FYS-4096 Computational Physics: Exercise 5

Ilkka Kylänpää

Week 6

General:

- You should have access to `ex5_help` on teacher's help file repo.
- There you will find files `runge_kutta.py`, `spline_class.py`, and `num_calculus.py` that are needed or can be useful in this exercise.
- **Deadline always before noon on Monday the following week!** That is, for these problems before noon on Monday February 15.
- Better to send unfinished in time, than finished, but past deadline.

Problem 1: Fourth order Runge-Kutta

- Download `runge_kutta.py` from teacher's repo.
- Complete the function for fourth order Runge-Kutta.
- Running the file as `python3 runge_kutta.py` should show you three essentially identical plots in one figure. (If not, then your implementation is not correct).
- The other two plots are obtained by scipy's ordinary differential equation solvers `solve_ivp` and `odeint`.
- In addition to the visual test, compare your result with one of the scipy's solvers. For example, print the maximum absolute difference (`np.amax(abs(...))`) of your solutions, etc.
- Add brief comments into the file `runge_kutta.py`, and **look into the usage of scipy's solvers**.

Problem 2: Trajectory of a charged particle influenced by both electric and magnetic field

- The Lorentz force on a charged particle is $\mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B})$, where $\mathbf{E}$ is electric field and $\mathbf{B}$ magnetic field.
- Derive general equation of motion for the charged particle under the influence of the Lorentz force.
- Code these equations in Python3 in a form that can be solved by ordinary differential equation solver, i.e., $d\mathbf{x}/dt = \mathbf{f}(\mathbf{x}(t), t)$.
- Then, consider a case where $\mathbf{E} = E\hat{\mathbf{i}}$ and $\mathbf{B} = B\hat{\mathbf{j}}$, and solve the ordinary differential equation using any of the methods in Problem 1. As initial conditions take $t_0 = 0$, $\mathbf{v}_0 = (0.1, 0.1, 0.1)$, and $\mathbf{r}_0 = (0.0, 0.0, 0.0)$ (arbitrary units). Also, use $qE/m = 0.05$ and $qB/m = 4.0$.
- Plot the trajectory of the particle for $t \in [0, 5]$.
  - For plotting you could use, for example,
    ```
    from mpl_toolkits.mplot3d import Axes3D
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.plot(x,y,z)
    ```
- At time $t = 5$ the coordinates should be roughly $(0.01, 0.5, 0.1)$. What is the velocity vector at time $t = 5$?

Problem 3: Poisson equation and relaxation methods

$$\frac{\partial^2 \Phi(x,y)}{\partial x^2} + \frac{\partial^2 \Phi(x,y)}{\partial y^2} = -\frac{\rho(x,y)}{\varepsilon_0} \tag{1}$$

- Write a program that solves the Poisson equation in 2D using the Jacobi, Gauss-Seidel, and SOR update schemes. Use same grid spacing for both dimensions, and the maximum step size for which the method is still stable.
- Use a tolerance option for termination of the update when good enough accuracy is obtained.
- Compare the results and convergence properties of the three relaxation schemes for Laplace equation (i.e. $\rho = 0$) with boundary conditions

$$\Phi(x=0,y) = \Phi_0, \quad \text{and} \tag{2}$$
$$\Phi(x>0, y=L_y) = \Phi(x>0, y=0) = \Phi(x=L_x, y) = 0, \tag{3}$$

  where $\Phi_0 = 1$, and $L_x = L_y = 1$.
- For SOR method use $\omega = 1.8$.
- If you use meshgrid with `indexing='ij'`, then for plotting you could use, for example,
```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_wireframe(X,Y,Phi,rstride=1,cstride=1)
```
  Otherwise use `X.T` and `Y.T` in plotting.
- **Notice that once you have coded the Jacobi method, the other two need only simple modifications to that. Also, remember to have the charge density in your equations, although in this problem it will be set to zero in the tests.**

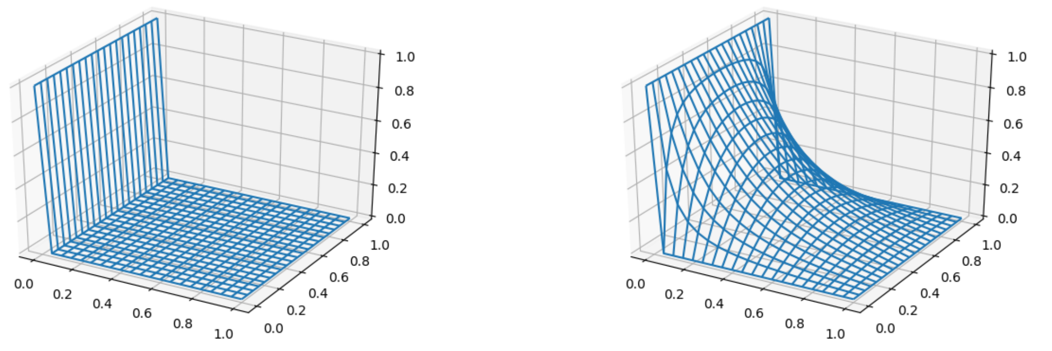

Figure 1: Figure related to Problem 3. On the left: initial condition. One the right: solution.

Problem 4: Two capacitor plates

- Consider a 2D case of two capacitor plates (third dimension considered infinite).
- The plates in 2D are described by vertical lines of length $L = 1$ locating at $x = \pm 0.3$.
- The plates are held at potentials $\Phi = 1$ and $\Phi = -1$, and they are inside a square boundary ($L^2_{\text{box}} = 4$) that is held at zero potential ($\Phi = 0$). See Fig. 2.
- When formulating the problem be careful in including the plates in the grid.
- Solve the potential profile $\Phi(x, y)$ inside the square and plot the result. Do not use excessively fine grid.
- Visualize the electric field inside the square using matplotlib's quiver function. (Remember that $\mathbf{E} = -\nabla\Phi$.)

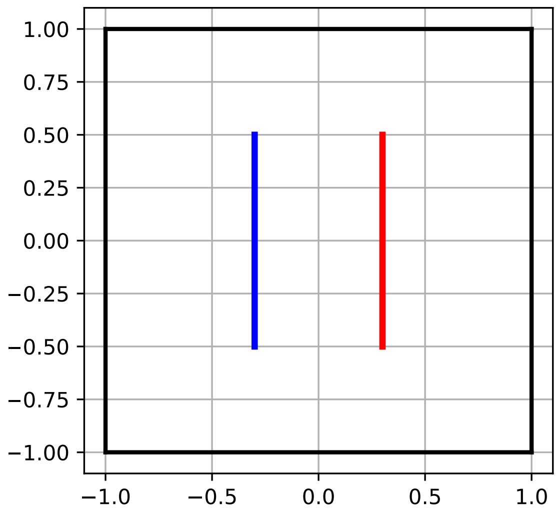

Figure 2: Figure related to Problem 4.

Returning your exercise

1. Make a new folder "exercise5" to your existing Computational Physics repo.
2. Create a file solvedProblems.txt at the root of your "exercise5" folder. Inside it, write a comma separated list of problems you have solved, e.g., 1,2,3x.
3. Make sure all your source files are under version control and push them to GitLab.
4. Push your commits (and the possible tags) to GitLab before noon on Monday February 15:
```
git push --all && git push --tags
```