Escape of the Particles

Gil Garcia

Wesleyan University, Department of Physics, Middletown, CT 06459 U.S.A.

May 17, 2019

Keywords examples: numerical integration, atom trapping, equilibration time

Introduction

With higher computing power, we are given the ability to model systems that were once taught unsolvable or that still have no analytic solution. One such case is modeling molecular dynamics. While this has been a fairly well studied area in physics, because of computer simulations, we are able to reach greater understanding than before of their dynamics and consider more complex systems. In this iteration of the molecular dynamics simulation, we consider the case where we have N number of particles in a trap potential. Allowing the particles to live initially in the trap potential allows us consider the behavior and interactions a particle when place in an initially confined region. Such scenarios that we consider are the strength of the trap and how particles can escape from their host trap. Here, we will consider the scenarios mentioned along with a discussion of how to best optimize the program and how to conserve energy.

Computational Methods

Rather than placing our particles into motion in free space, as mentioned earlier, a more interesting (and applicable) scenario would be to introduce a trap well. Such a barrier will increase the energy of the system by producing a potential energy for the particles with the trap wall. We can define the trap wall and the potential energy of the particle interacting with the trap wall with the following equation, called U_1:

$$U_1(r): \frac{\mu r^2}{1+r^2} \tag{1}$$

Where r is the distance of the particle from the center of the trap and is calculated using the x and y components of the particle since we update the trajectories by their components. Thus, $r=\sqrt{x^2+y^2}$. And where μ is the ratio of the strength of the particles interacting among each other to the strength of the trap potential. The particles that we introduce will live and interact in the well, but there is a possibility that the particles can, in fact, escape the well. We include a discussion on this later.

Additionally, we would like to have a system that that can hold more than 1 particle, so we must define a way to calculate the potential energy due to the atomic interactions. To create more realistic atomic interactions, (that is, one that is repulsive close range and attractive long range), we define the following potential equation, U_2:

$$U_2(r)=\frac{1}{2}mr_0^2\frac{(r-r_0)^2}{r^2} \tag{2}$$

This produces the desired inter particle potential. The sum of U_1 and U_2 make up our total potential energy, but, of course, this is only one half of the pieces needed get total energy.

Next, We must define the kinetic energy of our system and we can achieve this in the traditional way in which kinetic energy is defined. That is,

$$K_e=\frac{1}{2}mv^2 \tag{3}$$

where in this case, since we are updating our velocities by their components, $v=\sqrt{v_x^2+v_y^2}$.

For the purposes of simplifying our set up, we take $\mu=1$, $r_0=1$, and and $m=1$.

We turn our attention to the heart of the program which is the `Gradiant` function and the various integrators: `Euler, 2nd Order Runge Kutta (RK2)`, and `4th order Runge Kutta (RK4)` [1].

The `Gradiant` function takes all parameters that describe the motion of the particles in our system. That is, it takes in the position in the x and y direction and the velocity in the x and y direction. Given these parameters, we use the change in position to calculate the velocity step. To do this, mathematically we use the following operation on the potential function P:

$$\nabla P=(\frac{\partial P}{dx}\hat{i},\frac{\partial P}{dy}\hat{j}) \tag{4}$$

To differentiate in our program, we use the explicit definition of derivatives:

$$\frac{df}{dx}=\frac{f(x+h)-f(x-h)}{2h} \tag{5}$$

where, in our case, f is the `potential` function. Operating on the x and y component of each particle, we obtain the gradient values needed to take the next velocity step. It naturally follows that we must now take this step and we do so using one of the three integration methods mentioned earlier.

For the sake of space, we will not discuss in detail details of each integration. However, it is important to note that there is a good reason for having three different integrators. While the accuracy and precision of the position and velocity step increase as we move from `Euler` to `RK2` to `RK4`, so does, too, the complexity of the function. Therefore, while `RK4` will produce finer position/velocity steps, it will take much more computational power and time to do so. This brings up a good argument for why a person may choose the `Euler` integrator over `RK4`. `RK2` is also included as a middle ground between the two other integrators.

The final piece that completes our program is the initial set up and conditions of our interacting particles. For the N particles in our system, we allow the initial position (x_0, y_0) to be given randomly within the interval $(\frac{-N}{2},\frac{N}{2})$. We define this to be our interval so that we ensure no collisions due to our own making. We randomize our initial positions with the `srandom(time(0))` function. Next, so that all the velocities are the same initially, we make the x and y component of velocity (v_x,v_y) all equal to 0.

This now gives us all the necessary functions needed to study the dynamics of interacting particles.

However, to be able to study the accuracy of our results, we calculate a couple of things. To look at how many particles have escaped our trap, we first run test trials to see the maximum orbit that particles take around the trap. Once we get a consistent results, we set the trap to be 3 times larger than that since it could be the case that a particle goes really far out but still comes back in (almost like a very elliptical orbit). The trap well is thus a function of N, the number of particles that we place in the trap and of t, the time we allow for the particles to evolve from the intial positions. We also calculate the average and standard deviation of our position and velocity and then separate our particles into bins according to their velocities in each component. Finally, to get another good measure for the accuracy in our results, we calculate the percent change in total energy.

Results

Our first scientific investigation is ensuring that our code has been constructed such that all physical properties that are being modeled hold to our expectations.

We do this in three ways: checking that particles do indeed interact with one another and with the trap, checking that energy is conserved, and checking that the velocity distributions are consistent with theory.

[1] Thank you Mr. Jacob Fanthorpe for writing the code for the integrators and sharing them with the class.

The simplest way to check that particles are behaving as expected is to construct a plot of their motion in the x and y direction. Doing so, we produce the following for $N=10$,$DT=0.001$, and $t=4\pi$:

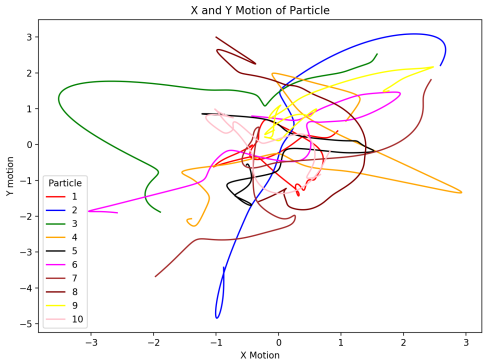

Figure 1: Particles interacting in trap well with 0 escaped particles.

Due to the trap potential and the inter-particle interactions, we produce particle trajectories that model what we would expect. That is, we expect for the particles to move about the trap and not escape immediately, which is what is seen. Then, while trajectories seem random, we see that the particles do 'gravitate' towards each other meaning that there is an attraction and repulsion interaction between them, as desired.

We then want to make sure that energy is conserved throughout every set up we choose to use and also for a large number of particles. Thus, we inspect the percent energy change from the initial time = 0 to the final time ($t=4\pi$ in this case). For various N,$DT=0.001$, and $t=4\pi$, we get

N	Δ E(%)
1	6.26×10^{-7}
10	4.26×10^{-5}
50	1.24×10^{-3}
100	2.61×10^{-3}
200	2.19×10^{-2}

We see an important trend which is that with more particles, it is harder to maintain a constant energy. Thus, energy conservation decreases as a function of time. In order to be able to run the code at much larger N values, a much smaller DT must be defined. The trade off here however is that every magnitude decrease in the time step is a magnitude increase in the time to run the code.

For all of these cases, a standard energy vs time plot looks as follows:

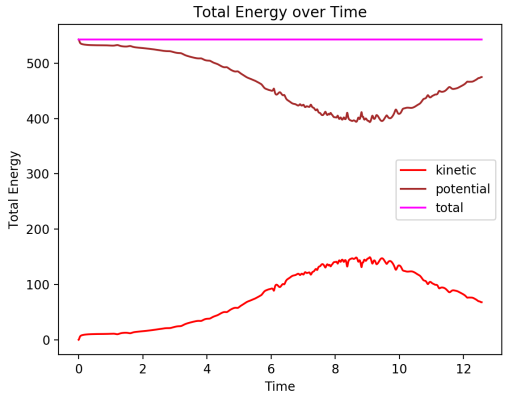

Figure 2: Energy over time for $N=50$,$DT=0.001$, and $t=4\pi$.

Any physics student would agree in saying this obeys the rules of energy conservation since as time progresses, energy remains constant, and when potential changes, kinetic changes in opposition to it.

Finally, we want to consider how the velocity distribution looks for particles that have interacted with the trap for a long time. We want to compare how the distribution compares to that of what theory tells us. Specifically, we want to fit the velocity distributions to the formula that describes free gas (which, according to Prof. Stewart, is valid for our purposes):

$$f(v)=\frac{1}{\sqrt{2\pi m(2E/2N)}}\exp(-\frac{1}{2}mv^2\frac{3N}{2E}). \quad (6)$$

Binning our particles by the x and y component, we can thus plot a density distribution and fit the $f(v)$ function above. Doing this, we get

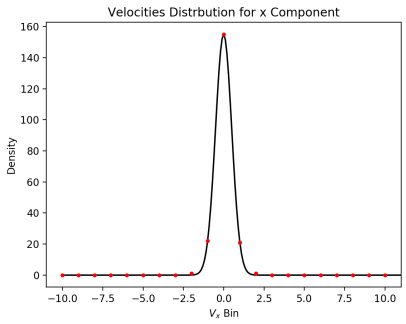

Figure 3: Velocity distributions in the x direction.

and

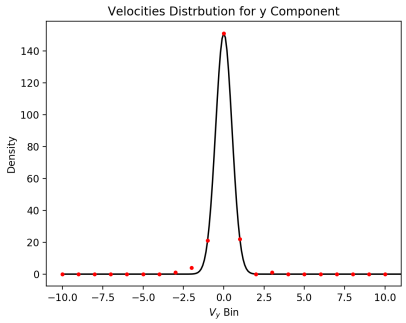

Figure 4: Velocity distributions in the y direction.

We see that even for a long time ($t = 20\pi$), the velocities in both directions still follow a Gaussian distribution and in particular, they both fit the $f(v)$ function defined above. Hence, they the velocities are consistent with the theory.

From here, we move on to discuss more, in more in depth detail, the trap loss.

To be able to classify a particle, initially in the trap, as having escaped, we first simply consider its distance. Similar to what has already been mentioned, we will first run the code a few times for the set up we are working with to observe where and how far the particles tend to move within the trap. We then define a radial component much bigger than that observed value and say that a particle has cleared step 1 for leaving the trap if it has a distance greater than what we define. Step 2 ensures that the particle will not eventually stop and turn around. To do this, we calculate the kinetic energy and the potential energy due to the trap wall, which is our function U_1, of the escaped candidate. If the particle's kinetic energy exceeds its U_1 potential energy, then it is likely that it will not turn around. So to test this, we plot once again the trajectories of the particles and observe visually the number of particles escaped and check if it matches our calculated number of particles escaped.

For a set up of $N = 10, DT = 0.001$, and $t = 20\pi$, the following particle trajectory plot is generated:

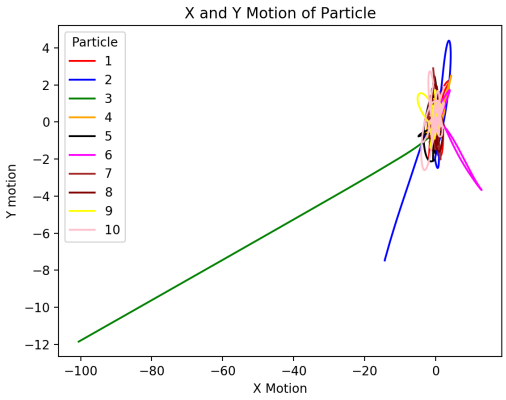

Figure 5: Trajectory of an escaped particle.

Graphically, we see that particle 3 (green path) clearly has a distance much larger than any reasonable trap wall we could set up. Additionally, it has traveled close to 100 units in the $-x$ direction indicating that it is probably moving at a rapid velocity and hence it has a high kinetic energy. Particle 3 should then be a particle that our 'escaped particles detector' should count as escaped. Looking at particle 6 (magenta path), a rather small trap wall would have considered it to be escaped which emphasizes the importance for also requiring a high kinetic energy since we see that at the turn, it would have a kinetic energy of 0. Thus,

our detector does not consider it escaped. Finally, we take a look at particle 2 (blue path). we see that it was previously traveling in the upwards direction but turned around. It then comes in and out of the trap and head downwards. We can deduce that particle 2 has a higher than average velocity since it traveled a fair amount. However, in this case, the trap wall is set to 30 so its radial position ($\approx\sqrt{20^2+10^2}\approx 22<30$) does not yet reach that. Thus, our escaped particle detector should tell us that there is only 1 particle escaped, and indeed it does.

Lastly, we want to look at the effects in variation of the μ factor in our U_1 function. This constant governs the ratio of the strength between the interparticle energy to the strength of the trap defined. Thua, for a large μ value, we expect the interpartical interactions to govern the trajectories of the particles. On the other hand, for a small μ, we would expect the trap to govern the motion of the particles. So running our code at different values of μ but same time value $9t=30\pi$, we produce the following trajectory plots:

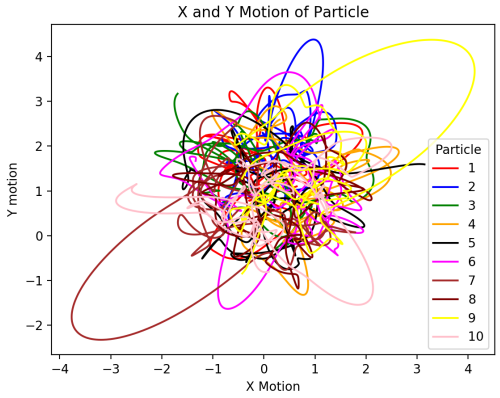

Figure 6: Trajectory of particles with $\mu=\frac{1}{10000}$.

and

From the plots, we see that we do indeed get the hypothesized results. That is, with a small μ, the trap potential should govern the movement. We see in figure 6 that because the trap is much stronger than the inter particle interactions, no particle escapes. We

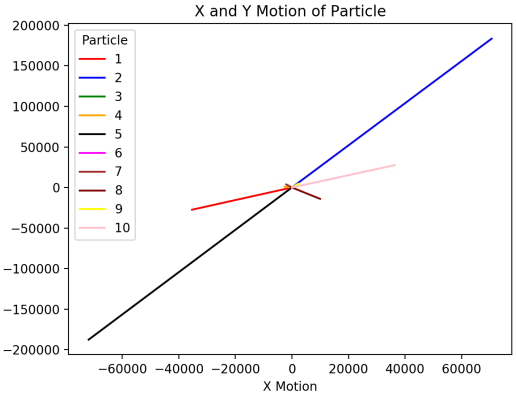

Figure 7: Trajectory of particles with $\mu=10000$.

then also see that a large μ does in fact correspond to stronger inter particle interactions. In figure 7, we see that more particles are ejected from the trap since the trap is much weaker and they feel other particles' force much more. The strong repulsion close range creates a lot of escaped particles.

Conclusions

We see that with this simulation, a lot can be learned from only asking the basic science questions. We first noted that because we were able to create a system that conserves energy and that also follows a Gaussian curve when looking at the velocity density, we are able to ask meaningful science questions. The first of which was looking at the conditions for trap loss. We showed that considering distance from the trap and the kinetic energy must be taken into account before we can confirm if a particle has escaped a trap. We then look at the effects that different trap strengths can have on the types of interactions and trajectories a particles has. Looking ahead, we would want to consider a much larger sample of particles, but in order to do so, we must also look into further optimizing our code. Additionally, we can ask more science questions and/or adapt our code to other scenarios, such as stellar interactions.