■ Concepts and Methods of 2D Infrared Spectroscopy

Peter Hamm and Martin T. Zanni

This progam was used to produce Figs. 10.6, i.e. the 2D IR lineshape of the OH stretch vibration in water, without using the Cumulant expansion

Set working directory (you will have to change it)

```
In[1]:= SetDirectory["C:\Dokumente und Einstellungen\p.hamm\
          Eigene Dateien/PCI/projekte/2010/2D-IR Book/SecSimul/SimulWater"];
```

Constants and proportionaltiy factor that translates forces into a frequency shift (Eq. 10.4)

```
In[12]:= NA = 6.023 * 10^23;
         w0 = 3550;
         mass = 1.66 * 10^-27;
         mH = 1.;
         mO = 16.;
         mu = mH * mO / (mH + mO);
         c = 3 * 10^10;
         h = 6.626 * 10^-34;
         Δ = 240;
         fact = Sqrt[9 / 4 (Δ / w0) 10^18 * 10^6 / NA^2 / (w0 * c * h) / (mu * mass) * 10^-24] / .188;
```

Read MD output files generated by Gromacs (see README file)

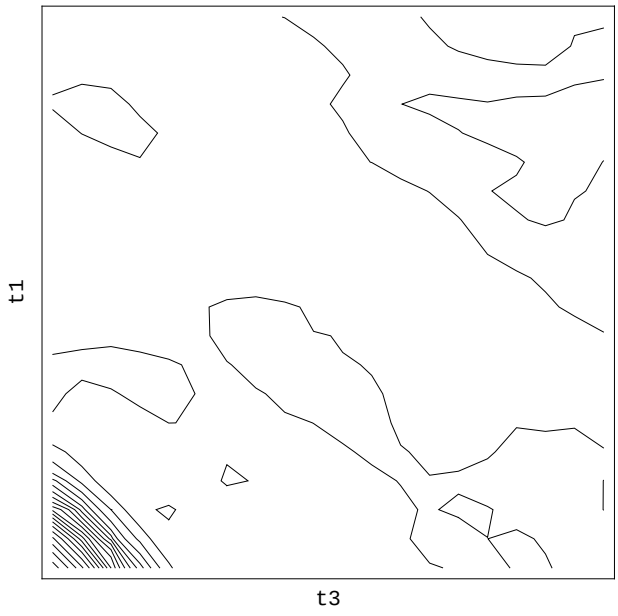
```
In[22]:= coord = ReadList["coord.xvg", {Real, Real, Real, Real, Real, Real, Real}];
         force = ReadList["force.xvg", {Real, Real, Real, Real, Real, Real, Real}];
```

Calculate frequency shift according to Eqs. 10.2 and 10.4 subtract mean of it, and translate it in units of [ps-1]. Take only time points separated by 4 step to save computation time

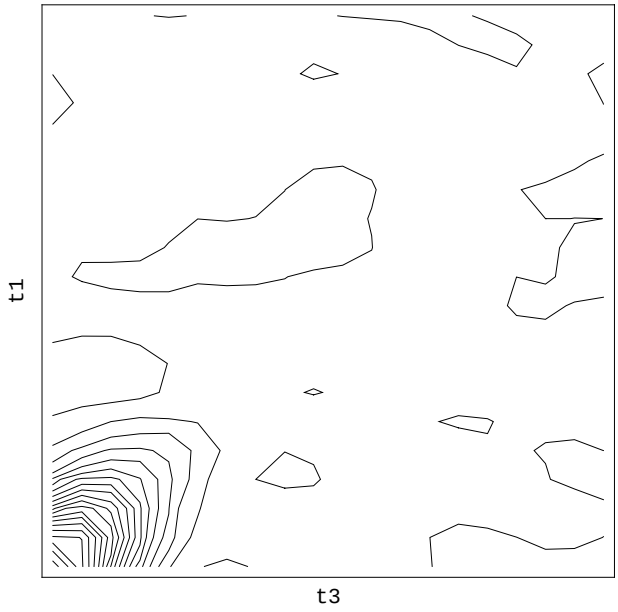
```
In[24]:= n = Length[force];
         dt = (force[[2, 1]] - force[[1, 1]]);
         bondOH = Table[{coord[[i, 5]] - coord[[i, 2]],
             coord[[i, 6]] - coord[[i, 3]], coord[[i, 7]] - coord[[i, 4]]} , {i, 1, n}];
         length = Sqrt[bondOH[[1, 1]]^2 + bondOH[[1, 2]]^2 + bondOH[[1, 3]]^2];
         forceOH = Table[{(force[[i, 5]] / mH - force[[i, 2]] / mO) * mu,
             (force[[i, 6]] / mH - force[[i, 3]] / mO) * mu,
             (force[[i, 7]] / mH - force[[i, 4]] / mO) * mu} , {i, 1, n}];
         step = 4;
         dt = dt * step;
         w = -Table[Sum[(forceOH[[i + j, 1]] * bondOH[[i + j, 1]] + forceOH[[i + j, 2]] *
                    bondOH[[i + j, 2]] + forceOH[[i + j, 3]] * bondOH[[i + j, 3]]),
                {j, 0, step - 1}], {i, 1, n - step, step}] / step / length * fact;
         n = Length[w];
         wmean = Sum[w[[i]], {i, 1, n}] / n;
         w = (w - wmean) * .188;
```

Calculate response functions through the integrals of Eq.10.10, for t2=dt*it2. If nstep=1, it will take some time, but gets out the best possible signal-to-noise form the 1 ns trajectory. If nstep=10, it is significantly faster, but results will be more noisy.

```
In[45]:= it2 = 2;
         nt = 20;
         nstep = 10;
         Rnr = Sum[Table[Exp[-I Sum[w[[is + itau]] dt, {itau, 1, it1}] -
                 I Sum[w[[is + itau]] dt, {itau, it1 + it2 + 1, it1 + it2 + it3}]] ,
               {it1, 0, nt - 1}, {it3, 0, nt - 1}], {is, 1, n - 200, nstep}] / (n - 200.) * nstep;
         ListContourPlot[Re[Rnr], PlotRange → All, Contours → 20, ContourShading → False,
          FrameTicks → None, FrameLabel → {t3, t1}]
         Rr = Sum[Table[Exp[+I Sum[w[[is + itau]] dt, {itau, 1, it1}] -
                 I Sum[w[[is + itau]] dt, {itau, it1 + it2 + 1, it1 + it2 + it3}]] ,
               {it1, 0, nt - 1}, {it3, 0, nt - 1}], {is, 1, n - 200, nstep}] / (n - 200.) * nstep;
         ListContourPlot[Re[Rr], PlotRange → All, Contours → 20, ContourShading → False,
          FrameTicks → None, FrameLabel → {t3, t1}]
         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];
```




```
Out[49]= - ContourGraphics -
```



Out[51]= - ContourGraphics -

Add anharmonically shifted term to it (Eq. 10.10). Note that we shift both peaks by +/-Δ/2, rather than just the 1-2 peak by -Δ. This is to center the spectrum as much as possible in teh very small frequency we use to save computation time. For the same reason, we the mean frequency wmean is not added, which allows us to us a much longer time-step.

```
In[54]:= RnrAnh = Table[ Rnr[[it1, it3]] *
             (Exp[-I * Δ / 2 * .188 * dt * (it3 - 1)] - Exp[I * Δ / 2 * .188 * dt * (it3 - 1)]),
           {it1, 1, nt}, {it3, 1, nt}];
         RrAnh = Table[ Rr[[it1, it3]] * (Exp[-I * Δ / 2 * .188 * dt * (it3 - 1)] -
              Exp[I * Δ / 2 * .188 * dt * (it3 - 1)]), {it1, 1, nt}, {it3, 1, nt}];
```

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

```
In[61]:= spectrum2Dr = Fourier[RrAnh];
         spectrum2Dr = Reverse[Drop[RotateRight[spectrum2Dr, {nt / 2, nt / 2}], 1, 1]];

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

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

Plot purely absorptive spectrum

In[86]:=
```
ticks = Table[{ w * .188 * nt * dt / 2 / Pi + nt / 2, w}, {w, -300, 300, 150}];
max = Max[{Max[spectrum2Dabs], -Min[spectrum2Dabs]}];
p1 = ListContourPlot[spectrum2Dabs,
    PlotRange → {0, max}, ContourShading → False, Contours → 5, Ticks → None,
    ContourStyle → {RGBColor[0, 0, 1]}, DisplayFunction -> Identity];
p2 = ListContourPlot[spectrum2Dabs, PlotRange → {-max, 0},
    ContourShading → False, Contours → 5, 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];
```