■ Concepts and Methods of 2D Infrared Spectroscopy
Peter Hamm and Martin T. Zanni

This progam was used to produce Figs. 4.14 and 10.8.

Number of single excited states, and their frequencies and transition dipoles, taken from the Gaussian output. An offset-frequency is subtracted (and added back in after the Fouriertransform) to save computation time. The two transition dipoles are in the x- and y-axis, respectively.

```
In[1]:= n = 2;
        w = {1889.184, 1947.621};
        mu = {Sqrt[.908] {1., 0, 0}, Sqrt[.671] {0, 1., 0}};
        woff = 1915;
        w -= woff;
```

Number of double excited states, and their frequencies and transition dipoles, taken from the Gaussian output. We use the harmonic approximation for the transition dipoles.

```
In[6]:= n2 = 3;
        w2 = {3767.517, 3883.911, 3813.698};
        w2 -= 2 woff;
        mu2 = {{Sqrt[2] mu[[1]], {0, 0, 0}, mu[[2]]}, {{0, 0, 0}, Sqrt[2] mu[[2]], mu[[1]]}};
```

Parameters and constants

```
In[10]:= nt = 128;
         dt = 0.25;
         T2 = 2.;
         c = .188;
         t2 = 0;
```

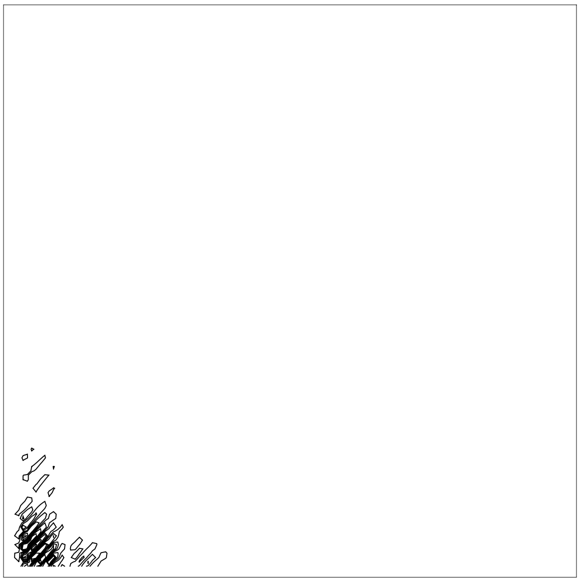
Calculate response functions for rephasing and non-rephasing diagrams (Eqs. 10.11, 10.12, and 10.13). The first time-point needs to be halved (Sect.9.5.3).

```
In[27]:= Rr = Table[0., {it1, 1, nt}, {it3, 1, nt}];
         Rnr = Table[0., {it1, 1, nt}, {it3, 1, nt}];
         For[j = 1, j ≤ n, j++,
          For[i = 1, i ≤ n, i++,
           mui = Sqrt[mu[[i]].mu[[i]]];
           muj = Sqrt[mu[[j]].mu[[j]]];
           cos1 = mu[[i]].mu[[j]] / mui / muj;
           angle = (1 + 2 cos1^2) / 15;
           dipole = mui^2 * muj^2;
           (*Rephasing diagram R1*)
           Rr -= dipole angle Table[Exp[+I w[[j]] c (it1 - 1) dt - I w[[i]] c (it3 - 1) dt +
                I (w[[j]] - w[[i]]) c t2 - (it1 + it3 - 2) dt / T2] , {it1, 1, nt}, {it3, 1, nt}];
           (*Rephasing diagram R2*)
           Rr -= dipole angle
             Table[Exp[+I w[[j]] c (it1 - 1) dt - I w[[i]] c (it3 - 1) dt - (it1 + it3 - 2) dt / T2] ,
              {it1, 1, nt}, {it3, 1, nt}];
```

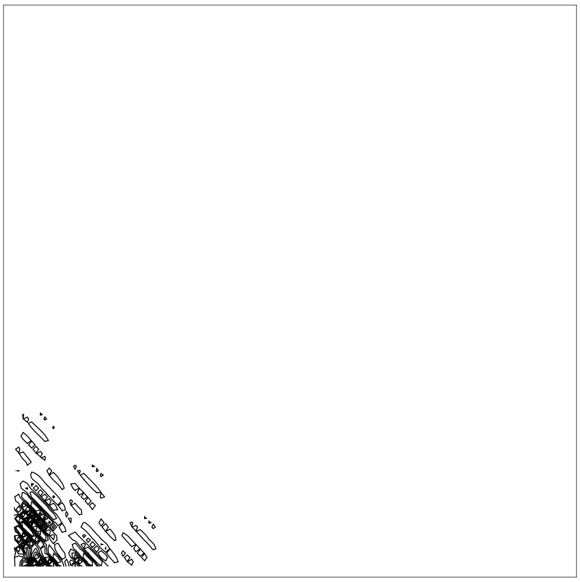
```
   For[k = 1, k ≤ n2, k++,
    mujk = Sqrt[mu2[[j, k]].mu2[[j, k]]];
    muik = Sqrt[mu2[[i, k]].mu2[[i, k]]];
    dipole = mui * muj * muik * mujk;
    cos2 = If[muik ≠ 0 && mujk ≠ 0, mu2[[i, k]].mu2[[j, k]] / muik / mujk, 0];
    cos3 = If[muik ≠ 0 , mu[[i]].mu2[[i, k]] / mui / muik, 0];
    cos4 = If[mujk ≠ 0, mu[[j]].mu2[[j, k]] / muj / mujk, 0];
    cos5 = If[mujk ≠ 0, mu[[i]].mu2[[j, k]] / mui / mujk, 0];
    cos6 = If[muik ≠ 0 , mu[[j]].mu2[[i, k]] / muj / muik, 0];
    angle = (cos1 cos2 + cos3 cos4 + cos5 cos6) / 15;
    (*Rephasing diagram R3*)
    Rr +=
     angle dipole Table[Exp[+I w[[j]] c (it1 - 1) dt - I (w2[[k]] - w[[j]]) c (it3 - 1) dt +
         I (w[[j]] - w[[i]]) c t2 - (it1 + it3 - 2) dt / T2] , {it1, 1, nt}, {it3, 1, nt}];
   ]
  ]
 ]

 For[j = 1, j ≤ n, j++,
  For[i = 1, i ≤ n, i++,
   mui = Sqrt[mu[[i]].mu[[i]]];
   muj = Sqrt[mu[[j]].mu[[j]]];
   cos1 = mu[[i]].mu[[j]] / mui / muj;
   angle = (1 + 2 cos1^2) / 15;
   (*Non-rephasing diagram R4*)
   Rnr -= dipole angle Table[Exp[-I w[[j]] c (it1 - 1) dt - I w[[j]] c (it3 - 1) dt -
        I (w[[j]] - w[[i]]) c t2 - (it1 + it3 - 2) dt / T2] , {it1, 1, nt}, {it3, 1, nt}];
   (*Non-rephasing Diagram R5*)
   Rnr -= dipole angle
     Table[Exp[-I w[[j]] c (it1 - 1) dt - I w[[i]] c (it3 - 1) dt - (it1 + it3 - 2) dt / T2] ,
      {it1, 1, nt}, {it3, 1, nt}];
   For[k = 1, k ≤ n2, k++,
    mujk = Sqrt[mu2[[j, k]].mu2[[j, k]]];
    muik = Sqrt[mu2[[i, k]].mu2[[i, k]]];
    dipole = mui * muj * muik * mujk;
    cos2 = If[muik ≠ 0 && mujk ≠ 0, mu2[[i, k]].mu2[[j, k]] / muik / mujk, 0];
    cos3 = If[muik ≠ 0 , mu[[i]].mu2[[i, k]] / mui / muik, 0];
    cos4 = If[mujk ≠ 0, mu[[j]].mu2[[j, k]] / muj / mujk, 0];
    cos5 = If[mujk ≠ 0, mu[[i]].mu2[[j, k]] / mui / mujk, 0];
    cos6 = If[muik ≠ 0 , mu[[j]].mu2[[i, k]] / muj / muik, 0];
    angle = (cos1 cos2 + cos3 cos4 + cos5 cos6) / 15;
    (*Non-rephasing diagram R6*)
    Rnr +=
     angle dipole Table[Exp[-I w[[j]] c (it1 - 1) dt - I (w2[[k]] - w[[i]]) c (it3 - 1) dt -
         I (w[[j]] - w[[i]]) c t2 - (it1 + it3 - 2) dt / T2] , {it1, 1, nt}, {it3, 1, nt}];
   ]
  ]
 ]
 For[i = 1, i ≤ nt, i++, Rnr[[i, 1]] /= 2; Rr[[i, 1]] /= 2];
 For[i = 2, i ≤ nt, i++, Rnr[[1, i]] /= 2; Rr[[1, i]] /= 2];
```

Plot time-domain data

```
In[33]:= ListContourPlot[Re[Rr], PlotRange → All, Contours → 10,
          ContourShading → False, FrameTicks → None, FrameLabel → {t3, t1}]
         ListContourPlot[Re[Rnr], PlotRange → All, Contours → 10,
          ContourShading → False, FrameTicks → None, FrameLabel → {t3, t1}]
```




```
Out[33]= - ContourGraphics -
```




```
Out[34]= - ContourGraphics -
```

Perform 2D Fourier transform and re-order data so that w1=w3=0 is centerd in the middle. Frequency axis w1 is inverted.

```
In[35]:= spectrum2Dr = Fourier[Rr];
         spectrum2Dr = Reverse[Drop[RotateRight[spectrum2Dr, {nt / 2, nt / 2}], 1, 1]];

         spectrum2Dnr = Fourier[Rnr];
         spectrum2Dnr = Drop[RotateRight[spectrum2Dnr, {nt / 2, nt / 2}], 1, 1];

         spectrum2Dabs = Re[spectrum2Dr + spectrum2Dnr];
```

Plot purely absorptive spectrum

```
In[55]:= ticks = Table[{ (w - woff) * .188 * nt * dt / 2 / Pi + nt / 2, w}, {w, 1800, 2000, 50}];
         max = Max[{Max[spectrum2Dabs], -Min[spectrum2Dabs]}];
         p1 = ListContourPlot[spectrum2Dabs,
            PlotRange → {0, max}, ContourShading → False, Contours → 20, Ticks → None,
            ContourStyle → {RGBColor[0, 0, 1]}, DisplayFunction -> Identity];
         p2 = ListContourPlot[spectrum2Dabs, PlotRange → {-max, 0},
            ContourShading → False, Contours → 20, Ticks → None,
            ContourStyle → {RGBColor[1, 0, 0]}, DisplayFunction -> Identity];
         Show[{p1, p2}, FrameTicks → {ticks, ticks, None, None},
           PlotRange → {{1, nt - 1}, {1, nt - 1}}, DisplayFunction → $DisplayFunction];
```