1st Report - *Introduction to Linux and Python*

Simulació de Sistemes Nanomètrics - *Nanociència i Nanotecnologia - 22/23*

Daniel Bedmar Romero 1565494

1 List the most important Linux commands used

One of the main goals of this session was to do an introduction to Linux, in particular, an introduction of the Linux Terminal and some of its basics commands. Here I list the most important commands introduced during the session and a brief explanation of what they do:

- *pwd*: pwd stands for Print Working Directory, therefore this command tells us the working directory, which is how the directory in which the terminal is working or placed is called.
- *ls*: This command gives us a list of what is located inside the working directory.
- *ls lst*: This command also gives us a list of what is located inside the working directory but with additional information.
- *cd "Directory Path"*: This command allows us to move to another directory by writing the path of the directory which we want to go after the cd. The "path" of a directory is the list of all the superior directror
- *cd "Sub-Directory Name"*: This command allows us to move from the working directory to a directory which is located inside it by writting its name after the *cd*, these directories are called sub-directories
- *cd ..*: With this command we can move from the working directory to the directory that contains the working directory, which is called "parent directory".
- *mkdir "Name"*: This command creates a new directory with the name we write after *mdir*.
- *lscpu*: This command give us a list of technical aspects of the computer
- *clear*: With this command we can clear the code we have typed in the terminal.
- *python3*: With this command we can start running scripts with Python 3.

2 Modification of a code

I've chosen to modify the parabolic flight code introducing the possibility of an explosion of the projectile. My goal was that every time you execute the code you can chose the launch velocity and the time at which explosion happens, apart from the launching angle. Also, I wanted the plot to show what would've been the trajectory without the explosion with dashed lines and also where the explosion took place with a red cross.

2.a Physical considerations

I based my modification in a simple physics problem used to explain the conservation of the kinetic momentum if there are no external forces. In this hypothetical case we have a projectile that, at some point can explode and divide into two equal pieces, one of them stops it's motion due to the explosion and the other one gets pushed by the explosion, this last piece is the one that we will track.
To solve this problem is important to notice that kinetic momentum is conserved because the explosion is an internal force of the projectile and that just after the explosion, the trajectory of the projectile that receive the explosion boost can be thought as a new parabolic flight. This "new" parabolic flight will have its initial position where the explosion took place (equations 1 and 2) and the initial velocity is given by the kinetic momentum conservation (equations 3 and 4). The time that we will use can be calculated as $t' = t - t_{exp}$.

$$x_{exp} = v_0 cos(\theta) t_{exp} \tag{1}$$

$$y_{exp} = v_0 sin(\theta) t_{exp} - \frac{1}{2} g t_{exp}^2 \tag{2}$$

$$p_x = 2mv_0 cos(\theta) = 0 + mv_{x,exp} \Rightarrow$$

$$v_{x,exp} = 2v_0 cos(\theta) \tag{3}$$

$$p_y = -2m\left[v_o sin(\theta) - g t_{exp}\right] = 0 + mv_{y,exp} \Rightarrow$$

$$v_{y,exp} = 2\left[v_0 sin(\theta) - g t_{exp}\right] \tag{4}$$

With all this considerations we arrive to the "new" parabolic flight equations after the explosion:

$$x' = x_{exp} + v_{x,exp} t' \tag{5}$$

$$y' = y_{exp} + v_{y,exp} t' - \frac{1}{2} g t'^2 \tag{6}$$

2.b The new code

In order to implement this in the code I had to modify the *draw trajectory* function. First of all, I had to increase the flight time because now the initial velocity is 2 times greater and, therefore, the projectile can fly 2 times longer. Then I also had to calculate where the explosion took place.
In order to have the plot with the explosion and the plot without it I created two more lists for the plot with the explosion, this lists will be filled by the points calculated with the normal parabolic flight equations until the time is equal to the explosion time but, at that time, the lists will be filled by the points calculated with the equations 5 and 6. Of course, I also had to add the explosive plot and the explosion point and also modify the old one so it appears with dashed lines.
Finally, I modified the code so the launch velocity and the explosion time are asked to the user when the code is executed. This modifications can be seen in the Figure 2 or in my Github repository [1].

For example, if the user wants to now the trajectory of the projectile when the launch angle is 45^{o}, the launch velocity is 33m/s and different explosion times, for example; 0 seconds, 1 seconds, 2 seconds and 3 seconds would get the plots shown in Figure (1).

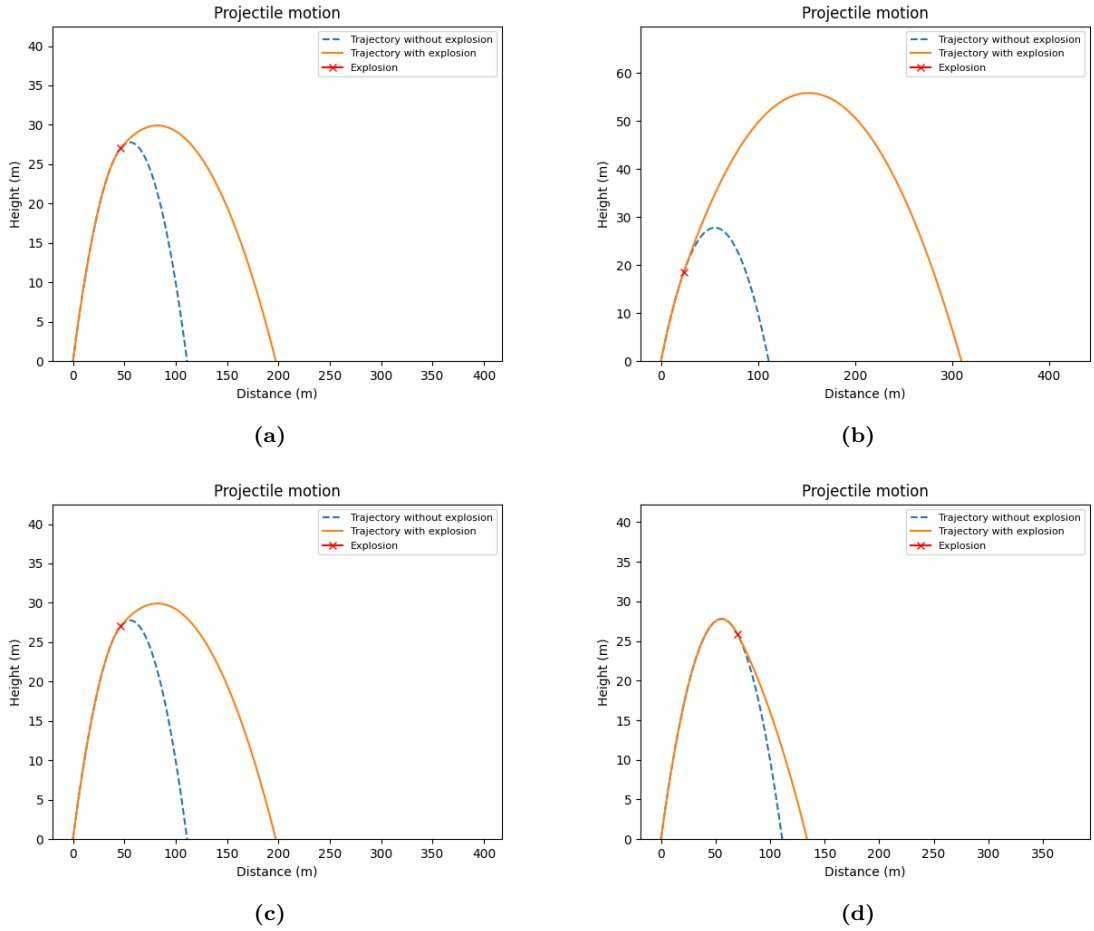

Figure 1: Plots for a launch with 45^{o} launch angle, 33m/s launch velocity and with an explosion time of 0s, 1s, 2s,and 3s in (a), (b), (c) and (d) respectively

References

[1] "Github repository with the new code." https://github.com/DaniBedmar/Nanometric-Systems-Simulation/tree/main/Project_01_Parabolic_motion.

```python
def draw_trajectory(u, theta,texp):
    #convert angle in degrees to rad
    theta = np.radians(theta)
    #gravity acceleration in m/s2
    g = 9.8
    # Time of flight
    t_flight = (4*u*np.sin(theta))/g
    # find time intervals
    intervals = np.arange(0, t_flight, 0.001)
    #Calculate where will the explosion occur
    xexpl=u*np.cos(theta)*texp
    yexpl=u*np.sin(theta)*texp-0.5*g*(texp**2)
    # create an empty list of x and y coordinates (non-explosive)
    x = []
    y = []
    #Do a loop over time calculating the coordinates for the non-explosive trajectory
    for t in intervals:
        x.append(u*np.cos(theta)*t)
        y.append(u*np.sin(theta)*t - 0.5*g*t*t)
    # create an empty list of x and y coordinates (explosive)
    xexp = []
    yexp = []
    #Do a loop over time calculating the coordinates for the explosive trajectory
    for t in intervals:
        if t <= texp:
            xexp.append(u*np.cos(theta)*t)
            yexp.append(u*np.sin(theta)*t - 0.5*g*t*t)
        else:
            te=t-texp
            xexp.append(xexpl+2*u*np.cos(theta)*te)
            yexp.append(yexpl+2*te*(u*np.sin(theta)-g*texp)-0.5*g*(te**2))

    #Plot the trajectories
    plt.plot(x, y, linestyle='dashed')
    plt.plot(xexp, yexp)
    plt.plot(xexpl,yexpl,marker='x',color='r')
    plt.ylim(bottom=0)
    plt.xlabel('Distance (m)')
    plt.ylabel('Height (m)')
    plt.title('Projectile motion')
```

(a)

```python
print("Parabolic motion of a projectile\n")

#Ask the user for angle, explosion time and velocity
print("Enter desired launch angle in degrees (recommended 45 degrees):")
theta=float(input())
print("Enter desired launch velocity in meters/s:")
u=float(input())
print("Enter desired explosion time in seconds:")
texp=float(input())

#
draw_trajectory(u, theta, texp)


# Add a legend and show the graph
plt.legend(['Trajectory without explosion', 'Trajectory with explosion','Explosion'],loc='upper right',prop={'size': 8})
plt.show()
```

(b)

Figure 2: In (a) the new draw trajectory function and in (b) the new main part of the code