Computer Modelling Design Document

Overview

This program will describe N-body systems interacting through a Lennard-Jones pair potential. We will simulate, using periodic boundary conditions, a Lennard-Jones solid, fluid and gas and investigate equilibrium properties of the simulated systems.

This document describes the classes written for this program, the classes we will use and how they interact with each other. All code is written in Python.

Class Layout

(to be added later)

Particle3D Class

Each instance represents a particle in 3D space. The particle is represented by it’s position, velocity and mass.

Properties

Name	Type	Notes
pos	np.array(3)	Position of the particle relative to the origin, consisting of x, y and z coordinates in that order.
vel	np.array(3)	Velocity of the particle, consisting of x, y and z velocities in that order.
mass	float	The mass of the particle.

Constructor

Arguments	Notes
float m, np.array(3) vel, np.array(3) pos	Creates a particle with mass m, at position pos moving with velocity vel.

Methods

__str__()
Returns a string output in the correct format for a VMD trajectory file.

kineticEnergy()
Returns the kinetic energy of the particle.

$$E_k = \frac{1}{2} \cdot m \cdot v^2$$

magVel()
Returns the magnitude of the particles velocity.

$$|v| = \sqrt{v_x^2 + v_y^2 + v_z^2}$$

magPos()
Returns the magnitude of the particles position from the origin.

$$|r| = \sqrt{r_x^2 + r_y^2 + r_z^2}$$

leapVelocity(dt, force)
First order velocity update, given a force vector and timestep.

$$v(t+dt) = v(t) + dt \cdot f(t)/m$$

leapPos1st(dt)
First order position update, given a timestep.

$$r(t+dt) = r(t) + dt \cdot v(t)$$

leapPos2nd(dt, force)
Second order position update, given a force vector and timestep.

$$r(t+dt) = r(t) + dt \cdot v(t) + dt^2 \cdot f(t)/2m$$

Static Methods

createParticle(File handle inFile)
Creates a particle from a file entry - taking a file handle as input and returning a Particle3D instance. The file handle should be in the format below:

$v_x\ v_y\ v_z$

$r_x\ r_y\ r_z$

mass
label

vectorSeparation(Particle3D p1, Particle3D p2)
Returns the vector separation between 2 particles as a np.array(3).

$\overline{r_{12}} = \overline{r_1} - \overline{r_2}$

scalarSeparation(Particle3D p1, Particle3D p2)
Returns the scalar separation between 2 particles.

$r = |\overline{r_{12}}|$

particleForce(Particle3D p1, Particle3D p2, r_c)
Returns the force vector between 2 particles. If the radii is below a cutoff of r_c (in reduced units) then the this method will return 0.

$F_1(r_{12}) = 48\left[1/r^{14} - 1/2r^8\right](\overline{r_1} - \overline{r_2})$ where (scalar) r denotes the distance between the particles.

particlePotEnergy(Particle3D p1, Particle3D p2, r_c)
Returns the potential energy between 2 particles. If the radii is below a cutoff of r_c (in reduced units) then this method will return 0.

$U(r) = 4\left[1/r^{12} - 1/r^6\right]$ where (scalar) r denotes the distance between the particles.

velocityUpdateList(1-dimensional array of Particle3D instances)
Takes in a list of Particle3D instances and updates the velocity of each one by considering the forces due to all the other particles in the list. The velocity update will be implemented using the leapVelocity method, but the combined force vector will look like:

$f_1(t) = f_{12}(t) + f_{13}(t) + \dots$

positionUpdateList(1-dimensional array of Particle3D instances)
Takes in a list of Particle3D instances and updates the position of each one using the 2nd order position update method and considering the forces from all other particles in the list. The force vector will be the same format as shown above.

totalParticleEnergy(1-dimensional array of Particle3D instances)
Takes in a list of Particle3D instances and returns the total energy due to the particles individual kinetic energies and all atom pair interactions. Individual kinetic energies will be found using the

kinetic energy method, and atom pair energies will be calculated using the static potential energy method.

Simulation Class

This class contains the main program that simulates the N-body system interacting through LJ pair potential, with given initial conditions.

This class does not contain any properties, constructors or instance methods.

main(str[] argv)

The main method reads in