Introduction to Computational Chemistry

Dustin Wheeler, Mateusz Marianski

Friday 5th February, 2021

This tutorial aims to give a basic introduction to electronic structure calculations for very simple systems.[1] As every quantum chemistry code has its own philosophy, this tutorial should familiarize you with the general-purpose Gaussian16 (hereafter abbreviated as G16) software. The experiments will also demonstrate the predictive power of quantum-chemical calculations.

First, the basic structure of an input file to the G16 software will be explained. The second part will introduce scanning along the binding curve and computing observables. The third part introduces geometric optimization of a small molecule and how to assess reliability of the result.

Prob. I: The hydrogen atom

Prob. II: Hydrofluoric acid: bond length and dipole moment

Prob. III: Hydronium cation: geometry relaxation, vibrations, and PES

As the first step, please use this link to clone the files into your Jupyter directory:

https://tinyurl.com/chem357-compchem

Do **not** copy and paste the code in this PDF into your input files. Invisible formatting characters are often copied from the PDF. Gaussian will not understand these characters and your calculations will not start.

This guide is meant to be used in tandem with the Jupyter notebook contained in the cloned folder ("pchem_comp-chem_template/CompChem_template.ipynb"). That notebook has a number of cells pre-populated for your convenience. Make sure you read the notes and comments in the notebook as you go through the lab and fill in any blanks before executing the cells.

A first look at the Shell and Gaussian16

The work in this lab will take place partly in a Jupyter notebook and partly in the Terminal program. You can open Terminal from the JupyterLab launcher or from the menu under `File` `New`. Prior to beginning work, you should familiarize yourself with the Bash shell by running through the exercises from DigitalOcean available at https://www.digitalocean.com/community/tutorial_series/getting-started-with-linux (except the part on Linux Permissions, as it isn't relevant).

The most common Bash commands are available for your reference in the appendix. Before starting, make sure you understand how to navigate the file system with `pwd`, `ls`, and `cd`, and that you can copy, move, and remove a sample file using the `cp`, `mv`, and `rm` commands. Also make sure you understand the difference between *relative* and *absolute* file paths.[2]

[2] Hint: read the appendix!

It is a good practice to perform each calculation a separate directory. The calculations are initialized by calling the `g16` command on an input file:[3]

[3] This input represents commands typed into the shell. The leading characters in blue (everything up to the dollar sign) represent the user prompt and should not be typed.

```
1   user:~$ g16 input &
```

By convention, the input file is named `input.com`, though any name will work. The basic input file is shown below.[4] This file starts a calculation on two processors using 400 MB of memory. The output will be redirected to the `input.log` output file. This file contains the basic information and results of the calculation such as the total energy, atomic forces, and so forth. Additional output files might be generated according to the specified settings. Individual components of the input file are described below.

[4] Content wrapped in angle brackets (`<*>`) should be **replaced** with the desired value and the angle brackets should be **removed** (they are not a recognized Gaussian input), e.g., `<basis>` ⇒ `STO-3G`.

Structure of a simple input file

```
1   %nproc=2
2   %mem=400MB
    #<method> <basis-set>
4   #sp scf=tight
        <empty line>
6   <title information>
        <empty line>
8   <charge> <multiplicity>
    <atom1>    <x1>  <y1>  <z1>
10  <atom2>    <x2>  <y2>  <z2>
    ...
12  <atomN>    <xN>  <yN>  <zN>
        <empty line>
```

`%nproc=2` This keyword specifies the number of processors that will be employed for calculations. The computer you are using has 2 processing cores and you should use them all.

`%mem=400MB` This keyword manages the maximum memory usage during the calculations. The most efficient memory specification is beyond this tutorial.

`method` Substitute `<method>` with the method of choice for electron–electron interactions. In this tutorial, we will use several methods, namely Hartree-Fock (HF), Møller-Plesset second order perturbation theory (MP2) and a few density-functionals. The details of each method will be covered in detail during the lecture.

`basis-set` Substitute the `<basis-set>` word with the desired basis set. This specifies a set of basis functions (for instance atomic orbitals, gaussian-type orbitals, plane waves) that will be used to express an electronic configuration. The recommended basis sets are specified in each exercise.

`sp` The `sp` command orders Gaussian to perform single-point calcula-
tions, i.e., energy evaluation of a specified structure using `method` and
`basis-set`

`scf=tight` The Schrödinger equation is solved in self-consistent
manner. The `scf=tight` option specifies tight convergence criteria for
the self-consistent cycle.

`your-comment` This line, surrounded by two empty lines, holds your
comment, usually a description of the molecule and/or calculation to be
performed.

`charge` The `charge` keyword should be substituted with the total
(integer-valued) charge of your system.

`multiplicity` The `multiplicity` keyword should be substituted
with the multiplicity of your system ($2S + 1$, where S is total spin). This
should always be an integer.

`atomX <X> <Y> <Z>` This block specifies the geometry of the system.
You can use either the atomic symbol (e.g., `C`) or the atomic number
(e.g., `6`) to specify the atom type, followed by it's cartesian coordinates in
units of Angstroms (Å).[5] This block must be followed by an empty line.

[5] Use a decimal-valued coordinate, even if it is a whole number (`0.0`, not `0`).

Remember, there are no bonds (sticks) in quantum chemistry. The
bonding is the result of the respective positions of atoms in space. The
'stick' visible in visualization programs is simply a rendering for more
intuitive display. A sample input for a square-planar molecule can be
found at the end of the section (with all values filled in).

Additional tools and programs

Bash shell A short list of the basic bash (command line) commands is
given in the appendix.

Text editor JupyterLab comes with a built-in text editor. Most files
can be opened and edited by double-clicking their icon in the file
browser panel on the left side of the window. Occasionally, you will
run into files with an extension that opens a different program (e.g.,
CSV files open the CSV viewer, which can't edit by default). In these
cases, you'll have to click "File > Open from Path…" and give the full
path to the file. New files can be created by clicking "File > New >
Text File" or clicking on the Text File icon in the launcher. Another
option is the command line text editor `vim`. When working from the
command line, this program is (almost) always available. If you plan
on continuing with computer work, it would behoove you to learn
the basics of Vim. A number of introductions to Vim are available

online. Two such examples are https://www.openvim.com and
https://vim-adventures.com/.

Scripts For some exercises, scripts are required for dedicated tasks. All
scripts you will need for this tutorial can be found in their respective
directories.

A sample `input.com` file for the XeF$_4$ molecule.

```
1   %nproc=2
2   %mem=400MB
    # b3lyp 6-31g
4   # sp scf=tight

6   Xenon tetrafluoride single point DFT calculation

8   0 1
    Xe   0.0    0.0    0.0
10  F    1.0    0.0    0.0
    F    0.0    1.0    0.0
12  F   -1.0    0.0    0.0
    F    0.0   -1.0    0.0
```

Problem 1: The hydrogen atom

In this exercise, we will look at different basis sets using the hydrogen
atom. The hydrogen atom is the only non-trivial system for which the exact
analytic solution is known. By the end of the first exercise, we will see how
various computational methods compare to each other and to the exact
solution. From a technical perspective, we will learn how to compose input
files, run basic Gaussian calculations, search for energy in the Gaussian
output, and perform basis set convergence tests.

Getting started - the hydrogen atom

Tasks

1. First, go to the `Problem_1` directory by typing in the terminal `cd`
   `~/pchem_comp-chem_template/Problem_1`. There, create a test di-
   rectory (`mkdir "name_of_directory"`) with a name of your choosing.
   Inside it, generate a simple `input.com` file (File ⟩ New ⟩ Text File) which
   contains only a single hydrogen atom, using the example shown in
   the introduction. This corresponds to a single hydrogen atom in a
   hypothetical ideal gas phase. It is located at the origin of the coordinate
   system, although its position does not matter here.

2. For the method, use `HF` (Hartree-Fock method) and minimal `STO-3G` basis set which represents each available atomic orbital with three contracted gaussian functions.[6]

3. Now, inside the directory, run G16 using the command:

```
1  user:~$ g16 input.com &
```

Once the calculation has finished, open the `input.log` file with a text editor (You may click it in the file browser or, for instance, type `less input.log` in the terminal).[7] You may need to right-click (or alt +right-click in Safari) to open the contextual menu in JupyterLab. In that menu, click Open with ⟩ Editor . If you find (Edit ⟩ Find... or cmd/ctrl + F) the line "Normal termination of Gaussian" near the end, then your calculation converged. We are now interested in the total energy. Search for "SCF Done:" inside the output file. You should find a following line:

```
SCF Done: E(UHF) = #####  A.U. after X cycles
```

This is the computed electronic energy of the H atom using Hartree-Fock theory in the STO-3G basis set. Compare it with the exact result for the hydrogen atom ($0.5\,\mathrm{Ha} \approx 13.6057\,\mathrm{eV} \approx 313.7545\,\mathrm{kcal/mol}$).[8]

4. Redo the calculation with different basis sets (`cc-pVDZ`, `cc-pVTZ`, `cc-pVQZ`) by creating a new directory, copying the input file into the new directory, and changing the respective keyword in the input file. Search the output file to find out how many basis functions are actually used in the calculations. Then, in your Jupyter notebook, plot the total energy as function of the basis set size. At which basis set does the energy converge to the exact solution?

Method performance

Repeat the calculations with different methods using the prepared bash script `performance.sh`. In the script, on the line that says `for m in METHODS`, replace the word `METHODS` with the following list of density functionals (including spaces):[9]

```
SVWN PBEPBE PBE1PBE
```

You can add in the `HF` method if you like, to check the results against your previous step. Next, execute the script by typing:

```
1  user:~$ bash performance.sh
```

The script will iterate over the specified methods and tested basis sets (STO-3G, cc-pV*x*Z, where $x =$D, T, Q) and create nested directories for each method/basis set pair. Next, it will execute the calculations. Finally,

---

[6] Gaussian commands are not case-sensitive, so `HF` is the same as `hf`, `Hf`, or `hF`.

[7] If your calculation results in an error, check your input file. The editor in JupyterLab automatically strips off the last empty line of a file, so you need to add **two** empty lines before saving in that program. If you're missing the empty line, an easy fix is to run `echo "\n " >> input.com`, then run `g16 input.com &` again. `\n` is the representation for "newline", and this appends an empty line to the end of your file. If you still can't get your input file to run, try comparing it to the sample input file and make sure you have all of the correct elements (and have replaced the appropriate placeholders with valid text).

[8] **TIP**: In later exercises, to find this value quickly and efficiently, use the command `grep "Done" input.log`. The `grep` command searches the `input.log` file looking for the phrase "Done" and outputs each line containing that phrase. Since the file contains only one such a phrase (it solved the electron-theory problem only once), there is only one such line. Please note that the capitalization matters (you can use the `-i` flag to perform a case-insensitive search).

[9] Edit the script file with the built-in editor or with Vim in the terminal.

it creates a `performance.dat` file which contains a list of basis sets, number of basis functions in the set, and the computed energy for different methods. Use this data to prepare a plot in your Jupyter notebook showing the convergence of different methods to the exact value of 0.5 Ha. Do all of them converge correctly to the same solution? The details of the listed theoretical methods to evaluate electron–electron interactions and why they converge to different values for the apparently trivial one-electron system are beyond this tutorial and will be covered in lecture later this semester.

Problem 2: Hydrogen Fluoride (HF): bond length and dipole moment

Hydrogen Fluoride (HF)

In the exercise, we will calculate the binding curve, atomization energy (ΔH_{at}), and dipole moment for the hydrogen fluoride (HF) molecule with two methods. From a technical perspective, this exercise teaches how simple shell scripting can be used to make your (computational) life easier.

1. The first task of this exercise will be to find the equilibrium bond distance of hydrogen fluoride (HF) from a series of calculations. Start by creating an input file (name it `input.temp`) which contains a F atom in the center of the coordinate system and a H atom at distance `DIST` along the z-axis. `DIST` is a variable name for the H-F distance in different computational steps. Please note that HF is a neutral closed-shell system (i.e., correctly assign the multiplicity).

   The template file should specify the method and the basis set used for computation. In this exercise, use `HF` (Hartree-Fock) and `6-31G(d,p)` basis set. Use same keywords (`scf=tight` and `sp`) from previous exercise.

2. Next, take a look at the bash script `run_scan.sh`, which runs `g16` calculations for a series of bond distances between 0.7 Å and 1.3 Å with 0.1 Å steps, and a denser step width of 0.02 Å between 0.85 Å and 0.95 Å.[10]

   Run the bash script `bash run.sh`, which will run the calculations and return you a file with a bond length vs energy. Which bond length corresponds to the lowest energy? How does the bond length compare to the experimental bond length of 0.917 Å?

3. To compare with experimental values, we compute the atomization energy (ΔH_{at}). In order to calculate ΔH_{at}, we will also need the total energy of the isolated H and F atoms. Compute the total energies for the single atoms using the methods `HF` and `6-31G(d,p)` basis set.

[10] In details, the script performs following tasks:

- create a unique directory for each computation
- copies your template input file
- replace the bond distance place holder `DIST` with the bond distance
- start G16 calculations
- `grep` for distance/energy and write it to a respective file.

Next, calculate the atomization energy (ΔH_{at}) of HF by subtracting the free-atom energies from the predicted total energy of HF (i.e., the minimum total energy found when varying bond distances).

$$\Delta H_{at} = E_{tot}^{HF} - E_{atom}^{H} - E_{atom}^{F} \tag{1}$$

How does this compare to the experimental value of $\Delta H_{at} = 135.2\,\text{kcal/mol}$ (5.86 eV)?

4. Now, let us look at the dipole moment. Search for the corresponding line in the output file. You can use a grep function for this task:

```
user:~$ grep "Dipole" file-name -A1 | grep "Tot" |
    ↳  awk '{print $8}'
```

The above command is a great example of an ugly "bash one-liner" that does the job and you don't question it. You can test the one-liner part by part (remove the last pipe(|) and everything following it, check the output, repeat) if you want to understand it better.[11] If you want to find the whole list of dipoles (from each folder beginning with "dist_" in your directory), you can run the following:[12]

```
user:~$ grep Dipole dist*/*.log -A1 | grep Tot | awk
    ↳  '{print $9}'
```

How does the dipole at the equilibrium distance compare with the experimental value of 1.82 debye? Plot the dipole moment vs. the bond distance. You will find a (mostly) linear correspondence. Do you expect this trend to continue at large distances? Why or why not?

5. Next, repeat the bond length determination using `PBE1PBE` method and same basis set. In order to do so, modify the input template, `mv` all the results into `HF` directory (`mkdir HF; mv dist_* HF/`) and rerun the bash script. In addition, you need to compute energies for hydrogen (H) and fluorine (F) again using new method. How does the optimal bond length, atomization energy and dipole moment change? In the lab report, prepare a plot with both dissociation curves, dipole moments and computed atomization energies.

Problem 3: Hydronium cation

Planar hydronium cation

This exercise covers how to perform geometry optimizations. Specifically, we will relax the H_3O^+ molecule starting from an initial planar guess for

[11] Briefly:

- `-A1` in the first `grep` command tells it to output one **(1)** line **a**fter the search term in addition to the line with the search term.

- The next `grep` command just grabs the lines containing **Tot**al values.

- `tail -n1` grabs the indicated **n**umber of lines **(1)** at the end of the input.

- `awk` is a "pattern-directed scanning and processing language" used in the Unix ecosystem. This command tells `awk` to print out the eighth record on each line of the input (the default record separator is a single space).

[12] If you want to get really fancy (by which I mean, cut down on the amount of task switch you need to do), you can run Bash commands inside of your Jupyter notebook. Just prefix the command with `!` and run the cell. For example, a cell containing `!ls` would list the contents of your current directory as output in your notebook. You can assign that output to a python variable by placing the exclamation point after the assignment equals sign: `my_dir_list = !ls`. Since you'll need to use this list of dipoles in your notebook, it's worth trying to run the `grep` function in your notebook to capture the list all at once. Just make sure you get the right path to the directory... check the contents of your directory with `!ls path/to/directory`.

the geometry.

1. The planar H_3O^+ geometry has been provided in the file `geom_planar.xyz`.

   ```
   Contents of geom_planar.xyz

   1  O     0.00    0.00   0.00
   2  H     0.92   -0.53   0.00
      H    -0.92   -0.53   0.00
   4  H     0.00    1.06   0.00
   ```

2. Create an `input.com` file, using the template provided in the first problem. Use the `HF` level of theory and the `6-31G(d,p)` basis set. We want to relax the geometry and perform the vibrational analysis of the ion. Therefore, replace the `sp` keyword ('single-point') from the template with `opt freq` ('optimization' and 'frequency'). After removing the information for the previous molecule, add the geometry of the cation at the end of the input file[13].

3. Run Gaussian.

   ```
   1  user:~$ g16 input.com &
   ```

   [13] copy by hand or, in the terminal, type `cat geom_planar.xyz >> input.com` to append the contents to then end of the file

4. Once the calculation is complete, visualize the results, we will use the notebook in JupyterLab. Execute the first few cells of the section until you get an output that shows you the molecule. This cell also outputs the total energy of the ion. Note the command that outputs this value. Also note that the molecule only shows bonds to two of the hydrogen atoms. What does the fully relaxed structure look like? Do you think that this is the structure of H_3O^+ in the gas phase?

5. We'll use two additional cells to define bonds between the oxygen and all three hydrogens and to reformat the vibrational information so we can visualize it. The final cells in this section of the notebook will show the corrected structure and list the normal modes/vibrations with their IR intensity, sorted by wavenumber (cm^{-1}). Click through the spectrum to animate some of the vibrations. You can select all of the vibrations by clicking the menu icon ($\vdots$) in the upper right corner of the output view window and using the dropdown menu for "Normal Mode". You should see that one of the frequencies is negative – select the normal mode to which it corresponds. In the discussion section (at the end of the notebook), include this table and indicate what kind of molecular motion each vibration corresponds to.

Pyramidal hydronium cation

Next, repeat the calculations for a pyramidal hydronium cation:

```
Contents of geom_pyramidal.xyz

1  O      0.00    0.00    0.00
2  H      0.92   -0.53   -0.66
   H     -0.92   -0.53   -0.66
4  H      0.00    1.06   -0.66
```

Visualize the results again using the steps from the previous section. You should see the H_3O^+ in a pyramidal conformation now. Note again the total energy of the ion and compare it with the planar structure in your discussion. Which conformation has lower energy? Next, view the vibrations table and spectrum. If calculations were done properly, all vibrations should have positive wavenumbers. Describe again in the discussion section the motion the vibrations correspond to.

Potential-Energy Surface Scan

In the final problem, we are going to inspect the potential-energy surface of the hydronium ion along its umbrella mode. In the planar cation problem, you have seen that the negative[14] frequency corresponds to such an 'umbrella' mode.

[14] In fact, it is imaginary, i is dropped by convention

- In a terminal window, change to the PES directory. You will an template input file already prepared. If you open it, you should see that the *xyz*-cartesian coordinates have been replaced with a *z*-matrix. The *z*-matrix allows precise control of the geometry within single calculations.

```
z-matrix in the PES input file

1  O
2  X    1 a
   H    1 a    2 HOX
4  H    1 a    2 HOX    3   120.0
   H    1 a    2 HOX    3  -120.0
6
   a=OH
8  HOX= 135. -1. 45
```

- The first column shows the bonding, second shows the angles between atoms and the third column specifies the dihedral angle. X is a dummy (non-existent) atom that enables control of the umbrella motion. Figure 1 explains the *z*-matrix graphically. First, calculate the $O-H$

distance using the functions defined in the first cell of this section and the optimized pyramidal geometry from the previous section. Replace the `OH`[15] in the input file with this value. Next, run the calculations using Gaussian. The calculations will perform a scan along the $X-O-H$ angle, performing a single point calculations every 1° from 135° to 90°. When the calculations are finished, import the results to Jupyter and plot the resulting potential-energy surface. Discuss these results in your report. The PES for angles below 90° is the mirror image. Use kcal/mol instead of Ha in the report.

Figure 1: Definition of hydronium ion internal coordinates. The calculations perform a scan along the $X-O-H$ coordinates for all three hydrogens from 135° to 90°. The 120° dihedral angle indicates the relative position of hydrogen atoms.

- Localize the lowest-energy and transition structure along the PES and calculate the reaction barrier of the internal flip of the hydronium ion. Whereas the energy of the pyramidal ion is comparable with the minimum on the PES, the energy of the optimized planar cation and the local maximum on the PES is different. Why? Compare the geometries.

- In the final step, rerun the calculations using the `MP2` method and compare your results. You'll need to rerun the geometry optimization (for the pyramidal molecule) using the MP2 level of theory so you get the right bond length to run the PES scan of the umbrella mode. Try plotting both PES scan curves in one plot to compare methods. Look at the difference in final energies and the difference in the barrier energy for the two methods.

Lab report

For the lab report, please prepare following data.[16] All of this can be done in the Discussion section of the Jupyter notebook. When you are finished, export the finished document as a PDF file by clicking File ⟩ ⟩ Export Notebook As… ⟩ ⟩ Export Notebook to PDF and email that PDF to me.

[16] This is the bare minimum; there are couple of open questions in the text that you should try to assess.

1. Plot the Total Energy vs. number of basis functions for a hydrogen atom for all methods used in Problem 1.

2. Plot the binding curves for HF using Hartree-Fock and PBE1PBE methods. Find the minimum distance and compute the atomization energy. Plot the dipole moment as a function of the bond distance. Plot both methods in the same image. You can do this with the `plt.plot()` method from the Matplotlib library. If you made the distances into the index for your dataframe, you can call those values with `df.index.values`. Multiple sets of *x-y* data can be plotted with `plt.plot(x1, y1, x2, y2)`, where `x1, ...` are the various lists of *x* and *y* data.

3. Prepare tables listing molecular vibrations in a hydronium ion in planar and pyramidal geometries. Make sure the wavenumbers are shown for all molecular vibrations.
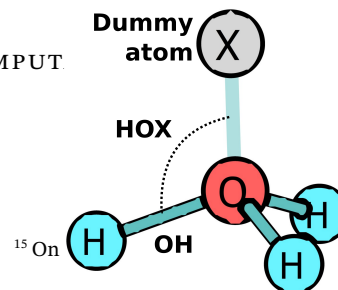
[15] On

4. Plot the PES along the HOX coordinate using HF and MP2 methods. Use kcal/mol for the *y*-axis. Compute the heigh of the barrier separating two pyramidal structures (the energy required to pass over the planar intermediate on the PES).

Appendix I: Bash and vi

Bash is a Unix shell and command language for the GNU Project and the default shell on Linux and OS X systems. We will use it to execute most programs and exercises. Below you find a list of the most import commands. Items in quotes indicate user-selected input (a directory/file name, a string of text, etc.). Bash furthermore offers a full programming language (often implemented via shell scripts) to automate tasks, e.g., via loops.

- **Basic navigation:**

  ```
  ls
  ```
  list all files and folders

  ```
  ls "dir-name"
  ```
  list files in the directory.

  ```
  ls -lh
  ```
  Detailed (**l**ong) list, **h**uman readable

  ```
  ls -l mypics/*.jpg
  ```
  list only the jpeg files in the "mypics" directory

  ```
  cd "folderName"
  ```
  change directory

  ```
  cd ..
  ```
  go up one folder, tip: string together multiple folders ../../..

- **Basic file operations:**

  ```
  cat "file"
  ```
  show all contents of a file

  ```
  head "file"
  ```
  show the top 10 lines of a file

  ```
  tail -n5 "file"
  ```
  show the last 5 lines of a file

  ```
  mkdir "dir-name"
  ```
  creates a new directory entitled "dir-name" (called folder in Windows and macOS)

  ```
  cp "file1" "file2"
  ```
  - copy "file1" to "file2"

  ```
  cp image.jpg mypics/
  ```
  - copy the file "image.jpg" to the "mypics" directory

  ```
  cp *.txt stuff/
  ```
  copy all of files ending with ".txt" to the directory "stuff"

A note on *relative* and *absolute* paths: Say you were giving directions to a location. You have two methods you can describe getting to the location:

- Relative to where you stand, or

- Relative to a landmark.

Both descriptions get you to the same location, but the former only works from your current location ("take a left, then a right, go through two lights then take another right" wouldn't necessarily work from the next town over, but works from where you stand).

In file systems, if you have `/home/user/documents/data.txt`, that's an absolute path (the starting / indicates the root of the drive). If you have `documents/data.txt`, it will only work so long as you're starting from `/home/user`. If you start in `/home/user/documents` you would need a `../` to get there correctly using the relative path.

However, no matter where you are on the hard drive, `/home/user/documents/data.txt` is a definitive way to get to that file.

```
mv "file1" "file2"
```
move (rename) "file1" to "file2"

```
mv "file1" "dir-name>/"
```
move "file1" to directory "dir-name"

```
mv "folderName/" ..
```
move directory up one level

```
rm "file1"
```
delete "file1"

```
rm -r "junk_stuff"
```
delete directory "junk_stuff" and all files contained in it

- **Extract, sort and filter data:**

```
grep "someText" "file1"
```
search for the text "someText" in "file1".[17] The `-i` flag tells grep to **i**gnore letter case (upper/lower).

```
grep -r "text" "folderName/"
```
return a list of lines in files contained in "folderName" with occurrences of "text"

- **Flow redirection and chain commands -redirecting results of commands:**

```
>
```
at the end of a command to redirect the result to a file

```
>>
```
at the end of a command to append the result to the end of a file

```
|
```
at the end of a command to send the output to another command
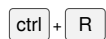
```
&
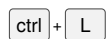```
run the command in the background

- **Basic control:**

⇥
auto completion of file or command

↑ / ↓
See previous/next commands

ctrl + R
reverse search history

ctrl + L
clear the terminal

[17] If your input has spaces, enclosing the input in double quotes (`"`) will preserve the spaces, e.g., `"Some quoted text"`.

```
!!
```
    repeat last command

**vi** is a terminal-based file edit program. By typing `vi` you open the program and create a new file that can be save later. By typing `vi "fileName"` , you open "fileName" to edit it. If "fileName" doesn't exist, you will create the new file and edit it immediately with this program.

The editor, despite its simplicity in appearance, is a very powerful terminal-based tool with numerous key-bindings. Therefore, be careful what you press. In order to start editing the file, you first need to press ⌷I ('**i**nsert') and then you can start typing. In order to save the file, press the Esc key to exit the editing mode, then type **:** ( ⇧ + ; ) to enter the command mode in the bottom of the editor and type `wq` (for '**w**rite **q**uit'). Confirm with Enter. If you want to quit without saving the file, type `q!` in command mode.

*References*

[1]   Mateusz Marianski. "Introduction to Computational Chemistry: Exercise Problems for Gaussian16". Feb. 2019.