8 — PHY 494: Homework assignment (20 points total)

Due Saturday, April 16, 2016, 5pm.

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

Enter the repository and run the script `scripts/update.sh` (replace *YourGitHubUsername* with your GitHub username):

```
cd assignments-2016-YourGitHubUsername
bash ./scripts/update.sh
```

It should create three subdirectories[1] `assignment_08/Submission`, `assignment_08/Grade`, and `assignment_08/Work` and also pull in the PDF of the assignment and an additional file.

To submit your assignment, commit and push Python code and Jupyter notebook inside the `assignment_08/Submission` directory. *Commit any other additional files exactly as required in the problems.*

Failure to adhere to the following requirements may lead to homework being returned ungraded with 0 points for the problem.

- Homeworks must be legible and intelligible (write complete English sentences when you are asked to explain or describe).
- If you are required to submit code, it must run without errors under Python 3 with the anaconda distribution.
- If you are required to submit data then the data files must be formatted exactly as required to be machine parseable.

8.1 Numerical solution of the plate capacitor potential (20 points)

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. The plates are $w = 50$ units wide and located at a distance $d = 20$. The capacitor is centered inside the box.

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.

[1] If the script fails, file an issue in the Issue Tracker for PHY494-assignments-skeleton and just create the directories manually.

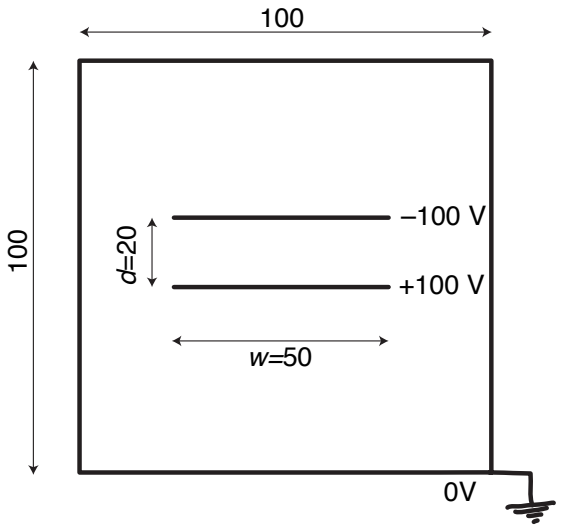

Figure 1: Schematic of the capacitor in the grounded box problem (2D). The capacitor is centered in the box. The capacitor plates are thin, conducting sheets of metal that are held at potential +100 V and -100 V, respectively.

`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 $||\Phi^{\text{new}} - \Phi^{\text{old}}||$ is less than the set tolerance `tol`.[2] 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 dimensions $(100, 100)$.

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.[3]

[2] See the code from class for how to implement the convergence check.

[3] The skeleton code contains a optimized function `Laplace_Jacobi()` 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()`.

Specifically, you need to fullfil the following objectives:

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

```
nosetests -v test_capacitor.py
```

(in the same directory as your `capacitor.py`). **[15 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` **[3 points]**

b) Make a 3D plot of the potential (you can use `capacitor.plot_phi(Phi)`) **[1 points]**

c) Make a 2D plot of the initial potential (boundary conditions) and of the final converged solution (you can use `capacitor.plot_panel(Phi)`) **[1 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.pdf` and `capacitor_potential_3d.pdf` (these are the default filenames when using the functions in `capacitor.py`).