Lab manual and instructor notes for: Revisiting the conformational isomerism of di-haloethanes; a hybrid computational and experimental laboratory for the undergraduate curriculum

B. I. Armstrong, M. Willans, E. L. Pearson, T. Becker, M. J. Hackett and P. Raiteri

February 2, 2023

Abstract

This laboratory manual outlines how experiments and simulations can be combined to approach a research project holistically. For the first time in your degree you will see quantum mechanics at work by performing highly accurate calculations of the electronic properties of molecular systems. You will perform a quantitative analysis of classical molecular dynamics simulations to extract the thermodynamic properties of molecular liquids, and you will collect Raman spectra of the liquids using a scanning probe microscope. You will use these three techniques to quantitatively determine the ratio of the *anti* and *gauche* isomers in 1,2-dichloroethane and 1,2-dibromoethane at a range of different temperatures. This will allow you to compute the free energy, enthalpy and entropy difference between the two conformations. After comparing the results obtained from the three different techniques you will better appreciate the limitations and benefits of each approach, and how theory and experiment can work synergistically in science.

Contents

How to read this document

Throughout this lab manual we will make use of boxes to highlight useful online resources, and describe commands and content of important files, particularly for the computational part. We will use a black box to show commands to be run in the terminal and their results. In order to differentiate between the two we will use the symbol `$` to indicate a command that has to be run in the terminal (the result of the command is without the leading `$`). For example these sequence of commands will show the path to the folder you are in and list the content of that folder:

```
$ pwd
/home/ubuntu/
$ ls
dce.input
```

where the outcome of the "pwd" command is "/home/ubuntu", and the outcome of the "ls" command is "dce.input".
Red boxes will be used to enclose the content of files:

```
# this file contains the coordinates of DCE #
...
```

and blue boxes will highlight URLs where you can find more information:

```
google.com
```

1 Introduction

This laboratory builds upon a wide rage of physical chemistry concepts and is most suited for an Honours degree and a small class. The main objective of this "advanced" physical chemistry laboratory is to determine the fraction of molecules in the *anti* or *gauche* conformations in liquid 1,2-dichloroethane (DCE) and 1,2-dibromoethane (DBE) as a function of temperature through the use of spectroscopic and computational techniques.

This laboratory will demonstrate how experiments and simulations are complementary to each other, and how they can be used in parallel to answer the seemingly simple question of determining the population of the *anti* and *gauche* conformations of DCE and DBE.
Specifically, in this laboratory you will:

1. Perform high-level quantum mechanical (QM) calculations to compute the Raman spectra of DCE and DBE in the gas phase;
2. Collect the Raman spectra of liquid DCE and DBE at different temperatures;
3. Run atomistic molecular dynamics (MD) simulations of liquid/vapour DCE and DBE at different temperatures to directly calculate the *anti* and *gauche* populations.

Due to the time constraints of this laboratory, the computational techniques will be mostly used like black boxes, but feel free to ask your lecturer and/or lab demonstrator for more details. A general knowledge of vibrational spectroscopy is however assumed.

As this is your last laboratory experience where you have to follow a pre-written script before doing original research for your Honours project, surveying the scientific literature for information and data to validate/interpret your results will be an integral, and substantial, part of the workload for this project.

Although the three "practical" parts of this laboratory can be performed in any order, the analysis of the experimental Raman spectra would benefit from having the results of the QM calculations. Therefore we will discuss the three parts starting from the QM calculations, then the MD simulation and the Raman spectroscopy last.

Why computer simulations?

`https://plato.stanford.edu/entries/simulations-science/#SimExp`

2 Miscellaneous online resources

Python for beginners:

```
https://wiki.python.org/moin/BeginnersGuide/NonProgrammers
```

Introduction to computer simulations (free pdf)

```
https://www.compadre.org/osp/items/detail.cfm?ID=7375
```

3 Background

The vibrational spectrum of a molecule is incredibly informative, enabling scientists to study molecular structure and concentration (or relative concentrations). As a chemistry undergraduate you will be very familiar with infrared spectroscopy, most often undertaken in the form of Fourier transform infrared (FTIR) spectroscopy. You therefore, should be familiar with the concept that an oscillating bond dipole can interact with electromagnetic radiation of the same frequency as the dipole oscillation, resulting in light absorption and excitation of the vibrational state to a higher energy level. There are however, additional techniques that can measure the vibrational spectrum of a molecule. One of the most common complementary techniques to FTIR is Raman spectroscopy. In contrast to infrared spectroscopy which measures light absorption (or reflection), Raman spectroscopy is an inelastic scattering technique. While infrared spectroscopy relies upon a net oscillating bond dipole during a molecular vibration, Raman spectroscopy requires a net change in polarizability during the vibration. These contrasting principles make infrared spectroscopy and Raman spectroscopy highly complementary. For example, bonds with strong dipoles are not easily polarised, are therefore inherently strong infrared absorbers but weak Raman scatterers. Bonds with weak dipoles are more easily polarised and are therefore, weak infrared absorbers and strong Raman scatterers. Likewise, if molecular symmetry renders a specific vibration inactive in the infrared spectrum (e.g. symmetric stretch of a linear molecule such as CO_2) it renders the same vibration Raman active (and vice versa, e.g., the asymmetric stretch of CO_2).

In Raman spectroscopy, the total scattering intensity of a given vibrational mode, I_i, which is defined as the integral of the area under the corresponding peak in the spectrum, is directly proportional to the concentration of the species

$$I_i = \sigma_i \; c_i \tag{1}$$

where σ_i is the scattering cross section of that particular vibrational mode.

In the case of a molecule that can assume multiple conformations, such as DCE and DBE, each conformer has its own specific set of vibrational frequencies, and therefore its own unique Raman spectrum. Therefore, the measured Raman spectrum of that compound is the weighted average of the spectra of the individual conformations, with the weights being the concentrations of the conformers. If we could then assign the individual peaks of the measured Raman spectrum to a specific mode and conformer we would be able to calculate the ratio between the populations of the conformers. Specifically to the case of DCE and DBE, where the molecules have only two stable conformers, *anti* and *gauche* , the ratio between the Raman intensities of two peaks originating from the two different conformers is proportional to the ratio between the number of molecules in those two states

$$\frac{I_i^a}{I_i^g} = \frac{\sigma_i^a}{\sigma_i^g}\frac{N_a}{N_g} \tag{2}$$

where $N = cV$ is the number of molecules in the *anti* or *gauche* conformation, and i indicates a given Raman active mode.

From the point of view of statistical thermodynamics, the number of molecules in each conformer is related to the free energy of that state through the Boltzmann factor

$$N_i \propto \exp\left[-\frac{G_i}{RT}\right] \tag{3}$$

hence the ratio between the two populations becomes

$$\frac{N_a}{N_g} = \exp\left[-\frac{\Delta G}{RT}\right] \tag{4}$$

where $\Delta G = G_a - G_g$ is the free energy difference between the *anti* and *gauche* conformations. Moreover, by performing a series of experiments at different temperatures, it is possible to break down the free

energy into its enthalpic and entropic terms

$$\ln\left[\frac{I_i^a \sigma_i^g}{I_i^g \sigma_i^a}\right] = \ln\left[\frac{N_a}{N_g}\right] = -\frac{\Delta G}{RT} = -\frac{\Delta H}{RT} + \frac{\Delta S}{T}. \tag{5}$$

In this laboratory experience you will:

- Compute the vibrational frequencies of the *anti* and *gauche* conformations of DCE and DBE using high-level QM methods
- Measure the Raman spectrum of liquid DCE and DBE as a function of temperature
- Calculate the total intensity of the appropriate Raman peaks to determine the ratio between the *anti* and *gauche* populations
- Perform molecular dynamics simulations to directly calculate N_a/N_g
- Determine the enthalpy and entropy difference between the *anti* and *gauche* conformations

4 Logistics of the laboratory

In this physical chemistry laboratory you will use a Raman spectrometer connected with a temperature stage and you will perform a variety of computer simulations, either locally or on a remote server. Although this physical chemistry laboratory is a self contained experience, there is also ample room to include more activities to expand both the experimental and computational sides. Some optional activities will be listed at the end, feel free to ask a supervisor if you are interested in expanding the scope of this laboratory. For example, it is possible to add more hands-on activities around writing programs/scripts to analyse the MD results, which entails processing relatively large data sets and it requires a pre-existing knowledge of some shell or Python programming (or another language).

For the experimental part of this laboratory you will use the micro-Raman spectrometer that is connected to the scanning probe microscope (SPM). SPM rooms often have limited size and access to the instrument at a given time, so it is recommended to perform the experiments in pairs with a round-robin scheduling. All the experiments should not take longer than one four-hour session.

For the computational part of this laboratory, you will need a computer, preferably a laptop, which you will use to either remotely connect to a virtual machine (VM) that your supervisor should have set up, or run locally on your own computer. If you don't have a laptop, please let the lecturer know as soon as possible so that different arrangements can be put in place. Running the simulations on your own computer will require you to download and install all the simulation packages which this can be a challenging task and so you should rely on your supervisor for help.

5 Creating the Virtual Machine

This section is here to demonstrate to the supervisor/sysadmin how to create a Unix/Linux-based VM that the students will connect to, and which pieces of software should be installed on it.

To create a VM capable of handling a group of students running intensive calculations simultaneously, one must use the resources from a supercomputing facility or a cloud computing facility. Your university may already have access to a supercomputing facility, and so there may already be a lecturer familiar with creating/requesting a VM from them. Otherwise, another option is to enlist the services of Amazon Web Services or Microsoft Azure and create a virtual private server through them.

```
https://lightsail.aws.amazon.com/ls/docs/en_us/articles/getting-started-with-amazon-lightsail
https://azure.microsoft.com/en-gb/products/virtual-machines/
```

Once connected, there are a handful of software packages that will need to be installed on the Unix/Linux system.

5.1 Computational chemistry software

In the computational part of this laboratory you will be using a suite of different programs, which are all free for academic use and can be easily installed on any Unix/Linux based system, and with some effort also on windows machines (outside the scope of this laboratory). The main programs that you will use are:

- ORCA `[orcaforum.kofo.mpg.de]`
  a recently developed code for quantum chemical calculations
- `LAMMPS [lammps.org]`
  a very versatile program for molecular dynamics simulations

- VMD [www.ks.uiuc.edu/Research/vmd/]
  a very efficient program for visualising atomistic MD trajectories
- Avogadro [avogadro.cc]
  an advanced molecule editor and visualizer (this will need to be installed locally on the students laptops, not on the VM)
- GPTA [github.com/praiteri/GPTA]
  a code developed at Curtin to help you perform basic pre-/post-processing tasks.

Additionally you would need to use some software to post-process the experimental Raman spectra, which could be the WITec suite or OPUS, or anything else equivalent.

If you are using a Windows computer for this laboratory, you should install `VMD` and `Avogadro` on you machine to visualise the molecular structures locally. Mac and Linux user should also consider installing `Avogadro`.

6 Accessing the Virtual Machine

Like all supercomputers in the world, the VM that you will be using for the computational part of this work will be using a Unix/Linux environment.

```
www.zdnet.com/article/linux-continues-to-rule-supercomputers/
itsfoss.com/linux-runs-top-supercomputers/
www.networkworld.com/article/3238034/how-did-linux-come-to-dominate-supercomputing.html
```

Specifically, the VM you will be using in this laboratory uses the Ubuntu operating system, which is one of the most commonly used (and free) Linux distributions available. It is therefore important that you know how to perform basic tasks using command lines instructions in the Linux environment. There are plenty of online tutorials where you can learn/refresh how to navigate your Linux environment, and at the end of this document there is a short cheat-sheet that contains a brief description of the most commonly used Linux commands. Although this is clearly a non-exhaustive list of videos that you can watch to learn the ropes of the main computational tools that you will be using in this laboratory, it's a good place to start. The first two are pre-requisites to starting the computational work, while the other are for visualising the simulation results.

```
linux/unix terminal [https://www.youtube.com/watch?v=2FiQSLdnBqA]
               vim [https://www.youtube.com/watch?v=IiwGbcd8S7I]
           gnuplot [https://www.youtube.com/watch?v=9QUtcfyBFhE]
               VMD [https://www.youtube.com/watch?v=LqKNzaHLfi0]
```

Although all the simulations will be run on the VM, which has a common environment, how you interact with the VM largely depends on the operating system that your computer has.

6.1 ...from Mac or Linux

If your laptop has either a Mac OSX or a Linux operating system, you are all set to go; there is only few small optional pieces of software that you may want to install, *e.g.* you can install `XQuartz` Although you may not have ever used it, the `terminal` is installed by default and you can just launch one. For the Mac users, you can launch it using spotlight, if you have a Linux laptop, you know where the terminal is...

In order to access the VM from you computer using a `terminal` you need to use encrypted secure shell connection (`ssh`). However, for security reasons, the machine is also protected by a *key/lock* authentication system and before being able to connect you need to generate a private/public key pair by running the `ssh-keygen` command in the terminal. It will make your life easier if you choose the default options...

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa.
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:g937OtqQi/ZdJGsscFAO+9X7POtC2Xkzf5z+AxiL3tI ubuntu@bignimbus2
```

```
The key's randomart image is:
+---[RSA 2048]----+
|      . .        |
|       =   .     |
|      o . . .    |
|       = o . .o .|
|      o S + *o =.|
|       o = B.o .*|
|        = B ..oo+|
|      .. X.E  ++.|
|     ...+.=o.  oB|
+----[SHA256]-----+
```

Then you would need to send the content of the file `~/.ssh/id\_rsa.pub` to your lab demonstrator, who will transfer your public key to the VM, hence granting you access to it. You can open the file with `vim` and send its content (one very long line) via email. We do not need to know your username and machine name, if that is embarrassing you can leave that out. For more information you can look at:

```
www.ssh.com/academy/ssh/keygen
```

Once that stage is done, you will be able to access the VM by simply using SSH, with a command like this:

```
$ ssh -XY your_username@XXX.YYY.ZZZ.QQQ
```

Your lab demonstrator will tell you what the actual IP address of the VM is, and assign you a username.

6.2 ...from Windows

Unfortunately life will not be so easy for you, particularly if you have a Microsoft Surface. Since Windows 10 operating system, it has become much easier to mimic a Linux partition on your laptop, without enabling dual boot and all the complications that come with it.

There are a handful of options to proceed:

- The option that has worked best for most students is to use `MobaXTerm` which will provide you with a Linux environment capable of connecting to the VM through `ssh` and allow `X11 Forwarding` so you can visualise the trajectories on the VM.

  ```
  https://mobaxterm.mobatek.net/download.html
  https://shorturl.at/guAGO
  ```

- A more comprehensive option is to install Windows Subsystem for Linux v2 (WSL2) which creates a complete Linux environment with all of the Unix functionality one could expect from an actual Linux distribution. This is a better option if you are interested in properly learning how to use Linux. Setting up WSL2 can be a bit of a challenge, the basic outline of the process is described below.
- The simplest way to connect to the VM is to use `Windows PowerShell` and access the VM in the same way as Mac OSX and Linux described in Section **??**. However, you won't have any graphical interface and have access only to terminal based applications. You will still have to generate an ssh-key as described in Section **??**.

If you have a windows desktop/laptop that you would like to use for the laboratory, you should use one of the three options described above. Alternatively, you could ask a supervisor for access to a local Mac desktop computer hosted in the department.

```
https://mobaxterm.mobatek.net/
docs.microsoft.com/en-us/windows/wsl/install-win10
```

This will give you access to all the basic functionalities to perform the computer laboratory but there might be limitations in using graphical applications remotely. Hence, you would probably have to install some visualisation software on your computer. Enabling remote visualisation on a Windows computer is possible but not trivial.

```
stackoverflow.com/questions/61110603/how-to-set-up-working-x11-forwarding-on-wsl2
```

Over the past 10 years years we had unpredictable results to get windows laptops properly set up, partially due to the operating system and partially because of the limited experience we have with the Windows operating system. To enable remote visualisation you would need to install additional programs such as `VcXsrv Windows X Server` or `Xming X Server for Windows`. Some help can be found on the internet but we never found a procedure that worked on every Windows machine.

6.3 ... using a browser

Alternatively, you can connect to the VM using a browser, through a JupyterHub. This has the significant advantage that you don't have to install anything on your computer to connect to the VM, and you would have direct access to jupyter notebooks to perform any data analysis using python, as well as to a terminal and an editor to modify the files and run the programs. However, all the visualisation has to be performed within the jupyter notebook using python packages, or on your local computer after you have downloaded the data. This applies to both visualising the atomic structure and simulations trajectories as well as plotting.

For this year's lab the VM can be reached using the jupyterHub from this address.

```
CompChemCurtin.webredirect.org
```

For security reasons, you will be granted access the VM only after you have registered using your student ID and email address.

After you have logged onto the VM, you can easily create a new Jupyter Notebook (Python 3), a new folder or file, or open a terminal and use all the Linux command line tools.

The jupyterHub has built-in file editor, which is quite similar to the Windows notepad, but the simulations have to be run in the terminal, see below.

The JupyterHub also has some basic functionalities for uploading/downloading files. Although you can upload multiple files simultaneously, you can download only single files, which is far from convenient.

In a unix operating system, you can easily create an archive of multiple files/folders using the `tar` command, *e.g.*

```
$ tar cvfz myArchiveFile.tgz myFolder/ myLocalFile1 myLocalFile2
```

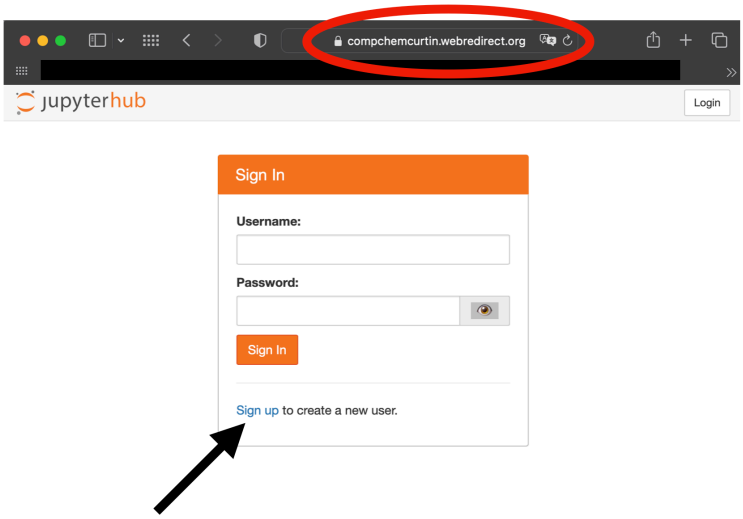

Figure 1: Access to the VM via a JupyterHub.

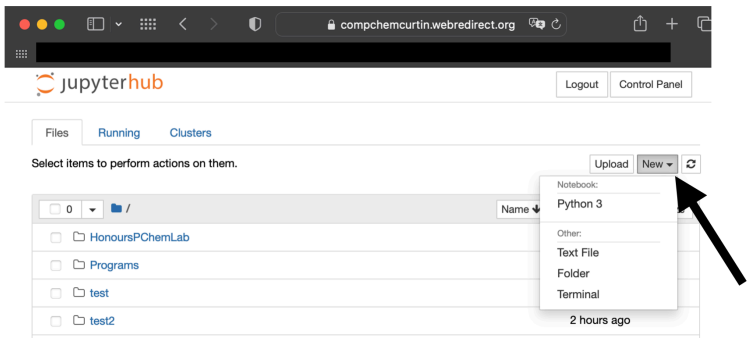

Figure 2: Using the Jupyter Hub to create new files or open a terminal.

7 First steps in the Unix world

Once you have opened a terminal, you can find extra help about a specific command, by running these the command line (or you can ask google).

- man -k [keyword]: Search a database for commands that involve the keyword.
- man [command]: Display the command help information.
- info [command]: Display the command help information in an alternate format.
- whatis [command]: Display a short blurb about the command.

Another important skill you need for this laboratory, is being able to to edit text files to prepare the simulations' inputs and analyse their outputs. There are many available file editors in the Unix operating system; `vim` is one of the most commonly used due to it's very light-weight nature, which allows for editing files remotely even with very low bandwidths. There are heaps of online tutorials and YouTube videos you can watch, but in your operating system there is already an embedded tutorial, which you can launch using:

```
$ vimtutor
```

This command will open a vim session where you can find the basic commands and instructions for using `vim`. *i.e.* you will use `vim` to learn `vim`. Alternatively you can browse some of these web-pages

```
www.javatpoint.com/linux-tutorial
www.tutorialspoint.com/vim/index.htm
vim.rtorr.com
```

8 Quantum mechanical simulations

www.faccts.de
www.orcasoftware.de/tutorials_orca/

In this laboratory you will be doing quantum mechanical calculations to assist you with the identification of the peaks in the Raman spectrum and to compute the thermodynamic stability of the *anti* and *gauche* conformers of the 1,2-dihaloethane molecules in the gas phase. As you probably have learnt during your undergraduate degrees, it is not possible to solve the Schrödinger equation analytically for any system that contains more than one electron. Many different methods have therefore been developed to produce approximate solutions of the Schrödinger equation for molecular systems. Although delving into the details of how approximate solutions of the Schrödinger equation can be obtained is beyond the scope of this laboratory, as a general (life) principle, the more accurate you want the solution the more computationally expensive the simulation becomes. In this work we will compute the electronic structure and energetics of molecules using a method called Density Functional Theory (DFT) and the sample inputs provided in this laboratory are a reasonable compromise between accuracy and efficiency, and there will be room for improvement, at the expense of longer times to obtain the results.

en.wikipedia.org/wiki/Density_functional_theory

The code that you will be using to perform the QM calculations is called `ORCA` and has been developed at the Max-Planck-Institut für Kohlenforschung in Mülheim an der Ruhr (Germany) that is strongly focused on chemical research in catalysis. The program is free for academic use and, if you want, it should be relatively easy to install on your personal (Linux/Mac) computer too.

The main steps of this part of the laboratory are:

1. Generate the atomic coordinates of DCE and DBE in the *anti* and *gauche* conformations
2. Relax the structures using ORCA to find the minimum energy state
3. Compute the vibrational frequencies of the molecules in the two conformations to estimate the free energy of the conformers
4. Compute the Raman spectrum of the the *anti* and *gauche* conformers
5. Repeat the calculations using an approximate representation of solvent effects (if time permits)

Before delving into the details of those tasks, it's worth mentioning that, like in most QM codes, the units of the energy in ORCA are Hartree (E_h)

$$1\ E_h = 2625.4\ \text{kJ/mol} = 27.211\ \text{eV}$$

8.1 Generating the input coordinates

The first task to do before starting any QM calculation is to generate the input coordinates for the system of interest. There are countless ways of generating or obtaining atomic coordinates for a given molecular systems, which include desktop (commercial) programs, web interfaces (mostly based on the JSME library), command line tools or pen&paper using some basic trigonometry.

en.wikipedia.org/wiki/Z-matrix_(chemistry)

Among the many available options we would recommend using Avogadro, a free cross-platform desktop program that can be installed from `[https://avogadro.cc]`. However, we could not get Avogadro

running remotely on the VM and it will have to be installed on your personal device. This has the advantage that it can directly generate inputs for, and read the output of, the most commonly used QM codes, including ORCA that you will use in this laboratory.

Alternatively, you can use one of the many JSME web interfaces, such as

```
biomodel.uah.es/en/DIY/JSME/draw.en.htm
```

or you can search the NIST database

```
cccbdb.nist.gov
```

for the coordinates of the molecule you are interested in, or you can even generate the atomic coordinates for your molecule and perform some basic QM calculations using this web interface.

```
chemcompute.org
```

In my personal opinion, one of the neatest ways of generating the atomic coordinates for a molecule on a Unix machine is to use SMILES (or SMARTS) to represent the structure of your molecule and OpenBabel to generate its coordinates.

```
en.wikipedia.org/wiki/Simplified_molecular-input_line-entry_system
openbabel.org/docs/dev/Command-line\_tools.
```

The examples below show you how to generate the coordinates for methane, dinitrogen, methyl isocyanate and nicotine

```
$ obabel -:"C" --gen3d -O ch4.xyz
$ obabel -:"N#N" --gen3d -O n2.xyz
$ obabel -:"CN=C=O" --gen3d -O mic.xyz
$ obabel -:"CN1CCC[C@H]1c2cccnc2" --gen3d -O nicotine.xyz
```

OpenBabel can then also be used to perform a search for all stable conformers of a molecule, *i.e.* to generate the coordinates of both the *anti* and *gauche* conformations.

```
$ obabel -ixyz mol.xyz -O conf.xyz --conformer --nconf 20 --systematic --writeconformers
```

8.2 Preparing an input file

```
sites.google.com/site/orcainputlibrary/general-input
```

ORCA's input files are pretty much free-format. Blank lines are allowed, and the input is usually not case-sensitive. One can create comment lines by adding the # symbol to a line. ORCA has both a Simple keyword syntax as well as a Block syntax. The Simple input is often the only input line needed (as well as the coordinates block) while some specific settings are only available using the Block Input. Note that settings specified in the Block input always takes precedence over the Simple input. In the Simple input syntax, keywords are added in any order to the line beginning with "!". Multiple "!" lines are also allowed. Advanced settings are often specified using the Block input for different modules. Blocks start

with"%nameofblock" and end with "end". For example, the input file below can be used to compute the energy of the methane molecules with the geometry specified in the `xyz` section, where the numbers `0 1` indicate the charge and spin multiplicity of the molecule.

```
! B3LYP def2-TZVP
* xyz 0 1
   C         0.03886       -0.50643        0.00000
   H         1.10886       -0.50643        0.00000
   H        -0.31780       -1.02686       -0.86420
   H        -0.31781        0.50221       -0.01862
   H        -0.31781       -0.99462        0.88281
*
```

If your ORCA input file is called `methane.inp`, the calculated properties will be written in a file called `methane\_property.txt`

8.3 Exchange and correlation functional and basis set

```
sites.google.com/site/orcainputlibrary/dft-calculations
sites.google.com/site/orcainputlibrary/dft-calculations/double-hybrid-dft
sites.google.com/site/orcainputlibrary/mp2-mp3
sites.google.com/site/orcainputlibrary/basis-sets
```

The single most important step of setting up a QM calculation is choosing an appropriate combination for the (exchange and correlation) functional and the basis set, which were indicated in the first line of the methane input file above. While going into the details of this is beyond the scope of this workshop, it is worth mentioning that this choice will highly affect the accuracy of the calculation, and the time it takes to finish. In general, the geometry of the molecules is reproduced fairly well even with cheap and approximate methods, however, more sophisticated and expensive methods are usually required for accurate energies and for computing complex properties, such as the RAMAN intensities. In the methane example, we chose `B3LYP` as the exchange and correlation functional and `def2-TZVP` as the basis set. Shown below is a list of methods for you to choose from

- `! MNDO`
  Not DFT yet, Semiempirical method - Modified Neglect of Diatomic Overlap
- `! AM1`
  Not DFT yet, Semiempirical method - Austin Model 1
- `! PM3`
  Not DFT yet, Semiempirical method - Parametric Method 3
- `! RI BP86 def2-TZVP def2/J`
  This GGA-DFT is typically the fastest useful DFT you can do
- `! B3LYP def2-TZVP`
  Hybrid GGA; an all time classic
- `! B3LYP RIJCOSX def2-TZVP def2/J`
  Same as above but with approximations to speed it up
- `! wB97X def2-TZVP def2/J RIJCOSX`
  Range-separated hybrid DFT calculations

- `! B3LYP D3BJ def2-TZVP`
  Dispersion corrected DFT
- `! RIJK RI-B2PLYP D3BJ def2-TZVP def2/JK def2-TZVP/C`
  Double hybrid DFT - fastest
- `! RIJCOSX RI-B2PLYP D3BJ def2-TZVP def2/J def2-TZVP/C`
  Double hybrid DFT - recommended
- `! RIJK RI-PWPB95 D3BJ def2-TZVP def2/JK def2-TZVP/C`
  Double hybrid DFT - most accurate
- `! RI-MP2 RIJCOSX aug-cc-pVTZ aug-cc-pVTZ/C def2/J`
  MP2 using the Resolution of Identity approximation (RI) for MP2 integrals
- `! SCS-MP3 cc-pVTZ`
  One step up from MP2 methods
- `! DLPNO-MP2 DEF2-QZVPP DEF2-QZVPP/C`
  Domain-based local pair natural orbital - MP2 - quadruple ζ - diffuse functions and polarisation
- `! DLPNO-CCSD DEF2-QZVPP DEF2-QZVPP/C`
  Domain-based local pair natural orbital - coupled clusters
- `! DLPNO-CCSD(T) EXTRAPOLATE(3) cc-pVQZ/C`
  Domain-based local pair natural orbital - coupled clusters - as good as it gets

Note that not all exchange correlations and DFT methods are compatible with the Raman calculations, ORCA will of course stop with an error if your choice cannot be used for that. In the examples above we have always used the triple-ζ basis set with polarisation function, `def2-TZVP`, but it is also possible to change that to a lower or higher quality basis set, `def2-SZVP` or `def2-QZVP`, respectively.

8.4 Geometry optimisation

`sites.google.com/site/orcainputlibrary/geometry-optimizations`

Typically the initial coordinates do not correspond to the minimum energy configuration of the molecule and the first step of any QM calculation is to do a geometry optimisation of the input structure. This is activated by the keyword `TightOpt` in the ORCA input file, which also sets the convergence criteria on the forces, energy and displacement to the "tight" settings. A faster, albeit less accurate geometry optimisation can be done using the `Opt` keyword. For the geometry optimisation it is advisable to choose a reliable exchange and correlation functional and a decent basis set, and tight convergence criteria.

```
# Geometry optimisation
! TightOpt TightSCF
```

8.5 Molecular orbitals

`sites.google.com/site/orcainputlibrary/printing-and-visualization`

One of the most common information that can be obtained from QM calculation is the shape and location of the molecular orbitals of a molecule. However, it is worth keeping in mind that the orbitals are just a mathematical construction used to project the electron density on a suitable frame of reference, and depending on the method used there could be spurious contributions [1]. Hence, although the electron

density, which is needed to obtain the approximate solution of the Schrödinger equation is computed at every cycle of the geometry optimisation, the molecular orbitals are not and they have to be requested in the input file.

```
# Compute the molecular orbitals for the final geometry
%output
  print[p_mos] true
  print[p_basis] 5
end
```

If you have installed `Avogadro` on your computer you can download `ORCA's` output file and open it with `Avogadro` (ensuring that you have installed the version of `Avogadro` capable of opening `ORCA` outputs).

8.5.1 Vibrational modes and thermochemistry

```
www.orcasoftware.de/tutorials_orca/prop/thermo.html
sites.google.com/site/orcainputlibrary/vibrational-frequencies-thermochemistry
```

Another very common and important analysis that can be done is to compute the vibrational spectrum of the molecule at the minimum energy geometry. This is useful because the frequencies will tell you whether the final structure is indeed an energy minimum and it can be used to calculate the free energy of the molecule. This can be achieved by adding the keyword `Freq` (or `NumFreq`) into the input file

```
! Freq
```

Please note that for some exchange and correlation functionals it is not possible to compute the vibrational frequencies analytically and an approximate numerical method has to be requested with the keyword `NemFreq`. `ORCA` will throw an error if the analytical frequencies are not available.

For a system wth N atoms `ORCA` will report $3N$ frequencies in the output files, however the first 6 are zero. This is because these "modes" correspond to the 3 translational and 3 rotational modes of the rigid molecule. The following $3N-6$ values should have positive frequencies if the molecule's geometry correspond to a (meta)stable state. If there are large and negative vibrational frequencies this indicates that the geometry optimisation has failed and/or corresponds to a transition state (saddle point), like this example here.

```
------------------------
VIBRATIONAL FREQUENCIES
------------------------
   0:         0.00 cm**-1
   1:         0.00 cm**-1
   2:         0.00 cm**-1
   3:         0.00 cm**-1
   4:         0.00 cm**-1
   5:         0.00 cm**-1
   6:       -70.85 cm**-1 ***imaginary mode***
   7:       -50.05 cm**-1 ***imaginary mode***
   8:        48.60 cm**-1
   9:       169.21 cm**-1
  10:       176.59 cm**-1
```

```
  11:       241.39 cm**-1
```

There is a large number of possibilities for why this can happen, but we should not have any problems to optimise the geometry of the simple molecules that we will be dealing with in this lab.

You can also visualise the vibrational frequencies with `Avogadro` by loading in the `ORCA` output file.

The vibrational frequencies also allows for calculating the free energy of the molecule at a given temperature using the quasi-harmonic approximation.

```
en.wikipedia.org/wiki/Quasi-harmonic_approximation
```

The temperature at which to compute the free energy can be selected by adding this block to the input file. The calculation of the vibrational frequencies can be expensive, particularly if they need to be computed numerically, and it is therefore advisable to do it only once and add the Block below in the same input you use for the geometry optimisation.

```
%freq
  Temp  298.15
end
```

If your input file was called geopt.inp, the output file `geopt_property.txt` will contain the main properties of your system, such as the electronic energy, the total energy, dipole moment, etc. One of the most important section of this file is the "THERMOCHEMISTRY_Energies". There you can see the electronic, translational, vibrational and rotational energies of your system. These quantities, together with the zero point energy, are then used to compute the enthalpy, entropy and free energy of your system at the temperature chosen in the input file.

8.6 Calculating of the RAMAN spectrum

```
www.orcasoftware.de/tutorials_orca/spec/IR.html
sites.google.com/site/orcainputlibrary/visualization-and-printing
```

The prediction of infrared (IR) and (non-resnonant) Raman spectra are nowadays a straightforward task in computational chemistry. Once the geometry optimisation is finished, you can perform a second run, starting from the final geometry (`geopt.xyz`) and compute the Raman spectrum by using this Block section in the input file. Note that for the Raman spectrum you must request the numerical frequencies `NumFreq`.

```
# Raman spectrum
! NumFreq NearIR TightSCF defgrid3
%elprop
  Polar 1
end
```

This requires preparing a new input file (*e.g.* `raman.inp`) that has those keywords and using the coordinates of the final structure that are also written at the end of the properties file. Make sure that there are no duplicate keywords before you run it, and that you remove the geometry optimisation keyword. For accurate calculations of the Raman intensities, you would require a fairly high level of theory, and we would recommend the use of a double hybrid functional and a triple-ζ basis set.

Once the calculations are finished, which could take up to a few hours, you can process the output and compute the Raman spectrum using this command

```
$ orca_mapspc raman.out raman -w50
```

where `raman.out` has to be replaced with the name of your output file. This will produce two files

```
raman.raman.stk
raman.out.raman.dat
```

which contain the list of the transitions with their intensities and a spectrum where the lines have been broadened to account for thermal effects, respectively. Changing the flag `-w50` will make the lines sharper or broader. You can quickly plot the `.dat` file using `gnuplot`, or you can fetch the file and plot it in `Excel`. Keep in mind that it is very difficult to compute the Raman intensity accurately, because they depend on the derivative of the molecular polarisability. Although in most cases there is a poor correlation between the experimental and theoretically predicted ones, the frequencies are generally well reproduced, and often a simple scaling of the values is enough to match the experimental values. Although scaling the frequencies may appear as a fudge, in general, once the level of theory has been chosen, the same scaling factor can be applied to different molecules and modes which can be done directly by ORCA

8.7 Adding solvent effects

```
www.orcasoftware.de/tutorials_orca/prop/CPCM.html
sites.google.com/site/orcainputlibrary/continuum-solvation-cpcmcosmo-smd
```

So far all the calculations have been performed in the gas phase (vacuum), and all solvent effects have been ignored. In general it is always difficult, and expensive, to simulate bulk liquids using QM methods, it is however possible to approximate the presence of a solvent using a continuum solvation model. In the example below we use the CPCM model for 1,2-dichloro-ethane.

```
! CPCM
%cpcm
  smd true
  SMDsolvent "1,2-DICHLOROETHANE"
end
```

If you want, and there is time, you can repeat the geometry optimisation and calculation of the Raman spectrum including the implicit solvent.

8.8 Identification of the transition state

Computational chemists are often interested in identifying the transition state of a chemical reaction. In fact, a transition state is a saddle point on the N-dimensional potential energy surface defined by the atomic coordinates, where N is the number of degrees of freedoms of the system. The shape of the potential energy surface is not known analytically, and can only be probed by computing the energy of the system for a given set of atomic coordinates. As such, locating a transition state is a problem that is conceptually similar to the energy minimisation. The main difference is that we want to locate a point

on the potential energy surface that is a minimum for each change of the coordinates except along the direction that connects reactants and products, where it is a maximum.

Several algorithms have been devised over the years that automate the search for a transition state. One of such algorithms is the Nudged Elastic Band (NEB), which was developed over 20 years ago by Jonson and Henckelman [2, 3, 4, 5, 6] The NEB takes its name from the fact that the algorithm builds a "band" of atomic configurations that connect the reactants and products states. This "band" represents one possible transition path for the reactions and it is usually initialised using a linear interpolation of the coordinates of reactants and products. This initial "band" is unlikely to represent the real reaction path and contain the actual transition state, so a series of optimisation steps have to be performed to optimise the band to find the best possible description of the transition path and of the transition state. During this optimisation stage the atomic configurations are maintained equally spaced on the potential energy surface by means of "elastic" springs, which make another part of the technique's name.

In order to demonstrate how this works, we will use this technique to calculate the transition state and activation energy for the rotoisomerisation between the *trans*and *gauche* states. The simplicity of this process makes the transition state apparent, and one could, in principle, compute the energy of a configuration where the Cl-C-C-Cl torsional angle is kept at 60 degrees and all the other degrees of freedom are relaxed. However, the NEB is a more general approach that can be applied to complicated reactions, where the location of the transition state is not obvious. The NEB is readily available in ORCA and can be easily performed after the relaxed structures of the *anti* and *gauche* conformers have been obtained by just adding a couple of lines in the input files. This is achieved by providing the coordinates of both the reactant (*anti*) and product (*gauche*) states and the number of intermediate states that we want to use to describe the reaction path.

```
! B3LYP def2-TZVP D3BJ TightSCF

# Nudged Elastic Band
! NEB-TS

# Input geometry
* xyzfile 0 1 react.xyz
%neb
  NEB_End_XYZFile "prod.xyz"
  Nimages 16
end
```

The calculation will produce a trajectory file that shows the minimum energy path between reactants and products (neb_MEP_trj.xyz) and the energy of those configurations (neb.final.interp), which can be visualised using Avogadro and plotted. Those files will also allow us to identify the transition state and the activation energy of the process.

8.9 Running ORCA

Although it is possible to run orca directly on the command line, for long runs it is better to use `tmux`, which allows your terminal session (and your simulations) to continue running even after you disconnect from the VM. By running this command:

```
$ tmux
```

you will launch a new `tmux` terminal, which basically works as a normal terminal, with the main difference being that you can detach your session from it, and reconnect later. To detach your session from the `tmux` terminal you can press: `ctrl+B D`. You can create as many `tmux` terminals as you like, and giving them meaningful names could be helpful.

```
$ tmux new -s tmux2
```

You can then check the list of active `tmux` terminals and connect to the one you want as in this example below

```
$ tmux ls
0: 1 windows (created Sun Sep  5 11:42:24 2021) [193x47]
tmux2: 1 windows (created Sun Sep  5 11:42:33 2021) [193x47]
$ tmux attach -t tmux2
```

If you want to delete a `tmux` terminal you can simply type `exit` while in connected to it. For more detailed information look *e.g.* at

```
www.howtogeek.com/671422/how-to-use-tmux-on-linux-and-why-its-better-than-screen/
```

Once inside a `tmux` terminal you can run ORCA with commands like this

```
$ orca geopt.inp > geopt.out &
$ orca raman.inp > raman.out &
```

Note the `&` at the end of the line, which makes your job run in the background leaving you access to the terminal. You can then check whether you have any processes running using either the `htop` or `jobs` commands.

```
$ jobs
[1]+  Running                 orca raman.inp > raman.out &
$ htop
```

`htop` is an interactive session that you can kill by pressing `q`. A typical geometry optimisation of DCE (or DBE) with a fairly accurate method should take approximately 45 minutes, although it mostly depends on how far the starting geometry is from the minimum energy structure, while the calculation of the Raman spectrum should take approximately 80 minutes on one CPU core. Another useful Linux command is `tail`, which allows you to print the last few lines of a file on the screen to check that the simulation is progressing.

```
$ tail -n 12 raman.out

Timings for individual modules:

Sum of individual times         ...     5379.208 sec (=  89.653 min)
GTO integral calculation        ...        0.242 sec (=   0.004 min)   0.0 %
SCF iterations                  ...       45.494 sec (=   0.758 min)   0.8 %
Solution of CP-SCF eqns.        ...        8.199 sec (=   0.137 min)   0.2 %
```

```
MP2 module                          ...       76.720 sec (=   1.279 min)   1.4 %
Numerical frequency calculation ...     5248.498 sec (=  87.475 min)  97.6 %
XTB module                          ...        0.055 sec (=   0.001 min)   0.0 %
                              ****ORCA TERMINATED NORMALLY****
TOTAL RUN TIME: 0 days 1 hours 28 minutes 52 seconds 305 msec
```

In the example above the simulation had already finished.

Alternatively, you can run the simulations in the "normal" terminal pre-pending the `nohup` command and appending the `&` symbol. `nohup` will prevent the program to hang-up when the terminal is closed and the will put the job in the background so that you will still be able to use the terminal after starting the job.

9 Molecular dynamics simulations

```
en.wikipedia.org/wiki/Ergodic_hypothesis
en.wikipedia.org/wiki/Molecular_dynamics
nznano.blogspot.com/2017/11/molecular-dynamics-in-python.html
```

In this part of the project, you will be using classical molecular dynamics simulations to directly calculate the fraction of DCE and DBE molecules in the *anti* and *gauche* conformations. You will also calculate the free energy profile for the isomerisation process. In these simulations, you will use an empirical force field to describe the intra- and inter-molecular interactions between the atoms, which is a huge simplification from using an approximate solution of the Schrödinger equation to compute energy and forces. In classical molecular dynamics, the atoms are treated as point particles that carry a mass and a partial charge, the covalent interactions are described with simple springs and the non-bonded interactions are approximated as the sum of the Coulomb and van der Waals potentials. See *e.g.* the wikipedia page about force fields in chemistry:

```
en.wikipedia.org/wiki/Force_field_(chemistry)
```

There is a large zoo of force fields for small organic molecules and in this work you will be using a modified version of the Generalised Amber Force Field (GAFF) [7].

Similarly there is a large number of packages (free or commercial) that can be used to run Molecular Dynamics simulations. This laboratory has been tested using `LAMMPS`, which is one of the most common packages for this type of simulations, and `OpenMM`, which is very efficient if the VM has access to a GPU.

```
lammps.org
openmm.org
```

9.1 What is molecular dynamics?

Molecular dynamics (MD) is a technique where atomistic trajectories are generated using an iterative numerical solution of the Newton's equations of motions for a system of interacting particles. Usually, this is achieved using the so-called velocity Verlet algorithm, where the time evolution of the acceleration, velocity and position of each particle i are written as

$$\vec{a}_i(t) = -\nabla_i U = -\frac{\partial U}{\partial \vec{x}_i} \tag{6}$$

$$\vec{v}_i(t+\Delta t) = \vec{v}_i(t) + \frac{\vec{a}_i(t) + \vec{a}_i(t+\Delta t)}{2}\Delta t \tag{7}$$

$$\vec{x}_i(t+\Delta t) = \vec{x}_i(t) + \vec{v}_i(t)\,\Delta t + \frac{1}{2}\,\vec{a}_i(t)\Delta t^2 \tag{8}$$

$$\tag{9}$$

The timestep, Δt, is the time interval between two successive evaluations of the forces and energies. In every MD simulation, the timestep has to be short enough to allow for an accurate description of the fastest motion in the system, which normally is the O-H stretching mode that has a typical frequency of around 3,000 cm^{-1}. By multiplying that for the speed of light (c=29.979 cm/ns), we can compute the frequency in time units

$$3,000 \text{ cm}^{-1} \times c \approx 90 \text{ THz}.$$

Hence, the period of the the O-H stretching is around 10^{-14} s and if we want to describe that motion accurately we need to use a timestep that is at least 10 times smaller than the oscillation period, *i.e.* $\Delta t \approx 1$ fs.

To put this into perspective, if `LAMMPS` were to take 5 ms to do one MD cycle and the system moves forward in time by 10^{-15} s, in order to simulate a chemical/physical phenomenon that occurs on a 1 ns timescale it will take us about 1.5 hr. If the process we are interested in occurs on a millisecond time scale, it will take us 160 years to simulate that. This is the fundamental reason as to why protein folding and biological processes are so hard to tackle with computer simulations. Of course, the situation is not always so grim and there are algorithms (enhanced sampling methods) that allow us to access extended time scales, but they require more complicated setups, and going into the details of those techniques is beyond the scope of the present laboratory.

The interaction energy and forces can be obtained in many different ways, but MD is most commonly used in conjunction with empirical force fields, where the energy, that is a function of all the atoms' positions, is typically decomposed into bonded and non-bonded terms:

$$U = U_{bonded} + U_{non-bonded} \tag{10}$$

where the non-bonded interactions comprise the van der Waals and Coulomb interactions. The bonded interactions instead describe the intramolecular forces, *i.e.* the vibrational modes of the molecules, where all the bond stretching, angle bending and torsion rotations are approximated with harmonic springs:

$$\begin{aligned} U &= \sum_{i\in\text{bonds}} k_{bi}(b_i - b_i^0)^2 + \sum_{i\in\text{angles}} k_{ai}(\theta_i - \theta_i^0)^2 && (11)\\ &+ \sum_{i\in\text{torsions}} \frac{1}{2}V_i[1+\cos(n\omega_i - \gamma_i)] && (12)\\ &+ \sum_{j=1}^{N-1}\sum_{i=j+1}^{N} 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] && (13)\\ &+ \sum_{j=1}^{N-1}\sum_{i=j+1}^{N} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} && (14) \end{aligned}$$

where the first two lines describe the intramolecular interactions and the last two the intermolecular (non-bonded) interactions.

The reason why we can use MD to compute the thermodynamical properties of a system is the ergodic hypothesis, which states that the time average of any property, A, computed from an (infinitely long) MD trajectory of an isolated system corresponds the the macroscopic thermodynamic average of that property in the microcanonical ensemble (isolated system):

$$\langle A\rangle = \lim_{T\to\infty} \frac{1}{T}\int_0^T A(t)\mathrm{d}t \approx \frac{1}{N}\sum_{i=1,N} A_i \tag{15}$$

where $\langle\rangle$ indicates the macroscopic value of a thermodynamic observable and N is the number of frames in the molecular dynamics trajectory. Although ergodicity has been proved valid only for the simplest of systems, it has also never been proved wrong for any system, and non-ergodic behaviour has only been observed due to the use of unsuitable algorithms for the problems at hand. Equipped with the ergodic principle, we can then use statistical mechanics to compute any property from a MD simulation and compare it with experimental numbers. An important consequence of the ergodic assumption is that the probability that a microstate is visited during a MD trajectory is proportional to the Boltzmann factor for that microstate

$$P_i \propto e^{-\beta G_i} \tag{16}$$

where G_i is the free energy of the system in that microstate and $\beta = 1/k_BT$. Note that because the argument of the exponent must not have units, if the energy is expressed *per mole* k_B should be replaced with the ideal gas constant, $R = N_a k_B$. Equation 16 is particularly important because when it is reversed, it shows that we can compute the free energy difference between two states as the natural logarithm of the ratio between the probabilities that those two states are visited

$$\Delta G_{12} = -k_B T \ln(P_1/P_2). \tag{17}$$

This is indeed what the ultimate goal of this part of the laboratory is; we will run long (enough) MD simulations at different temperatures and use the output trajectories to compute the fraction of molecules in the *anti* and *gauche* conformations, and therefore determine the free energy difference between those two states.

9.2 Generating the input coordinates

Analogously to the QM calculations, the first step to set up an MD simulation is to generate the initial coordinates of the system. However, because we are now interested in a liquid "sample" we need to generate the coordinates of thousands of atoms, which requires the aid of a piece of software. There are many different ways of going about this. We could start from the crystal structure of the compound, which can be found on the Cambridge Crystallography database, and melt it, or we could randomly place a number of molecules in a "box" and then equilibrate the system at the conditions we are interested in. Both approaches have pros and cons, and they can both be useful in different circumstances. Because we already have the XYZ coordinates of the compounds we are interested in, the second approach is probably more efficient. Conceptually, what we want to do is choose a volume for our simulation box and fill that space with a random arrangement of DCE/DBE molecules, which is something that can be easily done with the aid of a computer program. For example you can use GPTA, a program that was developed at Curtin University over the last few years.

```
github.com/praiteri/GPTA/tree/main/doc
```

The command below performs three actions; it creates a cubic "box" of size `XXX`Å for your simulations, it creates `YYY` molecules of the species contained in the file `f1.xyz` and `ZZZ` molecules of the species contained in the file `f2.xyz`, and then it writes the coordinates of this new system in a file called `coord.pdb`, which can be directly visualised with VMD.

```
$ gpta.x --add box XXX --add solvent +f f1.xyz,f2.xyz +n YYY,ZZZ +rmin 3 --top --o coord.pdb
$ vmd coord.pdb
```

Your first task is therefore to find the molecular density of DCE and DBE and create a suitable box to start your simulations. There are a few things to keep in mind when choosing the values for `XXX`, `YYY` and `ZZZ` are

- The more atoms in your system the slower the simulations will be
- The minimum box size that can be simulated within the constraints of the force field is 20Å
- The random packing of molecules in the box is less effective than the "real" molecular packing, hence the volume will shrink during equilibration
- If you try to cram too many atoms in a the "box" the code will not be able to complete the action and eventually will stop

Please also note that if the initial density if too high or too low, the simulations may be (or quickly become) unstable. This is related to the algorithms that are used to adjust the cell size and the atoms' velocities to control the pressure and temperature in the system. If this happens there is no general rule to solve the problem, and it is often best to start from a different configuration that has a density closer to the experimental value. A reasonable starting configuration would have density slightly lower the experimental one, *e.g.* no more than 50% lower, which would account for some inaccuracy of the computational model used.

Optionally, you could also use the same procedure to generate input configurations for DCE and DBE in the gas phase. In this case you would put a handful of molecules in a fairly large simulation box. A simple way to estimate the volume of the DCE and DBE vapour by using the ideal gas low and then add some to account for the fact that DCE and DBE are not ideal gasses. The results of any MD simulations for gas DCE/DBE will be more directly comparable to the QM calculations, while those for the liquid phases are a better approximations for the experimental results.

Once you have generate a set of coordinates for the simulation of the liquids, you can either run the simulations with openMM (faster if your VM has a GPU) or using `LAMMPS` (enables MPI parallelisation).

```
lammps.org
openmm.org
```

9.3 Running openMM

Strictly speaking, `openMM` is not a program, but is more akin to a "library", and in order to run a molecular dynamics simulation you would have to write a program calling the appropriate functions to produce a Molecular Dynamic code. This is not a trivial undertaking particularly if you are not familiar with python and molecular dynamics. Therefore, in order to overcome this barrier, a program has been already prepared for you and you would just have to set the simulation parameters on the command line, available in a custom python library, which was called dynamicEntropy. Using a terminal in the VM, you

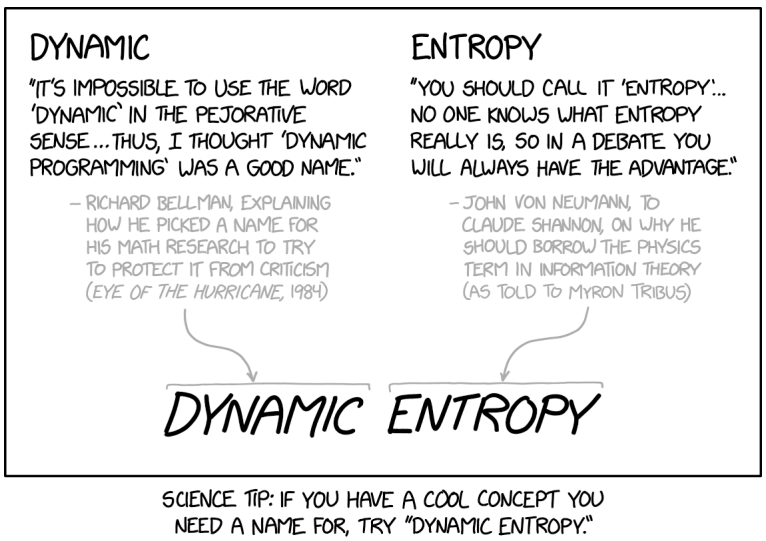

Figure 3: Dynamic Entropy, from https://xkcd.com/2318/

can get a breif help about the options for the command

```
$ python -m dynamicEntropy -h
usage: __main__.py [-h] [--keys | --no-keys | --info | --no-info] [--energy | --no-energy]
                   [-q | --quiet | --no-quiet] [--i INPUTFILE]
                   [--openMM KEY=VALUE [KEY=VALUE ...]] [--ff KEY=VALUE [KEY=VALUE ...]]
                   [--rest KEY=VALUE [KEY=VALUE ...]] [--minimise [KEY=VALUE ...]]
                   [--md [KEY=VALUE ...]] [--mtd [KEY=VALUE ...]]
                   [--fep [KEY=VALUE ...]] [--osmotic [KEY=VALUE ...]]

optional arguments:
  -h, --help            show this help message and exit
  --keys, --no-keys, --info, --no-info
                        write all the user available options

  --energy, --no-energy
                        write all energy contributions

  -q, --quiet, --no-quiet
                        no screen output

  --i INPUTFILE         read commands from JSON file

  --openMM KEY=VALUE [KEY=VALUE ...], --openmm KEY=VALUE [KEY=VALUE ...]
                        set the hardware parameters

  --ff KEY=VALUE [KEY=VALUE ...]
                        set the force field parameters

  --minimise [KEY=VALUE ...]
                        perform energy minimisation

  --md [KEY=VALUE ...]  perform Molecular dynamics

...
```

or an extended list of all the variables ("KEYS") that you can set

```
$ python -m dynamicEntropy --keys
Settings for --openMM
   text = #--- openMM parameters ----------------#
   Platform = CUDA
   Precision = mixed
   DeviceIndex = 0
Settings for --force field
   text = #--- force field parameters  -----------#
   force field = newFF.xml
   amoeba = False
   coordinates = coord.pdb
   nonbondedMethod = PME
   ewaldErrorTolerance = 1e-05
   nonbondedCutoff = 0.9
```

```
   polarization = mutual
   mutualInducedTargetEpsilon = 1e-05
   useDispersionCorrection = False
   constraints = None
   rigidWater = False
Settings for --minimise
   text = #--- Energy minimisation parameters ---#
   active = False
   tolerance = 10
   maxIterations = 0
   output = geopt.pdb
Settings for --md
   text = #--- Molecular dynamics parameters ----#
   ensemble = NVT
   integrator = LANG
   timestep = 0.001
   numberOfSteps = 10000
   simulationTime = -1
   temperature = 300
   thermostatParameter = 0.1
   pressure = 1
   barostat = ISO
   barostatUpdate = 25
   output = output.0.out
   trajectory = trajectory.0.dcd
   reportInterval = 1000
   checkpointInterval = 1000000
   restartFrom = None
   restartFile = restart.0.xml
...
```

Most of the options available won't be used in this lab, but they allow for expanding the computational part of the laboratory by including different systems or understanding the technical aspects of running MD simulations. For this laboratory you will have to change the name of the files containing the coordinates and force field, the ensemble, temperature and length of the simulations. For example

```
$ python -m dynamicEntropy --ff coordinates=coord.pdb force field=newFF.xml \
  --md ensemble=NPT temperature=298 numberOfSteps=1000000
```

will run a simulation in the constant temperature and pressure ensemble (NPT) for 100 ps (numberOfSteps=100000) at 298 K starting from the atomic coordinates in the files `coord.pdb` using the atomic interactions listed in `newFF.xml`. The code writes on the screen the value of all the input parameters that are relevant for the current simulation and some info about the system, before starting the actual MD simulation.

```
...
#--- System details -------------#
  Periodic cell vector a (nm) =  5.0044  0.0000  0.0000
  Periodic cell vector b (nm) =  0.0000  5.0044  0.0000
  Periodic cell vector c (nm) =  0.0000  0.0000  5.0044
```

```
  Number of atoms = 6800
  Number of residue types in the simulation =  1
    Number of residues of type M1 (DCE) = 850
      Atom C1  - FF charge -0.0581000 type  (Cc/C1)
      Atom Cl1 - FF charge -0.2041000 type  (Cl/Cl1)
      Atom H1  - FF charge +0.1311000 type  (Hc/H1)
      Atom H2  - FF charge +0.1311000 type  (Hc/H2)
      Atom C2  - FF charge -0.0581000 type  (Cc/C2)
      Atom Cl2 - FF charge -0.2041000 type  (Cl/Cl2)
      Atom H3  - FF charge +0.1311000 type  (Hc/H3)
      Atom H4  - FF charge +0.1311000 type  (Hc/H4)
  Total charge = 0.000
#--- Using isotropic barostat ---#
  MonteCarloPressure             =                           1.0
  MonteCarloTemperature          =                         298.0
  PME parameters = [3.654825709716916, 125, 125, 125]
  Ewald Tolerance = 1e-05
#--- Debug energies -------------#
  Total energy                   =     -23452.203443189566 kJ/mol
  HarmonicBondForce              =      21.847927316176715 kJ/mol
  HarmonicAngleForce             =      460.51180305791974 kJ/mol
  PeriodicTorsionForce           =       5176.702461225909 kJ/mol
  NonbondedForce                 =     -29111.265638930236 kJ/mol
  CMMotionRemover                =                     0.0 kJ/mol
  MonteCarloBarostat             =                     0.0 kJ/mol
#--------------------------------#
#--- Molecular dynamics ---------#
#"Progress (%)" "Potential Energy (kJ/mole)" "Temperature (K)" "Box Volume (nm^3)"
"Speed (ns/day)" "Elapsed Time (s)" "Time Remaining"
2.0% -22777.605622246145 69.52743007795135 104.32217580425323 0 0.00010180473327636719 --
4.0% -19915.550241619287 118.04774008625598 101.73192768024046 144 1.2008285522460938 0:57
...
98.0% -849.4996708636536 299.11130905167863 118.47743141384466 141 58.869502544403076 0:01
100.0% -493.6633357230967 297.7925936311518 117.9006092382699 141 60.09319829940796 0:00
```

9.4 Preparing the input files for LAMMPS

If you are going to use `LAMMPS` for the molecular dynamics simulations, once you have generated a "box" for your liquid, you then have to prepare more two input files for `LAMMPS`, one with the coordinates in an appropriate format and one with the settings of the MD simulations. To run the simulation you will also need another file, which contains the force field information, but it does not have to be modified.

Although you have just created a simulation "box", it is not in the right format for `LAMMPS`. Indeed `LAMMPS` will also need to know the list of all covalent bonds, angles and torsional terms in the system. This is a tedious task, which is again best suited for a computer than a human. GPTA will again help you in this, unless you want to try writing your own program/script for this. If your molecules are not too close to each other, this command will create a `LAMMPS` input file for you

```
$ gpta.x --i coord.pdb --top --o coord.lmp +f force field.lmp
```

Please note that you must have the force field file in the same folder where you run this command, because

GPTA needs to read it to know what force field terms are present. GPTA has a pretty verbose screen output, which you may want to read to make sure the code has done what you wanted/expected, and that it has found the correct number of atoms and molecules in the system. If GPTA is not detecting the write molecules, you will need to either build your system with your molecules spaced out more, or use the +def flag after the –top command, the details of which are explained in the GPTA manual.

Finally you need to edit the `lammps.inp` file to set up the simulations. In that file you can choose the temperature, pressure and length of the simulation, together with many other parameters that are not too important for this laboratory. The `LAMMPS` input file is a sort of simple program and all commands are executed in the order they appear; it must therefore be constructed with a certain logic, which may not be obvious for those who have not done molecular dynamics before. Hence, you can use a template file that can be found on the VM, where you can just modify some variables at the beginning of the file and the complicated `LAMMPS` syntax is taken care of in the rest of the file.

```
# ------ Units --------------------------------------
units metal                                          # eV, ps, Angstrom, bar

# ------ Simulation files ---------------------------
variable ff_file        index        force field.lmp   # force field file
variable coord_file     index             coord.lmp    # coordinates file
variable traj_file      index        trajectory.dcd    # trajectory file

# ------ Simulation type ----------------------------
variable minimisation   index                   no     # energy minimisation
variable md             index                  yes     # molecular dynamics

# ------ Molecular dynamics variables ---------------
variable ens            index                  npt     # ensemble (nve, nph, nvt, npt)
variable timestep       index                0.001     # simulation timestep (time units)
variable nsteps         index              1000000     # number of MD steps

# ------ Thermostat ---------------------------------
variable temp           index                300.      # starting temperature (K)
variable trel           index                0.1       # thermostat relaxation time (time units)

# ------ Barostat (NPT runs only) -------------------
variable pres           index                  1.      # pressure (pressure units)
variable prel           index                  1.      # barostat relaxation time (time units)

# ------ Output -------------------------------------
variable nthermo        index                1000      # thermodynamic data output frequency
variable ntraj          index                1000      # trajectory output frequency

# ------ Random number seeds ------------------------
variable iseed0         index               15973      # Random seed for initial velocities
variable iseed1         index               96248      # Random seed for the thermostats
```

The file is reasonably commented and each line contains a short description of what the variable is used for. Not all variables are used, and which ones are relevant depends on the run type, the others are ignored. One of the most important pieces of information in that file is the choice of units, at the top of the file. The command `units metal` set the input and output units to a consistent set of units, which are used in all the inputs and outputs of the MD simulations. There is also a variable for setting up a short initial equilibration stint (`nequil`), which may be useful if the simulation crashes due to a nonphysical starting configuration.

The main variable you will have to set is the temperature (`temp`), which in the example above was set to 300 K, and the number of step (`nsteps`), which when multiplied by the time step size (`timestep`) give you the total simulated time. Other important variables are the names of the coordinates (`coord.lmp`) and force field (`force field.lmp`) file, which have to match the names that you have chosen for your files; in particular the coordinates file may have a different name. Of some importance are also the "seeds" for the random number generators, which is useful if you want to produce different trajectories starting from the same input coordinates.

Table 1: Units used in the LAMMPS input and output files

Quantity	Units
mass	grams/mole
distance	Angstroms
time	picoseconds
energy	eV
velocity	Angstroms/picosecond
temperature	Kelvin
pressure	bar
charge	multiple of proton charge
density	gram/cm^3

```
en.wikipedia.org/wiki/Random_seed
en.wikipedia.org/wiki/Pseudorandom_number_generator
```

Note that if you want to run simulations of DCE and DBE in the gas phase, *i.e.* when your initial density is very low, you would have to set the ensemble to NVT in the `LAMMPS` input file, otherwise the simulation will crash or produce a liquid sample. This can be done by setting the variable `ens` to `nvt`. It is always good practice to visualise your MD trajectory to ensure that the simulations has done what you expected it to do.

9.4.1 Running LAMMPS

```
docs.lammps.org/Run_basics.html
```

Once you have prepared the coordinates and input files, running an MD simulation with `LAMMPS` is pretty straightforward. It is again best to use `tmux` to ensure that the simulations keep running when you disconnect your computer from the VM. The basic command to run `LAMMPS` using multiple cores in a terminal is

```
$ mpirun -np 2 lammps -i lammps.inp -log lammps.out
```

where we have chosen to use 2 cores of the VM, `lammps.inp` is the input files and `lammps.out` is the output file. The simulation will create also one extra file, `trajectory.0.dcd`, which contains the atomic trajectory and it can be visualised with `VMD`.

It is also possible to change the values of any variables of type `index` that are listed in the input file using the `-v` flag on the command line, *e.g.* this command will run a simulation with different seeds for the random number generators

```
$ mpirun -np 2 lammps -i lammps.inp -log lammps.out -v iseed0 567 -v iseed1 432
```

9.4.2 How many cores should you use?

Choosing an appropriate number of cores for your simulation is a matter of compromise, and fairness. It is worth keeping in mind that doubling the number of cores never leads to a simulation running two

times faster. That is because there is a communication overhead that increases with the number of cores. Imagine having to do a simple repetitive task, peeling and chopping 20 potatoes to make a mash, and say it takes one minute to peel each potato. If you do the job alone it will take you 20 minutes; on the other hand if you share the job with three another people, before peeling the potatoes you need to divide the lot in four batches, and after each of you has finished peeling you need to go around your friends and collect their "results", two tasks that you didn't need to do when doing the job alone. It immediately follows that the more people you share the job with, the larger overhead there is and there is a point where you're trying to share the job with too many friends, and it will take longer than doing the job alone, although it may be more fun. Molecular dynamics simulations work in a similar way, the calculation of energies and forces can be divided across multiple cores, but these quantities need to be communicated between the cores to generate the correct atomic trajectory, which generates a time overhead. Although it is always a worthwhile exercise to test the performance and scaling of a code for a given system, we would leave that as an optional exercise to do at the end of the lab and for now we would consider that simulations runs efficiently if (number of atoms / number of cores) > 1,000.

In order to demonstrate this, I have run a short 10 ps MD simulation with a 4,000 atoms system. I have estimated the performance of `LAMMPS` (Table 2) and it is evident that beyond 4 cores there is a significant drop off in efficiency, and that using 64 cores makes the simulation only 11 times faster, and actually slower than on 32 cores. Although you might be tempted to say that 32 cores is best, consider that you would need to run multiple simulations at different temperatures, which do not need to run sequentially. Indeed, for the system used in this example running 4 simulations at the same time is more efficient that running the same 4 simulations on 16 cores sequentially, 57 s *vs* 85.2 s; and running 16×1 core simulations in parallel is significantly better (197 s) than running those simulations sequentially using 16 cores each (340 s). Table 3 shows the fraction of the simulation time that the `LAMMPS` spent computing

Table 2: Scaling performance of LAMMPS using a 4,000 atoms system

Cores	ns/day	hours/ns	timesteps/s	Run time(s)	Speedup	Efficiency %
1	4.4	5.5	50.7	197.1	1.0	100
2	8.2	2.9	95.5	104.7	1.9	94
4	15.1	1.6	174.9	57.2	3.4	86
8	26.1	0.9	302.5	33.0	6.0	74
16	40.5	0.6	469.3	21.3	9.3	57
32	55.2	0.4	639.1	15.6	12.6	39
64	47.9	0.5	554.1	18.0	10.9	17

the pairwise interactions, computing the long range electrostatics and in "communication" between the cores. After the simulation has finished, you can check where `LAMMPS` has spent the most amount of time by opening the output file.

```
docs.lammps.org/Run_output.html
```

We also need to make an argument about fair-sharing of the resources. The VM you are using is shared among all the students in the class, and so it is appropriate that each student has access to the same amount of resources. In the introductory briefing session you will be told how many cores you can use without impeding your peers.

Table 3: Percentage time spent by LAMMPS in computing the pairwise interactions, in the reciprocal space calculation and in "communication" between the cores.

Cores	Pairwise (%)	K-space (%)	Communication (%)
1	77.7	14.0	0.5
2	73.6	17.4	2.0
4	67.8	11.2	4.4
8	58.9	22.9	7.8
16	46.3	30.0	12.7
32	31.8	37.7	18.9
64	16.0	48.6	22.8

9.5 Data analysis & Post-processing

```
docs.lammps.org/Run_output.html
docs.lammps.org/dump.html
```

In general, molecular dynamics simulations produce two main types of output, one containing the instantaneous thermodynamic properties of the system and one containing periodic snapshots of the atoms' positions (the trajectory). From the thermodynamic information contained in the two files you can then compute almost any property of your system; for example the fluctuations of the cell volume allow you to compute the isothermal compressibility of your system, the fluctuations of the dipole moment of the system allow you to compute the static dielectric constant of your material, etc. The thermodynamic quantities are buried in the midst of `LAMMPS`' output file in a section that is enclosed between a line starting with `Step` and a line starting with `Loop`, which allows us to easily extract the data and write them in a more manageable format. For example this simple `awk` command will extract the section with the thermodynamic data from the `LAMMPS` output files and write it in a new file called `data.out`

```
$ awk '/Loop/{f=0}; {if(f==1){print}}; /Step/{f=1;print "#",$0}' log.lammps > data.out
```

For alternative ways to achieve the same result and a detailed explanation of this command, have a look on the web, *e.g.* `stackoverflow.com` is a very good source of information about Unix/Linux and programming in general.

```
stackoverflow.com/questions/29967395
```

Table 4 shows a list of the quantities that are commonly written in the output file, however, that is customisable from inside the `LAMMPS` input file. By plotting some of those quantities *vs* Time you can check whether your system is in equilibrium or not, *i.e.* the thermodynamic quantities oscillate around a fixed value and don't drift. As an exercise, try to process the data to compute the average volume and temperature using the portion of the simulation that you think it's in equilibrium, try also to compute the distribution of the instantaneous values of those quantities.

9.6 Visualisation of the trajectories

The atomistic trajectory will give us access to structural and dynamical properties of the system such as the radial pair distribution function, the molar volume of a molecule, the diffusion coefficient, etc.

Table 4: Thermodynamic quantities written in the LAMMPS output files using the provided input file. All quantities are expressed in the chosen units (Table 1)

Output label	Meaning
Step	Number of Steps
Time	Simulated time
E_pair	Pairwise energy (van der Waals + Coulomb)
E_mol	Intramolecular energy (bonds, angles,…)
PotEng	Potential Energy (E_pair + E_mol)
Econserve	Conserved quantity
Temp	Temperature
Press	Pressure
Volume	Volume
Cella	Cell length
Density	Density
CPU	Elapsed time (s)
CPULeft	Time to the end of the simulation (s)

However, the post-processing of atomistic trajectories is more complicated than computing averages and distributions of data and it requires reading the positions of all the atoms, frame by frame, and them computing a property that depends on those coordinates in an often complex way. In this particular case, the atomic trajectory is NOT written in a human-readable format (try opening the `dcd` file with a text editor) and therefore you need the aid of a computer program to analyse it. There are quite a few different programs that allow you to visualise atomic trajectories, which one to use for this lab depends on how you chose to connect to the VM, of if you want to download the data on your personal computer.

9.6.1 VMD

`VMD` is one of the most commonly used visualisation programs, and it allows you to perform some basic analysis of the coordinates.

```
https://www.ks.uiuc.edu/Research/vmd
```

VMD can be readily used on the VM if you had connected to it via SSH, alternatively you could install it on your computer, it is free but requires you to register. From the command line, you can simply run

```
$ vmd coord.pdb trajectory.0.dcd
```

and `VMD` should start.

9.6.2 MDAnalysis

If you connected to the VM using the JupyterHub you have to use a python the notebook and call the nglview library. This is a much less powerful visualisation tool than `VMD` but it would suffice for this lab.

Moreover, there is no post-processing tool embedded in the visualiser and MDAnalysis has to be used instead.

```
import MDAnalysis as md
import nglview as ng

path = "../testOrca"
types = "coord.pdb"
traj = "trajectory.0.dcd"

types = "coord.xyz"
traj = "geopt_trj.xyz"

system = md.Universe(path+"/"+types, path+"/"+traj)
view = ng.show_mdanalysis(system, gui=True)
view.center()
view.representations = [{"type": "ball+stick", "params": {"sele": "all"}} ]
view.camera = 'orthographic'
# view.add_unitcell()
view
```

9.6.3 Calculation of the distribution

The distribution of torsional angles can be computed using GPTA by running this command

```
$ gpta.x --i coord.pdb trajectory.0.dcd --frames 100:200 --top \
    --molprop +id M1 +torsion 1,2,3,4 +histo 0,twopi +out histogram.out
```

Note that you must provide the indices of the atoms that form the dihedral angle that you are considering in the order they appear in your molecule, which depends on how the initial geometry was prepared. In the example above, the torsional angle is formed by the atoms in the order `1-2-3-4`, where the dashes represent covalent bonds. In the screen output you will see key information about your system and a section about the calculation of the distribution of the torsional angle that you have chosen.

```
*snip*
--------------------------------------------------------------------------------
Computing molecular properties
...Output file.....................................: histogram.out
...Molecule's type selected........................: M1
...Number of molecules selected....................:         500
...Computing -> Torsional angle
......Atom indices used............................: 1 2 3 4
...Computing the histogram
......Distribution limits..........................:      0.0000         6.2832
......Number of bins...............................:         100
--------------------------------------------------------------------------------
*snip*
```

Besides the ordered list of atom indices to compute the torsional angle, the other important part of the command is the selection of frames in the trajectory file that you want to process, which limits the analysis

to the frames between 1,000 and 2,000 (`--frames 100:200`). Things to keep in mind when selecting the range of frames to process

- The initial part of the simulations is likely not to be equilibrated, as it takes time for the system to reach the set temperature
- The more frames you select the better your statistics

A good way to check whether your distribution is converged and your system is in equilibrium is to analyse to separate chunks of the trajectory and verify that the distribution remains unchanged, *e.g.* compute the distribution of torsional angles using frames 100-200 and frames 300-400. You can used `gnuplot` to quickly check the shape of the distribution and you should obtain something like the curve in Figure 4.

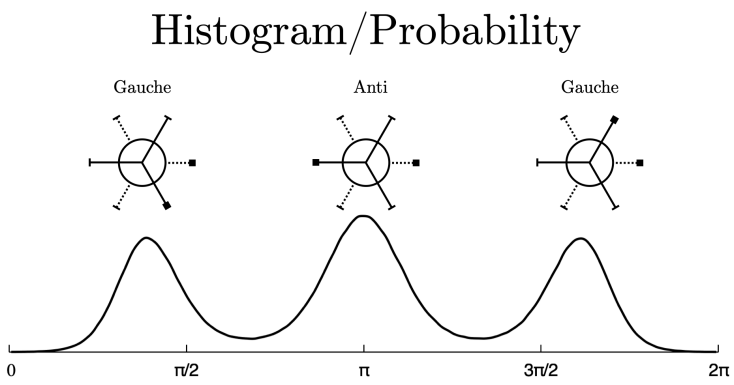

Figure 4: Histogram of the recorded configurations during a MD simulations of 1,2-dihaloethane.

Another option available to analyse your trajectories and extract the dihedral angle distribution is to use Python. More specifically, a Python package called MDAnalysis

`https://www.mdanalysis.org/`

this has the added benefit of allowing the student to get a more "hands-on" experience in extracting useful information from the trajectory that they created. An example Python script and Jupyter Notebook have been included that demonstrate how to use MDanalysis to do this.

9.6.4 Theory

The probability that a molecule has a certain torsional angle is just the histogram normalised so that the area underneath the curve is equal to one

$$\int \mathrm{d}\theta \; P(\theta) = 1. \tag{18}$$

In this case, because the histogram computed by GPTA is just counting the number of molecules that had any given angle, the integral of the histogram would give the number of molecules in the system times the frames in the trajectory, which is indeed the normalisation constant that would transform the histrogram into a probability

$$P(\theta) = \frac{h(\theta)}{N_{mol} N_{frames}}. \tag{19}$$

Using Equation 16 we could transform the probability into the free energy for the rotation around the C-C bond (Figure 5)

$$G(\theta) = -k_B T \ln[P(\theta)] \tag{20}$$

which will give us the well-known energy profile with three minima, one at 180 degrees, which corresponds to the *anti* conformation, and two at ∼60 and ∼120 degrees, which corresponds to the two *gauche* conformations. However, that is not very useful to compute the free energy between the *anti* and *gauche* conformations. That is because, unlike what you normally see in textbooks, the *anti* conformation does not only correspond to a dihedral angle of 180 degrees, but due to thermal fluctuations there is a fairly large range of angles that could be assigned to that conformation, and analogously for the *gauche* conformation. You can indeed consider that all the states between the two maxima at ∼120 and ∼240 degrees can be assigned to the *anti* conformation. It is therefore more correct to compute the free energy difference

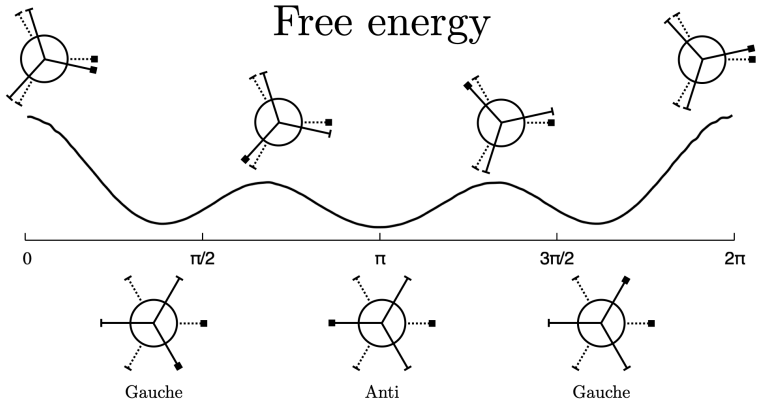

Figure 5: Free energy for the rotation around the C-C bond in 1,2-dihaloethane.

between the *anti* and *gauche* states from the probability distribution using Equation 17

$$\Delta G = -k_B T \ln(P_{anti}/P_{gauche}). \tag{21}$$

Indeed, we can compute the probability that a molecule is in the *anti* conformation by integrating the probability over the range of angles that can be assigned to that conformation

$$P_{anti} = \int_{\theta_0}^{\theta_1} \mathrm{d}\theta \; P(\theta) \tag{22}$$

and, consequently, the probability that a molecule is in the *gauche* conformation is the integral over the other angles, or simply

$$P_{gauche} = 1 - P_{anti}. \tag{23}$$

10 Micro-Raman experiments

The Raman spectrum of 1,2-dichloroethane and 1,2-dibromoethane will be collected using the microRaman instrument in the SPM facility at Curtin. The liquids will be sealed inside capillaries and the temperature will be varied using the cooling/heating stage of the instrument. There is a specific workflow to follow to enable collection of high quality Raman data, as described below.

10.1 Starting the Instrument

To start the instrument you will first need to ensure that the power is turned on for the spectrometer/microscope, CCD detector, and computer, at which point you can open the control software from the desktop icon (Project 4). The laser can then be turned on, but be sure to ensure shutters are in the "closed" position before starting the laser.

10.2 Focusing the Raman microscope

The Raman microscope operates in a confocal optical configuration. This is advantageous as the photons are focused into a small lateral spot size (0.5 x 0.5 μm, 1 σ, in X and Y dimension), but also a small volume (1 μm in Z dimension). The confocal "depth" is therefore, $\approx 1\mu$m. The result is much improved S/N(?) within the confocal volume, and the ability to resolve chemical features that occur at different depths (or "z-planes") in your sample. However, the disadvantage is that it is very difficult to detect any signal, and therefore difficult to optimise/align your instrument, if you are out of focus. Fortunately, the focus point of Raman laser (we will be using green laser, 532 nm) is approximately the same as that of white light when using the instrument as an optical microscope. Therefore, an important first step is to focus the microscope, using white light illumination and bright field optical configuration. There are generally two ways you can do this; focusing on a structured surface (e.g. a piece of dirt on a Si wafer), or by adjusting the field stop aperture. Your lab demonstrate will provide a demonstration of how to find the focal position of the microscope by using the field stop aperture.

10.3 Signal Optimisation

Once you have found the focal position of the microscopy using white light and a bright field optical configuration, you can then switch over to illumination with the green laser and confocal Raman configuration. We suggest initially monitoring signal using a piece of Si wafer. With the laser focused on the piece of Si, you can use the oscilloscope to monitor the number CCD counts for Si (use the Si lattice fundamental mode at ≈ 520 cm^{-1}). As you open the aperture and increase illumination on the Si, you should see the Raman scattering intensity increase. Continue to open the aperture until the signal no longer increases. You now need to fine tune your alignment. Initially, you will fine tune the alignment by adjusting the focus, by making small movements in the Z dimension (e.g., steps of 50 μm or less). Remember, the Raman optics and lenses are fragile, and we don't want to break them! The "safest" approach, is to initially move the sample out of focus by moving the sample further away from the microscope lens (*i.e.* , by lowering the stage). As you lower the stage (in small 50 μmm steps) you should see your Raman Si signal decrease. Next, you should bring the sample back into focus, by moving the stage in the other direction. As you move the stage in small steps, monitor the intensity of the Raman Si signal, and find the position of maximum intensity (this is the confocal point for the green laser).

If you don't have a strong Si signal, it might be due to misalignment of the fibre optic cables. Your demonstrator will show you how to optimise alignment of the cables. Remember: only ever try to re-align the fibres if you have some signal. If you do not see any signal, DO NOT attempt to re-align the fibre optics. Once the fibre optics are misaligned, it is a very hard (and very frustrating) job to realign them.

10.4 Energy calibration (what can cause calibration mis-alignment)

You should now have your Raman microscope in focus and Si signal optimised. The next step is to record a Si spectrum for energy calibration. You will calibrate the spectrometer such that the Raman scattering band of Si is centred at a Raman shift of 520 cm-1

10.5 Determining data collection parameters

You should now have the Raman microscope in focused, signal optimised, an energy calibration spectrum recorded, and you are now ready to collect data. At this stage, you probably have the laser aperture fully open, which may burn your sample. It is a good idea, while the Si wafer is still in place, to reduce the laser aperture so that the Si signal is reduced to barely detectable (*i.e.* , you are using the minimum number of photons to see your signal).

You are now ready to change from the Si wafer to the DCE and DBE samples. You should start with a test sample to begin with (DCE or DBE at room temperature), and optimise your integration time, co-added scans, laser power, and grating setting.

10.6 Heating and cooling the sample

The microRaman has a temperature stage to control the temperature of the sample, which you can now use to collect a series of spectra in a wide range of temperatures. Before starting the lab, you should find what is a suitable temperature range for your two compounds, *e.g.* a range that sits comfortably between the melting and boiling points of DCE and DBE. You should also now have an idea of approximately how long each measurement will take so you can plan how many experiments you can do in the allocated time slot and decide what temperature increments you can do.

Note that the laser is a powerful energy source and it can locally heat up your sample, so it's advisable to keep the shutter closed while the temperature stage is heating/cooling your sample.

10.7 Peak integration

Using information from the literature and the QM calculations that we have performed, we are now able to select two peaks to integrate, one for the *anti* and one for the *gauche* conformations. [There are two approaches that can be used to analyse the band areas in Raman spectra (or any spectra for that matter). One approach is to simply sum the total counts (signal intensity) across a defined spectral range, such as the spectra range occupied by a Raman scattering peak (you should baseline correct the data first). Alternatively, when there are partially overlapping peaks, you can use a peak fitting approach to obtain a more accurate measurement of the peak area. In this lab, your laboratory demonstrator will show you both approaches. As part of your report you may wish to try both methods and then compare the outcomes. There is a range of software available that can be used for summing signal intensity and peak fitting of spectra. In this lab you will have Project 4, and Opus software available, but other common software such as Origin or Matlab can also perform these functions.

11 Bringing it all together

Once we get to this stage, we should have the following data

- The vibrational frequencies of DCE and DBE in the *anti* and *gauche* confirmations, and the theoretical Raman spectra obtained from the QM calculations.
  This information helped us to interpret the measured Raman spectra and to identify which peaks you should be using for the integration and the calculation of the populations of the two conformers.
- The theoretical enthalpy, entropy and free energy of the conformers (with implicit solvent) from the QM calculations at a given temperature in the gas phase (or implicit solvent).
- The histograms and the free energy difference between the *anti* and *gauche* conformers for liquid DCE and DBE in the same temperature range of the experiments.
- The measured Raman spectra of liquid DCE and DBE at a range of different temperatures and the integrals of the peaks which are unique to the *anti* and *gauche* conformers.
  This is however not enough to compute the population of the two conformers and you should now look in the literature to find an (estimate) for the scattering cross-sections.

You can then put all this information together and plot the free energy difference between the *anti* and *gauche* conformers as a function of temperature, and extract the enthalpic and entropic contributions of the free energy by fitting the standard Gibbs free energy equation

$$\Delta G = \Delta H - T\Delta S. \tag{24}$$

Then the last things to do is to survey the literature for any experimental or theoretical values for any (all) of the quantities you have calculated to validate and support you results, and of course write the final report.

12 Pre-lab questions

1. Do you expect the *anti* or *gauche* conformer of DCE or DBE to be more stable?
2. How do you expect the relative populations of the *anti* and *gauche* conformers to change with temperature?
3. What are the dipole moments of DCE and DBE?
4. What are the dielectric constants of liquid DCE and DBE?
5. What are the melting temperatures of liquid DCE and DBE?

13 Post-lab questions

1. Draw a schematic and briefly explain how a field stop aperture works, allowing you to find the focal point of the microscope?
2. Why do you need a spectrum for energy calibration? Why do we use Si?
3. What could cause the energy calibration of the Raman spectrometer to change?
4. How does the choice of grating effect your Raman spectrum?
5. Why does a higher number of lines per mm on your grating give you better spectral resolution?
6. Explain why the ratio of *anti* /*gauche* molecules in the liquid phase is different from the gas phase.
7. What do you expect the relative populations of the *anti* and *gauche* conformers in the solid phases to be?
8. If you were to run some of these experiments/simulations for $BrCH_2CH_2Cl$ and ICH_2CH_2I, what do you expect the results to look like?

14 Final report & assessment

This report is designed to be a more substantial and professional undertaking than undergraduate reports and will help prepare you to write a 100+ pages Honours thesis.
Your report should answer and discuss the following questions:

1. What is the fraction of DCE/DBE molecules in the *anti* and *gauche* conformation in the gas phase?
2. What is the fraction of DCE/DBE molecules in the *anti* and *gauche* conformation in the liquid phase?
3. Why is the population of the conformers different in the liquid and gas phase?
4. Why does the population of the conformers change with temperature?
5. What is your best estimate for the enthalpy and entropy difference between the *anti* and *gauche* DCE and DBE in the gas and liquid phase?
6. Are your numbers consistent with the literature?
7. What are the reasons of any discrepancies with literature values (if any)?

Follow the standard structure outlined below and do not exceed 3,000 words. Scientific writing is clear and concise. A marking rubric with excellent criteria is given on the next page.

- **Introduction**
  - Gives the importance/reason for the experiment (what is the question you want to answer and why?).
  - Describes the background behind your experiment (what is currently reported in the literature and what is not - the gap).
  - States how you plan to do the experiment, why it will work and how the question will be answered.
- **Methods**
  - Should have sufficient detail such that another researcher can accurately repeat your experiment.
- **Results and Discussion**
  - Report the results in an appropriate manner, using figures and tables where required.
  - The discussion explains the meaning of your data and includes:
    - Your interpretation of and the significance of the results.
    - Acknowledgement of exceptions/unexpected results and an explanation/justification for them.
    - The theoretical and practical implications of the results.
    - Comparison with previous experiments/simulations from the literature.
- **Conclusions**
  - Briefly reiterates the importance of the work (the research problem).
  - Concisely and clearly summarises the implications of your results and how the research question has been answered (or not).
  - Suggests potential future directions.
- **References**
  - You may use any referencing style supported by Curtin university, but you must be correct and consistent in your use.

15 Writing your report in the JupyterHub

If you are familiar with LaTeX and or Markdown, or want to learn how to use them, you should consider writing your report in a Jupyter notebook. Markdown has fairly limited functionalities, but you can readily embed text inside the Jupyter Notebook, which you can then export in its entirety as a pdf of as a latex bundle that you can then later compile and refine.

```
https://stymied.medium.com/why-you-should-and-should-not-use-markdown-1b9d70987792
```

For the purpose of this report, using the Jupyter Notebook and Markdown to create your document (or a draft) is that you can readily embed the python code you you to post-process the data and make the plots into a document. The final product will always look a bit less polished than a well prepared Word document or a LaTeX document, but it would be acceptable for this work here. Something that would be quite unpleasant to do in the Jupyter Notebook using Markwon is the referencing.

```
https://stymied.medium.com/why-you-should-and-should-not-use-markdown-1b9d70987792
https://www.makeuseof.com/why-is-markdown-popular-reasons-you-should-use-it/

https://www.markdownguide.org/basic-syntax/
https://www.markdowntutorial.com
https://commonmark.org/help/tutorial/
```

16 Assessment rubric

Criterion	Excellent
Overall organisation	Report has clarity, precision and sophistication. It is clearly structured with a logical flow of ideas and sound paragraph structure.
Scientific writing style	Scientific writing has mature syntax, grammar and formality as appropriate for the intended audience.
Evidence integration	Claims are well supported with relevant evidence - data or literature - (with no direct quotations, well-paraphrased, considered use of information prominence). Evidence is thoroughly explained and connected back to the claim with clear reasoning.
Introduction	Sets the tone and clearly describes the problem and the intended solution. Sufficiently draws upon current literature to
Methods	Gives a clear concise and complete description of the experimental procedures. Demonstrates the use of scientifically valid methods.
Results and discussion	Uses figures and tables to appropriately present the data (Figures and tables have clear captions, labels and colours). Discuses the meaning of each finding giving a clear and logical interpretation. Discusses how the results align with the research question and identifies key variables and limitations. Sources of error taken into account (error analyses completed when possible). Results aligned with literature.
Conclusion	Restates the significance of the experiment and results. Summarises results and incorporates new insights and questions leading to future research
Citations and referencing	References are well integrated, support the argument, and are contrasted. The referencing is fully consistent in style, prominence and placement

17 References

[1] R. v. Meer, O. V. Gritsenko, and E. J. Baerends, "Physical Meaning of Virtual Kohn–Sham Orbitals and Orbital Energies: An Ideal Basis for the Description of Molecular Excitations," *Journal of Chemical Theory and Computation*, vol. 10, no. 10, pp. 4432–4441, 2014.

[2] B. J. Berne, G. Cicotti, and D. F. Coker, eds., *Classical and quantum dynamics in condensed phase simulations: Proceedings of the international school of physics*, (Singapore, Singapore), World Scientific Publishing, June 1998.

[3] G. Henkelman and H. Jónsson, "Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points," *J. Chem. Phys.*, vol. 113, pp. 9978–9985, Dec. 2000.

[4] G. Henkelman, B. P. Uberuaga, and H. Jónsson, "A climbing image nudged elastic band method for finding saddle points and minimum energy paths," *J. Chem. Phys.*, vol. 113, pp. 9901–9904, Dec. 2000.

[5] S. Smidstrup, A. Pedersen, K. Stokbro, and H. Jónsson, "Improved initial guess for minimum energy path calculations," *J. Chem. Phys.*, vol. 140, p. 214106, June 2014.

[6] P. Lindgren, G. Kastlunger, and A. A. Peterson, "Scaled and dynamic optimizations of nudged elastic bands," *J. Chem. Theory Comput.*, vol. 15, pp. 5787–5793, Nov. 2019.

[7] J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, and D. A. Case, "Development and Testing of a General Amber Force Field," *Journal of Computational Chemistry*, vol. 25, no. 9, pp. 1157 – 1174, 2004.

18 Extras

This section will briefly mentions additional ”experiences” that can be easily implemented to extend the scope and/or duration of this laboratory.

1. Infra-Red spectra of the compounds
2. Gas phase Raman spectroscopy
3. Programming
   - Compute distributions and averages
   - Scripting to automate the submission of the MD simulations at different conditions
   - Compute torsional angle from the atomic trajectory
4. Transition state search using ORCA

   `sites.google.com/site/orcainputlibrary/geometry-optimizations#h.113i1lwlykn4`
   `www.orcasoftware.de/tutorials_orca/react/nebts.html`

5. Other compounds (easier for the simulations than for the experiments)
   - X-Y DCE/DBE mixture
   - 1-Bromo-2-chloroethane
   - 1,2-Diiodo-ethane
   - 1,2-Difluoro-ethane (interesting for the *gauche* effect) [flammable] `en.wikipedia.org/wiki/Gauche_effect`
6. Compute more properties of the liquids from MD
   - molar volume
   - density *vs* temperature
   - specific heat
   - isothermal compressibility
   - static dielectric constant *vs* temperature
   - ...

Table 5: Brief description of the most commonly used Linux commands

Command	Description
cat [filename]	Display file's contents to the standard output device (usually your monitor).
cd /directorypath	Change to directory.
chmod [options] mode filename	Change a file's permissions.
chown [options] filename	Change who owns a file.
clear	Clear a command line screen/window for a fresh start.
cp [options] source destination	Copy files and directories.
date [options]	Display or set the system date and time.
df [options]	Display used and available disk space.
du [options]	Show how much space each file takes up.
file [options] filename	Determine what type of data is within a file.
find [pathname] [expression]	Search for files matching a provided pattern.
grep [options] pattern [filesname]	Search files or output for a particular pattern.
kill [options] pid	Stop a process. If the process refuses to stop, use kill -9 pid.
less [options] [filename]	View the contents of a file one page at a time.
ln [options] source [destination]	Create a shortcut.
locate filename	Search a copy of your filesystem made at around 3am for the specified filename.
ls [options]	List directory contents.
man [command]	Display the help information for the specified command.
mkdir [options] directory	Create a new directory.
mv [options] source destination	Rename or move file(s) or directories.
passwd [name [password]]	Change the password or allow (for the system administrator) to change any password.
ps [options]	Display a snapshot of the currently running processes.
pwd	Display the pathname for the current directory.
rm [options] directory	Remove (delete) file(s) and/or directories.
rmdir [options] directory	Delete empty directories.
ssh -XY user@machine	Remotely log in to another Linux machine, over the network. Leave an ssh session by typing exit.
tail [options] [filename]	Display the last n lines of a file (the default is 10).
tar [options] filename	Store and extract files from a tarfile (.tar) or tarball (.tar.gz or .tgz).
top	Displays the resources being used on your system. Press q to exit.
touch filename	Create an empty file with the specified name.
who [options]	Display who is logged on.