Addendum B: ESR Spectrum of TEMPO

This addendum provides additional material to Sections 3.3.1 to 3.3.3 of Essentials of Dynamic Nuclear Polarization, Spindrift Publications, 2016—to be denoted henceforward shortly as *EofDNP*. It presents a more precise determination of the ESR spectrum of TEMPO dissolved in a glass of butanol.

As in *EofDNP* we determine the ESR spectrum in two steps. In the first step we calculate the spectrum of just the electron spin $S = \frac{1}{2}$ and the spin $I = 1$ of the ^{14}N nucleus in the nitroxide (NO$^\bullet$) group. In the second step we add the remaining interactions of the electron spin by convoluting this spectrum with a Gaussian.

In Section 3.3.2 of *EofDNP* we introduced two approximations in the first step. Except for A_{ZZ} we skipped all components of the hyperfine coupling between the electron spin $S = \frac{1}{2}$ and the spin $I = 1$ of the ^{14}N nucleus. Furthermore we skipped the Zeeman interaction of this nuclear spin. The more precise determination presented here does not invoke these two approximations and determines the ESR spectrum from the exact eigenstates and energy levels of the Hamiltonian describing the electron spin and the spin of the ^{14}N nucleus.

1. Principles of Calculating the Spectrum

The calculation of the exact ESR spectrum starts with the Hamiltonian

$$\mathscr{H} \;=\; -\,\hbar\mathbf{B}_0\cdot\boldsymbol{\gamma}\cdot\mathbf{S} \;-\; \hbar\gamma_I\mathbf{B}_0\cdot\mathbf{I} \;+\; \hbar\mathbf{I}\cdot\mathbf{A}\cdot\mathbf{S} \tag{1}$$

describing the electron spin $S = \frac{1}{2}$ and the spin $I = 1$ of the ^{14}N nucleus. Here the first term is the Zeeman interaction of the electron spin with the externally applied static magnetic field $\mathbf{B}_0$, the second term the Zeeman interaction of the nuclear spin with $\mathbf{B}_0$ and the last term the hyperfine interaction between the electron spin and the nuclear spin. In these terms $\hbar$ is Planck's constant, $\boldsymbol{\gamma}$ the gyromagnetic ratio tensor of the electron spin, γ_I the gyromagnetic ratio of the ^{14}N spin and $\mathbf{A}$ the hyperfine tensor. In TEMPO the principal axes of the gyromagnetic ratio tensor and the hyperfine tensor coincide. On these axes their components are

$$\boldsymbol{\gamma} \;=\; \begin{pmatrix} \gamma_{XX} & 0 & 0 \\ 0 & \gamma_{YY} & 0 \\ 0 & 0 & \gamma_{ZZ} \end{pmatrix} \tag{2}$$

and

$$\mathbf{A} \;=\; \begin{pmatrix} A_{XX} & 0 & 0 \\ 0 & A_{YY} & 0 \\ 0 & 0 & A_{ZZ} \end{pmatrix}. \tag{3}$$

We determine the ESR spectrum using first order perturbation theory. We choose the Hamiltonian (1) as the zero order term in the perturbation expansion and calculate its eigenstates and energy levels. Next we add a first order term

$$\mathscr{H}_{\rm m} \;=\; -2\hbar\mathbf{B}_1\cdot\boldsymbol{\gamma}\cdot\mathbf{S}\cos\omega_{\rm m}t \tag{4}$$

representing a microwave field $2\mathbf{B}_1 \cos \omega_{\mathrm{m}} t$ with an amplitude $2|\mathbf{B}_1|$ and a frequency ω_{m} perpendicular to $\mathbf{B}_0$. The ESR spectrum then follows from the transitions that this first order term induces between the eigenstates of the Hamiltonian (1).

As a first step we determine matrix representations $\mathscr{M}_0$ of the Hamiltonian (1) and $\mathscr{M}_{\mathrm{m}}$ of and $\mathscr{H}_{\mathrm{m}}$ on a basis of eigenstates

$$\begin{array}{c} |m_S\,, m_I\rangle \\ \hline |+\tfrac{1}{2};+1\rangle \\ |+\tfrac{1}{2},\ 0\rangle \\ |+\tfrac{1}{2},-1\rangle \\ |-\tfrac{1}{2},+1\rangle \\ |-\tfrac{1}{2},\ 0\rangle \\ |-\tfrac{1}{2},-1\rangle \end{array} \tag{5}$$

of the Z-components S_Z and I_Z of the electron spin and the ^{14}N spin. Next, we determine the unitary operator $\mathscr{U}^\dagger$ and its hermitian conjugate $\mathscr{U}$ diagonalizing $\mathscr{M}_0$. Then the diagonal elements $\hbar\omega_{nn}$ of

$$\mathscr{U}^\dagger \cdot \mathscr{M}_0 \cdot \mathscr{U} \;=\; \mathscr{E}, \tag{6}$$

are the energy levels, and the columns of $\mathscr{U}$ are the eigenvectors of $\mathscr{M}_0$. In the following step we transform the matrix representation $\mathscr{M}_{\mathrm{m}}$ of $\mathscr{H}_{\mathrm{m}}$ with the unitary operator $\mathscr{U}^\dagger$:

$$\mathscr{U}^\dagger \cdot \mathscr{M}_{\mathrm{m}} \cdot \mathscr{U} \;=\; \mathscr{M}'_{\mathrm{m}} \tag{7}$$

The non-diagonal elements $(\mathscr{M}'_{\mathrm{m}})_{nm}$ of this matrix cause transitions between energy levels $\hbar\omega_{nn}$ and $\hbar\omega_{mm}$ at a rate proportional to

$$|(\mathscr{M}'_{\mathrm{m}})_{nm}|^2 \delta(\omega_{\mathrm{m}} - |\omega_{nn} - \omega_{mm}|) \tag{8}$$

The ESR spectrum is then obtained by plotting the squares of all non-diagonal matrix elements $|(\mathscr{M}'_{\mathrm{m}})_{nm}|^2$ as a function of the energy differences $|\omega_{nn} - \omega_{mm}|$.

From this point onward we follow the same procedure as in Section 3.3.1 of *EofDNP*. We extend the treatment to TEMPO dissolved in a glass, so the TEMPO molecules are oriented randomly, or, inversely, the magnetic field is oriented randomly with respect to principal axes. We introduce spherical coordinates

$$\mathbf{B}_0 \;=\; \begin{pmatrix} B_{0X} \\ B_{0Y} \\ B_{0Z} \end{pmatrix} \;=\; B_0 \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix}. \tag{9}$$

The probability that the magnetic field is oriented in direction (θ, ϕ) is proportional to the solid angle

$$\mathrm{d}\Omega \;=\; \mathrm{d}\cos\theta \mathrm{d}\phi \tag{10}$$

We can therefore extend the ESR spectrum to TEMPO dissolved in a glass by integrating the spectra obtained in the previous step over ϕ and $\cos\theta$.

We finish the determination of the ESR spectrum adding the hyperfine and super-hyperfine interaction with the other nuclear spins in the sample. As in Section 3.3.3 of *EofDNP* we assume that these two interactions lead to Gaussian broadening

$$g_{SI}(\omega) \;=\; \frac{1}{\sqrt{2\pi\Delta_A^2}}\,\exp\left(\frac{\omega^2}{2\Delta_A^2}\right) \tag{11}$$

of the spectrum. The full ESR spectrum is then the convolution

$$g(\omega) \;=\; \int_{-\infty}^{\infty} \mathrm{d}\omega' g_0(\omega') g_{SI}(\omega-\omega') \tag{12}$$

of the spectrum $g_0(\omega)$ calculated above and the Gaussian broadening $g_{SI}(\omega)$.

2. Matrix Representations

We continue with some details of the procedure outlined above. In this section we consider the matrix representation of the Hamiltonian (1) on a basis of eigenstates (5) of S_Z and I_Z. To shorten the notation we define the frequency vector with components

$$\begin{pmatrix} \omega_{0S}^X \\ \omega_{0S}^Y \\ \omega_{0S}^Z \end{pmatrix} = -\begin{pmatrix} \gamma_{XX}B_{0X} \\ \gamma_{YY}B_{0Y} \\ \gamma_{ZZ}B_{0Z} \end{pmatrix} = -\begin{pmatrix} \gamma_{XX}B_0\sin\theta\cos\phi \\ \gamma_{YY}B_0\sin\theta\sin\phi \\ \gamma_{ZZ}B_0\cos\theta \end{pmatrix} \tag{13}$$

and write

$$\begin{aligned} \omega_{0S}^{\pm} &= \omega_{0S}^X \pm i\omega_{0S}^Y, \\ \omega_{0I}^{\pm} &= \gamma_I\left(B_{0X} \pm iB_{0Y}\right), \\ \omega_{0I}^{Z} &= \gamma_I B_{0Z}. \end{aligned} \tag{14}$$

Then, on a basis of eigenstates (5) the electron Zeeman energy is represented by

$$\tfrac{1}{2}\hbar\begin{pmatrix} \omega_{0S}^Z & 0 & 0 & \omega_{0S}^- & 0 & 0 \\ 0 & \omega_{0S}^Z & 0 & 0 & \omega_{0S}^- & 0 \\ 0 & 0 & \omega_{0S}^Z & 0 & 0 & \omega_{0S}^- \\ \omega_{0S}^+ & 0 & 0 & -\omega_{0S}^Z & 0 & 0 \\ 0 & \omega_{0S}^+ & 0 & 0 & -\omega_{0S}^Z & 0 \\ 0 & 0 & \omega_{0S}^+ & 0 & 0 & -\omega_{0S}^Z \end{pmatrix} \tag{15}$$

and the nuclear Zeeman energy by

$$\tfrac{1}{2}\hbar\begin{pmatrix} -2\omega_{0I}^Z & \omega_{0I}^-\sqrt{2} & 0 & 0 & 0 & 0 \\ \omega_{0I}^+\sqrt{2} & 0 & \omega_{0I}^-\sqrt{2} & 0 & 0 & 0 \\ 0 & \omega_{0I}^+\sqrt{2} & 2\omega_{0I}^Z & 0 & 0 & 0 \\ 0 & 0 & 0 & -2\omega_{0I}^Z & \omega_{0I}^-\sqrt{2} & 0 \\ 0 & 0 & 0 & \omega_{0I}^+\sqrt{2} & 0 & \omega_{0I}^-\sqrt{2} \\ 0 & 0 & 0 & 0 & \omega_{0I}^+\sqrt{2} & 2\omega_{0I}^Z \end{pmatrix}. \tag{16}$$

To obtain the matrix representation of the hyperfine interaction, we introduce step operators

$$\begin{aligned} S_\pm &= S_X \pm iS_Y, \\ I_\pm &= I_X \pm iI_Y \end{aligned} \tag{17}$$

and introduce the short notation

$$\begin{aligned} A_{++} &= A_{XX} + A_{YY}, \\ A_{--} &= A_{XX} - A_{YY}, \end{aligned} \tag{18}$$

allowing us to rewrite it as

$$\begin{aligned} \hbar \mathbf{I}\cdot\mathbf{A}\cdot\mathbf{S} &= \hbar\big(I_X A_{XX} S_X + I_Y A_{YY} S_Y + I_Z A_{ZZ} S_Z\big) \\ &= \hbar\big(\tfrac{1}{4}(I_+S_+ + I_-S_-)A_{--} + \tfrac{1}{4}(I_+S_- - I_-S_+)A_{++} + I_Z A_{ZZ} S_Z\big) \end{aligned} \tag{19}$$

Thus we find for the matrix representation of the hyperfine interaction;

$$\tfrac{1}{4}\hbar \begin{pmatrix} 2A_{ZZ} & 0 & 0 & 0 & A_{--}\sqrt{2} & 0 \\ 0 & 0 & 0 & A_{++}\sqrt{2} & 0 & A_{--}\sqrt{2} \\ 0 & 0 & -2A_{ZZ} & 0 & A_{++}\sqrt{2} & 0 \\ 0 & A_{++}\sqrt{2} & 0 & -2A_{ZZ} & 0 & 0 \\ A_{--}\sqrt{2} & 0 & A_{++}\sqrt{2} & 0 & 0 & 0 \\ 0 & A_{--}\sqrt{2} & 0 & 0 & 0 & 2A_{ZZ} \end{pmatrix}. \tag{20}$$

We finish this section with the matrix representation of the interaction (4) of the electron spin with the microwave field. Just as for the static magnetic field $\mathbf{B}_0$ we introduce a frequency vector

$$\begin{pmatrix} \omega_{1S}^X \\ \omega_{1S}^Y \\ \omega_{1S}^Z \end{pmatrix} = -\begin{pmatrix} \gamma_{XX} B_{1X} \\ \gamma_{YY} B_{1Y} \\ \gamma_{ZZ} B_{1Z} \end{pmatrix} = -\begin{pmatrix} \gamma_{XX} B_1 \sin\theta\cos\phi \\ \gamma_{YY} B_1 \sin\theta\sin\phi \\ \gamma_{ZZ} B_1 \cos\theta \end{pmatrix} \tag{21}$$

and shorten our notation, writing

$$\omega_{1S}^\pm = \omega_{1S}^X \pm i\omega_{1S}^Y. \tag{22}$$

Then, on a basis of eigenstates (5) of S_Z and I_Z the interaction of the electron spin with the microwave field is represented by

$$2\hbar \begin{pmatrix} \omega_{1S}^Z & 0 & 0 & \omega_{1S}^- & 0 & 0 \\ 0 & \omega_{1S}^Z & 0 & 0 & \omega_{1S}^- & 0 \\ 0 & 0 & \omega_{1S}^Z & 0 & 0 & \omega_{1S}^- \\ \omega_{1S}^+ & 0 & 0 & -\omega_{1S}^Z & 0 & 0 \\ 0 & \omega_{1S}^+ & 0 & 0 & -\omega_{1S}^Z & 0 \\ 0 & 0 & \omega_{1S}^+ & 0 & 0 & -\omega_{1S}^Z \end{pmatrix} \cos\omega_{\mathrm{m}} t. \tag{23}$$

The sum of (15), (16) and (20) is the matrix $\mathscr{M}_0$ that we wish to diagonalize in the procedure described above, and (23) is the matrix $\mathscr{M}_{\rm m}$ that we wish to transform with the unitary operator $\mathscr{U}^\dagger$, in order to determine the ESR spectrum.

3. Numerical Methods

We use numerical methods to determine the eigenvalues of the matrix $\mathscr{M}_0$ and the unitary operators $\mathscr{U}^\dagger$ and $\mathscr{U}$ diagonalizing it. The matrices involved are complex, and the first step consists of transforming the problem to real arithmetic. We follow the method described in [1] and notice that complex numbers $a+ib$ obey the same multiplication rules as real asymmetric 2×2 matrices

$$\begin{pmatrix} a & -b \\ b & a \end{pmatrix} \tag{24}$$

As a result a Hermitian $n\times n$ matrix $\mathscr{A}=\mathbf{A}+i\mathbf{B}$ transforms in the same way as a real symmetric $2n\times 2n$ matrix

$$\begin{pmatrix} \mathbf{A} & -\mathbf{B} \\ \mathbf{B} & \mathbf{A} \end{pmatrix} \tag{25}$$

To show this we first transform $\mathscr{A}=\mathbf{A}+i\mathbf{B}$ with the unitary matrix $\mathscr{U}^\dagger=\mathbf{U}-i\mathbf{V}$:

$$\mathscr{U}^\dagger\cdot\mathscr{A}\cdot\mathscr{U} \;=\; (\mathbf{U}-i\mathbf{V})(\mathbf{A}+i\mathbf{B})(\mathbf{U}+i\mathbf{V}) \;=\; (\mathbf{C}+i\mathbf{D}) \;=\; \mathscr{C}, \tag{26}$$

where

$$\begin{aligned} \mathbf{C} &= \mathbf{UAU}-\mathbf{UBV}+\mathbf{VBU}+\mathbf{VAV} \\ \mathbf{D} &= \mathbf{UBU}+\mathbf{UAV}-\mathbf{VAU}+\mathbf{VBV}. \end{aligned} \tag{27}$$

Next we write $\mathscr{A}=\mathbf{A}+i\mathbf{B}$ as a $2n\times 2n$ matrix (25), $\mathscr{U}^\dagger=\mathbf{U}-i\mathbf{V}$ as a $2n\times 2n$

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ -\mathbf{B} & \mathbf{A} \end{pmatrix} \tag{28}$$

and perform the transformation again. We find:

$$\begin{pmatrix} \mathbf{U} & \mathbf{V} \\ -\mathbf{V} & \mathbf{U} \end{pmatrix}\begin{pmatrix} \mathbf{A} & -\mathbf{B} \\ \mathbf{B} & \mathbf{A} \end{pmatrix}\begin{pmatrix} \mathbf{U} & -\mathbf{V} \\ \mathbf{V} & \mathbf{U} \end{pmatrix} = \begin{pmatrix} \mathbf{C} & -\mathbf{D} \\ \mathbf{D} & \mathbf{C} \end{pmatrix}. \tag{29}$$

This observation enables us to proceed as follows. We write the 6×6 Hermitian matrix representation of $\mathscr{H}_0$ as a 12×12 real matrix in the shape (25). Next we use the routine `jacobi.f` from [1] to determine the real orthogonal 12×12 matrix which diagonalizes this real 12×12 matrix. This provides the eigenvalues of $\mathscr{H}_0$ needed to insert in (8). Next we also write the 6×6 Hermitian matrix representation (23) of $\mathscr{H}_{\rm m}$ as a 12×12 real matrix in the shape (25) and transform it using (29). From the result we finally determine the squares of the transition matrix elements needed to insert in (8).

4. Example Result

Figure 1 presents a result of the procedure described above. The calculation is performed for an externally applied magnetic field $B_0=3.4$ T. For the components of the

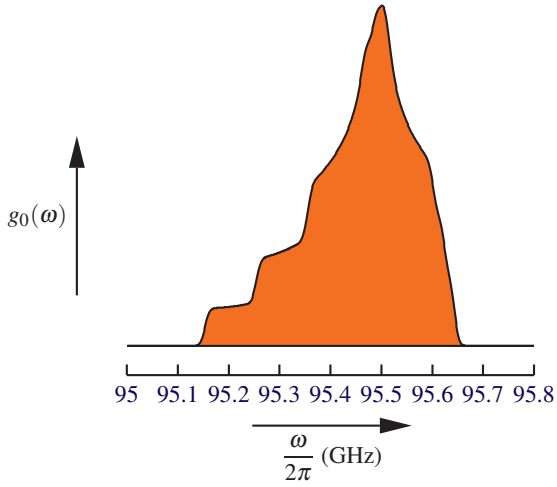

Figure 1: Theoretical ESR spectrum for TEMPO dissolved in a glass of butanol. The static magnetic field is set at 3.4 T. The dotted curve presents the spectrum before convolution with Gaussian broadening by hyperfine and super-hyperfine interaction. The drawn curve shows the spectrum after convolution with Gaussian broadening.

gyromagnetic ratio tensor and the hyperfine tensor we use the values given in [2] for TEMPO in butanol:

$$\begin{aligned} \gamma_{XX} &= -2\pi \times 28.124 \cdot 10^9\,\mathrm{s}^{-1}\mathrm{T}^{-1}, \\ \gamma_{YY} &= -2\pi \times 28.084 \cdot 10^9\,\mathrm{s}^{-1}\mathrm{T}^{-1}, \\ \gamma_{ZZ} &= -2\pi \times 28.016 \cdot 10^9\,\mathrm{s}^{-1}\mathrm{T}^{-1} \end{aligned} \tag{30}$$

and

$$\begin{aligned} A_{XX} &= 2\pi \times 20.5 \cdot 10^6\,\mathrm{s}^{-1}, \\ A_{YY} &= 2\pi \times 17.7 \cdot 10^6\,\mathrm{s}^{-1}, \\ A_{ZZ} &= 2\pi \times 100.9 \cdot 10^6\,\mathrm{s}^{-1}. \end{aligned} \tag{31}$$

Furthermore we insert the value $\gamma_I = +2\pi \times 3.0777 \cdot 10^6\,\mathrm{s}^{-1}\mathrm{T}^{-1}$ for the gyromagnetic ratio of ^{14}N—as calculated from [3]. Finally we estimate the second moment of the Gaussian broadening to be the same as in Section 3.3.3 of *EofDNP*:

$$\Delta_A^2 = 0.2 \cdot 10^{16}\,\mathrm{s}^{-2}\,. \tag{32}$$

Figure 1 shows the result of this procedure for a static magnetic set at 3.4 T. For comparison with experiment, Figure 2 reproduces Figure 3.12 from *EofDNP*. It represents the the ESR spectrum $g(\omega)$ as a function of $\omega/2\pi$ for OH-TEMPO dissolved in frozen water. The curve is obtained by integrating the experimental results for the derivative of $g(\omega)$ presented in [4]. The magnetic field is 3.4 T. Except that the experimental spectrum is somewhat further smoothed, the two ESR spectra appear to be the same.

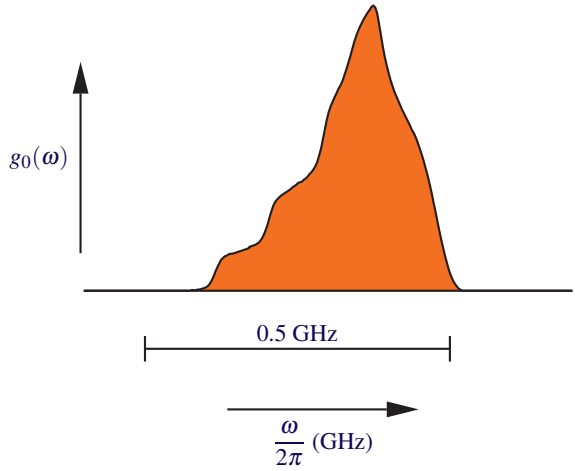

Figure 2: The ESR spectrum $g(\omega)$ as a function of $\omega/2\pi$ for OH-TEMPO dissolved in frozen water. The curve is obtained by integrating the experimental results for the derivative of $g(\omega)$ presented in [4]. The magnetic field is 3.4 T.

References

[1] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery: *Numerical Recipes in Fortran*, 2nd Ed., Cambridge University Press, Cambridge, 1992.

[2] S.T. Goertz, J. Harmsen, J. Heckmann, Ch. Hess, W. Meyer, E. Radtke, G. Reicherz: *Nucl. Instrum. Methods Phys. Res. A* 526 (2004) 43-52
DOI:10.1016/j.nima.2004.03.148.

[3] N.J. Stone: *At. Data Nucl. Data Tables* 90 (2005) 75-176
DOI:10.1016/j.adt.2005.04.001.

[4] M.R. Fuchs, T.F. Prisner, K. Möbius: *Rev. Sci. Instr.* 70 (1999) 3681-3683
DOI:10.1063/1.1149977

Appendix: Fortran Codes

This appendix lists the Fortran codes used to produce Figure 1. They were compiled and tested with `gfortran` on a Dell Mobile Precision ® 5510 CTO running Xubuntu 16.04 for 64 bits processors. Note that the original subroutine `jacobi.f` from [1] assumes that underflows are set to zero. The routine has been modified, in order to work with machines for which this is not true.

Main program:

```
C*****************************************************************************
C*                                esr.f                                      *
C*****************************************************************************
C* calculates ESR spectra for TEMPO                                          *
C*****************************************************************************
C* cost        = cos(theta) (direction B_0)                (double real) *
C* dcost       = step size cost                            (double real) *
C* costmin     = -1 = minimum value cost                   (double real) *
C* costmax     = +1 = maximum value cost                   (double real) *
C* phi         = phi        (direction B_0) (radians)      (double real) *
C* dphi        = step size phi                             (double real) *
C* phimin      = 0   = minimum value phi                   (double real) *
C* phimax      = 2pi = maximum value phi                   (double real) *
C* h(i,j)      = matrix representation of the Hamiltonian  (double real) *
C* hh(i,j)     = h(i,j) at first                           (double real) *
C*               jacobi transforms it to matrix with upper triangle = 0    *
C* d(j)        = vector with eigenvalues                   (double real) *
C* v(i,j)      = matrix with eigenvectors                  (double real) *
C*               j = number of eigenvector (column)                        *
C*               i = component of eigenvector                              *
C* h1(i,j)     = matrix representation of perturbation     (double real) *
C*               (due to microwave field)                                  *
C* hh1(i,j)    = matrix representation of perturbation     (double real) *
C*               (on basis of eigenvectors of h(i,j)                       *
C* iom         = bucket number                             (integer)     *
C* iom0        = number of centre of gravity of spectrum   (integer)     *
C* iommin      = -600 = minimum bucket number              (integer)     *
C* iommax      = 600  = maximum bucket number              (integer)     *
C* esr0(iom)   = ESR spectrum as a function of bucket number (double real) *
C*               (before convolution with Gauss)                           *
C* esr1(iom)   = ESR spectrum as a function of bucket number (double real) *
C*               (after convolution with Gauss)                            *
C* mom0        = normalization of ESR spectrum             (double real) *
C* mom1        = first moment of ESR spectrum              (double real) *
C* mom2        = second moment of ESR spectrum             (double real) *
C* iomco       = cut-off for Gaussian                      (integer)     *
C* eps         = amplitude unnormalized Gaussian at iomco  (double real) *
C*****************************************************************************

      program main

      implicit none

      include "blocks.f"

      integer i, j, nrot, nsw
      integer it, ip, nt, np
      integer iomco, iom0
      double precision eps
      double precision cost, dcost, costmin, costmax
      double precision phi, dphi, phimin, phimax
      double precision h(1:12,1:12), h1(1:12,1:12)
      double precision d(1:12)
      double precision v(1:12,1:12)
```

```
      integer k, l, iom, iommin, iommax
      double precision hh(1:12,1:12)
      double precision hh1(1:12,1:12)
      parameter (iommin = -600)
      parameter (iommax = 600)
      double precision esr0(iommin:iommax)
      double precision esr1(iommin:iommax)
      double precision mom0,  mom1, mom2

      character outf*60

      outf = "inout/outf.txt"
      open(20,file=outf)

C*************************************************************************
C* enter data for the calculation                                        *
C*************************************************************************
      call init

C*************************************************************************
C* frequency space is divided in buckets spaced by 1 MHz                 *
C* the spectrum will be generated by filling the buckets with            *
C*   the squares of transition matrix elements                           *
C*   coupling states with a frequency difference                         *
C*   corresponding to the bucket number                                  *
C* determine number iom0 of bucket at centre of gravity of spectrum      *
C* set contents of all buckets equal to zero                             *
C*************************************************************************
      iom0 = -idnint(b0*1.0d3*(gx+gy+gz)/3.0d0)
      do iom=iommin,iommax,1
        esr0(iom) = 0.0d0
      end do

C*************************************************************************
C* vary phi in 1000 steps from 0 to 2pi                                  *
C* vary cos(theta) in 1000 steps from -1 to +1                           *
C*************************************************************************
      nt      = 1000
      np      = 1000
      costmin = -1.0d0
      costmax = 1.0d0
      dcost   = (costmax-costmin)/nt
      phimin  = 0.0d0
      phimax  = 2.0d0*pi
      dphi    = (phimax-phimin)/np

      cost    = costmin
      do it=1,nt,1
        phi = phimin
        do ip=1,np,1

C*************************************************************************
C* determine matrix representation of the Hamiltonian                    *
C* h  = unperturbed part                                                 *
C* h1 = perturbation due to microwave field                              *
C*************************************************************************
          call matrix(cost,phi,h,h1)

          do j=1,12,1
            do i=1,12,1
              hh(i,j) = h(i,j)
            end do
          end do

C*************************************************************************
C* determine matrix v diagonalizing h                                    *
C*************************************************************************
```

```
          call jacobi(h,d,v,nrot,nsw)
          if (nsw.ge.50) then
            write(*,*) "to many sweeps in jacobi: nsw = ", nsw
          end if

C*************************************************************************
C* transform h1 with v                                                   *
C*************************************************************************
          do i=1,12,1
            do j=1,12,1
              hh1(i,j) = 0.0d0
              do k=1,12,1
                do l=1,12,1
                  hh1(i,j) = hh1(i,j)+v(k,i)*h1(k,l)*v(l,j)
                end do
              end do

C*************************************************************************
C* for each non-diagonal element:                                        *
C* determine energy splitting between the states it connects             *
C* determine bucket corresponding to this splitting                      *
C* fill this bucket with the square of this matrix element               *
C*************************************************************************
              iom = idnint(dabs(d(i)-d(j))*1.0d3)-iom0
              if ((iom.gt.iommin).and.(iom.lt.iommax)) then
                esr0(iom) = esr0(iom)+hh1(i,j)*hh1(i,j)
              end if
            end do
          end do
          phi = phi+dphi
        end do
        cost = cost+dcost
      end do

C*************************************************************************
C* convolute with a Gaussian                                             *
C* cut-off of Gaussian set at 7 times its width                          *
C*************************************************************************
      iomco = idnint(7.0*del)
      eps = e**(-7*7/2.0d0)
      call convolute(iomco,eps,iommin,iommax,esr0,esr1)

C*************************************************************************
C* normalize spectrum (basis normalization 1 MHz)                        *
C* write normalization to standard output                                *
C*************************************************************************
      mom0 = 0.0d0
      do iom=iommin,iommax,1
        mom0 = mom0+esr1(iom)
      end do
      write(*,*) mom0
      do iom=iommin,iommax,1
        esr1(iom) = esr1(iom)/mom0
      end do

C*************************************************************************
C* write normalized spectrum in output file                              *
C* first column:  frequency (GHz)                                        *
C* second column: intensity (GHz^-1)                                     *
C*************************************************************************
      do iom=iommin,iommax,1
        write(20,100) (iom+iom0)*1.0d-3, esr1(iom)*1.0d3
      end do

C*************************************************************************
C* determine centre of gravity and second moment of the spectrum         *
C* write them to standard output                                         *
C*************************************************************************
```

```
      mom1 = 0.0d0
      mom2 = 0.0d0
      do iom=iommin,iommax,1
        mom1 = mom1+iom*esr1(iom)
      end do
      write(*,*) mom1+iom0
      do iom=iommin,iommax,1
        mom2 = mom2+esr1(iom)*(iom-mom1)*(iom-mom1)
      end do
      write(*,*) sqrt(mom2)
      write(*,*)

      close(20)

  100 format(f12.5,"  ",f12.5)

      end
```

Subroutine init:

```
C*************************************************************************
C*                              init.f                                   *
C*************************************************************************
C* reads and initiates parameters for the calculation                    *
C*************************************************************************

      subroutine init

      implicit none

      include "blocks.f"

      character inf*60

      inf = "inout/inf.dat"
      open(10,file=inf)

      call rnul(inf,1)
      call rnul(inf,2)
      call rnul(inf,3)
      call rnul(inf,4)
      call rreal(inf,5,e)
      call rreal(inf,6,pi)
      call rnul(inf,7)
      call rreal(inf,8,gx)
      call rreal(inf,9,gy)
      call rreal(inf,10,gz)
      call rreal(inf,11,gn)
      call rreal(inf,12,ax)
      call rreal(inf,13,ay)
      call rreal(inf,14,az)
      call rreal(inf,15,del)
      call rnul(inf,16)
      call rreal(inf,17,b0)

      close(10)

      end
```

Subroutines reading lines of data file:

```
C*************************************************************************
C*                              rnul.f                                   *
C*************************************************************************
C* reads empty line from iofile                                         *
```

```
C*************************************************************************
C* Internal variables:                                                   *
C* dummy    = character string for intermediate use                      *
C* linenr   = number of line being read                    (integer)     *
C* iofile   = character string denoting input file                       *
C*************************************************************************

      subroutine rnul(iofile,linenr)

      implicit none

      integer   linenr
      character iofile*60
      character dummy*80

      read (10,"(a80)",err=70) dummy

      go to 1000

   70 continue
      write (*,*) 'ERROR reading line ', linenr, ' of ', iofile

 1000 continue
      end

C*************************************************************************
C*                            rreal.f                                    *
C*************************************************************************
C* reads double real number from iofile                                  *
C*************************************************************************
C* Internal variables:                                                   *
C* dummy    = character string for intermediate use                      *
C* dummy1   = character string for intermediate use                      *
C* linenr   = number of line being read                    (integer)     *
C* pos      = number of column being read                  (integer)     *
C* iofile   = character string denoting input file                       *
C* realin   = real to be read from file                    (double real) *
C*************************************************************************

      subroutine rreal(iofile,linenr,realin)

      implicit none

      integer   linenr, pos
      character iofile*60
      character dummy*80
      character dummy1*80
      double precision realin

      read (10,"(a80)",err=70) dummy
      pos = index(dummy,':') + 2
      dummy1 = dummy(pos:)
      dummy = dummy1
      read (dummy,*,err=70) realin
      go to 1000

   70 continue
      write (*,*) 'ERROR reading line ', linenr, ' of ', iofile

 1000 continue
      end
```

Data file read by subroutine init:

```
01..5....0....5....0....5....0....5....0....5....0....5....0....5....0....5....0....5
02. Input file for the calculation of the ESR spectrum
03.
```

```
04. Basic constants:
05. e                          : 2.718281828   (double real)
06. pi                         : 3.141592654   (double real)
07. ESR parameters TEMPO:
08. XX-component g-tensor      : -28.124       (GHz/T double real)
09. YY-component g-tensor      : -28.084       (GHz/T double real)
10. ZZ-component g-tensor      : -28.016       (GHz/T double real)
11. gamma_I (14N)              : 0.0030777058  (GHz/T double real)
12. XX-comp. hyperfine tensor  : 0.0205        (GHz double real)
13. YY-comp. hyperfine tensor  : 0.0177        (GHz double real)
14. ZZ-comp. hyperfine tensor  : 0.1009        (GHz double real)
15. width Gaussian             : 7.1           (MHz double real)
16. Magnetic field:
17. B_0                        : 3.4           (T double real)
```

Common blocks storing data read from data file:

```
C*************************************************************************
C*                              blocks.f                                 *
C*************************************************************************
C* declares and saves common varaiables                                  *
C*************************************************************************
C* e       = 2.718281828                                 (double real) *
C* pi      = 3.141592654                                 (double real) *
C* gx      = XX component of g-tensor (GHz/T)            (double real) *
C* gy      = YY component of g-tensor (Ghz/T)            (double real) *
C* gz      = ZZ component of g-tensor (GHz/T)            (double real) *
C* gn      = gyromagnetic ratio nuclear spin (GHz/T)     (double real) *
C* ax      = XX component of hyperfine tensor 14N (GHz)  (double real) *
C* ay      = YY component of hyperfine tensor 14N (GHz)  (double real) *
C* az      = ZZ component of hyperfine tensor 14N (GHz)  (double real) *
C* del     = width of the Gaussian broadening (MHz)      (double real) *
C* b0      = B_0 = static magnetic field                 (double real) *
C*************************************************************************

      double precision e, pi
      double precision gx, gy, gz, gn, ax, ay, az, del
      double precision b0

      common /block1/e, pi
      common /block2/gx, gy, gz, gn, ax, ay, az, del
      common /block3/b0

      save /block1/,/block2/,/block3/
```

Subroutine matrix:

```
C*************************************************************************
C*                              matrix.f                                 *
C*************************************************************************
C* calculates matrix representation of the unperturbed Hamiltonian       *
C* calculates matrix representation of interaction with microwave field  *
C*************************************************************************
C* input:                                                                *
C* b0          = magnetic field strength B_0 (T)             (double real) *
C* cost        = cos(theta) (direction B_0)                  (double real) *
C* phi         = phi        (direction B_0) (radians)        (double real) *
C*************************************************************************
C* input from common blocks:                                             *
C* gx          = XX component of g-tensor (GHz/T)            (double real) *
C* gy          = YY component of g-tensor (Ghz/T)            (double real) *
C* gz          = gamma_ZZ component of g-tensor (GHz/T)      (double real) *
C* gn          = gyromagnetic ratio nuclear spin (GHz/T)     (double real) *
C* ax          = XX component of hyperfine tensor 14N (GHz)  (double real) *
C* ay          = YY component of hyperfine tensor 14N (GHz)  (double real) *
C* az          = ZZ component of hyperfine tensor 14N (GHz)  (double real) *
C*************************************************************************
```

```
C* output:                                                                    *
C* mat0(1:12,1:12) = matrix representation of the Hamiltonian                 *
C* mat1(1:12,1:12) = matrix representation microwave field                    *
C*****************************************************************************

C* internal parameters:                                                       *
C* sint        = sin(theta) (direction B_0)                  (double real) *
C* cosp        = cos(phi) (direction B_0)                    (double real) *
C* sinp        = sin(phi) (direction B_0)                    (double real) *
C* om0x        = X-component freq. vector electron spin (GHz) (double real) *
C* om0y        = Y-component freq. vector electron spin (GHz) (double real) *
C* om0z        = Z-component freq. vector electron spin (GHz) (double real) *
C* om1x        = X-component microwave field            (GHz) (double real) *
C* om1y        = Y-component microwave field            (GHz) (double real) *
C* om1z        = Z-component microwave field            (GHz) (double real) *
C* omnx        = X-component freq. vector nuclear spin (GHz)  (double real) *
C* omny        = Y-component freq. vector nuclear spin (GHz)  (double real) *
C* omnz        = Z-component freq. vector nuclear spin (GHz)  (double real) *
C*****************************************************************************

      subroutine matrix(cost,phi,mat0,mat1)

      implicit none

      include "blocks.f"

      integer i, j
      double precision cost, phi
      double precision mat0(1:12,1:12), mat1(1:12,1:12)
      double precision sint, cosp, sinp
      double precision om0x, om0y, om0z, om1x, om1y, om1z
      double precision omnx, omny, omnz
      double precision zs(1:12,1:12)
      double precision zn(1:12,1:12)
      double precision hh(1:12,1:12)

      sint = sqrt(1-cost*cost)
      cosp = cos(phi)
      sinp = sin(phi)

      om0x = -gx*b0*sint*cosp
      om0y = -gy*b0*sint*sinp
      om0z = -gz*b0*cost
      om1x = -gx*cost*cosp
      om1y = -gy*cost*sinp
      om1z = gz*sint
      omnx = gn*b0*sint*cosp
      omny = gn*b0*sint*sinp
      omnz = gn*b0*cost

      do i=1,12,1
        do j=1,12,1
          zs(i,j)  = 0.0d0
          zn(i,j)  = 0.0d0
          hh(i,j)  = 0.0d0
          mat0(i,j) = 0.0d0
          mat1(i,j) = 0.0d0
        end do
      end do


      zs(1,1)   = om0z
      zs(2,2)   = om0z
      zs(3,3)   = om0z
      zs(4,4)   = -om0z
      zs(5,5)   = -om0z
      zs(6,6)   = -om0z
```

```
zs(1,4)    = om0x
zs(2,5)    = om0x
zs(3,6)    = om0x
zs(4,1)    = om0x
zs(5,2)    = om0x
zs(6,3)    = om0x

do i=1,6,1
  do j=1,6,1
    zs(i+6,j+6) = zs(i,j)
  end do
end do

zs(7,4)    = -om0y
zs(8,5)    = -om0y
zs(9,6)    = -om0y
zs(10,1)   = om0y
zs(11,2)   = om0y
zs(12,3)   = om0y

do i=1,6,1
  do j=1,6,1
    zs(i,j+6) = -zs(i+6,j)
  end do
end do

zn(1,1)    = -2.0d0*omnz
zn(3,3)    = -zn(1,1)
zn(4,4)    = zn(1,1)
zn(6,6)    = zn(3,3)

zn(1,2)    = sqrt(2.0d0)*omnx
zn(2,3)    = zn(1,2)
zn(2,1)    = zn(1,2)
zn(3,2)    = zn(1,2)
zn(4,5)    = zn(1,2)
zn(5,6)    = zn(1,2)
zn(5,4)    = zn(1,2)
zn(6,5)    = zn(1,2)

do i=1,6,1
  do j=1,6,1
    zn(i+6,j+6) = zn(i,j)
  end do
end do

zn(8,1)    = sqrt(2.0d0)*omny
zn(9,2)    = zn(8,1)
zn(11,4)   = zn(8,1)
zn(12,5)   = zn(8,1)
zn(7,2)    = -zn(8,1)
zn(8,3)    = -zn(8,1)
zn(10,5)   = -zn(8,1)
zn(11,6)   = -zn(8,1)

do i=1,6,1
  do j=1,6,1
    zn(i,j+6) = -zn(i+6,j)
  end do
end do

hh(1,1)    = az
hh(3,3)    = -az
hh(4,4)    = -az
hh(6,6)    = az
```

```
hh(1,5)    = (ax-ay)/sqrt(2.0d0)
hh(2,6)    = hh(1,5)
hh(5,1)    = hh(1,5)
hh(6,2)    = hh(1,5)
hh(2,4)    = (ax+ay)/sqrt(2.0d0)
hh(3,5)    = hh(2,4)
hh(4,2)    = hh(2,4)
hh(5,3)    = hh(2,4)

do i=1,6,1
  do j=1,6,1
    hh(i+6,j+6) = hh(i,j)
  end do
end do

do i=1,12,1
  do j=1,12,1
    mat0(i,j) = 0.5d0*(zs(i,j)+zn(i,j)+hh(i,j))
  end do
end do

mat1(1,1)    = om1z
mat1(2,2)    = om1z
mat1(3,3)    = om1z
mat1(4,4)    = -om1z
mat1(5,5)    = -om1z
mat1(6,6)    = -om1z

mat1(1,4)    = om1x
mat1(2,5)    = om1x
mat1(3,6)    = om1x
mat1(4,1)    = om1x
mat1(5,2)    = om1x
mat1(6,3)    = om1x

do i=1,6,1
  do j=1,6,1
    mat1(i+6,j+6) = mat1(i,j)
  end do
end do

mat1(7,4)    = -om1y
mat1(8,5)    = -om1y
mat1(9,6)    = -om1y
mat1(10,1)   = om1y
mat1(11,2)   = om1y
mat1(12,3)   = om1y

do i=1,6,1
  do j=1,6,1
    mat1(i,j+6) = -mat1(i+6,j)
  end do
end do

end
```

Subroutine jacobi:

This subroutine is modified from [1]. Copyrights are owned by Cambridge University Press, and only the headings showing the modifications are provided here.

```
C*****************************************************************************
C*                                  jacobi.f                                 *
C*****************************************************************************
C* Modified from Numerical Recipies in Fortran, 2nd Edition, 1994            *
C*****************************************************************************
C* calculates eigenvalues and eigenvectors of real symmetric matrix          *
```

```
C*********************************************************************
C* assumes that underflow is set to zero                             *
C* This OK on Dell Mobile Precision 5510 with Xubuntu 16.04          *
C*********************************************************************
C* input:                                                            *
C* a(ip,iq)     = 12x12 symmetric matrix                (double real) *
C*********************************************************************
C* output:                                                           *
C* a(ip,iq)     = transformed matrix with upper triangle = 0 (double real) *
C* d(ip)        = vector with eigenvalues               (double real) *
C* v(ip,iq)     = matrix with eigenvectors              (double real) *
C*                iq = number of eigenvector (column)              *
C*                ip = component of eigenvector                    *
C* nrot         = number of rotations                   (integer)     *
C* nsw          = number of sweeps                      (integer)     *
C*********************************************************************
C* eps          = precision of the diagoanlization      (double real) *
C*********************************************************************

      subroutine jacobi(a,d,v,nrot,nsw)

      implicit none

      integer nrot, nsw
      double precision a(1:12,1:12),d(1:12),v(1:12,1:12)
      integer i, ip, iq, j, n
      double precision c, g, h, s, t, tau, theta, tresh
      double precision b(1:12), z(1:12)
      double precision sm, tr, eps

      eps = 1.0d-100
      n   = 12
```

Subroutine convolute:

```
C*********************************************************************
C*                          convolute.f                              *
C*********************************************************************
C* convolutes the ESR spectrum with a Gaussian                       *
C*********************************************************************
C* input:                                                            *
C* iomco     = cut-off for Gaussian                  (integer)     *
C* eps       = amplitude unnormalized Gaussian at iomco (double real) *
C* iommin    = -600 = minimum bucket number          (integer)     *
C* iommin    = 600  = maximum bucket number          (integer)     *
C* esr0      = input ESR spectrum                    (double real) *
C*********************************************************************
C* input from common blocks:                                         *
C* e         = 2.718281828                           (double real) *
C* pi        = 3.141592654                           (double real) *
C* del       = width of the Gaussian (MHz)           (double real) *
C*********************************************************************
C* output:                                                           *
C* esr1      = output ESR spectrum                   (double real) *
C*********************************************************************
C* internal parameters:                                              *
C* iom       = bucket number                         (integer)     *
C* iomcomax  = counter for trailing zero's           (integer)     *
C* iomcomin  = counter for trailing zero's           (integer)     *
C*********************************************************************

      subroutine convolute(iomco,eps,iommin,iommax,esr0,esr1)

      implicit none

      include "blocks.f"
```

```
      integer iomco, iommin, iommax
      double precision eps
      double precision esr0(iommin:iommax), esr1(iommin:iommax)
      integer iom, k, m
      integer iomcomin, iomcomax
      double precision gauss(-iomco:iomco)

C*************************************************************************
C* check input spectrum on number of trailing zero's                     *
C*************************************************************************
      iomcomin = 0
      do iom=iommin,iommax,1
        if (esr0(iom).lt.eps) then
          iomcomin = iomcomin+1
        else
          goto 10
        end if
      end do
 10   continue
      iomcomax = 0
      do iom=iommax,iommin,-1
        if (esr0(iom).lt.eps) then
          iomcomax = iomcomax+1
        else
          goto 20
        end if
      end do
 20   continue
      if ((iomcomin.lt.iomco).or.(iomcomax.lt.iomco)) then
        write(*,*) "Insufficient trailing zero's for convolution"
        go to 30
      end if

C*************************************************************************
C initiate output ESR spectrum                                          *
C*************************************************************************
      do iom=iommin,iommax,1
        esr1(iom) = 0.0d0
      end do

C*************************************************************************
C Calculate a normalized Gaussian with width del                        *
C*************************************************************************
      do k=-iomco,iomco,1
        gauss(k) = 1.0d0/sqrt(2.0d0*pi)/del*e**(-k*k/2.0d0/del/del)
      end do

C*************************************************************************
C Convolute the input spectrum with this Gaussian                       *
C*************************************************************************
      do iom=iommin+iomco+1,iommax-iomco-1,1
        esr1(iom) = 0.0d0
        do k=-iomco,iomco,1
          esr1(iom)=esr1(iom)+esr0(iom+k)*gauss(k)
        end do
      end do

 30   continue

      end
```