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

Due Saturday, April 23, 2016, 5pm.[1]

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[2] `assignment_09/Submission`, `assignment_09/Grade`, and `assignment_09/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_09/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.

9.1 Numerical solution of Laplace's equation (20 points)

Compute the electrostatic potential of two plates in a grounded box, shown in Fig. 1. Solve the problem in 2D. The square box has sides of length 140. Consider the capacitor plates to be conducting sheets of metal of thickness 10. 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 $d = 10$. The top plate contains a hole at the center with radius $r = 10$. The whole "capacitor" is centered inside the box.

[1] Corrections and updates to this problem:

2016-04-18 correction: The hole in the negatively charged plate of the capacitor has a radius $r = 10$ (and not 20).

2016-04-20 corrections: (1) updated tests, (2) fixed what the functions should be called (now `calculate_potential()` instead of `calculate_electrostatics()`) and fixed the call signature of `calculate_chargedensity(Phi)`.

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

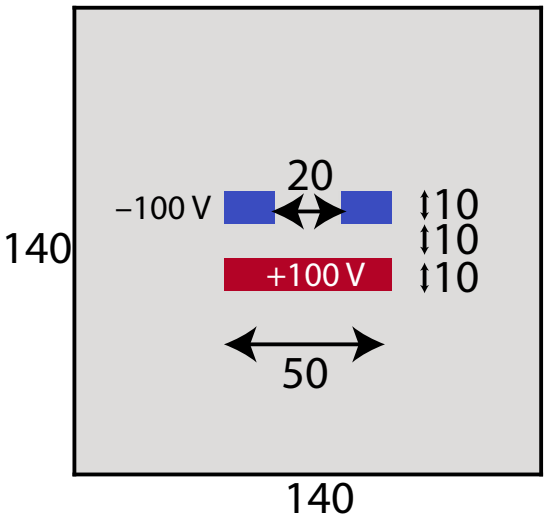

Figure 1: Schematic of the electrostatic problem.

Submit your code as a python module `electrostatics.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_potential(Nmax=140, Max_iter=30000, tol=1e-6)` This function should calculate the potential Φ on the lattice. It needs to take at least the three keyword arguments: `Nmax` to set the lattice size, `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`.[3] 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 $(Nmax, Nmax)$.

`calculate_chargedensity(Phi, Delta=1)` This function should calculate the charge-density, given the potential Φ (and the lattice spacing Δ).

It must be possible to run these functions as

```
import electrostatics
```

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

```
Phi = electrostatics.calculate_potential(Nmax=140, Max_iter=30000, tol=1e-6)
rho = electrostatics.calculate_chargedensity(Phi)
```

Start from the skeleton code in `electrostatics.py` to ensure that your functions adhere to the prescribed interface.[4]

Specifically, you need to fullfil the following objectives:

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

```
nosetests -v test_electrostatics.py
```

(in the same directory as your `electrostatics.py`). **[13 points]**

(b) Converge your results to `tol = 1e-6`. 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 `electrostatics.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 `electrostatics.plot_panel(Phi)`) **[1 points]**

d) Make a 2D plot of the charge density ρ, derived from the converged potential Φ. You can use the provided `electrostatics.plot_chargedensity(rho, filename)` function:

```
rho = electrostatics.calculate_chargedensity(Phi)
electrostatics.plot_chargedensity(rho, "electrostatics_chargedensity.pdf")
```

[2 points]

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

- `electrostatics.py`
- `iterations.txt`
- `electrostatics_potential_2d.pdf` and `electrostatics_potential_3d.pdf` (these are the default filenames when using the functions in `electrostatics.py`) and `electrostatics_chargedensity.pdf`.

[4] The skeleton code contains the optimized function `Laplace_Gauss_Seidel_odd_even()`, which you should use.