Practical session 5: *Ab initio* Molecular Dynamics and Thermodynamic Properties

Prepared by Christian Carbogno, Luca M. Ghiringhelli, and Mariana Rossi

July 20, 2011

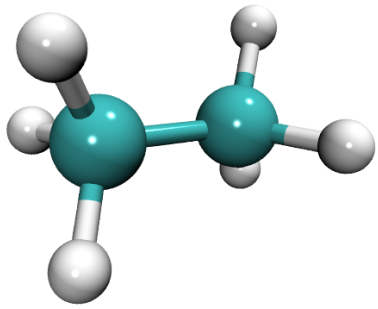

Figure 1: The C_2H_6 (ethane) molecule.

All the useful files for this tutorial (together with solutions) are located in:
`/pub/tutorial5`
All the scripts for this tutorial are located, as usual, in:
`/pub/tutorial5/scripts`

The microcanonical ensemble

Exercise 1: The importance of the SC convergence criteria

To start our first Molecular Dynamics (MD) simulation, we will use a very simple molecule, which is ethane (C_2H_6), and we will investigate the importance of the self-consistency convergence criteria when simulating the microcanonical ensemble.

A geometry for C_2H_6 is already provided in the folder `exercise_1` (all files for exercise x are located in the folder named `exercise_x`). You will notice that this file also contains specific velocities for each atom, so that the Molecular Dynamics run will not use a random initialization. These velocities come from a previous equilibration of the molecule at ∼300K. You will see how such a thermalization is done in exercise 3.

- First, build an input file for FHI-aims using the LDA (**pw-lda**) functional, no spin polarization (**spin none**), and using the "**light**" standards for the species. Please refer to the manual for the exact syntax of these flags.

  IMPORTANT: Add the following flag to your control.in file:

  ```
  wf_extrapolation none
  ```

This flag will be explained in a moment.

Do not add any flags that we do not mention. They are not needed and might hinder the performance of the calculation.

- Start a 0.15ps MD run in the microcanonical ensemble, using a 0.0005ps ($\Delta t = 0.5$fs) time step (flags `MD_run` and `MD_time_step` respectively), with the following "loose" self consistency convergence criteria:

```
MD_run 0.15 NVE
MD_time_step 0.0005
sc_accuracy_rho 1E-2
sc_accuracy_eev 1E-1
sc_accuracy_etot 1E-3
sc_accuracy_forces 5E-2
```

In order to start the run type:
`mpirun -np 4 "FHI-aims binary" > "output file" &`

  - The & puts the run in the background, so that the output file is created, but the terminal is free for other use.
  - If you anyway would like to have a dynamic view of what happens in your output, after starting the simulation you can type:
    `tail -f "output file"`
  - ATTENTION: do not start another FHI-aims run simultaneously.
    That would slow down BOTH calculations considerably.

- When the previous calculation is over, run another simulation, keeping all parameters mentioned above but changing the name of the output and also changing to the "accurate" self-consistency criteria:

```
sc_accuracy_rho 1E-5
sc_accuracy_eev 1E-4
sc_accuracy_etot 1E-6
sc_accuracy_forces 5E-4
```

  When it is done, use the "aims_MD_eval.pl" script to analyze your run by typing in the terminal:

  `perl aims_MD_eval.pl "FHI-aims-output-file" > "script-output-file"`

  Do this for both outputs. Now plot the total energy (fifth column of the script output file) vs. the simulation time (first column of the script output file) in `xmgrace`. For a short guide on how to plot files with multiple columns in `xmgrace` see the Appendix to this tutorial.

  Can you see how the energy drifts with the "light" settings?

- Now do an extra simulation, also of 0.15ps in total, using the same loose accuracy settings of the first simulation, but set the flag:

```
wf_extrapolation polynomial 3 1
```

  When it is done, use the "aims_MD_eval.pl" script to analyze your run again. Compare the total energy of this simulation (fifth column of the script output file) vs. the time of simulation (first column of the script output file) with the other two from above.

  Do you see that the energy drift is diminished?

Ideally, there should be no energy drift whatsoever, since the energy is conserved in the micro-canonical ensemble. The reason for this drift is that we leave the true Born-Oppenheimer surface if we don't converge well our electronic structure; this leads to an unphysical (and undesirable) energy drift.

The flag `wf_extrapolation` specifies which type of wave-function extrapolation is being used, which in this case is just a polynomial extrapolation. A pictorial representation of several polynomial extrapolations can be seen in Figure 2. The numbers "3 1" means that 3 points are considered, the polynomial is exact up to first order (1) and the higher odd orders (3, in this case), are used to enhance the time reversibility of the algorithm [1, 2].

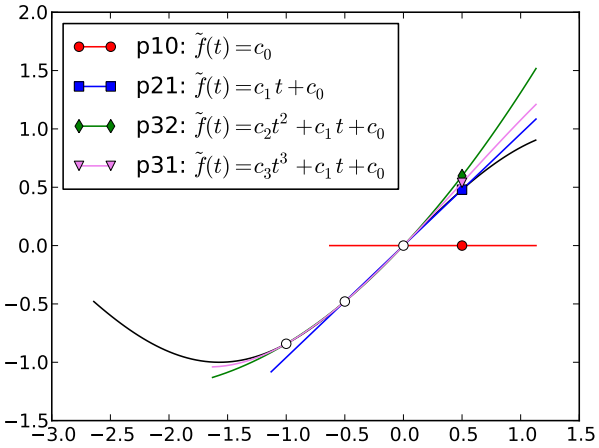

Figure 2: Various types of extrapolations of a model function, with the respective formulas

The extrapolation reduces a possible energy drift in the MD runs, even for moderate accuracy settings, and additionally (depending on the settings) can lower the number of SCF cycles per time step. This makes your simulation faster. However, if the force accuracy is low, dynamical quantities like the ones that will be calculated in Exercise 4 (auto-correlation functions) will not be accurate, even if no energy drift is seen.

Timing: ∼3 minutes per simulation, ∼20 minutes total

Exercise 2: The importance of the time step size

In the second exercise, we will investigate the effects of the time step size in a microcanonical simulation. For a better illustration, we will not only consider the C_2H_6 molecule, but also its heavier, deuterated counterpart C_2D_6[1]. In order to speed up the exercise, each group will simulate either C_2H_6 or C_2D_6. Geometry files, containing both C_2H_6 or C_2D_6 are already provided in the folder `exercise_2`. The species default for deuterium, that you should put in your `control.in` file, is provided in this exercise's folder. The suggested choice of the molecule will be made clear (announced) by the tutors at hand.

- Copy the geometry file corresponding to the molecule you will calculate to `geometry.in`. If you are doing the deuterated molecule, append to your `control.in` file from the last exercise the deuterium species default, found in this exercise's folder. Take few moments to understand the differences between the hydrogen and the deuterium species default. With respect to exercise 1, the following modifications should be made in your `control.in`:
  - Increase the time step (Δt) of the MD simulation to 0.001ps
  - Switch off the wave-function extrapolation (`wf_extrapolation none`)
  - Keep the **accurate** settings for the self-consistency convergence parameters

  Then run FHI-aims (total of 0.15ps) and keep the output.
- Now increase the time step to 0.003ps and run the simulation again, redirecting the output to another file.
- As done in Exercise 1, plot the total energy vs. simulation time in `xmgrace` for:
  - The result you obtained in exercise 1 for $\Delta t = 0.0005$ps (Only if you are doing C_2H_6!)
  - The run you just obtained with $\Delta t = 0.001$ps
  - The run you just obtained with $\Delta t = 0.003$ps

  How do the energy fluctuations develop? Do you notice something strange happening for the $\Delta t = 0.003$ps run?

From a numerical point of view, a larger time step is desirable, since it allows to assess longer trajectories in shorter computational times. Notice, however, that the $\Delta t = 0.003$ps simulation diverges for C_2H_6. In fact, the molecule dissociates. You can inspect the dynamics of the molecule by running

```
create_xyz_movie.pl "FHI-aims-output-file" > "script-output-file".xyz
```

on your output and opening it in VMD (or Jmol, or Molden, whichever you prefer). The reason for the dissociation is that the integrator is unable to deal with these "big" time steps. This integrator uses a simple Verlet algorithm [3], where the error in the trajectory goes with Δt^4. If you are simulating C_2D_6, the $\Delta t = 0.003$ps simulation does not diverge, although the energy fluctuations become very large. You can also look at the dynamics of this molecule in VMD.

Although such large energy fluctuations would be already not accurate enough for a production run with this molecule, the fact that it does not explode illustrates an important point: the largest Δt that can be used in a particular integration algorithm depends on the highest vibrational frequency of the system. Since the D atoms, being heavier, have a larger vibrational period (do you understand why this is obvious?), the used Δt can also be larger.

Timing: ∼20 minutes total

[1] The deuterium atom has one proton and one neutron in its nucleus, being nearly twice as heavy as hydrogen.

The canonical ensemble

Exercise 3: Testing thermostats

Most "real-life" experiments cannot be done in a situation where the energy is explicitly kept constant, but where other quantities like the average temperature for instance. In order to simulate such a canonical ensemble, the system has to be coupled to a heat bath. From a statistical mechanics point of view, the average kinetic energy in the canonical ensemble follows the equipartition theorem, which says that it is equally distributed on the various degrees of freedom of the system. Therefore, the momenta $\vec{p} = M\vec{v}$ follow the Maxwell-Boltzmann (MB) distribution:

$$P(|\vec{p}|) = \left(\frac{\beta}{2\pi M}\right)^{3/2} \exp\left(-\beta|\vec{p}|^2/(2M)\right). \tag{1}$$

The instantaneous temperature is given by the relation $T = \frac{|\vec{p}|^2}{3MNk_B}$, where M is the mass of the system and N is the number atoms. This means that the temperature is not constant but can (and should) fluctuate around the average value. The theoretical standard deviation is $\sigma^2 = \frac{2T^2}{3N}$.

Here we apply three schemes which simulate a thermostat in MD. Below is a short summary of the thermostats we will use in this tutorial:

1. Stochastic / velocity-resetting thermostat: the Andersen thermostat [3, 4].
   This thermostat tries to apply the concept of "coupling with a heat bath" almost literally. In practice, occasionally a particle is randomly selected and its velocity is drawn from the MB distribution at the target temperature. The algorithm requires the specification of the temperature and of the coupling parameter ν, so that the probability that a particle is selected in a time step Δt is $\nu\Delta t$.

2. Velocity rescaling: the Berendsen temperature coupling (not sampling the canonical ensemble!) [3, 7]
   According to the Berendsen algorithm, a small deviation of the instantaneous kinetic energy K from the target kinetic energy $\overline{K}$ is corrected as follows:

   $$dK = \left(\overline{K} - K(t)\right)\frac{dt}{\tau} \tag{2}$$

   which describes an exponential decay of a kinetic energy perturbation. In practice, the velocities are rescaled at each time step by a factor λ:

   $$\lambda = \left[1 + \frac{\Delta t}{\tau}\left(\frac{T_0}{T} - 1\right)\right]^{1/2} \tag{3}$$

   where T_0 is the target temperature, T is the actual one, Δt is the time step and τ is a parameter that controls the strength of the coupling. If $\tau = \Delta t$ the scheme brings back the actual temperature to the target one exactly at each time step. With increasing τ the rescaling becomes milder and milder.

3. Stochastic / velocity-rescaling thermostat: the Bussi-Donadio-Parrinello thermostat[5].
   In this algorithm, a deviation of the instantaneous kinetic energy is corrected in the following way:

   $$dK = \left(\overline{K} - K(t)\right)\frac{dt}{\tau} + 2\sqrt{\frac{K(t)\overline{K}}{N_f}}\frac{dW(t)}{\sqrt{\tau}} \tag{4}$$

   where $\overline{K}$ is the target kinetic energy, $K(t)$ is the instantaneous kinetic energy, τ is the relaxation time of the thermostat, N_f is the number of degrees of freedom, and dW is a Wiener noise [2].
   In practice the trajectory is first propagated for one time step with e.g. a velocity-verlet

[2] An example of a Wiener process $W(t)$ is the Brownian motion (you might have heard about it...). $W(t)$ has the following characteristics: $W(0) = 0$; $W(t)$ is continuous; the increments are independent and $W(t_2) - W(t_1)$ is a Gaussian with average 0 and $\sigma = t_2 - t_1$

integrator and the new velocities are calculated as usual. Then, the new kinetic energy K is evaluated and the velocities are rescaled by a factor α such that:

$$\begin{aligned}\alpha^2 &= e^{-\Delta t/\tau} + \frac{\overline{K}}{N_f K}\left(1 - e^{-\Delta t/\tau}\right)\left(R_1^2 + \sum_{i=2}^{N_f} R_i^2\right) \\ &+ 2e^{-\Delta t/2\tau}\sqrt{\frac{\overline{K}}{N_f K}\left(1 - e^{-\Delta t/\tau}\right)} R_1\end{aligned}$$

where the R_i's are independent random numbers from a Gaussian distribution with unitary variance (Note that $\sum_{i=2}^{N_f} R_i^2$ can be drawn directly from a suitable Gamma distribution).

For this thermostat a conserved pseudo-Hamiltonian $\tilde{H}(t)$ can be defined:

$$\tilde{H}(t) = H(t) - \int_0^t \left(\overline{K} - K(t')\right)\frac{dt'}{\tau} - 2\int_0^t \sqrt{\frac{K(t')\overline{K}}{N_f}}\frac{dW(t')}{\sqrt{\tau}}$$

where $H(t)$ is the total energy of the atomic system.

The Bussi-Donadio-Parrinello thermostat yields the correct distribution of K, does not have ergodicity problems, does not perturb the dynamics, and its accuracy and efficiency is rather independent of τ.

4. Extended Lagrangian approach: the Nosé-Hoover thermostat [3, 6].
Equations of motion derived from the Lagrangian of the system conserve the total energy of the system. One can write an *extended* Lagrangian, by adding fictitious degrees of freedom, such that the overall total energy is conserved but the atomic subsystem can span ensembles other than microcanonical. With the Nosé-Hoover Lagrangian, the atomic subsystem samples the canonical ensemble. The equations of motion of the Nose-Hoover thermostat are:

$$\dot{\mathbf{r}}_i = \mathbf{p}_i/m_i \tag{5}$$

$$\dot{\mathbf{p}}_i = -\frac{\partial \mathscr{U}\left(\mathbf{r}^N\right)}{\partial \mathbf{r}_i} - \frac{\Pi \mathbf{p}_i}{Q} \tag{6}$$

$$\dot{\eta} = \frac{\Pi}{Q} \tag{7}$$

$$\dot{\Pi} = \left(\sum_i \frac{\mathbf{p}_i^2}{m_i} - \frac{g}{\beta}\right) \tag{8}$$

where g is the number of degrees of freedom of the system, $\mathscr{U}$ is the potential energy, Q the "thermostat mass", and $\mathbf{p}_i$ and m_i the momenta and masses of the ith particle of the system, respectively. The conjugated momentum Π of the extra coordinate η acts as a fluctuating drag parameter to the atomic subsystem. The conserved energy associated to the equations of motion is:

$$\mathscr{E} = \sum_i \frac{\mathbf{p}_i^2}{2m_i} + \mathscr{U}\left(\mathbf{r}^N\right) + \frac{1}{2}\frac{\Pi^2}{Q} + g\frac{\eta}{\beta} \tag{9}$$

- Use the same `control.in` (tight convergence settings) and `geometry.in` for C_2H_6 (including the velocities that are in there) that you used in exercise 1. You will only have to change the MD related flags, as explained below.
- Each group will run only one of the four different thermostats discussed above, and the division will be made clear by the tutors at hand. The runs will be 0.5 ps in total, using a **0.001**ps time step. Use the scheduler to set the temperature to 300 and 800 K, respectively. The key words you should use are shown below, and the `MD_run` flag should be erased from your input. The general syntax of the thermostat flags is:

  `<MD command> <time> <NVT_thermostat name> <temperature> <thermostat parameter>`

You should provide an educated guess for the value of each thermostat's parameter, following the explanations given above and the hints below. If you have time, try to play with these thermostat parameters afterwards.

1. Nosé-hoover

   `MD_schedule`

   `MD_segment 0.15 NVT_nose-hoover 300` Q

   `MD_segment 0.35 NVT_nose-hoover 800` Q

   Hint: For the mass Q of the thermostat degree of freedom, consider it as an oscillator whose proper frequency should be in the range of global vibrational frequencies (phonons) of the atomic system. Considering C_2H_6 has such modes (see next exercise) between 400 and 1500 cm^{-1} , derive an expression that links Q with its frequency f and estimate Q. Alternatively, and much more straightforwardly, the flag `MD_thermostat_units cm^-1` can be used to input the thermostat mass directly in cm^{-1}. (A control file `control.nose-hoover_programmed.in` with a reasonable value of Q is provided in `exercise_3/solution/scheduler`)

2. Andersen

   `MD_schedule`

   `MD_segment 0.15 NVT_andersen 300` ν

   `MD_segment 0.35 NVT_andersen 800` ν

   Hint: ν (the unit is ps^{-1}) should have a value such that every few (< 10) time steps there is a fair chance (20-40%) the system interacts with the bath.

3. Berendsen

   `MD_schedule`

   `MD_segment 0.15 NVT_berendsen 300` τ

   `MD_segment 0.35 NVT_berendsen 800` τ

   Hint: Use τ (the unit is ps) equal to few time steps.

4. Bussi-Donadio-Parrinello (BDP)

   `MD_schedule`

   `MD_segment 0.15 NVT_parrinello 300` τ

   `MD_segment 0.35 NVT_parrinello 800` τ

   Hint: Use τ (the unit is ps) equal to $\approx$ 20–50 time steps. Unless τ is too small with respect to the time step (1–5 time steps), the actual value of τ does not affect the performance of this thermostat; with a milder coupling, one observes a slower response to the imposed change of temperature, though.

- While waiting for the simulations to complete, you are challenged to demonstrate Eq. 1.
- Analyse the output by running the "aims_MD_eval.pl" script again and by plotting temperature (column 2) vs. time steps (column 1). Do you see the change in temperature after 0.15 ps? Is the molecule already equilibrated at the new temperature when the simulation ends?
- Compare the temperatures of "your" thermostat with the reference outputs for the other thermostats that you can find in `exercise_3/solution/scheduler`.
- Plot total energy (column 5) vs time step (column 1). Did you expect such a behavior? Now, for the Nose-Hoover and the BDP case, use the script "aims_NH-BDP.pl" (works analogously to "aims_MD_eval.pl") to extract the respective conserved quantity from the trajectory and plot it (sixth column) vs. the time steps (first column). What do you observe?

Time estimation: ∼40 min

Applications

Exercise 4: Harmonic vs. anharmonic vibrations - the C_2D_6 molecule.

Note: *Before reading all the introduction of this exercise, we invite you to read and set to run the "task 2" (anharmonic vibrations), because this run takes about 40 minutes, so that you will have time to read and understand everything while it runs. If you find yourself waiting yet for this run to finish, read the introduction of Exercise 5, and if you still have time left, we challenge you to derive equation 18 from equation 15.*

Vibrational spectroscopy is, nowadays, a very important tool for the characterization of molecules. Usually, the frequencies measured experimentally are compared to theoretical calculations in order to determine the geometry and electronic structure of the molecule. For this purpose, the most common approach is to relax the geometry of a molecule on the Born-Oppenheimer potential energy surface (PES) and then to perform a vibrational analysis in the the *rigid-rotor/harmonic-oscillator* approximation. There are a few problems in this approach: the inability to probe all representative conformations of the molecule and the inability to include anharmonic effects for particularly floppy vibrational modes. Furthermore, rotations along certain axis of symmetry in the molecule cannot be considered rigid rotors. This is important for the molecule studied here, which vibrates while it rotates and thus changes its moment of inertia.

It is possible to go beyond the harmonic approximation by "brute-force", but calculating the shape of the potential energy surface even for very few degrees of freedom is an amazingly demanding task.

One can overcome some of these drawbacks by performing a Molecular Dynamics simulation of the system in question. In the framework of Linear Response Theory, one can rewrite the Fermi Golden rule by means of the Fourier transform of the dipole moment time correlation function [8]:

$$I(\omega) = F(\omega) \int_{-\infty}^{\infty} dt\, \mathrm{e}^{i\omega t} \langle \vec{M}(t) \cdot \vec{M}(0) \rangle_{t_0} \tag{10}$$

In this formula, $I(\omega)$ is the intensity of the vibrations and $F(\omega)$ is a quantum corrector factor which must be multiplied with the classical line shape in order to reproduce the measured amplitudes [9, 10]. The angular brackets denote a statistical time average for the auto-correlation of the dipole moment of the molecule. Formula 10 will give all frequencies that are active in the IR range. Therefore, the whole IR spectrum of the molecule can be calculated within one MD run, since one can choose various t=0 to average the dipole auto-correlation over.

A similar relation can be found for the time average of the velocity auto correlation function:

$$\mathrm{VDOS}(\omega) = \sum_{i=1,N} \int_{-\infty}^{\infty} dt\, \mathrm{e}^{i\omega t} \langle \vec{v}_i(t) \cdot \vec{v}_i(0) \rangle_{t_0} \tag{11}$$

In this case, N is the number of atoms in the molecule and the quantities that are computed are all vibrational frequencies of the molecule, not only the ones active in the IR range (due to selection rules). The advantage is that this function allows to assign the frequencies to the displacements of the individual atoms [8].

For this exercise we will use our heavier molecule, C_2D_6, since it allows a larger time-step and hence a longer simulation (in time). The instructions follow:

1. **Harmonic vibrations**

   - The harmonic vibrational frequencies of the molecule are given in:

     `exercise_4/harmonic_vibs`.

     such that they can be compared to the anharmonic vibrations later on. They were obtained just like you did for NH_3 in the first practical section (Tutorial 1) of this workshop.

2. **Calculating the anharmonic vibrations**

- We will simulate eight (8) temperatures, namely: 100K, 200K, 300K, 400K, 500K, 600K, 700K, and 800K. Due to the time this simulation takes, each group will only inspect one temperature. A geometry file called `geometry.in."your temperature"`, which includes a thermalized geometry for the respective temperature is provided in the exercise's folder.
- Copy `geometry.in."your temperature"` to `geometry.in`
- Use the `control.in` file provided in this folder. There are some settings for the integration grids that are different, so that the simulation is computationally lighter.
- We will now run a 4ps MD simulation in the canonical ensemble with the BDP thermostat. As has been discussed above, *this specific* thermostat does not perturb strongly the dynamics of the system. The dynamic quantities are preserved, and thus the autocorrelation functions can be reliably calculated. Note that this is not the case for other thermostats. The "traditional" way of performing this simulation would be to first thermalize the molecule at the desired temperature and then start an MD run in the microcanonical ensemble, where the dynamical quantities would be computed. In the case of few degrees of freedom, though, the microcanonical sampling would be wrong, because the distribution of velocities is rather different from a Maxwell-Boltzmann one. With increasing number of degrees of freedom (more than ≈ 50), the distribution of velocities in the microcanonical and canonical ensembles become indistinguishable.

  Your MD block should read:

  ```
  MD_run 4.0 NVT_parrinello "your temperature" 0.1
  MD_time_step 0.002
  ```

  This run takes $\sim$35 minutes.
- **Dipole-dipole correlation function: IR spectrum.** For the analysis of this simulation, we will use the script `auto-correlate.py` in order to see the evolution of the auto correlation function and of the spectrum with the time of the run. Besides doing this analysis at the end of the simulation, you can do it after approximately 1ps, 2ps, and 3ps of simulation (or more often if you want). To check interactively how far the simulation is, you may write:

  ```
  grep "Updated atomic structure" "Output-File" | wc -l
  ```

  which gives the number of completed time steps.

  In order to run `auto-correlate.py`, you need to prepare an input file called '`control.autocorr.in`'. The flags are the following:

  - `path` *string*

    'string' is the name of the FHI-aims output
  - `choice` *string*

    'string' is the type of autocorrelation you want to calculate. Accepts either 'velocity' or 'dipole'
  - `sampling_interval` *integer*

    'integer' is an integer number that defines the sampling interval (each t=0) to calculate the autocorrelation. Recommended value is 1.
  - `cutoff_ratio` *float*

    'float' is a float number in the interval $[0, 1]$ that defines the ratio of the tail of the autocorrelation function you wish to leave out in order to make the fourier transform. Recommended value is 0.1.

– `broadening` *float*

'float' is a float number in units of wavenumbers (cm^{-1}) that defines the broadening of the Gaussian to be convoluted with the Fourier transform. Recommended value is 3.

An example of input file for this script is provided in the folder `exercise_4/anharmonic_vibs/`. Copy also the executable `"home_made_ft.x"` to the folder where you are performing this exercise. You can then run the script by typing:

```
python auto-correlate.py
```

Three files will be generated, namely:

– **autocorr.dat** contains 3 columns, the first being time in ps, the second being the autocorrelation function, and the third being the autocorrelation function times a window function that makes it go to zero on the edges. The window function is essential for reducing the noise in the Fourier transform.

– **raw_fourier_transform.dat** contains 2 columns, the first being wavenumbers in cm^{-1} and the second the intensities in arbitrary units

– **convoluted_fourier_transform.dat** contains 2 columns, the first being wavenumbers in cm^{-1} and the second the intensities in arbitrary units convoluted with a gaussian curve (width given in the input).

- Run first the python script with the '`dipole`' option in the `dipole-autocorr` folder. Remember to rename the outputs of the script so that they don't get overwritten. Visualize them in xmgrace and see how the autocorrelation function and the peaks of the Fourier transform evolve with time. What does the "long" time (average) correlation indicate? Did you expect the curve not to go to zero?
- **Watch the movie.** After extracting the xyz movie of the MD run, via the script `create_xyz_movie.pl`, you may also want to visualize the trajectory, e.g., with VMD. Checking the visualized trajectory, together with the plot of the total(-should-be-conserved)-energy, is generally an unmatched test to see whether something (and what, if the case) went wrong.
- Now that you have the anharmonic and the harmonic vibrational frequencies, try to plot them on top of one another to see the differences. The outputs are in arbitrary units, therefore you should scale one of the two spectra in order to compare them. Higher temperatures should show more anharmonic effects, while low temperatures should be closer to the harmonic result. Which peak shows more anharmonicity?
- **Velocity auto correlation function and projection onto normal modes.** After the whole FHI-aims run is finished, run also the python script with the '`velocity`' option, in a new folder that you can call "`velocity-autocorr`", for example.
- Copy the "`basic.xyz`" output from the harmonic analysis into the `velocity-autocorr` folder. Copy the script `project_vacf.pl` and `project_vacf_autocorrelate_one.py` script into the same folder. The full contents of your folder should be:

  – `control.autocorr.in` with the `velocity` option.

  – `basic.xyz` file containing the harmonic vibrational modes of CD_3-CD_3

  – `project_vacf.pl` script

  – `project_vacf_autocorrelate_one.py` script

  – `home_made_ft.x` executable

  – The output of you MD run.

  The `project_vacf.pl` projects the velocity autocorrelations onto the harmonic normal modes. The parameters of this script are:
  1) the xyz containing the normal modes (i.e. `basic.xyz`)
  2) the name of the MD output

  A typical call will look like:

```
project_vacf.pl basic.xyz "fhi-aims output"
```

You will get as output one intensity file for each *stable* frequency (i.e. without the 6 rigid body modes at 0 frequency) of the molecule. The outputs are named:
`convoluted_fourier_transform."fhi-aims-out".mode."id-mode"_"mode-freq".vacf`
where `"id-mode"` is just a counter that orders the modes.
Plot them all together, e.g. using
`xmgrace convoluted_fourier_transform."fhi-aims-out".mode.*`

Are the normal modes still vibrating with a localized frequency? Attention: at high temperatures, if your molecule changes conformation, this projection will break down.

Which mode(s) is/are present in the spectrum of the velocity auto correlation function but not in the spectrum of the dipole auto correlation function?

Can you assign the modes to a particular vibration, for instance by visualizing the eigenvectors related to each mode that were calculated for the harmonic case (visualize the harmonic modes in Molden or Jmol, like in Tutorial 1)? Attention: if your molecule changes conformation such an analysis will be meaningless.

Timing: 40min for the simulations , 1h total

Exercise 5: Free energy estimation via Thermodynamic Integration - the C_2D_6 molecule.

While the potential energy surface is shaped by the $3N$ coordinates (degrees of freedom) that describe a molecule, the *free* energy surface, which is the quantity actually measured in experiments, is a function of thermodynamical variables (temperature, pressure, entropy, volume, etc.). This is the quantity of fundamental interest for comparison with experiments and the one that rules all dynamics of the system. Obtaining free energies always involves averaging out degrees of freedom for specific thermodynamic conditions. In order to perform this task, it is necessary to define the partition function $Z(T)$ of the system of interest. With respect to the canonical partition function $Z(T)$, the Helmholtz free energy can be written as[11]:

$$F(T) = -k_B T \ln[Z(T)], \tag{12}$$

where k_B is the Boltzmann constant and T is the temperature.

In the rigid-rotor/harmonic-oscillator approximation, the vibrational partition function can be written as a product of the several vibrational energy levels, weighted according to the Bose-Einstein statistics to take into account the quantum nature of the nuclei:

$$Z_{vib}(T) = \prod_{i=1}^{3N-6} \frac{e^{-\frac{\hbar\omega_i}{2k_BT}}}{1 - e^{-\frac{\hbar\omega_i}{k_BT}}} \tag{13}$$

where ω_i are the normal modes of vibration of the molecule and the product runs over all modes except the ones corresponding to translations and rotations. Substituting 13 in 12, we get the following expression for the vibrational contributions to the harmonic free energy:

$$F_{vib}(T) = \sum_{i=1}^{3N-6} \left[\underbrace{\frac{\hbar\omega_i}{2}}_{\text{Zero Point Energy}} + k_B T \ln\left(1 - \exp^{-\frac{\hbar\omega_i}{k_BT}}\right) \right] \tag{14}$$

There are limitations to the use of the harmonic approximation. It is not expected to be valid at high temperatures, due to its harmonic nature - when the atoms start to explore higher regions of the potential well, it cannot be approximated as a parabola. [3]

A way to obtain an approximation for the anharmonic corrections to the free-energy will be the subject of this exercise.

From elementary thermodynamics, one can write for a system in the canonical (NVT) ensemble:

$$\frac{\partial(\beta F)}{\partial\beta} = \langle U \rangle_{NVT} \tag{15}$$

where β is $1/k_BT$, F is the Helmholtz free energy, and $\mathcal{U}$ is the potential energy of the system and the brackets $\langle \ldots \rangle_{NVT}$ still denote the canonical average.

We can then evaluate the free energy at a temperature T by integrating Eq. 15:

$$\frac{F(T)}{k_BT} = \frac{F_0(T_0)}{k_BT_0} + \int_{\beta_0}^{\beta} d\beta' \langle U \rangle_{\beta'} \tag{16}$$

where $F_0(T_0)$ is the free energy at a reference temperature, which is low enough in order to consider the system as harmonic.

In practice, the integral is evaluated numerically by sampling $\langle U \rangle_T$ for a discrete set of MD runs at thermostatted temperatures from T_0 to the desired T. Then, a numerical integration can be performed.

[3] The "real" anharmonic modes are more closely spaced than what is estimated by the harmonic picture.

The anharmonic contribution to the average internal energy, $\left\langle \Delta U^{anharm.} \right\rangle_\beta (T)$, can be written as:

$$\left\langle \Delta U^{anharm.} \right\rangle_\beta (T) = \left\langle \Delta U^{MD} \right\rangle_\beta (T) - U^{harm.}(T), \tag{17}$$

where $\left\langle \Delta U^{MD} \right\rangle_\beta (T)$ is taken from the MD simulations [4] and $U^{harm.}(T)$ is the harmonic reference internal energy. Since the MD simulations treat the nuclei as *classical*, the reference internal energy must be the harmonic classic energy, given by the equipartition theorem $U^{harm.}(T) = \frac{3N}{2} k_B T$. We here make an approximation, where we consider the anharmonic contributions in a classical framework, but add their effect to the "quantum" harmonic free energy of equation 14.

The expression used for computing the correction is:

$$F(T) = F_{vib}(T) + k_B T \int_{\beta_0}^{\beta} d\beta' \left\langle \Delta U^{anharm.} \right\rangle_\beta (\beta'), \tag{18}$$

where $F_{vib}(T)$ is given by equation 14.

- Compute the average internal energy $\left\langle \Delta U^{MD} \right\rangle_\beta (T)$ for the simulation you did in the previous exercise. A script to calculate this average from the FHI-aims output is available. The syntax is:

  ```
  get_average.pl "output_file" cutoff
  ```

  where `cutoff` should be given in order to specify how much of the beginning of the simulation you wish to disregard, i.e. `cutoff`=0.1 means you disregard the initial 10% of your simulation. In the present case, you have been provided in your `geometry.in` with a pre-thermalized structure. This means that you can use all the data for your average (i.e. `cutoff` = 0).

  The output of the script will give you the temperature T in the first column, the average internal energy U in the second column, and the standard deviation σ_U in the third column. Provide this data to the tutors. Detailed instructions will be given on the projected screen.

- After the data from all groups are processed by us, you will find in `/pub/tutorial5` a file with the data for all the temperatures given in the following format:

  $$\begin{array}{lll} T_1 & \left\langle \Delta U^{MD} \right\rangle_{\beta_1} & \sigma_U(T_1) \\ T_2 & \left\langle \Delta U^{MD} \right\rangle_{\beta_2} & \sigma_U(T_2) \\ & \vdots & \end{array}$$

  Name the file as you like, e.g. `average_u.dat`.

- Use the harmonic vibrations for C_2D_6 provided in the previous exercise and copy the file `basic.vib.out` to this folder.

- Now perform the integration by using another script. The syntax is the following:

  `integrate_fe.pl average_u.dat basic.vib.out` N_{atoms}

  where N_{atoms} is the number of atoms of your system, i.e. 6.

  The output of this script is the temperature in the first column, the total free energy $F(T)$ (Eq. 18) in the second column, and the harmonic (quantum) free energy $F_{vib}(T)$ in the third column.

- Plot $F(T)$ and $F_{vib}(T)$ as a function of temperature (in xmgrace, for example).

  What are the differences? Can you estimate up to which temperature the harmonic approximation should be a good approximation?

Timing: ≈30min total

[4] This quantity is the instantaneous electronic energy of the system along the MD run, minus the electronic energy of the optimized molecule. This resets the zero of the energy for commodity of plotting.

Exercise 6: Heat conduction of a one-dimensional chain - the CD_2 chain.

In *continuum mechanics*, the *heat diffusion equation*

$$\frac{\partial T(\vec{r},t)}{\partial t} + \alpha(T)\ \nabla^2_{\vec{r}}\, T(\vec{r},t) = 0 \tag{19}$$

describes how heat –and hence energy– is transported in macroscopic systems. The *heat diffusivity* $\alpha(T)$ that appears in this equation is an intrinsic, temperature dependent property of each material that quantifies how efficient (or inefficient) heat transport is. As discussed in the morning lecture, there are various techniques to assess $\alpha(T)$ from *first-principles simulations*; in the following, we will discuss one of these techniques, the so called *laser-flash method* [12]. In this approach, a longish supercell is specifically prepared in thermodynamic **non-equilibrium**, so that one small slice of the supercell is hotter than the rest of the system (see Fig. 3). By using *ab initio* MD, we will observe how equilibrium ($T = 25$ K) is re-established in the system and we will characterize this process by monitoring the temperature $T(t)$ in the central part of the previously cold cell (green slice in Fig. 3). To determine α, we will then fit this temperature profile $T(x,t)$ with the analytic solution of the *heat diffusion equation* (19), which is given by

$$T(x,t) = T_{cold} + \Delta T \sum_n (-1)^n \exp\left(-n^2\pi^2\alpha\, t/x^2\right)\ . \tag{20}$$

As you will see during this exercise, the finite size of our system leads to large temperature fluctuations that do not allow to accurately determine α from a single trajectory. We can however achieve convergence by averaging over multiple trajectories with different initial conditions.

Preparation of the supercell in non-equilibrium

The cardinal point in this type of simulations is the accurate preparation of the supercell in non-equilibrium. To achieve this goal, we exploit the fact that in the *harmonic* approximation (see Tutorial 1 – Vibrations) the positions $\vec{r}_i$ and the velocities $\vec{v}_i$ of the single atoms can be written as a superposition of *harmonic oscillations* [13]:

$$\vec{r}_i = \vec{r}_{0i} + \Delta\vec{r}_i \quad = \quad +\sum_s A_s(T)\ \frac{\cos\left(\Phi_s + \omega_s t\right)}{\sqrt{M_i}} \cdot \vec{e}_s \tag{21}$$

$$\vec{v}_i \quad = \quad -\sum_s A_s(T)\ \frac{\sin\left(\Phi_s + \omega_s t\right)}{\sqrt{M_i}} \cdot \omega_s \cdot \vec{e}_s\ . \tag{22}$$

Hereby, $\vec{r}_{0i}$ denotes the equilibrium position of atom i with mass M_i, $A_s(T)$ denotes the amplitude of mode s, Φ_s is its phase, and its eigen-frequency and -vector are given by ω_s and $\vec{e}_s$, respectively. It is straightforward to see that in this case the *average* kinetic energy of each mode is given by

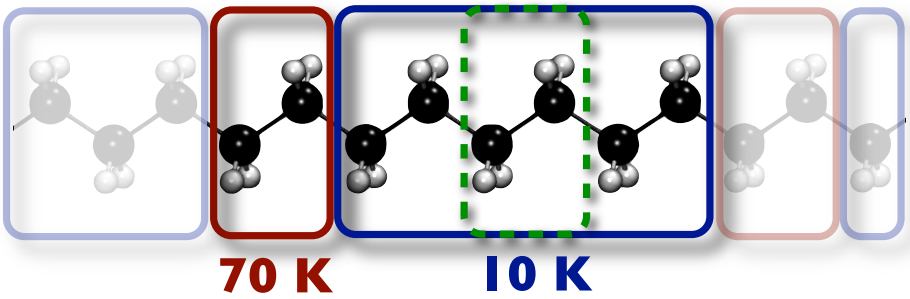

Figure 3: C_8D_{16} supercell for the linear one-dimensional CD_2 chain: Initially, the red slice of the supercell is prepared at 70 K, the remaining blue part of the supercell at 10 K. During the MD run, heat will diffuse through the system to re-establish equilibrium. We will characterize this process by monitoring the temperature in the central part of the previously cold cell (green).

$0.25\,\omega_s^2 A_s^2(T)$. On the basis of this relationship, we can easily generate non-equilibrium supercells by (a) assigning a random phase Φ_s to each mode and by (b) choosing random amplitudes $A_s^{\text{cold/hot}}$ that fulfill a Maxwell-Boltzmann distribution for the desired temperatures T_{cold} and T_{hot}.

A: Running the calculation

All the configuration files required for this exercise
(`geometry.in`, `control.in` and `eigenfreq-and-vectors.dat`)
are located in:
`/pub/tutorial5/exercise_6`

For the success of this exercise it is essential that you do not modify these files.

FHI-aims provides convenient routines to automatically generate non-equilibrium supercells according to the equations (21) and (22) discussed in the previous paragraph. For this purpose, we have to supply information about the eigenmodes of this system, which is done via the ASCII-file `eigenfreq-and-vectors.dat`. In the first section of this file, the equilibrium positions $\vec{r}_{0i}$ of the atoms are listed with a similar syntax to the one used in `geometry.in`:

```
# Equilibrium geometry
lattice_vector  50.00000   0.00000   0.000000
lattice_vector   0.00000  50.00000   0.000000
lattice_vector   0.00000   0.00000  10.101755
atom             0.00000   0.00000   0.000000  C   1
atom             0.67134   0.88230   0.000000  H   2
⋮                      ⋮         ⋮          ⋮  ⋮   ⋮
atom            48.50412  49.11762   8.839036  H  24
```

Additionally, an integer **index** i_{atom} is assigned to each atom in the last column. In the next part of the file, the *eigenfrequencies* of the system are given in ascending order:

```
# Eigenfrequencies in meV
n_acoustic                4
frequency     0.0000000000   1
frequency     0.0000000000   2
frequency     0.0000000000   3
frequency     0.0000000000   4
frequency     8.2962255208   5
⋮                          ⋮   ⋮
frequency   273.1260576050  71
frequency   273.5062375662  72
```

Again, each mode is assigned an **index** i_{freq}, whereby the tag `n_acoustic` allows to specify the number of *acoustic* modes.
Question: Why are there **4** acoustic modes in this system?

Last but not least, the components of the normalized eigenvectors are listed

`eigenvector` i_{freq} i_{atom} *x-component* *y-component* *z-component*

for all eigenmodes i_{freq} and all atoms i_{atom}. Please note that such a file can be easily generated in *phonon calculations*, a topic that is not covered by this tutorial. For the sake of simplicity, please use the provided `eigenfreq-and-vectors.dat` file and **do not modify** it.

To tell *FHI-aims* which part of the supercell shall be set up at which temperature, the tag

`MD_QH_init` $i_{\text{atom start}}$ $i_{\text{atom end}}$ T `eigenfreq-and-vectors.dat`

is used in `control.in`. Thereby, $i_{\text{atom start}}$ and $i_{\text{atom end}}$ are used to define a **range** of atoms that shall be set up at a specific temperature T (Kelvin) using the eigenmode data provided in the file `eigenfreq-and-vectors.dat`. In our case (see Fig. 3), we hence need to specify two segments, a small hot one at 70 K and a large cold one at 10 K:

```
MD_QH_init   1    6   70.0   eigenfreq-and-vectors.dat
MD_QH_init   7   24   10.0   eigenfreq-and-vectors.dat
```

Again, the provided `control.in` file already contains all relevant lines and should **not be modified**.

Please copy these three input files (`geometry.in`, `control.in` and `eigenfreq-and-vectors.dat`) from `/pub/tutorial5/exercise_6` into a directory of your choice. For the success of this exercise, it is crucial that you do **not** modify these files. Then, you should start *FHI-aims* in the usual fashion and redirect the ouput to your home directory:

```
mpirun -np 4 FHI-AIMS-BINARY > $HOME/MD.out
```

Please check that the calculation is indeed running on **4** CPUs and that your output is redirected to `$HOME/MD.out`. Furthermore, please terminate all other CPU-consuming applications (e.g. Firefox, Acrobat Reader,...) that are running on your computer so that *FHI-aims* can access all available resources.

Timing: 15 minutes for starting the simulation, 16h CPU time (overnight)

B: Evaluating the calculation(s)

All the scripts required for the evaluation of the calculations
(`extract_temperature_central_cell.pl` and `fitting.grace`)
can be found in:
`/pub/tutorial5/scripts`

Fitting a single trajectory

1. Copy the output of the *FHI-aims* run (`$HOME/MD.out`) to the directory in which you have started your MD run.
2. Copy the scripts `extract_temperature_central_cell.pl` and `fitting.grace` from `/pub/tutorial5/scripts`
   to the directory in which you have started your MD run.
3. Execute
   `./extract_temperature_central_cell.pl MD.out`
   to compute the temperature of the central cell as function of time. The script will save the computed temperatures in `MD.temperature.out`.
4. Execute
   `xmgrace -batch fitting.grace MD.temperature.out`
   to plot the temperature over time.
5. Use Xmgrace's fitting procedure *(Data → Transformations → Non-linear curve fitting)* to fit the plotted temperature to the analytic solution (20) of the *heat diffusion equation*, as shown in Fig. 4. Thereby, $A0$ corresponds to $\alpha\pi^2/x^2$, $A1$ to ΔT and $A2$ to T_{cold}. Are the resulting parameters consistent with the chosen initial conditions?

Averaging over multiple trajectories

1. For your convenience, we have already collected **all** trajectories that you have computed over night and we have stored them in your home directories `$HOME/results/tutorial5/exercise_6/`.

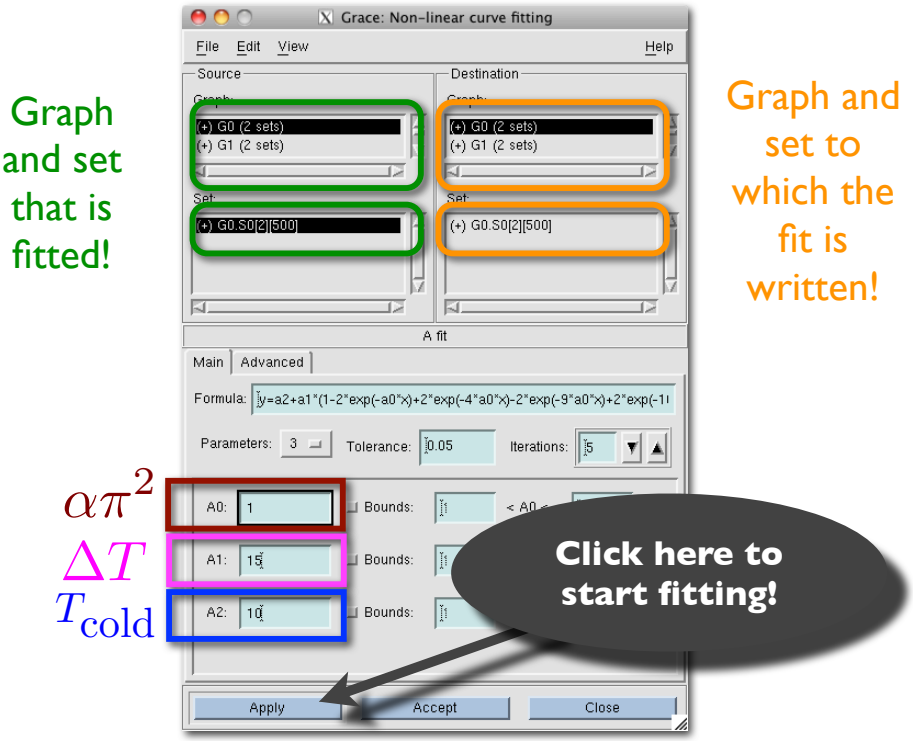

Figure 4: Non-linear curve fitting interface of Xmgrace, accessible through *(Data → Transformations → Non-linear curve fitting)*. Please make sure that you pick the right *source* graph and *source* set (i.e. data that is fitted) and the right *destination* graph and *destination* set (i.e. where the fit is written to). In the shown image, set $S0$ in graph $G0$ is fitted and the resulting function is written into a **new** set in graph $G0$.

2. Execute the provided script
   `./summarize_and_average.sh`
   to compute the temperature of the central cell as function of time for **all** trajectories. Furthermore, the script will average over multiple (1/5/10/20/40) runs and directly produce an *Xmgrace file* `AverageTemperatures.agr` containing this information (in 5 different graphs $G0 - G4$).
3. Execute `xmgrace -batch fitting.grace AverageTemperatures.agr` to open this file.
4. Again, use Xmgrace's fitting procedure *(Data → Transformations → Non-linear curve fitting)* to fit the plotted temperature to the analytic solution (20) of the *heat diffusion equation*, as shown in Fig. 4. Thereby, $A0$ corresponds to $\alpha\pi^2/x^2$, $A1$ to ΔT and $A2$ to T_{cold}.
5. How do the resulting parameters change for the different graphs? Can you judge the quality of the calculation from the resulting ΔT and T_{cold}?

Timing: 16h for the simulations, overnight

A. Plotting files in xmgrace

In order to plot files containing multiple Y columns, follow these steps:

1. Open xmgrace and click on Data → Import → ASCII (Figure 5).
2. Find the file that you want to plot and choose "Load as Block Data" in the dialogue box that will open (Figure 6).
3. Set X from column 1 and Y from whatever other column you want to plot in the new dialogue box that will open (Figure 7).
4. Press Apply and then press Close on all dialogues.

In order to plot bars, useful when you want to plot for example the harmonic frequencies as sticks, in step 3 above, instead of leaving "Set type XY", change for "Set type BAR" (Figure 8).

Properties of the plots can be changed by double clicking on the data or on the axis.

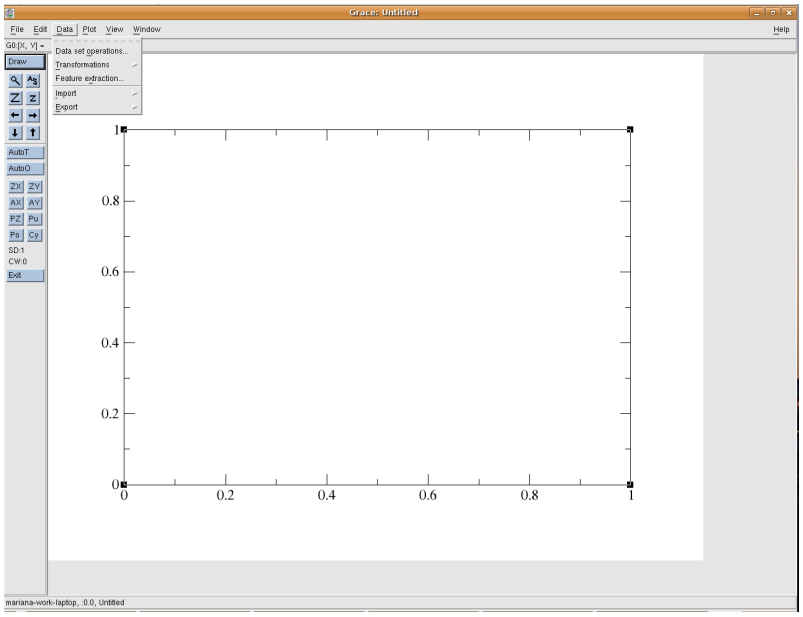

Figure 5: Step 1 - Open xmgrace and click on Data → Import → ASCII

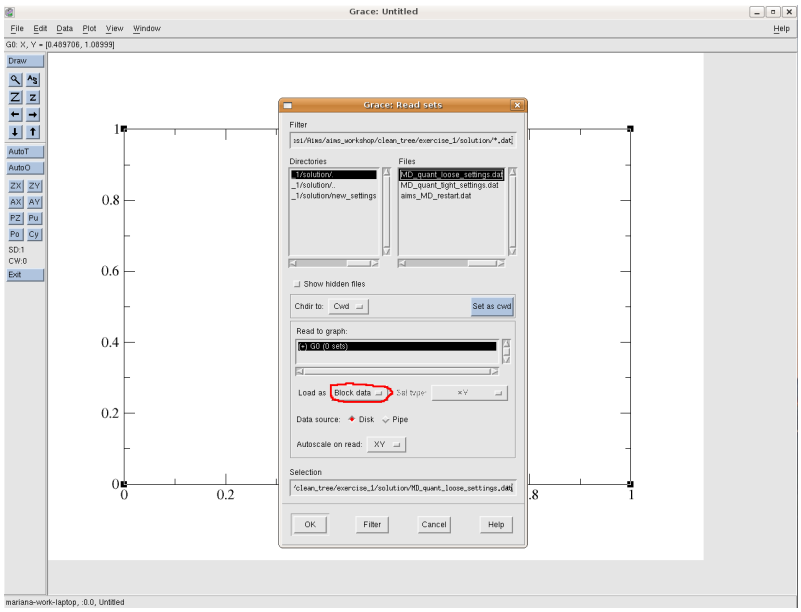

Figure 6: Step 2 - Find the file that you want to plot and choose “Load as Block Data” in the dialogue box that will open.

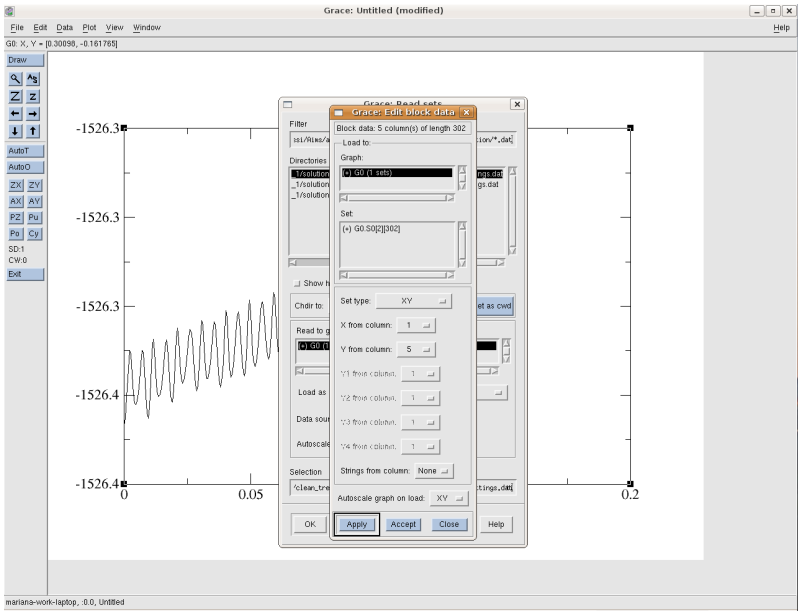

Figure 7: Step 3 - Set X from column 1 and Y from whatever other column you want to plot in the new dialogue box that will open.

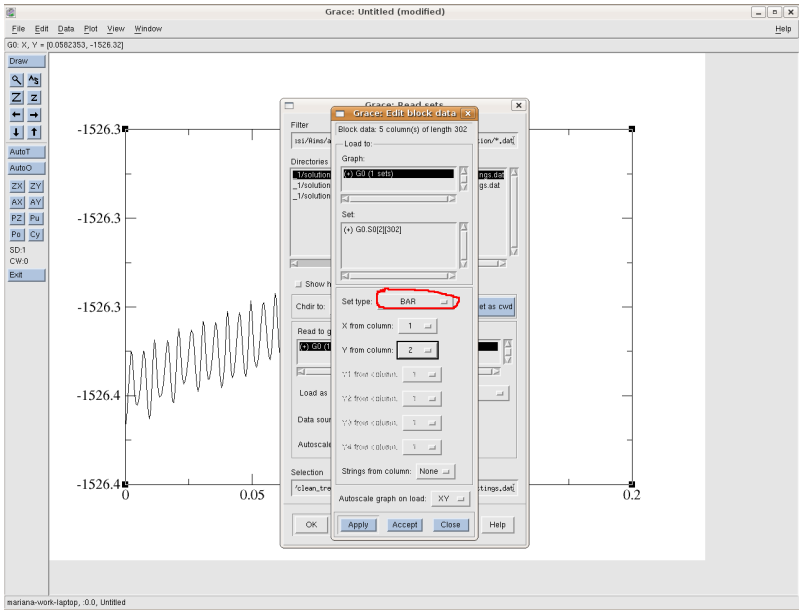

Figure 8: Useful to plot harmonic frequencies as sticks.

References

[1] J. Kolafa, Mol. Simul. **18**, 193 (1996)

[2] Kühne, Thomas D. *et al.*, Phys. Rev. Lett. **98**, 066401 (2007)

[3] D. Frenkel and B. Smit, *Understanding Molecular Simulation: from algorithms to applications*, second edition, Academic Press 2002.

[4] H. C. Andersen, J. Chem. Phys. **72**, 2384 (1980)

[5] G. Bussi, D. Donadio, and M. Parrinello, J. Chem. Phys. **126**, 014101 (2007)

[6] S. Nosé, J. Chem. Phys. **81**, 511 (1984). W.G. Hoover, Phys Rev. A **31**, 1695 (1985)

[7] H. J. C. Berendsen et al., J. Chem. Phys. **81**, 3684 (1980)

[8] M-P. Gaigeot, M. Martinez, and R. Vuilleumier, Mol. Phys. **105**, 2857 (2007)

[9] J. Borysow, M. Moraldi, and L. Frommhold, Molec. Phys. **56**, 913 (1985)

[10] R. Ramirez, T. Lopez-Ciudad, P. Kumar, and D. Marx, J. Chem. Phys. **121**, 3973 (2004)

[11] D. McQuarrie, *Statistical Mechanics*, University Science Books, 1st. ed., (2000)

[12] T. M. Gibbons and S. K. Estreicher, Phys. Rev. Lett. **102**, 255502 (2009).

[13] N. W. Ashcroft and N. D. Mermin, *Solid State Physics*, Cengage Learning, (1976).