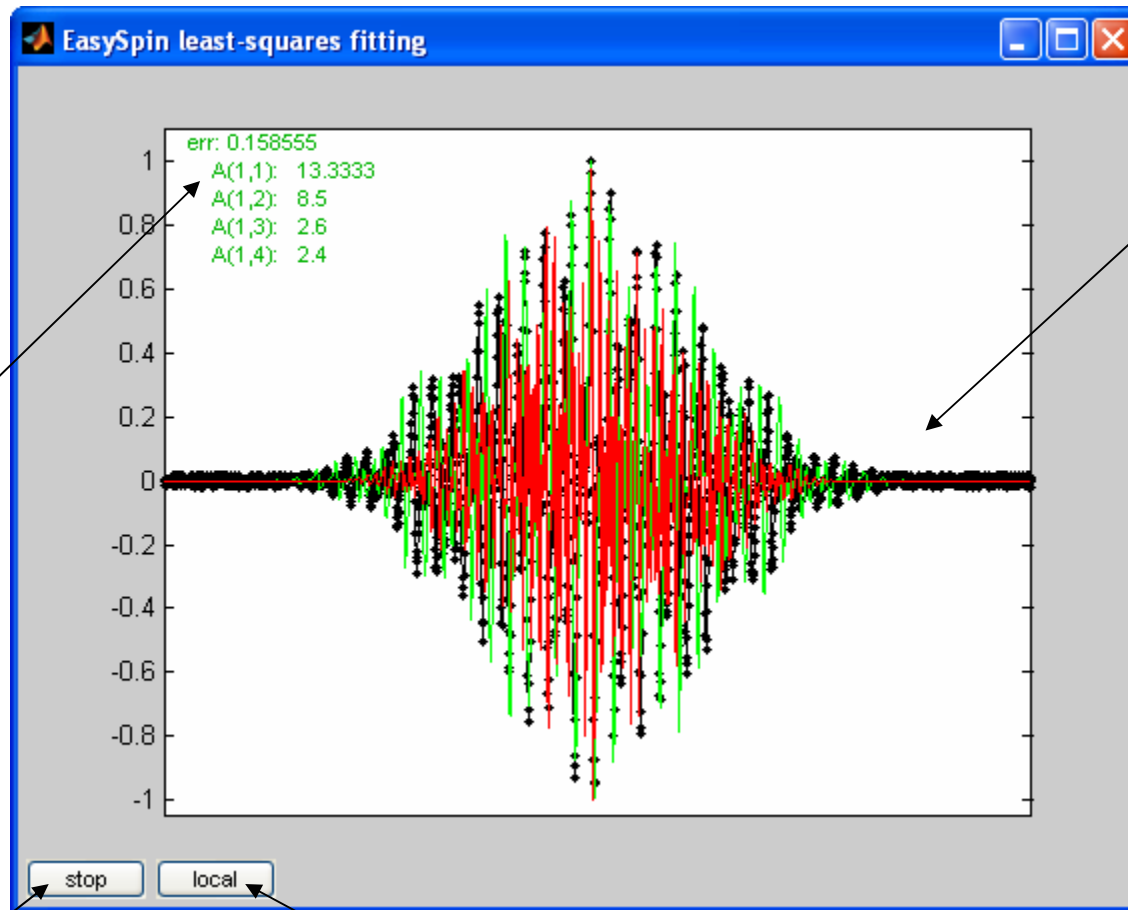
each algorithm has different additional parameters



Least-squares fitting

new in 3.0

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.



Rotations, orientations and vectors

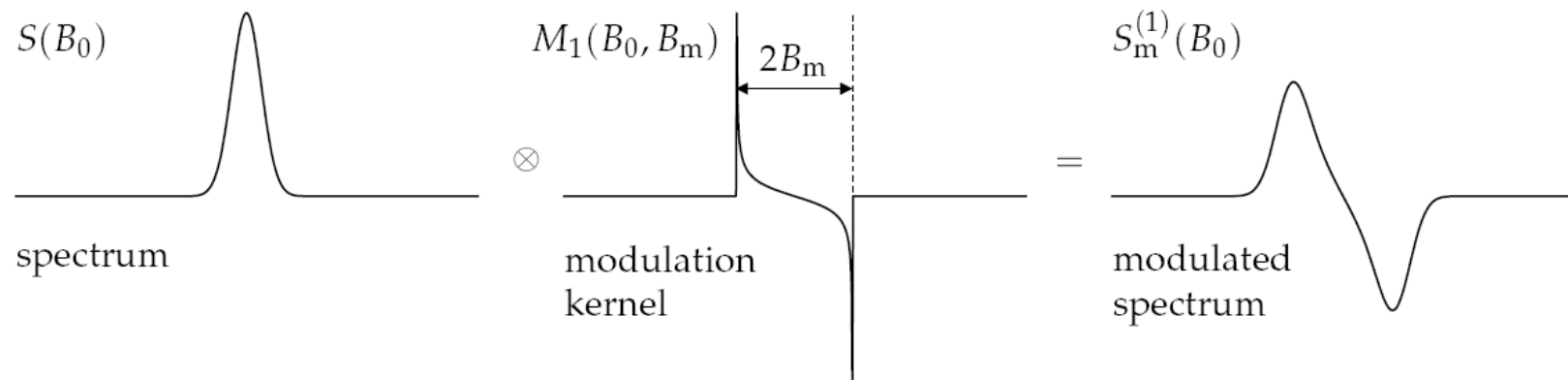
EasySpin functions for dealing with angles, orientations and rotations

<code>degree</code>	for angle unit conversion, $\text{pi} = 180 \cdot \text{degree}$
<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

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

$$\begin{pmatrix} 0 & 0.5 \\ 0.5 & 0 \end{pmatrix}$$

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

$$\begin{pmatrix} 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.5 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

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

`Sx =`

$$\begin{pmatrix} 0 & 0.7071 & 0 \\ 0.7071 & 0 & 0.7071 \\ 0 & 0.7071 & 0 \end{pmatrix}$$

`Sy =`

$$\begin{pmatrix} 0 & 0 - 0.7071i & 0 \\ 0 + 0.7071i & 0 & 0 - 0.7071i \\ 0 & 0 + 0.7071i & 0 \end{pmatrix}$$

`Sz =`

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$



Angular momenta

EasySpin includes basic support for general angular momenta computations

<code>clebschgordan</code>	Clebsch-Gordan coefficients $\langle j_1 j_2 m_1 m_2 j_1 j_2 j m \rangle$
<code>wigner3j</code>	Wigner 3- j symbols $\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$
<code>wigner6j</code>	Wigner 6- j symbols $\begin{Bmatrix} j_1 & j_2 & j_3 \\ & j_4 & j_5 \\ & & j_6 \end{Bmatrix}$
<code>plegendre</code>	Associated Legendre polynomials $P_l^m(x)$
<code>spherharm</code>	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)
```



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, 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 15059, 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)