



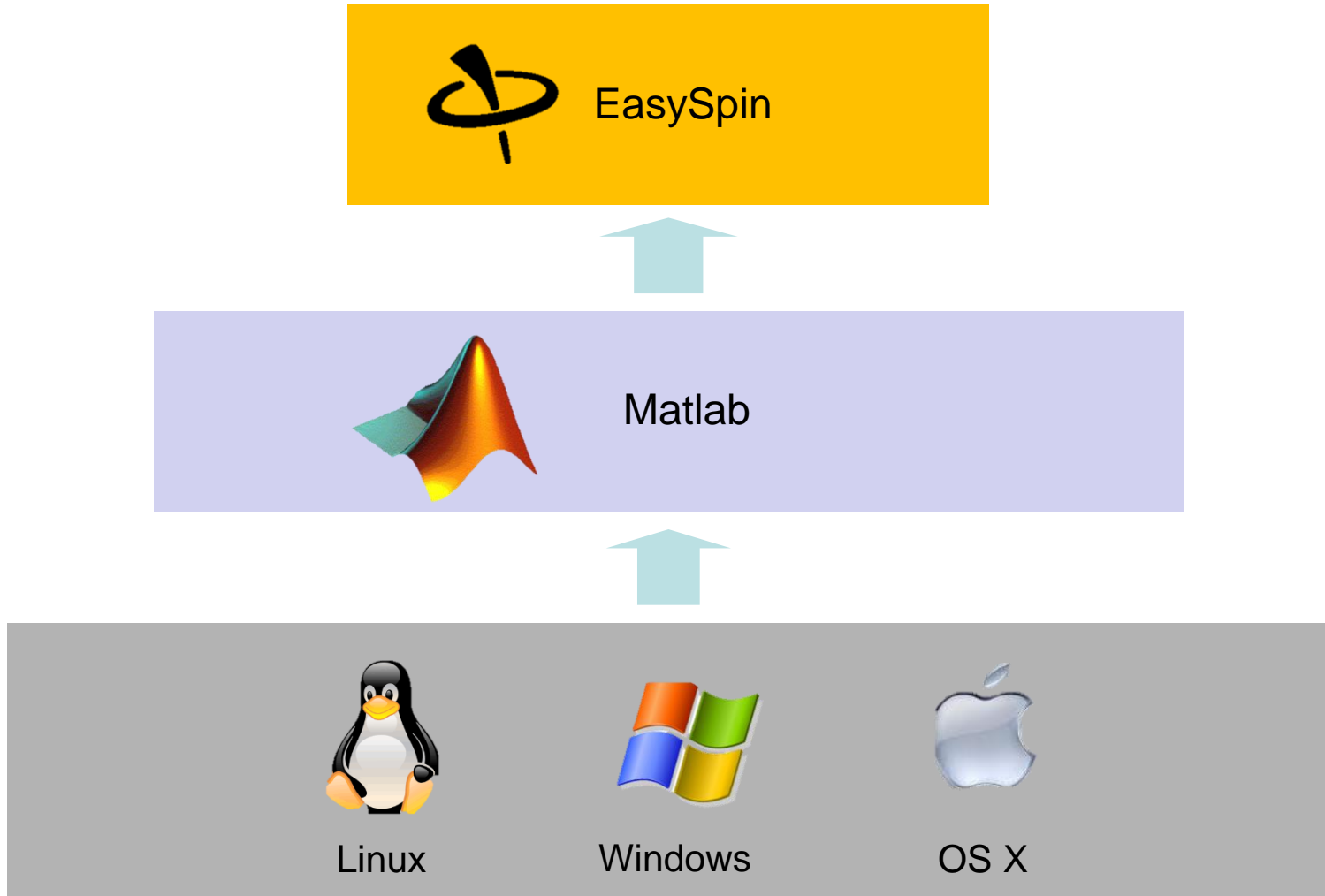
# EasySpin: EPR Spectral Analysis, Simulation and Fitting

Stefan Stoll

Department of Chemistry  
University of Washington

EasySpin 5.0  
July 2015

# EasySpin runs on Matlab, cross-platform





**Stefan Stoll** University of Washington

**Jeremy Lehner** University of Washington  
Stochastic Liouville equation solver

**Joscha Nehr Korn** Helmholtz-Zentrum Berlin  
Frequency-domain simulations  
General excitation geometries

**Matthew Krzyaniak** Northwestern University  
SpecMan4EPR data import



# What's new in EasySpin 5

- New** **Frequency-swept EPR spectra**
  - for all levels of theory and motional regimes
- New** **Rapid-scan EPR signals**
  - time-domain steady-state solutions of Bloch equations
- New** **More data formats**
  - SpecMan, Magnostech, Adani, ActiveSpectrum, JEOL
- New** **New frame definitions, improve single-crystal interface**
  - simplifies specifications of tensor and crystal orientations
- New** **Import of ORCA calculations**
  - read in calculated  $g$ ,  $A$ ,  $Q$ , and  $D$  tensors
- New** **General excitation schemes**
  - circular and linear polarization, unpolarized mw
  - arbitrary angles between  $\mathbf{B}_0$ ,  $\mathbf{B}_1$  and mw beam
- New** **Simulation of magnetization curves**
  - $\chi$ ,  $\chi^T$ ,  $\mu_{\text{eff}}$  for general spin systems



# EasySpin and Matlab: Versions

## Matlab

- Two new releases per year (spring and fall)
- Names: R2013a, R2013b, R2014a, R2014b, R2015a, etc.

**Always use most recent Matlab version.**

## EasySpin

- Major new version about once a year, bug fix releases in between
- Supports all Matlab versions starting from 7.5 (R2007b)
- Older Matlab versions cannot supported, since they lack some of the features needed in EasySpin

**Regularly check [easyspin.org](http://easyspin.org) for updates.  
Keep EasySpin updated!**



# EasySpin and Matlab documentation

## Web resources

EasySpin documentation <http://easyspin.org>  
EasySpin user forum <http://easyspin.org/forum>

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

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 user forum at easyspin.org/forum

The screenshot shows the EasySpin forum interface. At the top, there is a navigation bar with the forum logo and name, a search bar, and a search button. Below this is a breadcrumb trail: Board index > EasySpin > General forum. There are also links for FAQ, Register, and Login. The main section is titled "General forum" and includes a "NEWTOPIC" button and a search box for this forum. A table lists the forum topics with columns for the topic name, replies, views, and the last post. The topics listed are:

TOPICS	REPLIES	VIEWS	LAST POST
<a href="#">Multicomponent in pepper mode</a> by jpablo_sa » Tue Jun 09, 2015 11:56 am	2	7	by jpablo_sa Tue Jun 09, 2015 1:51 pm
<a href="#">levelsplot</a> by Wakka wakka wakka » Thu Jun 04, 2015 12:30 pm	4	47	by <b>Matt Krzyaniak</b> Tue Jun 09, 2015 6:42 am
<a href="#">Frequency Sweep</a> by Wakka wakka wakka » Thu Jun 04, 2015 12:23 pm	1	17	by <b>Stefan Stoll</b> Thu Jun 04, 2015 12:31 pm
<a href="#">Hamiltonian matrix labeling</a> by marius » Wed Jun 03, 2015 12:09 am	2	35	by <b>Stefan Stoll</b> Wed Jun 03, 2015 2:55 pm
<a href="#">saffron: identical vs. no orientations of A and g</a> by JuliaL » Mon Jun 01, 2015 12:12 pm	7	63	by <b>Matt Krzyaniak</b> Tue Jun 02, 2015 12:08 pm
<a href="#">easyspin expire soon?</a> by chemshd » Sun May 31, 2015 1:14 pm	1	38	by <b>Stefan Stoll</b> Mon Jun 01, 2015 10:18 pm
<a href="#">Ramsey Sequence code example</a> by jcammera » Mon May 18, 2015 10:18 am	1	41	by <b>Stefan Stoll</b> Wed May 27, 2015 4:35 pm
<a href="#">Separating Nuclear Manifolds</a> by AKittell » Fri May 08, 2015 8:06 am	7	95	by <b>Stefan Stoll</b> Wed May 27, 2015 4:29 pm
<a href="#">Can I use it as an alternative to XEPR</a> by AndrewLim » Thu May 07, 2015 8:18 pm	6	78	by <b>Stefan Stoll</b> Sat May 16, 2015 6:15 pm
<a href="#">Simulating Average Dipolar Coupling in Solution State</a> by AdamS » Thu May 14, 2015 12:23 pm	3	52	by <b>Stefan Stoll</b> Fri May 15, 2015 10:11 am



# Matlab: Graphical user interface

The screenshot displays the MATLAB R2015a interface with several key components highlighted by yellow boxes:

- ribbon menu:** Located at the top, it contains tabs for HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW, with various icons for file operations, workspace management, and code execution.
- workspace:** A table showing the current workspace variables:

Name	Value
A_Cu	[0.0019,0.0203]
A_N	[0.0015,0.0017]
Exp	1x1 struct
Opt	1x1 struct
Sys	1x1 struct
- editor:** A code editor window showing the script `copper_nitrogens.m` with MATLAB code for simulating an EPR spectrum.
- figures:** A plot window titled "Figure 1" showing an EPR spectrum with intensity (arb. u.) on the y-axis and magnetic field (mT) on the x-axis. The plot is labeled "9.05 GHz".
- command window:** A window at the bottom right for entering and executing MATLAB commands, currently showing `f_x >>`.

## Running 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];
```

Column vector (= Mx1 array)

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

Matrices (= 2D array)

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

Array manipulations

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

commas or spaces along rows  
semicolons to separate rows

Array functions

<code>zeros</code>	array with all 0
<code>ones</code>	array with all 1
<code>rand</code>	random numbers

Mathematical operations

<code>x*y</code>	vector/matrix multiply
<code>x.*y</code>	elementwise multiply
<code>f.^2</code>	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 = 'Ivar''s Salmon House';
```

**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 = [4 5 5];  
Software.Price = 0;  
Software.Users = 'many';  
Software.Developers = 1;  
Software.Budget = 0;  
Software.Functions = 97;  
Software.UnitTests = 495;  
Software.Citations = 1100;
```

## 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
```



**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: Installation

## 1. Install auxiliary software

Win 64bit: download

## 2. Download zip file from EasySpin website [easyspin.org](http://easyspin.org)

## 3. Unpack zip file to a directory *<mydir>*

e.g. to **C:/easyspin-x.y.z** or **/usr/var/easyspin-x.y.z**

- EasySpin folder structure:

main folder

*<mydir>/easyspin-x.y.z*

toolbox functions

*<mydir>/easyspin-x.y.z/easyspin*

documentation

*<mydir>/easyspin-x.y.z/documentation*

examples

*<mydir>/easyspin-x.y.z/examples*

## 4. Tell Matlab about EasySpin

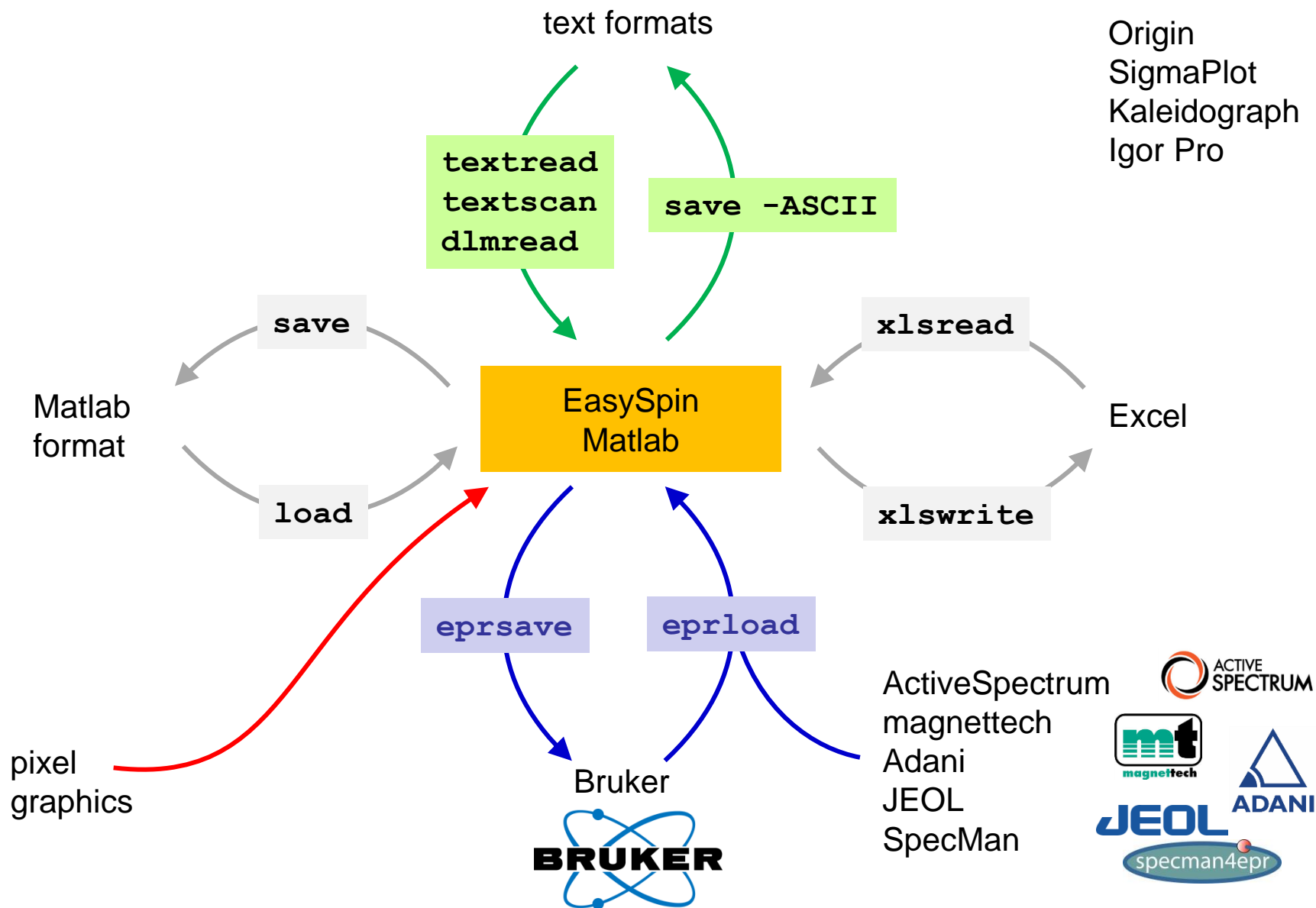
- Launch Matlab
- Go to *File > Set path ...*
- Remove any old EasySpin entries from Matlab path
- Add the folder *<mydir>/easyspin-x.y.z/easyspin* to Matlab path
- Save and close the Set path window

## 5. Finish

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



# Data import and export







# Data import and export

## Loading and importing

**eprload** reads data from Bruker spectrometers  
ESP: .spc/.par, BES3T: .DTA/.DSC  
also loads experimental parameters  
supports ActiveSpectrum, magnettech and Adani spectrometers

**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');
```

`eprsave` saves data in Bruker EPR format (BES3T)



## Baseline correction etc

`basecorr`

polynomial baseline correction

`exponfit`

fits exponentials to data

`mean`

computes the mean

`datasmooth`

data smoothing

`addnoise`

adds noise to data

## Fourier transform etc

`fft`

Fast Fourier Transform (complex)

`fdaxis`

frequency-domain axis

`ctafft`

cross-term averaged FFT

`apowin`

apodization window (Hamming etc)

## Complex data

`real`

real part

`imag`

imaginary part

`abs`

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)
```

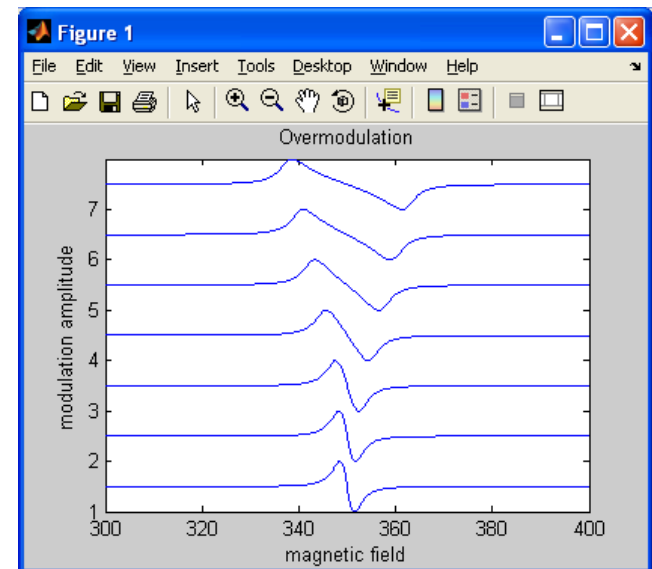
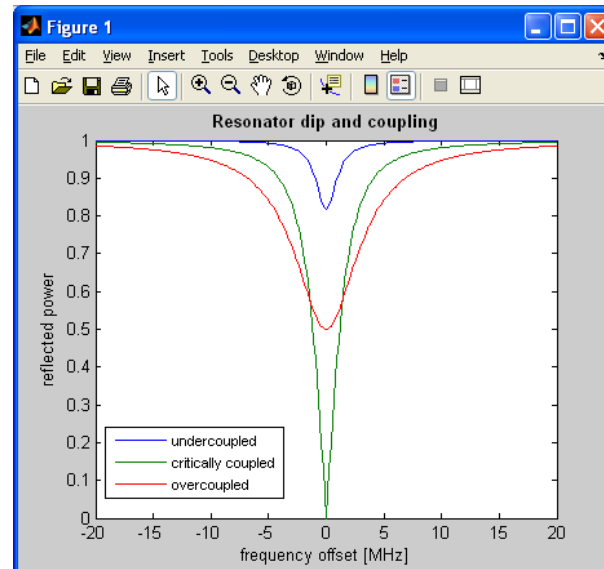
```
xlim([0 30])
ylim([0 2])
```

## Changing the appearance of plots

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

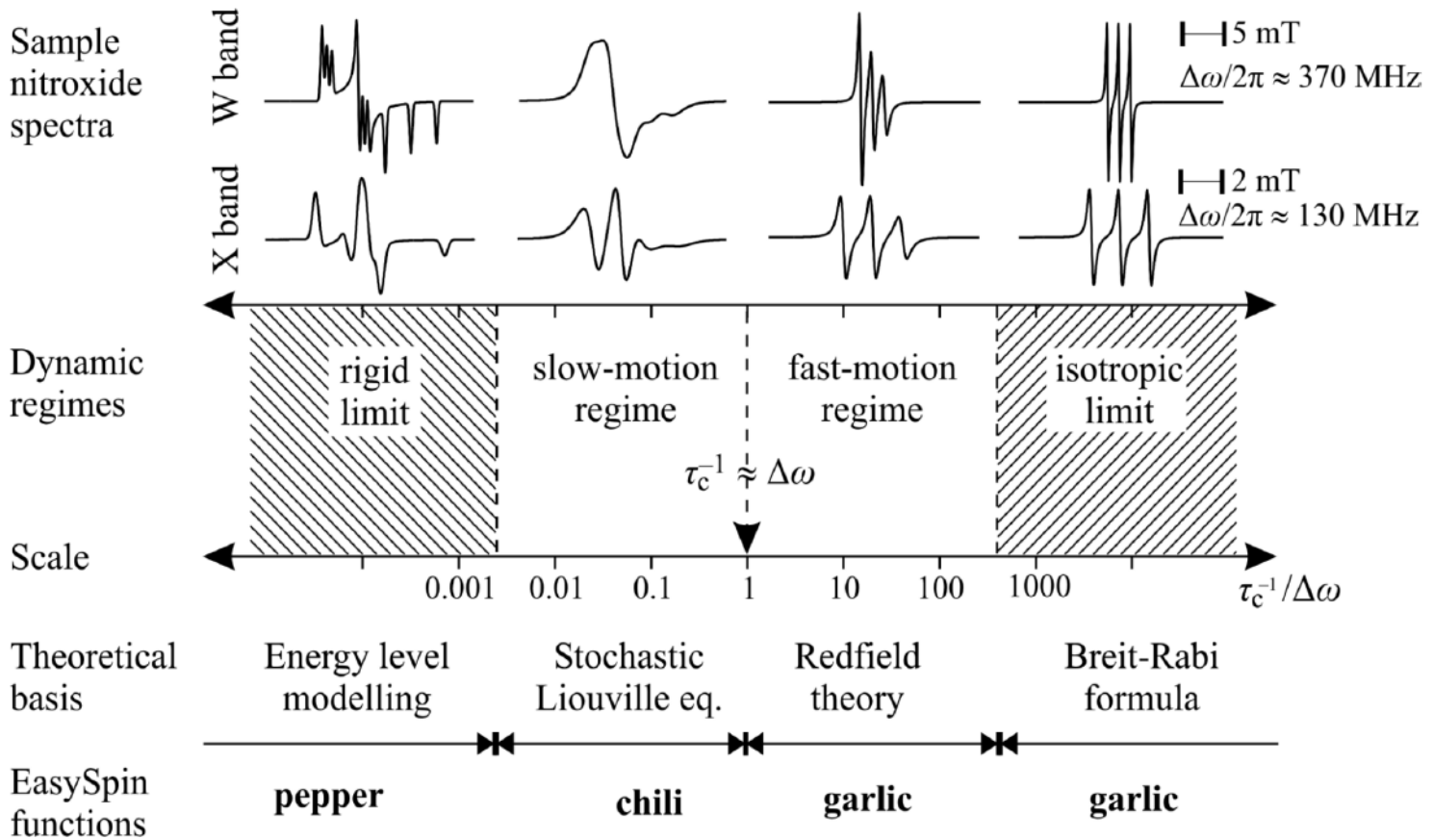
```
hold on
hold off
```

```
axis tight
```





# Motional regimes





# EasySpin simulation functions

cw EPR:      rigid-limit: **pepper**  
                 isotropic-limit and fast-motion: **garlic**  
                 slow-motion cw EPR: **chili**

ENDOR:      **salt** (mostly cw)

pulse EPR: **saffron** (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



# Spin system: 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

`pepper`

solid-state cw EPR

arbitrary number, arbitrary spin

`garlic`

isotropic, fast-motion

one spin with  $S = 1/2$

`chili`

slow-motion

one spin with  $S = 1/2$

`salt`

ENDOR

arbitrary number, arbitrary spin

`saffron`

pulse EPR

one spin with arbitrary  $S$



# Spin system: g matrices

$$(\mu_B/h)\mathbf{B}g\hat{\mathbf{S}} = (\mu_B/h)(B_xg_x\hat{S}_x + B_yg_y\hat{S}_y + B_zg_z\hat{S}_z) \quad g \text{ eigenframe}$$

## 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.gFrame = [0,pi/4,0]`

one row per electron spin

orientation of *g* tensor frame in molecular frame

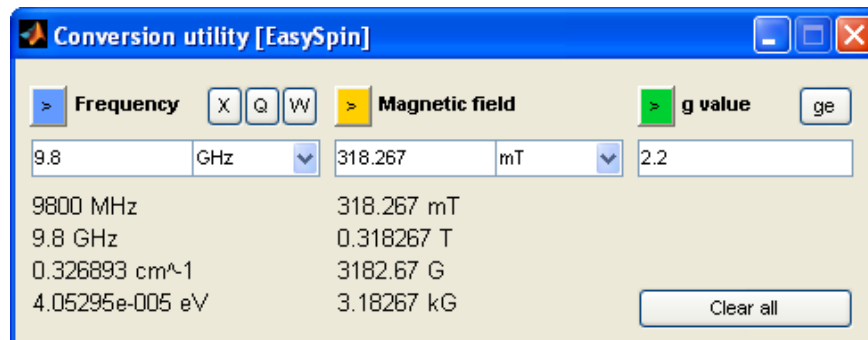
Units: radians

## Frequency/field/g conversions

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

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

or use `eprconvert`





# Spin system: Zero-field splitting tensors

**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<sup>-1</sup> = 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.DFrame = [0 40 32]*pi/180;
```

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



# Spin system: 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);
```



# Spin system: Magnetic nuclei

## Nuclear constants

**isotopes** interactive periodic table 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 where elements are color-coded by their nuclear spin properties. Below the table is a data table for the selected element, 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%

Magnetic field [mT]



# Spin system: Isotope mixtures

**Natural-abundance mixtures:** omit mass number

`Sys.Nucs = 'Cu' ;`

69%  $^{63}\text{Cu}$ , 31%  $^{65}\text{Cu}$

`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%  $^{12}\text{C}$ , 1.1%  $^{13}\text{C}$

"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] ;`



# Spin system: Multiple components

## One component

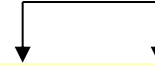
one spin system



```
... = pepper (SysA, Exp, ...)
```

## Two components

two spin systems



```
... = pepper ({SysA, SysB}, Exp, ...)
```

Weights in SysA.weight, SysB.weight.

Arbitrary number of components possible.



# Spin system: Hyperfine matrices

**Hamiltonian** (in frequency units)

$$\hat{S}\hat{A}\hat{I} = \hat{S}_x A_x \hat{I}_x + \hat{S}_y A_y \hat{I}_y + \hat{S}_z A_z \hat{I}_z$$

Units: MHz

**Principal values**

one row per nucleus

1 mT = 28 MHz  
 $10^{-4} \text{ cm}^{-1} = 3 \text{ MHz}$

`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];`

$[a_{\text{iso}}, T, \rho]$

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

**Orientation**

one row per nucleus

Units: radians

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

Euler angles of  $A$  eigenframe in molecular frame



# Spin system: Nuclear quadrupole tensors

**Hamiltonian** (in frequency units)

$$\hat{I}\mathbf{P}\hat{I} = P_x\hat{I}_x^2 + P_y\hat{I}_y^2 + P_z\hat{I}_z^2$$

(in eigenframe of  $\mathbf{P}$ )

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

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

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

**Tensor principal values**

one row per nucleus

Units: MHz

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

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

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

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

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

3 principal values (in MHz)

**Tensor orientation**

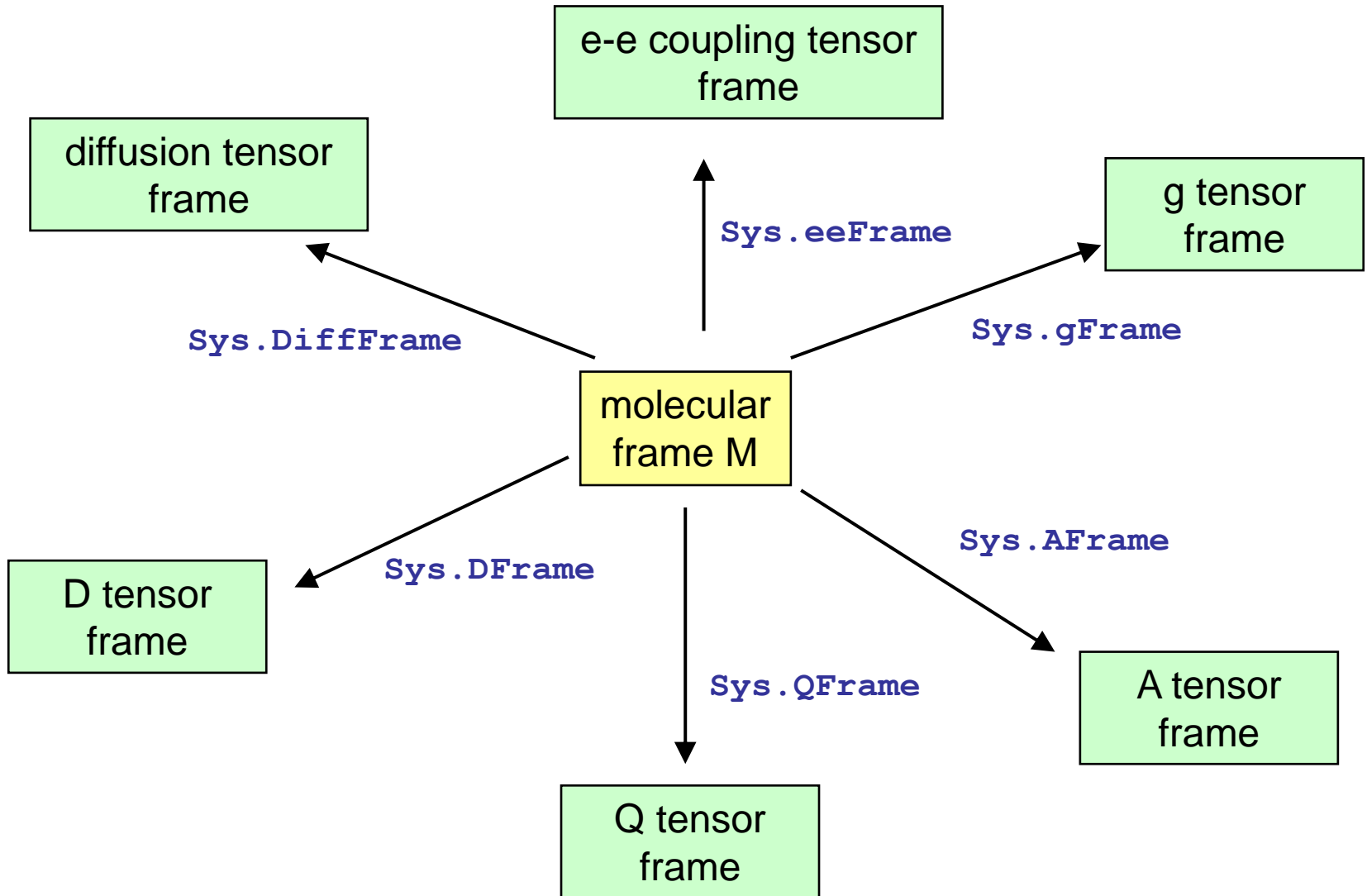
one row per nucleus

Units: radians

$$\text{Sys.QFrame} = [0, \pi/4, 0] \text{ Euler angles of Q eigenframe in molecular frame}$$



# Frames and tensor orientations

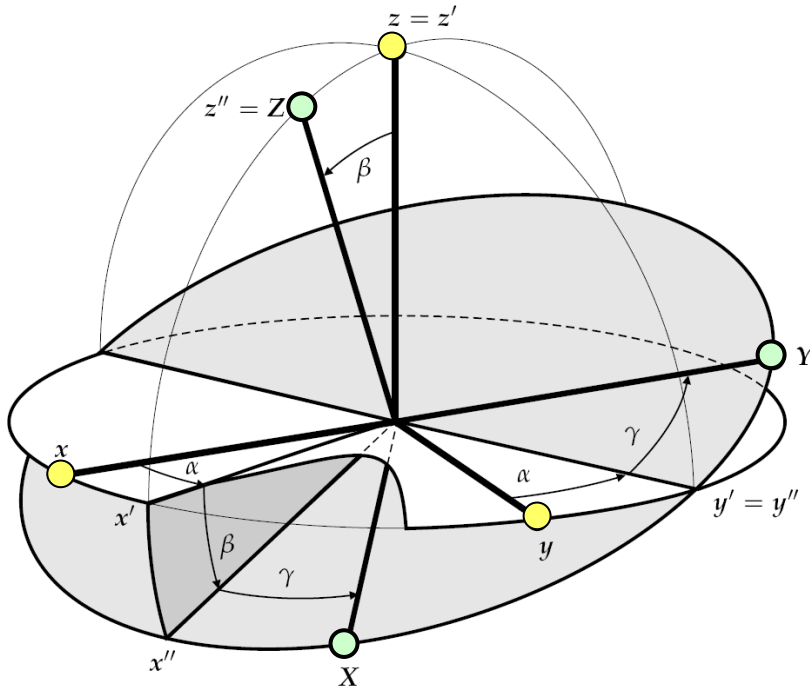
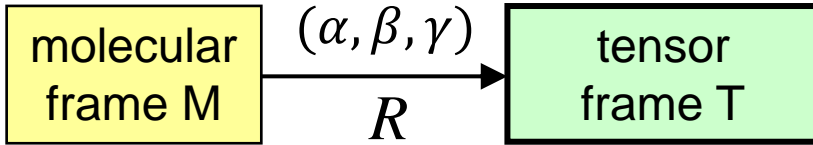


For each frame orientation: 3 Euler angles  $(\alpha, \beta, \gamma)$  that transform M frame to tensor frame.





# Frames and tensor orientations



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

```

% 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: g axes in molecular frame
columns of R: molecular axes in g frame
    
```

```

% g tensor in molecular frame
g_mol = R.'*g_eig*R
2.0218     -0.0224      0.0809
-0.0224      1.9657      0.0453
0.0809      0.0453      2.0125
    
```

```

% get Euler angles from R
eulang(R)*180/pi
10.0000     45.0000     30.0000
    
```



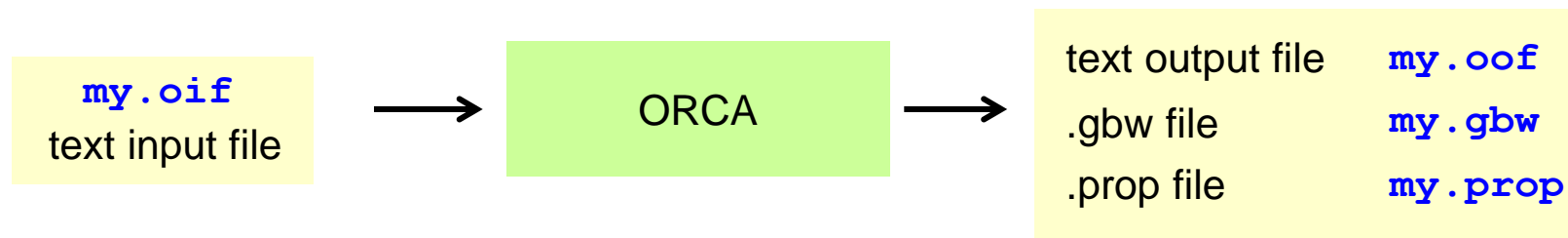
# Spin system: Import from ORCA calculation

## Requesting EPR parameters in ORCA

Include `%eprnmr` section:

```
%eprnmr
  gtensor 1
  Nuclei = all H {aiso, adip, aorb, fgrad, rho}
end
```

## ORCA calculation



## Import into EasySpin

Obtain EPR parameters from .prop file:

```
Sys = orca2easyspin('my.prop')
```

reads the .prop file

Thanks to Frank Neese

## Visualization

```
orcaview
```



# Spin system: 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)



# Spin system: 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.DiffFrame = [0 20 0]*pi/180;
```

tilt angles for  
tensor orientation



# Spin system: Ordering potential

## Ordering potential coefficients (chili)

Expansion coefficients for potential

$$U(\boldsymbol{\Omega}) = -k_B T \sum_{L,M} c_{L,M} D_{M0}^L(\boldsymbol{\Omega})$$

`Sys.lambda = [1 0 0 0 0];`

First 5 terms:  $(L, M) = (2,0), (2,2), (4,0), (4,2), (4,4)$

Frame aligns with diffusion tensor frame



# Static orientational distributions

Orientation of a spin center in the sample:

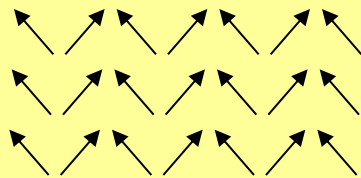
$$\Omega = (\phi, \theta, \chi)$$

Distribution of orientations of the spin center:

$$P(\Omega)$$

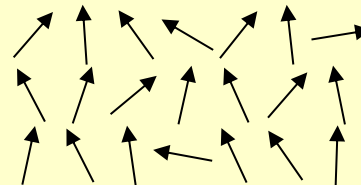
## Crystals

one or several distinct orientations with equal probabilities



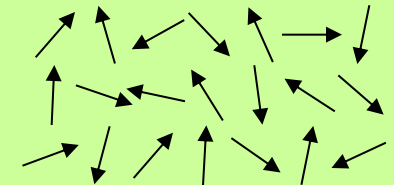
## Partially oriented systems

probability is varying function of orientation



## Powders

all orientations with equal probability



**Exp.Orientations**  
**Exp.CrystalSymmetry**

**Exp.Ordering**  
Slow motion: "partial MOMD"

*default*  
Slow motion: "MOMD"



# Single crystals

## Specify site splittings via the space or point group of the crystal

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

`Exp.CrystalSymmetry = 'P21/m';` space group symbol, Hermann-Mauguin  
`Exp.CrystalSymmetry = 11;` space group number (between 1 and 230)

Not all space group settings are supported.

EasySpin requires the third axis of the crystal abc frame to be the unique axis.

`Exp.CrystalSymmetry = 'C2h';` point group, Schönflies notation  
`Exp.CrystalSymmetry = '2/m';` point group, Hermann-Mauguin notation

Use the point subgroup of the space group, not the local point group of the paramagnetic center.

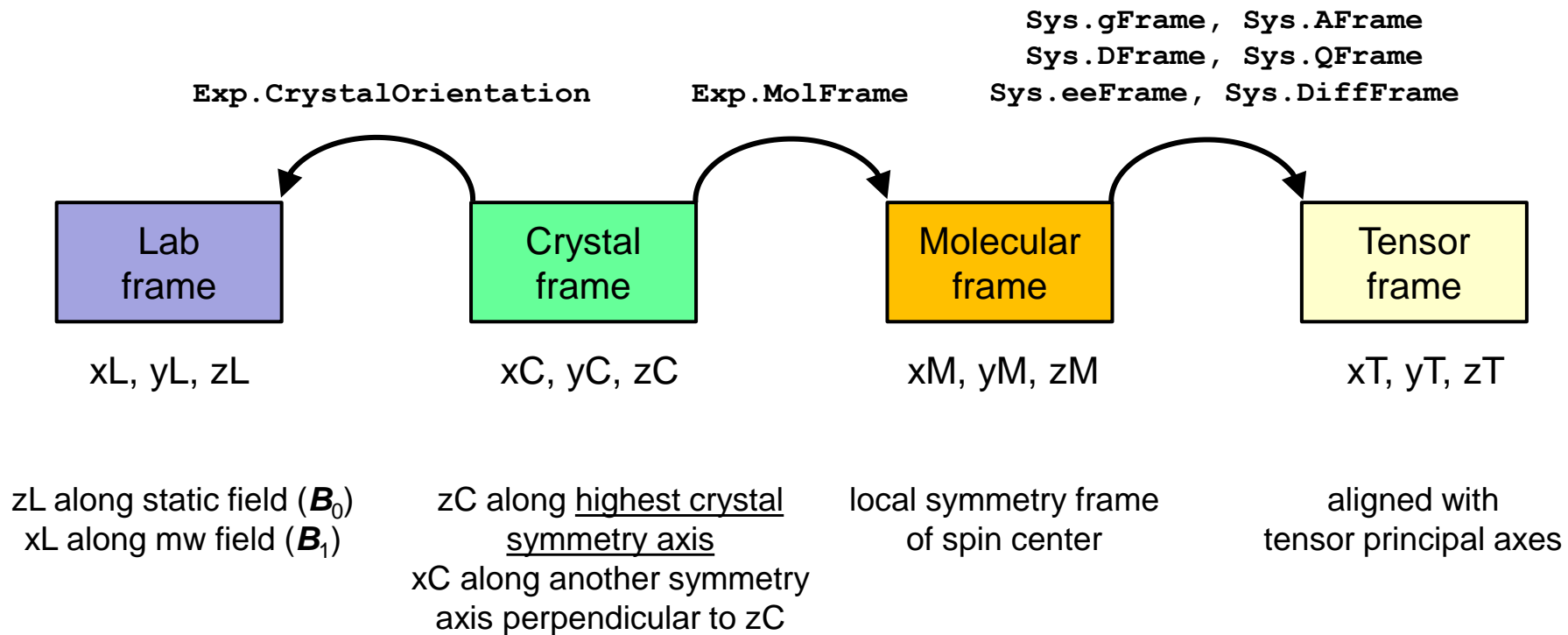
## Crystal orientation in the spectrometer

three Euler angles

`Exp.CrystalOrientation = [0 0 0];` orientation of a crystal  
`Exp.CrystalOrientation = [0 0 0; 0 pi/2 0];` twin crystal



# Frames







## Crystal rotations

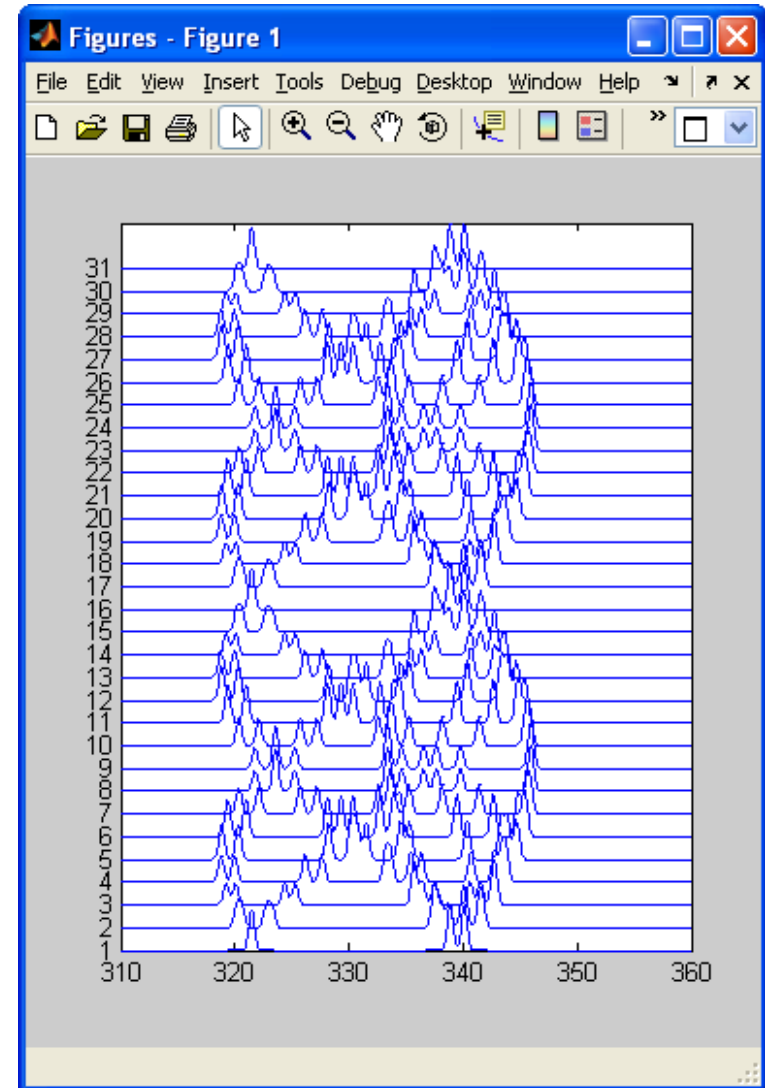
- specify rotation axis
- generate rotated orientations
- specify crystal symmetry

```
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);
```

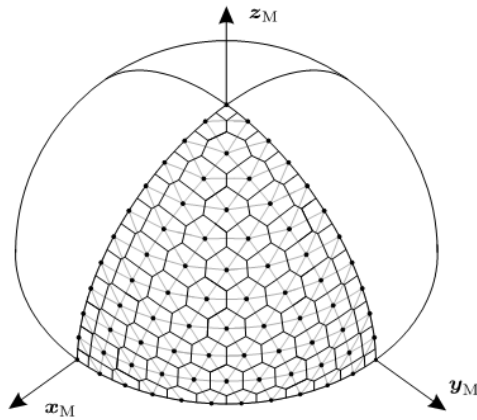
```
stackplot(B,spc);
```



# Powder spectra: Orientational averaging

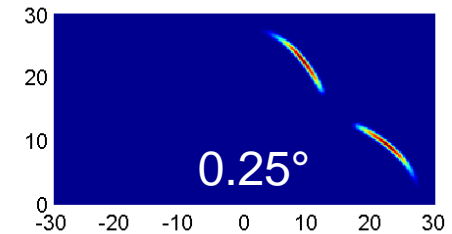
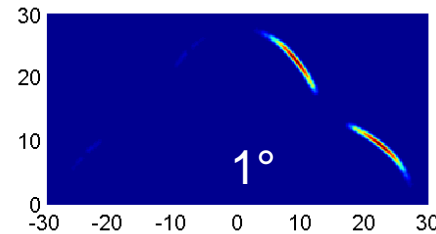
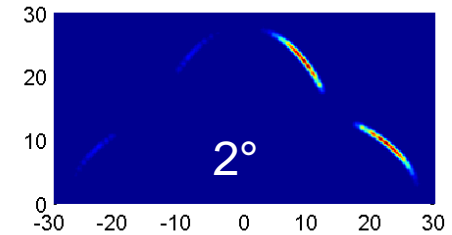
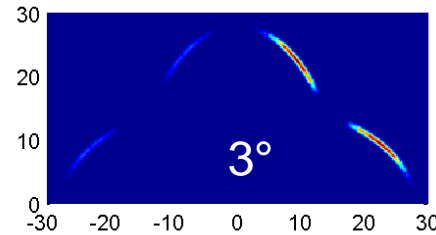
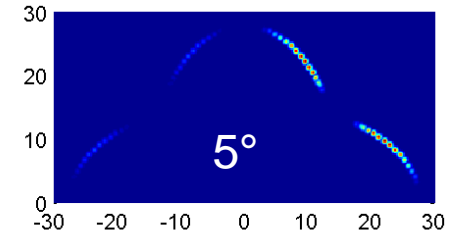
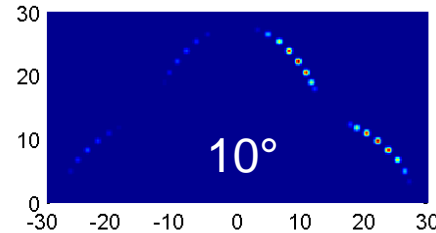
EasySpin computes powder spectra by default.

## 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;  $10^\circ$   
**Opt.nKnots** = 31;  $3^\circ$   
**Opt.nKnots** = 91;  $1^\circ$   
**Opt.nKnots** = 361;  $0.25^\circ$   
**Opt.nKnots** = 901;  $0.1^\circ$

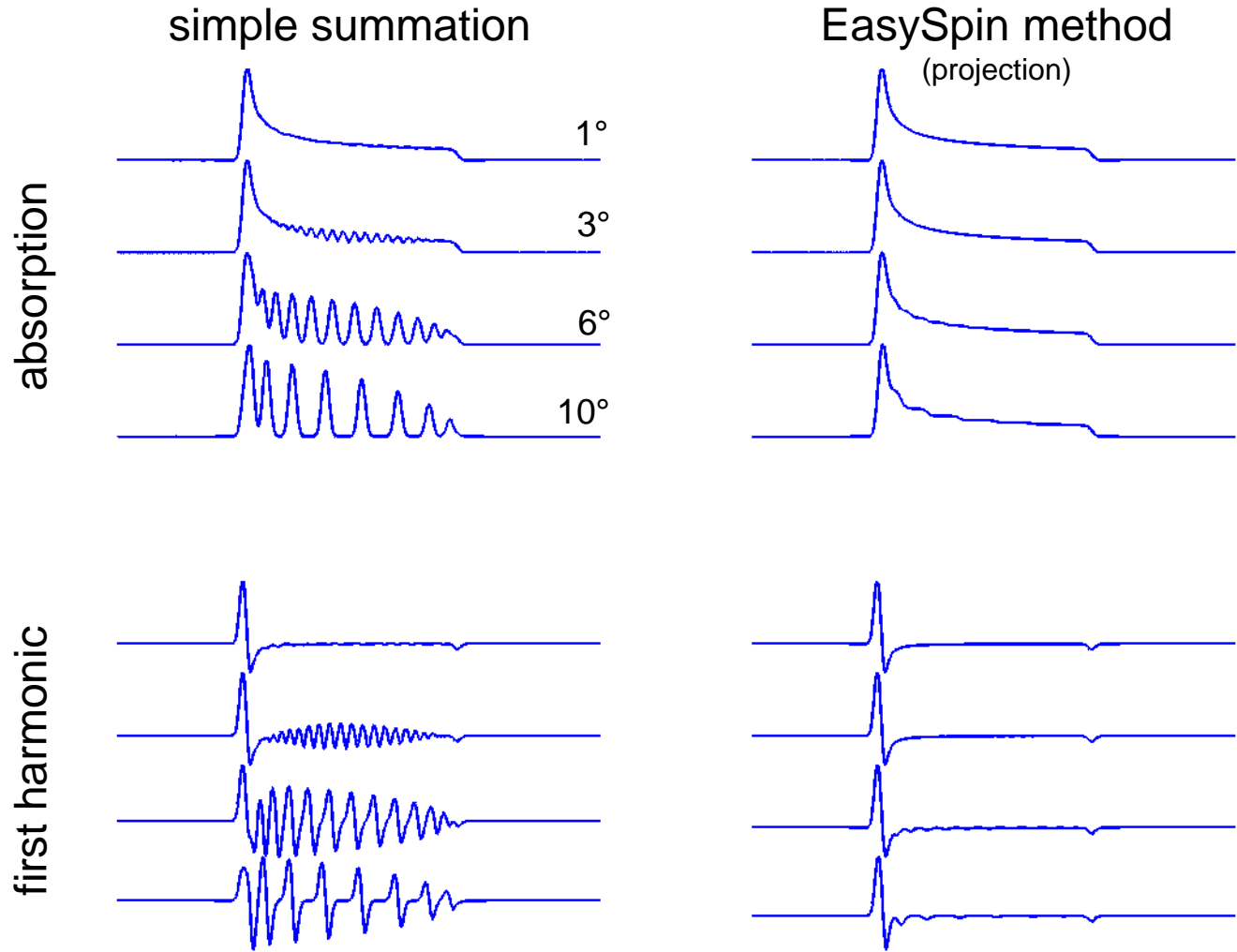


**The more orientations, the**  
- smoother the spectrum  
- longer the computation time



# Powder spectra: orientational averaging

## "Simulation grass"



pepper, salt, saffron: Opt.nKnots



# Partially oriented systems

Orientational distribution

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

Orientational potential

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

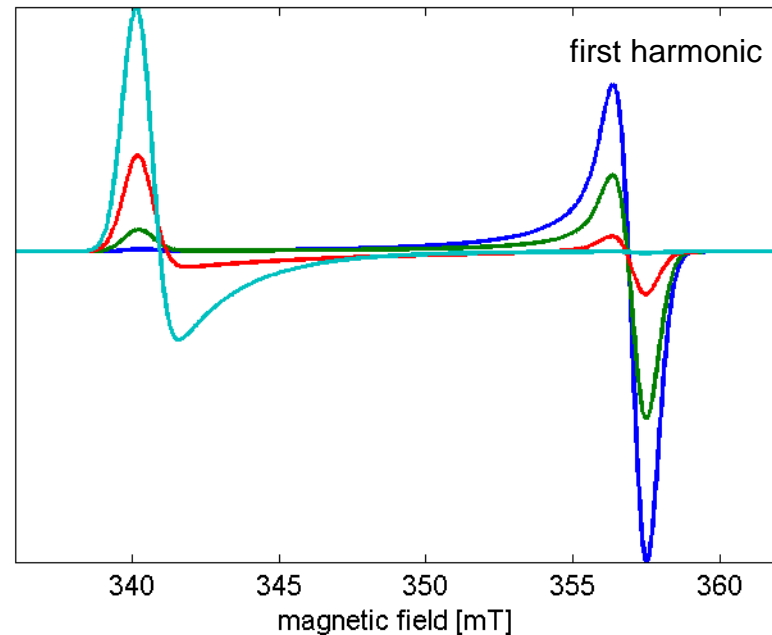
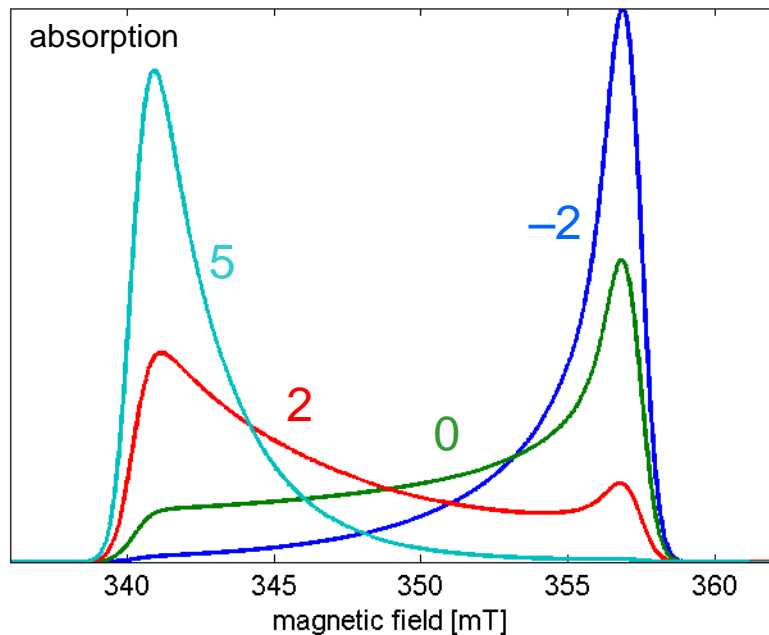
Predefined potential in EasySpin:

`Exp.Ordering = 2;`

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

Custom potential in EasySpin:

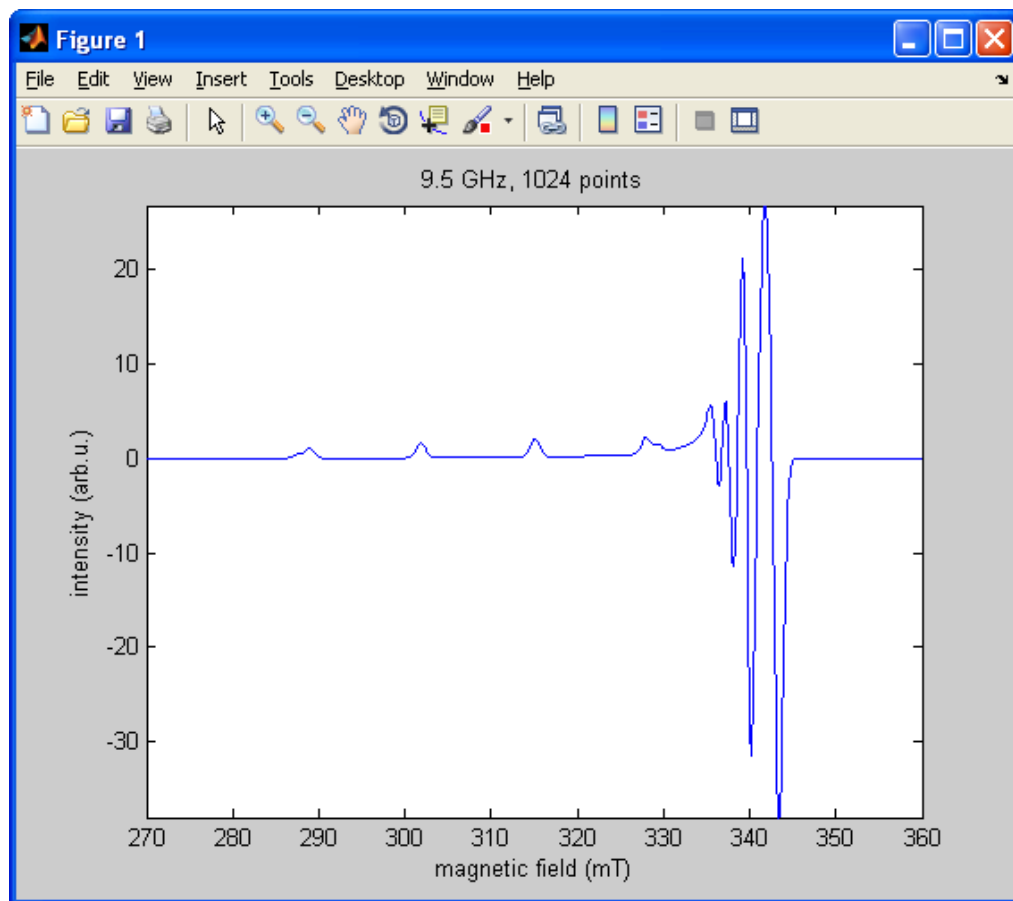
`Exp.Ordering = @(ph,th) cos(3*th).^2;`





# cw EPR: a simple 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);
```





# Field sweeps and frequency sweeps

## Field sweep

```
Sys.g = [2.05 2.3];  
Sys.Nucs = '63Cu';  
Sys.A = [40 300];
```

```
Sys.lwpp = 1; % mT
```

```
Exp1.mwFreq = 9.5; % GHz  
Exp1.Range = [250 360]; % mT
```

```
pepper(Sys, Exp1);
```

## Frequency sweep

```
Sys.g = [2.05 2.3];  
Sys.Nucs = '63Cu';  
Sys.A = [40 300];
```

```
Sys.lwpp = 1; % MHz
```

```
Exp2.Field = 350; % mT  
Exp2.mwRange = [7 10]; % GHz
```

```
pepper(Sys, Exp2);
```

- Field sweep: use **Exp.mwFreq** and **Exp.Range**
- Frequency sweep: use **Exp.Field** and **Exp.mwRange**
- If **Exp.Range** is omitted, EasySpin automatically determines range
- Units for **Sys.lw** and **Sys.lwpp** different for field and frequency sweeps
- Works equally for pepper, chili, garlic and all theory levels
- Field sweep gives first-harmonic, frequency sweep absorption spectrum



# Matrix diagonalization vs. perturbation theory

## Matrix diagonalization

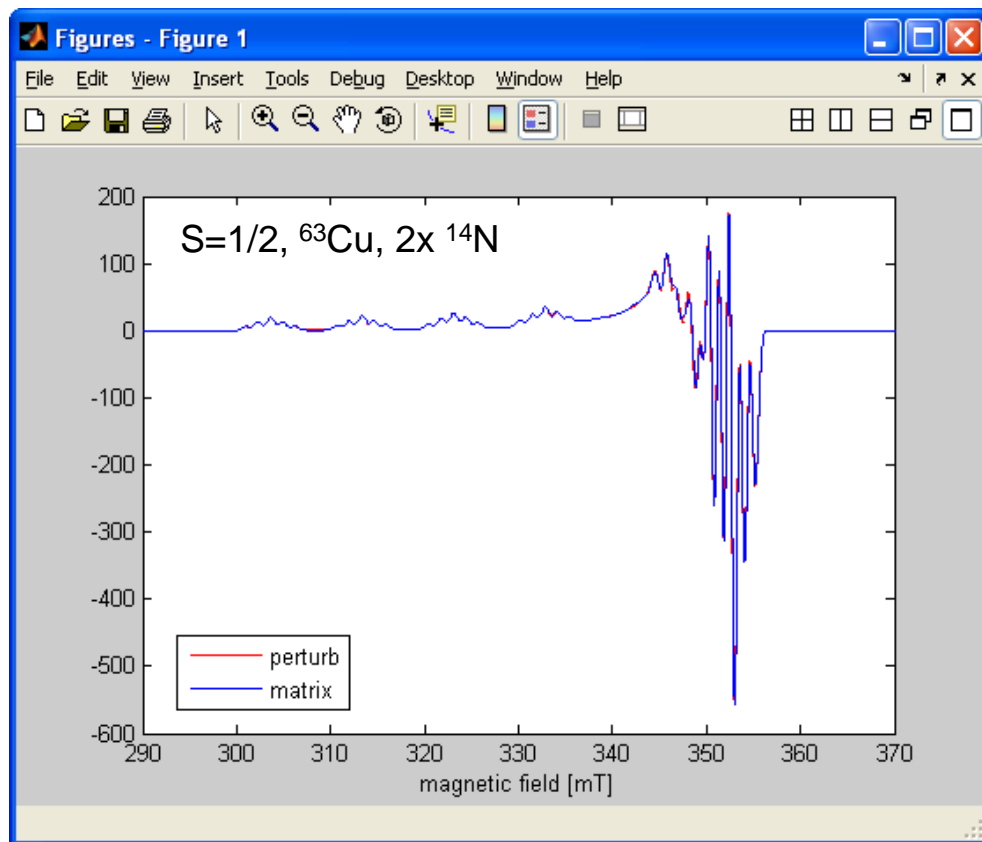
- + arbitrary number of electron and nuclear spins
- + general spin Hamiltonian
- + all transitions
- + always exact
- slow

**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

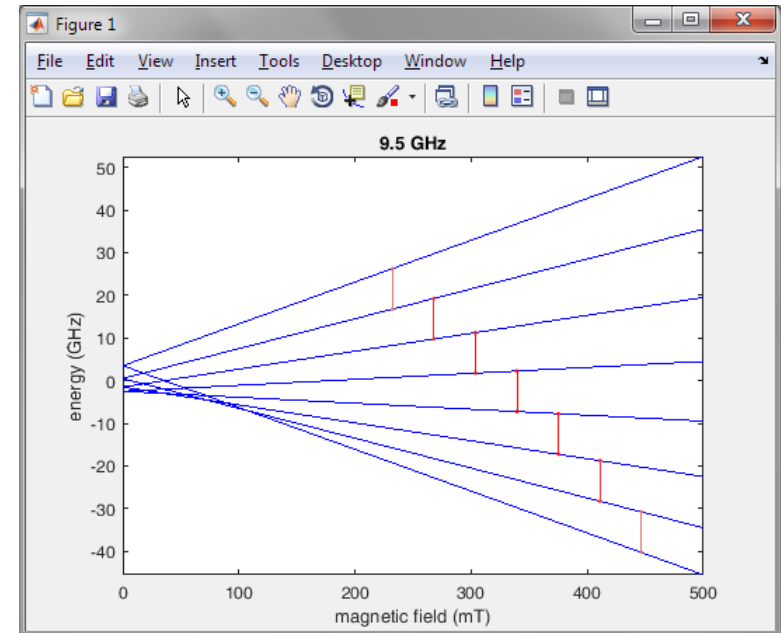


# Energy level diagrams

## Plotting of energy level diagrams:

```
levelsplot(Sys, Ori, Range, mwFreq)
```

**Sys** spin system  
**Ori** orientation  
**'x', 'y', 'z'**, or angles  $[\phi, \theta]$   
**Range** field range (mT)  
[minB0 maxB0]  
**mwFreq** microwave frequency (GHz)  
optional



(color indicated allowed/forbidden)

## Calculation of energy level diagrams without plotting:

```
E = levels(Sys, Ori, Range, mwFreq)
```





# Slow-motion regime simulations

EasySpin implements  
Freed's SLE solver

## Rotational dynamics

Rotational correlation time

```
Sys.tcorr = 1e-9;  
Sys.tcorr = [1 2 10]*1e-9;
```

isotropic  
anisotropic (axial or rhombic)

Rotational diffusion tensor

```
Sys.Diff = 1e-9;  
Sys.Diff = [1 2 2]*1e-9;
```

isotropic  
anisotropic (axial or rhombic)

## Partial ordering

Ordering potential

```
Sys.lambda = [3 1 0 0 0];
```

 potential coefficients

## Powders vs. single orientation

Powder spectrum (MOMD) by default

Single-orientation spectrum: use `Exp.CrystalOrientation`



# Slow-motion regime simulations

## **Orientational basis set**

Wigner functions  $D_{K,M}^L(\phi, \theta, \chi)$

Maximum values for even L, odd L, K and M:

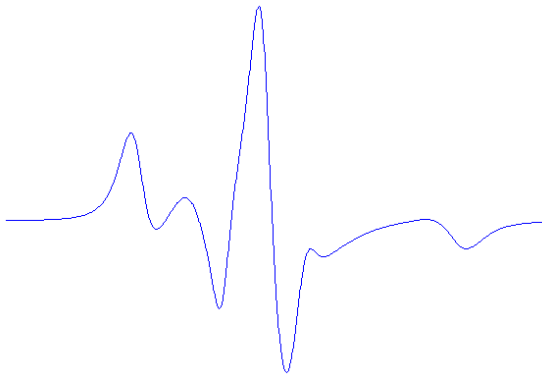
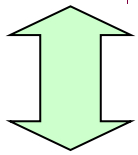
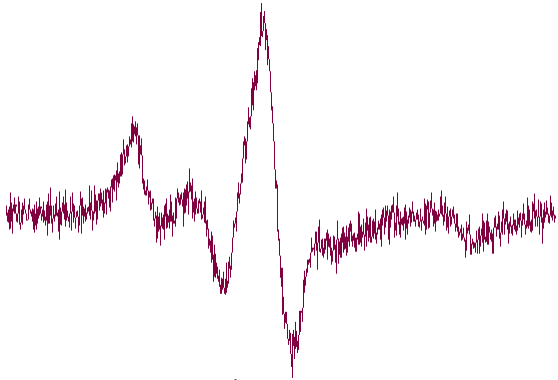
```
Opt.LLKM = [12 0 6 6];
```

The slower the motion and the more anisotropic the spectrum, the larger these values have to be to produce artifact-free spectra.



# Least-squares fitting: Overview

experimental spectrum  $y_{\text{exp}}$



model simulation,  $y_{\text{sim}}$

Residuals

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

Sum of squares

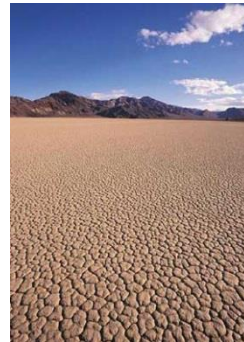
$$\chi^2 \propto \sum_{i=1}^n r_i^2$$

rmsd

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

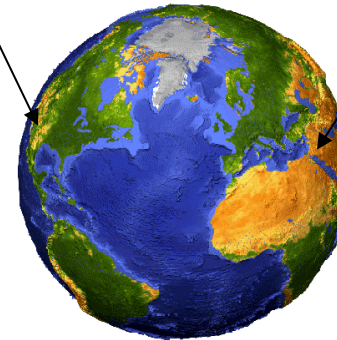
**The global minimum is very difficult to locate!**

a local minimum



Death Valley

the global minimum



Dead Sea shore



# Least-squares fitting: Methods

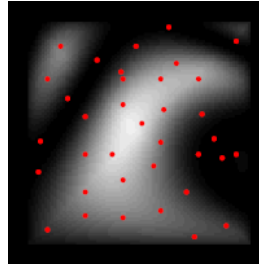
EasySpin does not use analytical derivatives

## Global methods

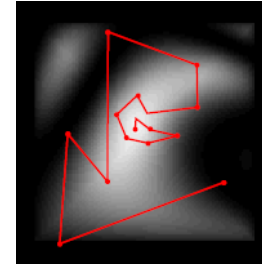
(search all over the place)

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

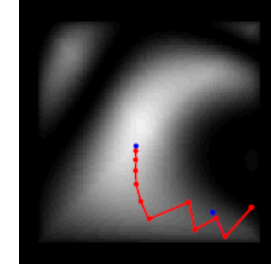
Monte Carlo



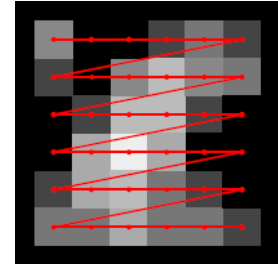
simplex



Lev/Mar



grid



## 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:

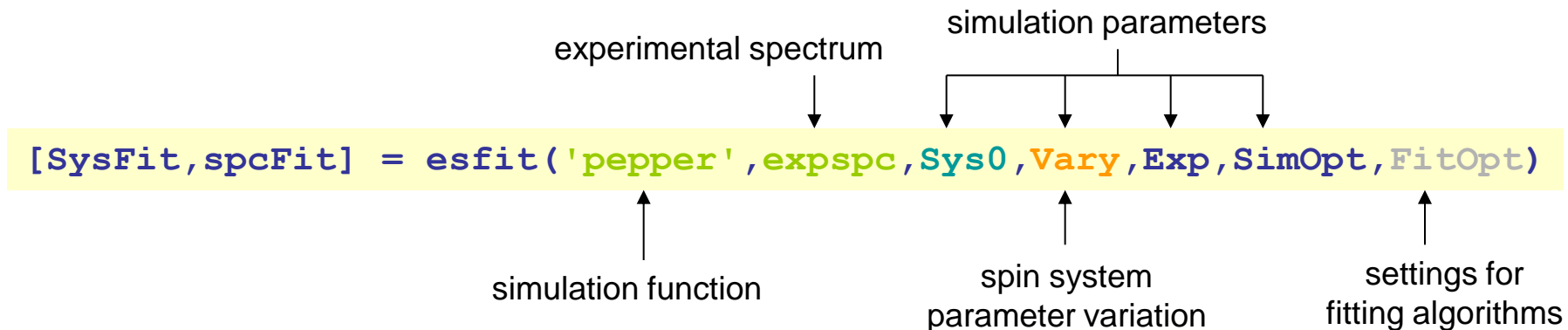
- Levenberg-Marquardt
- Nelder-Mead simplex

hybrid methods



# Least-squares fitting: esfit

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: Multiple components

## One component

one spin system



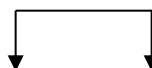
```
... = esfit('pepper', expspc, SysA, VaryA, Exp, ...)
```



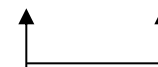
one spin system  
parameter variation

## Two components

two spin systems



```
... = esfit('pepper', expspc, {SysA, SysB}, {VaryA, VaryB}, Exp, ...)
```



two spin system  
parameter variations

Weights in SysA.weight, SysB.weight, VaryA.weight, VaryB.weight.



# Least-squares fitting: Settings

## 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: GUI

set method, target, scaling

start/stop

current and best parameter sets

black: data  
green best fit  
red: current simulation

error progress

list of stored parameter sets

	Name	best	current	center	vary
<input checked="" type="checkbox"/>	A(1)	14.4422	14.390806	14.2451	1
<input checked="" type="checkbox"/>	A(2)	10.9601	10.909001	9.96007	1
<input checked="" type="checkbox"/>	A(3)	2.64232	2.605673	2.73243	1
<input checked="" type="checkbox"/>	A(4)	2.35549	2.294728	2.12149	1

log10(rmsd)

RMSD: 7.24771  
50: 7.24771e+00 1.08635e-01 contraction inside

Fit sets





## 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  $\sigma$ :

- 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: Best practices

## 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

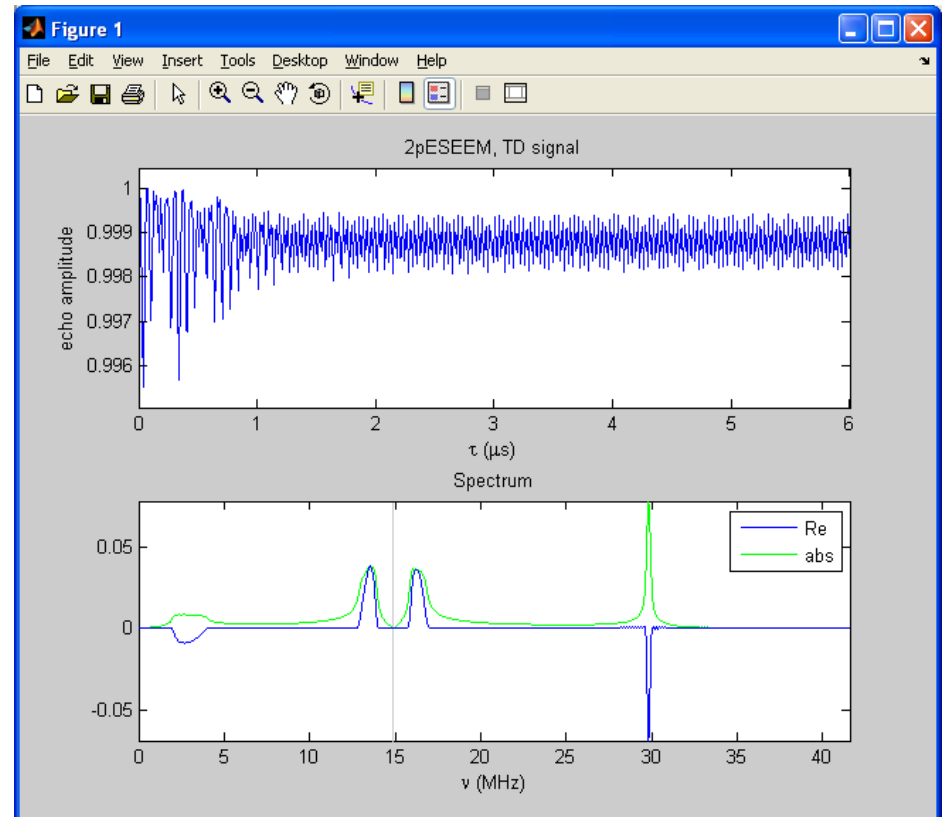
```
clear
```

```
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 = '4pESEEM' ;  
Exp.Sequence = 'HYSCORE' ;  
Exp.Sequence = 'MimsENDOR' ;
```

## Timings

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

Units:  $\mu\text{s}$

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

## 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 $\mu$ s)

```
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

## 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

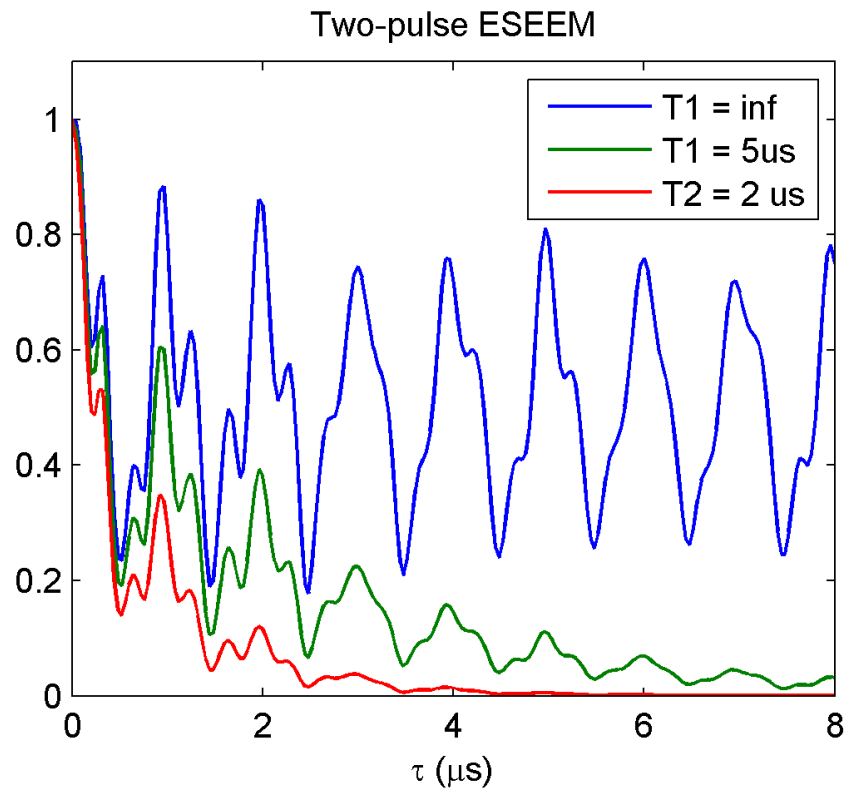
```
Exp.Sequence = 'MimsENDOR';  
Exp.tau = 0.100;  
Exp.Range = [0 30];
```

all times and delays in microseconds



# Pulse EPR: $T_1$ and $T_2$ relaxation

**Relaxation time constants** (time for fall-off to 36.8%)  
`Sys.T1T2 = [10 2];` first element:  $T_1$   
second element:  $T_2$





# Pulse EPR, ENDOR: Orientation selection

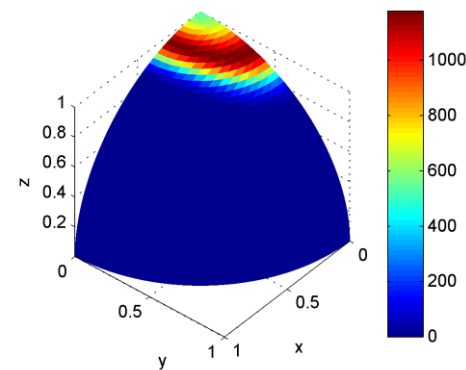
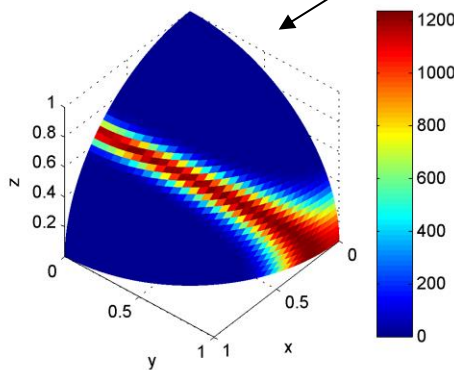
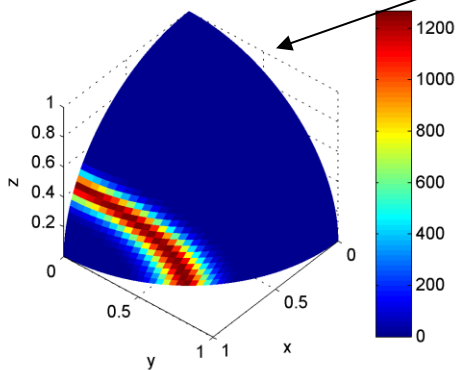
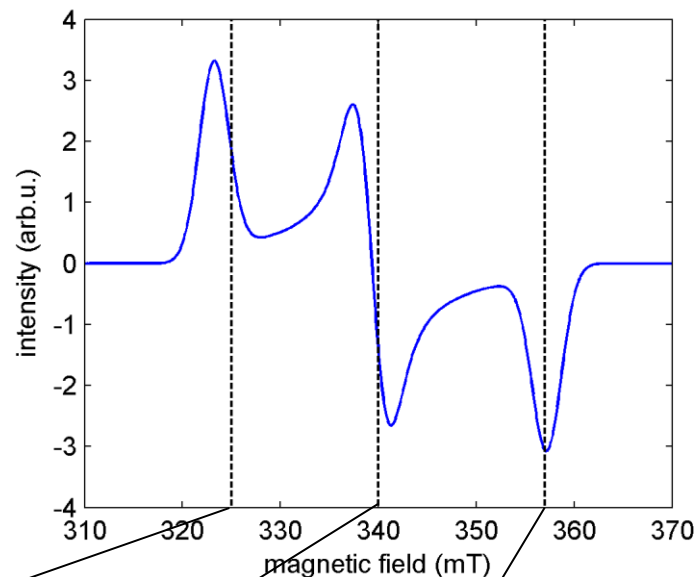
## Orientation/transition selection

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

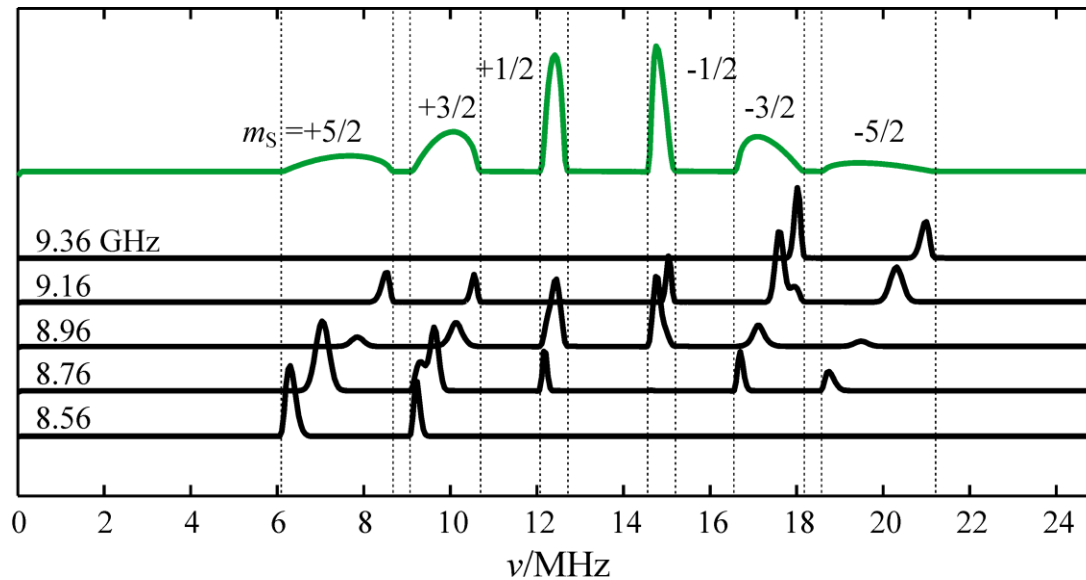
`Sys.S = 1;`  
`Sys.S = 5/2;`  
`Sys.S = 7/2;`

## Set zero-field interaction

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

## What to remember

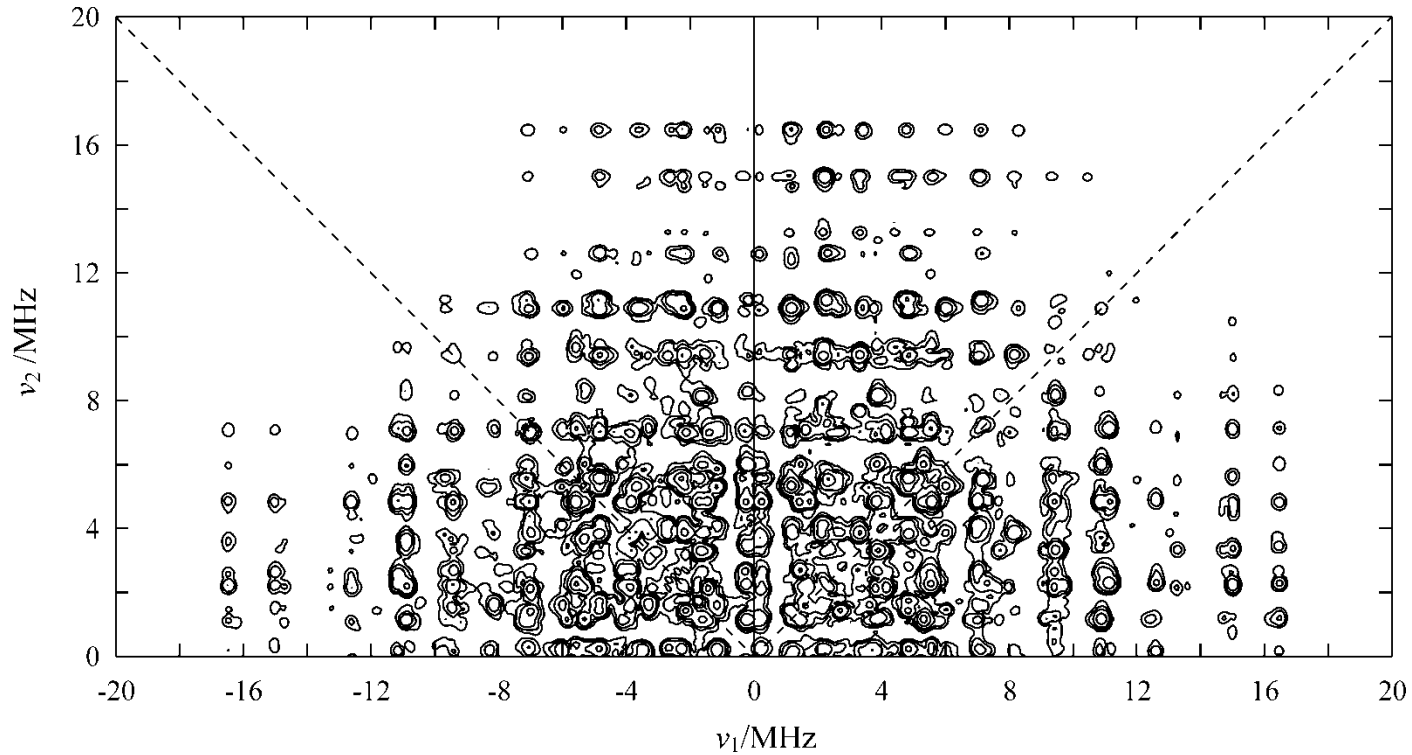
- flip angles are different for different transitions
- optimized echo amplitude might not correspond to  $90^\circ$  flips for any transition
- often strong orientation selection
- nuclear frequencies strongly depend on  $m_S$





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

**Mn(II)(Im)<sub>6</sub>**  $S = 5/2$ ,  $I = 5/2$ , 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` pulse flip angle in multiples of  $90^\circ$
- `Exp.Inc` incrementation scheme (0 = const., 1 = first dim., 2 = second dim.)
- `Exp.t` initial delays, in microseconds

## Example: Two-pulse ESEEM

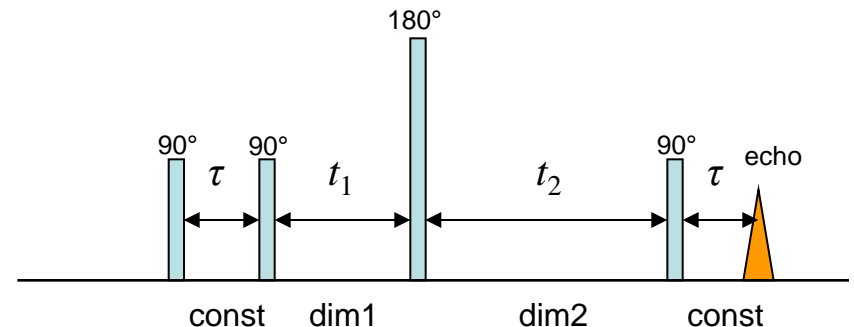
```
Exp.Flip = [1 2];
Exp.Inc = [1 1];
Exp.t = [0 0];
```

## Example: 2D three-pulse ESEEM

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

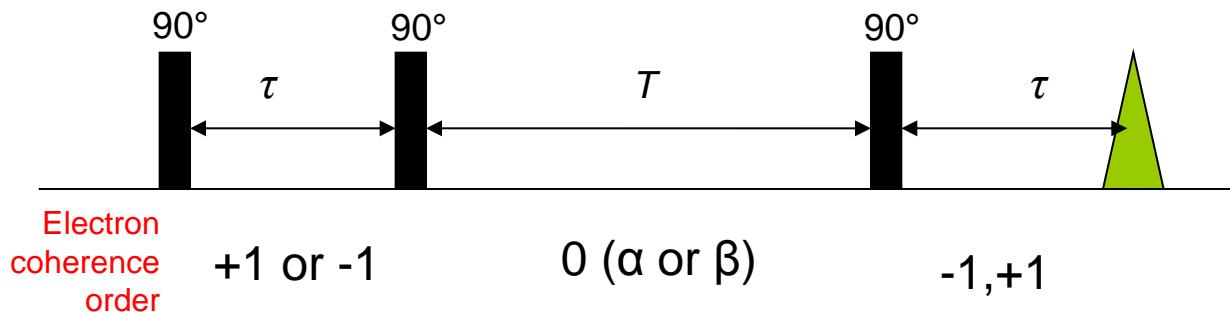
## Example: HYSCORE

```
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:

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

## Pathway filter in EasySpin:

**Exp.Filter**    one character per delay

'+'    +1  
 '-'    -1  
 'a'    0 ( $\alpha$ )  
 'b'    0 ( $\beta$ )  
 '.'    anything  
 '0'    0 ( $\alpha$  or  $\beta$ )  
 '1'    + or -

`Exp.Filter = '.a.'`

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



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

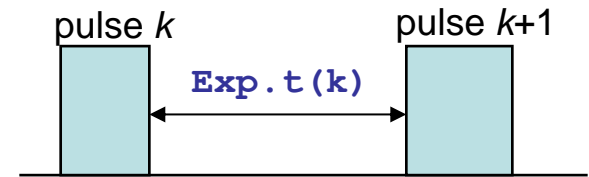
## Soft pulses in user-defined sequences

`Exp.tp` length of pulses

Units:  $\mu\text{s}$

### 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  $1/t_p$  of first pulse; typically 100 MHz)  
`Opt.nOffsets` number of integration points (typically 10-30)

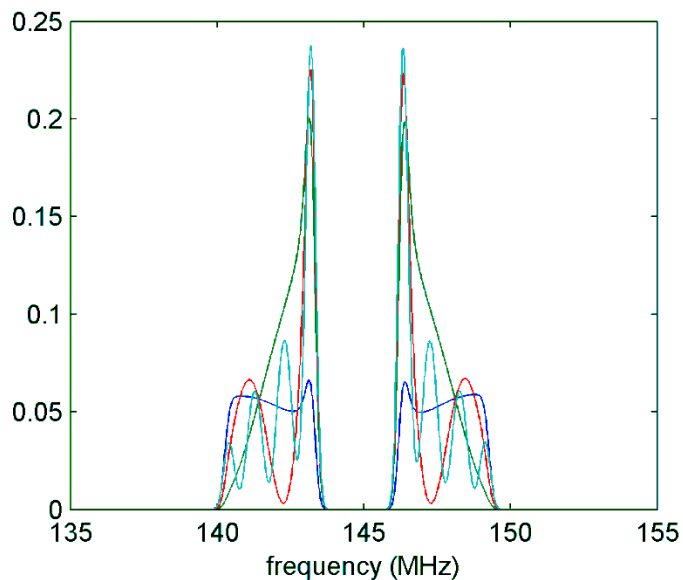
**Soft pulses are not possible with pre-defined sequences!**



# 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

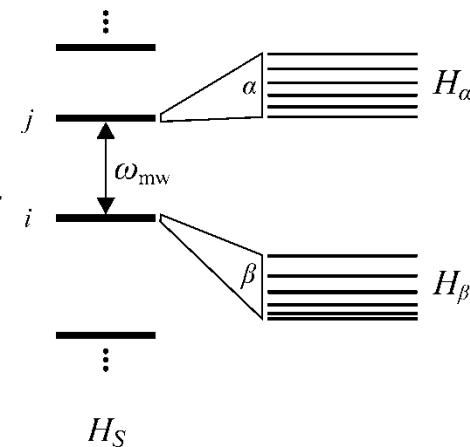
`Opt.Window = 'ham';` apodization window, applied before FFT  
`Opt.ZeroFillFactor = 2;` factor for zero filling before FFT



# 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 ( $\alpha$ ) and lower ( $\beta$ ) 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

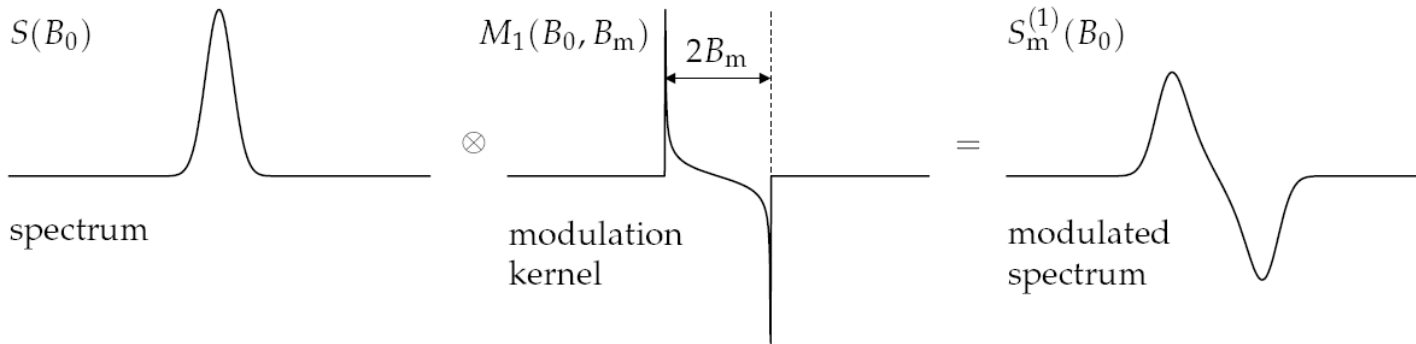
## EasySpin functions for dealing with angles, orientations and rotations

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



**Built into simulation functions**

```
Exp.ModAmp = 0.5;      % 5 G pp modulation amplitude  
[B,spc] = pepper(Sys,Exp);
```

**Separate modulation function**

```
spcmod = fieldmod(B,spc,ppAmp1);
```



# Fundamental constants

## 2010 CODATA recommended values in S.I. units

<code>gfree</code>	g value of the free electron <i>latest value (2010):</i> 2.00231930436153(53)
<code>bmagn</code>	Bohr magneton
<code>nmagn</code>	Nuclear magneton
<code>planck</code>	Planck constant
<code>hbar</code>	Reduced 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>eps0</code>	Electric constant
<code>faraday</code>	Faraday constant
<code>hartree</code>	Atomic unit of energy
<code>molgas</code>	Molar gas constant
<code>mu0</code>	Magnetic 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 quadrupole moments
<code>evolt</code>	Electron volt



# Units and conversions

## Units used in EasySpin:

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 $2\pi$ )

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

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

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

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

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

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

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

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

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

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



# Spin operators

all spin operators are in units of  $\hbar$

**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 momentum

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>cgmatrix</code>	Matrix for transformation between coupled and uncoupled representations	
<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)
```



# Common EasySpin mistakes

## 1. Left-over fields in spin system or experiment structure

```
sys = nucspinadd(sys, '14N', ...);
```

➡ Start every simulation with `clear`.

## 2. Wrong capitalization

```
sys.g = [2.3 2]; pepper(sys, Exp);
```

➡ Make sure capitalization is what Matlab and EasySpin expect.

## 3. Options not included in simulation function call

```
Opt.Method = 'perturb';  
pepper(sys, Exp);
```

➡ Whenever you add options, immediately add `opt` to the function call.

## 4. Using incorrect units

## 5. Using the wrong simulation function

## 6. Frames

## 7. Fitting: Parameter values and ranges





# 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 principal publication. Please cite if you publish results obtained with the help of EasySpin.

A PDF version is included in every EasySpin installation.

S. Stoll

CW-EPR Spectral Simulations: Solid State

*Methods Enzymol.*, in press, 2015

An overview of the key decision and choices to be made by a practitioner when simulating EPR spectra.

S. Stoll

Computational Modeling and Least-Squares Fitting of EPR Spectra

in S. Misra, ed.

Handbook of Multifrequency Electron Paramagnetic Resonance, Wiley-VCH, 2014

A comprehensive review of the underlying theory with over 500 references.

S. Stoll, R. D. Britt

General and efficient simulation of pulse EPR spectra

*Phys. Chem. Chem. Phys.* 11, 6614-6635 (2009), <http://dx.doi.org/10.1039/b907277b>

Contains background to EasySpin's 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)