

# Programming Project 2: Hessian

## *Computing the Hessian matrix by finite differences*

Compute the Hessian matrix of a molecule (equation 1) from single-point energies at displaced geometries. The default displacement size should be  $0.005 a_0$ . You should import and use your `Molecule` object from Project 0. Wherever it makes sense, you may want to add methods to your `Molecule` class in order to clean up your code.

### Extra Files

<u>file name</u>	<u>description</u>
<code>project2_input.dat</code>	sample RHF/cc-pVDZ input file for H <sub>2</sub> O
<code>template.dat</code>	an template for generating input files; after reading this to a <code>str</code> , you can fill in the geometry block using <code>.format()</code>

### Equations

Let  $N$  be the number of atoms and let  $(x_A, y_A, z_A)$  be the Cartesian coordinates of the  $A^{\text{th}}$  atom.

$$(\mathbf{H})_{AB} = \frac{\partial^2 E}{\partial X_A \partial X_B} \quad (X_{3A-2}, X_{3A-1}, X_{3A-0}) = (x_A, y_A, z_A) \quad \text{for } A \in \{1, \dots, N\} \quad (1)$$

$$\frac{\partial^2 E}{\partial X_A^2} = \frac{E(X_A + h) + E(X_A - h) - 2E(X_A)}{h^2} \quad \text{for } X_A = X_B \quad (2)$$

$$\frac{\partial^2 E}{\partial X_A \partial X_B} = \frac{1}{2h^2} (E(X_A + h, X_B + h) + E(X_A - h, X_B - h) - E(X_A + h, X_B) - E(X_A - h, X_B) - E(X_A, X_B + h) - E(X_A, X_B - h) + 2E(X_A, X_B)) \quad \text{for } X_A \neq X_B \quad (3)$$

### Procedure

1. build molecule object (`mol`) from `molecule.xyz`
2. build input file template (`template`) from `template.dat`
3. `def generate_inputs(mol, template, disp_size = 0.005, directory = "DISPS"):`
  - (a) make directories with input files for the reference geometry and for the  $3N(3N+1)$  unique displacements needed to evaluate equations 2 and 3
4. `def run_jobs(mol, command = "psi4", directory = "DISPS"):`
  - (a) walk through the directories created in step 3 and execute `command` in each one
5. `def build_hessian(mol, energy_prefix, disp_size = 0.005, directory = "DISPS"):`
  - (a) write a helper function to grab the energy value (as a `float`) immediately following `energy_prefix` in an output file (for the given template, `energy_prefix` should be `"@DF-RHF Final Energy:"`)
  - (b) initialize an empty `numpy.array` to hold the Hessian matrix
  - (c) loop over elements of the Hessian (equation 1) and evaluate them using equations 2 and 3
  - (d) save the matrix to a file and return it at the end of the function
6. import your `frequencies` function from Project 1 and use it to calculate frequencies and normal modes from the return value of `build_hessian`

Possible extensions of this project: 1. make it object-oriented; 2. generalize it to work with programs other than PSI4; 3. generalize it to allow cluster job submission as well as direct execution.