

Lab 3 - Many particle simulations

Thursday 11 September - Due: Thursday 18 February

We will learn some new features of Python in this lab which will focus primarily on upgrading the `balls_in_boxes.py` code we discussed in class to allow us to draw the box walls, and to create and simulate an arbitrary number of balls with arbitrary force interactions.

1 Drawing walls

Download the two codes `balls_in_boxes.py` and `integrate.py` from the labs link on the PHY307 page to your SU filespace. These are the two python codes needed for simulating two balls in a box interacting with a short distance repulsive force. Make sure you put them into the same folder - I recommend making a folder called `lab3`. Load the former into the python editor IDLE (fire this up by looking in the Windows start menu for SU Academic Departments, then Physics, and look for Vpython).

1. First we would like to write some python code to draw out the enclosing box. We can use the `curve` object to do this. The following code draws one face of a cubical box of side length $2L$ and whose center is at the origin of the coordinates.

```
face = curve(pos=[(-L,-L,-L),(L,-L,-L),(L,L,-L),(-L,L,-L),(-L,-L,-L)],
             color=color.blue, radius=0.005)
```

In general the `curve` will draw line segments between successive pairs of coordinates. Construct similar code to draw another parallel face. You will still need to supply 4 more connecting lines with code like

```
zaxis = curve(pos=[(-L,-L,-L),(-L,-L,L)],color=color.blue,radius=0.005)
```

Insert this code into the program (near the top making sure you define `L` before calling this code).

2. Comment out the lines in which the tracks are drawn to the screen by placing `##` at the beginning of the line

```
balla.track.append(pos=balla.pos)
```

This will make it easier to see whats going on later.

3. Run the code. Make a screen capture of the resultant simulation. (hold down alt and print screen having selected the window). Include your new code in your writeup.

2 Adding many balls

We would like to use this code to simulate systems with many degrees of freedom. We could do this by manually creating many spheres, assigning them all positions, velocities etc but this gets rapidly tedious as the number of particles grows. Instead we will create the balls and assign their properties using some tricks associated with python `lists`.

1. The following code creates an empty list and then loops N times each time creating a sphere object and adding it to the growing list with the `append` function.

```
number_on_side=3
# total number of balls N
N=number_on_side*number_on_side
# initial lattice spacing a
a=(2.0*L)/(number_on_side+1)
system=[]
for n in range(0,N):
    i=(n/number_on_side)
    j=(n-i*number_on_side)
    ball=sphere(radius=0.05,color=color.red)
    ball.mass=1.0
    ball.acc=0.0
    ball.acc2=0.0
    ball.pos=vector(-L+(i+1)*a,-L+(j+1)*a,0.0)
    system.append(ball)
```

Edit the file `balls_in_boxes.py` to add this code after the definition of `L`. Delete the old lines of code responsible for creating `ball1`, `ball2` and assigning their properties. Also delete the original line `system=[ball,ball2]` as this has now been superseded by the final line in the new block of code.

2. Notice that we have now assigned the initial positions of the balls on a *lattice* with `number_on_side` balls in each direction with spacing `a`. This will allow us to use the code to study crystals at low temperature.
3. Let us assign initial velocities at random. This can be accomplished by inserting the line

```
ball.vel=VMAX*vector(random()-0.5,random()-0.5,random()-0.5)
```

after assigning each ball's position. Do this. Set `VMAX` the scale of the initial velocity to 1.0 at the beginning of the code. Also you will need to import the `random` module by adding the line `from random import *` after the other import lines near the top of the file.

4. Run the new code and capture a picture of what you see. Include your new code listing in your writeup.

3 Towards a true molecular dynamics

1. Lets modify the function `newforce()` to model a system with a long range attractive force and a short range repulsion. This gives a crude representation of a typical molecular interaction. The magnitude of the random initial velocities controls the temperature of the system. Edit the code to replace the hardcore repulsion with a force whose magnitude varies with separation r between any two of our ball or particle objects

$$F = \frac{A}{r} \left[2 \left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right]$$

Modify `newforce()` to use this force. Set $\sigma = 0.12$ and $A = 1.0$. The function `pow(x,p)` is the python function which raises x to the power p . This force is of the Lennard-Jones type.

2. Sketch this force as a function of separation r . When is the force zero ?
3. Experiment with changing `vmax`. Do you notice anything as `vmax` is lowered ? You might want to set the initial velocity to have no z-component to keep the system two dimensional and increase the frequency of interactions