Noah Kasmanoff
nsk367

Homework 5

Computational Physics
November 25, 2019

9.7

link to code

Replacing the second derivative in this equation with its finite-difference approximation, the relaxation-method equation for solving the kinematic equation of gravity on a "grid" derived as

$$\frac{d^2x}{dt^2} = -g => \frac{dx^2}{d^2t} = \frac{x(t+h) - 2x(t) + x(t-h)}{h^2} = -g$$

where h is the designated step size, but in the context of solving for the boundary conditions of this problem, is analagous to grid spacing.

Re-arranging both sides of the equation to solve for the equation of motion, we find

$$x'(t) = \frac{1}{2}[gh^2 + x(t+h) + x(t-h)]$$

Once the initial conditions are set and the code linked above is set to run, we recover the following graph

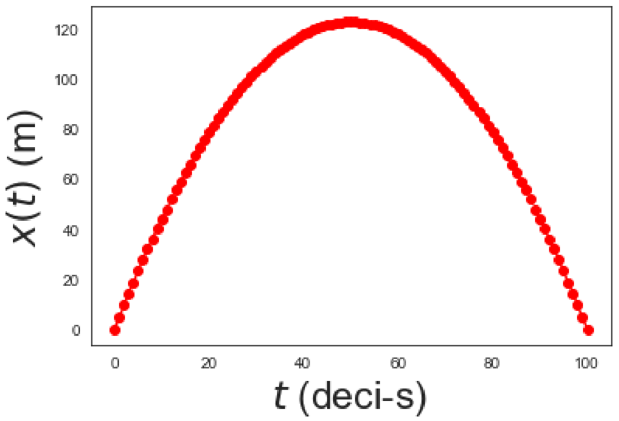

Figure 1: The trajectory of a ball thrown into the air at $t = 0$ and who's initial conditions state must return to the ground at $t = 10$.

Due to the required grid spacing the units of this figure are recorded in tenths of a second, as 100 dece-seconds make up 10 seconds. Regardless, this method proves successful in integrating the trajectory of a thrown ball, and while it did not require an intial velocity such as the shooting method, we can still recover such an initial velocity the ball is thrown by differentiating $x(t)$ with respect to it's first two values. When doing so, we find that the initial velocity is equal to 48.5 meters per second.

Particles on a Grid

link to code

a) The first step to this problem is to transform our point cloud based dataset into a grid based one. This means turning points given as (x,y) coordinates into specific positions on a grid which are then used for the differential equation solving technique when solving for the potential of that group of particles.

To get this started, we apply the "cloud-in-cell" technique, which assigns each point cloud to it's nearest grid points, on a weighted basis where each neighboring grid point receives a certain contribution from the point cloud based on how much overlap they have. Please see figures 2 and 3 below, which demonstrate the transition from point cloud to grid.

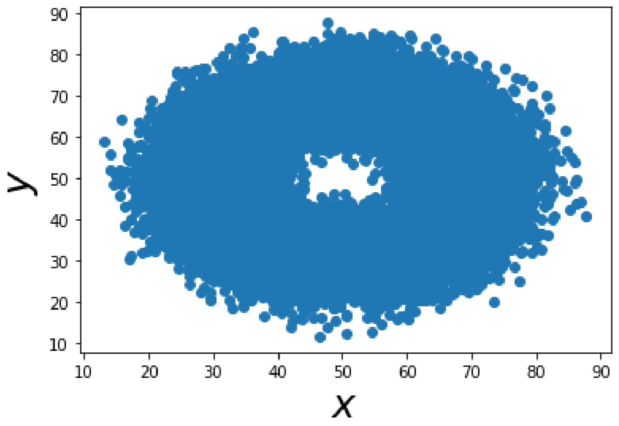

Figure 2: The point cloud representation of the charged particles. One can understand this figure as each particle possessing a certain x y value, and plotted accordingly on a scatter plot.

Next, we seek to represent these particles on a grid, leveraging the cloud-in-cell algorithm.

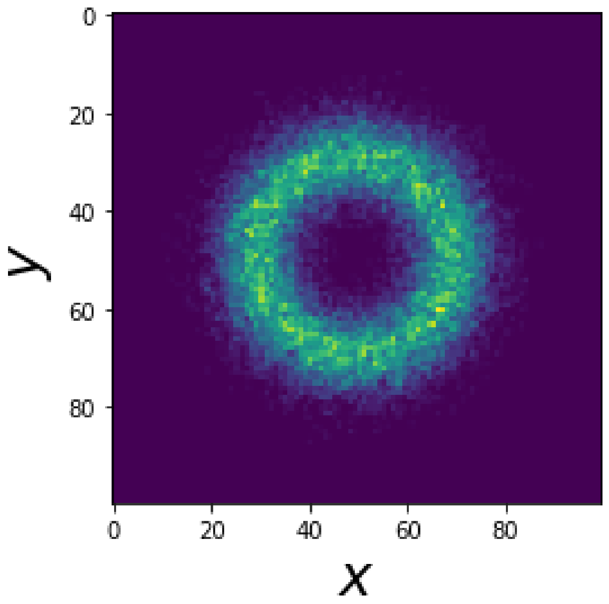

Figure 3: The grid representation of the charged particles.

Now that the particles are in a grid space, there is the obvious advantage of now being able to use this grid to solve for a potential field, but an extra layer of value which is that we need not stop at the given boundaries! If we were truly interested, we could have designed the grid space to be much larger, and solved for the potential equation over a far larger region. The biggest drawback to doing so now is the extra layer of computation time demanded to doing so, but later in this question, we will address different techniques for solving this sort of problem faster via. over-relaxation and Gauss-Seidel.

b) Using the standard relaxation method, we can solve for the potential equation of this system. We recover the following figure for the potential field in this system:

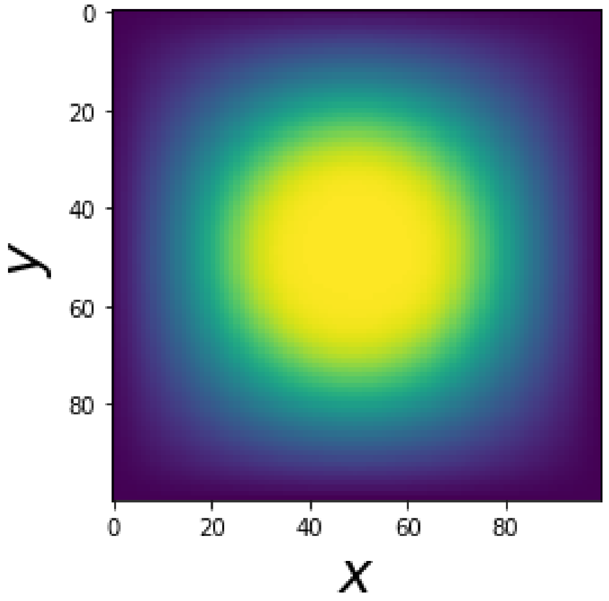

Figure 4: The potential over this grid. Solved for using the relaxation method.

This took approximately ten minutes to generate, and 27565 iterations, which is clearly not tractable systems larger than this.

c) Searching for a faster way to solve for the potential over this space, we can combine two proven to work techniques; Gauss-Seidel and over-relaxation.

The over-relaxation method is the natural follow-up to the relaxation technique just used, where an additional parameter omega is added to the algorithm which speeds up convergence by "pushing" the value of phi even further in the proper direction. However, this technique is unstable in the event of ϕ being too large, and as such a carefully selected parameter of omega must be chosen.

Furthermore, the Gauss-Seidel technique leverages values of potential in the grid already calculated. We can use the just learned values for ϕ at the next grid space. This also means only using one array for the calculation (whereas relaxation needed a new and old array), saving memory allocation to further optimize the procedure.

Combining these two techniques, we can calculate the potential field rapidly. But first, reasonable value of ω must be selected.

For this problem, we apply the golden-ratio search, and search over the parameter space of possible ω, and solve for the value corresponding to the lowest of steps to convergence. For clarity, I have plotted below the results of the golden-search method in it's last few iterations, finding the best value of ω, which creates a routine that is nearly 54x faster than the ordinary relaxation method!

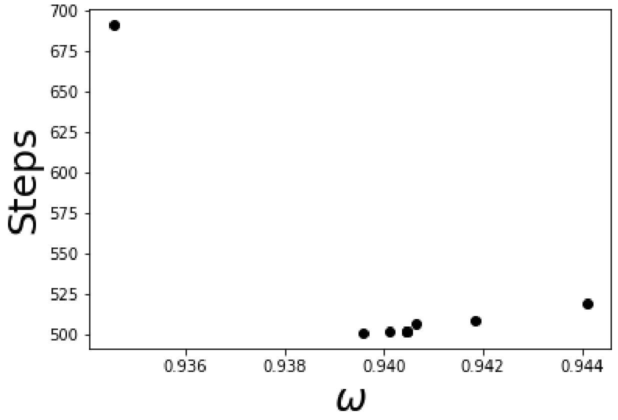

Figure 5: The relationship between steps to convergence and over-relaxation parameter ω.

After setting an error tolerance to 0.001, the optimal starting value for $\omega = 0.94065$, which converges after approximately 500 steps.

Doing so generates the following characterization of the potential ϕ.

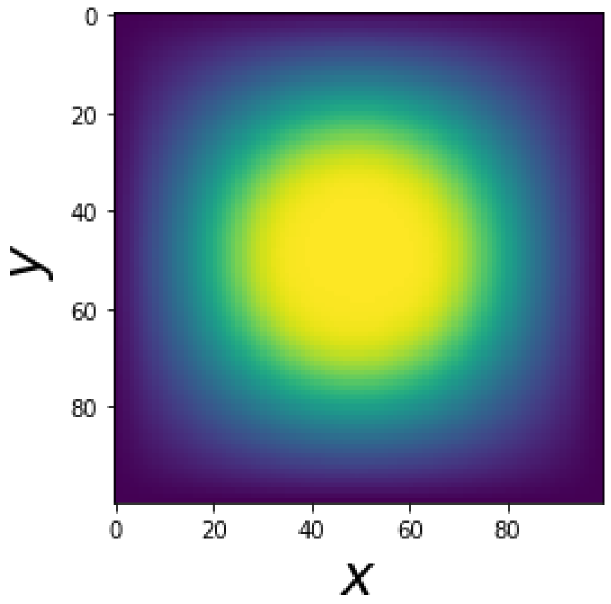

Figure 6: The recovered potential for $\phi(x, y)$ surrounding a ring of charged particles.

Through cloud in cell and likewise interpolation methods for converting point cloud structures into grid spaces, coupled with powerful ODE solving techniques like over-relaxed Gauss-Seidel, Poisson's and other differential equations can be quickly be solved with relative ease for complicated systems.