```
(*-- QUADRUPOLE version 1.0 ---- September 18, 2006 --*)
(*---------- with Wolfram Mathematica 5.0 ------------*)
(*--------------- Author: R. HAJJAR ------------------*)

(*-- run --*)
(*-- starts the notebook and determines the nature of the sample: --*)
(*-- a single crystal if numberOfGammaAngles is 1, otherwise a powder --*)
run := Module[{QCC, q, amplitude},
    ω0 = 2 π * larmorFrequencyMhz;
    dim = IntegerPart[2 quadrupoleSpin + 1];
    ms = Range[quadrupoleSpin, -quadrupoleSpin, -1];
    Iz = DiagonalMatrix[ms]; (* initial state  *)
    ωrot = 2 π * spinRatekHz * 10^-3;
    QCC = 2 π * QCCMHz;
    q = 2 quadrupoleSpin (2 quadrupoleSpin - 1);
    A = Sqrt[6]/(2 q) QCC * DiagonalMatrix[(3 ms^2 - quadrupoleSpin (quadrupoleSpin + 1)) / 3];
    If[quadrupoleOrder == 2, {
      W0 = Sqrt[5] (3 + η^2) / 10;
      amplitude = -1/ω0 QCC^2/q^2;
      A4 = amplitude/(2 Sqrt[70]) (18 quadrupoleSpin (quadrupoleSpin + 1) - 34 ms^2 - 5) ms;
      A2 = amplitude/(2 Sqrt[14]) (8 quadrupoleSpin (quadrupoleSpin + 1) - 12 ms^2 - 3) ms;
      A0 = amplitude/Sqrt[5] ( quadrupoleSpin (quadrupoleSpin + 1) -  3 ms^2) ms;
     }];
    If[numberOfGammaAngles == 1, crystal[αPR, βPR, γPR],
      powder[powderFile, numberOfGammaAngles]];
  ];
  (*-- radiofreq --*)
  (*-- provides the matrix of an RF pulse --*)
  radiofreq[ωRF_] := Module[{RF, ymax},
    HRF = Table[0, {dim}, {dim}];
    RF = - π * ωRF * 10^-3 Sqrt[quadrupoleSpin (quadrupoleSpin + 1) - ms (ms - 1)];
    ymax = IntegerPart[quadrupoleSpin + 0.5];
    For [y = 1, y ≤ ymax, y++,
      HRF[[y, y + 1]] = HRF[[y + 1, y]] =
       HRF[[dim - y, dim - y + 1]] = HRF[[dim - y + 1, dim - y]] = RF[[y]];
    ];
  ];
  (*-- pulse --*)
  (*-- generates an RF pulse --*)
  pulse[tau_, ωRF_] := (
     radiofreq[ωRF];
     nbPulseIncrement = IntegerPart[tau / Δt];
     If[nbPulseIncrement == 1, pulsevar, pulsefixe];
    );
  (*-- pulseFunction--*)
```

```
(*-- provides the density matrix describing the spin system --*)
(*-- excited by an RF pulse --*)
pulseFunction[poub_] :=
  Module[{V20, W20, W40, frequency, HQ1, HQ2, Ha, HT, Tp, T, mat, ρ11, ρ22},
  frequency = (poub - 1) * Δt * ωrot;
  V20 =  d1C * Cos[frequency]   +  d2C * Cos[2 frequency]
      + d1S * Sin[frequency]   +  d2S * Sin[2 frequency];

  HQ1 = V20 * A;    HQ2 = 0;

  If[quadrupoleOrder == 2, {
    W20 = d21C * Cos[frequency] + d22C * Cos[2 frequency]
        + d21S * Sin[frequency] + d22S * Sin[2 frequency];
    W40 =  a40
         + d44S * Sin[4 frequency]  +  d44C * Cos[4 frequency]
         + d43S * Sin[3 frequency]  +  d43C * Cos[3 frequency]
         + d42S * Sin[2 frequency]  +  d42C * Cos[2 frequency]
         + d41S * Sin[ frequency]  +  d41C * Cos[ frequency];

    HQ2 = DiagonalMatrix[A4 * W40 + A2 * W20 - A0 * W0 ];
   }];

  Ha = HRF + HQ1 + HQ2;
  {HT, Tp} = Eigensystem[N[Ha]];    T = Transpose[Tp];
  mat = DiagonalMatrix[Exp[-ⅈ * Δt * HT]];
  ρ11 = T.mat.Tp;   ρ22 = T.Conjugate[mat].Tp;   ρ0 = ρ11.ρ0.ρ22;
 ];
(*-- pulsevar, pulse with variable duration--*)
pulsevar := (
  coef4 += 1;
  pulseFunction[coef4];
 );
(*-- pulsefixe, pulse with fixed duration --*)
pulsefixe := Module[{pulseDuration},
  For [b = 1, b ≤ nbPulseIncrement, b++, {
     pulseDuration = b + coef4;
     pulseFunction[pulseDuration];
   }];
  coef4 += nbPulseIncrement;
 ];
(*-- acq0 --*)
(*-- records the first signal amplitude --*)
acq0 := (s[0] = ρ0;);
(*-- acq --*)
(*-- records the signal amplitude --*)
acq[z_] := (s[z] = ρ0;);
(*-- store --*)
(*-- saves the state of the system and the delay from the beginning --*)
store[sr_] := (
  d[sr] = ρ0;
  indicRotor = coef4;
 );
(*-- recall --*)
(*-- provides the state of the system stored previously --*)
```

```
recall[r_] := (
  ρ0 = d[r];
  coef4 = indicRotor;
 );
(*-- filterElt --*)
(*-- keeps the density matrix with specific matrix elements --*)
filterElt[elements_] := Module[{ρ6, long, elt1, elt2},
  ρ6 = Table[0, {dim}, {dim}];
  long = Length[elements];
  For [i = 1, i ≤ long, i++, {
     elt1 = elements[[i, 1]];
     elt2 = elements[[i, 2]];
     ρ6[[elt1, elt2]] = ρ0[[elt1, elt2]];
    }];
  ρ0 = ρ6;
 ];
(*-- filterCoh --*)
(*-- keeps the density matrix with specific coherences --*)
filterCoh[coh_] := Module[{ρ6, quanta, qq, long, x, elt1, elt2},
  ρ6 = Table[0, {dim}, {dim}];

  For [i = 1, i ≤ Length[coh], i++, {
     quanta = coh[[i]];
     qq = Abs [quanta];
     long = dim - qq;
     x = Range[1, long];
     For [z = 1, z ≤ long, z++, {
        If[quanta ≥ 0, {elt1 = x[[z]]; elt2 = x[[z]] + qq},
                     {elt2 = x[[z]]; elt1 = x[[z]] + qq}];
        ρ6[[elt1, elt2]] = ρ0[[elt1, elt2]];
      }];
    }];

  ρ0 = ρ6;
 ];
(*-- tabgraph --*)
(*-- presents the simulated amplitudes in a table --*)
(*-- and plots the simulated amplitudes versus pulse duration --*)
tabgraph[nom_String] := (
  Print["***********************************"];
  For[a = 0, a ≤ np, a++, pas[a] = a Δt;];

  tableau = Table[{t, pas[t], NumberForm[h[t], 10]}, {t, 0, np}];
  Print[TableForm[tableau,
     TableHeadings -> {None, {"Rang", "t(μs)", "intensity"}}]];

  Print["***********************************"];
  ListPlot[Table[{pas[t], h[t]}, {t, 0, np}],
     PlotJoined → True,
     AxesLabel → {"t (μs)", "Intensity (A.U.)"},
     PlotStyle → {Hue[0.1]},
     TextStyle → {FontFamily → "Times", FontSize → 12}];
  Print["***********************************"];
```

```
  Clear[excel];
  excel[data_List] :=
     Module[ { file1 = OpenWrite[nom] },
            Scan[(
                  WriteString[file1, First[#]];
                  Scan[
                      WriteString[file1, "\t", #] &,
                      Rest[#]
                  ];
                  WriteString[file1, "\n"]
                ) &,
             data
            ];
            Close[file1]
     ];
  excel[tableau];
 );
(*-- crystal --*)
(*-- provides the signal amplitude for a single crystal --*)
crystal[αd_, βd_, γd_] := Module[{αangle, βangle, γangle, elt11, elt22},
  αangle = αd * π / 180;    βangle = βd * π / 180;    γangle = γd * π / 180;

  For[i = 0, i ≤ np, i++, h[i] = 0];

  elt11 = detectelt[[1, 1]];  elt22 = detectelt[[1, 2]];

  ρ0 = startOperator; (* initial state *)

  coefv20Rot[αangle, βangle, γangle];
  If[quadrupoleOrder == 2, coefwx0Rot[αangle, βangle, γangle]];

  coef4 = 0;   fsimulation;

  For[i = 0, i ≤ np, i++,
     h[i] = N[Im[s[i][[elt11, elt22]] ] ] / numberOfGammaAngles ];
 ];
(*-- powder --*)
(*-- provides the signal amplitude for a powder sample --*)
powder[rep_, maxγ_] :=
  Module[{αangle, βangle, γangle, proba, elt11, elt22, fileSize, xtalFile},
  For[i = 0, i ≤ np, i++, h[i] = 0];

  elt11 = detectelt[[1, 1]];  elt22 = detectelt[[1, 2]];

  xtalFile = ReadList[rep , {Number, Number, Number}];
  fileSize = Length[xtalFile];
  Print["powderFile: rep", fileSize, "_simp"];

  For[j = 1, j ≤ fileSize, j++, {
     Print[j, "/", fileSize];
     αangle = xtalFile[[j, 1]] * π / 180;
     βangle = xtalFile[[j, 2]] * π / 180;
     proba = xtalFile[[j, 3]];
```

```
For [g = 0, g < maxγ, g++, {
   ρ0 = startOperator; (* initial state *)
   γangle = 2 g π / maxγ;
   coefv20Rot[αangle, βangle, γangle];
   If[quadrupoleOrder == 2, coefwx0Rot[αangle, βangle, γangle]];

   coef4 = 0;  fsimulation;

   For[k = 0, k ≤ np, k++, h[k] = h[k] + proba * s[k]];
  }];
 }];

For[i = 0, i ≤ np, i++,
   h[i] = N[Im[h[i][[elt11, elt22]] ] ] / numberOfGammaAngles ];
];
(*-- coefv20Rot --*)
(*-- provides the parameters dxC and dxS of V20 --*)
coefv20Rot[αeuler_, βeuler_, γeuler_] := Module[{a1, a2, b1, b2},
  c2α = Cos[2 αeuler];   s2α = Sin[2 αeuler];
  cβ = Cos[βeuler];      sβ = Sin[βeuler];
  c2β = Cos[2 βeuler];   s2β = Sin[2 βeuler];
  cγ = Cos[γeuler];      sγ = Sin[γeuler];
  c2γ = Cos[2 γeuler];   s2γ = Sin[2 γeuler];
  a1 = -η * s2α * sβ / √3;        b1 = - (-3 + η * c2α) s2β / (2 √3);
  a2 = -η * cβ * s2α / √6;        b2 = - (η * c2α * (3 + c2β) + 6 sβ^2) / (4 √6);
  d2S = a2 * c2γ + b2 * s2γ;      d1S = a1 * cγ + b1 * sγ;
  d2C = a2 * s2γ - b2 * c2γ;      d1C = a1 * sγ - b1 * cγ;
 ];
(*-- coefwx0Rot --*)
(*-- provides the parameters d4xC and d4xS of W40 --*)
(*-- and the parameters d2xC and d2xS of W20 --*)
coefwx0Rot[αeuler_, βeuler_, γeuler_] :=
  Module[{a22, b22, a21, b21, a41, a42, a43, a44, b41, b42, b43, b44},
  c4α = Cos[4 αeuler];
  c2α = Cos[2 αeuler];   s2α = Sin[2 αeuler];

  c4β = Cos[4 βeuler];   s4β = Sin[4 βeuler];
  c2β = Cos[2 βeuler];   s2β = Sin[2 βeuler];
  cβ = Cos[βeuler];      sβ = Sin[βeuler];

  cγ = Cos[γeuler];      sγ = Sin[γeuler];
  c2γ = Cos[2 γeuler];   s2γ = Sin[2 γeuler];
  c3γ = Cos[3 γeuler];   s3γ = Sin[3 γeuler];
  c4γ = Cos[4 γeuler];   s4γ = Sin[4 γeuler];

  a22 = -√(2/7) η * cβ * s2α;     b22 = - (η * c2α (3 + c2β) + sβ^2 (-3 + η^2)) / (2 √14);
  a21 = - (2 / √7) η * sβ * s2α;  b21 = (-3 - 2 c2α * η + η^2) s2β / (2 √7);
  d22S = a22 * c2γ + b22 * s2γ;   d21S = a21 * cγ + b21 * sγ;
  d22C = a22 * s2γ - b22 * c2γ;   d21C = a21 * sγ - b21 * cγ;
  a40 =
```

$$\frac{-\sqrt{7/10}}{2304}\left((18+\eta^2)\,(9+20\,c_{2\beta}+35\,c_{4\beta})+240\,\eta\,c_{2\alpha}\,(5+7\,c_{2\beta})\,s_\beta^2+280\,\eta^2\,c_{4\alpha}\,s_\beta^4\right);$$

$$a41 = \left(\sqrt{5/7}\,/72\right)\eta * s_{2\alpha} * s_\beta\,(15+21\,c_{2\beta}+14\,\eta * c_{2\alpha} * s_\beta^2);$$

$$b41 = \left(\sqrt{5/7}\,/288\right)\left((-18-\eta^2-12\,\eta * c_{2\alpha}+7\,\eta^2 * c_{4\alpha})\,s_{2\beta}-7\,(-3+\eta * c_{2\alpha})^2\,s_{4\beta}\right);$$

$$a42 = -\left(\sqrt{5/14}\,/18\right)\eta * c_\beta * s_{2\alpha}\,(-9+21\,c_{2\beta}+14\,\eta * c_{2\alpha} * s_\beta^2);$$

$$b42 = \frac{-1}{72}\sqrt{5/14}\left(3\,\eta * c_{2\alpha}\,(5+4\,c_{2\beta}+7\,c_{4\beta})+(7\,\eta^2 * c_{4\alpha}\,(3+c_{2\beta})+(18+\eta^2)\,(5+7\,c_{2\beta})) * s_\beta^2\right);$$

$$a43 = -\left(\sqrt{35}\,/72\right)\eta\,(-3-9\,c_{2\beta}+\eta * c_{2\alpha} * (5+3 * c_{2\beta})) * s_{2\alpha} * s_\beta;$$

$$b43 = -\left(\sqrt{35}\,/288\right)(-18-\eta^2-12\,\eta * c_{2\alpha}+7\,\eta^2 * c_{4\alpha}+2\,(-3+\eta * c_{2\alpha})^2 * c_{2\beta})\,s_{2\beta};$$

$$a44 = -\left(\sqrt{35/2}\,/72\right)\eta * c_\beta * s_{2\alpha}\,(\eta * c_{2\alpha} * (3+c_{2\beta})+6\,s_\beta^2);$$

$$b44 = -\frac{\sqrt{35/2}}{2304}\left(\eta^2 * c_{4\alpha}\,(35+28\,c_{2\beta}+c_{4\beta})+48\,\eta * c_{2\alpha}\,(3+c_{2\beta})\,s_\beta^2+8\,(18+\eta^2)\,s_\beta^4\right);$$

$$d41S = a41 * c_\gamma + b41 * s_\gamma; \quad d41C = a41 * s_\gamma - b41 * c_\gamma;$$

$$d42S = a42 * c_{2\gamma} + b42 * s_{2\gamma}; \quad d42C = a42 * s_{2\gamma} - b42 * c_{2\gamma};$$

$$d43S = a43 * c_{3\gamma} + b43 * s_{3\gamma}; \quad d43C = a43 * s_{3\gamma} - b43 * c_{3\gamma};$$

$$d44S = a44 * c_{4\gamma} + b44 * s_{4\gamma}; \quad d44C = a44 * s_{4\gamma} - b44 * c_{4\gamma};$$

```
 ];
(* Save these functions in a notebook called QUADRUPOLE *)
Save["QUADRUPOLE", run, radiofreq, pulse, pulseFunction,
  pulsevar, pulsefixe, acq0, acq, store, recall, filterElt,
  filterCoh, tabgraph, crystal, powder, coefv20Rot, coefwx0Rot];
```