Modelação de Sistemas Físicos - Aula Prática nº7

Realização e resolução de problemas sobre:

- Forças conservativas

- Conservação de energia

Exercício 1

Uma bola pequena é rolada por uma pista cuja forma é dada pela função

$$y(x) = \begin{cases} (0.1 - 0.05x)\,\mathrm{m}, & 0 \le x < 2\,\mathrm{m} \\ 0\,\mathrm{m}, & x \ge 2\,\mathrm{m} \end{cases}$$

A bola acelera devido à força da gravidade, com aceleração que depende do declive da pista. A aceleração horizontal é dada por:

$$a_x = -g\frac{dy}{dx}\frac{dx}{|dr|} \approx -gf'(x) = \begin{cases} 0.5g\,\mathrm{m}, & 0 \le x < 2\,\mathrm{m} \\ 0\,\mathrm{m}, & x \ge 2\,\mathrm{m} \end{cases}$$

1. Faça uma simulação do movimento da bola usando o método de Euler-Cromer com as seguintes condições inicias:

$$\begin{aligned} x(t=0) &= 0\,\mathrm{m} \\ v_x(t=0) &= 0\,\mathrm{m/s} \end{aligned}$$

Solução:

A solução numérica para o problema (Método de Euler-Cromer) inicia-se com o cálculo da aceleração, que é aproximada a $a_x \approx -g\,f'(x)$ ao longo da trajetória. Para isso necessitamos da derivada $f' = dy/dx$,

$$f'(x) = \begin{cases} -0.05, & 0 \le x < 2\,\mathrm{m} \\ 0, & x \ge 2\,\mathrm{m} \end{cases}$$

Portanto

$$a_x = \begin{cases} 0.05\,g, & 0 \le x < 2\,\mathrm{m} \\ 0, & x \ge 2\,\mathrm{m} \end{cases}$$

A solução pelo método de Euler-Cromer segue então a receita usual:

$$\frac{\mathrm{d}v_x}{\mathrm{d}t} = a_x$$

$$\frac{\mathrm{d}x}{\mathrm{d}t} = v_x$$

O que na forma discreta se traduz em:

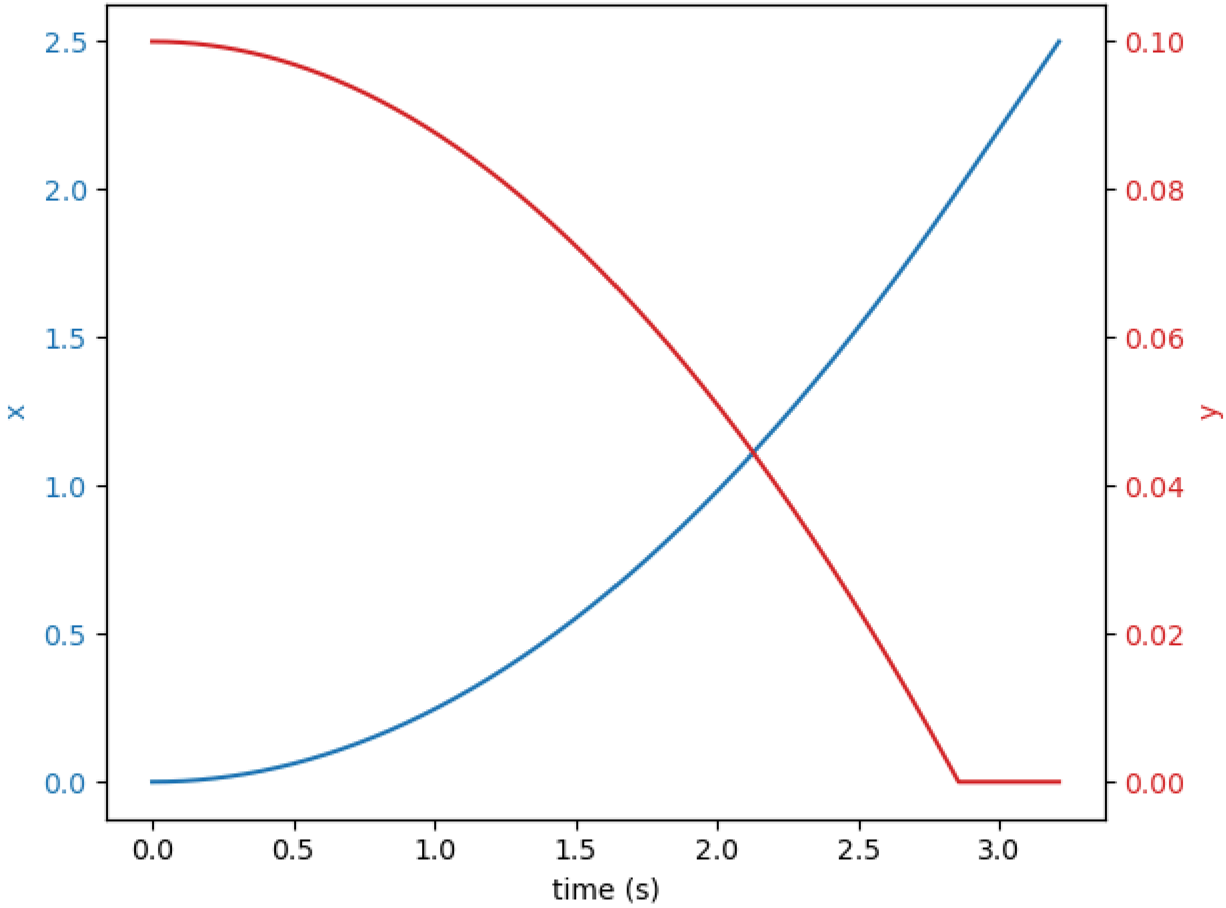
$$v_{x,i+1} = v_{x,i} + a_{x,i}\,\delta\tau$$

$$x_{i+1} = x_i + v_{x,i+1}\,\delta\tau$$

Note que utilizamos de $v_{x,i+1}$ no cáculo da posição (no método de Euler usa $v_{x,i}$).

In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt

t0 = 0.0                    # condição inicial, tempo [s]
tf = 4.0                    # limite do domínio, tempo final [s]
dt = 0.001                  # passo [s]
vx0 = 0.0                   # condição inicial, módulo da velocidade inicial [m/s]
x0 = 0.0                    # condição inicial, coordenada x da posição inicial [m]
y0 = 0.1                    # condição inicial, coordenada y da posição inicial [m]
g = 9.80665                 # aceleração gravitacional (valor standard) [m/s^2]

# Trajetória y(x)
# y(x) = 0.1 - 0.05*x para x < 2, de outra forma y(x) = 0
def y_func(x: float) -> float:
    return 0.1 - 0.05 * x if x < 2.0 else 0.0

# Derivada de y em ordem a x
# dy/dx = -0.05 para x < 2, de outra forma dy/dx = 0
def dydx_func(x: float) -> float:
    return -0.05 if x < 2.0 else 0.0

# inicializar domínio [ano]
t = np.arange(t0, tf, dt)

# inicializar solução, aceleração a 1D [m/s^2]
ax = np.zeros(np.size(t))

# inicializar solução, velocidade [m/s]
vx = np.zeros(np.size(t))
vx[0] = vx0

# inicializar solução, posição [m]
x = np.zeros(np.size(t))
y = np.zeros(np.size(t))
x[0] = x0
y[0] = y0

for i in range(np.size(t) - 1):

    # aceleração
    ax[i] = -g * dydx_func(x[i])

    # Método de Euler-Cromer
    vx[i + 1] = vx[i] + ax[i] * dt
    x[i + 1] = x[i] + vx[i + 1] * dt
    y[i + 1] = y_func(x[i+1])


fig, ax1 = plt.subplots()
color = 'tab:blue'
ax1.set_xlabel('time (s)')
ax1.set_ylabel('x', color=color)
ax1.plot(t[x<2.5], x[x<2.5], color=color)
ax1.tick_params(axis='y', labelcolor=color)
```

```python
ax2 = ax1.twinx()  # partilhar eixo horizontal

color = 'tab:red'
ax2.set_ylabel('y', color=color)
ax2.plot(t[x<2.5], y[x<2.5], color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()
plt.show()
```



2. Encontre a velocidade final e o tempo em que a bola atinge $x = 2.5$ m.

In [2]:
```python
x1 = x
y1 = y
vx1 = vx
i25 = np.size(x[x<=2.5])
v25 = vx[i25]
t25 = t[i25]
print("Quando x = 2.5 m, a velocidade é v = {0:.5f} m/s²".format(v25))
print("Quando x = 2.5 m, o tempo decorrido é t = {0:.5f} s".format(t25))
```

```
Quando x = 2.5 m, a velocidade é v = 1.40039 m/s²
Quando x = 2.5 m, o tempo decorrido é t = 3.21300 s
```

3. Podemos comparar os resultados com os obtidos através da conservação da energia. Calcule a energia potencial inicial, E_p, e a energia cinética final, E_c. Qual é a velocidade final? Concorda com o resultado obtido na simulação?

$$\begin{aligned} E_{\mathrm{p}} &= M\,g\,y \\ E_{\mathrm{c}} &= \tfrac{1}{2}M\,v^2 \end{aligned}$$

Igualando,

$$M\,g\,y = \frac{1}{2} M\,v^2$$

simplificando,

$$v = \sqrt{2\,g\,y}$$

Assim, considerando $g = 9.80665\ \mathrm{m/s^2}$ e $y = 0.1$ m, chegamos à velocidade final $v = 1.40048\ \mathrm{m/s}$.

A velocidade obtida a partir da conservação da energia difere muito pouco do valor numérico. Isto deve-se à aproximação assumida para a aceleração $a_x \approx -gf'(x)$, que é válida para inclinações pequenas, o que é o caso.

Exercício 2

Repita o exercício anterior para a seguintes forma para a pista:

$$y(x) = \begin{cases} 0.025(x-2)^2\ \mathrm{m}, & 0 \leq x < 2\ \mathrm{m} \\ 0\ \mathrm{m}, & x \geq 2\ \mathrm{m} \end{cases}$$

1. Qual é a derivada desta função $f(x)$? Então qual é a formula aproximada para a aceleração horizontal?

A derivada da função $f(x)$ é:

$$f'(x) = \frac{\mathrm{d}y}{\mathrm{d}x} = \begin{cases} 0.05(x-2)\ \mathrm{m}, & 0 \leq x < 2\ \mathrm{m} \\ 0\ \mathrm{m}, & x \geq 2\ \mathrm{m} \end{cases}$$

A fórmula aproximada para a aceleração horizontal é:

$$a_x \approx -g\,f'(x) = \begin{cases} -0.05\,g\,(x-2)\ \mathrm{m}, & 0 \leq x < 2\ \mathrm{m} \\ 0\ \mathrm{m}, & x \geq 2\ \mathrm{m} \end{cases}$$

2. Faça uma simulação do movimento da bola usando o método de Euler-Cromer com as seguintes condições inicias:

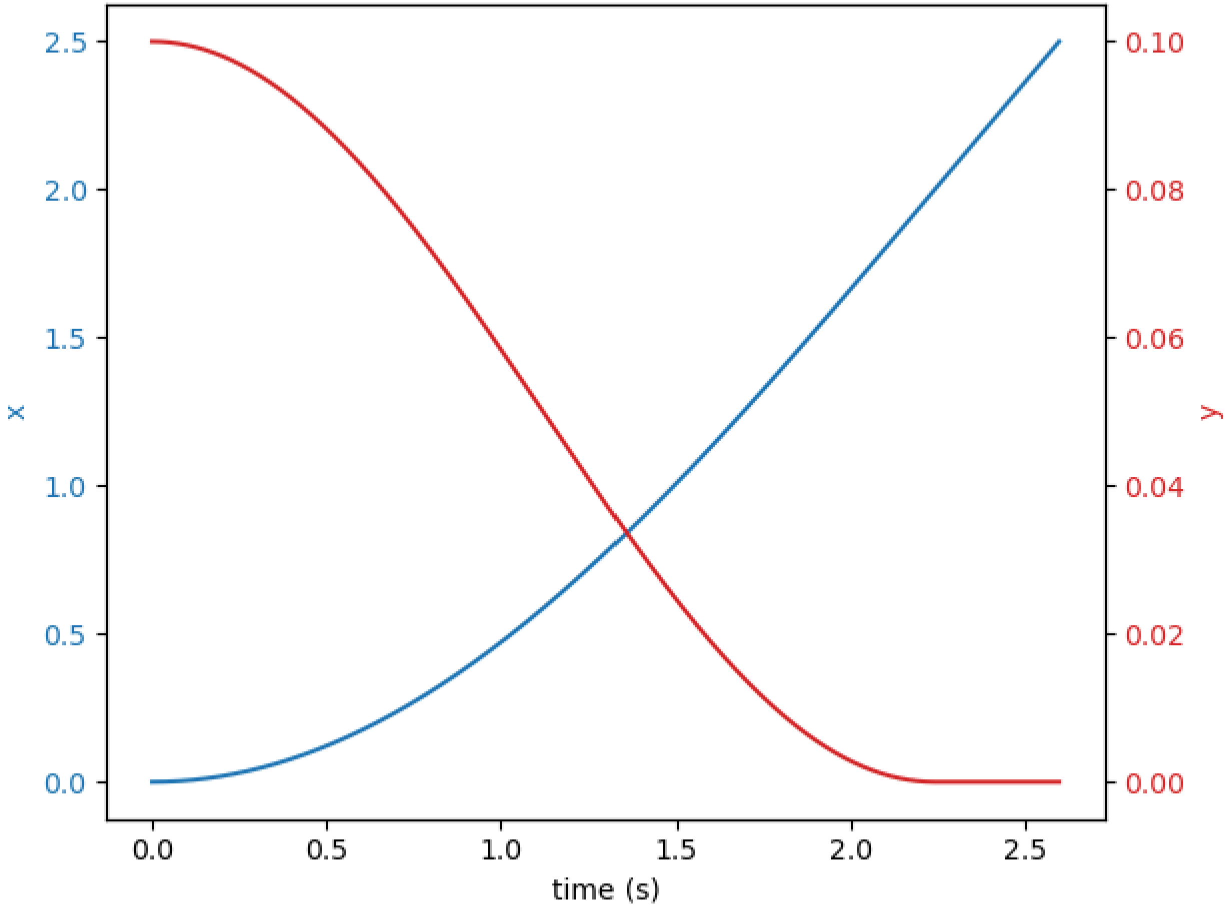
$$x(t=0) = 0\ \mathrm{m}$$

$$v_x(t=0) = 0\ \mathrm{m/s}$$

simule o movimento até a bola atingir a distância $x = 2.5\ \mathrm{m}$.

In [3]:
```python
import numpy as np
import matplotlib.pyplot as plt

t0 = 0.0                    # condição inicial, tempo [s]
tf = 4.0                    # limite do domínio, tempo final [s]
dt = 0.001                  # passo [s]
vx0 = 0.0                   # condição inicial, módulo da velocidade inicial [m/s]
x0 = 0.0                    # condição inicial, coordenada x da posição inicial [m]
```

```python
y0 = 0.1                        # condição inicial, coordenada y da posição inicial [m]
g = 9.80665                     # aceleração gravitacional (valor standard) [m/s^2]

# Trajetória y(x)
# y(x) = 0.025 * (x - 2)**2 para x < 2, de outra forma y(x) = 0
def y_func(x: float) -> float:
    return 0.025 * (x - 2)**2 if x < 2.0 else 0.0

# Derivada de y em ordem a x
# dy/dx = 0.05 * (x - 2) para x < 2, de outra forma dy/dx = 0
def dydx_func(x: float) -> float:
    return 0.05 * (x - 2) if x < 2.0 else 0.0

# inicializar domínio [ano]
t = np.arange(t0, tf, dt)

# inicializar solução, aceleração a 1D [m/s^2]
ax = np.zeros(np.size(t))

# inicializar solução, velocidade [m/s]
vx = np.zeros(np.size(t))
vx[0] = vx0

# inicializar solução, posição [m]
x = np.zeros(np.size(t))
y = np.zeros(np.size(t))
x[0] = x0
y[0] = y0

for i in range(np.size(t) - 1):

    # aceleração
    ax[i] = -g * dydx_func(x[i])

    # Método de Euler-Cromer
    vx[i + 1] = vx[i] + ax[i] * dt
    x[i + 1] = x[i] + vx[i + 1] * dt
    y[i + 1] = y_func(x[i+1])


fig, ax1 = plt.subplots()
color = 'tab:blue'
ax1.set_xlabel('time (s)')
ax1.set_ylabel('x', color=color)
ax1.plot(t[x<2.5], x[x<2.5], color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()  # partilhar eixo horizontal

color = 'tab:red'
ax2.set_ylabel('y', color=color)
ax2.plot(t[x<2.5], y[x<2.5], color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()
plt.show()
```



3. Quanto tempo é que a bola demora a atingir $x = 2.5$ m? Foi mais rápido ou

mais lento do que no exercício 1?

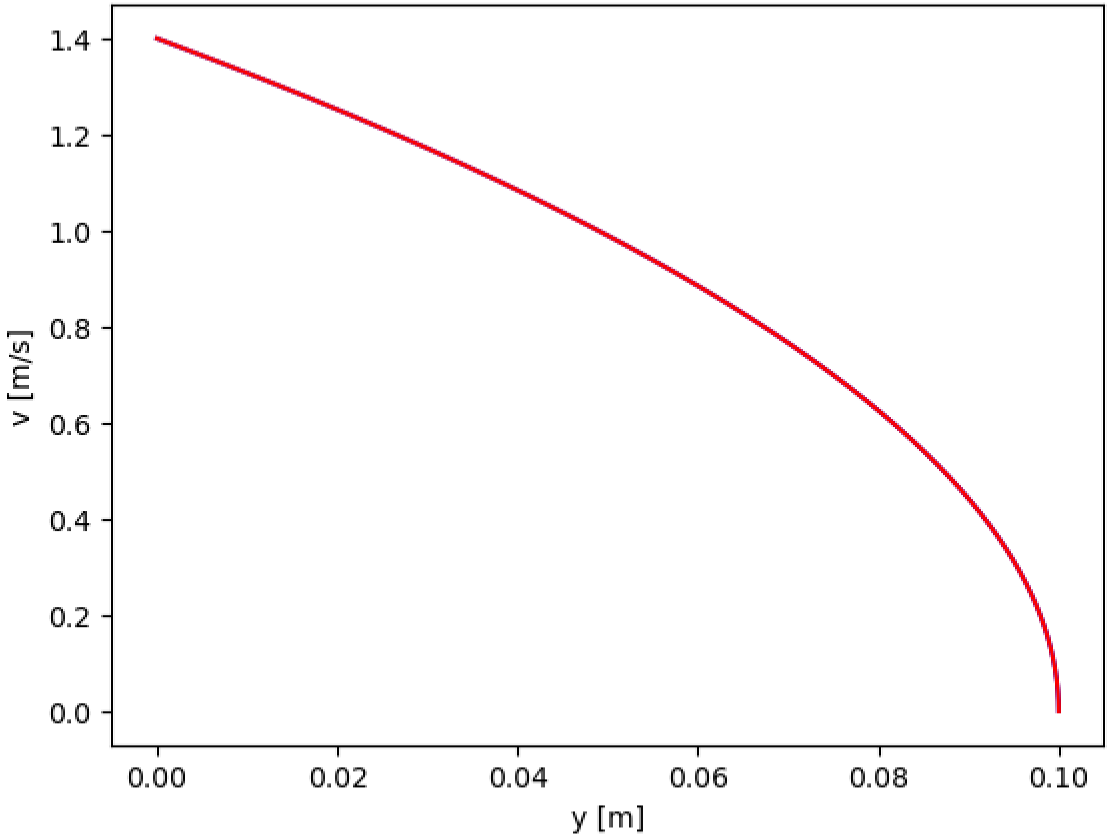
In [4]:
```
x2 = x
y2 = y
vx2 = vx
i25 = np.size(x[x<=2.5])
v25 = vx[i25]
t25 = t[i25]
print("Quando x = 2.5 m, a velocidade é v = {0:.5f} m/s²".format(v25))
print("Quando x = 2.5 m, o tempo decorrido é t = {0:.5f} s".format(t25))
```

```
Quando x = 2.5 m, a velocidade é v = 1.40047 m/s²
Quando x = 2.5 m, o tempo decorrido é t = 2.60000 s
```

A velocidade final é a mesma, mas a bola da trajetória parabélica chega ao destino ($x = 2.5$ m) mais rapidamente.

4. Faça o gráfico da velocidade v_x em função da altura y para cada forma da pista. Explique o resultado.

In [5]:
```
plt.plot(y1, vx1, 'b-', y2, vx2, 'r-')
plt.xlabel("y [m]")
plt.ylabel("v [m/s]")
plt.show()
```



A velocidade é idêntica quando a altura é idêncica. Isto é uma consequência da conservação da energia e da conversão, sem dissipação, da energia potencial em energia cinética.

Pergunta 2:

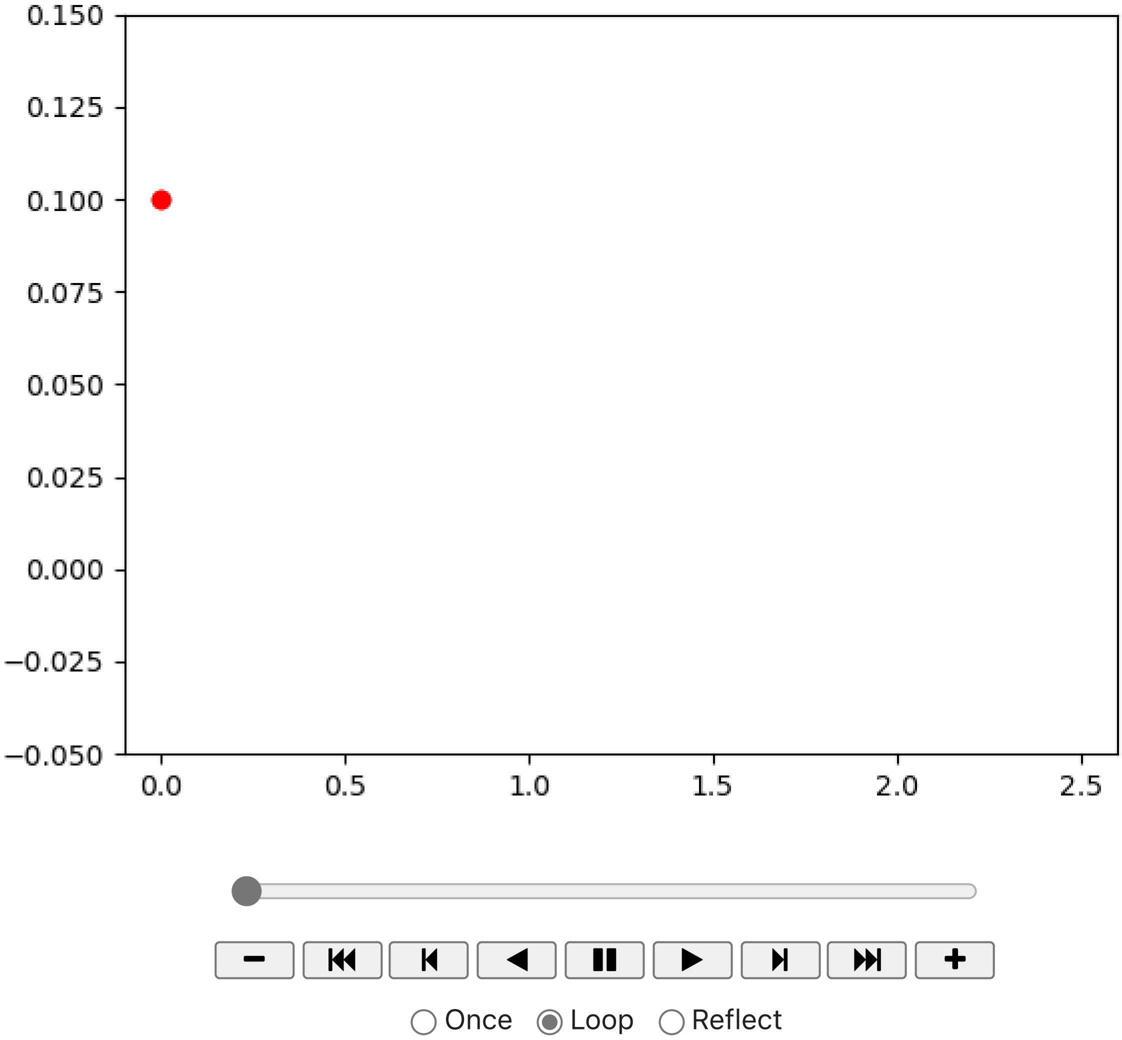
As velocidades finais são iguais ou diferentes para cada forma da pista? Explique.

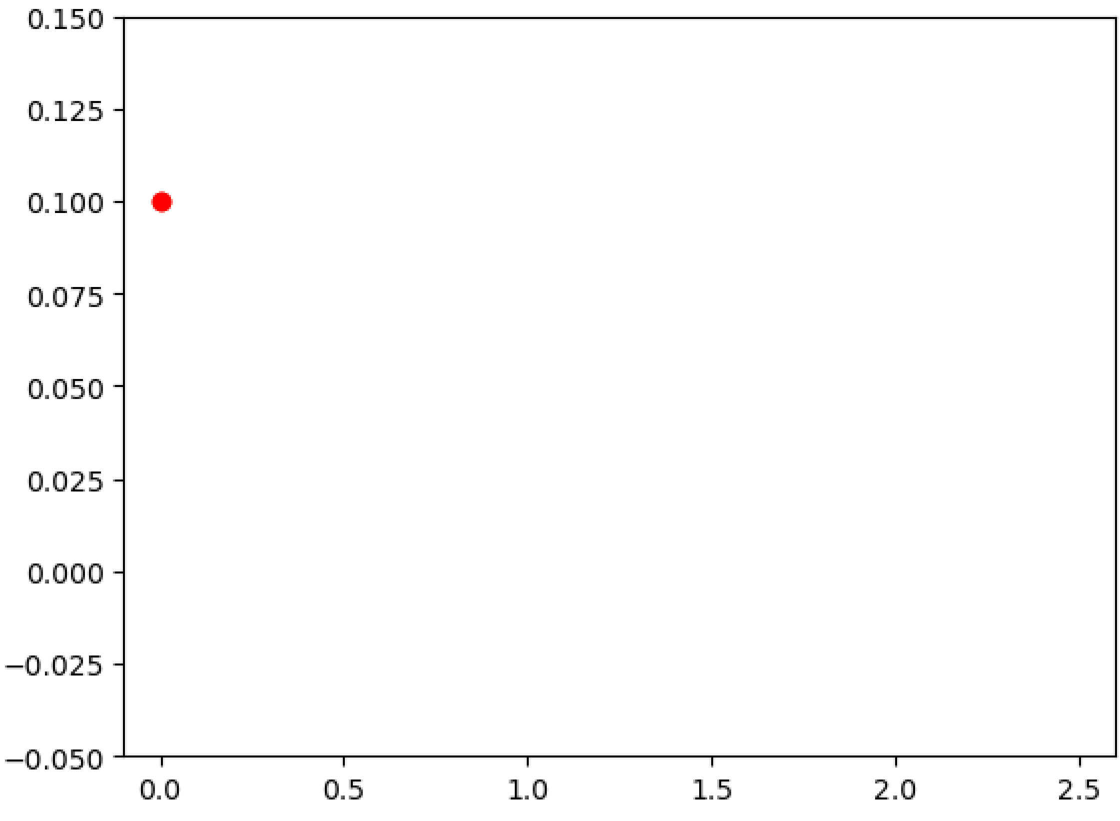
Exercício 3. Faça uma animação do movimento da bola para cada forma da pista. Para qual delas é que a bola atinge $x = 2\ \mathrm{m}$ primeiro?

In [7]:
```
from matplotlib.animation import FuncAnimation
from IPython.display import HTML

fig = plt.figure()
ax = plt.axes(xlim=(-0.1, 2.6), ylim=(-0.05, 0.15))
bola = ax.plot([], [], 'ro', [], [], 'bo')[0]        # bola, posição inicial

def update(frame):
    # atualizar o plot da posição da bola
    bola.set_xdata([x1[frame], x2[frame]])
    bola.set_ydata([y1[frame], y2[frame]])
    return bola

nframes = 100
total_frames = np.size(t)
iframes = np.arange(0, total_frames, total_frames // nframes)
ani = FuncAnimation(fig=fig, func=update, frames=iframes, interval=100)
```

```
HTML(ani.to_jshtml())
```

Out[7]:





A bola que segue a trajetória parabólica chega a primeiro a $x = 2.5$ m.

Consegue inventar uma forma da pista que seja mais rápida?

O campo gravítico é conservativo, logo a energia potencial perdida na descida é completamente convertida para energia cinética. Desta forma, só é possível conceber uma pista com uma velocidade final mais elevada se a diferença entre a altura inicial, $y(t=0)$, e a altura final, $y(t=3\,\mathrm{s})$ for superior a $10\,\mathrm{m}$.