2 — PHY 494: Makeup assignment 2 (30 points total)

Due Thursday, April 30, 2021, 11:59pm.
This is a **optional Makeup Assignment**. If you choose to hand it in by the deadline, it will be graded like a normal homework assignment. If its grade is better than your worst homework grade then it will replace that grade.

Submission is to your **private GitHub repository**.

2.1 Numerical solution of the plate capacitor potential (30 points)

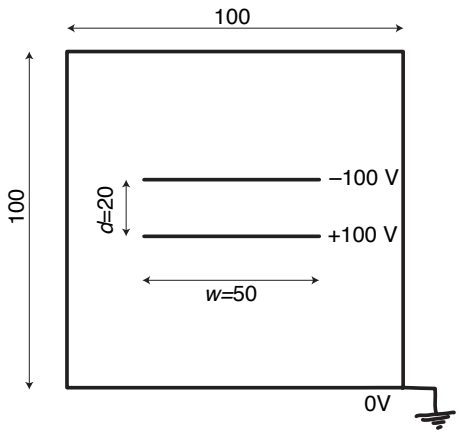

Figure 1: Schematic of the capacitor in the grounded box problem (2D). The capacitor is centered in the box. The capacitor plates are thin, conducting sheets of metal that are held at potential +100 V and -100 V, respectively.

Compute the electrostatic potential of a plate capacitor in a grounded

box, shown in Fig. 1. Solve the problem in 2D. The square box has sides of length 100. Consider the capacitor plates to be conducting, very thin sheets of metal. The plates are held at an electric potential of +100 V and -100 V as shown in Fig. 1. The plates are $w = 50$ units wide and located at a distance $d = 20$. The capacitor is centered inside the box.

Solve the problem on a 100×100 grid where the grid spacing is $\Delta x = \Delta y = 1$. Consider the thin capacitor plates to be of thickness 1.

Submit your code as a python module `capacitor.py`. In particular, you should have the following functions:

`set_boundaries(Phi)` Given the potential on the lattice, enforce the boundary conditions by setting the appropriate lattice points to the prescribed values.

`calculate_phi_capacitor(Max_iter=10000, tol=1e-3)` This function should calculate the potential Φ on the lattice. It needs to take at least the two keyword arguments `Max_iter` to set the maximum number of iterations and `tol` to set the convergence criterion.

You must implement a convergence check: consider the solution converged when the Frobenius norm

$$||\Phi^{\text{new}} - \Phi^{\text{old}}|| := \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}\left|\Phi_{ij}^{\text{new}} - \Phi_{ij}^{\text{old}}\right|^2} \tag{1}$$

is less than the set tolerance `tol`. Report the number of iterations that were required. Also report if convergence was not achieved within `Max_iter` iterations.

The function *must* return the potential Φ as a numpy array of shape $(100, 100)$.

It must be possible to run these functions as

```
import capacitor
Phi = capacitor.calculate_phi_capacitor(Max_iter=2000, tol=1)
```

Start from the skeleton code in `capacitor.py` to ensure that your functions adhere to the prescribed interface.[1]

Specifically, you need to fullfil the following objectives:

(a) Your code must produce correct results, as tested with the included tests. You can run these tests yourself with

```
pytest -v
```

(in the same directory as your `capacitor.py`). **[20 points]**

(b) Converge your results to `tol = 1e-3`. Include in your submission:

a) Report how many iterations you required for convergence. Simply write the number in a *textfile* `iterations.txt`. The number should be on the first line of the file. **[4 points]**

b) Make a 3D plot of the potential and name it `capacitor_potential_3d.png` (you can use `capacitor.plot_phi(Phi)`) **[3 points]**

c) Make a 2D plot of the initial potential (boundary conditions) and of the final converged solution and name it `capacitor_potential_2d.png` (you can use `capacitor.plot_panel(Phi)`) **[3 points]**

Note: You don't have to submit any notebooks or written text. It will be sufficient to submit

- `capacitor.py`
- `iterations.txt`
- `capacitor_potential_2d.png` and `capacitor_potential_3d.png` (these are the default filenames when using the functions in `capacitor.py`).

[1] The skeleton code contains a optimized function `Laplace_Gauss_Seidel_odd_even()` which replaces the Gauss-Seidel code from the lecture. It runs 50 times faster so you are advised to use it... but you may use the slower code, if you prefer. The original code (as developed in class) is in the function `Laplace_Gauss_Seidel()`.