PHY321: Classical Mechanics 1

Homework 2, due January 27 (Midnight)

Jan 16, 2023

Practicalities about homeworks and projects.

1. You can work in groups (optimal groups are often 2-3 people) or by yourself. If you work as a group you can hand in one answer only if you wish. **Remember to write your name(s)**!

2. Homeworks are available 10 days before the deadline.

3. How do I(we) hand in? You can hand in the paper and pencil exercises as a scanned document. For this homework this applies to exercises 1-5. The scanned document should be uploaded to D2L. Alternatively, you can hand in everyhting (if you are ok with typing mathematical formulae using say Latex) as a jupyter notebook at D2L. The numerical exercise(s) (exercise 6 here) should always be handed in as a jupyter notebook by the deadline at D2L.

Exercise 1 (10 pt), Forces, discussion questions, test your intuition.

- 1a (2pt) Single force. Can an object affected only by a single force have zero acceleration?

- 1b (2pt) Zero velocity. If you throw a ball vertically it has zero velocity at its maximum point. Does it also have zero acceleration at this point?

- 1c (3pt) Acceleration of gravity. You measure the acceleration of gravity in an elevator moving at a velocity of 9.8m/s downwards. What will you measure?

- 1d (3pt) Air resistance. You throw a ball straight up and measure the velocity as it passes you on its way down. Will the velocity be larger, the same, or smaller if you did the same experiment in vacuum?

Exercise 2 (10 pt), setting up forces, Newton's second law. Useful material here to read is

1. Taylor chapters 1.3 and 1.4 and
2. Malthe-Sørenssen chapters 5.1, 5.2 and 5.3

A person jumps from an airplane, falling freely for several seconds before the person pulls the cord of the parachute and the parachute unfolds.

- 2a (3pt) Identify the forces acting on the parachuter and draw a free-body diagram of the parachuter before the person has pulled the cord.
- 2b (3pt) Identify the forces acting on the parachuter and draw a free-body diagram of the parachuter after the person has pulled the cord.
- 2c (4pt) Sketch the net force acting on the parachuter as a function of time, F(t).

Exercise 3 (10 pt), Space shuttle with air resistance. Useful material here to read is

1. Malthe-Sørenssen chapters 5.1, 5.2 and 5.3

During lift-off of the space shuttle the engines provide a force of 35×10^6 N. The mass of the shuttle is approximately 2×10^6 kg.

- 3a (3pt) Draw a free-body diagram of the space shuttle immediately after lift-off.
- 3b (3pt) Find an expression for the acceleration of the space shuttle immediately after lift-off.

Let us assume that the force from the engines is constant, and that the mass of the space shuttle does not change significantly over the first 20 s.

- 3c (4pt) Find the velocity and position of the space shuttle after 20 s if you ignore air resistance.

Exercise 4 (15 pt), now hitting a golf ball. Useful material here to read is

1. Taylor chapters 1.3-1.6 and
2. Malthe-Sørenssen chapter 6.3-6.4 and 7.1-7.3

Taylor exercise 1.35. The formulae you obtain here will be useful for the numerical exercises below (see exercise 6 below).

Exercise 5 (15 pt), hitting a puck instead. Taylor exercise 1.38.

Exercise 6 (40pt), Numerical elements, moving to more than one dimension. This exercise should be handed in as a jupyter-notebook at D2L. Remember to write your name(s).

Last week we:

1. Analytically mapped 1D motion over some time
2. Gained practice with functions
3. Reviewed vectors and matrices in Python

This week we will:

1. Practice using Python syntax and variable manipulation
2. Utilize analytical solutions to create more refined functions
3. Work in two, three or even higher dimensions

This material will then serve as background for the numerical part of homework 3. The first part is a simple warm-up, with hints and suggestions you can use for the code to write below.

```
# As usual, here are some useful packages we will be using. Feel free to use more and experiment
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
%matplotlib inline
```

In class (the falling baseball example) we used an analytical expression for the height of a falling ball. In the first homework we used instead the position from experiment (Usain Bolt's 100m record run) and stored this information with one-dimensional arrays in Python.

Let us get some practice with this. The cell below creates two arrays, one containing the times to be analyzed and the other containing the x and y components of the position vector at each point in time. This is a two-dimensional object. The second array is initially empty. Then we define the initial position to be $x = 2$ and $y = 1$. Take a look at the code and comments to get an understanding of what is happening. Feel free to play around with it.

```
tf = 4 #length of value to be analyzed
dt = .001 # step sizes
t = np.arange(0.0,tf,dt) # Creates an evenly spaced time array going from 0 to 3.999, with step s
p = np.zeros((len(t), 2)) # Creates an empty array of [x,y] arrays (our vectors). Array size is s
p[0] = [2.0,1.0] # This sets the inital position to be x = 2 and y = 1
```

Below we are printing specific values of our array to see what is being stored where. The first number in the array $r[]$ represents which array iteration we are looking at, while the number after the represents which listed number in the array iteration we are getting back.

```
print(p[0]) # Prints the first array
print(p[0,:]) # Same as above, these commands are interchangeable


print(p[3999]) # Prints the 4000th array


print(p[0,0]) # Prints the first value of the first array


print(p[0,1]) # Prints the second value of first array
print(p[:,0]) # Prints the first value of all the arrays
```

Then try running this cell. Notice how it gives an error since we did not implement a third dimension into our arrays

```
print(p[:,2])
```

In the cell below we want to manipulate the arrays. In this example we make each vector's x component valued the same as their respective vector's position in the iteration and the y value will be twice that value, except for the first vector, which we have already set. That is we have $p[0] = [2, 1], p[1] = [1, 2], p[2] = [2, 4], p[3] = [3, 6], ...$

Here we set up an array for x and y values.

```
for i in range(1,3999):
    p[i] = [i,2*i]
# Checker cell to make sure your code is performing correctly
c = 0
for i in range(0,3999):
    if i == 0:
        if p[i,0] != 2.0:
            c += 1
        if p[i,1] != 1.0:
            c += 1
    else:
        if p[i,0] != 1.0*i:
            c += 1
        if p[i,1] != 2.0*i:
            c += 1

if c == 0:
    print("Success!")
else:
    print("There is an error in your code")
```

You could also think of an alternative way of storing the above information. Feel free to explore how to store multidimensional objects.

Last week we studied Usain Bolt's 100m run and in class we studied a falling baseball. We made basic plots of the baseball moving in one dimension. This week we will be working with a three-dimensional variant. This will be useful for our next homeworks and numerical projects.

Assume we have a soccer ball moving in three dimensions with the following trajectory:

1. $x(t) = 10t\cos 45°$
2. $y(t) = 10t\sin 45°$
3. $z(t) = 10t - \frac{9.81}{2}t^2$

Now let us create a three-dimensional (3D) plot using these equations. In the cell below we write the equations into their respective labels. We fix a final time in the code below.

Important Concept: Numpy comes with many mathematical packages, some of them being the trigonometric functions sine, cosine, tangent. We are going to utilize these this week. Additionally, these functions work with radians, so we will also be using a function from Numpy that converts degrees to radians.

```
tf = 2.04  # The final time to be evaluated
dt = 0.1  # The time step size
t = np.arange(0,tf,dt) # The time array
theta_deg = 45 # Degrees
theta_rad = np.radians(theta_deg) # Converts degrees to their radian counterparts
x = 10*t*np.cos(theta_rad) # Equation for our x component, utilizing np.cos() and our calculated
y = 10*t*np.sin(theta_rad) # Put the y equation here
z = 10*t-9.81/2*t**2# Put the z equation here
```

Then we plot it

```
## Once you have entered the proper equations in the cell above, run this cell to plot in 3D
fig = plt.axes(projection='3d')
fig.set_xlabel('x')
fig.set_ylabel('y')
fig.set_zlabel('z')
fig.scatter(x,y,z)
```

- 6a (8pt) How would you express $x(t)$, $y(t)$, $z(t)$ for this problem as a single vector, $\boldsymbol{r}(t)$?

Then run the code and plot using the array r

```
## Run this code to plot using our r array
fig = plt.axes(projection='3d')
fig.set_xlabel('x')
fig.set_ylabel('y')
fig.set_zlabel('z')
fig.scatter(r[0],r[1],r[2])
```

- 6b (8pt) What do you think the benefits and/or disadvantages are from expressing our three equations as a single array/vector? This can be both from a computational and physics stand point. Use the **Numpy** package to also print the maximum x, y and z components from $\boldsymbol{r}$.

Complete Exercise 4 above (Taylor exercise 1.35) before moving further. (Recall that the golf ball was hit due east at an angle θ with respect to the horizontal, and the coordinate directions are x measured east, y north, and z vertically up.)

- 6c (8pt) What is the analytical solution for our theoretical golf ball's position $\boldsymbol{r}(t)$ over time from Exercise 4? Also what is the formula for the time t_f when the golf ball hits the ground? Use this to develop a program with a function called for example Golfball that utilizes our analytical solutions. This program should take in an initial velocity and the angle θ that the golfball was hit with in degrees. It should also produce a 3D graph of the motion. You need also to find the maximum values for x, y and z.
- 6d (8pt) Given initial values of $v_i = 90m/s$, $\theta = 30°$, what would our maximum x, y and z components be?
- 6e (8pt) Given initial values of $v_i = 45m/s$, $\theta = 45°$, what would our maximum x, y and z components be?