Introduction to Computational Chemistry

Dustin Wheeler, Mateusz Marianski

February 19, 2024

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 Psi4 software. The experiments will also demonstrate the predictive power of quantum-chemical calculations.

[1] Adapted from Marianski [1].

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.

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 File ⟩ New . 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`,[2] as the Terminal is another valuable tool in your arsenal as you learn to use computers to their utmost capabilities.

[2] Go ahead and skip the section on Permissions for now, as that can get complicated very quickly.

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.[3]

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 files 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

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

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

maximum memory usage during the calculations. The most efficient memory specification is beyond this tutorial. The default value is 500 MB, which should be sufficient for most simple computations.

`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` This option sets 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 for the calculation. 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 I: 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 Psi4 calculations, search for en-
ergy in the Gaussian output, and perform basis set convergence tests.

Getting started

Tasks

1. We'll begin by setting some options for the calculation. Because
   we're dealing with a hydrogen atom, we have to use an "unrestricted
   Hartree-Fock" method (the unpaired electron must be properly ac-
   counted for), hence `"uhf"`. We'll set the basis set to a very minimal
   set called "STO-3G" (each available atomic orbital is represented
   by three contracted gaussian functions). Finally, we define the
   molecule geometry and assign it a name. For this *very* simple sys-
   tem, setting the basis, method, and geometry is as simple as:

   ```
   1  h_atom = psi4.geometry("H")
   2  basis = 'sto-3g'
      psi4.set_options({
   4                   'reference': 'uhf'
                      })
   ```

   With a single atom, there's no need to identify connectivity or loca-
   tion of the atom in space. We use the `set_options()` method and
   create a dictionary of options we'd like to set.

2. Now that we've defined a molecule and basis, we'll tell Psi4 what
   sort of calculation we'd like to run. To begin, we'll perform a sim-

ple energy calculation and ask Psi4 to return the energy and the wavefunction for the system.

```
1  energy, wfn = psi4.energy('hf'+basis,
   ↳  return_wfn=True)
2  nbf = wfn.basisset().nbf()
   print(f'{energy=}\t{nbf=}')
```

The option `return_wfn=True` tells Psi4 to return information about the wavefunction for the system. From that, we pull out the number of basis functions (`nbf`) being used in our calculation and print out the energy and number of basis functions.

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 ($0.5\,E_h \approx 13.6057\,\mathrm{eV} \approx 313.7545\,\mathrm{kcal/mol}$).

3. Redo the calculation with your original basis set (STO-3G) and the following basis sets: cc-pVDZ, cc-pVTZ, cc-pVQZ by creating a list of the basis set names (as strings), then setting up a loop to perform the calculation and save the energy and number of basis sets to a pair of lists.[4] Then, plot the total energy as function of the basis set size. At which basis set does the energy converge to the exact solution?

[4] You can find a rough overview of basis sets at Wikipedia and a Psi4-specific overview is also available.

Method performance

In this step, we'll repeat the previous calculations with different methods by wrapping your previous work in another loop. This loop should run over the following list of methods: HF, SVWN, PBE, and PBE0. As before, we create a list containing these methods (as strings), and run a loop inside a loop. In order to save our information in a nice structure of recalling later (as we want to keep the methods separate for plotting), we're going to store our results in a dictionary object. This means we need to initialize an empty dictionary in which to save our data. This has been set up for you in the notebook template. Spend some time looking over the nested loops to make sure you see what's being done in each loop.

In the next cell, we set up a small loop to plot the data 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 II: Hydrofluoric acid (HF) – bond length and dipole moment

The hydrogen fluoride molecule (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 to use Python loops to perform repeated computations with Psi4.

1. The first task of this exercise will be to find learn a bit about the output data from a calculation Start by defining a new hydrogen fluoride molecule. This is again done for you in the first cell. You need to fill in values for the molecular charge and the spin multiplicity (defined above).[5] The input block also has space for you to specify the method and the basis set used for computation. In this exercise, use `HF` (Hartree-Fock) and the `6-31G(d,p)` basis set.

   Notice the geometry is defined in terms of the atomic connectivities (the first atom is the "center", labeled 1, other atoms are labeled sequentially with connectivity assigned by number. The line "`H 1 r`" means that a hydrogen atom is placed distance r away from atom 1.) The geometry could alternatively be defined explicitly in Cartesian coordinates like so:

   ```
   F 0.0 0.0 0.0
   H 0.0 0.0 r
   ```

   We use the variable `r` as a placeholder so we can define this distance later in our code.[6] Before we start making multiple calculations, let's make a single one and figure out which computational variables we'd like to collect. We can set the distance variable to a value with `hf_bond.r = 1.0`, then perform our energy calculation (again, making sure to return and store the wavefunction data, just as before.)

   Once we have this, you can print out all of the data saved in the wavefunction variable with `wfn.variables()`.[7] This will print out all of the properties returned by the current calculation. The data are returned as dictionary items, with the dictionary key in capital letters and the value given after the colon. An individual property can be recalled using `wfn.property('name')`, where `'name'` is the (case-insensitive) name of the key. Go ahead and print out the value for the property `'current dipole'`. Notice that the value is returned as a vector: a three-element array of x, y, and z components. Recall that, by convention, the principle bond axis of a molecule is defined as z. In a simple diatomic, this is the *only* bond,

[5] Please note that HF is a neutral, closed-shell system. Use this information when counting electrons and assigning multiplicity.

[6] This variable can be anything we like, as long as it's a valid variable name in Python (*i.e.,* `interatomic_distance` would be perfectly acceptable).

[7] Assuming you named the wavefunction variable `wfn`.

and so the entirety of the dipole should lie along the z-axis. You should verify this by making sure the x and y components of the dipole array are zero. If they are, you can take the following easy shortcut: rather than needing to calculate the total length of the vector ("taking the norm"),[8] you can just grab the z component as the full magnitude of the dipole vector. In the rest of this exercise, we'll be using the energy and dipole values from this calculation.

[8] Easily done using the NumPy function `np.linalg.norm`.

2. The next step of this exercise will be to find the equilibrium bond distance of hydrogen fluoride (HF) from a series of calculations. Create a loop to calculate the energy of the molecule at a range of r values between $0.7\,a_0$ to $1.3\,a_0$. Save the energy of each calculation as an element in a list, then plot the list of r values against the list of energies. The plot should have a distinct minimum. To extract the minimum value from the energy list, use the `min()` and `argmin()` functions from NumPy. Both functions take a list as the input and return the minimum value and index of the minimum value, respectively. Which bond length corresponds to the lowest energy? How does the bond length compare to the experimental bond length of $0.917\,\text{Å}$?

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. You can look back to our work on the lone hydrogen atom in Problem I to guide your work.

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).

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

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

4. Now, let us look at the dipole moment. How does the dipole at the equilibrium distance compare with the experimental value of $1.82\,\text{D}$?[9] 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?

[9] Psi4 returns values in terms of atomic units. The debye (D) is the most commonly used dipole unit, and is one of the few remaining relics in use from the cgs system. The SI unit of dipole is the coulomb-meter ($C \cdot m$), but atomic dipoles are on the order of 1×10^{-30} C·m, or a quectocoulomb-meter qC·m (no, I'm not making that prefix up, it's one of the four newest SI prefixes), making it an unwieldy quantity. The true *atomic* unit of the dipole is ea_0, or the product of the charge on one electron multiplied by the Bohr radius.

5. Next, repeat the bond length determination using `PBE0` method and same basis set. 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 of both dissociation curves, a plot of both dipole moments and a comparison of the two computed atomization energies (and the experimental value).

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 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. We'll create a Psi4 molecule based on the example provided in Problem 1. Use the `HF` level of theory and the `6-31G(d,p)` basis set. We also need to set the values for charge and multiplicity. Because H_3O^+ carries a charge, we want to tell Psi4 explicitly what values to use for these fields.

   `charge` The program will guess the charge of the molecule, but this value can be explicitly set.

   `multiplicity` As above, the program will try to guess the spin multiplicity of the system ($2S + 1$, where S is total spin). This should always be an integer. It can be explicitly set when necessary.

   We want to relax the geometry and perform the vibrational analysis of the ion. Rather than calculating energy with `psi4.energy()`, we'll let Psi4 optimize the geometry, then ask it to perform a frequency analysis on the optimized molecule. This is done by calling `psi4.optimize()` followed by `psi4.frequency()`, specifying the appropriate method. This is set up for you in the second code cell in the section for Problem III.

3. Once the calculation is complete, we will visualize the results 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 doesn't show bonds between any of the atoms. What does the fully relaxed structure look like? You can use your mouse to click and drag on the molecule to rotate it. Do you believe that this is the structure of H_3O^+ in the gas phase?

4. 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 (⋮) 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
```

It's probably easiest to copy the group of cells from the planar section, then work through each one and modify the "planar" values to be "pyramidal". 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[10] frequency corresponds to such an "umbrella" mode.

Again, many of the steps have been set up for you already. In the main folder, you should see a file called `h3o.zmat`. 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  @H
> 2  O   1 a
>    H   2 a   1 HOX
> 4  H   2 a   1 HOX   3  120.0
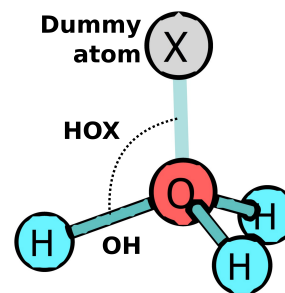>   H   2 a   1 HOX   3 -120.0
> ```

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.

The first column shows the atom identity, the second specifies to which atom it's bonded, the third column is the bond length, the fourth and fifth give a second reference atom and the angle between the line atom and the two reference atoms, final two columns specifies the dihedral angle between the prior bond and a third reference bond. @H is a dummy (non-existent) atom that enables control of the umbrella motion. Figure 1 explains the *z*-matrix graphically.

1. 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. Set the value of `a`[11] in the input molecule to this value.

2. Next, run a set of energy scans using Psi4. You should write a loop to perform an energy calculation for different values of the X−O−H angle, saving the energy every 1° from 90° to 135°.[12]

3. When the calculations are finished, plot the resulting potential-energy surface. Discuss these results in your report. The PES for angles below 90° is the mirror image of the values above 90°. Use kcal/mol instead of E_h in the report.

4. 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.

5. In the final step, rerun the calculations using the MP2 method and compare your results. You'll need to rerun the geometry optimiza-

tion (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. You do not need to save the output file; it is possible to get geometry data directly out of the Psi4 output. This is left as an exercise for the student. 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.[13] All of this can be done in the Discussion section of the Jupyter notebook. When you are finished, save a copy of your report to the folder `shared/submissions`.

[13] This is the bare minimum requirement; 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. Remember, 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.

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.

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.

- **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"

```
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".[14] 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
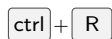
```
&
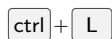```
run the command in the background

- **Basic control:**

⇥
auto completion of file or command

↑/↓
See previous/next commands

ctrl + R
reverse search history

ctrl + L
clear the terminal

[14] 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 typ-
ing `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.