032CM - 2025

PROGRAMMING FOR COMPUTATIONAL CHEMISTRY

Functions in Python

Gianluca Levi

gianluca.levi@units.it, giale@hi.is

Office: Building C11, 3rd floor, Room 329

Fall 2025

Specific vs. general implementation

Midpoint method: $$\int_a^b f(x)\,dx \approx h\sum_{i=0}^{n-1} f(x_i) \qquad x_i = \left(a + \frac{h}{2}\right) + ih$$

Specific implementation (no functions)

Code designed for integral of one particular problem

- **Error-prone**: Formulating the general problem for our specific problem leads to mistakes
- **Repeated formula**: Integrand might need to be repeated in several places
- **Hard to change**: Integral of a different function involves many edits (and new errors)

Specific vs. general implementation

Midpoint method: $$\int_a^b f(x)\,dx \approx h\sum_{i=0}^{n-1} f(x_i) \qquad x_i = \left(a + \frac{h}{2}\right) + ih$$

Specific implementation (no functions)

Code designed for integral of one particular problem

- **Error-prone**: Formulating the general problem for our specific problem leads to mistakes
- **Repeated formula**: Integrand might need to be repeated in several places
- **Hard to change**: Integral of a different function involves many edits (and new errors)

General implementation (using functions)

Algorithm implemented with **functions** that can be **reused** for multiple specific problems

- Requires some **abstract thinking**
- Code can be used **over and over again**
- Improve **portability** and to **avoid repeating lines** of code
- more **understandable** and **maintainable** code

General structure of a function

```
def function_name(p1, p2, p3=default_val ...): # header
    """
    This is a docstring
    """

    var_loc = p1 + +p2
    <expression>
    <expression>
      ...
      ...

    return result_1, result_2, ...          # last line

var_glob = value
```

General structure of a function

Positional parameters

Keyword parameter

```
def function_name(p1, p2, p3=default_val ...): # header
    """
    This is a docstring
    """

    var_loc = p1 + +p2
    <expression>
    <expression>
        ...
        ...

    return result_1, result_2, ...          # last line

var_glob = value
```

General structure of a function

Positional parameters **Keyword parameter**

```
def function_name(p1, p2, p3=default_val ...): # header
    """
    This is a docstring
    """

    var_loc = p1 + +p2
    <expression>
    <expression>
       ...
       ...

    return result_1, result_2, ...          # last line

var_glob = value
```

Function body

General structure of a function

Positional parameters

Keyword parameter

```
def function_name(p1, p2, p3=default_val ...): # header
    """
    This is a docstring
    """

    var_loc = p1 + +p2
    <expression>
    <expression>
       ...
       ...

    return result_1, result_2, ...          # last line

var_glob = value
```

Function body

Local variable

Global variable

Problem 1

The Morse potential can be used to approximate the interatomic potential of diatomic molecules and is defined as

$$V(r) = D_e \left(1 - e^{-\alpha(r-r_e)}\right)^2,$$

where D_e is the well depth, α is a parameter controlling the curvature of the potential, and r_e is the equilibrium bond length.

Use the following parameters for the Morse potential of carbon monoxide (CO):

$$D_e = 11.22 \text{ eV}, \qquad r_e = 1.128 \text{ Å}.$$

(a) Write a Python implementation of the midpoint method in vectorized form to compute the integral

$$I = \int_a^b V(r)\, dr.$$

(b) Compute the integral of the Morse potential between 1.5 and 2 Å^{-1} for two different values of the parameter α: $\alpha_1 = 2.594\ \text{Å}^{-1}$, $\alpha_2 = 1.0\ \text{Å}^{-1}$.

For each case, evaluate the integral for increasing numbers of subintervals n, where $n = 2, 4, 8, 16, \ldots, 2^{20}$. In the same figure, plot the difference between the computed integral and its value at the finest grid, i.e. $I(n) - I\left(n = 2^{20}\right)$, as a function of n for the two cases.

(c) Compare the convergence behavior of the two curves and discuss the differences. How does the steepness of the potential (controlled by α) affect the convergence of the midpoint rule?

Hints: The midpoint method for n equally spaced subintervals computes an integral according to

$$I \approx h \sum_{i=0}^{n-1} f(x_i), \qquad x_i = a + \left(i + \tfrac{1}{2}\right) h, \qquad h = \frac{b-a}{n}.$$

To better analyze the convergence differences between the two cases, vary the range of the x-axis in your plot to more clearly observe how the curves behave at different values of n.

Assignment 4

Problem 2

Load the data from the file `october_temperatures_trieste_reykjavik_1961_2024.csv` using the NumPy function `np.genfromtxt()`. You can find the file under `/home/pcc/gianluca/assignment4/`. This file contains three columns:

Year - Average Oct. temperature in Trieste (°C) - Average Oct. temperature in Reykjavík (°C).

(a) Plot the average October temperature as a function of year for both Trieste and Reykjavík in the same figure. Label the axes clearly and include a legend.

(b) Compute the average October temperature over the first and last 10 years of the dataset for each city. Compare the results and discuss whether the average temperature in October has increased, decreased, or remained the same in each city.

(c) Find the year with the highest and lowest October temperature for both Trieste and Reykjavík, and print both the year and the corresponding temperature.

(d) Create a bar plot showing the averages of the October temperature computed over the first and last 10 years in question (b). Include in the same plot the bars for the two cities to visualize how the climate has changed over time for each of them.

Hint: You can read the file and skip the header line using the command:

```
data = np.genfromtxt('filename.csv', delimiter=',', skip_header=1)
```

Then access the columns using:

```
years = data[:, 0],   trieste = data[:, 1],   reykjavik = data[:, 2].
```

Use `np.mean()`, `np.argmax()`, and `np.argmin()` to compute averages and find extreme values.

Problem 3

The potential energy of a carbon dioxide (CO_2) molecule confined to one dimension along the molecular axis can be approximated as the sum of two Morse potentials, one for each C–O bond:

$$V(r_{CO_1}, r_{CO_2}) = V_{CO_1}(r_{CO_1}) + V_{CO_2}(r_{CO_2}),$$

where each Morse potential is given by

$$V_{CO}(r) = D_{CO}\left(1 - e^{-\alpha_{CO}(r_{CO} - r_e)}\right)^2.$$

The parameters can be chosen as: $D_{CO} = 7.65$ eV, $\quad r_e = 1.162$ Å, $\quad \alpha_{CO} = 2.5$ Å^{-1}.

(a) Define a Python function that calculates the Morse potential for a single bond and a Python function that sums two Morse potentials to give the potential energy of the CO_2 molecule:

$$E_{CO_2}(r_1, r_2) = D_{CO}\left(1 - e^{-\alpha_{CO}(r_1 - r_{CO})}\right)^2 + D_{CO}\left(1 - e^{-\alpha_{CO}(r_2 - r_{CO})}\right)^2.$$

(b) Generate a grid of r_1 and r_2 values from 0.75 Å and 3.0 Å using `np.linspace()` and `np.meshgrid()` and calculate the CO_2 potential energy over this grid.

(c) Create a contour plot of the potential energy surface using `contourf()`. Label the axes appropriately, add a colorbar, and give the plot a title.

(d) Can you identify the position of the minimum of the potential energy surface? In which direction does the energy rise more steeply, moving one bond while the other is fixed or stretching both bonds simultaneously?