Visualizing Particle Trajectories in Electric Fields

Arjun Chandra

April 2024

1 Problem Statement

The central problem that was addressed in my project was that of simulating the electric field generated by a distribution of stationary charges and visualizing the trajectory of a test charge placed in the field at an arbitrary location. The basis for this simulation is the following differential equation which describes the motion of the test charge:

$$dx = E(x)dt \tag{1}$$

The charge distribution in the simulation is discrete. Therefore, the following equation was used to calculate the electric field in Eq. (1):

$$\vec{E} = \sum_{q_i \in \mathcal{D}} \frac{1}{4\pi\epsilon_0} \frac{q_i}{\mathcal{R}^3} \vec{\mathcal{R}} \tag{2}$$

Eq. (2) is a sum of the electric field contributions over all of the charges in the distribution. There were slight modifications to Eq. (2) in the actual implementation which will be described in the next section. Solving this computational problem was inspired by a recently developed family of generative AI models known as Poisson Flow Generative Models (PFGM) which take inspiration from electrodynamics. The specifics of the model are not important, but the main inspiration which influenced my project was to embed the charge distribution on the x-axis while the test charge can be placed anywhere in the x-y plane. Having the test charge reside in an additional dimension over the charge distribution allows for more interesting trajectories and is used in PFGM to avoid the problem of mode collapse.

2 Computational Methods

In order to simulate the trajectory of the particle according Eq (1), I used the Euler algorithm. In particular, a time-step dt was chosen so that at each iteration of the algorithm, the electric field at the present position of the test charge is calculated according to Eq. (2), and the position of the test charge is subsequently evolved along the direction of the electric field with its magnitude scaled by dt. In practice, there were several modifications that needed to be made to the algorithm in order to produce a physically accurate simulation. First, in calculating the electric field, all constants and the charge q in Eq. (2) are set to 1 since including them does not influence the direction of the net electric field (assuming all distribution charges have the same charge q) but does put the electric field on a scale that is orders of magnitude different than the OpenGL window scale. This does not, however, limit the simulation in only allowing for distribution charges of equal (unit) magnitude; the following section describes how the user can interact with the program to "stack" charges, effectively allowing for charges of varying magnitude.

Another modification to the algorithm needed to be made because the electric field approaches infinity as the test charge approaches the distribution charges ($\lim_{\mathcal{R}\to 0} \frac{1}{\mathcal{R}^3} = \infty$), which is problematic because even after scaling the field by the time-step dt, the test charge takes a step of essentially infinite size and disappears from the OpenGL window. Several approaches were considered in order to remedy this. One approach was to scale the time-step dt by $\mathcal{R}^3$ in order to cancel the factor $\frac{1}{\mathcal{R}^3}$ when it causes the field to "blow up". However, determining precisely when to scale the time step and by which of factor of $\frac{1}{\mathcal{R}^3}$ in the sum in Eq. (2) was challenging and resulted in nonphysical simulations. In the end, the approach taken was to effectively enforce a ceiling on the step size according to the following procedure: if $|E_x| > 1$ or $|E_y| > 1$, set the electric field to $\alpha\hat{E}$ where α is a constant which can be adjusted to change the maximum step size of the particle. In the source code $\alpha = 0.2$. This approach was successful because using a unit vector for $\vec{E}$ preserves the direction of the step but prevents the step size from becoming infinite. With these modifications, the Euler algorithm achieves a physically accurate simulation.

3 Code Developed

The code developed for the simulation is contained entirely within the file titled "term_project_driver.c". In writing the code, the "gbounce.c" file that was distributed in the course software was taken advantage of to draw the charges in the OpenGL window and provided a framework for displaying the time evolution of the test charge. The code also gathers parameters of the simulation from the user at the start of execution. Specifically, the user inputs the number of the charges in the distribution, the locations of each charge in the distribution, and the location of the test charge. Dynamic memory allocation is used to allow for a virtually unlimited number of distribution charges. Multiple charges can also be placed at the same location or overlap in order to effectively create a single charge with a larger magnitude. The code is written to assume well-formatted inputs; if by mistake the inputs are not well-formatted the code can simply be re-executed. Once the user has entered in the parameters, the OpenGL Window will appear and the window should be left-clicked to open the menu and start the simulation. The simulation will automatically end when the test charge reaches the x-axis, but the OpenGL window will not be closed until the user chooses to exit. In the working directory where the source code is located, enter the following command to compile the source and create the executable:

```
cc term_project_driver.c -o term_project_driver -lm -lglut -lGLU -lGL -lXext -lX11
```

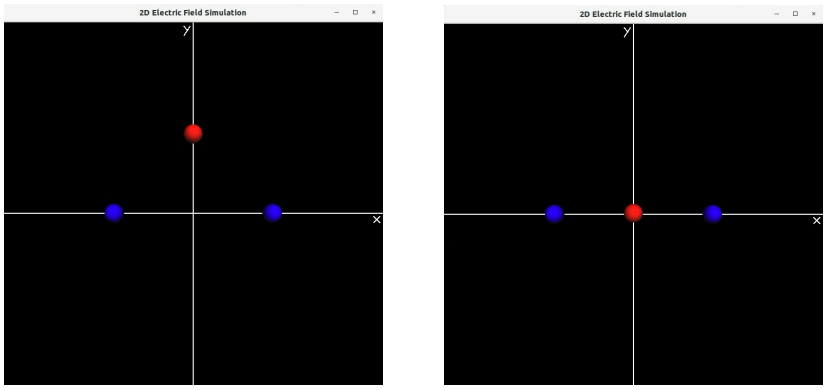
The executable will be created in the working directory and can be run with "./term_project_driver".

4 Results

There are many interesting results that can obtained from running the code with different parameters and charge distributions. In this section, some of these results will be discussed along with the parameters used to obtain them so that they can be reproduced. The first interesting result is that the test charge will not always converge to one of the charges in the distribution. For example, consider the following set of parameters:

$$\begin{gathered} \text{Number of Distribution Charges: } n = 2 \\ \text{Distribution Charge Locations: } x = -4, x = 4 \\ \text{Test Charge Location: } x = 0, y = 4 \end{gathered} \tag{3}$$

The results of running the simulation with these parameters are shown on the next page.



(a) Initial State (b) Final State

Figure 1: Initial and Final Simulation States for Parameters in (3)

These results are interesting because they offer insight into how the electric field can be used in generative AI. If we imagine that the blue charges are data points in the known distribution and the red test charge is a randomly sampled point in the x-y plane, we can see that time-evolving the test charge along the electric field generated a new point in the distribution that didn't exist before (at $x = 0$). Of course the details are much more involved than this, but it's a nice example to illustrate the inspiration for PFGM.

Another interesting result is that we can use discrete charge locations to approximate continuous distributions and qualitatively explore their properties. For example, the following parameters can be used to approximate a Gaussian distribution (terms in parenthesis indicate the number of stacked charges at each location):

$$\begin{gathered}\text{Number of Distribution Charges: } n = 19 \\ \text{Distribution Charge Locations: } x = -3\ (1x),\ x = -2\ (2x),\ x = -1\ (4x), \\ x = 0\ (5x),\ x = 1\ (4x),\ x = 2\ (2x),\ x = 3\ (1x)\end{gathered} \tag{4}$$

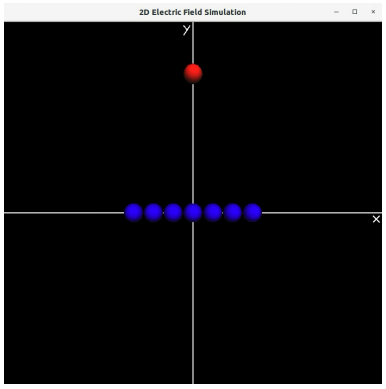


Figure 2: Gaussian Distribution (Charges Are Stacked)

Using these parameters, if we randomly decide where to place the test charge, the test charge will have a tendency to converge close to the mean of the distribution ($x = 0$). There are many other distributions that can be explored this way and it is an interesting unforeseen application of the simulation. In closing, I

recommend that the user experiments with the parameters of the simulation to obtain new results which they find to be interesting.