

Phy321

Q1) a) A conservative force is dependent on 4 main ideas.

i) path independence: work done by the force on an object moving between two points is independent of the path taken, it only depends on the final and initial position

ii) potential energy: force is equal to the negative difference of the potential energy

iii) The sum of the forces must equal zero

$$b) W = \Delta K = W = K_f - K_i$$

$$W = -\Delta U = W = -(U_f - U_i)$$

$$\text{So, } -U_f + U_i = K_f - K_i$$

$$K_f + U_f = K_i + U_i \Rightarrow E_f = E_i$$

$$c) P = \sum_{i=1}^N p_i = \sum_{i=1}^N m_i v_i$$

Newton 2nd law:

$$\frac{dp_i}{dt} = F_i^{\text{net}} \quad \text{where } F_i^{\text{net}} = \sum_{j=1, j \neq i}^N F_{ji}$$

$$\frac{dP}{dt} = \sum_{i=1}^N \frac{dp_i}{dt} = \sum_{i=1}^N F_i^{\text{net}} \Rightarrow \frac{dP}{dt} = \sum_{i=1}^N \sum_{j=1, j \neq i}^N F_{ji} \quad \text{--- } \textcircled{1}$$

Where Newton 3rd law says:

$$F_{ji} + F_{ij} = 0$$

so $\textcircled{1} = 0$ thus: $\frac{dP}{dt} = 0$ which implies
that:

$$P(t) = P_0$$

therefore the total linear momentum is conserved

from the lecture notes.

2)

a) we know L: angular moment is defined as:

$$\text{angular moment } \vec{L} = \vec{r} \times \vec{p} = m(\vec{r} \times \vec{v})$$

taking the derivative:

$$\frac{d\vec{L}_{sys}}{dt} = \frac{d}{dt}(m(\vec{r} \times \vec{v})) = m \frac{d\vec{r}}{dt} \times \vec{v} + m\vec{r} \times \frac{d\vec{v}}{dt}$$

$$\frac{d\vec{L}_{sys}}{dt} = \vec{r} \times m \frac{d\vec{v}}{dt} = \boxed{\tau = \vec{r} \times \vec{F}}$$

b) $L_i = \vec{r}_i \times \vec{p}_i$, so angular is: $L = \sum_{i=1}^N \vec{L}_i = \sum_{i=1}^N \vec{r}_i \times \vec{p}_i$

taking the time derivative:

$$T_i = \vec{r}_i \times \vec{F}_i^{net} \leftarrow \text{the net force of the whole system.}$$

$$\Rightarrow L = \sum_{i=1}^N T_i = \sum_{i=1}^N \vec{r}_i \times \vec{F}_{ext,i} + \sum_{i=1}^N \sum_{j=1, j \neq i}^N \vec{r}_i \times \vec{F}_{ji}$$

internal forces pair cancellation.

$$\vec{r}_i \times \vec{F}_{ji} + \vec{r}_j \times \vec{F}_{ij} = 0 \rightarrow \sum_{i=1}^N \sum_{j=1, j \neq i}^N \vec{r}_i \times \vec{F}_{ji} = 0$$

so,

$$L = \sum_{i=1}^N \vec{r}_i \times \vec{F}_{ext,i} \rightarrow \tau = \frac{dL}{dt}$$

⇒ Two conditions need to be satisfied.

1) $F_{ij} = -F_{ji}$

2) $F_{ij} \propto (r_i - r_j)$

• This works cuz Newton 3rd law
ensure that that the internal forces
cancel each other.

• Central forces: By making the internal
forces act along a line connecting the
interacting objects. that means:

$$(r_i - r_j) \times F_{ij} = 0$$

$$\textcircled{1}) v(x, y, z) = A \exp\left(-\frac{x^2 + z^2}{2a^2}\right)$$

From the book:

	x-comp	y-comp	z-comp
∇	$\frac{\partial}{\partial x}$	$\frac{\partial}{\partial y}$	$\frac{\partial}{\partial z}$
F	F_x	F_y	F_z

$$\nabla \times F = \begin{vmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ F_x & F_y & F_z \end{vmatrix} = \begin{vmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ \frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z} & \frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x} & \frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y} \end{vmatrix}$$

Just in case we need it.

$$\textcircled{2}) \nabla V = \left(\frac{\partial V}{\partial x}, \frac{\partial V}{\partial y}, \frac{\partial V}{\partial z} \right)$$

So, the force is

$$F = -\frac{A_x}{a^2} \exp\left(-\frac{x^2 + z^2}{2a^2}\right) \hat{x} + 0 - \frac{A_z}{a^2} \exp\left(-\frac{x^2 + z^2}{2a^2}\right) \hat{z}$$

$F = -\nabla V$ so,

$$F = -\left(\frac{\partial V}{\partial x} \exp\left(-\frac{x^2 + z^2}{2a^2}\right) \hat{x} - \frac{\partial V}{\partial z} \exp\left(-\frac{x^2 + z^2}{2a^2}\right) \hat{z}\right)$$

Satisfying the first condition.

\rightarrow Second condition $\text{curl } F = 0$ or $\nabla \times F = 0$

We already assumed our $\text{curl } F$

so

$$\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z} = 0 \quad \text{since } F_z \text{ don't depend on } y \quad \text{and } F_y \text{ is } 0$$

$$\frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x} = 0 \quad \text{using wolfram alpha this would give } \underline{\text{zero}}$$

Now the ~~the~~ east one.

$$\frac{\delta F_b}{\delta x} - \frac{\delta F_x}{\delta b} = 0 \quad \text{it's obvious why.}$$

So $\boxed{\text{curl } \mathbf{F}(0,0,0)}$

Since the force is derivable from the potential and the curl F. we have satisfied the conditions that the force is conservative.

b) For this question and the next we are using the same concept.

Noether's theorem: if the potential is independent of a particular coordinate, the corresponding component of the momentum along that coordinate is conserved.

Now, $V(x, y, z)$ is dependent on x and z .
we can see that by the term $(-\frac{x^2 + z^2}{2a^2})$
 z and x are not invariant under translation
of their respected coordinates.

where y is invariant under translation.

Conclusion:

p_x is not conserved

p_y is conserved

p_z is not conserved

c) Using the same approach
rotational symmetries of the potential function
Noether's theorem of potential invariance

$$L = (L_x, L_y, L_z), L_x = y p_z - z p_y$$

$$L_y = z p_x - x p_z$$

$$L_z = x p_y - y p_x$$

• now L_x is not conserved
since it rotation would mix y and z

• L_y is ~~not~~ conserved due to its
rotational symmetry about its
axis

• L_z is not conserved for the same
reason L_x is, but it rotation would
mix x and y

So,
 L_x : not conserved.

L_y : conserved.

L_z : not conserved.

Q4)

a) $F = m \frac{dv}{dt}$

$$F = F\hat{x} \quad \text{so, } F\hat{x} = m \frac{dv}{dt}$$

$$\Rightarrow \frac{F\hat{x}}{m} dt = dv$$

$$\int \frac{F\hat{x}}{m} dt = \int dv \quad \text{since } v(0)=0$$

we get $\int_0^t \frac{F\hat{x}}{m} dt = v(t) = \frac{F}{m} t \hat{x}$

$$P(t) = m v(t) \Rightarrow m \frac{F}{m} t \hat{x}$$

$$P(t) = F t \hat{x}$$

$$D_{x0} + \int_0^t v(t) dt = x(t) = \frac{F t^2}{2m} \hat{x} + x_0$$

Substituting x_0

Final $r(t) = (x_0 + \frac{F t^2}{2m}) \hat{x} + y_0 \hat{y}$

c) angular momentum is defined as

$$\mathbf{L}(t) = \mathbf{r}(t) \times \mathbf{p}(t)$$

we have both taking their cross product

so the result is:

note*

I used chatopt to do
the cross product.

$$\mathbf{L}(t) = b_0 \mathbf{F} t \hat{\mathbf{z}}$$

torque is

$$\frac{d\mathbf{L}}{dt} = \mathbf{T}(t)$$

So,

$$\mathbf{T}(t) = b_0 \mathbf{F} \hat{\mathbf{z}}$$

angular momentum is not conserved since

$$\mathbf{T}(t) \neq 0$$

5. Numerical Elements -- Bouncing Object (40pts)

This exercise builds on the code you wrote for Exercise 4 in Homework 3. Here, we will make the motion of a falling object in two dimensions more realistic by allowing the object to bounce up again due to a normal force from the floor. Please take a look at the posted solution for Homework 3 as you will only need to change the forces at play.

In Homework 3, we introduced gravity and air resistance and studied their effects via a constant acceleration due to gravity and the force arising from air resistance. But what happens when the ball hits the floor? What if we would like to simulate the normal force from the floor acting on the ball? This exercise shows how we can include more complicated forces with no pain! And the force we include here is an example of a case where analytical solutions may either be difficult to find or we cannot find an analytical solution at all.

We need then to include a force model for the normal force from the floor on the ball. The simplest approach to such a system is to introduce a contact force model represented by a spring. So, we model the interaction between the floor and the ball as a single spring and the normal force is zero when there is no contact.

This is a strong simplification of the actual process occurring at the contact between the ball and the floor due to the deformation of both the ball and the floor. The deformed region corresponds roughly to the region of **overlap** between the ball and the floor. The depth of this region is $\Delta y = R - y(t)$, where R is the radius of the ball. This is supposed to represent the compression of the spring. Our model for the normal force acting on the ball is then

$$N = -k(R - y(t))e_y.$$

The normal force must act upward when $y < R$, hence the sign must be negative. However, we must also ensure that the normal force only acts when the ball is in contact with the floor, otherwise the normal force is zero. The full formation of the normal force is therefore

$$N = k(R - y(t))e_y,$$

when $y(t) < R$ and zero when $y(t) \leq R$. In the numerical calculations you can choose $R = 0.1$ m and the spring constant $k = 1000$ N/m.

a. (20pts) Update the code from Exercise 4 in Homework 3 by completing the following steps:

1. Initialize any new constants for this physical system.
2. Set up a vector for the Gravitational Force (in 2D).
3. Set up arrays for time, position (in 2D), and velocity (in 2D).
4. Define the initial conditions for the position and velocity (in 2D) as $x_0 = (10, 0)$ and $v_0 = (5, 0)$
5. Use Euler's method to calculate new positions, velocities, and times. Make sure to identify all of the forces acting on the ball to find the acceleration of the falling ball now with the normal force as well. See AMS Ch. 7.5.1, in particular Figure 7.10 and Equation 7.10.
6. Print the position velocities, and time. Comment on the structure of those arrays.

```
In [6]: %matplotlib inline
```

```
# let's start by importing useful packages we are familiar with
import numpy as np
import pandas as pd
from math import *
import matplotlib.pyplot as plt
from math import sqrt, ceil, log, cosh, tanh

%matplotlib inline
```

```
In [18]: import numpy as np
from math import sqrt, log, cosh, tanh, ceil
```

```
# Constants
g = 9.80655 # m/s^2, gravitational acceleration
D = 0.00245 # drag constant (mass/length) kg/m
m = 0.2 # kg, mass of the falling object
DeltaT = 0.001 # time step
tfinal = 10.0 # extend final time for multiple bounces
n = ceil(tfinal / DeltaT) # number of points
k = 1000 # N/m, spring constant for the normal force
R = 0.1 # radius of the ball in meters

# Define scaling constant vT used in analytical solution
vT = sqrt(m * g / D)

t = np.zeros(n)
vx = np.zeros(n)
vy = np.zeros(n)
x = np.zeros(n)
y = np.zeros(n)
yanalytic = np.zeros(n)
vyanalytic = np.zeros(n)
```

```

x[0] = 10.0
y[0] = R
vx[0] = 5.0
vy[0] = 0.0

for i in range(n - 1):
    speed = sqrt(vx[i]**2 + vy[i]**2)

    ax = -D * vx[i] * speed / m
    ay = -g - D * vy[i] * speed / m

    if y[i] < R:
        ay += k * (R - y[i]) / m
        vy[i] = -vy[i]

    vx[i + 1] = vx[i] + DeltaT * ax
    vy[i + 1] = vy[i] + DeltaT * ay

    x[i + 1] = x[i] + DeltaT * vx[i]
    y[i + 1] = y[i] + DeltaT * vy[i]

    t[i + 1] = t[i] + DeltaT

yanalytic[i + 1] = y[0] - (vT**2 / g) * log(cosh(g * t[i + 1] / vT)) + vy[0] * t[i + 1]
yanalytic[i + 1] = vy[0] - vT * tanh(g * t[i + 1] / vT)

if y[i + 1] < 0.0:
    break

max_y = np.max(y)
min_y = np.min(y)
max_vx = np.max(vx)
min_vx = np.min(vx)
max_vy = np.max(vy)
min_vy = np.min(vy)
max_time = t[np.argmax(y)]
min_time = t[np.argmin(y)]

# with the help of chatgpt I tried to make the result as informative as possible
print(f"Maximum height reached by the ball: {max_y:.3f} m at time {max_time:.3f} s")
print(f"Minimum height reached by the ball: {min_y:.3f} m at time {min_time:.3f} s")
print(f"Maximum horizontal velocity (vx): {max_vx:.3f} m/s")
print(f"Minimum horizontal velocity (vx): {min_vx:.3f} m/s")
print(f"Maximum vertical velocity (vy): {max_vy:.3f} m/s")
print(f"Minimum vertical velocity (vy): {min_vy:.3f} m/s")

print(f"Final x-position: {x[i+1]:.3f} m")
print(f"Final y-position: {y[i+1]:.3f} m")

```

Maximum height reached by the ball: 0.149 m at time 9.942 s
 Minimum height reached by the ball: 0.099 m at time 8.669 s
 Maximum horizontal velocity (vx): 5.000 m/s
 Minimum horizontal velocity (vx): 3.093 m/s
 Maximum vertical velocity (vy): 0.982 m/s
 Minimum vertical velocity (vy): -0.972 m/s
 Final x-position: 48.973 m
 Final y-position: 0.134 m

b. (20pts) Change the final time to a large enough value so you can study the ball bouncing up and down several times. Plot the trajectory of the bounce (y vs x) and the evolution of each of the velocity components (\mathbf{v} vs t). Comment your results.

```

In [21]: # Constants
g = 9.80655 # m/s^2, gravitational acceleration
D = 0.00245 # drag constant (mass/length) kg/m
m = 0.2 # kg, mass of the falling object
DeltaT = 0.001
tfinal = 10.0
n = ceil(tfinal / DeltaT)
k = 1000
R = 0.1

t = np.zeros(n)
vx = np.zeros(n)
vy = np.zeros(n)
x = np.zeros(n)
y = np.zeros(n)

x[0] = 10.0

```

```

y[0] = R
vx[0] = 5.0
vy[0] = 0.0

for i in range(n - 1):
    speed = sqrt(vx[i]**2 + vy[i]**2)

    ax = -D * vx[i] * speed / m
    ay = -g - D * vy[i] * speed / m

    if y[i] < R:
        ay += k * (R - y[i]) / m
        vy[i] = -vy[i]

    vx[i + 1] = vx[i] + DeltaT * ax
    vy[i + 1] = vy[i] + DeltaT * ay

    x[i + 1] = x[i] + DeltaT * vx[i]
    y[i + 1] = y[i] + DeltaT * vy[i]

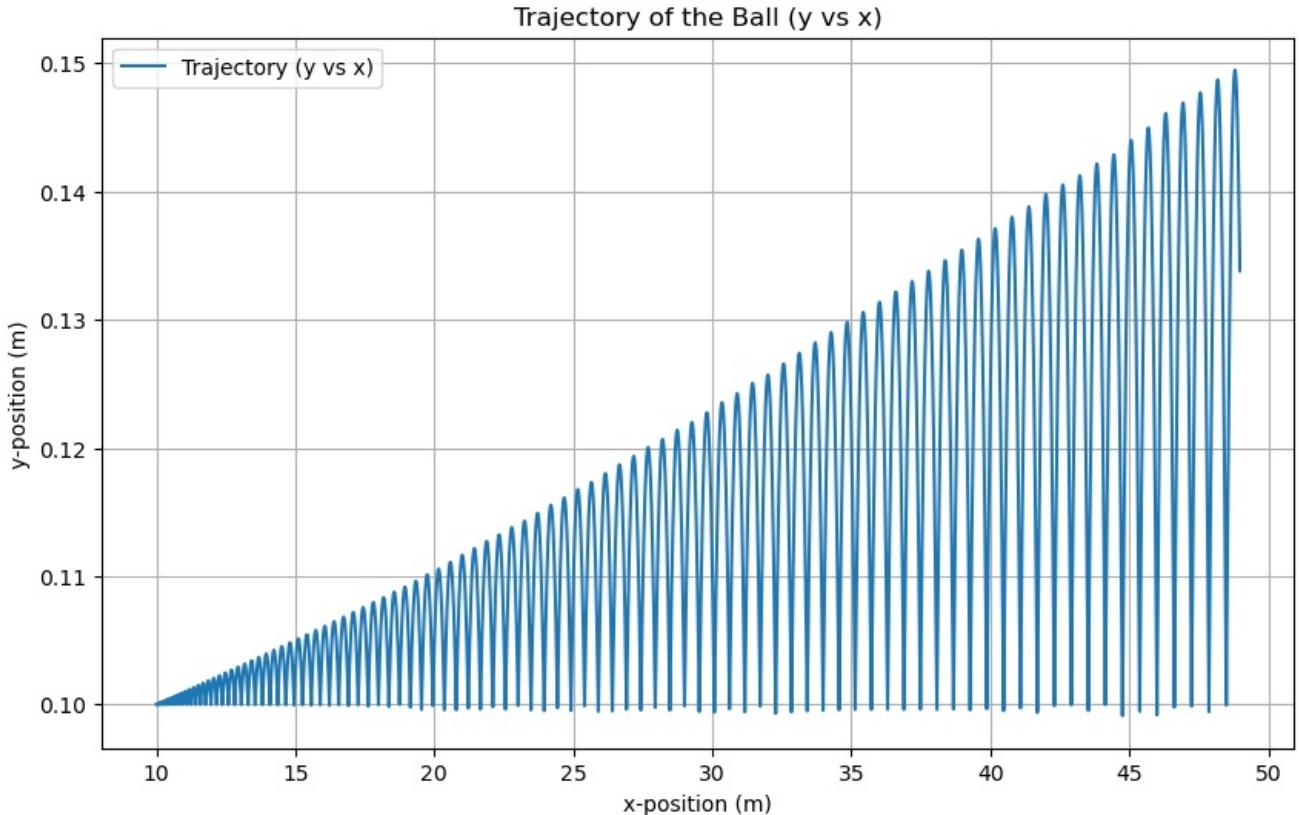
    t[i + 1] = t[i] + DeltaT

plt.figure(figsize=(10, 6))
plt.plot(x, y, label="Trajectory (y vs x)")
plt.title('Trajectory of the Ball (y vs x)')
plt.xlabel('x-position (m)')
plt.ylabel('y-position (m)')
plt.grid(True)
plt.legend()

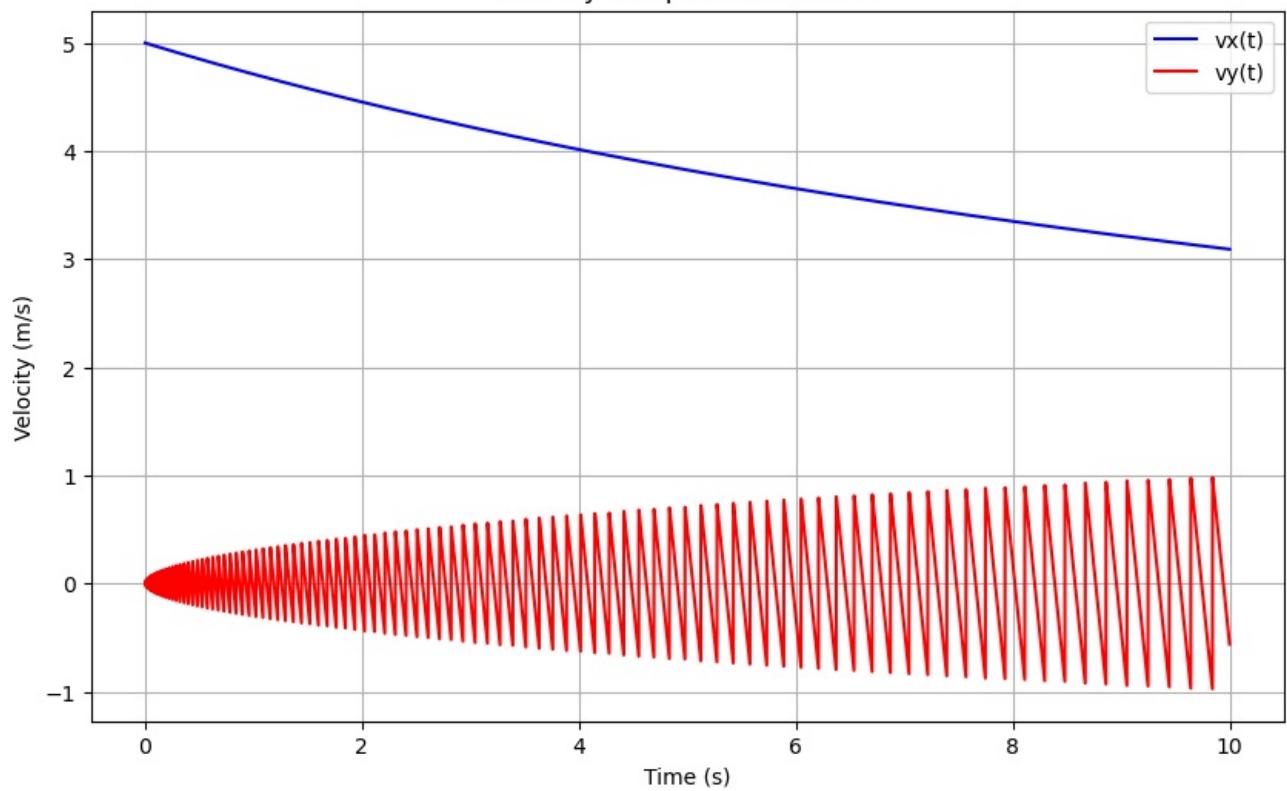
plt.figure(figsize=(10, 6))
plt.plot(t, vx, label="vx(t)", color='blue')
plt.plot(t, vy, label="vy(t)", color='red')
plt.title('Velocity Components vs Time')
plt.xlabel('Time (s)')
plt.ylabel('Velocity (m/s)')
plt.grid(True)
plt.legend()

# Show the plots
plt.show()

```



Velocity Components vs Time



In []:

Processing math: 100%