032CM - 2026

PROGRAMMING FOR COMPUTATIONAL CHEMISTRY

Plotting with Matplotlib

Gianluca Levi

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

Office: Building C11, 3rd floor, Room 329

Fall 2026

Why Matplotlib?

Minimize manual post-processing

- **Easy to automate** figure production
- **Reproducibility**

Standard, most popular Python plotting library

- **Everywhere:** well documented, easy to find examples

Publication-quality figures

- Tweak everything
- Export PNG/SVG/PDF

When it's not ideal

- Can be **verbose/low-level** for complex graphics
- Does not provide statistical functions

Plot customization

https://matplotlib.org/stable/users/explain/quick_start.html

Keyword arguments for plt.plot()	Description
color	Line color
alpha	Line opacity (0–1)
linestyle	Line style
linewidth	Line width in points (default 1.5)
markersize	Marker size in points
markerfacecolor	Marker fill color

Markers

. Point
o Circle
+ Plus
x Cross
D Diamond
v Downward triangle
^ Upward triangle
s Square
* Star

Line styles

- solid
-- dashed
-. dash-dot
: dotted

Example

```
plt.plot(x, y, color='blue', linestyle='--', marker='o',
         markersize=6, markerfacecolor='white')
```

Computing the numerical integral of a function

Split up original integral into **sum of several integrals**, each covering a small part of the original interval, where the **function is approximated by a simple shape**

Computing the numerical integral of a function

Split up original integral into **sum of several integrals**, each covering a small part of the original interval, where the **function is approximated by a simple shape**

Midpoint method

$$\int_a^b f(x)\,dx = \int_{x_0}^{x_1} f(x)\,dx + \int_{x_1}^{x_2} f(x)\,dx + \cdots + \int_{x_{n-1}}^{x_n} f(x)\,dx,$$

$$\approx hf\left(\frac{x_0+x_1}{2}\right) + hf\left(\frac{x_1+x_2}{2}\right) + \cdots + hf\left(\frac{x_{n-1}+x_n}{2}\right),$$

$$\approx h\left[f\left(\frac{x_0+x_1}{2}\right) + f\left(\frac{x_1+x_2}{2}\right) + \cdots + f\left(\frac{x_{n-1}+x_n}{2}\right)\right]$$

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

Computing the numerical integral of a function

Split up original integral into **sum of several integrals**, each covering a small part of the original interval, where the **function is approximated by a simple shape**

Midpoint method

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

Try it!

Use Python to **numerically calculate the work required to stretch a CO bond** from its equilibrium length, r0 = 1.128 Å, to 2.0 Å by integrating the external force $F_{ext}(r) = dV/dr$ with the midpoint method. Use De = 11.2) eV and alpha = 2.3 $Å^{-1}$. Compare the result using two integration grids (e.g. 10 and 1000 subintervals) with the analytical result.

Plotting types

plot() — Line plot connecting data points

scatter() — Individual data points without connecting lines

bar() — Draws rectangular bars

contour() — Draws contour lines of constant values for 2D functions

contourf() — Similar to contour() but fills the regions between contours

https://matplotlib.org/stable/gallery/index.html

Assignment 4

Problem 1

The Morse potential can be used to approximate the interatomic potential of a diatomic molecule 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 controls the width of the potential, and r_e is the equilibrium bond length. The external force required to stretch the bond is

$$F_{\text{ext}}(r) = \frac{dV}{dr} = 2\alpha D_e e^{-\alpha(r-r_e)} \left[1 - e^{-\alpha(r-r_e)}\right].$$

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

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

Compare two values of the parameter α:[1]

$$\alpha_1 = 2.2994 \text{ Å}^{-1}, \qquad \alpha_2 = 1.0 \text{ Å}^{-1}.$$

(a) Create an array of 1000 equally spaced r values between 0.75 and 3.0 Å using `np.linspace()`. For each value of α, calculate arrays containing the Morse potential $V(r)$ and the corresponding external force $F_{\text{ext}}(r)$.

(b) Plot the two Morse potentials in the same figure, and the two external forces in a second figure. Label the axes with the appropriate units and include legends.

(c) Write a Python implementation of the midpoint method in vectorized form to integrate the external force.

(d) For each potential, compute the work required to stretch the bond from r_e to 2.0 Å:

$$W = \int_{r_e}^{2.0 \text{ Å}} F_{\text{ext}}(r)\, dr.$$

Use increasing numbers of subintervals $n = 2, 4, 8, 16, \ldots, 2^8$. The analytical value is

$$W_{\text{exact}} = V(2.0 \text{ Å}) - V(r_e).$$

In the same figure, plot $W(n) - W_{\text{exact}}$ as a function of n for the two cases.

(e) Compare the convergence of the two curves. How does increasing the number of subintervals affect the accuracy? Which potential requires a finer grid to reach the same absolute error?

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

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

[1] The first value is taken from A. K. Roy, "Accurate ro-vibrational spectroscopy of diatomic molecules in a Morse oscillator potential," *Results in Physics* **3**, 103–108 (2013), doi:10.1016/j.rinp.2013.06.001.

Assignment 4

Problem 2

Load the data from `october_temperatures_trieste_reykjavik_1961_2025.csv` using `np.genfromtxt()`. The file is available in `/home/pcc/gianluca/assignment4/`. Each row contains one year and the mean temperature during October in Trieste and Reykjavík, in °C. The data cover 1961–2025 and are arranged chronologically.

(a) Plot the mean October temperature as a function of year for both cities. Label the axes, include temperature units, and add a legend.

(b) For each city, calculate the average of the October temperatures over 1961–1970 and 2016–2025, the first and last ten years of the dataset. Print the two averages and their difference in °C.

(c) Identify the warmest and coldest Octobers in the dataset for each city. Print the year and the corresponding mean October temperature in each case.

(d) Create a bar plot showing the four averages calculated in (b). Group the bars by city, with two bars for each city representing the two periods. Label the axes and include a legend identifying the periods.

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. `np.argmax()` and `np.argmin()` return array indices, use these indices to obtain the corresponding years.