task1

October 6, 2022

1 Task 1

Anastasiia Stepanova

[1]:
```
import numpy as np

k_param = 1.3 * 10 ** (-5)
L = 1500
mass = 0.0136  #kg
v_0 = 870
gravity = 9.8

x_0 = 0
# max height position of a sniper
y_0 = 0

# the max height of the soldier
height = 2
```

2 1. Without air resistance

$$\begin{cases} L = x_0 + V_0 * cos(\alpha) * t \\ y = y_0 + V_0 * sin(\alpha) * t - \frac{gt^2}{2}, \text{ where } y \in [0, 2]m \\ V_0 = 870m/s \\ x_0 = 0m \\ y_0 = 0m \end{cases}$$

$$2V_0 sin(\alpha) = gt$$
$$\alpha = \frac{1}{2} arcsin(\frac{Lg}{V_0^2})$$

[2]:
```
alpha_1 = np.arcsin(L * gravity / (v_0 ** 2)) / 2
print(f'Alpha is equal to: \\alpha = {alpha_1}')
```

```
Alpha is equal to: \alpha = 0.009711272471294132
```

3 2. Find the max height of the cargo ship

The maximum value for height of the cargo ship will be reached when the bullet

$$\vec{V}(t_1) = \vec{V}(t_1)_x$$

$$t_1 = \frac{V_0 sin(\alpha)}{g}$$

$$y(t_1) = \frac{(V_0 sin(\alpha))^2}{2g}$$

[3]:
```
max_height = (v_0 * np.sin(alpha_1)) ** 2 / (2 * gravity)

print(f'Max height of the cargo ship, non-considering its wide: {max_height} m')
```

```
Max height of the cargo ship, non-considering its wide: 3.641841663376592 m
```

4 3. With air resistance

4.0.1 Conditions:

$$\begin{cases} x_0 = 0 & x_f = L \\ \dot{x}_0 = V_0 * cos(\alpha) & \dot{x}_f - ? \\ y_0 = 0 & y_f \in [0, 2]m \\ \dot{y}_0 = V_0 * sin(\alpha) & \dot{y}_f - ? \\ t_0 = 0 & t_f - ? \end{cases}$$

4.0.2 Kinematics analysis:

$$\begin{cases} x = x_0 + V_x * t + \frac{a_x * t^2}{2} \\ x = x_0 + V_x * t + \frac{a_y * t^2}{2} \end{cases}$$

$$\begin{cases} x = x_0 + \dot{x} * t + \frac{\ddot{x} * t^2}{2} \\ x = x_0 + \dot{y} * t + \frac{\ddot{y} * t^2}{2} \end{cases}$$

4.0.3 Force analysis:

$$m\ddot{r} = m\vec{g} + \vec{F}_c$$

x: $m\ddot{x} = -F_{cx} = -k * \dot{x}\sqrt{\dot{x}^2 + \dot{y}^2}$ y: $m\ddot{y} = -F_{cy} - mg = -k * \dot{y}\sqrt{\dot{x}^2 + \dot{y}^2} - mg$

4.0.4 Solution:

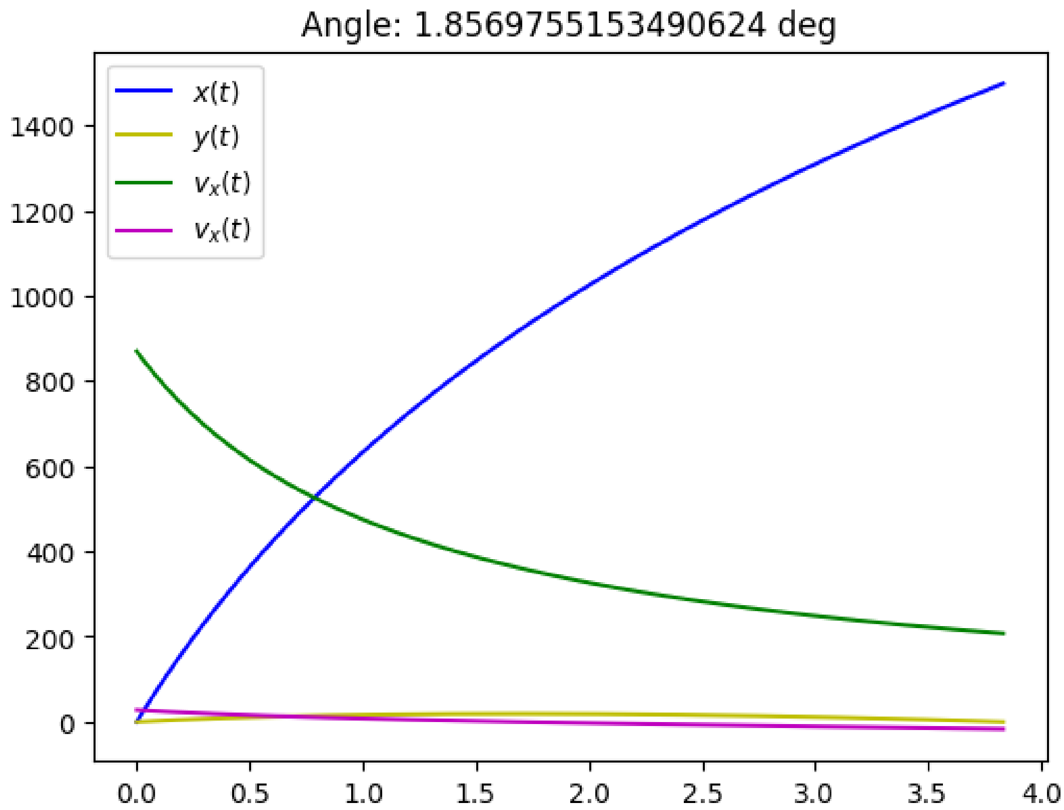
$$\begin{cases} \ddot{x} = -k\dot{x}\sqrt{\dot{x}^2 + \dot{y}^2} \\ \ddot{y} = -\frac{k}{m}\dot{y}\sqrt{\dot{x}^2 + \dot{y}^2} - g \\ \dot{x}(0) = V_0 cos(\alpha) \\ \dot{y}(0) = V_0 sin(\alpha) \\ y(t_f) \in [0, 2]m \\ V_0(0) = \sqrt{\dot{x}(0)^2 + \dot{y}(0)^2} = 870m/s \\ x_0 = 0m \\ y_0 = 0m \end{cases}$$

[4]:
```
import scipy as sc

dalpha = 0.001
# dalpha_rad = dalpha * np.pi / 180
alpha_max = np.pi / 2
n_alpha = int(alpha_max // dalpha)
alpha = np.linspace(0, alpha_max, n_alpha)

N = 1000
start_time = 0
end_time = 5
t = np.linspace(start_time, end_time, N)
```

[5]:
```
def diff(s, t, k, g, m):
    x, y, v_x, v_y = s
    dsdt = [
        v_x,
        v_y,
        -k * np.sqrt(v_x ** 2 + v_y ** 2) * v_x / m,
        -g - k * np.sqrt(v_x ** 2 + v_y ** 2) * v_y / m
    ]
    return dsdt
```

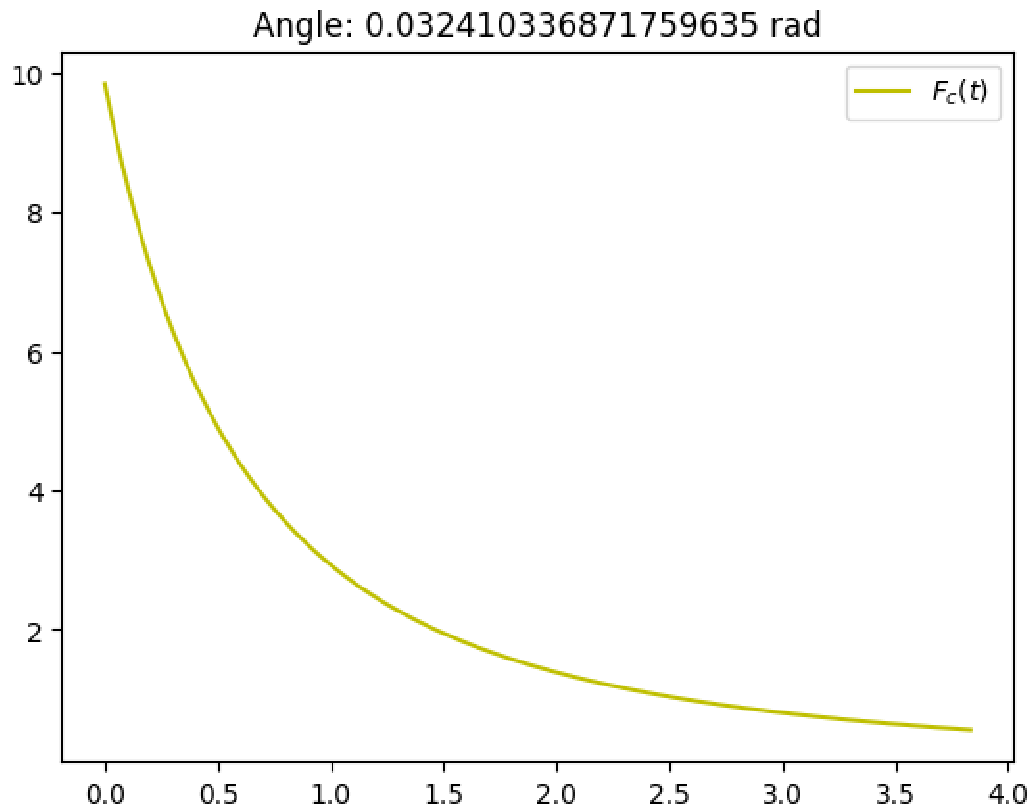
[30]:
```
from scipy.integrate import odeint
import matplotlib.pyplot as plt
from multiprocessing import Pool


def check_angle(a):
    # s = (x, y, dx, dy)
    s0 = [0, 0, v_0 * np.cos(a), v_0 * np.sin(a)]
    sol = odeint(diff, s0, t, args=(k_param, gravity, mass))

    x = sol[:, 0]
    y = sol[:, 1]
```

```python
    dx = sol[:, 2]
    dy = sol[:, 3]

    n_x_bigger_L = np.where(x >= L)
    n = n_x_bigger_L[0][0] - 1

    if height >= y[n] >= 0:
        time = t[:n]
        x_reach = x[:n]
        y_reach = y[:n]
        v_x_reach = dx[:n]
        v_y_reach = dy[:n]
        plt.plot(time, x_reach, 'b', label='$x(t)$')
        plt.plot(time, y_reach, 'y', label='$y(t)$')
        plt.plot(time, v_x_reach, 'g', label='$v_x(t)$')
        plt.plot(time, v_y_reach, 'm', label='$v_x(t)$')
        plt.title(f'Angle: {a * 180 / np.pi} deg')
        plt.legend()
        plt.show()
        print(f'The min height of the soldier should be: h = {y_reach[-1]} m')
        return sol[:n], y[n]

    else:
        return None, y[n]


y_t = np.empty(shape=(n_alpha, 1))

# with Pool(6) as p:
#     sol = (p.map(check_angle, alpha))
sol = []
alpha_reach = 0
for a in alpha:
    sol, y_t = check_angle(a)
    if sol is not None:
        alpha_reach = a
        break
```




```
The min height of the soldier should be: h = 0.12347497930291995 m
```

[31]:
```
v_x = sol[:, 2]
v_y = sol[:, 3]
F_c = k_param * (v_x ** 2 + v_y ** 2)
n = v_x.size
plt.plot(t[:n], F_c, 'y', label='$F_c(t)$')
plt.title(f'Angle: {alpha_reach} rad')
plt.legend()
plt.show()
```


Angle: 0.032410336871759635 rad
$F_c(t)$
10
8
6
4
2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

```
[ ]:
```