About EasySpin

EasySpin is a MATLAB toolbox for simulating and fitting **Electron Paramagnetic Resonance (EPR)** spectra. It supplements the numerical and visualisation power of MATLAB with the best computational methods devised by EPR spectroscopists. *EasySpin* runs on Windows, Linux and Mac, and is available free of charge.

Current version: **4.0.0** (16 Oct 2011)

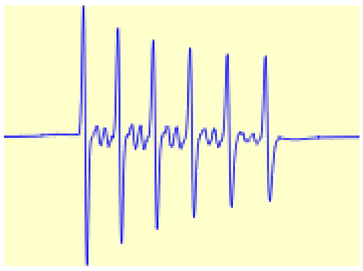
If you use results obtained with *EasySpin* in any scientific publication, cite

Stefan Stoll, Arthur Schweiger
EasySpin, a comprehensive software package for spectral simulation and analysis in EPR
J. Magn. Reson. 178(1), 42-55 (2006)

Features

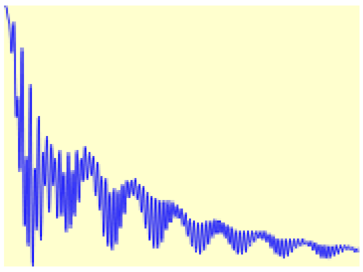
EasySpin can simulate a wide range of EPR spectra. For all simulations, **interactive least-squares fitting** using hybrid methods based on Simplex, Levenberg-Marquardt and genetic algorithms are possible.

Solid-state cw EPR spectra



- powders and crystals (with space groups)
- arbitrary number of electron and nuclear spins
- all interactions, including high-order operators and nuclear quadrupole
- matrix diagonalization and perturbation methods
- broadening models: g, A and D strain, unresolved hyperfine splittings
- non-equilibrium populations
- perpendicular and parallel detection mode

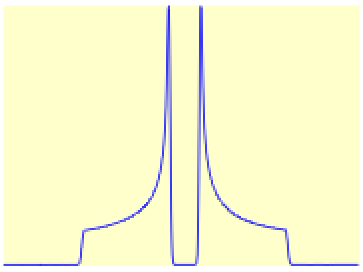
Pulse EPR



- two- and three-pulse ESEEM, HYSCORE
- user-defined sequences
- arbitrary number of nuclear spins
- high-spin systems
- nuclear quadrupole interaction included
- built-in processing

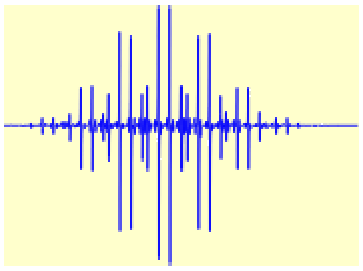
Slow-motion cw EPR spectra

- one unpaired electron, several nuclei
- axial and rhombic rotational diffusion tensor
- arbitrary tensor orientations
- orientational potential
- single-orientation and MOMD models



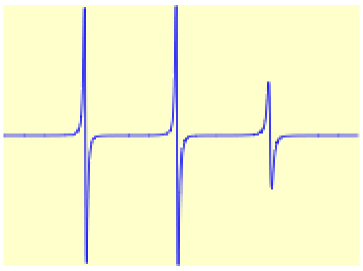
Solid-state ENDOR spectra

- powders and crystals (with space groups)
- arbitrary number of nuclei, includes nuclear quadrupole
- hyperfine enhancement
- built-in orientation selection
- matrix diagonalization and perturbation methods



Isotropic cw EPR spectra

- one unpaired electron, arbitrary number of nuclei
- resonance fields are exact (no perturbation formulae)
- automatic determination of magnetic field range



Fast-motion cw EPR spectra

- one unpaired electron, arbitrary number of nuclei
- user provides rotational correlation time and tensor anisotropies, line widths are automatically computed
- includes effects from nuclear quadrupole interaction

EasySpin also provides a variety of other tools:

- **Basic spin physics**: spin operators, Stevens operators, spin Hamiltonians with arbitrary number of electron and nuclear spins, angular momentum.
- **Utilities**: line shapes, Euler angle utilities, orientational grids, data import from all common EPR file formats (BES3T, ESP, Varian), nuclear isotope database, field/frequency/g value conversion
- **Data analysis**: RC filtering, field-modulation, cross-term averaged FFT, moving average, many apodization windows, baseline correction
- **Resonances**: ENDOR and cw EPR resonance line computations, energy level diagrams
- **Time-domain**: support for propagators and density matrix evolutions.

EasySpin reference @ http://www.easyspin.org/

Basics:

- Defining a spin system
- Line broadenings

Spectral simulations and least-squares fitting:

- garlic: cw EPR (isotropic and fast motion) user guide
- chili: cw EPR (slow motion) user guide
- pepper: cw EPR (solid state) user guide
- salt: ENDOR (solid state) user guide
- saffron: pulse EPR/ENDOR (solid state)
- esfit: least-squares fitting user guide

There is also a collection of examples.

List of all EasySpin functions: alphabetically, by category.

New features and changes, from release to release
How to install EasySpin

EPR reference

EPR tools

Spin operators and matrices
Spin Hamiltonian
High-order spin operators
Linewidths in the fast-motion regime

Rotations and Euler angles
Line shapes

Nuclear isotope table

garlic

Synopsis: Computes isotropic and fast-motional cw EPR spectra of radicals in solution.

```
garlic(Sys,Exp)
garlic(Sys,Exp,Opt)
spec = garlic(...)
[B,spec] = garlic(...)
```

See also the user guide on how to use `garlic`.

Description

`garlic` computes isotropic and fast-motional cw EPR spectra of doublet radicals in solution, i.e., of spin systems with an electron spin S=1/2 coupled to an arbitrary number of nuclear spins I>=1/2 with small hyperfine couplings.

The composition of the spin system is specified in `Sys`, and the experimental settings are given in `Exp`.

`garlic` then returns the spectrum in `spec` and, if requested, a field range vector in `B` (in units of mT). If neither `B` nor `spec` are requested, `garlic` plots the simulated spectrum.

The following table lists all possible fields in the spin system structure `Sys`. Note that `Sys` here contains only a few fields of the general spin system structure as used by functions like pepper and salt. All fields except `n` are mandatory.

`g`	Array with 1, 2 or 3 elements, either isotropic g factor or principal values of an axial or orthorhombic g tensor.
	
	`Sys.g = 2.005;                 % isotropic g`
	`Sys.g = [2.001 2.004];         % axial g`
	`Sys.g = [2.001 2.002 2.004];   % orthorhombic g`

`Nucs`	String with comma-separated list of isotopes, e.g. `Sys.Nucs = '1H,13C'`.
`n`	Vector of number of equivalent nuclei, e.g. `Sys.n = [2,3]`, if the spin system contains two ^{1}H and three ^{13}C nuclei. Can be omitted if all nuclei in `Sys.Nucs` occur only once.
`A`	1xN or Nx3 array Vector of isotropic hyperfine couplings in MHz, e.g. `Sys.A = [10 52]`. Alternatively, array containing the principal values for all hyperfine tensors, one row per nucleus. E.g., `Sys.A = [15 15 40;-4 -3 7]` for two nuclei.
`lwpp`	1- or 2-element array of peak-to-peak linewidths (all in mT). • 1 element: `GaussianFWHM`. • 2 elements: `[GaussianFWHM LorentzianFWHM]`.
`lw`	1- or 2-element array of FWHM linewidths (all in mT). • 1 element: `GaussianFWHM`. • 2 elements: `[GaussianFWHM LorentzianFWHM]`.

For simulations in the fast motional regime, the principal values of the g and all A tensors have to be given. One more parameter in `Sys` specifies the speed of the rotational motion:

`tcorr`	Scalar Rotational correlation time for isotropic rotational diffusion, in seconds. See also the function fastmotion. If `tcorr` is omitted or set to zero, the isotropic limit spectrum is computed. For isotropic rotational motion, the correlation time `tcorr` and the diffusion rate `D` are related by `tcorr = 1/(6*D)`.
`logtcorr`	

	Base-10 logarithm of the correlation time, offering an alternative way to input the correlation time. If given, `tcorr` is ignored.

If `tcorr` or `logtcorr` is given, the fast-motional spectrum is computed. The necessary line widths are computed via the function fastmotion (for details see there). The resulting spectrum is additionally broadened by Lorentzian and Gaussian broadenings specified in `Sys.lw` using convolution, just as in the isotropic case.

If the inverse of the correlation time becomes similar in magnitude to the spectral anisotropy, the fast-motional model used by `garlic` (via fastmotion) is not valid anymore.

For simulating a multi-component mixture, `Sys` should be a cell array of spin systems, e.g. `{Sys1,Sys2}` for a two-component mixture. Each of the component spin systems should have a field `weight` that specifies the weight of the corresponding component in the final spectrum.

The following table lists all possible fields in the experiment structure `Exp`. Of these fields, only `mwFreq` is mandatory.

`mwFreq`	Spectrometer frequency, in GHz. E.g. `Exp.mwFreq = 9.5;` for X band.
`nPoints`	Number of points along field axis (default 1024)
`CenterSweep`	2-element vector `[center sweep]` with center field `center` and full field sweep range `sweep`, both in mT. If both `CenterSweep` and `Range` are not specified, the magnetic field range is automatically determined to cover the full spectral range.
`Range`	2-element vector `[minField maxField]` with lower and upper limit of field scan range in mT. `Range` is only used if `CenterSweep` is not given. If both `CenterSweep` and `Range` are not specified, the magnetic field range is automatically determined to cover the full spectral range.
`Harmonic`	Detection harmonic (0, 1 or 2), default is 1.
`ModAmp`	Modulation amplitude (peak-to-peak), in mT.
`mwPhase`	The reference microwave phase, in radians. 0 is pure absorption (default value), and `pi/2` is pure dispersion. `mwPhase` is used only if the convolutional broadening given in `Sys.lw` or `Sys.lwpp` has a Lorentzian component. `mwPhase` allows you to include absorption/dispersion admixture in the simulation.
`Temperature`	Gives the temperature of the spin system in the EPR experiment, in Kelvin. If given, Boltzmann populations are computed and included in the EPR line intensities. E.g., `Temperature = 298` corresponds to room temperature. If not given (or set to `inf), all transitions are assumed to have equal polarizations.`

`Opt` contains simulation options:

`Method`	Specifies the simulation method. `'exact'` selects the exact Breit/Rabi solution (see below). Any of `'perturb1'`, `'perturb2'`, `'perturb3'`, `'perturb4'`, `'perturb5'` selects perturbation theory of the corresponding order.

Algorithm

To compute resonance fields, `garlic` uses a fixed-point iteration based on the exact Breit-Rabi solutions for a S=1/2 with an arbitrary nuclear spin. This is superior to using perturbation expressions, since it gives resonance field values accurate to within numerical error.

Only allowed transitions are computed. If the hyperfine couplings are too large, `garlic` will refuse to run. All transition intensities are assumed to be equal.

Sets of equivalent nuclei are transformed into a coupled representation (see equivcouple). Non-equivalent groups of equivalent nuclei are treated sequentially, i.e. cross terms are neglected.

For the computation of fast-motional line widths, the function fastmotion is used.

Examples

Spectra from systems with many nuclei are easily simulated.

```
Sys = struct('g',2,'Nucs','1H,14N','A',[30,40],'n',[5 4]);
Sys.lwpp = [0 0.1]; % only Lorentzian broadening
Exp = struct('mwFreq',9.7);
garlic(Sys,Exp);
```

To simulate a radical spectrum with its ^{13}C satellite lines, just specify `'C'` instead of `'13C'` for the carbon nucleus, and EasySpin will automatically simulate the spectra of all isotope combinations, in this case 98.93% with ^{12}C and 1.07% with ^{13}C.

```
Sys.g = 2;
Sys.Nucs = '1H,1H,C';
Sys.n = [2 3 1];
Sys.A = [10 11 3];
Sys.lwpp = [0 0.01];
Exp.mwFreq = 9.7;
Exp.CenterSweep = [346.5 2.8];
garlic(Sys,Exp);
```

Zoom in to see the ^{13}C satellite lines.

A simple example of a spectral simulation in the fast motional regime using the rotational correlation time:

```
A = mt2mhz([5.8 5.8 30.8]/10);
Sys = struct('g',[2.0088 2.0061 2.0027],'Nucs','14N','A',A);
Sys.tcorr = 5e-9;
Exp = struct('mwFreq',9.5);
garlic(Sys,Exp);
```

pepper

Synopsis: Calculation of single-crystal and powder cw EPR spectra (solid-state).

```
pepper(Sys,Exp);
pepper(Sys,Exp,Opt);
spec = pepper(...);
[B,spec] = pepper(...);
[B,spec,trans] = pepper(...);
```

See also the user guide on how to use `pepper`.

Description

`pepper` calculates cw EPR spectra for powders, frozen solutions and single crystals.

There are up to three possible output arguments

- If no output argument is requested, `pepper` plots the simulated spectrum.
- `spec` contains the calculated spectrum or spectra.
- `B` is a vector of magnetic field abscissa values over which the spectrum was calculated in units of mT.
- `trans` is a list of level number pairs indicating the transitions which where included in the spectrum calculations. Level numbers refer to the energy levels of the Hamiltonian in ascending order, so level 1 is that which lowest energy and so on. If `spec` is a matrix, `spec(k,:)` is the spectrum of the transition `trans(k,:)`.

There are three arguments to the function, the last one optional. They are similar to those of the function resfields.

`Sys` is a spin system structure containing spin Hamiltonian parameters and line broadening parameters.

For simulating a multi-component mixture, `Sys` should be a cell array of spin systems, e.g. `{Sys1,Sys2}` for a two-component mixture. Each of the component spin systems should have a field `weight` that specifies the weight of the corresponding component in the final spectrum.

`Exp` contains experimental parameters such as the microwave frequency, the magnetic field range and temperature. Here is a full list of its fields:

`mwFreq`	Spectrometer frequency, in GHz. E.g. `Exp.mwFreq = 9.5;` for X band.
`CenterSweep`	*2-element vector* `[center sweep]` Contains center field and sweep width of the external magnetic field sweep range. Values should be in mT, e.g. `Exp.CenterSweep=[310 100]`. The magnetic field sweep range can be specified either in `CenterSweep` or in `Range`. If both are given, `CenterSweep` has precedence.
`Range`	*2-element vector* `[loField hiField]` Contains lower and upper limit of the external magnetic field sweep range. Values should be in mT, e.g. `Exp.Range=[260 360]`. The magnetic field sweep range can be specified either in `CenterSweep` or in `Range`. If both are given, `CenterSweep` has precedence.
`nPoints`	Number of points on the magnetic field abscissa axis. If not given, the default is 1024.
`Temperature`	*scalar* (default `inf`) or *vector* This field specifies populations for the states of the spin system, either directly or via a temperature. *Thermal equilibium:* `Temperature` is the temperature of the spin system in the EPR experiment, in Kelvin. If given, Boltzmann populations are computed and included in the EPR line intensities. E.g., `Temperature = 298` corresponds to room temperature. If not given (or set to `inf`), all transitions are assumed to have equal polarizations. *Non-equilibrium populations:* `Temperature` can also be used to specify non-equilibrium populations. For a spin system with `N` electron states (e.g. 4 for a biradical), it must be a vector with `N` elements giving the populations of the zero-field electron states, from lowest to highest in energy. E.g., if `Temperature = [0.85 0.95 1.2]` for an S=1 system, the population of the lowest-energy zero-field state is 0.85, and the highest-energy zero-field state has a population of 1.2. The populations of all nuclear sublevels within an electron spin manifold are assumed to be equal.
`Harmonic`	`0`, `1` (default) or `2` Harmonic of the detection. `1` and `2` specify the first and the second *derivative* of the absorption spectrum, respectively. `0` returns the absorption spectrum without differentiation. To explicitly include the effect of field modulation, use `Exp.ModAmp`.
`Mode`	`'perpendicular'` (default) or `'parallel'` Relative orientation of the microwave field B_1 with respect to the static magnetic field B_0. `'perpendicular'` means $B_1 \perp B_0$. `'parallel'` means that $B_1 \| B_0$.
`ModAmp`	Modulation amplitude (peak-to-peak), in mT.
`mwPhase`	The reference microwave phase, in radians. 0 is pure absorption (default value), and `pi/2` is pure dispersion. `mwPhase` is used only if the convolutional broadening given in `Sys.lw` or `Sys.lwpp` has a Lorentzian component. `mwPhase` allows you to include absorption/dispersion admixture in the simulation.

`Orientations`	3xN array of real
	Specifies single-crystal orientations for which the EPR spectrum should be computed. Each column of `Orientation` contains the three Euler rotation angles `[phi;theta;chi]` of the magnetic field in a molecule fixed frame. If `Orientation` is empty or not specified, the full powder spectrum is computed.
	`Exp.Orientations = [0;0;0];              % crystal with z axis aligned with B0`
	`Exp.Orientations = [0;pi/2;0];           % crystal with z axis perpendicular to B0`
	`Exp.Orientations = [0 0 0;0 pi/2 0].';   % two orientations`
	`Exp.Orientations = [];                   % powder`

`CrystalSymmetry`	Specifies the symmetry of the crystal, if single-crystal spectra to be simulated (that is, if `Exp.Orientations` is specified). The crystal symmetry can be either the number of the space group (between 1 and 230), the symbol of the space group (e.g. `'P21212'` or the symbol for the point group

	(e.g. `'C2h'` or `'2/m'`).
	
	`Exp.CrystalSymmetry = 'P21/m'; % space group symbol`
	`Exp.CrystalSymmetry = 11;      % space group number (between 1 and 230)`
	`Exp.CrystalSymmetry = 'C2h';   % point group, Schönflies notation`
	`Exp.CrystalSymmetry = '2/m';   % point group, Hermann-Mauguin notation`
	
	When `CrystalSymmetry` is given, `pepper` automatically computes the spectra of all symmetry-related sites in the crystal. If `CrystalSymmetry` is not given, `pepper` assumes space group 1 (P1, point group C1), which has only one site per unit cell.

`Ordering`	*scalar* (default: zero) or *function handle*
	
	If a number is given in this field, it specifies the orientational distribution of the paramagnetic molecules in the sample.
	
	If not given or set to zero, the distribution is isotropic, i.e. all orientations occur with the same probability. If it is given, the orientational distribution is non-isotropic and computed according to the formula $P(\theta) = \exp(\lambda(3 \cos^2\theta - 1)/2)$, where θ is the angle between the molecular z axis and the static magnetic field, and λ is the number specified in `Exp.Ordering`.
	
	Typical values for λ are between about -20 and +20. For negative values, the orientational distribution function is maximum at $\theta = 90°$, for positive values at $\theta = 0°$ and $\theta = 180°$. The larger the magnitude of λ, the sharper the distributions.
	
	To plot a distribution depending on λ, use
	
	`lambda = 5;`
	`theta = linspace(0,pi,1001);`
	`p = exp(lambda*plegendre(2,0,cos(theta)));`
	`plot(theta*180/pi,p);`
	
	If `Exp.Ordering` is a function handle, `pepper` will use the user-supplied function to obtain the orientational distribution. It calls the function with two vector arguments, `phi` and `theta` (in radians). The function must return a vector `P` containing probabilities for each orientation, that is `P(k)` is the probability of finding the paramagnetic molecules with orientation specified by `phi(k)` and `theta(k)`. Here is an example with an anonymous function:
	
	`Exp.Ordering = @(phi,theta) gaussian(theta,0,15/180*pi);`
	
	Of course, the function can also be written and stored in a separate file, e.g. `myori.m`. Then use `Exp.Ordering = @myori`.
	
	When using a custom orientational distribution, make sure that the symmetry used in the simulation corresponds to the symmetry of the distribution. If the distribution is very narrow, increase the number of knots in the options structure.

`mwFreq` and `Range` have to be provided by the user, all other fields are optional and have default values.

The structure `Opt` collects computational parameters. `Opt` need not be specified, in which case default values for all fields are used. The field names and their possible values are listed below.

`Verbosity`	Determines how much information `pepper` prints to the screen. If `Opt.Verbosity=0`, `pepper` is completely silent. 1 logs relevant information, 2 gives more details.
`Output`	`'summed'` (default) or `'separate'` Determines in what form the spectrum is returned. If set to `'separate'`, `pepper` returns one spectrum for each transition in a matrix `spec`. The transition spectra are along the rows. `spec(k,:)` is the spectrum of transition k. If `'summed'` is specified, the total spectrum is returned in `spec` as a vector.
`nKnots`	`[N1]` or `[N1 N2]` Determines the number of orientations (knots) in a powder simulation for which spectra are calculated.

	• `N1` gives the number of orientations between θ=0° and θ=90° for which spectra are explicitely calculated using the physical theory. Common values for `N1` are between 10 (10° increments) and 91 (1° increments). The larger the anisotropy of the spectrum and the narrower the linewidth, the higher `N1` must be to yield smooth powder spectra.
	• `N2` is the refinement factor for the interpolation of the orientational grid. E.g. if `N2=4`, then between each pair of computed orientations three additional orientations are calculated by spline interpolation. Values higher than 10 are rarely necessary. If `N2` is not given, a default value is used.
	
	`Opt.nKnots = 91;       % 1° increments, no interpolation`
	`Opt.nKnots = [46 0];   % 2° increments, no interpolation`
	`Opt.nKnots = [31 6];   % 3° increments, 6-fold interpolation (giving 0.5° increments)`

`Symmetry`	`'auto'` (default), `'Dinfh'`, `'D2h'`, `'C2h'` or `'Ci'` Determines the symmetry used for the powder simulation. Based on this the set of orientations for which spectra are computed is chosen. `'Dinfh'` corresponds to a line from θ=0° to &theta=90° (with φ=0°), `'D2h'` to one octant, `'C2h'` to two octants, and `'Ci'` to one hemisphere (four octants). `auto` is the default, meaning that `pepper` determines the correct symmetry automatically from the given spin system. With any other setting, `pepper` is forced into using the specified symmetry, even if it is incorrect for the spin system. See also symm.
`Transitions`	*mx2 vector of integers* Determines manually the level pairs which are used in the spectrum calculation. If given, `pepper` uses them and skips its automatic transition selection scheme. Level pairs are specified in `Transitions(k,:)` by the level numbers which start with 1 for the lowest-energy level.
`Threshold`	Specifies the threshold for `pepper`'s transition pre-selection. Any transition with an estimated relative average amplitude less than this number is not included in the calculation. The relative average amplitude of the strongest transition is 1, the default is `1e-4`. The pre-selection is an approximate procedure, and it might miss transitions for complicated spin systems. In these cases, setting it to zero will include all transitions in the computation.
`Intensity`	`'on'` (default) or `'off'` With `'on'`, transition rates, i.e. line intensities, are computed correctly. Allowed transitions will be more intense then quasi-forbidden ones. `'off'` simply sets all transition rates of all transitions to 1. Allowed and forbidden transitions will have the same intensity. Be very careful when switching this option to `'off'`! The resulting spectra are not correct.
`Method`	`'matrix'` (default), `'perturb'`, `'perturb1'`, `'perturb2'`, `'hybrid'` Determines how `pepper` computes the resonance fields. • `'matrix'` indicates matrix diagonalization. This method is very reliable and accurate and works for spin systems with any number of spins. All interactions, including quadrupole, are included in the computation. • `'perturb1'` indicates first-order perturbation theory, and `'perturb'` or `'perturb2'` indicates second-order perturbation theory. These methods ares limited to spin systems with one electron spin 1/2 (and possibly some nuclei). In addition, nuclear Zeeman and nuclear quadrupole terms are neglected, and only allowed transitions are computed. The resulting spectrum is reasonably correct only for small hyperfine couplings (e.g. organic radicals). • `'hybrid'` indicates matrix diagonalization for all the electron spins, and perturbation treatment for all nuclei, using effective nuclear sub-Hamiltonians for each electron spin manifold. `'matrix'` is the method of choice for systems with only a few low-spin nuclei (and any number of electron spins). For spin systems with many nuclei and small hyperfine couplings, simulations using perturbation theory are orders of magnitude faster. `'hybrid'` is the method of choice for systems with several large electron spins coupled to several nuclei such as in oligometallic clusters.
`IsoCutoff`	For isotope mixtures, determines which isotopologues to include in the simulation. Any isotopologue with relative abundance smaller than `IsoCutoff` is excluded. The default value is 1e-4.

Algorithm

Spectra are calculated over a triangular orientational grid using resfields to obtain the resonance field positions and amplitudes. For each orientation line positions, and possibly widths and intensities, are evaluated.

This gridded data is then interpolated with cubic splines in a combined 1D/2D approach. Resampling of the spline surface gives much quicker many more position/intensity/width data than quantum-mechanical calculation.

Finally, the refined data are projected onto the magnetic field axis using a Delaunay triangulation of the resampled spline surfaces. Linear interpolative projection of these triangles yields an extremely smooth spectrum with very low powder simulation noise. In the case of full anisotropic width treatment, a simple sum-up of Gaussian line shapes is used instead of the projection.

Apart from the main steps above, there is an automatic transition selection, which works along the same line as the overall algorithm, except that its results are only used for determining which level pairs possibly contribute to the spectrum.

For line width calculations, Gaussian distributions are assumed both in the magnetic field and the frequency dimension. The overall line width for a given orientation is

$$\mathit{\Gamma}^2 = \mathit{\Gamma}_{\text{res}}^2 + \mathit{\Gamma}_{gA}^2 + \mathit{\Gamma}_{DE}^2$$

where $\mathit{\Gamma}_{\text{res}}$is the residual line width specified in `Sys.HStrain`, $\mathit{\Gamma}_{gA}$is the line width due to correlated g-A strain (`Sys.gStrain` and `Sys.AStrain`), and $\mathit{\Gamma}_{DE}$the width arising from D-E strain (`Sys.DStrain`).

Although quite robust and general, `pepper` still has some limitations.

- In the case of looping resonance fields, interpolation is not possible. If the spectrum is coarse, increase `Opt.nKnots`.
- In the case of looping resonance fields, there might appear bumps around the coalescence points in the spectrum. To get rid of them, increase the line widths or `Opt.nKnots`.

Examples

As an illustration, we explore the influence of various `pepper` options on the zeroth-harmonic (DC) spectrum of a simple orthorhombic system. First the spin system, the experiment at X-band and some options are defined. An anisotropic line width is included in the spin system.

```
Sys = struct('S',1/2,'g',[1.9 2 2.3]);
Exp = struct('CenterSweep',[325 80],'mwFreq',9.5,'Harmonic',0);
Opt = struct('Verbosity',1);
```

Next we compute spectra for some combinations of broadening parameters.

```
[x,y1] = pepper(Sys,Exp,Opt);
Sys.lw = 2;
y2 = pepper(Sys,Exp,Opt);
Sys.lw = 0;
Sys.HStrain = [170 40 50];
y3 = pepper(Sys,Exp,Opt);
```

The final plot reveals the differences between the spectra.

```
plot(x,y1/sum(y1),x,y2/sum(y2),x,y3/sum(y3));
legend('no broadening','convolution broadening','H strain');
```

Running the simulation

Solid-state cw EPR spectra of powders and single crystals are computed by the EasySpin function pepper. It can be called with two or three parameters and can return both field axis and spectrum.

```
pepper(Sys,Exp);                      % plots the spectrum
[Field,Spec] = pepper(Sys,Exp);       % returns the field axis and the spectrum
[Field,Spec] = pepper(Sys,Exp,Opt);   % additional simulation options in Opt
```

Don't forget the ; (semicolon) at the end of the line to suppress output to the screen.

The first argument `Sys` tells `pepper` all about the spin system, and the second argument `Exp` gives the experimental parameters. The third, optional, argument `Opt` contains settings concerning the simulation itself, like the number of orientations for powder simulations.

The outputs `Field` and `Spec` are arrays containing the magnetic field values and the spectrum, respectively. If no output is requested, `pepper` simple plots the spectrum. If the outputs are requested, `pepper` does not plot the spectrum, but you can plot it yourself using

```
plot(Field,Spec);
```

Setting up a simulation and running it takes only a few lines of code. A very simple one would be

```
Sys.g = [2 2.1];
Sys.lwpp = 0.5;
Exp.mwFreq = 9.5;
pepper(Sys,Exp);
```

This simulates and plots the 9.5 GHz EPR spectrum of an S=1/2 system with an axial g tensor. Copy and paste the code above to your MATLAB command window to see the graph.

The spin system

The first input argument to `pepper` is a structure specifying the spin system. It contains fields for the electron spin(s), the nuclear spins, and the various interaction parameters like g and hyperfine tensors.

`pepper` automatically assumes `S=1/2` for the spin quantum number. For systems with higher spin or more than one unpaired electron, the spin quantum number should be given in the field `S`.

```
Sys.S = 1;          % a triplet state
Sys.S = 5/2;        % for e.g. high-spin Mn2+ or high-spin Fe3+
Sys.S = [1/2, 1/2];  % for a biradical
```

The field `g` contains the principal values of the g tensor(s). A simple rhombic S=1/2 system (e.g., a low-spin Fe^{3+}) is

```
Sys.g = [1.8, 2, 2.1];
```

Nuclear spins are included by specifying `Nucs` (comma-separated list of nuclei) and `A` (array of hyperfine tensor principal values, in MHz).

```
Sys.Nucs = '2H';        % one 2H (deuterium) nucleus
Sys.A = [-1,-1,2]*4.2;  % hyperfine principal values in MHz
```

If the A tensor is tilted with respect to the molecular frame, the tilt angles can be provided via the field `Apa` (standing for "A principal angles")

```
Sys.Apa = [0 30 0]*pi/180; % [alpha beta gamma] in radians
```

The zero-field splitting is specified in the `D` field, in units of MHz. There are several different input possibilities:

```
Sys.D = 120;          % D = 120 MHz, E = 0
Sys.D = [120 15];     % D = 120 MHz, E = 15 MHz
Sys.D = [-25,-55,80]; % principal values of D tensor, in MHz
```

D and E are related to the principal values of the D tensor (see reference page on the zero-field splitting).

Details about all the spin Hamilton parameters can be found on the spin Hamiltonian reference page. It is also possible to include several electron spins. Refer to the page about spin system structures for details.

Broadenings

No cw EPR spectrum is infinitely sharp. Lines are usually broadened due to several reasons. `pepper` provides means to include several line broadening models in a simulation.

The simplest way to include line broadening is to convolute a stick spectrum with a (Gaussian or Lorentzian) lineshape after the end of the simulation. Such a convolution broadening is specified in the spin system field `lwpp`.

```
Sys.lwpp = 0.5;     % Gaussian broadening of 0.5 mT PP
Sys.lwpp = [0 2];   % Lorentzian broadening of 2 mT PP
Sys.lwpp = [1 2];   % Gaussian PP of 1mT + Lorentzian PP of 2 mT
```

The line width is in mT and refers to peak-to-peak (PP) line widhts. Instead, FWHM (full width at half height) line widths can be provided in the field `Sys.lw`.

```
Sys.lw = 0.5;     % Gaussian broadening of 0.5 mT FWHM
Sys.lw = [0 2];   % Lorentzian broadening of 2 mT FWHM
```

For details about line shapes and conversion formulas to/from FWHM and peak-to-peak widths, see the page on line shapes.

Physically, there are several origins for line broadening. Large contribution to broadening often comes from unresolved hyperfine couplings and from distributions in the various magnetic parameters lige g, A and D that result from structural variations from one paramagnetic center to the next.

To include effects from unresolved hyperfine couplings, an orientation-dependent phenomenological broadening can be specified in `HStrain`:

```
Sys.HStrain = [50 50 87];   % [along x, along y, along z], in MHz
```

Distributions in magnetic parameters are called *strains*. g and A strains are given in similar fields:

```
Sys.gStrain = [0.01 0.02 0.005];
Sys.AStrain = [10 10 30]; % in MHz
```

The three values in `gStrain` are the FWHM parameters of the Gaussian distributions of the respective g principal values given in `Sys.g`. `AStrain` is the same for the A tensor. The g and A strains are correlated.

Distributions of the D tensor values can be given in `DStrain`, where the first value is the width of the (scalar) D distribution, and the second is the width for the E distribution.

All these broadening parameters can be combined. However, usually a modelling of the broadening with `lwpp` or `HStrain` is absolutely sufficient.

Basic experimental settings

The second input argument, `Exp`, collects all experimental settings. Just as the spin system, `Exp` is a structure containing several fields.

For a simulation, Easyspin needs the spectrometer frequency in the field `mwFreq` in units of GHz.

```
Exp.mwFreq = 9.385;   % X-band
Exp.mwFreq = 34.9;    % Q-band
```

There are two ways to enter the magnetic field sweep range. Either give the center and the sweep width (in mT) in `Exp.CenterSweep`, or specify the lower and upper limit of the sweep range (again in mT) in `Exp.Range`.

```
Exp.CenterSweep = [340 80]; % in mT
Exp.Range = [300 380];      % in mT
```

On many cw EPR spectrometers, the field range is specified using center field and sweep width, so `Exp.CenterSweep` is the more natural choice.

`Exp.CenterSweep` and `Exp.Range` are only optional. If both are omitted, `pepper` tries to automatically determine a field range large enough to accomodate the full spectrum. If, on the other hand, you happen to give both `Exp.CenterSweep` and `Exp.Range`, `pepper` takes the values given in `Exp.CenterSweep` and ignores those in `Exp.Range`.

By default, `pepper` computes a 1024-point spectrum. However, you can change the number of points to a different value using

```
Exp.nPoints = 5001;
```

By default, `pepper` computes the first-harmonic absorption spectrum, i.e. the first derivative of the absorption spectrum. By changing `Exp.Harmonic`, you can request the absorption spectrum directly or the second-harmonic (second derivative) of it.

```
Exp.Harmonic = 0; % absorption spectrum, direct detection
Exp.Harmonic = 1; % first harmonic (default)
Exp.Harmonic = 2; % second harmonic
```

If you want to include effects of field modulation, use `Exp.ModAmp`

```
Exp.ModAmp = 0.2; % 0.2 mT (2 G) modulation amplitude, peak-to-peak
```

If you want to include the effect of the time constant, apply the function rcfilt to the simulated spectrum.

More experimental settings

For more advanced spectral simulations, `pepper` offers more configuration possibilities in the experimental parameter structure `Exp`.

Most cw EPR resonators operate in *perpendicular* mode, i.e., the oscillating magnetic field component of the microwave in the resonator is perpendicular to the static field. Some resonators can operate in *parallel* mode, where the microwave field is parallel to the static one. `pepper` can simulate both types of spectra:

```
Exp.Mode = 'perpendicular'; % perpendicular mode (default)
Exp.Mode = 'parallel';      % parallel mode
```

The polarizing effects of low sample temperatures can also be included in the simulation by specifying the temperature:

```
Exp.Temperature = 4.2; % temperature in Kelvin
```

With this setting, `pepper` will include the relevant polarization factors resulting from a thermal equilibrium population of the energy levels. For S=1/2 systems, it is never necessary to include the temperature. However, it is important in high-spin systems with large zero-field splittings, and in coupled spin systems with exchange couplings.

Occasionally, the EPR absorption signal has a small admixture of the dispersion signal. This happens for example when the microwave phase in the reference arm is not absolutely correctly adjusted. `pepper` can mix dispersion with absorption if a Lorentzian broadening is given:

```
Sys.lwpp = [0.2 0.01];          % Lorentzian broadening (2nd number) required

Exp.mwPhase = 0;                % pure absorption
Exp.mwPhase = pi/2;             % pure dispersion
Exp.mwPhase = 3*pi/180;         % 3 degrees dispersion admixed to absorption
```

Powders and crystals

If not specified otherwise, `pepper` computes a powder spectrum. But it is as well straightforward to simulate spectra for a single crystal. The orientation (or orientations if more than one) of the single crystal can be provided in the experiment structure field `Exp.Orientations`. This field should contain the tilt angles between molecular and laboratory frame (right-handed coordinate system with z along the static field and x along the microwave magnetic field), one set of three angles per column.

For a crystal with its molecular frame aligned with the laboratory frame, the setting is

```
Exp.Orientations = [0;0;0];
```

If you need more than one crystal at the same time, then just specify more than one orientation.

```
Exp.Orientations = [0 0 0;0 pi/4 0].';
```

In many crystals, there are several symmetry-related sites with identical paramagnetic centers differing only in their orientation in the crystal. You can tell `pepper` about this by providing the crystal symmetry in the field `Exp.CrystalSymmetry`, e.g.

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

With the crystal symmetry given, `pepper` not only computes the spectrum for the orientation given in `Exp.Orientations`, but also for all symmetry-related sites.

If `Exp.Orientations` set to `[]` (an empty array), `pepper` simulates the powder spectrum.

Simulation options

The third input argument to `pepper` contains simulation options. All of them have reasonable default values, but sometimes it might be necessary to change one of them. In the following the most important ones are presented.

If you want `pepper` to print information about the simulation to the command window during the computation, use

```
Options.Verbosity = 1;
```

`'Verbosity'` tells `pepper` how much of progress information to show in the command window. 0 (the default) suppresses all output, 1 is normal intormation, and 2 prints more information, relevant only for debugging.

Another useful option is `nKnots`, which determines how many orientations `pepper` will include in the simulation of a powder spectrum. If this value is too low, the spectrum shape contains ripples. `nKnots` is the number of orientations between the z axis and the x axis (between theta = 0 and theta = 90 degrees).

```
Options.nKnots = 31; % corresponds to 3-degree increments
```

The higher `nKnots`, the finer the orientational grid. The default value of 19 (5-degree increments) is appropriate for most systems. A value larger than 181 (0.5-degree increments) is rarely needed.

After having computed the spectrum for a number of orientations specified by `nKnots`, the simulation function interpolates these spectra for additional orientations before summing up all spectra. This interpolative refinement can be configured with a second number in `nKnots`. `nKnots = [19 4]` means that `pepper` interpolates additional 4 spectra between two adjacent orientations evaluated.

```
Options.nKnots = [19 10];  % massive interpolation
Options.nKnots = [19 0];   % no interpolation
```

The option `Output` can be used to determine the form in which `pepper` returns the spectral data.

```
% single crystal: orientations separately
% powders: transitions separately
Options.Output = 'separate';

% sum over all orientations and transitions
Options.Output = 'summed';
```

There are more option fields available. For details, see the documentation page on pepper.

Multiple components

Often, an EPR spectrum shows a mixture of spin species. To simulate these spectra, each of the component spectra has to be simulated and added with the appropriate weight (depending on spin concentration) to the total spectrum.

This can be done automatically by `pepper`. Just provide the component spin systems with their weights as a cell array (in curly braces) to `pepper`. For example, here is the simulation of a very simple two-component mixture with 2:1 ratio of spin concentrations.

```
Sys1.g = 2;
Sys1.lwpp = 1;
Sys1.weight = 2;
Sys2.g = 2.1;
Sys2.lwpp = 0.8;
Sys2.weight = 1;

Exp.mwFreq = 9.5;
Exp.Range = [300 360];

pepper({Sys1,Sys2},Exp);
```

You don't have to specify `Sys.weight` - if it's absent it is assumed to be 1. These weights are absolute, i.e. a simulation with `Sys.weight=20` yields a spectrum that is 10 times more intense than the one obtained with `Sys.weight=2`. There is no limit to the number of components in a simulation.

Systems with several nuclei

`pepper` uses matrix diagonalization as thed default method for simulating spectra. For systems with several nuclei this can be very time-consuming. To accelerate such computations, `pepper` provides first- and second-order perturbation theory as an alternative methods. The relevant simulation option that tells EasySpin about is `Opt.Method`.

As an example, we look at the simulation of the spectrum of Cu^{2+} porphyrin.

```
Sys.S = 1/2;
Sys.g = [2 2.2];
Sys = nucspinadd(Sys,'63Cu',[50 500]);
A = [20 30];
Sys = nucspinadd(Sys,'14N',A);
Sys = nucspinadd(Sys,'14N',A);
Sys = nucspinadd(Sys,'14N',A);
Sys = nucspinadd(Sys,'14N',A);
Sys.lwpp = 0.5;
```

With matrix diagonalization (`Opt.Method='matrix'`, which is the default), the simulation needs several hours. With second-order perturbation theory (`Opt.Method='perturb2'`), the simulation is orders of magnitude faster, but potentially less accurate. We can compare the full matrix diagonalization to the perturbation simulation.

```
Exp.mwFreq = 9.5;
Exp.Range = [260 380];
Opt.Method = 'matrix';
[x,y1] = pepper(Sys,Exp,Opt);
Opt.Method = 'perturb2';
[x,y2] = pepper(Sys,Exp,Opt);
plot(x,y1,'r',x,y2,'b');
```

Non-equilibrium populations

`pepper` can handle both thermal equilibrium and non-equilibrium populations. Both are specified in the field `Temperature` of the experimental settings structure.

For thermal equilibrium, just give the temperature in Kelvin:

```
Exp.Temperature = 77; % 77K, boiling point of liquid nitrogen
```

For non-equilibirum populations, `Temperature` must be a vector. If the spin systems contains *N* electron states, then this vector must contain *N* elements, each specifying the population of one of the electron states at *zero field*, sorted according to their energy from lowest to highest.

E.g., an organic triplet with S=1 and I=1 has 3 electron states, each further split into three sublevels by the coupling to the nuclear spin. The population vector in this case should contain three elements:

```
Exp.Temperature = [0.6 0.8 1.1]; % highest state is most populated
```

This specifies that all the sublevels of the lowest zero-field electron states have a population of 0.6, etc. The sublevels of the highest-energy zero-field electron state have a population of 1.1. The populations don't have to be normalized, `pepper` takes care about that.

To compute the state populations for a non-zero field state, `pepper` decomposes it into a linear combination of zero-field states and combines the zero-field populations using the resulting linear combination coefficients.

A simple example of a non-equilibrium triplet system is

```
Sys.S = 1; Sys.g = 2; Sys.lw = 0.2;
Sys.D = 100;
Exp.mwFreq = 9.5; Exp.Range = [320 360]; Exp.Harmonic = 0;
Exp.Temperature = [0.5 0.6 0.9];
pepper(Sys,Exp);
```

Partially ordered systems

In powders and frozen solutions (disordered systems), paramagnetic molecules are randomly oriented in the sense that any orientation can occur with equal probability. In other situations, like in polymers, biomembranes or liquid crystals, the paramagnetic molecules may be partially aligned or ordered, so that some orientations are more probable than others. As a result, the spectra of such partially ordered systems are different from those of powders and frozen solutions.

pepper can include partial ordering in the spectral simulation. For this, set a value (different from zero) in the experiment structure field `Exp.Ordering`. It is a number which specifies the nature and the degree of the ordering.

If it is zero, no ordering is used. Positive values mean that the molecules are partially aligned so that the magnetic field vector is mostly approximately parallel to the molecular z axis. Negative values specify preferential orientation away from the z axis, that is, orientations with the magnetic field vector in the xy plane are more probable.

```
Exp.Ordering = 0;  % all orientations equally populated
Exp.Ordering = -1; % slightly preferential orientation in the xy plane
Exp.Ordering = 10; % strongly aligned along the z axis
```

The ordering function used is very simple (see the pepper documentation for more information), but sufficient for most cases. Here is a simulation of a sample where the molecules are preferentially oriented so that the molecular z axis is close to the magnetic field vector:

```
Sys.g = [2 2 2.2]; Sys.lwpp = 1;
Exp.mwFreq = 9.5;
Exp.Ordering = +2;
pepper(Sys,Exp);
```

You can also define your own custom orientational distribution in a separate function and supply to `pepper` as a function handle in `Exp.Ordering`. See the pepper documentation for details. Written as an anonymous function, the built-in orientational distribution is equivalent to

```
Exp.Ordering = @(phi,theta) exp(lambda*plegendre(2,0,cos(theta)));
```

where the EasySpin function plegendre returns the associated Legendre polynomial and `lambda` corresponds to the number given in `Exp.Ordering`.