Allan Wei
PHYS 4250
Dr. Jason Fiege
January 2, 2021

Assignment 1 report

Abstract

The butterfly effect refers to a small change in the initial conditions in a dynamic system, which will drive the long-term and huge chain reaction of the entire system. It is a chaotic phenomenon. The "butterfly effect" also often appears in chaos theory.

In the winter of 1961, the American meteorologist Edward Lorenz was using a computer program to calculate the mathematical model he designed to simulate the air flow in the atmosphere. When doing the second calculation, he wanted to reduce the running time, so he started directly from the middle of the program and input the data printed out of the previous simulation result. However, the calculated result is completely different from the first time. After inspection, it is found that the cause of this is that, the printed data is 0.506 with accuracy of only 3 decimal places, while the correct value of the data is 0.506127 with 6 decimal places accuracy. In 1963, Lorenz published his paper "Deterministic Nonperiodic Flow" (Deterministic Nonperiodic Flow), which analyzed this effect. He also wrote in another journal article, "A meteorologist mentioned that if this theory is proven correct, a seagull flapping its wings is enough to change the weather forever."[1] He switched from seagull to a more poetic animals, butterflies, in his later speeches and papers. The most common explanation of this effect is that "a butterfly flapping its wings in Brazil can cause a tornado in Texas a month later."

The purpose of this assignment is to illustrate this effect by simulating 100 particles interacting with each other in a box by using MATLAB, with small perturbation in initial velocity for each run. The result is as predicted that a small change in initial condition in a nonlinear dynamic system would result in a tremendous difference at the end state.

Theory

In this simulation, the particle is treated as a point like object. We also ignore the rolling of the particle to simplify our simulation. We assumed that each particle produces a force that is radially outward from the point mass. The force of repulsion between two particles is proportional to the square of distance between them.

The force experience by a particle at position r_1, by another particle at r_2, is given by

$$F_1 = F_0 \frac{1}{|r_{12}|^2} \hat{r}_{12}$$

Where F_0 is the force magnitude.

We shall divide the force law into two cases, either the force produced by one particle has influence on all other particles, or it only exert on finite range of particles.

---

[1] Lorenz, Edward N. The Predictability of Hydrodynamic Flow (PDF). Transactions of the New York Academy of Sciences. 1963, 25 (4): 409–432 [1 September 2014].

For the first case, where the force experienced by one particle at position r due to N other particles is

$$F(r) = F_0 \sum_{i=1}^{N} \frac{1}{|\boldsymbol{r} - \boldsymbol{r_i}|^2} \widehat{\boldsymbol{R}}_{\boldsymbol{i}}$$

Where $\widehat{\boldsymbol{R}}_{\boldsymbol{i}}$ is the direction of $\boldsymbol{r} - \boldsymbol{r_i}$.
The second case is similar but instead summing up to N, we sum up the force within finite range, say R_0. Hence the force is given by

$$F(r) = F_0 \sum_{i=1}^{N'} \frac{1}{|\boldsymbol{r} - \boldsymbol{r_i}|^2} \widehat{\boldsymbol{R}}_{\boldsymbol{i}} \,, \forall |\boldsymbol{r} - \boldsymbol{r_i}| \leq \boldsymbol{R_0}$$

The particle's position and velocity are implemented by Verlet algorithm (also known as Störmer's method). At any given time t_n, the position one step forward and backward is given by

$$\mathrm{x}_{n+1} = x_n + v_n \Delta t + \frac{1}{2} a_n (\Delta t)^2 \qquad (1)$$

$$\mathrm{x}_{n\text{-}1} = x_n - v_n \Delta t + \frac{1}{2} a_n (\Delta t)^2 \qquad (2)$$

Combining (1) (2) we have

$$\mathrm{x}_{n\text{-}1} + \mathrm{x}_{n+1} = 2x_n + a_n (\Delta t)^2 \qquad (3)$$

Hence the original Verlet for position is

$$\mathrm{x}_{n+1} = 2x_n - \mathrm{x}_{n\text{-}1} + a_n (\Delta t)^2 \qquad (4)$$

Where the original Verlet for velocity is given by

$$v_n = \frac{x_{n+1} - x_{n-1}}{2\Delta t} \qquad (5)$$

An equivalent method we used in this assignment, known as Velocity Verlet algorithm is developed from the original Verlet method.

If we substitute equation (1) into (2) we have

$$x_{n-1} = x_{n+1} - 2v_n \Delta t \qquad (6)$$

Substitute (6) into (3) we have

$$2x_{n+1} = 2x_n + a_n (\Delta t)^2 + 2v_n \Delta t$$

Hence we have the derivation of position Verlet

$$x_{n+1} = x_n + v_n \Delta t + \frac{1}{2} a_n (\Delta t)^2 \quad (7)$$

For velocity, from (5) we write v_{n+1} as

$$v_{n+1} = \frac{x_{n+2} - x_n}{2\Delta t} \quad (8)$$

From equation (4) we write x_{n+2} as

$$x_{n+2} = 2x_{n+1} - x_n + a_{n+1} (\Delta t)^2 \quad (9)$$

Now we substitute back into (8) we get

$$v_{n+1} = \frac{x_{n+1} - x_n}{\Delta t} + \frac{1}{2} a_{n+1} \Delta t \quad (10)$$

After substituting (7) into (10) we get the derivation of Velocity Verlet

$$v_{n+1} = v_n + \frac{a_n + a_{n+1}}{2} \Delta t \quad (11)$$

We use (7) and (11) to update our position and velocity for each forward time step in our simulation.

MATLAB interpretation

Parameters

We set mass=1 for all particles. By such choice the accelartion of a particle is just the force excert on it.The position of particles start off random in a 2D 1 by 1 box with random velocity follows the normal Gaussian distribution. The number of games is chosen to be 25 which is sufficent for the purpose of this simulation. All initial set ups for 25 games are the same besides the initial velocities has a small pertubation from the first game, the difference is roughly ~ 1e-12. Considering the limit of computational resources, around 1e6 of iterations would be reasonable amount to demonstrate the purpose. In order to make more accurate calculations our time step is chosen to be 1e-5. In each time step we calculate the position and velocity of all 25 games simoutinously, as well as the despersion between them. The force magnitude F_0 is chosen to be 20 in order to see sensible dispersion between games in 100,000 iterations with the parameters set up above, and will be discuss further in the result.

Functions

Since the force is proportional to $\frac{1}{r^2}$, if two particles getting too close, the force will quickly diverge. To prevent this, a 'soften parameter' is included in the numerator, such that the force law is adjusted to be proporetional to $\frac{1}{r_0^2 + r^2}$.

A more clever way to introduce the finite range force law, is to take the maximum

between 0 and *F-F′*where F' is a constant chosen to be $\frac{1}{2r_0^2}$. By such construction any force excert on the particle which is less than $\frac{1}{2r_0^2}$ is neglected.

The funcion CHECK_BOUNDARY is used in the Verlet algorithm to ensure particles stay inside the box.

Results

null test

To ensure that all games stay the same without us introducing any perturbations, it is necessary to simulate a run with all 25 games having the same initial parameters.

Here we run 25 games simultaneously with 100,000 steps of iterations using two different force law. As a result, we can see that the dispersion of position remains non-increasing, within the value of 10^{-15} throughout the whole simulation. Theoretically the dispersion is 0, the non-zero value in the plot is due to round off error in MATLAB.

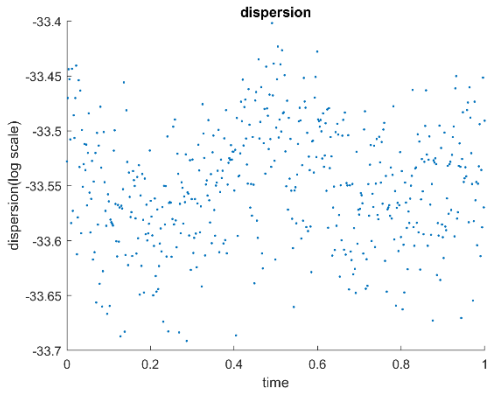


Image 1 dispersion of position after 100,000 steps-finite range force law (Y axis log scale of base e)

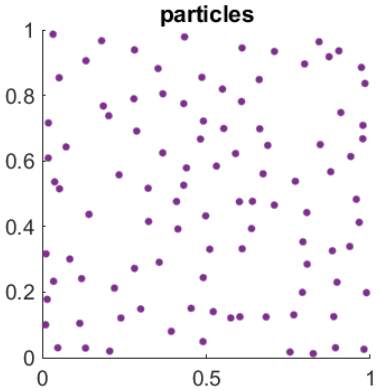


image 2 particles in all games after 100,000 steps (stacking on top of each other)-finite range force law

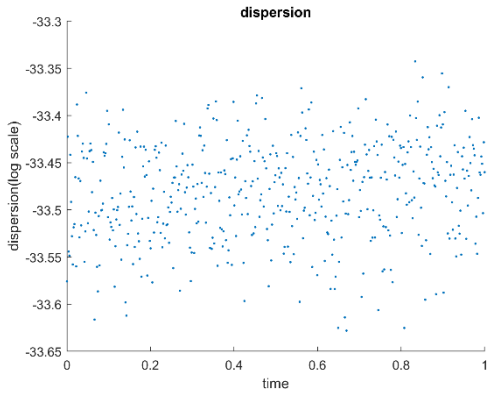


image 3 dispersion of position after 100,000 steps-infinite range force law (Y axis log scale of base e)

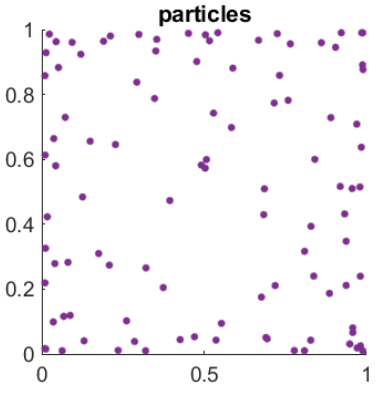


image 4 snapshot of particles in all games after 100,000 steps (stacking on top of each other)-infinite range force law

As we can see, all games remain identical for same initial setups. We can now

introduce perturbations to observe how games diverge from each other.

Results using infinite range force law

Now we introduce a small perturbation in velocity for all games besides the first one, with the scale of 10^{-12} following normal Gaussian distributions. All other setups remain the same as the null test. The particles interact via infinite range force law.

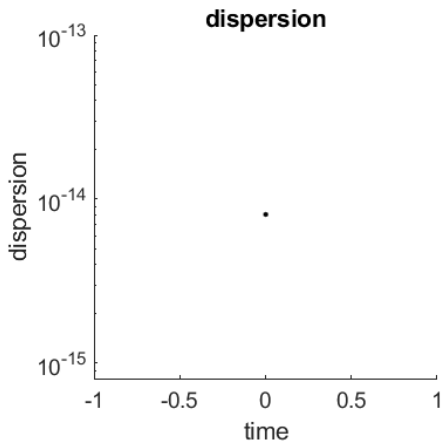


image 5 dispersion of position after 200 iterations (Y axis log scale of base 10)

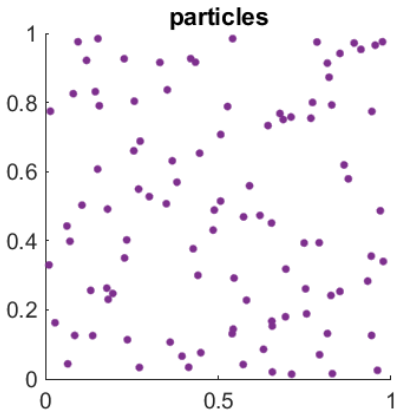


image 6 snapshot of particles in all games after 200 steps (stacking on top of each other)

After 15,000 steps of iterations (t=0.15) the games start to diverge from each other and reaching to plateau.

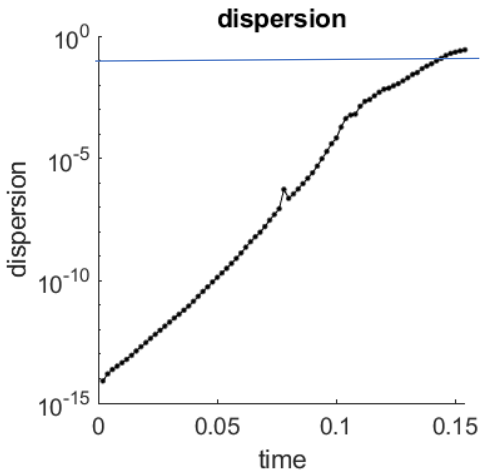


image 7 dispersion of position after 15,000 iterations (Y axis log scale of base 10, blue line indicates dispersion of 10^{-1})

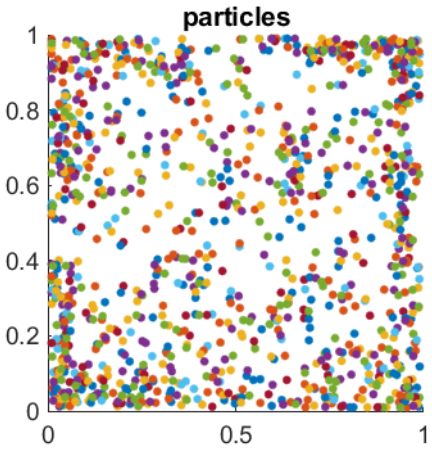


image 8 snapshot of particles in all games after 15,000 steps

At 20,000 iterations all games become undisguisable from one another.

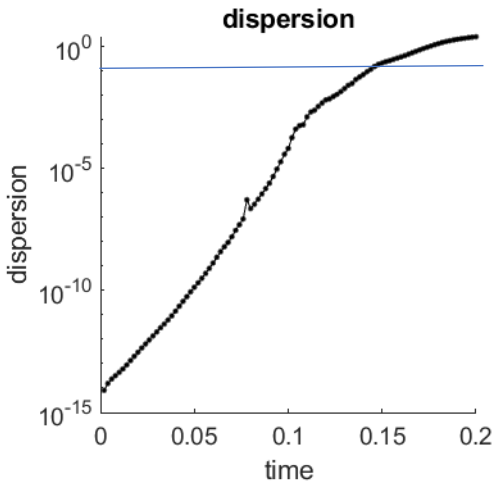


image 9 dispersion of position after 20,000 iterations (Y axis log scale of base 10, blue line indicates dispersion of 10^{-1})

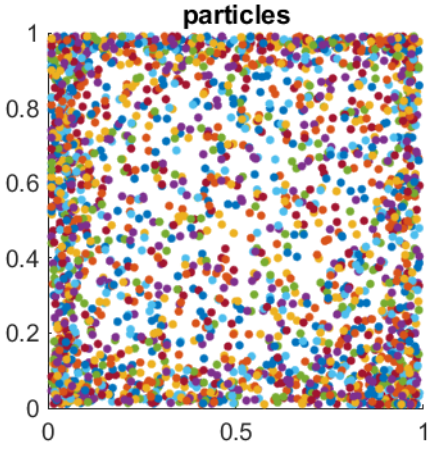


image 10 snapshot of particles in all games after 20,000 steps

To study how quickly 25 games diverge from each other, we observe from plot that dispersion increases exponentially. It is reasonable for us to assume that the points follow the exponential e^{-kt}. We only interested in the data before the curve reaching plateau to better study the behavior of dispersion. To determine k, we take the natural log of the data and plot it. As shown below, the new points now follow a linear distribution. We use POLYFIT of degree 1 to estimate k and found the value to be 224.5605.

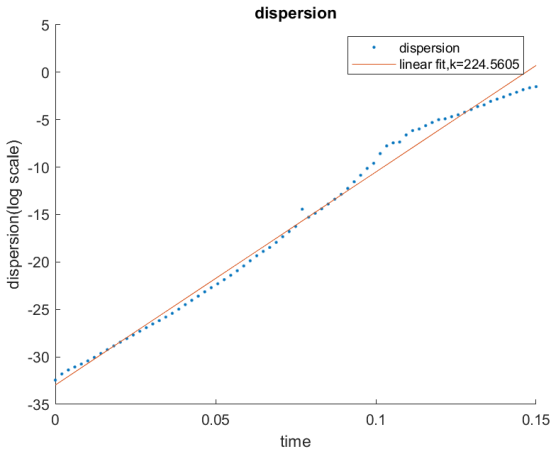


image 11 dispersion in all games before reaching plateau.

It is also valuable to compare how k differs from different force laws.

Result using finite range force law

We now use finite range force law to see how games diverge from each other. All

initial setups are the same as the previous one. We observe that particles move way slower when experienced force from finite range of other particles.

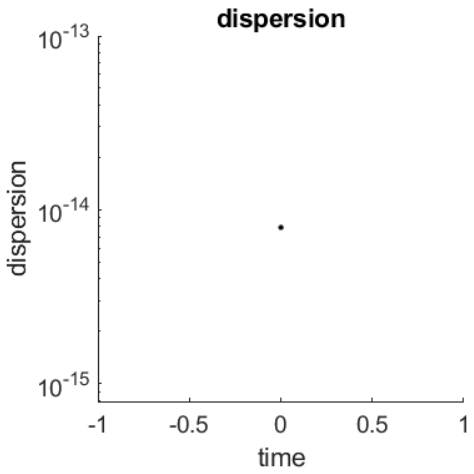


Image 12 dispersion of position after 200 iterations (Y axis log scale of base 10)

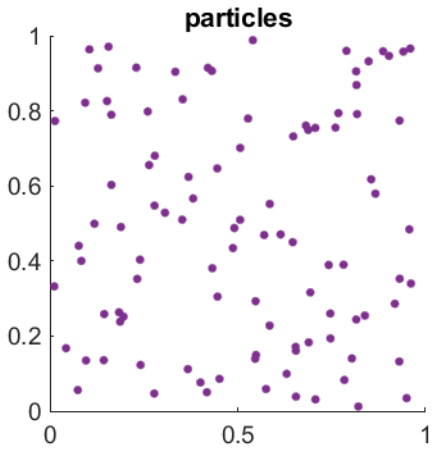


image 13 snapshot of particles in all games after 200 steps (stacking on top of each other)

As a comparison with infinite range force law at step 15,000 (t=0.15), we do not observe any noticeable dispersion from games.

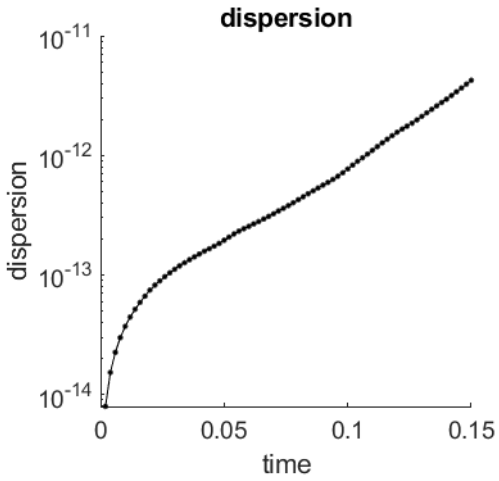


image 14 dispersion of position after 15,000 iterations (Y axis log scale of base 10

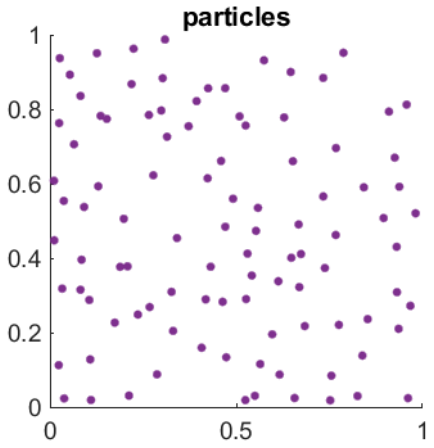


image 15 snapshot of particles in all games after 15,000 steps (stacking on top of each other)

After 76,600 steps, games start to have clear dispersion from one another. The curve of dispersion steadily reaching to a plateau.

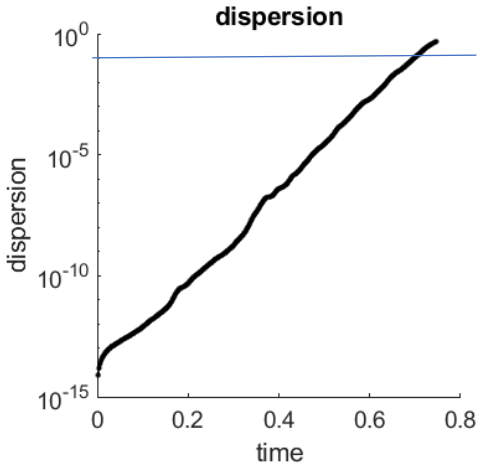


image 16 dispersion of position after 76,600 iterations (Y axis log scale of base 10, blue line indicates dispersion of 10^{-1})

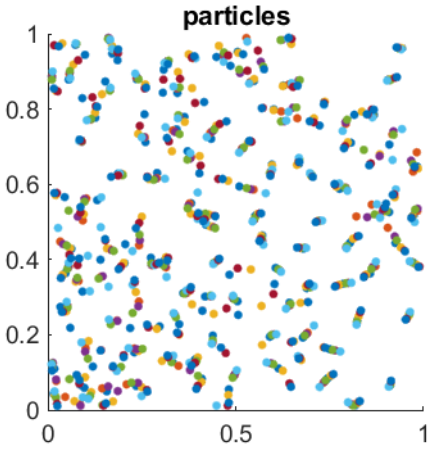


image 17 snapshot of particles in all games after 76,600 steps

At 100,000 steps, we see the particles have completely different position from one game to another at any given time t.

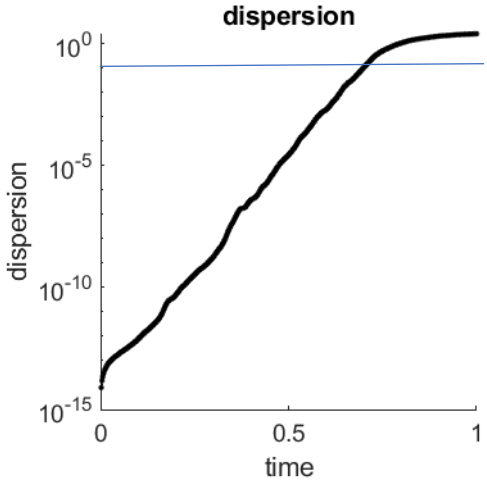


image 18 dispersion of position after 20,000 iterations (Y axis log scale of base 10, blue line indicates dispersion of 10^{-1})

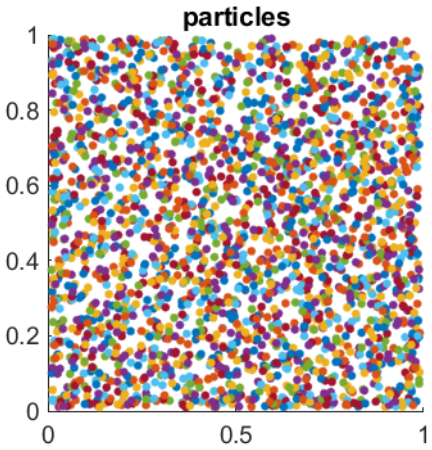


image 19 snapshot of particles in all games after 100,000 steps

We estimate k using the same method as the previous simulation and found its value to be 41.1564. We conclude that when particle interacts under finite range force law, games diverge way slower than using infinite range force law.

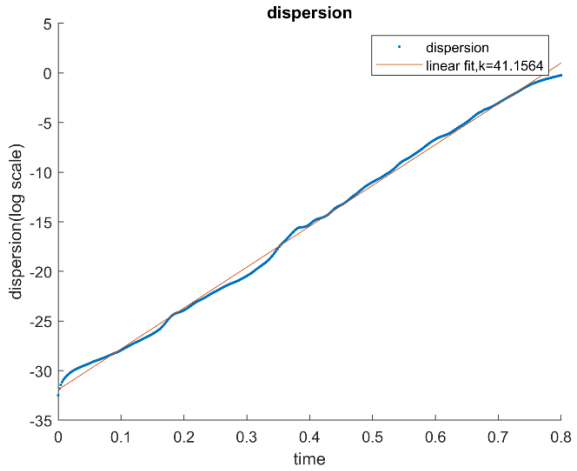


image 20 dispersion of all games before reaching plateau.

Behaviors of particles under different force law

For simulations we discussed above, the force magnitude F_0 in both force laws we chose are 20. The specific number is chosen in order to make the behavior of particles more realistic. For finite range force law, we noticed that when the number is differed from 20, say 10, there are some cases where particles pass through each other due to weak interaction. If the force magnitude was chosen to be 40, some particles will be bouncing along the boundary for a long time. The behavior of particles under infinite range force law is somehow weird and unrealistic regardless of our choice of F_0. It is due to the fact that, since the force is global, majority of particles experience same amount of force due to the symmetry of their starting position. The soften parameter in the numerator determines the minimum and maximum amount of force that one particle can exert on the other. In the case where $F_0 = 1 \quad , r_0 = 0.1$, we have $F_{min} \approx 0.99$ and $F_{\max} \approx 100$. Under such construction the particles will eventually reach equilibrant at the corner of the boundary.

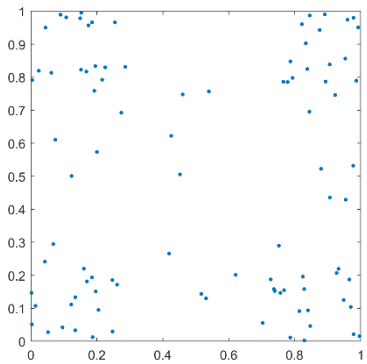


image 21 particles under infinite range force law after a few amounts of iterations.

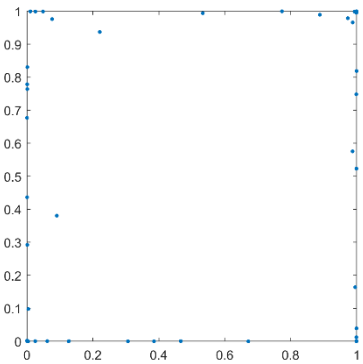


image 22 particles under infinite range force law after sufficiently large amounts of iteration.

Results comparing different choice of force magnitude

Regardless of how the particles behave, we still interested in how the games

diverge from one another with different choice of force magnitude. Now we shall conduct this simulation using only finite range force law (Since we can easily draw conclusion from one another). The setups are all the same except we change F_0 to 10 this time.

As we can see below, the games start diverging from each other after around 95,000 steps. The time it takes to diverge in this case is way longer than we set F_0 =20. (see image 16 and 17 for comparison)

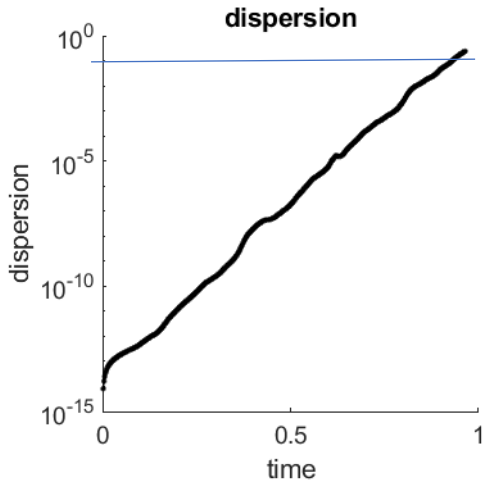


image 23 dispersion of position after 95,000 iterations (Y axis log scale of base 10, blue line indicates dispersion of 10^{-1})

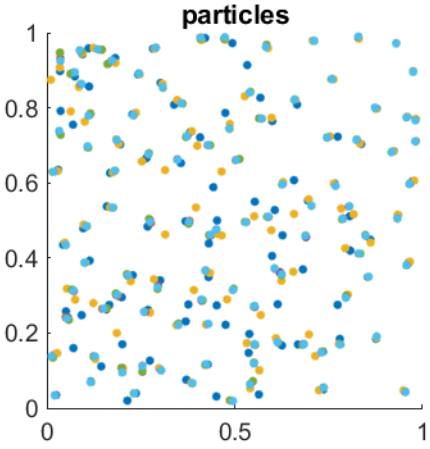
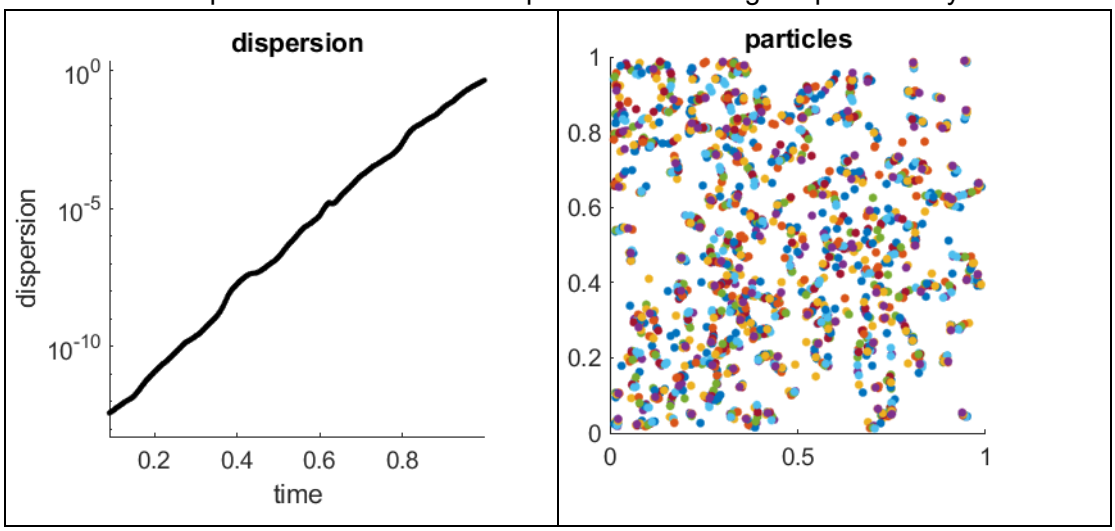


image 24 snapshot of particles in all games after 95,000 steps

5000 steps further the curve of dispersion is reaching the plateau at y=1.



We estimate the k value same as above and found its value to be 34.4310, which is less than 41.1564 we had for F_0 =20.

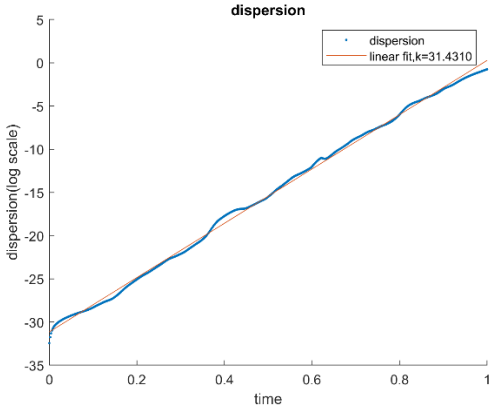


image 25 dispersion of all games before reaching plateau.

We can conclude that, the larger the choice of our F_0, the faster the games diverge from each other.

Reasons behind different force law and magnitude having different diverge speed

Before stepping into the details, we need to adjust our problem in order to simplify our calculations. Consider we only have two particles in 1D, with distance d away from each other, where the particles have velocities with magnitude v_0.

The initial acceleration of blue particle is given by

$$a_0 = F_0 \frac{1}{d^2}$$

We update the position using velocity Verlet

$$x_1 = x_0 + v_0 \Delta t + \frac{1}{2} a_0 (\Delta t)^2$$

Hence the increment Δx is

$$\Delta x_1 = v_0 \Delta t + \frac{1}{2} a_0 (\Delta t)^2$$

We now update our acceleration

$$a_1 = F_0 \frac{1}{(d + \Delta x)^2}$$

And velocity

$$v_1 = v_0 + \frac{a_0 + a_1}{2} \Delta t$$

Now we calculate the increment of position for next time step to be

$$\Delta x_2 = v_1 \Delta t + \frac{1}{2} a_1 (\Delta t)^2 \qquad (11)$$

We shall do this again with initial velocity v_0 plus some perturbation, say σ. After going through the steps we get

$$\Delta x_1' = V_0 \Delta t + \sigma \Delta t + \frac{1}{2} a_0 (\Delta t)^2$$

We found our acceleration to be

$$a_1' = F_0 \frac{1}{(d + \Delta x_1')^2} = F_0 \frac{1}{(d + \Delta x_1 + \sigma \Delta t)^2}$$

And velocity

$$v_1' = v_0 + \sigma + \frac{a_0 + a_1'}{2} \Delta t$$

Finally, our increment of position is

$$\Delta x_2' = v_1' \Delta t + \frac{1}{2} a_1' (\Delta t)^2 \qquad (12)$$

If we subtract (12) and (11) we get the dispersion

$$\Delta x_2' - \Delta x_1' = (v_1' - v_1) \Delta t + \frac{1}{2} (\Delta t)^2 (a_1' - a_1)$$

If we substitute v_1' and a_1' cancels some terms in common we get

$$\Delta x_2' - \Delta x_1' = \sigma \Delta t + F_0 \frac{1}{2} \left(\frac{1}{(d + \Delta x_1 + \sigma \Delta t)^2} - \frac{1}{(d + \Delta x_1)^2} \right) \Delta t$$
$$+ \frac{1}{4} F_0 \left(\frac{1}{(d + \Delta x_1 + \sigma \Delta t)^2} - \frac{1}{(d + \Delta x_1)^2} \right) (\Delta t)^2$$

The algebra gets very ‘ugly’ as we proceed further. If we are allowed to be less careful, we can conclude that when introducing the perturbation in initial velocity, it scales up in both position, velocity and acceleration after each iteration, which will end up in tremendous different at the end state. We can also see that the choice of our F_0 amplify the perturbation, which explains why the k values differ when we conduct our simulation on different F_0. It also explains why using different force laws can result in the games diverge in a different speed. Further discussion is needed in order to determine the closed form solution of the dispersion (in this simplify case). However, in general it is not possible to keep track of every particle, since the dynamic system is chaotic in the first place.