Computing Methods for Physics – 11 February 2022

Your exam must be submitted via google classroom by 13:30 as a single zip file containing all relevant code files, plots, and datafiles.

Capacitors

A capacitor is a device that stores electrical energy in an electric field. A capacitor C is characterized by its capacitance C — measured in Farad [F] — which can be linked to the are A of its plates and to their separation s by the formula
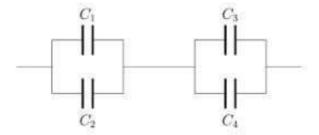
$$C = \epsilon_0 \epsilon_r \frac{A}{s},$$

(1)

where $\epsilon_0 = 8.854 \cdot 10^{-12}$ F/m is is the vacuum permittivity, and $\epsilon_r \geq 1$ is the relative permittivity of the material between the plates. By definition, $\epsilon_r = 1$ for vacuum.

You will have to use C++ to handle capacitors as objects, produce some data with operations on these objects, and fit and plot such data in Python.

Part 1 – C++

1. Design and write a class `Capacitor` with appropriate constructors, setters, and getters.

2. Write an application `app2.cpp` to showcase the methods of the class `Capacitor`.

3. Overload the plus (+) and or (||) operators so that they return a `Capacitor` instance representing the capacitor equivalent to the series and the parallel of two given capacitors, respectively.

4. Write an application `app4.cpp` where you set $C_2 = 10\,\text{pF}$, $C_3 = 5\,\text{pF}$, and $C_4 = 15\,\text{pF}$, draw 10^3 random values for C_1 from a uniform distribution between $[1, 100]\,\text{pF}$ and determine the set of corresponding 10^3 equivalent capacities for the configuration below.

Note: If you prefer, you can generate the random C_1 values with a Python script, store them as a text file, and read them in your C++ application app4.cpp in order to calculate the C_{eq} values.

Store the values of your C_1 C_{eq} pairs in a textfile called true.dat.

Part 2 – Python

Use Python for the following tasks.

1. Read in the true C_1–C_{eq} values and displace each C_1 and C_{eq} by an offset randomly drawn from a Gaussian distribution centered in 0, with standard deviation 0.5. Save these $\tilde{C}_1$–$\tilde{C}_{eq}$ values in a textfile called offset.dat.

2. Use scipy.optimize.curve_fit to fit the function $1/[1/(x + a) + b]$ to the $\tilde{C}_1$ $\tilde{C}_{eq}$ data. How do the results for a and b compare to your expectations? Plot your data and the result of your fit.

3. Read in the true C_{eq} values and this time associate an error value ΔC_{eq} to each C_{eq} by randomly drawing from a Gaussian distribution centered in 0, with standard deviation 3. Save these values in the textfile errorbar.dat.

4. Use a Monte Carlo package of your choice to fit the $\tilde{C}_1$ vs $C_{eq} \pm \Delta C_{eq}$ data with the function $1/[1/(x+a)+b]$. How do the results for a and b compare to your expectations? Plot the posterior distributions for a and b and comment your results.

Important Remarks

- C++ evaluation will be based on: correct syntax, proper return types, proper arguments of functions, data members and class interfaces, setters/getters, unnecessary void functions, comments throughout the code, separation of class implementations and interfaces.

- Python evaluation will be based on: correct syntax, avoiding C-style loops, using Python features in general, comments throughout the notebook/scripts, labels, legends and plot styling and clarity in general. The Python coding may be carried out in a notebook or in scripts, as you wish.

- The various *.dat output files you produce must be submitted.