OOP-ESEEM trEPR stick spectra

Gianluca Marcozzi

August 2024

Chapter 1

Calculations of powder trEPR spectrum

1.1 Parameters

We try to simulate the powder trEPR spectrum of PSI. The used parameters are:

- g1 = [2.0030, 2.0026, 2.0023], g2 = [2.0062, 2.0051, 2.0022] for P700+ and A1- respectively;
- Euler angles [-10, -128, -83] to transform from the reference frame of P700+ to the one of A1-;
- dipolar interaction and exchange coupling dip = -0.177 mT and J = 0.001 mT (following the spin Hamiltonian convention used by Zech);
- z_D, direction of the spin-spin interaction, corresponding to the negative x-axis of the A1- frame of reference (hence Euler angles [0, 90, 0] to go from the dipolar frame of reference to the A1- frame of reference);
- hyperfine coupling of the A1- radical with three equivalent hydrogen nuclei, A = [9, 9, 12.8] MHz and [-60, 90, 0] Euler angles to transform from the frame of the hyperfine to the one of A1-.

1.2 Transition frequencies and intensities

To calculate the powder average, we keep our frame fixed with the frame of A1- and calculate the trEPR stick spectra for different orientations of the external magnetic field. As explained in Zech's thesis [1] chapter 4, the stick spectrum at a given orientation of the magnetic field is going to be similar to what shown in Fig. 1.1. As shown in Fig. 1.1, the position of the resonances is given by the values of J, dip and:

$$\omega_0 = \frac{\mu_B B_0}{h}(g1 + g2)/2 + \frac{1}{2}\sum_j (A_{1,j} + A_{2,j})m_j, \tag{1.1}$$

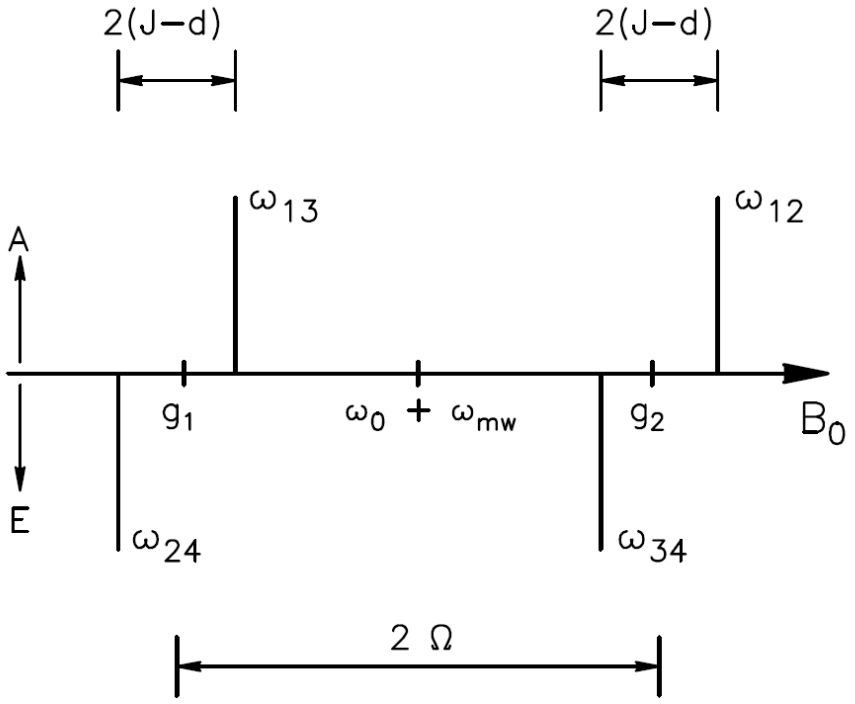

Figure 1.1: Stick spectrum for an orientation such that d = dip·(cos(θ_D)−1/3) < 0. Taken from [1], Fig 4.2.

and:

$$\Omega = \sqrt{\Delta\omega^2 + (J + d/2)^2}, \tag{1.2}$$

where:

$$\Delta\omega = \frac{\mu_B B_0}{h}(g1 - g2)/2 + \frac{1}{2}\sum_j (A_{1,j} - A_{2,j})m_j. \tag{1.3}$$

The absolute value of the intensity is the same for all the transitions at a given orientation. In particular:

$$I_{12} = -I_{34} = I_{13} = -I_{24} \propto \frac{\Delta\omega^2}{\Omega^2} = \sin^2(2\alpha), \tag{1.4}$$

where α is the mixing angle used in Zech's thesis, defined as:

$$\sin(2\alpha) = \frac{\Delta\omega}{\Omega}, \cos(2\alpha) = \frac{J + d/2}{\Omega}, \quad \tan(2\alpha) = \frac{\Delta\omega}{J + d/2}. \tag{1.5}$$

Note that $|\alpha| \to \pi/4$ for weakly interacting spin pair while $\alpha \to 0$ for strongly interacting spin pair.

1.3 Computational steps

1. Calculate d, effective g-values and effective hyperfine coupling for each orientation

2. Calculate ω_0, $\Delta\omega$ and Ω
3. Find the transition frequencies
4. Calculate the lineshapes for each transition with a certain linewidth and the correct intensity
5. Sum all the lines and average over all orientations

1.3.1 Calculate orientation-dependent parameters

We define a grid of angles θ and ϕ. These correspond to the polar and azimuthal angles of the external magnetic field with respect to the frame of reference of A1- (which we keep fixed).
An example of the code where a step of 3 degree is used both for θ and for ϕ:

```
% Theta and phi grid
thetas = (0:3:180)*pi/180;
nTheta = numel(thetas);
phis = (0:3:360)*pi/180;
nPhi = numel(phis);
```

In this case the array thetas will be thetas = [0, 3, 6, ..., 180]^o and the number of angles will be nTheta = 61.
From here we can calculate the effective g-values at each orientations. First we calculate the versor n indicating the direction of B_0:

```
% Direction of B0
clear('nVers')
nVers(1, :, :) = sin(thetas')*cos(phis);
nVers(2, :, :) = sin(thetas')*sin(phis);
nVers(3, :, :) = cos(thetas')*ones(1, nPhi);
```

The matrix nVers will be a 3D matrix of dimensions 3 x nTheta x nPhi. For example, for the 61st value of the first dimension ($\theta = 180$ degrees) and the first value of the second dimension ($\phi = 0$ degrees), we will obtain nVers(:, 61, 1) = [0, 0, -1], as expected.
Next step: calculate g2, that is the effective g-value of A1-. Since the g2 tensor is diagonal in this reference frame, for each orientation it will be:

$$g_{\text{eff}} = \sqrt{(g_{xx} \cdot n_x)^2 + (g_{yy} \cdot n_y)^2 + (g_{zz} \cdot n_z)^2}. \tag{1.6}$$

We basically want to multiply element-wise g2 by the value of nVers for each orientation and then take the square root of the sum of the squares of the projections. The code looks like:

```
% Effective g-values
g2 = squeeze(sqrt( sum( (Sys.g(2, :)'.*nVers).^2)));
```

First, the element-wise multiplication is carried out: Sys.g(2, :)' is the column vector 3 x 1 of $[g_{xx}, g_{yy}, g_{zz}]$. The operator .* does the element-wise multiplication at each fixed θ and ϕ value. As a consequence, the result of Sys.g(2, :)'.*nVers is still a 3 x nTheta x nPhi matrix. Afterwards these values are squared (also element wise), then they are summed along the first dimension

and then the square root is calculated. At this point the result is a 1 x nTheta x nPhi matrix. The function squeeze() makes it a nTheta x nPhi matrix corresponding to the effective g-values for each orientation.
For the P700+ radical the process to calculate the effective g-values is the same, but we first need to transform the g-tensor to the frame of A1-. The code is:

```
euMatrixg1 = erot(Sys.gFrame(1, :));
g1TensorInFrame2 = euMatrixg1'*diag(Sys.g(1, :))*euMatrixg1;
g1 = squeeze(sqrt( sum( (pagemtimes(g1TensorInFrame2, nVers)).^2)));
```

Here erot(Sys.gFrame(1, :)) is the rotation matrix with Euler angles Sys.gFrame(1, :) = [-10, -128, -83]. It is applied transposed on the left (' sign) and not transposed on the right side of the diagonal matrix (diag(Sys.g(1, :))). Afterwards we calculate the effective g-value multiplying the g-tensor by the nVers for each orientation. This is done by the function pagemtimes(), that enables multiplications of 3 x 3 matrices by 3 x 1 vectors for each value of θ and ϕ. The rest is analogous to the A1- case.
For the dipolar interaction:

```
% Dipolar interaction
zD = erot(Sys.eeFrame)*[0, 0, 1]';
dd = squeeze(dipFunc(dip, nVers, zD));
```

We first transform the z_D vector to the A1- frame using the Euler angles Sys.eeFrame = [0, 90, 0], therefore obtaining zD = [-1, 0, 0]. Afterwards we the function dipFunc to calculate the value of the dipolar coupling for every orientation. The function is defined as:

```
% Dipolar interaction
% Makes use of the fact that cos(thetaD) = dotProduct(B0, zD)
% Expected input:
%   dip:     1 x 1
%   nVers:   3 x nTheta x nPhi
%   zD:      3 x 1
dipFunc = @(dip, nVers, zD) dip*((sum(nVers.*zD)).^2 - 1/3);
```

Finally we calculate the effective hyperfine interaction for each orientation.

```
%
% Hyperfine interaction with a number nNuc of equal nuclei
%
% For spin 2
euMatrixHfi2 = erot(Sys.AFrame(1, 4:6));
Ahfi2TensorInFrame2 = euMatrixHfi2'*diag(Sys.A(1, 4:6))*euMatrixHfi2;
Ahfi2 = squeeze(sqrt( sum( (pagemtimes(Ahfi2TensorInFrame2', nVers)).^2)));
```

We calculate it only for spin 2, being the radical on A1-, because that is the electron that should interact with the nNuc = 3 identical hydrogen atoms. erot(Sys.AFrame(1, 4:6)) is the rotation matrix using Euler angles [-60, 90, 0] and it is used to transform the values of Sys.A(1, 4:6) = [9, 9, 12.8] MHz to the reference frame of A1-. Afterwards, the effective hyperfine Ahfi2 is calculated, analogously to Eq. 1.6.

1.3.2 Calculate ω_0, $\Delta\omega$, Ω and the transition frequencies

We want to calculate ω_0, $\Delta\omega$ and Ω using Eq. 1.1, Eq. 1.3 and Eq. 1.2. First we calculate ω_0 and $\Delta\omega$ neglecting the hyperfine couplings:

```
w0 = g2wFunc(B0, (g1 + g2)/2);   % GHz
deltaw = g2wFunc(B0, (g1 - g2)/2);   % GHz
```

where:

```
% Expected input: B0 in mT
% Output: frequency nu in GHz
g2wFunc = @(B0, g) bmagn*B0/planck*g*1e-12;
```

Afterwards we define $A_+ = A_1 + A_2$ (in our case $A_1 = 0$ because there is no interaction between electron 1, that is the radical on P700+, and the nuclei), we adjust the shapes of the vectors and then we add to ω_0 the following values: $[-\text{nNuc}/2]A_+$, $\cdot[-(\text{nNuc}-1)/2]A_+$, ... , $[(\text{nNuc}-1)/2]A_+$, $[\text{nNuc}/2]A_+$. We obtain therefore the values of Eq. 1.1 for each orientation.
We follow analogous steps to obtain $\Delta\omega$ as in Eq. 1.3.
The code look like:

```
% Shift resonances due to hyperfine interaction
Aplus = (Ahfi1 + Ahfi2)/2*1e-3;
% Make w0 have size [nHfiLine, nTheta, nPhi]
w0 = insertDimensionPos1(w0, nHfiLine);
% Make Aplus have size [1, nTheta, nPhi]
Aplus = reshape(Aplus, [1, size(Aplus)]);
% Multiply by all possible values of sum of quantum number of the nuclei
% Final size = [nHfiLine, nTheta, nPhi]
Aplus = pagemtimes((-nNuc/2:1:nNuc/2)', Aplus);
w0 = w0 + Aplus;
% Same for deltaw and Aminus
Aminus = (Ahfi1 - Ahfi2)/2*1e-3;
deltaw = insertDimensionPos1(deltaw, nHfiLine);
Aminus = reshape(Aminus, [1, size(Aminus)]);
Aminus = pagemtimes((-nNuc/2:1:nNuc/2)', Aminus);
deltaw = deltaw + Aminus;
```

In our case for example, we have nNuc = 3, which means that we expect a number of lines equal to $\text{nHfiLine} = 2 \cdot \text{nNuc} \cdot 1/2 + 1 = \text{nNuc} + 1 = 4$. The distance of the lines from ω_0 will be: $-3/2 \cdot A_+$, $-1/2 \cdot A_+$, $1/2 \cdot A_+$, $3/2 \cdot A_+$. That is what the code is doing.
We must take into consideration that the intensity of the lines will follow the Pascal triangle. Therefore we calculate the Pascal triangle:

```
% Pascal factors because some values of the sum of the quantum number come
% up more than others
pascalMatrix = pascal(nHfiLine);
pascalFactor = pascalMatrix(nHfiLine:nHfiLine - 1:end - 1);
% Antidiag
```

With the Matlab built-in function pascal(n) we obtain a square matrix, where each super anti-diagonal corresponds to a row of the Pascal triangle. Since

we are interested in the row number nHfiLine, we generate the square pascal(nHfiLine) and then we calculate the antidiagonal and store it in the vector pascalFactor. This will be useful later to properly calculate the intensity of the lines. Finally we calculate Ω:

```
Omega = OmegaFunc(deltaw, JJ, insertDimensionPos1(dd, nHfiLine));
% GHz
```

where:

```
% Expected input: deltaw in GHz, J in MHz, d in MHz
% Output: Omega in GHz
OmegaFunc = @(deltaw, J, d) sqrt(deltaw.^2 + (J*1e-3 + d/2*1e-3).^2);
```

The position of each resonance depends on the difference between the energies of the eigenstates of the Hamiltonian. Following Zech's thesis [1]:

```
dipInteraction = (JJ - insertDimensionPos1(dd, nHfiLine))*1e-3;
% GHz
wReson(1, :, :, :) = w0 - dipInteraction - (Omega);   % w12
wReson(2, :, :, :) = w0 + dipInteraction - (Omega);   % w34
wReson(3, :, :, :) = w0 - dipInteraction + (Omega);   % w13
wReson(4, :, :, :) = w0 + dipInteraction + (Omega) ;   % w24
```

The function insertDimensionPos1 is used to have proper dimensions and can be ignored here. The matrix wReson has dimensions [4, nHfiLine, nTheta, nPhi].

1.3.3 Calculate the lineshapes for each transition with a certain linewidth and the correct intensity

At this point we calculate the lines for each of these transitions:

```
% Lineshapes for each transition
trSignal = ...
        gaussiantransitions(xxSim', wReson, trlw, "fwhm");
```

where guassiantransitions returns the gaussian lineshapes in the range xxSim centered at wReson with FWHM equal to trlw. In our case, trlw is around [0.7, 0.7, 0.35, 0.35] mT, that means that the transitions of P700+ have larger linewidth than the ones of A1-.
The intensity of each of these lines is determined by Eq. 1.4:

```
intensityReson = 1/8*(deltaw.^2)./(Omega.^2);
% Sign of intensityReson alternates for the transitions w12, w34, w13, w24
intensityReson = [1; -1; 1; -1].*insertDimensionPos1(intensityReson, 4);
```

The normalization factor such that the integral of each of these lines is equal to the intensities of Eq. 1.4 is equal to:

```
% Normalization to account for possible different linewidths
intensityNorm = insertDimensionPos1(intensityReson, 1) ./ sum(trSignal);
```

Finally we multiply the lineshapes by the normalization factor and the Pascal triangle factors:

```
trSignal = trSignal.*intensityNorm.*insertDimensionPos1(pascalFactor, 1);
```

1.3.4 Sum all the lines and average over all orientations

The final part of the code is:

```
trSignalSumHfi = squeeze(sum(trSignal, 3)); % Sum over hfi lines
trSignalSum = squeeze(sum(trSignalSumHfi, 2)); % Sum over transitions

solidAngleWeight = sin(thetas)/sum(sin(thetas')*ones(1, nPhi), 'all');
trSignalPowder = sum(squeeze(sum(trSignalSum.*solidAngleWeight, 3)), 2);
```

The matrix trSignalSumHfi is a [nAx, 4, nTheta, nPhi] matrix, where the lineshapes due to the hyperfine have been summed. Here nAx is the number of points of the frequency sweep that we are calculating (usually more than 100, I use 1024 or 2048). Each line of trSignalSumHfi is long nAx and is equivalent to the spectrum of the separate transitions $1 \to 2$ or $3 \to 4$ or $1 \to 3$ or $2 \to 4$ for each orientation. Note that the Pascal triangle factors were already taken into consideration.
The matrix trSignalSum is a [nAx, nTheta, nPhi] matrix, where the lineshapes due to the four transitions have been summed. This matrix shows the crystal spectrum for each orientation.
Finally, trSignalPowder is an array long nAx, where all the contribution have been averaged over all θ and ϕ values with weight equal to $\sin(\theta)$ (normalized by the sum of $\sin(\theta)$ over the all sphere, neglecting global proportionality factors that play no role in determining the lineshape of the spectrum).

1.4 Comparison with Easyspin

We calculate the spectrum using our code with parameters:

- linewidth trlw = 0.42 mT = 12 MHz, same for all the transitions;
- all the other parameters as specified at the beginning of this document.

We compare it with a simulation obtained using Easyspinusing the same parameters (changing the sign of the dipolar coupling and dividing it by 1.5 due to a different definition of the spin Hamiltonian). The comparison is in Fig. 1.2.

1.5 Comparison with experimental data

We calculate the spectrum using our code with parameters:

- linewidth trlw = [0.84, 0.84, 0.42, 0.42] mT = [];
- all the other parameters as specified at the beginning of this document.

The calculation from our code is compared to experimental data in Fig. 1.3 (the spectrum is flipped and the axis is transformed to magnetic field axis). The high field part of the spectrum is not well reproduced. Nevertheless, the simulation is in agreement with the simulation reported in [2] Fig. 5, where the used linewidths were [0.7, 0.7, 0.35, 0.35] mT.

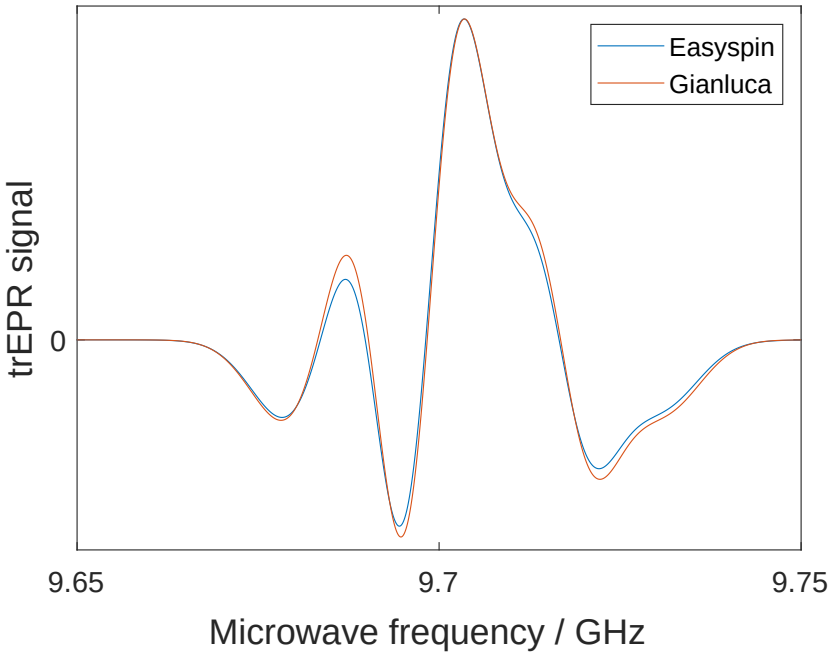

Figure 1.2: Comparison between Easyspin and our spectrum.

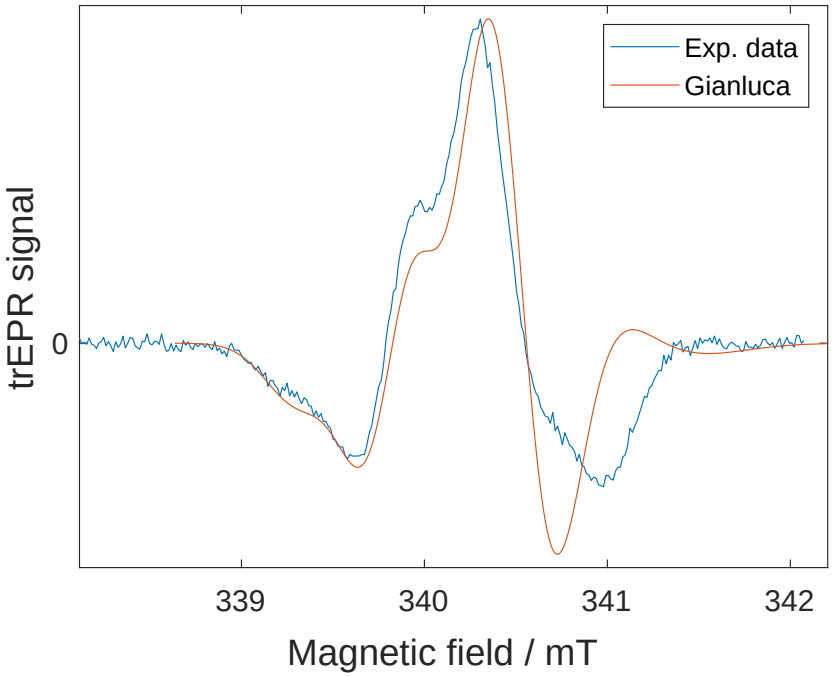

Figure 1.3: Comparison between experimental data and our spectrum.

Bibliography

[1] "Pulsed and transient electron paramagnetic resonance spectroscopy on light induced radical pairs in photosynthetic reaction centers". Aachen: Shaker, 1998.

[2] Andreas Kamlowski et al. "The Radical Pair State P7•+00A1•- in Photosystem I Single Crystals: Orientation Dependence of the Transient Spin-Polarized EPR Spectra". In: ().