Atoms and The Void
(hard-sphere molecular dynamics)

HPC Project Option

February 16, 2023

The Swerve

Sometime around 60 BC, the Roman poet Lucretius wrote *De rerum natura* (On the nature of things). That work contained much of what we consider a foundation for modern scientific philosophy, including this passage:

> *"When atoms move straight down through the Void by their own weight, they deflect a bit in space at a quite uncertain time and in uncertain places, just enough that you could say that their motion has changed. But if they were not in the habit of swerving, they would all fall straight down through the depths of the Void, like drops of rain, and no collision would occur, nor would any blow be produced among the atoms. In that case, nature would never have produced anything."*

Here are the rules for our Lucretian universe :

- We have a collection of n hard disks (the Atoms), each with radius σ, and mass m. The Atoms could be identical, or each could have a different mass and radius.
- These disks occupy a two-dimensional space – the Void. We'll discuss what geometry you can use for the Void later.
- At each time $= t$, the ith marble has a centre position $\underline{x}_i(t) = (x_i(t), y_i(t))$ and velocity $\underline{v}_i(t) = (u_i(t), v_i(t))$.
- The Atoms fly through the Void unimpeded until a specific time they collide and bounce (the Swerve) off each other according to a well-defined set of rules.

Particle Collisions

We say that the ith and jth Atoms *Swerve* according to the following:

- Before a collision, the Atoms move in straight lines according to

$$\underline{x}_i(t + \Delta t) \;=\; \underline{x}_i(t) + \Delta t\, \underline{v}_i. \tag{1}$$

- A collision happens *at a time and place where*

$$|x_i(t^*) - x_j(t^*)|^2 + |y_i(t^*) - y_j(t^*)|^2 = (\sigma_i + \sigma_j)^2. \tag{2}$$

  You have to compute $t^* = t + \Delta t^*$ with the quadratic formula.

- *Be careful about which root you take.* There are only two options.
  Define *current* (time $= t$) separation and relative velocity

$$\underline{r}_{i,j} \;=\; \underline{x}_i - \underline{x}_j, \qquad \underline{v}_{i,j} \;=\; \underline{v}_i - \underline{v}_j. \tag{3}$$

  Either there is a finite positive solution to the equation

$$||\underline{r}_{i,j}||^2 + 2\Delta t^* \left\langle \underline{r}_{i,j}, \underline{v}_{i,j} \right\rangle + (\Delta t^*)^2 ||\underline{v}_{i,j}||^2 \;=\; (\sigma_i + \sigma_j)^2. \tag{4}$$

  Or the particles never collide on their current course. There are some conditions for when this can happen.

  1. By definition

$$|\sigma_i + \sigma_j| \;<\; ||\underline{r}_{i,j}||. \tag{5}$$

     This means they haven't collided yet.

  2. To have a collision, we must have

$$\left\langle \underline{r}_{i,j}, \underline{v}_{i,j} \right\rangle \;<\; 0. \tag{6}$$

     The Atoms much be headed *at* each other.

  3. They can't have too large a separation

$$||\underline{r}_{i,j}||^2 - \frac{\left\langle \underline{r}_{i,j}, \underline{v}_{i,j} \right\rangle^2}{||\underline{v}_{i,j}||^2} \;<\; (\sigma_i + \sigma_j)^2 \tag{7}$$

     Geometrically, this means

$$||\underline{r}_{i,j}|| |\sin\theta_{i,j}| \;<\; |\sigma_i + \sigma_j|, \tag{8}$$

     where $\theta_{i,j}$ is the angle between $\underline{r}_{i,j}$ and $\underline{v}_{i,j}$.

- If the conditions are satisfied, then the expected collision time is

$$\Delta t^* \;=\; \frac{||\underline{r}_{i,j}||^2 - (\sigma_i + \sigma_j)^2}{-\left\langle \underline{r}_{i,j}, \underline{v}_{i,j} \right\rangle + \sqrt{\left\langle \underline{r}_{i,j}, \underline{v}_{i,j} \right\rangle^2 - \left(||\underline{r}_{i,j}||^2 - (\sigma_i + \sigma_j)^2\right) ||\underline{v}_{i,j}||^2}}. \tag{9}$$

  All the terms in the formula satisfy the necessary conditions and $\Delta t^* > 0$.

- If one of the conditions is unmet, then $\Delta t^* = \infty$, by definition.

- *If* a collision happens *then* the velocities update according to the rule for perfectly elastic collisions.

$$\underline{v}_i' \;=\; \underline{v}_i \;-\; P_{i,j}/m_i \tag{10}$$
$$\underline{v}_j' \;=\; \underline{v}_j \;+\; P_{i,j}/m_j \tag{11}$$

The change of momentum is

$$P_{i,j} \;=\; M_{i,j}\,\hat{r}_{i,j}\cdot\left(\underline{v}_i-\underline{v}_j\right)\hat{r}_{i,j}, \quad \text{where} \quad M_{i,j} \;=\; \frac{2}{m_i^{-1}+m_j^{-1}} \tag{12}$$

The unit vector $\hat{r}_{i,j}$ points from the centre of Atom-i to Atom-j,

$$\hat{r}_{i,j} \;=\; \frac{\underline{x}_i-\underline{x}_i}{\sigma_i+\sigma_j}. \tag{13}$$

If the masses are equal then $M_{i,j}=m_i=m_j$. The mass formula also works if one of the masses is infinite, e.g., $1/m_i=0$. If both masses are infinite, you get to discover what happens when an immovable object meets an irresistible force.

You can check that the collision satisfies three essential properties

1. The Atoms rebound *normal* (perpendicular) to the surfaces of contact.
2. The collision conserves total momentum

$$m_i\,\underline{v}_i'+m_j\,\underline{v}_j' \;=\; m_i\,\underline{v}_i+m_j\,\underline{v}_j \tag{14}$$

3. The collision conserves total energy

$$\frac{m_i\,||\underline{v}_i'||^2}{2}+\frac{m_j\,||\underline{v}_j'||^2}{2} \;=\; \frac{m_i\,||\underline{v}_i||^2}{2}+\frac{m_j\,||\underline{v}_j||^2}{2}. \tag{15}$$

Wall Collisions

The particles have to be confined somehow, or they will eventually fly apart, and collisions cease. There are many ways you can provide confinement.

- A 2-disk with radius R,

$$\mathcal{D}_R \;=\; \left\{\,(x,y)\in\mathbb{R}^2 \;:\; x^2+y^2\le R^2\,\right\} \tag{16}$$

- A rectangle with sides L_x, L_y,

$$\mathcal{R}_{L_xL_y} \;=\; \left\{\,(x,y)\in\mathbb{R}^2 \;:\; 0\le x\le L_x\;0\le y\le L_y\,\right\} \tag{17}$$

- A 2-sphere with radius R,

$$\mathcal{S}_R^2 \;=\; \left\{\,(x,y,z)\in\mathbb{R}^3 \;:\; x^2+y^2+z^2=R^2\,\right\} \tag{18}$$

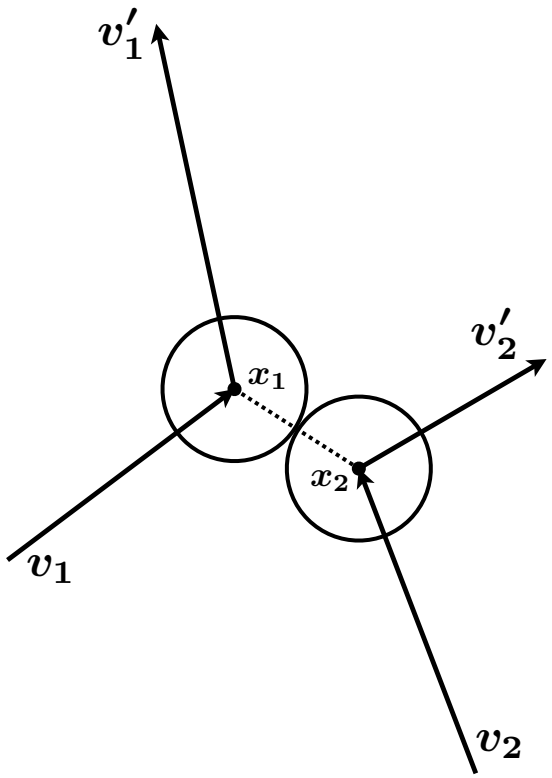

Figure 1: A geometrically accurate collision between two equal mass disks. Fun fact: it's possible to draw this picture using only techniques from Euclidian geometry.

- A 2-torus with radius sides L_x, L_y,

$$\mathcal{T}_{L_x L_y} \;=\; \left\{ (x,y) \in \mathbb{R}^2 \;:\; x = 0 \sim L_x,\; y = 0 \sim L_y \right\} \tag{19}$$

There are other possibilities. But these are some of the more interesting possibilities.

By the way, back in the late 13th century, Dante Alighieri considered the topology of the universe. He suggested that if you could look far enough, you would see the realm of the angels surrounding you in a celestial sphere — the sky. But then, he suggested that the view looks the same from the realm of the angles. Meaning they see our world in their sky. The *separation* of the Earth and heavens is a two-dimensional sphere. This scenario is quite possible from a geometric perspective. Dante conceived of the universe as a *three-spehre*:

$$\mathcal{S}^3_R \;=\; \left\{ (x,y,z,w) \in \mathbb{R}^4 \;:\; x^2 + y^2 + z^2 + w^2 = R^2 \right\}. \tag{20}$$

Of course, he phased it poetically, not mathematically. But a three-sphere he described. The interesting thing is that today, we *might* actually live in a universe with this topology. It would be *much* larger than Dante could imagine. And there wouldn't be a clear division between the heavens and Earth into two equal hemispheres. But astronomical observations have not determined the large-scale topology of our universe. If the universe is a three-sphere, then our solid Earth is analogous to a solid disk sitting on a two-dimensional sphere's

surface. The disk's boundary is a circle, i.e., a one-dimensional sphere, and the outside of the disk is also the shape of a giant stretched disk. You can make an n-sphere by gluing two n-dimensional disks together at their boundaries. Dante knew this intuitively.

You can use any geometry for the Void you like. But of all of them, I like the disk the best. There is only one wall. And the condition for a collision with the wall is very similar to the condition for particle collisions.

$$|x_i(t^*)|^2 + |y_i(t^*)|^2 = R^2. \tag{21}$$

You can reuse your code to compute this time if you are careful. Also, this always has a solution with $\Delta t^* \geq 0$, unlike the two-particle case. And the root you want from the quadratic formula is the alternative compared to the root for two-particle collisions.

$$\Delta t^* \;=\; \frac{-\langle \underline{x}_i, \underline{v}_i \rangle + \sqrt{\langle \underline{x}_i, \underline{v}_i \rangle^2 + (R^2 - ||\underline{x}_i||^2)\,||\underline{v}_i||^2}}{||\underline{v}_i||^2} \tag{22}$$

In the case a particle collides with a real wall, it rebounds in a direction reflected across the wall-normal direction. You can think of the wall as being stationary and having infinite mass. And the normal direction at the wall is just the particle coordinate. Therefore

$$\underline{v}_i' \;=\; \underline{v}_i \;-\; \frac{2}{R^2}\, x_i \cdot \underline{v}_i\, x_i. \tag{23}$$

You might also find conditions for collisions in a rectangle. They are not difficult. I simply like the circle better. You should try it if you want.

Originally, I thought the 2-torus was going to be the simplest. But there are especially tricky things about it. *Particles can wrap around an infinite number of times.* This problem is not *usually* horrendous. But imagine a system with only two particles. They might pass through space an arbitrary number of times before colliding. Solving for this time requires considering all integer multiples of the lattice. It's possible but difficult. I don't recommend it.

1 PROFIT – An engine of computation

You need to write code that can simulate the hard-sphere collision process described above. You can do it however you like within reason. Here are a few thoughts about the process.

- Normally, this problem is solved with one of two main algorithms: Time-based or Event-based. However, the Time-based method is essentially what the soft sphere problem uses. Therefore for this project, the idea is to use only the Event-based method, which I'll describe momentarily.
- The main purpose of this project is to simulate a moderately large number of Atoms (e.g., possibly $\approx 10^4$-10^5).
- Simulating on a single processor would typically only allow a few hundred up to about 1,000 Atoms. However, if you distribute the work across processors, you'll be able to scale up your simulation to respectable sizes.

Event-based algorithm

This scheme is quite elegant — it almost reduces the computational cost to roughly linear in the number of Atoms. Here's how it goes:

Startup

- Start by computing the expected *absolute* collision time t^* for all pairs of Atoms, i, j. This requires $\mathcal{O}(N^2)$ cost, but it's the last time we'll have to compare everyone.
- Put the collision times into a *heap* data structure. This is a big list with the following data

$$H \;=\; \left[\begin{array}{l} (t_0^*, \tau_0, i_0, j_0) \\ (t_1^*, \tau_1, i_1, j_1) \\ (t_2^*, \tau_2, i_2, j_2) \\ (t_3^*, \tau_3, i_3, j_3) \\ \quad\vdots \\ \end{array}\right] \tag{24}$$

  By definition, assume $i > j$.
- Suppose

$$H_k \;=\; (t_k^*, \tau_k, i_k, j_k)\,. \tag{25}$$

  The heap is sorted according to the rule that

$$t_k^* < t_{2k+1}^*, \qquad \text{and} \qquad t_k^* < t_{2k+2}^*. \tag{26}$$

Most programming languages have the built-in ability to create and work with heap data structures. There will be "magic words" like `push` and `pop` that allow you to put things in and take them out so that the first entry is the one with the highest priority. The computer does everything efficiently under the hood.

- In each heap entry, you also need to include τ_k, which is the absolute time the entry became included in the heap. When you solve for the time *increment* to the next collisions you are computing $\Delta t^* = t^* - \tau$.

- For each Atom, i make a list of the last time it collided with something.

$$L \;=\; \left[\tau_0^*, \tau_1^*, \ldots, \tau_{N-1}^*\right] \tag{27}$$

where the ith element in the list $L_i = \tau_i^*$ is the last time Atom i collided with anything. At the start of the simulation, you can set all the times to some negative value.

Ready, set, go!

Once you have your heap setup, do the following

- Assume the current time is t_n.

- Pop the first element from H,

$$H_0 \;=\; (t_0^*, \tau_0, i_0, j_0)\,. \tag{28}$$

The word "pop" in the context of a heap means to take the first item out of the list and rearrange the list, so the new first item is now the most important one. But the computer does this for you.

- Check to see if Atom i_0 or j_0 has already participated in a collision since τ_0. This means checking if $\tau_{i_0}^* > \tau_0$ or $\tau_{j_0}^* > \tau_0$. If either of these conditions is satisfied, H_0 is an invalid event. We simply discard it and pop another event out of the list. We keep doing this until we find a valid event.

- Advance time and the positions of the particles to $t_{n+1} = t_0^*$.

- Recompute the new velocities according to the elastic scattering rule.

- For Atoms i_0, j_0, compute the next collision time data with all other Atoms and the walls. And put these into the heap.

Parallelism

- Before attempting a parallel code, I highly recommend you first build a serial version of the above algorithm. You might even experiment with some of the dynamical challenges to ensure you understand the method's basic premise.

- Simultaneously with creating a serial version of your code, you should experiment with the main parallel aspect of your project: ***using heap data structures across processors***.

- You can find *heaps* or information on the internet about heaps and parallel implementation. For starters, here is a classic paper about the subject:

  Parallel Heap: An Optimal Parallel Priority Queue
  Deo, N. & Prasad, S. *The Journal of Supercomputing*, 6, 87-98 (1992)
  `https://link.springer.com/article/10.1007/BF00128644`

- After you have a serial version of your code working and you understand parallel heaps, then the only thing left is to adapt your code to run on multiple processors.

- This project allows you to experiment and explore fundamental ideas in physics. However, if accurately modelling statistical ensembles of realistic molecules, plasmas, or self-gravitating objects, then the hard sphere approximation is not for you. The aspect of this model that makes it especially interesting is that continuous-time dynamics exactly becomes a discrete map for the entire space-time event structure. More "computer-sciencey" ideas like event tracking and sorting come into the picture. Having a command of these concepts is becoming more and more useful in our modern world, e.g.modelling hard-sphere dynamics more like modelling web traffic than modelling a fluid.

2 FUN – Exploring the dynamics

This part is somewhat open-ended. Moreover, exploring the dynamics is mostly your reward for implementing things well. The real point of the project is to get it working in parallel. Nevertheless, here is a list of possible things to consider

- If you (*(i)* start with some initial conditions, (*(ii)* run for some amount of time, (*(iii)* stop the simulation and reverse all the velocities of your Atoms, you should be able to recreate the initial conditions. This is an important test of your code. You all should absolutely do this first. You won't get the *exact* same answer, but you should be very close if you don't run too long before reversing. If you can't get this to work, it's a good indication that you have a bug in your code.

- Start with well-ordered initial conditions. But (perhaps) with a single atom moving somehow to cause it to interfere with the well-ordered Atoms. One definition of the *temperature* is *variance* of the velocities. If you begin with two populations streaming at each other with a constant velocity, that is a situation with effectively zero temperature.

After some time, the two populations (each moving with uniform but opposite speed) will have mixed. At late times, you should expect to see a Gaussian distribution with a *width* given by the total kinetic energy in the systems. This is the temperature. Here is a big question: how is temperature born? When can you say it's well-defined? What interesting things can you say before this stage but after the system has started to interact? This process is called *thermalisation*. If you throw an object at a wall fast enough, it will thermalise upon impact. This is what's going on; Atoms randomly dispersed into a Gaussian distribution of velocities. You can cook an egg if you have a good enough throwing arm!

The temperature is very subtle in some ways. In the simplest interpretation, it's proportional to the system's average kinetic energy. But if you split the atoms into two populations, you could have one temperature for each population. And rather than the total energy, you could define it as the energy after subtracting the average speed. This is why a bunch of Atoms all moving in unison has zero temperature. But after the two populations interact, they should both come to the same temperature. This is why thermometers work. Because if you generalise the notion of temperature away from strict kinetic energy, you can say that temperature is the quantity that two objects share when they are in contact with each other. You don't need to know what temperature is *philosophically* to know that if the lake is 5 Celsius, and you put your beverage can in the lake; then after some time, your beverage will be 5 Celsius. I like cold beverages. Temperature is great!

- Along with the temperature, you'll want to measure the pressure. This will be the average force per unit length on the outer walls.

- As I said before, your simulation should be good enough to recreate well-ordered initial conditions even from a highly disordered state. But what if you slightly perturb just one Atom by (e.g.) 1% in angle? How long can you run back the clock before the state looks nothing like what you started with? Notably, the system is technically *reversible* but not *stably reversible*.

- You could consider how the dynamics mix angular momentum. The angular momentum of a single particle is

$$\mathcal{L}_i \;=\; m_i\, \underline{x}_i \wedge \underline{v}_i \;=\; m_i \left(x_i v_i - y_i u_i\right). \tag{29}$$

  You should convince yourself that before and after a collision

$$\mathcal{L}'_i + \mathcal{L}'_j \;=\; \mathcal{L}_i + \mathcal{L}_j. \tag{30}$$

  The same is true after a collision with the wall *provided it's a circular domain*. The system conserves total angular momentum, but this doesn't mean it doesn't get mixed up. It might be worthwhile to look at how this happens. Also, collision with a rectangular wall alters angular momentum. In a rectangular container, you could watch an initial amount of angular momentum go away.

- You can look at a single large heavy Atom sitting in the middle of a domain being pummelled by a large number of smaller light atoms. You should see the position of the large Atom walk around randomly. You might even call it a "random walk". This picture is what Einstein considered when he described Brownian motion of a pollen grain in a petri dish. In this case, the pollen grain is the large particle, and the small particles are the light atoms. The astonishing thing about this is that you can actually see the pollen move around directly under a pretty ordinary microscope. But the speed it moves around is governed by the speed and size of the atoms. This was one of the first direct pieces of evidence for the reality of atoms.

- This one is a wee bit of a challenge. But it would be amazing if you could make it work. You should think about how you can make the container a function of time. In general, this can pose several challenges to the algorithms. However, I think there is a nice safe way to do it. Here's how it goes.

  1. Rather than use a single heap for all the collisions, you should make *two separate heaps*; one for two-particle collisions, the other for particle-wall collisions.
  2. Fix the domain size and run the simulation until it gets nice and equilibrated; say time $t = T$. You can use a rectangular or circular domain; either will work equally well for this experiment.
  3. After running until $t = t_{\rm eq}$, pause the simulation.
  4. When you restart the simulation, you should assume (e.g.) radius varies linearly in time

  $$R(t) \;=\; R_0 + \frac{t-T}{\Delta t}(R_1 - R_0). \tag{31}$$

  This formula should stay valid for $T < t < T + \Delta t$. It is important that Δt is much longer than a typical collision time. It's probably best to have at least a few thousand collisions in the amount of time it takes to move the radius by (e.g.) $\approx 10\%$. These are all rough figures, but hopefully, you get the point.

  5. Before restarting the simulation, you should *completely recompute* the heap for wall collisions. You'll now need to solve

  $$||\underline{x}_i(t^*)|| \;=\; R(t^*). \tag{32}$$

  But after you recompute the heap for the wall collisions, you can keep replenishing it the way you did before with the stationary wall or the moving particles, i.e., one collision at a time. Also, recomputing the wall heap is relatively inexpensive. It takes roughly the time needed to recompute pairwise new collisions each time interval.

  6. Keep running until you reach $t = T + \Delta t$. At that point, recompute all the wall heap assuming the radius is fixed $R = R_1$.

  This has a few moving parts (you see what I did there?). But it's straightforward in principle. Being able to move the walls allows you to study several interesting

phenomena. Most notably, you are *not* keeping the energy fixed in the process. From simple arguments, you can deduce that

$$\textit{velocity} \times \textit{diameter} \;\approx\; \textit{constant} \tag{33}$$

In two dimensions, this should imply that

$$\textit{energy} \times \textit{area} \;\approx\; \textit{constant} \qquad \Longleftrightarrow \qquad \textit{pressure} \;\propto\; \textit{density}^2 \tag{34}$$

These are things you can confirm empirically.

- If you think that moving the walls very slowly is a bit contrived, let me put it this way. Seldom in your life have you ever moved anything fast compared to molecules; a bullwhip is a notable exception. Almost by definition, atoms and molecules travel at the speed of sound. The speed of sound is how fast molecules move. Their movement is how the information gets propagated.