Numerical Algorithms

Alexander Gillies

November 21, 2016

1

Start with an arbitrary number of particles and timesteps, using a random number generator each particle is given an x, y and z coordinate. The interaction will start by iterating through every particle in the simulation, for each one calculating the euclidean distance from the current particle to each other particle and using this distance to calculate the force on the current particle, this will then be split into its constituent parts, x,y and z. These are totalled to give the total force in each direction for each particle. Whilst iterating through each particle pair, the shortest distance between any particle pair is stored, this will be used to calculate the timestep, initially the time step is set to the minimum possible value. A variable timestep is used because of the tendency of particle to accelerate together, due to the implementation it will do this in jumps, if the timestep is too large then the particle will pass each other without the particles ever being close enough to repel and will thus not be an accurate model. The way that I calculate the timestep is based on experimental data, I set two particles at closer and closer distances together and set a timestep that I thought was acceptable based on paraview play speed, I then created a stepped timestep such that if the shortest distance between any two particle pairs was between two thresholds then the timestep would change based on this, and when the distance dropped bellow the lower threshold then the timestep would change, this was jerky with the particles accelerating together then coming to a halt as the timestep changed, this was done several times as the distance between any two particles lessened. To overcome the stepping I plotted threshold/timestep points and used a line of best fit to give me the equation of the line that would best fit the thresholds I had found. The equation happened to be the shortest distance$\hat{}$(2.7469) $10\hat{}$(10). This was accurate above a distance of 0.00015 at this point just the minimum distance was used as the timestep, this allows high accuracy at very short distances. The only other feature of the simulation is periodic bounding, I had done this by modulo mathematics but this introduced a few issues, so decided that comparison was faster and more reliable. The comparisons are done is such a way that they are used every time

a particle has its coordinates changed. The particles will have their coordinates changed after all the forces have been calculated for every particle, this improves accuracy as the first particle doesn't have it's coordinates changed then this new position is used for the other particles in the simulation. An improvement that I did make was to reduce the number of euclidean distances calculated was to use force calculated on the current particle on the other particle as well, this has the affect of halving the number of euclidean distance calculation and force calculations that have been done. The way I have implemented wrap around forces is to iterate through the 27 or 7 different dimensions, dependent on the file, I have run both to test run times. Each of the dimensions corresponds to an index in an array that stores the change in coordinates that a particle would have to go through to translate to the corresponding mirror particle, this mirror particle is then used to calculate the force on the current particle from the wrap around in that direction. If this were to be made more accurate then the wrap around would be done for an infinite number of wrap around forces, this is infeasible so I have done it for both 27 and 7.

2

- N = 2, 1 at 0.4 2 at 0.6 with no velocity - oscillation at variable time steps - at low time steps the particles initially don't move at all for a very long time then when they do eventually come together and oscillate as they do in the variable timestep simulation - at high time steps the particles will pass each other then come back and this will cause some issues - they will initially come together and then each of the different timesteps will have it's own effect, if they oscillate they will do so around the 0.5 mark - - at 0.1 and 0.9 they will do the same but the oscillation point will be 0.0 due to the periodic bounding conditions - everything else will be the same - run all the Ns - talk about this when we have results - what happens - how long does it take to compute

3

The data structure I used is an variable length array of vectors, each vector corresponding to one of the particles present. At set time steps, these vectors are populated with two pieces of data for the corresponding particle, the particle index and the index of the translations done to the current particle. At any other timesteps, the vector for each particle is iterated through and the distances and forces are only calculated for any particle that is present in the vector. This is so that any particle that is not close enough to the current particle is not calculated as in theory the force exerted by a particle this far off is negligible.

- talk about what we implemented, verlet lists - the results doesn't change for variable time steps - for large time steps - - for small timesteps - - it is still reasonably correct - very significant run time - show experimentally - - Taylor expansion?