Capstone Project: N-Particle Interaction in Electric and Magnetic Fields

Isabella Deutsch and Weizhi (Tom) Wang

Written by Isabella Deutsch

Abstract

Particle interactions without the presence of external fields are characterized by Coulomb's Law, depending on the distance between the particles, making it relatively simple to visualize their motion. However, when adding in external electric and magnetic fields into play, understanding particle motion intuitively becomes more difficult, now considering the Lorentz force and its effects. Through calculation, animation, and incorporating user input, our resulting simulation animates the motion of N particles interacting in the presence of these external fields, with the aim to help students better visualize particle motion in more complex cases than typically showcased.

Motivation

The initial motivation for this project was to construct a sort of N-body problem for particles interacting with one another and model it through animation. This would require analyzing the motion of particles in a case where the electric fields generated by each particle were constantly changing. As this idea of constantly updating the field came into fruition, we decided it would be interesting to then incorporate the ability to insert external electric and magnetic fields as part of the problem. The ultimate goal then, was to create a simulation similar to ones like the *Charges and Fields* Phet Simulation (University of Colorado Boulder) but in situations of electrodynamics, and one where students could actively visualize particle motion rather than just place particles and view the resultant fields.

Methods

The project developed in a set of 3 stages: calculation, animation, and finally taking in user input. Isabella mainly worked on calculations (therefore will be the main focus of my report), Tom worked on animations, and both of us worked on user inputs and revising.

Calculations

In the case of N particles interacting with each other, to analyze a single particle's motion we take the summation of the net forces given by Coulomb's Law,

$$F = \frac{k\,q_1 q_2}{r^2}\,,\ E = \frac{k\,q_1}{r^2}.$$

However, if we add external fields into the equation, we also then have to incorporate the Lorentz force equation,

$$F = q\,(E + v \times B).$$

These equations became the basis of our calculation section. We first created a function (get_fields2) that loops through each particle and calculates the sum of Coulombic fields from every surrounding particle on the desired particle. Then, we created a separate function (get_function and find_pos_vel) that called these fields, as well as any external fields, and calculated the net force, and output an array of initial velocities and accelerations for each particle. Using solve_ivp within the scipy package, we used the outputs of find_pos_vel as the function input into solve_ivp, as well as the initial parameters inputted by the user. Solve_ivp then solved for the subsequent positions and velocities of each particle over a specified time interval. The initial challenge in this process was making sure that the fields were constantly updating in step with solve_ivp, so that the positions/new initial inputs were correct. Thus why we called get_fields2 within the find_pos_vel function so that after each timestep the next fields generated would be based off of the new initial conditions just calculated. Additionally, originally we had never seen a scenario for which solve_ivp solves for the positions of *multiple* particles at once—we wanted to generalize to N particles being solved all at once. This prompted us to create a class called Particles, so that we could store the positions/velocities/etc of all N particles within one access point. We could then still run solve_ivp for each particle in a way that got us the correct output without any accidental error, but still was happening simultaneously for all desired particles.

Additionally, it's important to note that we did not include the effects of induced fields within this project, as after consulting Professor Jazaeri, we found that their impact on the motion of each particle was negligible, and the calculations required to find the effects were beyond the time period of the project and scope of the course. Therefore, we believe our position and velocity calculations to be a good approximation of what the motion of particles would look like in electric/magnetic fields, but could still be improved long term.

Animation

The animation portion of the project was largely based around the pyglet library, which is commonly used for game development, which Tom primarily worked on. We defined the camera class, which can take in our updating positions in the update_particles function and then draw a circle at that position that has size and color dependent on the particle's mass and charge. Camera has the ability to be set at multiple different viewpoints, as pyglet largely works in 2D but we wanted the possibility to show all 3 dimensions of motion. Then, we created another class called Simulator, which is where we actually animate our particles and place our cameras. We then together ran simulation tests to make sure our fields were working, then a quick animation test for between 2 and 5 particles just to make sure things were running smoothly.

Refining and User Input

After we got the animation down, we noticed that since we had our simulation break when particles collided, for more than two particles the physical animation was actually quite short. To extend this a bit longer, we decided that I would figure out how to incorporate a form of a weak force when particles came within a certain distance of one another into the calculations section. We thus created the get_fields_collisions function, which at a distance of five or less we added in a sort of weak field to push particles away from each other, which we defined as

$$F_{weak} = \frac{k\,p\,q_1^{\,2}|q_2|}{r^4},\ E_{weak} = \frac{k\,p\,q_1|q_2|}{r^4}$$

Where p is the vector of position of the first particle minus position of the second (mainly this was incorporated for directional reasons). With this revision to our program, we found that we could have our simulations run for much longer periods of time, especially in scenarios of more than two particles. However, we noticed that for some reason this force seemed to overpower the electric fields generated by the particles, and so the weak force often did more harm than good in situations with no external fields. However, when tested with the presence of external fields, this worked out better as we were able to reduce the effect of the force (see example 1 in Testing and Results for more information). We were unable to figure out how to deal with this issue due to time constraints, but if continued figuring out how to create a more stable and effective weak force would be a top priority (during this process we found a force proportional to $\frac{1}{r^3}$ did not prevent impact, and so would start from there).

We also wanted to be able to incorporate user input into our simulation. We created the user_inputs file, in which we created two functions, get_user_inputs, which relies on inputting directly into the terminal (which could potentially cause error if done incorrectly), and get_user_inputs2, which reads the inputs from a txt file so that users can save and properly input initial conditions. Within the user_inputs file, we also defined a compute function, which condenses the entire process of our simulation into one function that we can run, and then animate that result. With everything figured out, our simulation was ready to test.

Testing and Results

While theoretically our simulation can be tested for any number of particles, initial positions, velocities, and fields, we tried to choose situations to test that weren't completely random (see Appendix for the values of these inputs). To test different aspects of our code, we did three different tests, all of which for simplicity had all particles start with 0 initial velocity (see Appendix for links to video results):

1) *Two particle collision using get_fields2 versus get_fields_collisions*

This test was implemented to demonstrate the difference in simulation length when we incorporated a weak force into simple particle collisions. When testing the collision of two particles of the same mass and equal/opposite charge in an external electric field using get_fields2 (no weak force), the simulation ran fairly quickly, and crashed after the particles collided, as solve_ivp breaks when the distance between the two is 0. Then, we tried the same conditions again using get_fields_collisions, which included the weak force (see Refining and User Input section). In this scenario, the particles did not collide, but instead when at a distance of less than or equal to 2, experienced a weak force that propelled them away from each other, creating indefinite oscillatory motion. The results show that incorporating a form of weak force makes the simulation last significantly longer for two particles, and thus if the weak force was perfected it would be even more useful for N particle collisions. However, as mentioned in the Refining section, this weak force long term would need to be altered in order to work for all situations (even ones with no external fields slowing things down).

2) *Five particles with no external fields*

This test was mainly to see how our functions worked out with larger numbers of particles, with the ideal being that if we had the capabilities to do so runtime wise, we could simulate up to N particle interactions. The result shows that for five particles of the same mass and alternating sign charge placed in a pentagon shape, that initially all five start to move towards the center, but as similar charges get closer and repel, as well as differing charges attract, we get a unique swirling pattern that leads to no collisions. This was likely also due to the initial positioning/angles at which each particle was placed, as creating a shape like a box would have more likely led to a direct hit. Looking at how each particle seems to be affected by all those around it based on distance, this test affirmed that our simulation was working for larger numbers of particles (basically that the get_fields2 looping was properly done). This situation also demonstrated the strength of the electric fields generated by each particle, as the remaining two particles after the others flew off still managed to be attracted to one another despite a larger distance.

3) *Three particles with no external fields, an electric field, and a magnetic field*

This final test was implemented to show the differences between particle interactions in the presence of no external fields versus with an external electric or magnetic field. For simplicity, in both the electric and magnetic cases we made the field the same ($E, B = (0, 10x^2, 0)$), knowing that the effects should look completely different. For no external fields, the two positive particles (of the three, two of equal and opposite charge and equal mass, the third half the mass and positive; see Appendix for more information) experienced slight initial attraction towards the one negative, but the positive charge with the larger mass ultimately repelled the other positive charge and created a stronger attraction with the negative, rotating around one another as the pair moved away from the lone charge. When we added in an electric field in the y-direction, we see a similar start to the simulation, but once the pair is far enough away from the other charge, the electric field's strength pulls them to the right side of the screen, and they eventually diverge from one another. Finally, we added in a magnetic field of the same magnitude in the y-direction, which created a unique situation. Initially the simulation starts out the same, just a little slower. Then, the influence of the magnetic field causes the three particles to stay relatively close, and at one point the heavier of the positive charges breaks away and moves to the left in a sort of oscillatory manner, while the other two move to the right in a similar oscillation, until the charge with less mass begins to pick up more speed. While this can be difficult to interpret, we can clearly see that our magnetic and electric fields, despite having the same magnitude, act completely differently, showing that our inputs and how external fields are taken in are correct. However, if we wanted to analyze each of their individual effects a bit better, we could have played around with and switched the camera positioning a bit more to get a better sense of what's going on 3 dimensionally (as sometimes 2D can be a bit misleading).

Conclusions

Overall, we believe that the resulting simulation decently models particle interactions in electric and magnetic fields, despite not including the effects of induced fields (and even if we don't include weak forces for collisions). We did notice while choosing examples that without specific electric/magnetic fields and initial conditions, this simulation may be less intuitive for beginners, but would work well for those with a good understanding of the math behind the motion who wanted visualization. In the future, I would find it even more interesting to try and incorporate induced field effects into the system, as well as going deeper into specific situations such as electromagnetic radiation. Additionally, since we ended up running out of time with our weak force working properly for situations without external fields, I would definitely want to revise that in order to create an overall more effective and longer duration simulation. Professor Jazaeri also mentioned perhaps creating a boundary at the edges of our animation so that we could confine particle motion to just what's on the screen, which I also felt would be a useful inclusion in the future.

References

Physics 77 Module 10 Lectures, ODE solvers

Professor Jazaeri, guidance on external field incorporation

https://pyglet.readthedocs.io/en/latest/programming_guide/quickstart.html, understanding pyglet

https://phet.colorado.edu/sims/html/charges-and-fields/latest/charges-and-fields_all.html, inspiration behind project, lacking animation but interactive

https://flothesof.github.io/charged-particle-trajectories-E-and-B-fields.html, ODE solving for particles, basic particle trajectories without incorporating changing fields and their effects

Appendix

Inputs used in Testing and Results (inputs.txt file):

2 particle collision (get_fields2 versus get_fields_collisions in the compute function):

```
1000,1000
1,-1
0,200,0,0,-200,0
0,0,0,0,0,0
[((0,0,0),(0,0,-1),(0,1,0))]
lambda x,y,z:(0,-10*(y**2),0,0,0,0)
```

5 particles no fields (get_fields2):

```
1000,1000,1000,1000,1000
1,-1,1,1,-2
0,0,200,0,-200,0,0,200,0,0,-100,-200,0,100,-200
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
[((0,0,0),(-1,0,0),(0,1,0))]
lambda x,y,z:(0,0,0,0,0,0)
```

3 particles no fields (get_fields2):

```
1000,500,1000
1,1,-1
100,0,0,0,100,0,0,0,0
0,0,0,0,0,0,0,0,0
[((0,0,0),(0,0,-1),(0,1,0))]
lambda x,y,z:(0,0,0,0,0,0)
```

3 particles E field (get_fields2):

```
1000,500,1000
1,1,-1
100,0,0,0,100,0,0,0,0
0,0,0,0,0,0,0,0,0
[((0,0,0),(0,0,-1),(0,1,0))]
lambda x,y,z:(0,10*(x**2),0,0,0,0)
```

3 particles B field (get_fields2):

```
1000,500,1000
1,1,-1
100,0,0,0,100,0,0,0,0
0,0,0,0,0,0,0,0,0
[((0,0,0),(0,0,-1),(0,1,0))]
lambda x,y,z:(0,0,0,0,10*(x**2),0)
```

Link to Video Results (two methods in case)

77 Capstone Simulation Videos

https://drive.google.com/drive/folders/1nH0dV8NiMo_A1iFPshVUFmiowvVp4sSm?usp=drive_link