

# Introduction to Computational Chemistry

Dustin Wheeler, Mateusz Marianski

February 9, 2024

This tutorial aims to give a basic introduction to electronic structure calculations for very simple systems.<sup>1</sup> As every quantum chemistry code has its own philosophy, this tutorial should familiarize you with the general-purpose Psi4 software. The experiments will also demonstrate the predictive power of quantum-chemical calculations.

First, the basic structure of interacting with Psi4 will be explained, along with some very basic concepts in computational chemistry. The second part will introduce the concept of scanning along a binding curve and computing observable quantities. 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>

This guide is meant to be used *in tandem* with the Jupyter notebook contained in the cloned folder ("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.

<sup>1</sup> Adapted from Marianski [1].

While going through this document, do **not** try to copy and paste the code in this PDF into your input files. PDFs are full of invisible formatting characters that can change the meaning of parts of your input files.

## A first look at the Shell and Psi4

The work in this lab will take place entirely in a Jupyter notebook. That said, it is sometimes more convenient to manipulate objects on a computer using a Terminal program, especially when we're working with many files and folders, or items on a different computer. You can open Terminal from the JupyterLab launcher or from the menu under  . You should familiarize yourself with the Bash shell (the most common way to interface with the Terminal) by running through the exercises from DigitalOcean available at [https://www.digitalocean.com/community/tutorial\\_series/getting-started-with-linux](https://www.digitalocean.com/community/tutorial_series/getting-started-with-linux),<sup>2</sup> as the Terminal is another valuable tool in your arsenal as you learn to use computers to their utmost capabilities.

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.

<sup>2</sup> Go ahead and skip the section on Permissions for now, as that can get complicated very quickly.

Also make sure you understand the difference between *relative* and *absolute* file paths.<sup>3</sup>

<sup>3</sup> Hint: read the appendix!

Psi4 has two ways to run calculations. The first is from the terminal, using a utility called `psi`. With this method, the user writes an input file (a plain text file with several directives in it), then calls that input file with the `psi` command. The results are written to a log file which can be referenced later.

The second method (and the one we will use) is a set of interactive calls available from within a Python session (*i.e.*, within a Jupyter notebook) using the [PsiAPI](#) interface. This means we first need to import Psi4 to make all of the module functions available, but we can use it inside of any valid Python code we might imagine, so everything we've learned about loops, conditional statements, variable management, and plotting will work with the data produced by Psi4.

Shown below is a sample input for Psi4.

This code starts a calculation using 500 MB of memory. It then defines a water molecule based on connectivity of each atom to the central oxygen. Finally, it computes the energy of the molecule using the `scf` method and the `cc-pvdz` basis set. We'll talk about these items later. By default, this output isn't *saved* anywhere in a file, only printed in the output of our Jupyter cell and stored as part of the returned variable. The output 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 in our input code. Individual components of the input file are described below.

#### Structure of a simple input file

```

1  # Sample HF/cc-pVDZ H2O Computation
2
3  psi4.set_memory('500 MB')
4
5  h2o = psi4.geometry("""
6  O
7  H 1 0.96
8  H 1 0.96 2 104.5
9  """)
10
11  psi4.set_options({
12      'reference' : 'rhf',
13      'basis'      : 'sto-3g',
14  })
15
16  psi4.energy('scf')
```

`set_memory(inputval)` This method sets manages the maximum

memory usage during the calculations. The most efficient memory specification is beyond this tutorial. `inputval` may be a string with units (B, kB, MB, *etc.*) or an integer number of bytes.

`geometry(geom, name='default')` `geom` is a string defining the geometry of the molecule. This can be a set of *xyz* coordinates for each individual atom or a connectivity graph called a Z-matrix (we'll discuss this later).

`reference` Specifies the type of SCF calculation to be performed, dependent on the molecule in question. `'rhf'` is the default, we'll use `'uhf'` later. If required, the value for reference will be given in each exercise.

`basis` Substitute the `'sto-3g'` 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.

`energy(name)` Substitute `name` 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. This particular calculation is done using the Hartree-Fock version of “self-consistent field theory” (hence `scf`).

There are *many* additional parameters that can be specified. Many of them are far beyond the scope of this tutorial, but we will introduce a few more options as we progress.

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.

### *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, regardless of which computer you’re using. 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/>.

### 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 directory (`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.<sup>4</sup>
3. Now, inside the directory, run G16 using the command:

```
1 user:~$ mkdir "name_of_directory"
```

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).<sup>5</sup> You may need to right-

<sup>4</sup> Gaussian commands are not case-sensitive, so HF is the same as hf, Hf, or hF.

<sup>5</sup> 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

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 ( $0.5 \text{ Ha} \approx 13.6057 \text{ eV} \approx 313.7545 \text{ kcal/mol}$ ).<sup>6</sup>

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?

<sup>6</sup> **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).

### 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):<sup>7</sup>

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:~$ grep "Done" input.log
```

<sup>7</sup> Edit the script file with the built-in editor or with Vim in the terminal.

The script will iterate over the specified methods and tested basis sets (STO-3G, cc-pVxZ, where  $x = \text{D, T, Q}$ ) and create nested directories for each method/basis set pair. Next, it will execute the calculations. Finally, 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  $0.5 \text{ Ha}$ . 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 ( $\Delta H_{\text{at}}$ ), 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.<sup>8</sup>

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 ( $\Delta H_{\text{at}}$ ). In order to calculate  $\Delta H_{\text{at}}$ , 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.

Next, calculate the atomization energy ( $\Delta H_{\text{at}}$ ) 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).

$$\Delta H_{\text{at}} = E_{\text{tot}}^{\text{HF}} - E_{\text{atom}}^{\text{H}} - E_{\text{atom}}^{\text{F}} \quad (1)$$

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

<sup>8</sup> In details, the script performs following tasks:

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

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:

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

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.<sup>9</sup> If you want to find the whole list of dipoles (from each folder beginning with “dist\_” in your directory), you can run the following:<sup>10</sup>

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

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  $\text{H}_3\text{O}^+$  molecule starting from an initial planar guess for the geometry.

1. The planar  $\text{H}_3\text{O}^+$  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
```

<sup>9</sup> Briefly:

- `-A1` in the first `grep` command tells it to output one (1) line after the search term in addition to the line with the search term.
- The next `grep` command just grabs the lines containing `Total` values.
- `tail -n1` grabs the indicated number 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).

<sup>10</sup> 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`.

2.

3.

```
1 user:~$ cat geom_planar.xyz >> input.com
```

4.

5.

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
```



## z-matrix in the PES input file

```

1 O   ^^I
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

```

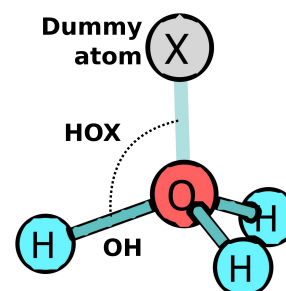


Figure 1: Definition of hydronium ion internal coordinates. The calculations perform a scan along the X–O–H coordinates for all three hydrogens from 135° to 90°. The 120° dihedral angle indicates the relative position of hydrogen atoms.

- 

- 

Export Notebook As... Export Notebook to PDF

File

1.

2.

3.

4.

- - 
  -
- - `ls`
  - `ls "dir-name"`
  - `ls -lh`
  - `ls -l mypics/*.jpg`
  - `cd "folderName"`
  - `cd ..`
- - `cat "file"`
  - `head "file"`
  - `tail -n5 "file"`
  - `mkdir "dir-name"`
  
  - `cp "file1" "file2"`
  - `cp image.jpg mypics/`
  
  - `cp *.txt stuff/`
  
  - `mv "file1" "file2"`
  - `mv "file1" "dir-name">/"`
  - `mv "folderName/" ..`
  - `rm "file1"`
  - `rm -r "junk_stuff"`
- - `grep "someText" "file1"`

```
grep -r "text" "folderName/"
```

- 

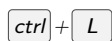
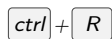
>

»

|

&

- 



!!



[1]