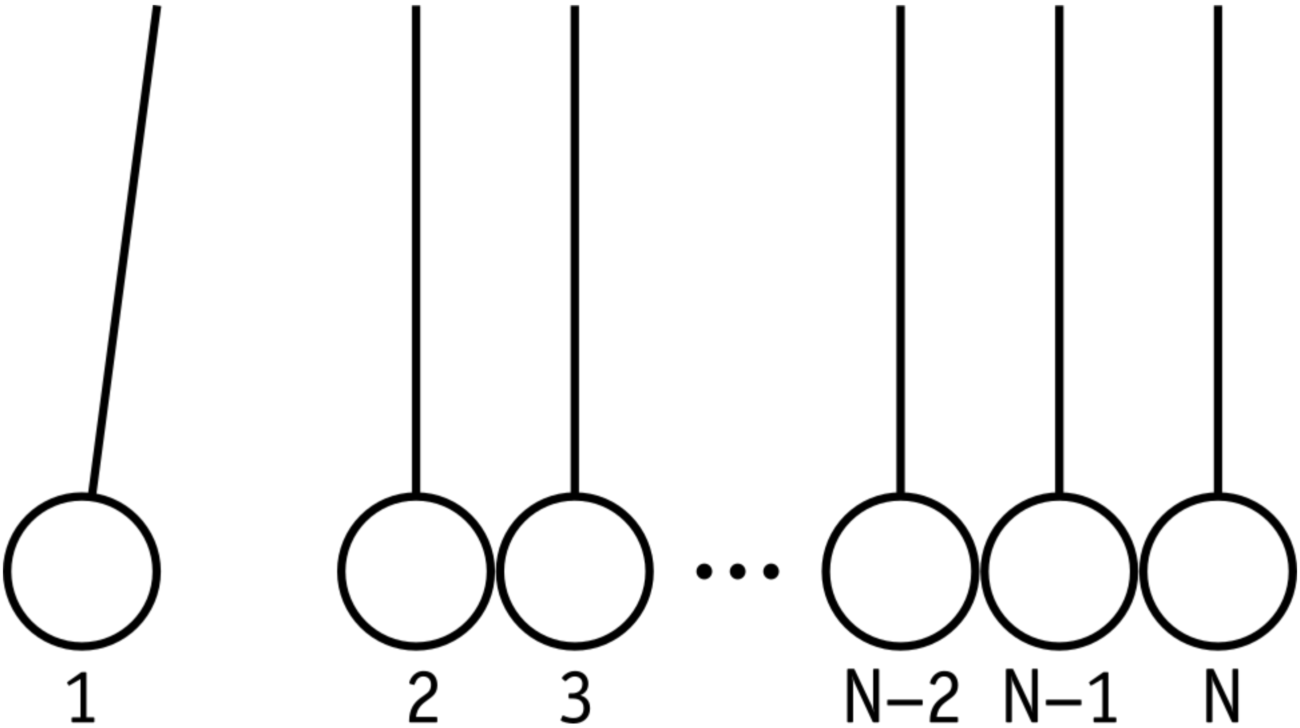
Modelação de Sistemas Físicos - Aula Prática nº9

Realização e resolução de problemas sobre:

- Colisões

Simular a dinâmica do Pêndulo de Newton

Este aparelho consiste numa série de esferas de material duro, suspensa em cordas. Todas as esferas são iguais e suspensas lado-a-lado a distâncias iguais aos seus diâmetros.



Quando não estão a interagir com as outras esferas, cada esfera move-se como um pêndulo simples, sujeito a uma força restauradora devido a gravidade:

$$F_{\text{grav},i} = -m\frac{g}{l}(x_i - x_{i,0})$$

Quando as esferas se tocam, modelamos a interação entre as esferas por via de uma força elástica, semelhante à força de uma mola. A força na esfera i devido ao impacto da esfera $i-1$ é:

$$F_{\text{elast},i,i-1} = \begin{cases} k(x_i - x_{i-1} - d)^2, & \text{for } \ x_i - x_{i-1} < d \\ 0, & \text{for } \ x_i - x_{i-1} \geq d \end{cases}$$

and

$$F_{\text{elast},i,i+1} = -F_{\text{elast},i+1,i}$$

Exercício 1: Duas esferas

1. Simule o movimento de duas esferas sujeitas às forças descritas em cima.

Use o método de Euler-Cromer. Tenha cuidado com a escolha de Δt.

Use os seguintes valores para os parâmetros:

Proprieadades físicas	
$d = 0.1\ \mathrm{m}$	diâmetro das esferas
$l = 10d$	comprimento das cordas
$m = 0.3\ \mathrm{kg}$	massa das esferas
x_i	posicao da esfera i
$x_{i,0} = d \cdot i$	posicao de equilibrio da esfera i
$k = 10^7\ \mathrm{N/m^2}$	coeficiente da força elástica
$g = 9.8\ \mathrm{m/s^2}$	aceleracao gravitica

Condições iniciais	
$x_0(0) = -5d$	Posição da primeira esfera
$v_0(0) = 0$	Velocidade inicial da primeira esfera
$x_1(0) = d$	Posição da segunda esfera
$v_1(0) = 0$	Velocidade inicial da segunda esfera

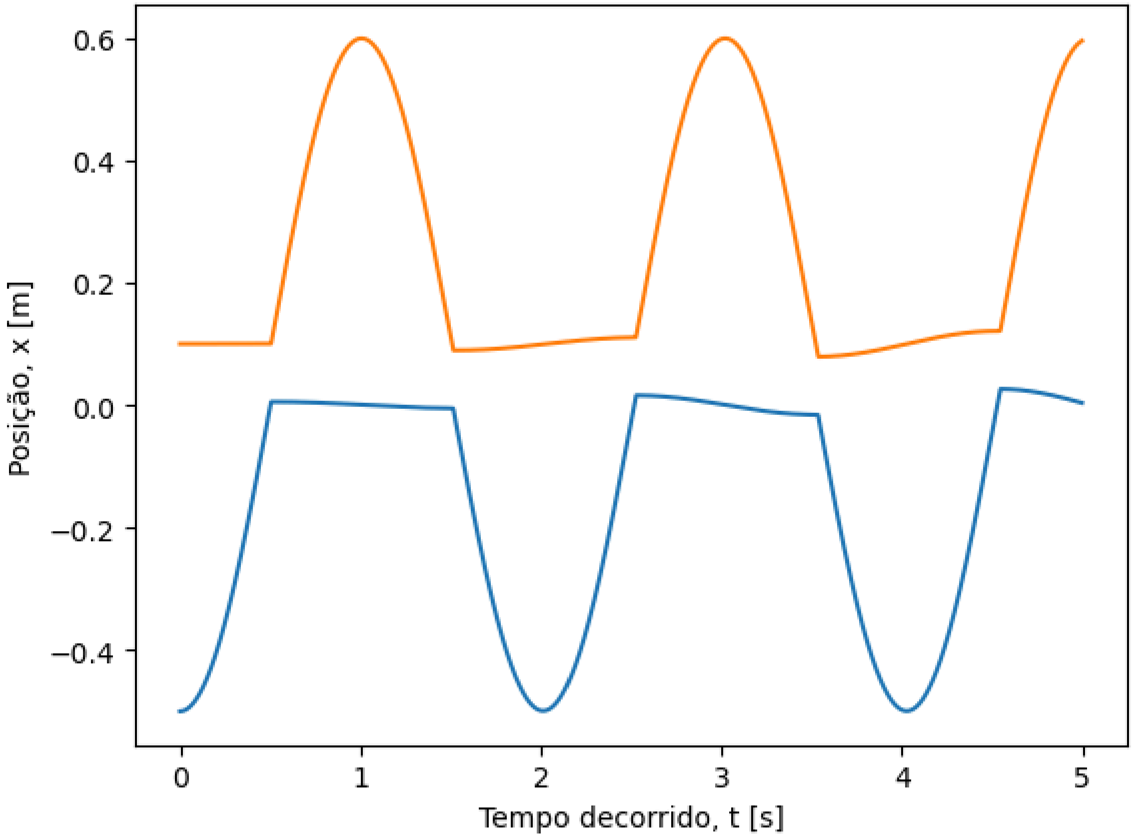
Simule o movimento durante 5 segundos.

Pode usar as funções fornecidas `acc_toque()` e `acc_i()` para calcular a aceleração de cada esfera.

```
def acc_toque(dx, d):
    # calcular a aceleração de uma esfera devido ao contacto
    # com a esfera à sua direita
    k = 1e7
    q = 2.0
    if dx < d:
        a = k * abs(dx - d)**q / m
    else:
        a = 0.0
    return a
def acc_i(i, x):
    # calcular a aceleração de esfera i, cuja posicao de
    # equilibrio é d*i
    a = 0

    if i > 0: # a primeira esfera não tem vizinho à sua esquerda
        a += acc_toque(x[i] - x[i - 1], d)
    if i < (N-1): # a última esfera não tem vizinho à sua direita
        a -= acc_toque(x[i + 1] - x[i], d)

    # aceleração de gravidade, afeta todas as esferas
    a -= g * (x[i] - d * i) / l
    return a
```

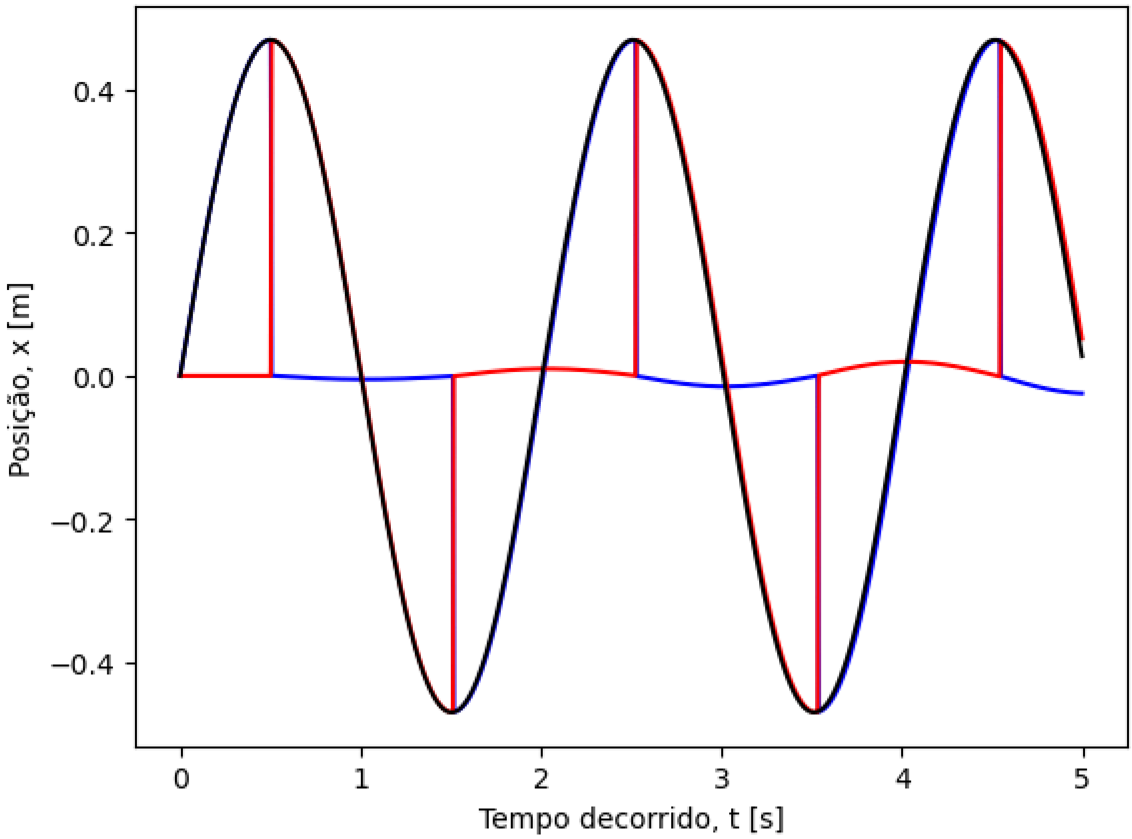
In [2]:
```
import numpy as np
import matplotlib.pyplot as plt

t0 = 0.0                          # condição inicial, tempo [s]
tf = 5.0                          # limite do domínio, tempo final [s]
dt = 0.001                        # passo [s]
```

```python
# inicializar domínio temporal [s]
t = np.arange(t0, tf, dt)
Nt = np.size(t)

# Parametros
N = 2                               # Numero de esferas
d = 0.1                             # Diametro das esferas [m]
l = 10 * d                          # Comprimento das cordas
m = 0.3                             # Massa das esferas [kg]
g = 9.8                             # aceleração gravítica [m/s^2]
k = 1e7                             # constante de forca elastica [N/m^2]
x0 = np.arange(0, N, 1) * d         # posições de equilibrio [m]

# inicializar das granzedas físicas
x_arr = np.zeros((N, Nt))           # posição das esferas i=0...N-1 [m]
v_arr = np.zeros((N, Nt))           # velocidades [m/s]
a_arr = np.zeros((N, Nt))           # aceleraçoes [m/s2]

# Condicoes iniciais
x_arr[:, 0] = x0                      # Posiçoes de equilibrio [m]
x_arr[0, 0] = - 5 * d                 # Esferas levantadas
v_arr[:, 0] = np.zeros(N)             # Velocidade iniciais [m/s]

# calcula a aceleração de uma esfera devido ao contacto
# com outra esfera à sua direita
def acc_toque(dx):
    if dx < d:
        a = k * abs(dx - d)**2 / m
    else:
        a = 0.0
    return a

# calcular a aceleração de esfera i, cuja posicao de
# equilibrio é d * i
def acc_i(i, x):
    a = 0

    if i > 0: # a primeira esfera não tem vizinho à sua esquerda
        a += acc_toque(x[i] - x[i - 1])
    if i < (N - 1): # a última esfera não tem vizinho à sua direita
        a -= acc_toque(x[i + 1] - x[i])

    # aceleração de gravidade, afeta todas as esferas
    a -= g * (x[i] - d * i) / l
    return a

# Método de Euler-Cromer
for j in range(np.size(t) - 1):    # loop no tempo
    for i in range(0, N):          # loop nas esferas
        a_arr[i, j] = acc_i(i, x_arr[:, j])

    v_arr[:, j + 1] = v_arr[:, j] + a_arr[:, j] * dt
    x_arr[:, j + 1] = x_arr[:, j] + v_arr[:, j + 1] * dt

# Representação grafica da posição
for i in range(0, N):          # loop nas esferas
    plt.plot(t, x_arr[i, :])

plt.xlabel("Tempo decorrido, t [s]")
plt.ylabel("Posição, x [m]")
plt.show()
```



2. Calcule o momento total em cada instante, e apresente os resultados num gráfico.

In [3]:
```
# Calculo do momento linear
p_arr = m * v_arr
p_tot = p_arr[0, :] + p_arr[1, :]

# Representação grafica do momento
plt.plot(t, p_arr[0, :], 'b-', t, p_arr[1, :], 'r-', t, p_tot, 'k-')
plt.xlabel("Tempo decorrido, t [s]")
plt.ylabel("Posição, x [m]")
plt.show()
```

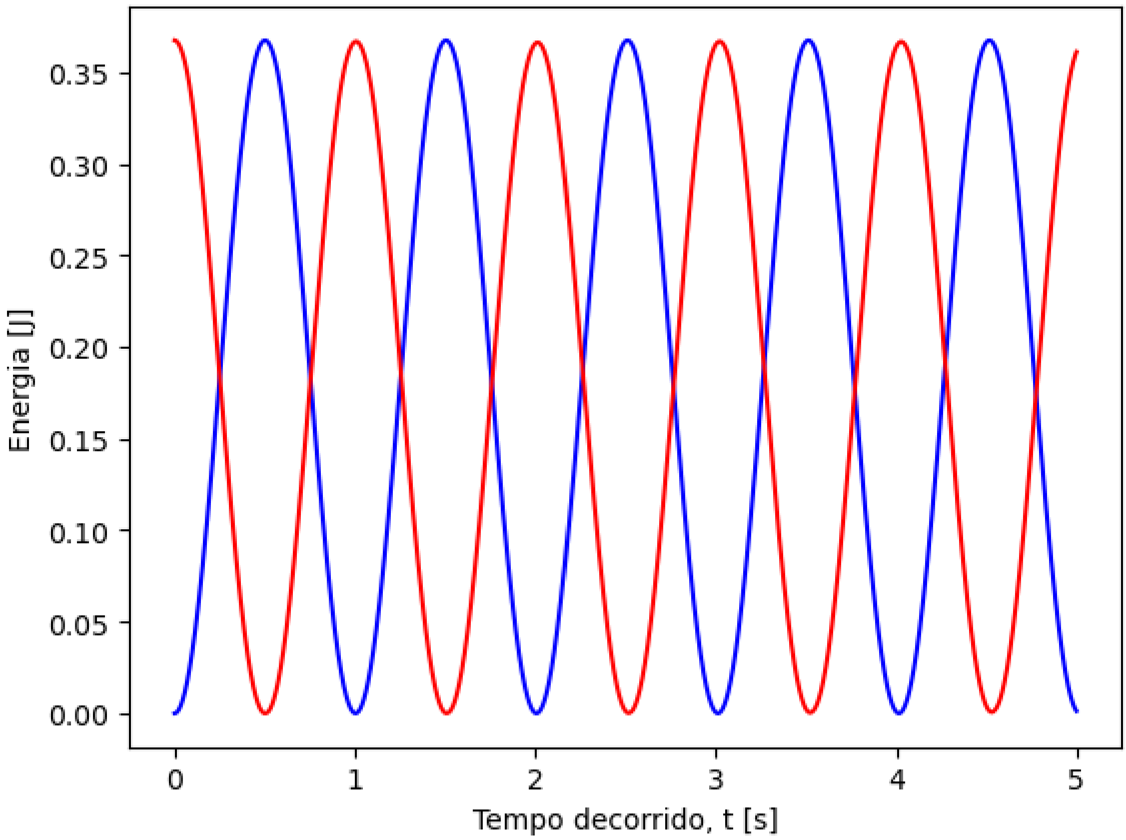


3. Calcule a energia cinética e a energia potencial gravitica em cada instante, e apresente-as num gráfico.

Nota: A energia potencial gravitica da esfera i é:

$$E_{\mathrm{p},i} = \frac{1}{2} m \frac{g}{l} (x_i - x_{i,0})^2.$$

In [4]:
```
# Energia cinética
E_c = p_tot**2 / (2 * m)

# Energia potencial
#E_p = np.zeros(np.size(E_c))
E_p  = m * g * (x_arr[0, :] - x0[0])**2 / (2 * l)
E_p += m * g * (x_arr[1, :] - x0[1])**2 / (2 * l)

# Representação das energias cinetica e potencial
plt.plot(t, E_c, 'b-', t, E_p, 'r-')
plt.xlabel("Tempo decorrido, t [s]")
plt.ylabel("Energia [J]")
plt.show()
```



4. Experimente com outras condições iniciais.

Pergunta 1:

O momento total é conservado? A energia total é conservada? Esses resultados são esperados? Discuta.

Exercício 2: Multiplas esferas

1. Adapte o código do exercício anterior para simular o movimento de N

esferas.

O programa deve ser feito tal que seja possível escolher quantas esferas serão levantadas inicialmente. Cada esfera elevada deve começar a uma distância de $-5d$ da sua posição de equilíbrio.

Por exemplo, se foram duas esferas levantadas:

$$x_0(0) = -5d$$

$$x_0(1) = -4d$$

$$x_0(i) = id, \;\; i > 1$$

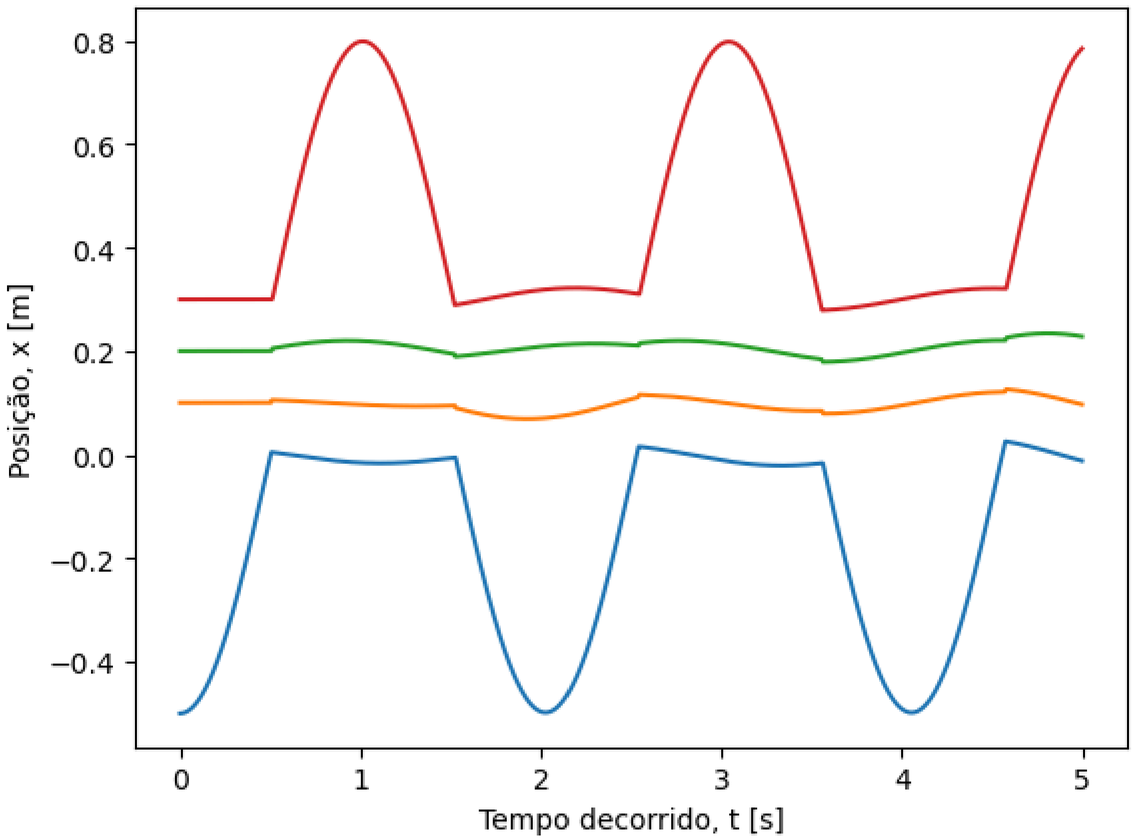
As velocidades iniciais devem ser todas nulas.

2. Simule o movimento de 4 esferas com 1 levantada inicialmente. Faça o

gráfico das posições em função do tempo.

In [13]:
```
import numpy as np
import matplotlib.pyplot as plt

t0 = 0.0                          # condição inicial, tempo [s]
tf = 5.0                          # limite do domínio, tempo final [s]
dt = 0.001                        # passo [s]

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

# Parametros
N = 4                             # Numero de esferas
d = 0.1                           # Diametro das esferas [m]
l = 10 * d                        # Comprimento das cordas
m = 0.3                           # Massa das esferas [kg]
g = 9.8                           # aceleração gravítica [m/s^2]
k = 1e7                           # constante de forca elastica [N/m^2]
x0 = np.arange(0, N, 1) * d       # posições de equilibrio [m]

# inicializar das granzedas físicas
x_arr = np.zeros((N, Nt))         # posição das esferas i=0...N-1 [m]
v_arr = np.zeros((N, Nt))         # velocidades [m/s]
a_arr = np.zeros((N, Nt))         # aceleraçoes [m/s2]

# Condicoes iniciais
x_arr[:, 0] = x0                   # Posiçoes de equilibrio [m]
x_arr[0, 0] = - 5 * d              # Esferas levantadas
v_arr[:, 0] = np.zeros(N)          # Velocidades iniciais [m/s]


# calcula a aceleração de uma esfera devido ao contacto
# com outra esfera à sua direita
def acc_toque(dx):
    if dx < d:
        a = k * abs(dx - d)**2 / m
    else:
        a = 0.0
    return a

# calcular a aceleração de esfera i, cuja posicao de
```

```python
# equilibrio é d * i
def acc_i(i, x):
    a = 0

    if i > 0: # a primeira esfera não tem vizinho à sua esquerda
        a += acc_toque(x[i] - x[i - 1])
    if i < (N - 1): # a última esfera não tem vizinho à sua direita
        a -= acc_toque(x[i + 1] - x[i])

    # aceleração de gravidade, afeta todas as esferas
    a -= g * (x[i] - d * i) / l
    return a

# Método de Euler-Cromer
for j in range(np.size(t) - 1):     # loop no tempo
    for i in range(0, N):           # loop nas esferas
        a_arr[i, j] = acc_i(i, x_arr[:, j])

    v_arr[:, j + 1] = v_arr[:, j] + a_arr[:, j] * dt
    x_arr[:, j + 1] = x_arr[:, j] + v_arr[:, j + 1] * dt

# Representação grafica da posição
for i in range(0, N):           # loop nas esferas
    plt.plot(t, x_arr[i, :])

plt.xlabel("Tempo decorrido, t [s]")
plt.ylabel("Posição, x [m]")
plt.show()
```

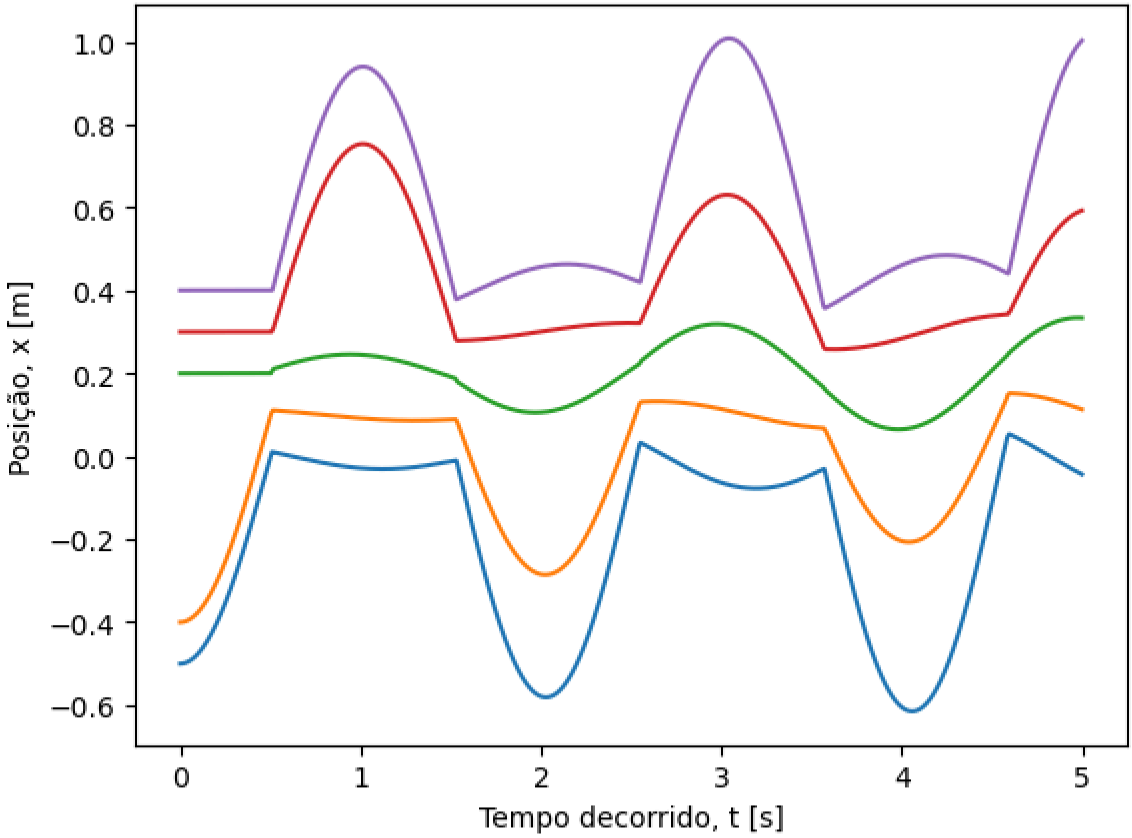


3. Simule o movimento de 5 esferas com 2 levantadas inicialmente. Faça o

gráfico das posições em função do tempo.

In [6]:
```python
import numpy as np
import matplotlib.pyplot as plt

t0 = 0.0                          # condição inicial, tempo [s]
tf = 5.0                          # limite do domínio, tempo final [s]
```

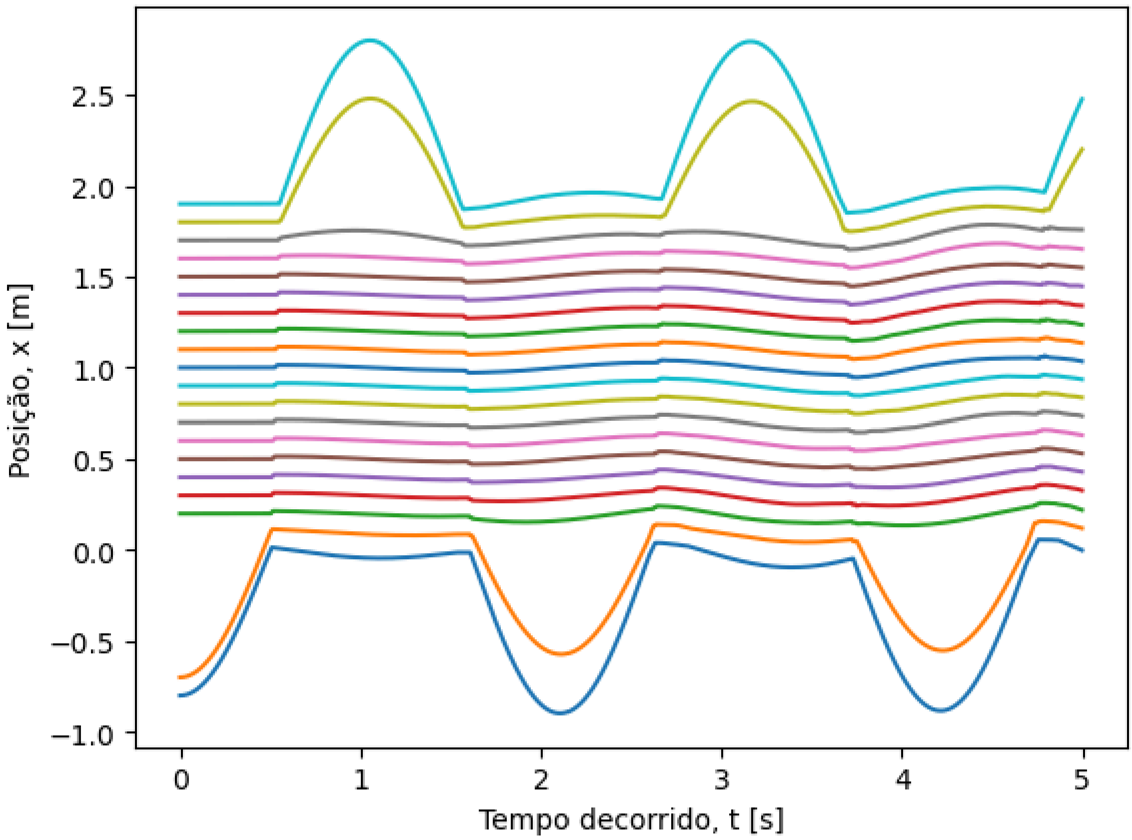
```
dt = 0.001                              # passo [s]

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

# Parametros
N = 5                                   # Numero de esferas
d = 0.1                                 # Diametro das esferas [m]
l = 10 * d                              # Comprimento das cordas
m = 0.3                                 # Massa das esferas [kg]
g = 9.8                                 # aceleração gravítica [m/s^2]
k = 1e7                                 # constante de forca elastica [N/m^2]
x0 = np.arange(0, N, 1) * d             # posições de equilibrio [m]

# inicializar das granzedas físicas
x_arr = np.zeros((N, Nt))               # posição das esferas i=0...N-1 [m]
v_arr = np.zeros((N, Nt))               # velocidades [m/s]
a_arr = np.zeros((N, Nt))               # aceleraçoes [m/s2]

# Condicoes iniciais
x_arr[:, 0] = x0                          # Posiçoes de equilibrio [m]
x_arr[0, 0] = - 5 * d                     # Esferas levantadas
x_arr[1, 0] = - 4 * d                     # Esferas levantadas
v_arr[:, 0] = np.zeros(N)                 # Velocidades iniciais [m/s]


# calcula a aceleração de uma esfera devido ao contacto
# com outra esfera à sua direita
def acc_toque(dx):
    if dx < d:
        a = k * abs(dx - d)**2 / m
    else:
        a = 0.0
    return a

# calcular a aceleração de esfera i, cuja posicao de
# equilibrio é d * i
def acc_i(i, x):
    a = 0

    if i > 0: # a primeira esfera não tem vizinho à sua esquerda
        a += acc_toque(x[i] - x[i - 1])
    if i < (N - 1): # a última esfera não tem vizinho à sua direita
        a -= acc_toque(x[i + 1] - x[i])

    # aceleração de gravidade, afeta todas as esferas
    a -= g * (x[i] - d * i) / l
    return a

# Método de Euler-Cromer
for j in range(np.size(t) - 1):    # loop no tempo
    for i in range(0, N):          # loop nas esferas
        a_arr[i, j] = acc_i(i, x_arr[:, j])

    v_arr[:, j + 1] = v_arr[:, j] + a_arr[:, j] * dt
    x_arr[:, j + 1] = x_arr[:, j] + v_arr[:, j + 1] * dt

# Representação grafica da posição
for i in range(0, N):              # loop nas esferas
    plt.plot(t, x_arr[i, :])

plt.xlabel("Tempo decorrido, t [s]")
plt.ylabel("Posição, x [m]")
plt.show()
```



4. Experimente com outras condições iniciais.

In [11]:
```
import numpy as np
import matplotlib.pyplot as plt

t0 = 0.0                           # condição inicial, tempo [s]
tf = 5.0                           # limite do domínio, tempo final [s]
dt = 0.001                         # passo [s]

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

# Parametros
N = 20                              # Numero de esferas
d = 0.1                            # Diametro das esferas [m]
l = 10 * d                         # Comprimento das cordas
m = 0.3                            # Massa das esferas [kg]
g = 9.8                            # aceleração gravítica [m/s^2]
k = 1e7                            # constante de forca elastica [N/m^2]
x0 = np.arange(0, N, 1) * d        # posições de equilibrio [m]

# inicializar das granzedas físicas
x_arr = np.zeros((N, Nt))          # posição das esferas i=0...N-1 [m]
v_arr = np.zeros((N, Nt))          # velocidades [m/s]
a_arr = np.zeros((N, Nt))          # aceleraçoes [m/s2]

# Condicoes iniciais
x_arr[:, 0] = x0                     # Posiçoes de equilibrio [m]
x_arr[0, 0] = - 5 * d                # Esferas levantadas
x_arr[1, 0] = - 4 * d                # Esferas levantadas
v_arr[:, 0] = np.zeros(N)            # Velocidades iniciais [m/s]


# calcula a aceleração de uma esfera devido ao contacto
# com outra esfera à sua direita
def acc_toque(dx):
    if dx < d:
```

```python
        a = k * abs(dx - d)**2 / m
    else:
        a = 0.0
    return a

# calcular a aceleração de esfera i, cuja posicao de
# equilibrio é d * i
def acc_i(i, x):
    a = 0

    if i > 0: # a primeira esfera não tem vizinho à sua esquerda
        a += acc_toque(x[i] - x[i - 1])
    if i < (N - 1): # a última esfera não tem vizinho à sua direita
        a -= acc_toque(x[i + 1] - x[i])

    # aceleração de gravidade, afeta todas as esferas
    a -= g * (x[i] - d * i) / l
    return a

# Método de Euler-Cromer
for j in range(np.size(t) - 1):     # loop no tempo
    for i in range(0, N):           # loop nas esferas
        a_arr[i, j] = acc_i(i, x_arr[:, j])

    v_arr[:, j + 1] = v_arr[:, j] + a_arr[:, j] * dt
    x_arr[:, j + 1] = x_arr[:, j] + v_arr[:, j + 1] * dt

# Representação grafica da posição
for i in range(0, N):           # loop nas esferas
    plt.plot(t, x_arr[i, :])

plt.xlabel("Tempo decorrido, t [s]")
plt.ylabel("Posição, x [m]")
plt.show()
```



Pergunta 2:

O movimento das esferas nas simulações é como esperado? Em que condições se afasta mais do comportamento esperado? Explique.