Modelação de Sistemas Físicos - Aula Prática nº6

Realização e resolução de problemas sobre movimento a 2D e a 3D

Recapitulando o método de Euler para resolver equações diferenciais de primeira order:

Suponhamos que temos a seguinte equação diferencial

$$\frac{\mathrm{d}f(t)}{\mathrm{d}t} = g(t).$$

O método de Euler discretiza a equação no domínio t, permitindo obter o valor de $f(\tau_{i+1}) = f_{i+1}$ a partir da solução no intervalo de tempo imediatamente anterior,

$$f_{i+1} = f_i + g_i \delta\tau$$

Problema 1

Uma bola de futebol é chutada com velocidade de 100 km/h, a fazer um ângulo de 10° com o campo (horizontal).

a) Desenvolva um programa que obtenha a lei do movimento e a lei da velocidade em função do tempo, usando o método de Euler. Considere inicialmente só a força de gravidade.

Neste tipo de problemas seguimos a seguinte ordem: lei da aceleração > lei da velocidade > lei do movimento (posição)

A lei da aceleração é dada por,

$$\mathbf{a} = -g\,\hat{\mathbf{j}}$$

Lei da velocidade,

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = \mathbf{a}$$

A lei do movimento (posição) é dada por,

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{v}$$

Em que:

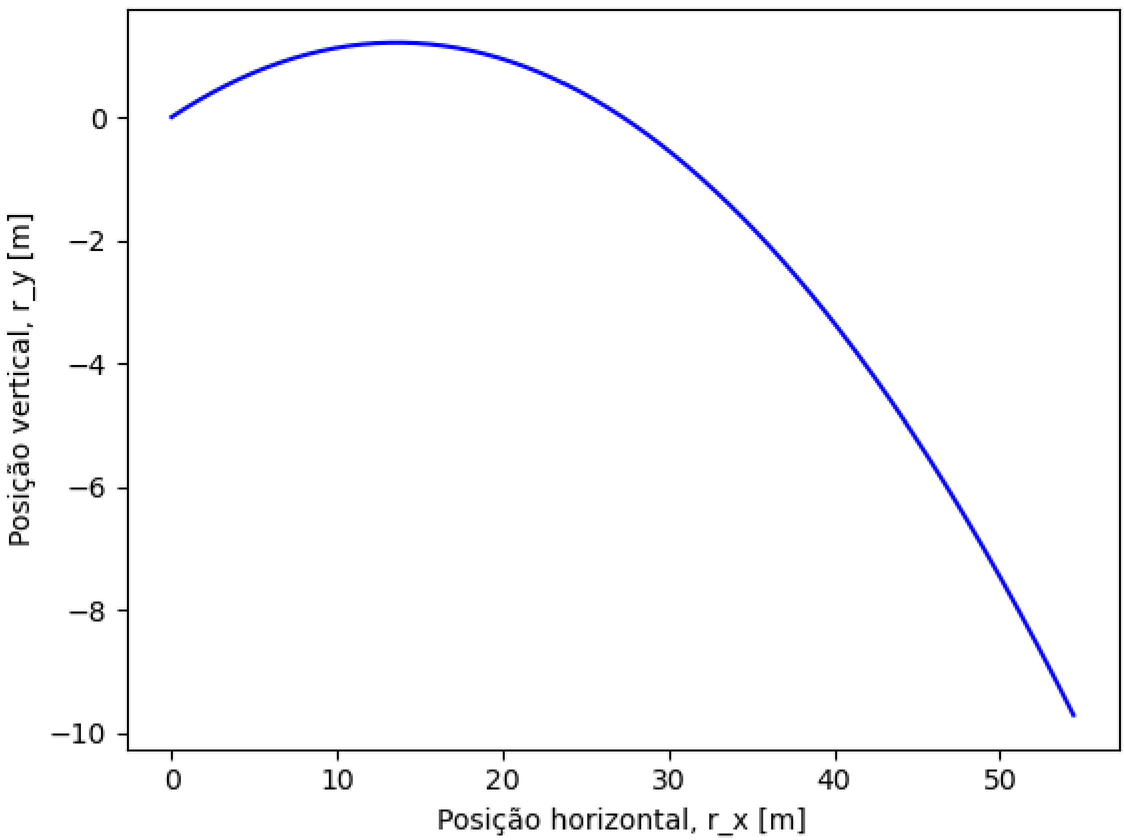
$g = \text{constante}$

$\mathbf{v}(t=0) = 100(\cos\theta,\, \sin\theta)\ \mathrm{km/h}$

$\mathbf{r}(t=0) = (0,\, 0)\ \mathrm{m}$

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

```python
t0 = 0.0                    # condição inicial, tempo [s]
tf = 2.0                    # limite do domínio, tempo final [s]
dt = 0.01                   # passo [s]
v0 = 100.0 * 1000 / 3600    # condição inicial, módulo da velocidade inicial [m/s]
theta0 = 10 * np.pi / 180.0 # condição inicial, ângulo do vetor velocidade inicial [r

g = 9.8                     # aceleração gravítica [m/s^2]

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

# inicializar solução, aceleração [m/s^2]
a = np.zeros([2, np.size(t)])
a[1,:] = -g                 # aceleração é um vetor constate (ao longo do -y)

# inicializar solução, velocidade [m/s]
v = np.zeros([2, np.size(t)])
v[:,0] = v0 * np.array([np.cos(theta0), np.sin(theta0)])  # velocidade para t = 0

# inicializar solução, posição [m]
r = np.zeros([2, np.size(t)])
r[:,0] = np.array([0.0, 0.0])    # posição para t = 0

for i in range(np.size(t) - 1):
    v[:, i + 1] = v[:, i] + a[:, i] * dt
    r[:, i + 1] = r[:, i] + v[:, i] * dt

plt.plot(r[0,:], r[1,:], 'b-')
plt.xlabel("Posição horizontal, r_x [m]")
plt.ylabel("Posição vertical, r_y [m]")
plt.show()
```



Podemos verificar se a solução está correta, comparando com a solução analítica:

In [2]:
```python
tm = v[1,0] / g                     # tempo para atingir altura máxima
ym = 0.0 + (v[1,0]) ** 2 / (2 * g)  # altura máxima
tsolo = 2 * v[1,0] / g              # tempo para atingir alcance máximo
xsolo = 2 * v[0,0] * v[1,0] / g     # alcance máximo
```

```
print("tm =", tm, "s")
print("ym =", ym, "m")
print("tsolo =", tsolo, "s")
print("xsolo =", xsolo, "m")
```

```
tm = 0.4922000500763332 s
ym = 1.1870783575462103 m
tsolo = 0.9844001001526664 s
xsolo = 26.929023630453887 m
```

b) Atualize o seu programa de modo a considerar a força de resistência do ar. A força de resistência do ar ao movimento da bola é:

$$\mathbf{F}^{(\mathrm{res})} = -m\, D\, |\mathbf{v}|\, \mathbf{v}$$

ou seja,

$$\begin{cases} F_x^{(\mathrm{res})} & = & -m\, D\, |\mathbf{v}|\, v_x \\ F_y^{(\mathrm{res})} & = & -m\, D\, |\mathbf{v}|\, v_y \end{cases}$$

em que $D = g/v_{\mathrm{T}}^2$, e em que a velocidade terminal $v_{\mathrm{T}} = 100\,\mathrm{km/h}$. Faça o gráfico da altura em função da distância percorrida na horizontal.

Assim, a aceleração não é constante (depende da velocidade, consequentemente do tempo), e é dada por

$$\mathbf{a} = -g\,\hat{\mathbf{j}} + \mathbf{F}^{(\mathrm{res})}/m = -g\,\hat{\mathbf{j}} - D\,|\mathbf{v}|\,\mathbf{v}$$

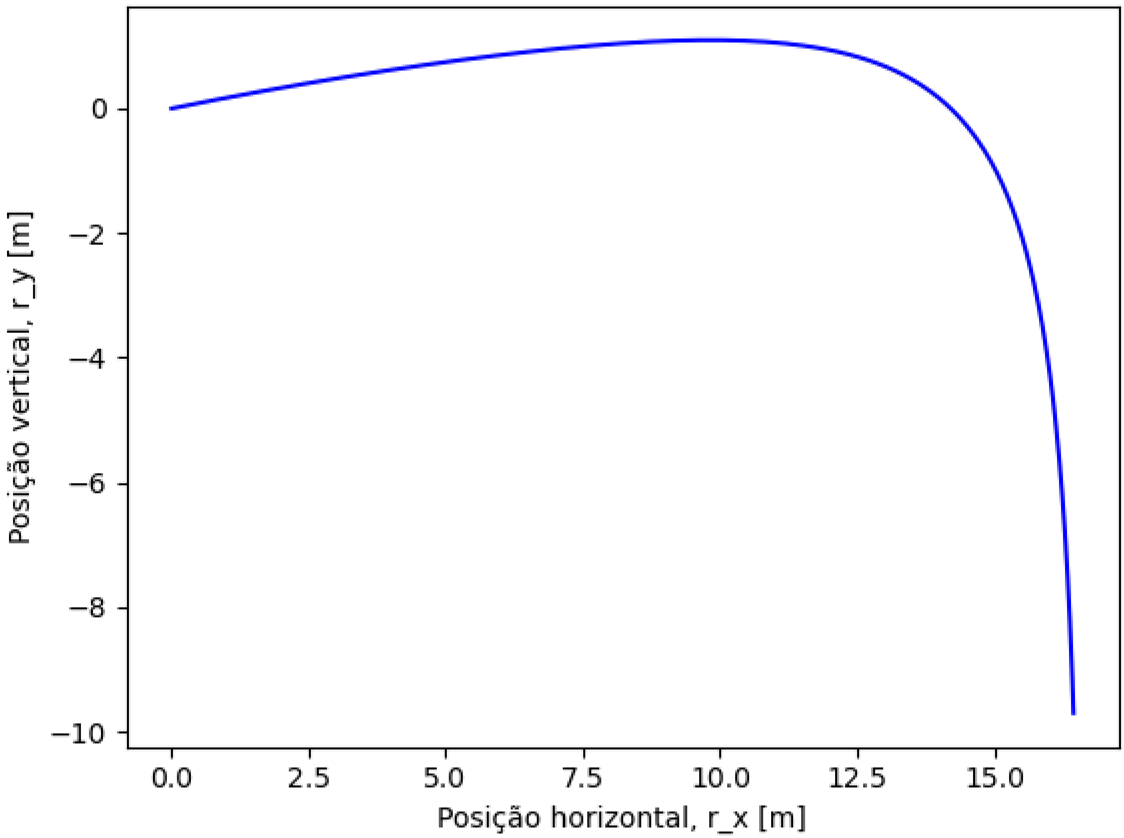
ou

$$\begin{cases} a_x & = & -m\, D\, |\mathbf{v}|\, v_x \\ a_y & = & -g - m\, D\, |\mathbf{v}|\, v_y \end{cases}$$

Podemos agora atualizar o programa anterior:

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

t0 = 0.0                    # condição inicial, tempo [s]
tf = 2.0                    # limite do domínio, tempo final [s]
v0 = 100.0 * 1000 / 3600    # condição inicial, módulo da velocidade inicial [m/s]
theta0 = 10 * np.pi / 180.0 # condição inicial, ângulo do vetor velocidade inicial [r
dt = 0.01                   # passo [s]

g = 9.8                     # aceleração gravítica [m/s^2]
v_T = 100.0 * 1000 / 3600   # velocidade terminal [m/s]
D = g / v_T ** 2            # coeficiente de resistência do ar [m^-1]

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

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

# inicializar solução, velocidade [m/s]
v = np.zeros([2, np.size(t)])
v[:,0] = v0 * np.array([np.cos(theta0), np.sin(theta0)])  # velocidade para t = 0

# inicializar solução, posição [m]
```

```python
r = np.zeros([2, np.size(t)])
r[:,0] = np.array([0.0, 0.0])    # posição para t = 0


for i in range(np.size(t) - 1):
    a[0, i] = -D * np.linalg.norm(v) * v[0, i]
    a[1, i] = -g - D * np.linalg.norm(v[:,i]) * v[1, i]
    v[:, i + 1] = v[:, i] + a[:, i] * dt
    r[:, i + 1] = r[:, i] + v[:, i] * dt

plt.plot(r[0,:], r[1,:], 'b-')
plt.xlabel("Posição horizontal, r_x [m]")
plt.ylabel("Posição vertical, r_y [m]")
plt.show()
```



c) Nas condições da alínea b), determine qual a altura máxima atingida pela bola e em que instante. Tem confiança no seu resultado?

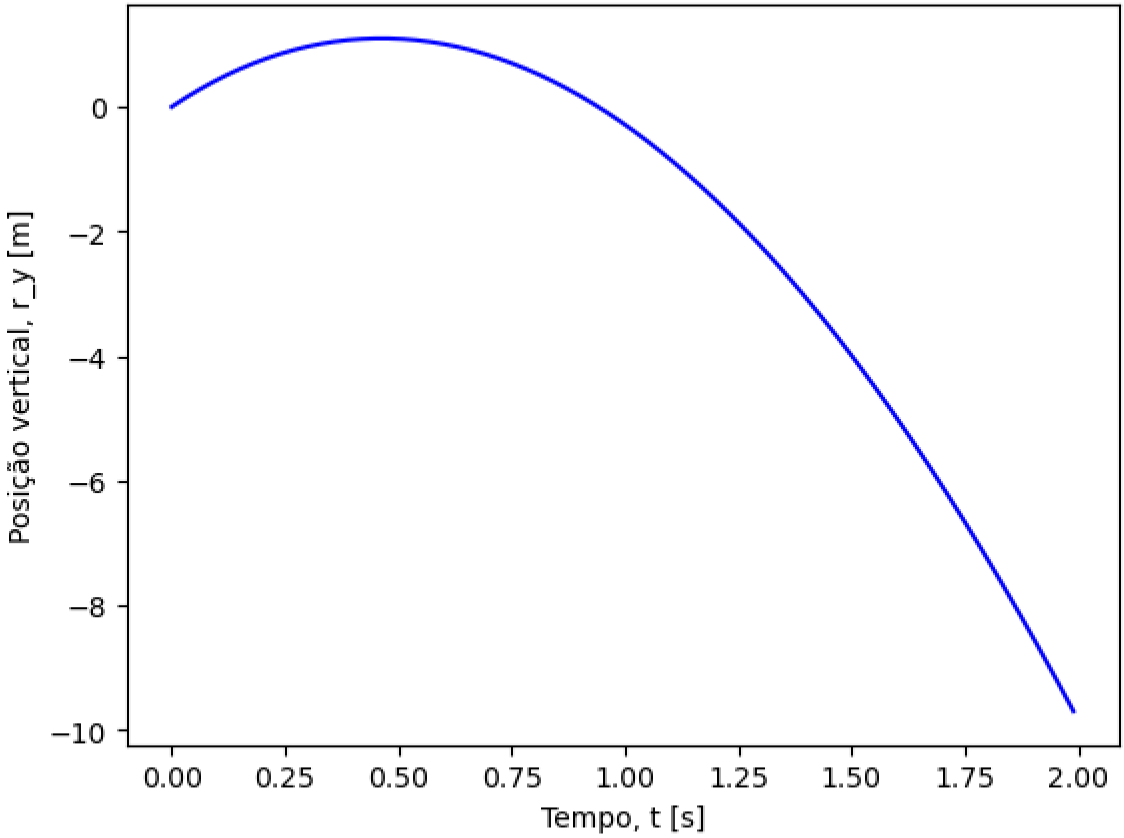
In [4]:
```python
# indice e tempo para o qual a altura é máxima
imax = np.argmax(r[1,:])
tmax = t[imax]
print("Tempo correspondente à altura máxima, tmax = ", tmax, "s")
print("Altura máxima, ymax = ", r[1,imax], "m")
```

```
Tempo correspondente à altura máxima, tmax =  0.46 s
Altura máxima, ymax =  1.098752947196496 m
```

Para verificar se a solução está correta, podemos verificar a solução grafica e verificar se *faz sentido*.

A altura máxima é $y_{\max} = 1.10\ \mathrm{m}$, i.e., menor do que sem atrito do ar ($y_{\max} = 1.18\ \mathrm{m}$)

In [5]:
```python
plt.plot(t, r[1,:], 'b-')
plt.xlabel("Tempo, t [s]")
plt.ylabel("Posição vertical, r_y [m]")
plt.show()
```



d) Nas condições da alínea b), qual o alcance (distância entre a posição onde foi chutada e o ponto onde alcançou no campo) da trajetória da bola e quanto tempo demorou? Tem confiança no seu resultado?

Para verificar se a solução está correta, podemos verificar a solução grafica e verificar se *faz sentido*.

O alcance máximo é $x_{\mathrm{max}} = 14.25\ \mathrm{m}$, i.e., muito menor do que sem atrito do ar ($x_{\mathrm{max}} = 26.9\ \mathrm{m}$). Portanto a solução é razoável do ponto de vista físico.

In [6]:
```
# indice e tempo para o qual a bola atinge o alcance máximo (volta a atingir o solo).
izero = np.size(r[1,:]) - np.size(r[1, r[1,:]<0])   # aqui usamos a "indexação condic
tzero = t[izero]
print("Tempo correspondente ao alcance máximo, tzero = ", tzero, "s")
print("Alcance máximo da bola, xmax = ", r[0,izero], "m")
```

```
Tempo correspondente ao alcance máximo, tzero =  0.9500000000000001 s
Alcance máximo da bola, xmax =  14.253819820440722 m
```

Problema 2

Determinar se é golo ou não, após a bola ser chutada do canto com rotação. Implementar o movimento da bola com rotação, usando o método de Euler.

Modificar o programa anterior e adicionar a parte do método de Euler correspondente à dimensão extra z.

A força que atua na bola é dada por:

$$\mathbf{F} = -mg\,\hat{\mathbf{j}} - m\,D\,|\mathbf{v}|\,\mathbf{v} + \frac{1}{2}A\,\rho_{\mathrm{ar}}\,R\,\boldsymbol{\omega}\times\mathbf{v}$$

ou seja, as componentes são dadas por,

$$\begin{cases} F_x &= -m\,D\,|\mathbf{v}|\,v_x + A\,\rho_{\mathrm{ar}}\,R\,\omega_y\,v_z/2 \\ F_y &= -mg - m\,D\,|\mathbf{v}|\,v_y \\ F_z &= -m\,D\,|\mathbf{v}|\,v_z - A\,\rho_{\mathrm{ar}}\,R\,\omega_y\,v_x/2 \end{cases}$$

Assim, a aceleração é dada por,

$$\begin{cases} a_x &= -D\,|\mathbf{v}|\,v_x + A\,\rho_{\mathrm{ar}}\,R\,\omega_y\,v_z/2m \\ a_y &= -g - D\,|\mathbf{v}|\,v_y \\ a_z &= -D\,|\mathbf{v}|\,v_z - A\,\rho_{\mathrm{ar}}\,R\,\omega_y\,v_x/2m \end{cases}$$

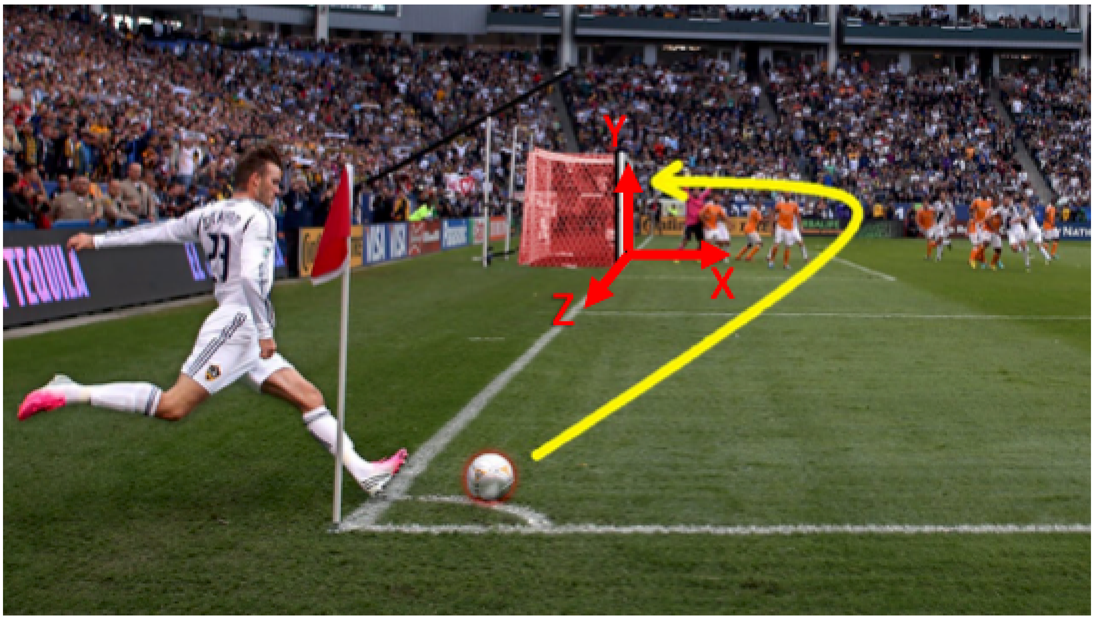
em que:

- $D = g/v_{\mathrm{T}}^2$ é o coeficiente de atrito do ar;
- $v_{\mathrm{T}} = 100\ \mathrm{km/h}$ é a velocidade terminal;
- $A = \pi R^2$ é a área de secção da bola;
- $R = 0.11\ \mathrm{m}$ é o raio da bola;
- $m = 0.45\ \mathrm{kg}$ é a massa da bola
- $\rho_{\mathrm{ar}} = 1.225\ \mathrm{km/m}^3$

e as seguintes condições iniciais:

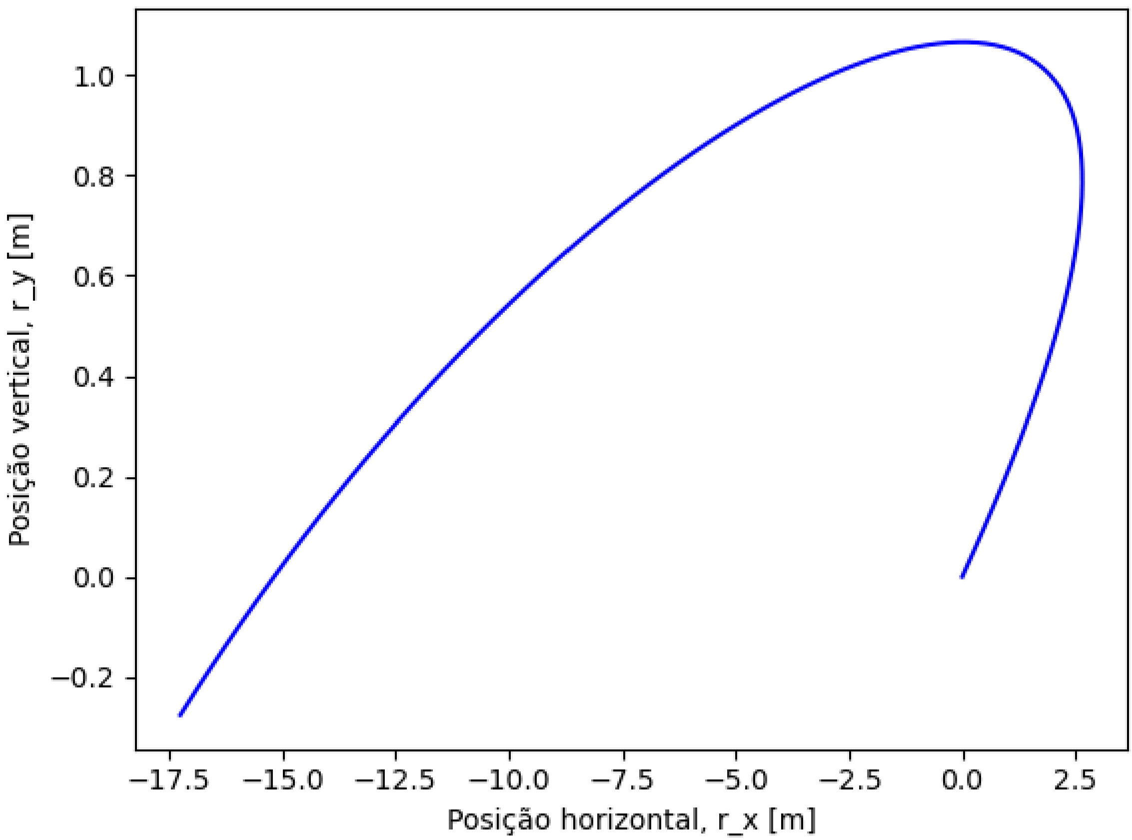
- $\mathbf{r}(t=0) = (0,\,0,\,23.8)\ \mathrm{m}$
- $\mathbf{v}(t=0) = (25,\,5,\,-50)\ \mathrm{m/s}$
- $\boldsymbol{\omega}(t=0) = (0,\,390,\,0)\ \mathrm{rad/s}$
- $t_0 = 0\ \mathrm{s}$

Vamos assumir o seguinte sistema de eixos, com orígem na linha de fundo **ao centro da baliza**.



In [7]:
```
import numpy as np
import matplotlib.pyplot as plt

t0 = 0.0                            # condição inicial, tempo [s]
tf = 1.0                            # limite do domínio, tempo final [s]
dt = 0.01                           # passo [s]

r0 = np.array([0.0, 0.0, 23.8])    # condição inicial, velocidade inicial [m/s]
v0 = np.array([25.0, 5.0, -50.0]) # condição inicial, velocidade inicial [m/s]
w = 390.0                           # condição inicial, velocidade angular CONSTANTE [r
```

```
g = 9.8                       # aceleração gravítica [m/s^2]
v_T = 100.0 * 1000 / 3600     # velocidade terminal [m/s]
D = g / v_T ** 2              # coeficiente de resistência do ar [m^-1]
R = 0.11                      # raio da bola [m]
A = np.pi * R ** 2           # área da secção da bola
m = 0.45                      # massa da bola [kg]
rho = 1.225                   # densidade do ar [kg/m^3]

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

# inicializar solução, aceleração [m/s^2]
a = np.zeros([3, np.size(t)])

# inicializar solução, velocidade [m/s]
v = np.zeros([3, np.size(t)])
v[:,0] = v0

# inicializar solução, posição [m]
r = np.zeros([3, np.size(t)])
r[:,0] = r0


for i in range(np.size(t) - 1):
    a[0, i] = -D * np.linalg.norm(v[:,i]) * v[0,i] + A * rho * R * w * v[2,i] / (2 *
    a[1, i] = -g - D * np.linalg.norm(v[:,i]) * v[1, i]
    a[2, i] = -D * np.linalg.norm(v[:,i]) * v[2,i] - A * rho * R * w * v[0,i] / (2 *
    v[:, i + 1] = v[:, i] + a[:, i] * dt
    r[:, i + 1] = r[:, i] + v[:, i] * dt

plt.plot(r[0,:], r[1,:], 'b-')
plt.xlabel("Posição horizontal, r_x [m]")
plt.ylabel("Posição vertical, r_y [m]")
plt.show()
```



In [8]:

```
# indice e tempo para o qual a bola atinge a "linha de fundo", ie, instante de
# tempo a partir do qual a coordenada x da bola se torna negativa.
ixzero = np.size(r[0,:]) - np.size(r[0, r[0,:]<0])   # usar indexação condicional
```

```python
txzero = t[ixzero]
print("Tempo correspondente ao cruzamento da linha de fundo, txzero =", tzero, "s")
print("Coordenadas da bola quando cruza a linha de fundo:")
print("    x = ", r[0,ixzero], "m")
print("    y = ", r[1,ixzero], "m")
print("    z = ", r[2,ixzero], "m")
```

```
Tempo correspondente ao cruzamento da linha de fundo, txzero = 0.9500000000000001 s
Coordenadas da bola quando cruza a linha de fundo:
   x =  -0.10174954431280686 m
   y =  1.0662007698467821 m
   z =  2.733224968674382 m
```

Conlcusões:

- A bola não cruza a linha de fundo por cima da baliza ($y = 0.617$ m comparado com 2.44 m da altura da baliza);
- A bola cruza a linha de fundo a $z = 2.73$ m do centro da baliza, ie, entra na baliza. Este valor é inferior a metade da largura da baliza (7.32 m).