Electrostatic and Potential Field Graphing Utility with GUI

Dylan Jake

1 Introduction

This code defines a graphical user interface (GUI) using Tkinter to interactively visualize electrostatic potentials and electric fields for various charge distributions. The user can choose from different shapes, specify dimensions and charge densities, and toggle between visualizing the electrostatic potential and the electric field.

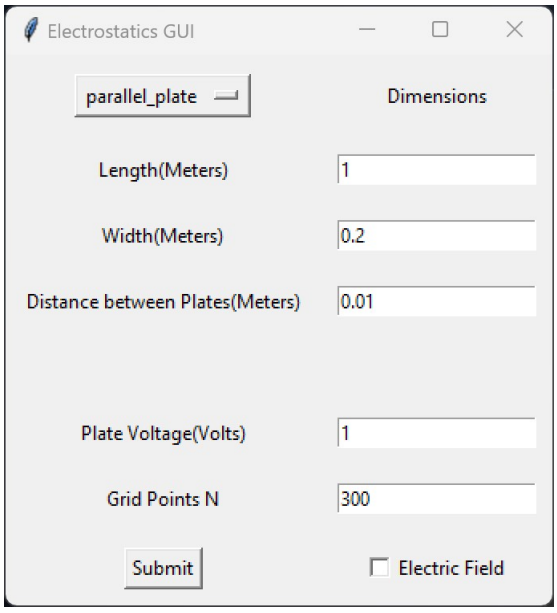

Figure 1: The GUI upon launching the python program.

2 Implementation

First, the necessary libraries are imported:

- numpy for numerical calculations
- scipy.sparse and scipy.sparse.linalg for working with sparse matrices and solving linear equations
- matplotlib.pyplot for plotting the results
- tkinter and tkinter.filedialog for creating the GUI and handling file inputs

The code defines several functions:

- `get_range`: Given a shape and its dimensions, this function returns the range for the x and y axes in the plot.
- `parallel_plate_rho`, `rho_rect`, `rho_circle`, `rho_annulus`, `rho_line_charge`, `rho_point_charge`, and `read_custom_rho_file`: These functions define the charge density (rho) for different shapes.
- `plot_graph_with_input`: This is the main function that generates the plots. It takes the shape, dimensions, whether to display electric field or not, grid points N, initial voltage V0 (for parallel plate capacitor only), and custom_rho_file (for custom charge distribution). The function calculates the potential and electric field for the given charge distribution using the Conjugate Gradient (CG) method from the scipy.sparse.linalg library. It then plots the scalar potential or electric field depending on the user's choice.
- The conjugate gradient (CG) method is an iterative algorithm used to solve systems of linear equations, particularly those arising from the discretization of partial differential equations. It is especially useful when the matrix of coefficients is symmetric and positive definite, as it converges faster than other methods in these cases. The main idea behind the CG method is to generate a sequence of search directions that are conjugate with each other, which leads to much faster convergence when compared so methods such as relaxation. In the given code, the cg function from scipy.sparse.linalg is used to solve the system of linear equations, which models the scalar potential in electrostatic problems. The Laplacian operator that is used in the system is made from the sparse diags function.
- `submit`: This function reads input from the GUI and calls `plot_graph_with_input` with the appropriate arguments.
- `update_input_fields`: This function updates the input fields and labels in the GUI based on the selected shape.

3 User Guide

The program itself is made to be as user-friendly as possible. No external hardware or imported libraries should be required, as Numpy, Matplotlib, Tkinter, and Scipy all should be installed on the raspberry pi. Once the program is run in the python interpreter, the charge distribution must be selected. By default the shape is a parallel plate shape will pre-filled dimensions and charge values that can be changed to fit the user's exact needs. The program will scan for any numerical issues with the input and prompt the user to fix the issues in the GUI itself. Once a shape and its dimensions are properly selected, the user can change the grid points N, which will increase or decrease the resolution and accuracy of the graphs, and pick whether they would like a graph of the scalar potential or the electric field. Once the pop-up window of the graph appears, the user can mouse over and click on any point in the graph to see the value of the field or potential at that point displayed in the GUI as well. There is a color bar to display the general range of data for the plot on the side. The user can save the graph with the built-in save function in the window. The program is ready to re-plot once the old graph window is closed. The custom shape allows the user to select a space-separated file of values that can be experimentally found. The values should be in the format "X Y Charge", once the file is selected in the pop-up window that appears when the submit button is pressed while having the custom shape selected, the program will graph the potential or field just as you would expect. Due to the fact that this program uses the conjugate gradient method for solving these differential equation systems, the program is quite quick when the grid points are below 500. It is possible for the grid points to be increased past that, but the computation time will increase for very little noticeable return. There are two provided custom data files that can be loaded into the custom shape, or the user can generate their own text file with space-separated values of X, Y, and charge if so desired.

4 Results

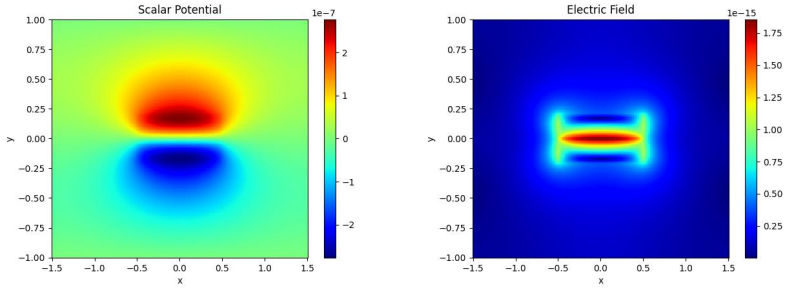

Figure 2: Example outputs of the program for the parallel plate shape.

Uses default values of length = 1 meter, width = 0.2 meters, distance between plates = 0.01 meters, plate voltage = 1 volt. The grid points are set to the default 300 points for a great mix of detail and program efficiency.

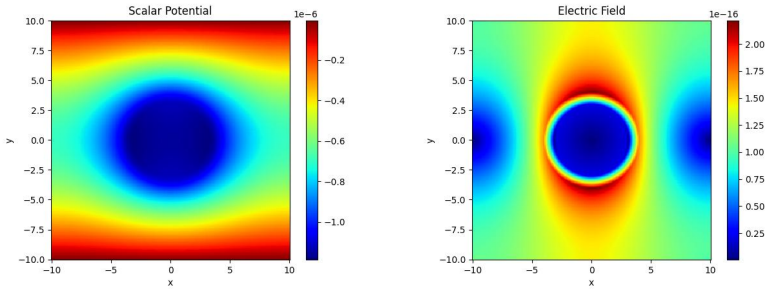

Figure 3: Example output of the program for the annulus shape.

Uses dimensions of outer radius = 4 meters, inner radius = 3 meters, charge density = 1e-9 $1\,\mathrm{C\,m^{-2}}$. The grid points are set to the default 300 points.

5 Results cont.

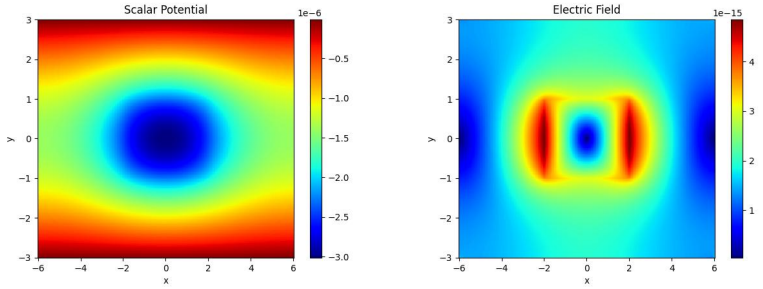

Figure 4: Example outputs of the program for the rect shape.

Uses dimensions of width = 4 meters, height = 2 meters, charge density = 1e-9 1 C m^{-2}. The grid points are set to the default 300 points.

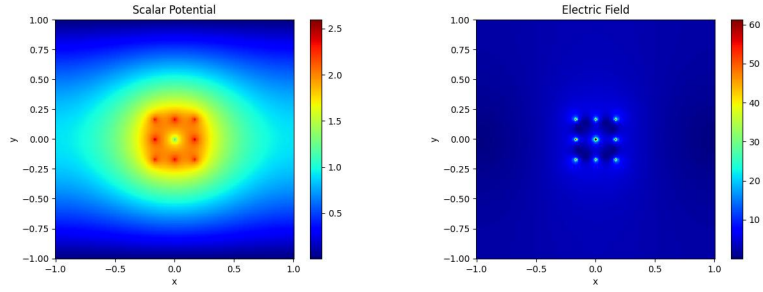

Figure 5: Example outputs of the program for the custom shape.

Uses the provided customdata2.txt file that was provided. The grid points are set to the default 300 points.