Physics 40: Laboratory Seventeen

Tuesday, May 26, 2020

Today's Goals:

Molecular Dynamics in Python

[1] Below is version one of a Molecular Dynamics code for a mass on a spring (no damping) in python. It really looks very similar to a C code: inputs for the number of time steps, mass, initial position, etc followed by a loop to push the position and velocity forward. Note the use of comment lines preceded by '#'. Although I have not emphasized it in this class, commenting your code is really an essential part of good programming.

```
# INPUTS
N =input('Enter N :  ')
dt=input('Enter dt:  ')
k =input('Enter k :  ')
m =input('Enter m :  ')
x =input('Enter x0:  ')
v =input('Enter v0:  ')

# DO THE MD AND APPEND EACH NEW TIME, POSITION, VELOCITY TO ITS ARRAY
for i in range (1,N):
    t=dt*i
    x=x+v*dt
    v=v-k*x*dt/m
    print(t,x)
```

If you called this script "md1.py", then (make sure you are in the same directory/folder where your file is) you can run it by typing:
python ./md1.py

[2] This first version above writes to the screen. We might instead want the data to go to a file. In some operating systems you can redirect output that goes to the screen to a file by using '>' as follows:
python ./md1.py > diadem
The problem with this is that the prompts for input also go to the file 'diadem' so you need to remember them.

You can similarly use the '<' symbol to handle your inputs. (This is convenient if you have a lot of inputs.) Make a file (eg call it 'ravenclaw') containing six lines with the values of N, dt, k, m, x_0, v_0:

```
1000
.01
3.0
0.75
5.0
0.0
```

And then type: python ./md1.py < ravenclaw > diadem
See if all that works in the environment you are using for your codes. It's a useful quick trick to sending output to a file, although not the most careful way to do it.

[3] Another way to write to a file is contained in this MD version two:

```
# OPEN A FILE
scabbers= open("pigwidgeon.dat","w+")

# INPUTS
N =input('Enter N :  \n')
dt=input('Enter dt:  \n')
k =input('Enter k :  \n')
m =input('Enter m :  \n')
x =input('Enter x0:  \n')
v =input('Enter v0:  \n')

# DO THE MD
for i in range (1,N):
    t=dt*i
    x=x+v*dt
    v=v-k*x*dt/m
    scabbers.write("%f   " % t )
    scabbers.write("%f \n" % x )

# CLOSE THE FILE
scabbers.close()
```

Now your data for t and x get written to the file 'pigwidgeon.dat'. This is preferable to using '>' because the prompts for the inputs still come to the screen. You could now use 'google' to plot the data. Fortunately, there is a better way!

[4] Plotting directly in python, step one: Let's make a MD version which stores the trajectory in an array. A good, quick on-line resource for learning python arrays is:

https://www.thegeekstuff.com/2013/08/python-array/

Read it carefully. It explains the syntax in the example below, and also alternate ways of adding/removing data from an array.

```
# THIS IS NEEDED TO USE ARRAYS IN PYTHON:
from array import *

# INPUTS
N =input('Enter N :  \n')
dt=input('Enter dt:  \n')
k =input('Enter k :  \n')
m =input('Enter m :  \n')
x =input('Enter x0:  \n')
v =input('Enter v0:  \n')

# START OFF TIME, POSITION, VELOCITY ARRAYS
time =array('f', [0.])
xsave=array('f',[x])
vsave=array('f',[v])

# DO THE MD AND APPEND EACH NEW TIME, POSITION, VELOCITY TO ITS ARRAY
for i in range (1,N):
    t=dt*i
    x=x+v*dt
    v=v-k*x*dt/m
    time.append(t)
    xsave.append(x)
    vsave.append(v)

# PRINT OUT TIME AND POSITION
for i in range (1,N):
    print(time[i],xsave[i])
```

[5] Making a plot. A good, quick on-line resource for learning python plotting is:

`https://matplotlib.org/users/pyplot_tutorial.html`

Read it carefully. It explains the example below, as well as ways to modify your plots by controlling the types of symbols used, line thickness and colors, titles, etc..

```
# THIS IS NEEDED TO USE ARRAYS IN PYTHON:
from array import *

# THIS IS NEEDED TO MAKE PLOTS IN PYTHON:
import matplotlib.pyplot as plt

# INPUTS
N =input('Enter N :  \n')
dt=input('Enter dt:  \n')
k =input('Enter k :  \n')
m =input('Enter m :  \n')
x =input('Enter x0:  \n')
v =input('Enter v0:  \n')

# START OFF TIME, POSITION, VELOCITY ARRAYS
time =array('f', [0.])
xsave=array('f',[x])
vsave=array('f',[v])

# DO THE MD AND APPEND EACH NEW TIME, POSITION, VELOCITY TO ITS ARRAY
for i in range (1,N):
    t=dt*i
    x=x+v*dt
    v=v-k*x*dt/m
    time.append(t)
    xsave.append(x)
    vsave.append(v)

# PRINT OUT TIME AND POSITION
for i in range (1,N):
    print(time[i],xsave[i])

# MAKE A PLOT
plt.plot(time, xsave)
plt.xlabel('t')
plt.ylabel('x')
plt.show()
```

[6] Making a fancier plot.

```
# THIS IS NEEDED TO USE ARRAYS IN PYTHON:
from array import *

# THIS IS NEEDED TO MAKE PLOTS IN PYTHON:
import matplotlib.pyplot as plt

# INPUTS
N =input('Enter N :  \n')
dt=input('Enter dt:  \n')
k =input('Enter k :  \n')
m =input('Enter m :  \n')
x =input('Enter x0:  \n')
v =input('Enter v0:  \n')

# START OFF TIME, POSITION, VELOCITY ARRAYS
time =array('f', [0.])
xsave=array('f',[x])
vsave=array('f',[v])

# DO THE MD AND APPEND EACH NEW TIME, POSITION, VELOCITY TO ITS ARRAY
for i in range (1,N):
    t=dt*i
    x=x+v*dt
    v=v-k*x*dt/m
    time.append(t)
    xsave.append(x)
    vsave.append(v)

# PRINT OUT TIME AND POSITION
for i in range (1,N):
    print(time[i],xsave[i])

# MAKE A PLOT
plt.figure(1)
plt.subplot(211)
plt.plot(time, xsave)
plt.xlabel('t')
plt.ylabel('x')
plt.subplot(212)
plt.plot(time, vsave)
plt.xlabel('t')
plt.ylabel('v')
plt.show()
```

[HW9-1] Compute the period of oscillation of a mass $m = 3.7$ kg attached to a spring of spring constant $k = 7400$ N/m. Explain what a good choice of time step would be for this problem. Run your MD program and make a plot showing that it gives the correct period. Does it matter what initial position and velocity you use?

[HW9-2] Using energy conservation, compute the amplitude A (the maximum distance from the origin) for a mass $m = 0.2$ kg on a spring $k = 75$ N/m if the initial position $x_0 = 0.1$ m and the initial velocity $v_0 = 1.4$ m/s. Run your code and make a plot showing it is giving the correct A. Similarly, use energy conservation to compute the maximum velocity $v_{\max}$. Run your code and make a plot showing it is giving the correct $v_{\max}$.

[HW9-3] Write a python script which does projectile motion. Consider a ball thrown from the top of the physics building $y_0 = 30$ m at a velocity $v_{x0} = 15$ m/s, $v_{y0} = 20$ m/s. Using equations from a prior physics course, compute the time before the ball hits the ground ($y = 0$), the maximum height reached, and the distance from the base of the building where the ball strikes. Make a set of four plots (using 'subplot') consisting of y vs t; v_y vs t; x vs t; and v_x vs t. Show that your code agrees with your calculations.