Modelação de Sistemas Físicos - Aula Prática nº11

Realização e resolução de problemas sobre Cap. 5:

- Osciladores amortecidos e forçados

- Osciladores caóticos

Exercício 1: Oscilador harmónico forçado

Um corpo de massa $m = 1\,\mathrm{kg}$ move-se num oscilador harmónico forçado. Se a posição de equilíbrio for a origem do eixo, $x_{\mathrm{eq}} = 0\,\mathrm{m}$, o oscilador harmónico tem uma energia potencial $E_{\mathrm{p}} = \frac{1}{2}kx^2$, correspondente a uma força que exerce no corpo,

$$F_{\mathrm{mola}} = -kx.$$

O oscilador é amortecido pela força $-bv$ e sujeito a uma força externa $F_{\mathrm{ext}} = F_0 \cos(\omega_{\mathrm{f}}\, t)$, onde v é a velocidade instantânea e,

- $k = 1\,\mathrm{N/m}$
- $b = 0.05\,\mathrm{kg/s}$
- $F_0 = 7.5\,\mathrm{N}$
- $\omega_{\mathrm{f}} = 0.5\,\mathrm{rad/s}$

a) Calcule numericamente a lei do movimento, assumindo que a velocidade inicial é nula, e que a posição inicial é $x_0 = 4$ m.

Podemos usar o método de Euler-Cromer (adequado para movimentos oscilatórios) para descrever a dinâmica do movimento da massa.

Sendo a força resultante dada por,

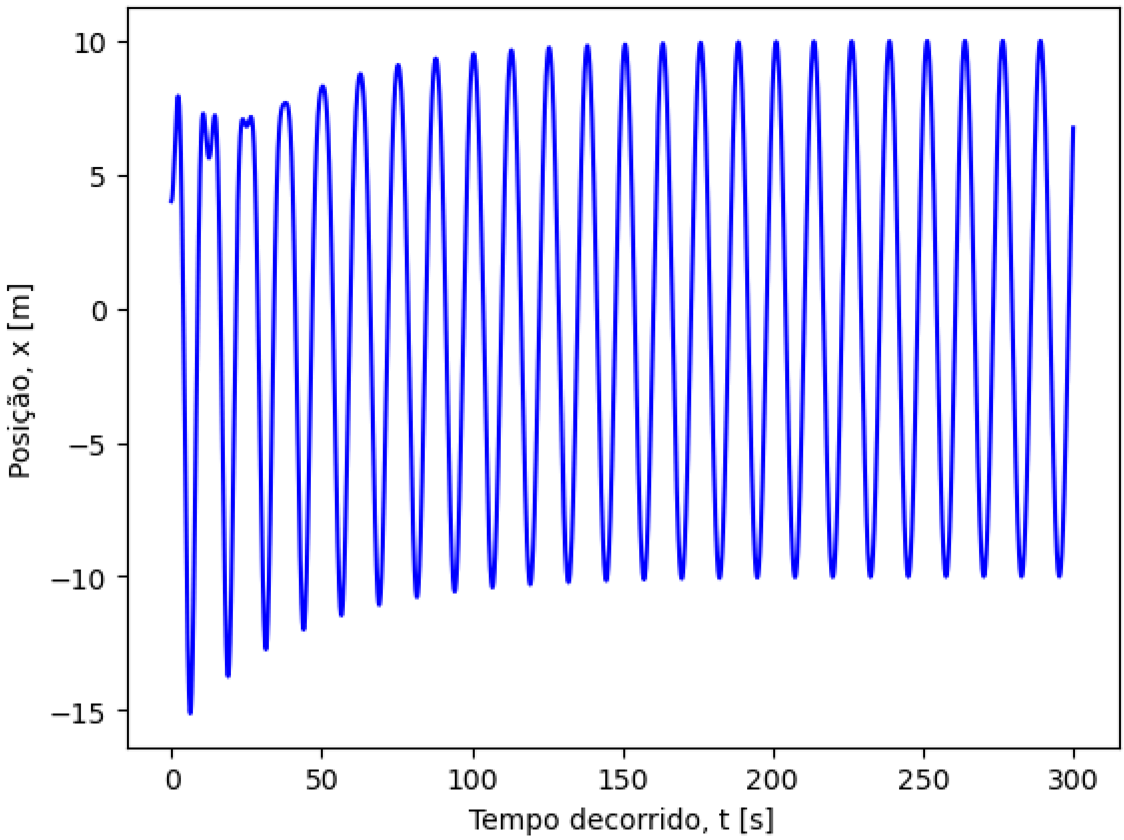
$$F = -kx - bv + F_0 \cos(\omega_{\mathrm{f}}\, t),$$

ou seja, a aceleração instantânea é

$$a = -\left[kx + bv - F_0 \cos(\omega_{\mathrm{f}}\, t)\right]/m,$$

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

t0 = 0.0                          # condição inicial, tempo [s]
tf = 300.0                         # limite do domínio, tempo final [s]
dt = 0.001                        # passo [s]

x0 = 4.0                          # condição inicial, posição inicial [m]
v0 = 0.0                          # condição inicial, velocidade inicial [m/s]

m = 1.0                           # massa [kg]
```

```
k = 1.0                           # constante da mola [N/m]
b = 0.05                          # constante de amortecimento [kg/s]
F_0 = 7.5                         # amplitude da força externa [N]
ω_f = 0.5                         # frequência angular da força externa [rad/s]

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

# inicializar solução
a = np.zeros(np.size(t))          # aceleração [m/s2]
v = np.zeros(np.size(t))          # velocidade [m/s]
x = np.zeros(np.size(t))          # posição [m]
x[0] = x0
v[0] = v0

# método de Euler-Cromer
for i in range(np.size(t) - 1):
    a[i] = - (k * x[i] + b * v[i] - F_0 * np.cos(ω_f * t[i])) / m
    v[i + 1] = v[i] + a[i] * dt
    x[i + 1] = x[i] + v[i+1] * dt

plt.plot(t, x, 'b-')
plt.xlabel("Tempo decorrido, t [s]")
plt.ylabel("Posição, x [m]")
plt.show()
```



b) Calcule a amplitude do movimento e o seu período no regime estacionário, usando os resultados numéricos.

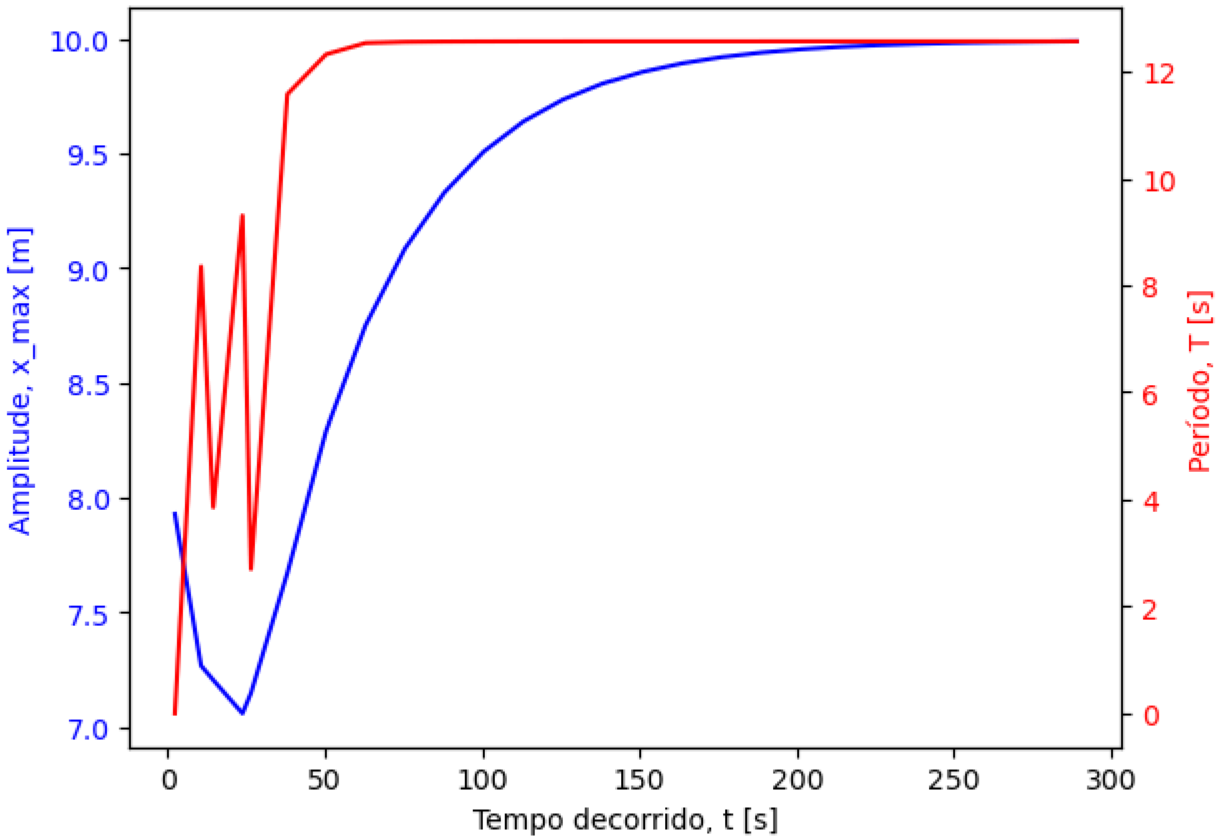
Aqui vamos novamente utilizar a função `maxminv` (interpolação de Lagrange definida na aula anterior) para calcular os vários máximos consecutivos. Daí poderemos obter a amplitude em regime estacionário e o respetivo período.

In [2]:
```
def maxminv(x0,x1,x2,y0,y1,y2):
    # Máximo ou mínimo usando o polinómio de Lagrange
```

```python
    # Dados (input): (x0,y0), (x1,y1) e (x2,y2)
    # Resultados (output): xm, ym
    xab = x0 - x1
    xac = x0 - x2
    xbc = x1 - x2
    a = y0 / (xab * xac)
    b = -y1 / (xab * xbc)
    c = y2 / (xac * xbc)
    xmla = (b + c) * x0 + (a + c) * x1 + (a + b) * x2
    xm = 0.5 * xmla / (a + b + c)
    xta = xm - x0
    xtb = xm - x1
    xtc = xm - x2
    ym = a * xtb * xtc + b * xta * xtc + c * xta * xtb
    return xm, ym
```

Aproveitamos também para representar graficamente a posição e o período de forma a verificar quando atingimos o estado estacionário.

In [3]:
```python
# arrays com valores máximos, respetivos instantes de tempo, e o período
t_max = np.array([])   # instante de tempo nos máximos
x_max = np.array([])   # máximos de amplitude
T = np.array([])       # período entre máximos

# Pesquisar pelo máximos de x.
# Aqui definimos uma "janela corrida" no tempo em passos de 2, i.e, analisamos
# os máximos que ocorrem entre t[i] e t[i+2], com i = 0, 2, 4, 6, etc.
# de forma a evitar encontros duplicados
for i in range(0, np.size(t) - 3, 2):
    # Percorrer domínio temporal em sequências de três:
    # x[i], x[i+1], x[i+2] e respetivos instantes de tempo para i = 0, ..., N-3
    tm, xm = maxminv(t[i], t[i+1], t[i+2], x[i], x[i+1], x[i+2])

    # verificar se extremo está dentro da "janela corrida" (t[i] <-> t[i+2])
    if t[i] < tm and tm < t[i+2]:
        # verificar se é máximo e adicionar a lista se esse for o caso
        if xm > np.maximum(x[i], x[i+2]):
            t_max = np.append(t_max, tm)
            x_max = np.append(x_max, xm)
            # calcular diferenca entre os dois ultimos instantes de tempo
            # e adicionar ao array dos periodos
            T = np.append(T, t_max[np.size(t_max) - 1] - t_max[np.size(t_max) - 2])


fig, ax1 = plt.subplots()
ax1.set_xlabel('Tempo decorrido, t [s]')
ax1.set_ylabel('Amplitude, x_max [m]', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')
ax1.plot(t_max, x_max, 'b-')

ax2 = ax1.twinx()  # criar segundo sistema de eixos com o mesmo eixo Ox
ax2.set_ylabel('Período, T [s]', color='red')
ax2.plot(t_max, T, 'r-')
ax2.tick_params(axis='y', labelcolor='red')

plt.show()
```

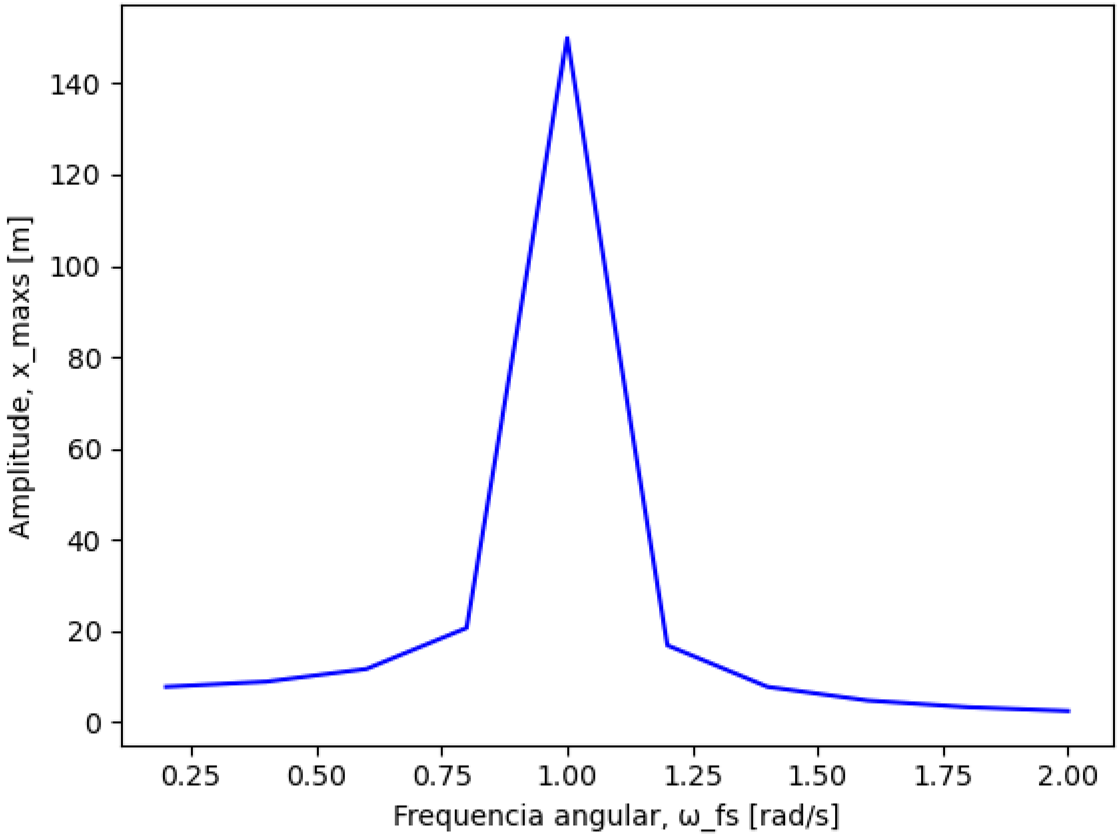


In [4]:
```
print("O período em regime estacionário é T = {0:.2f} s".format(T[-1]))
print("A amplitude em regime estacionário é x_max = {0:.2f} m".format(x_max[-1]))
```

```
O período em regime estacionário é T = 12.57 s
A amplitude em regime estacionário é x_max = 9.99 m
```

c) Repita para valores de ω_f entre 0.2 e 2 rad/s. Faça um gráfico da amplitude em regime estacionário em função de ω_f. Qual a frequência angular ω_f que corresponde à maior amplitude?

In [5]:
```
ω_fs  = np.array([0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0])
x_maxs = np.array([7.81, 8.93, 11.71, 20.71, 149.91, 16.9, 7.79, 4.80, 3.34, 2.50])

plt.plot(ω_fs, x_maxs, 'b-')
plt.xlabel("Frequencia angular, ω_fs [rad/s]")
plt.ylabel("Amplitude, x_maxs [m]")
plt.show()
```



A amplitude máxima em regime estacionário corresponde à frequência angular $\omega_\mathrm{f} = 1\ \mathrm{rad/s}$. Este valor corresponde à frequência angular natural da mola,
$\omega_0 = \sqrt{k/m} = \sqrt{(1\ \mathrm{N/m})/(1\ \mathrm{kg})} = 1\ \mathrm{rad/s}$. A este fenómono chamamos "resonância".

Pergunta 1:

A força externa realiza trabalho? Como é que isso pode ser medido?

Exercício 2: Oscilador Não Harmónico Caótico, com Metodo de Runge-Kutta

Um corpo de massa $1\ \mathrm{kg}$ move-se num oscilador quártico não harmónico forçado. A posição de equilíbrio é em $x = 0\ \mathrm{m}$, e o oscilador tem a energia potencial

$$E_\mathrm{p} = \frac{1}{2}kx^2 + \alpha x^4,$$

que está associado a uma força,

$$F = -kx - 4\alpha x^3.$$

O oscilador é amortecido pela força $-bv$ e sujeito à força externa $F_\mathrm{ext} = F_0 \cos(\omega_\mathrm{f} t)$, onde v é a velocidade instantânea e

- $\alpha = 1\ \mathrm{N/m^3}$
- $k = 0.2\ \mathrm{N/m}$
- $b = 0.01\ \mathrm{kg/s}$
- $F_0 = 5\ \mathrm{N}$

- $\omega_\mathrm{f} = 0.6\,\mathrm{rad/s}$

a) Use o método de Runge-Kutta da 4ª ordem (RK4) para calcular numericamente a lei do movimento, no intervalo de tempo entre 0 até 50 s, considerando que a velocidade inicial é nula e a posição inicial é $x = 1\,\mathrm{m}$.

O Método de Runge-Kutte de ordem 4 (RK4), à semelhânça do método de Euler, é um procedimento iterativo que permite resolver equações do tipo,

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t,y),\ \ \text{sabendo à partida que}\ \ y_0 = y(t_0),$$

em que y é a função (desconhecida) que se pretende obter. Tudo o que sabemos é o seu *declive* (f) e *condições iniciais* (t_0, y_0).

O método RK4 parte da seguinte discretização,

$$y_{i+1} = y_i + \bar{f}_i\delta\tau$$

onde $y_i \equiv y(t_i)$ e $\bar{f}_i$ é um *declive efetivo* que permite obter um valor aproximado do para y_{i+1} em função de y_i e do passo $\delta\tau$.

No método de Euler temos

$$\bar{f}_i = f(t_i, y_i),$$

o que resulta num erro de truncatura local da ordem de $O(\delta\tau^2)$ e um erro global linear em $\delta\tau$.

Segundo o método de Runge-Kutta de ordem 4, o declive $\bar{f}_i$ é obtido de forma iterativa, permitindo obter um valor de y_{i+1} mais próximo do valor real. Assim, $\bar{f}_i$ é definido pela seguinte média de quatro declives k_i, com $(i = 1, \ldots, 4)$,

$$\bar{f}_i = \frac{k_1 + 2k_2 + 2k_3 + k_4}{6},$$

em que,

$$\begin{aligned}
k_1 &= f(t_i, y_i) && (1)\\
k_2 &= f\left(t_i + \frac{\delta\tau}{2},\ y_i + k_1\frac{\delta\tau}{2}\right) && (2)\\
k_3 &= f\left(t_i + \frac{\delta\tau}{2},\ y_i + k_2\frac{\delta\tau}{2}\right) && (3)\\
k_4 &= f(t_i + \delta\tau,\ y_i + k_3\delta\tau). && (4)
\end{aligned}$$

O método pode ser entendido a partir da seguinte representação gráfica,

Método de Runge-Kutta de ordem 4

No presente problema temos uma força total resultante que atua no corpo dada por,

$$F = -kx - 4\alpha x^3 - bv + F_0\cos(\omega_\mathrm{f} t).$$

Daqui retiramos a aceleração instantânea,

$$a(t) = -(kx + 4\alpha x^3 + bv - F_0\cos(\omega_\mathrm{f} t))/m.$$

Podemos então usar o método RK4 para encontrar a velocidade,

$$\frac{\mathrm{d}v}{\mathrm{d}t} = a(t), \;\; \text{em que} \;\; v(0) = 0\,\mathrm{m/s},$$

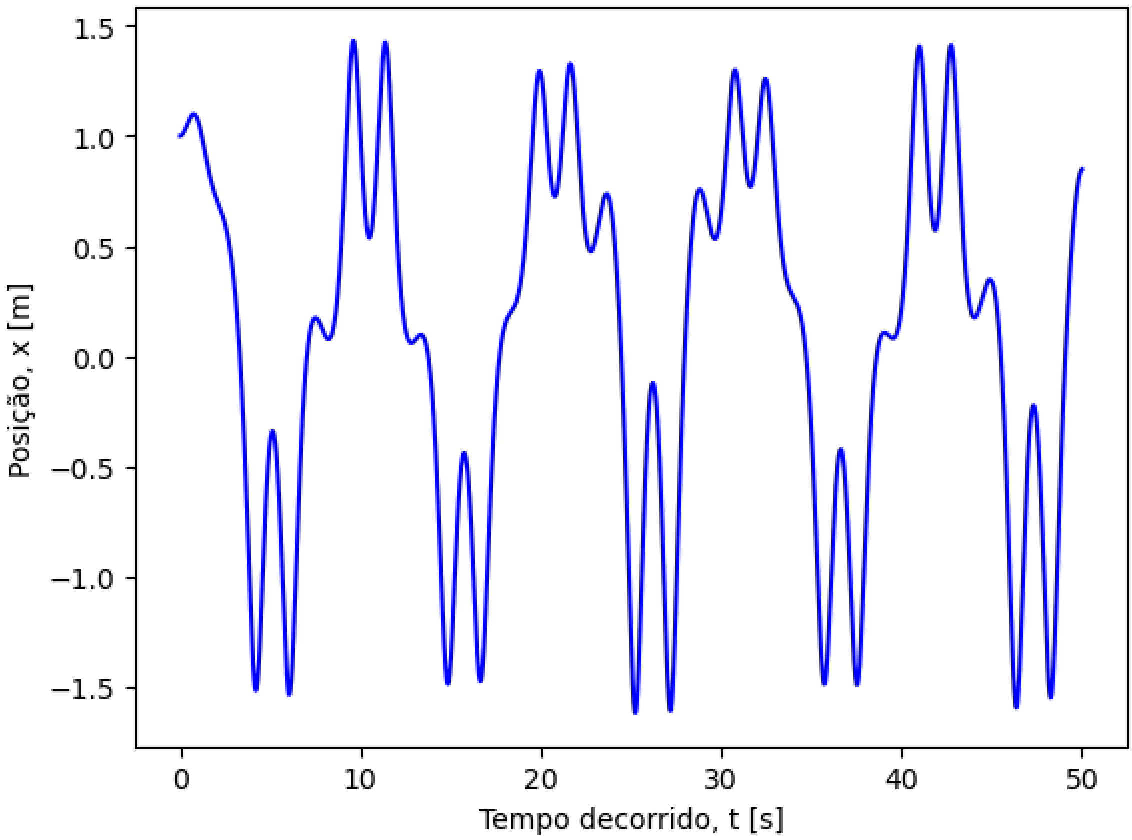
e sequencialmente podemos também encontrar a posição a partir de,

$$\frac{\mathrm{d}x}{\mathrm{d}t} = v(t), \;\; \text{em que} \;\; x(0) = 1\,\mathrm{m},$$

In [6]:

```
import numpy as np
import matplotlib.pyplot as plt

t0 = 0.0                              # condição inicial, tempo [s]
tf = 50.0                             # limite do domínio, tempo final [s]
dt = 0.001                            # passo [s]

x0 = 1.0                              # condição inicial, posição inicial [m]
v0 = 0.0                              # condição inicial, velocidade inicial [m/s]

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

# inicializar solução
a = np.zeros(np.size(t))              # aceleração [m/s2]
v = np.zeros(np.size(t))             # velocidade [m/s]
x = np.zeros(np.size(t))             # posição [m]
x[0] = x0
v[0] = v0


def a_res(t, x, v):
    """
    Aceleração resultante (total) em função do tempo e velocidade
    input:  t = instante de tempo [escalar]
            x = posição instantânea [escalar]
            v = velocidade instantânea [escalar]
    output: aceleração instantânea resultante [escalar]
    """
    m = 1.0                               # massa [kg]
    α = 1.0                               # coef. potencial quártico [N/m3]
    k = 0.2                               # constante da mola [N/m]
    b = 0.01                              # constante de amortecimento [kg/s]
    F_0 = 5.0                             # amplitude da força externa [N]
    ω_f = 0.6                             # frequencia angular da força externa [rad/s]
    return - (k * x + 4 * α * x ** 3 + b * v - F_0 * np.cos(ω_f * t)) / m


def rk4_x_v(t, x, v, a, dt):
    """
    Integração numérica das equações diferenciais:
                        dx/dt   = v(t,x)
            d2x/dt2 = a(t,x)
        Erro global:  proporcional a dt**4
    input:  t = instante de tempo
            x(t) = posição
            v(t) = velocidade
            a = dv/dt = Força(t,x,v) / massa : é uma FUNÇÃO
            dt = passo temporal
    output: xp = x(t+dt)
                    vp = v(t+dt)
    """
```

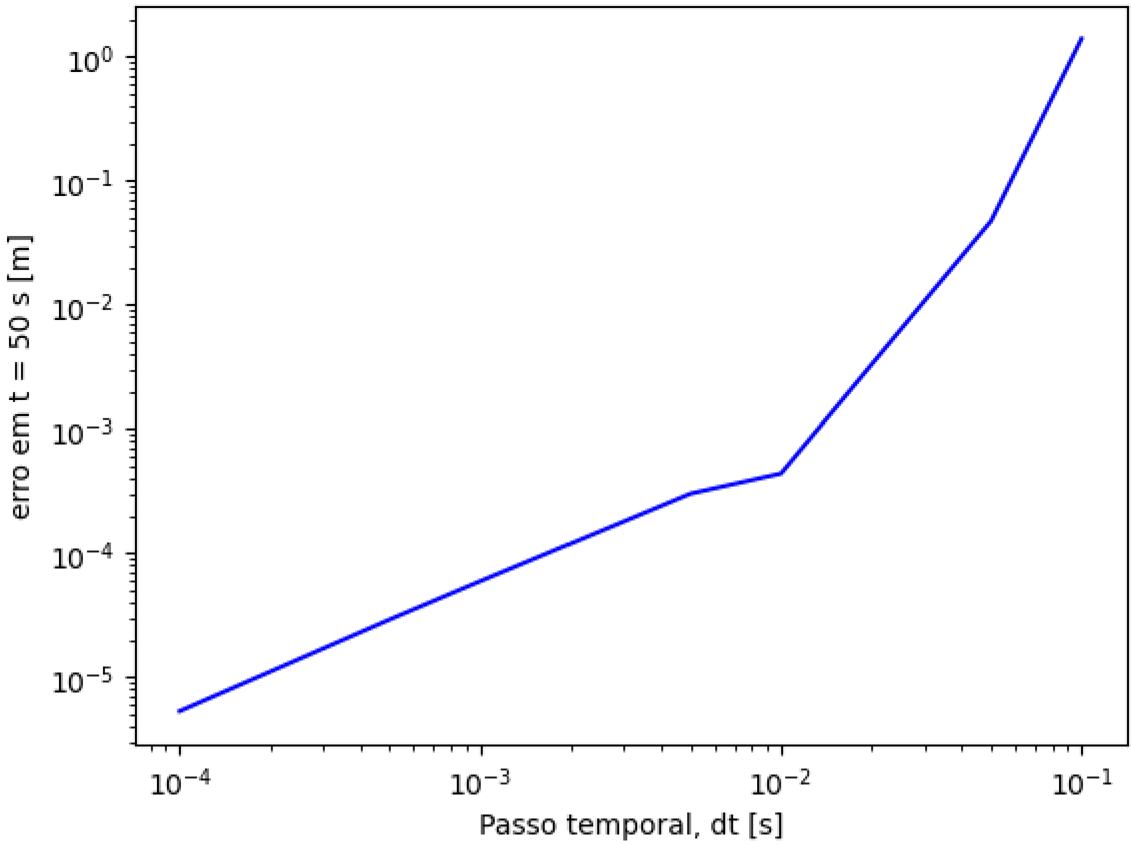
```
    # cálculo dos declives
    k1v = a(t, x, v)
    k1x = v
    k2v = a(t + dt / 2.0, x + k1x * dt / 2.0, v + k1v * dt/ 2.0)
    k2x = (v + k1v * dt / 2.0)
    k3v = a(t + dt / 2.0, x + k2x * dt / 2.0, v + k2v * dt / 2.0)
    k3x = (v + k2v * dt / 2.0)
    k4v = a(t + dt, x + k3x * dt, v + k3v * dt)
    k4x = (v + k3v * dt)
    # cálculo da posição e velocidade
    xp = x + (k1x + 2.0 * k2x + 2.0 * k3x + k4x) / 6.0 * dt
    vp = v + (k1v + 2.0 * k2v + 2.0 * k3v + k4v) / 6.0 * dt
    return xp, vp


# percorrer o domínio temporal e resolver a equação
# de movimento usando o método RK4
for i in range(np.size(t) - 1):
    x[i + 1], v[i + 1] = rk4_x_v(t[i], x[i] ,v[i] ,a_res , dt)


plt.plot(t, x, 'b-')
plt.xlabel("Tempo decorrido, t [s]")
plt.ylabel("Posição, x [m]")
plt.show()

# Guardamos a velocidade e a posição para mais tarde
vA = v
xA = x
```



b) Experimente diferentes valores de δt. Apartir de que valor tem confiança nos resultados?

In [7]:
```
# para dt = 1e-5 obtemos x[-1] = 0.8486637491970063 m

dts  = np.array([0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001])
```

```python
x50 = np.array([-0.5463796756200047, 0.895950675274198, 0.8482249513860772, 0.8483616
erro = np.abs(x50 - 0.8486637491970063)

plt.loglog(dts, erro, 'b-')
plt.xlabel("Passo temporal, dt [s]")
plt.ylabel("erro em t = 50 s [m]")
plt.show()
```



Existe confiânca nos resultados para $\delta t < 0.01$ s. Com este passo o valor da posição ao fim de $t = 50$ s é

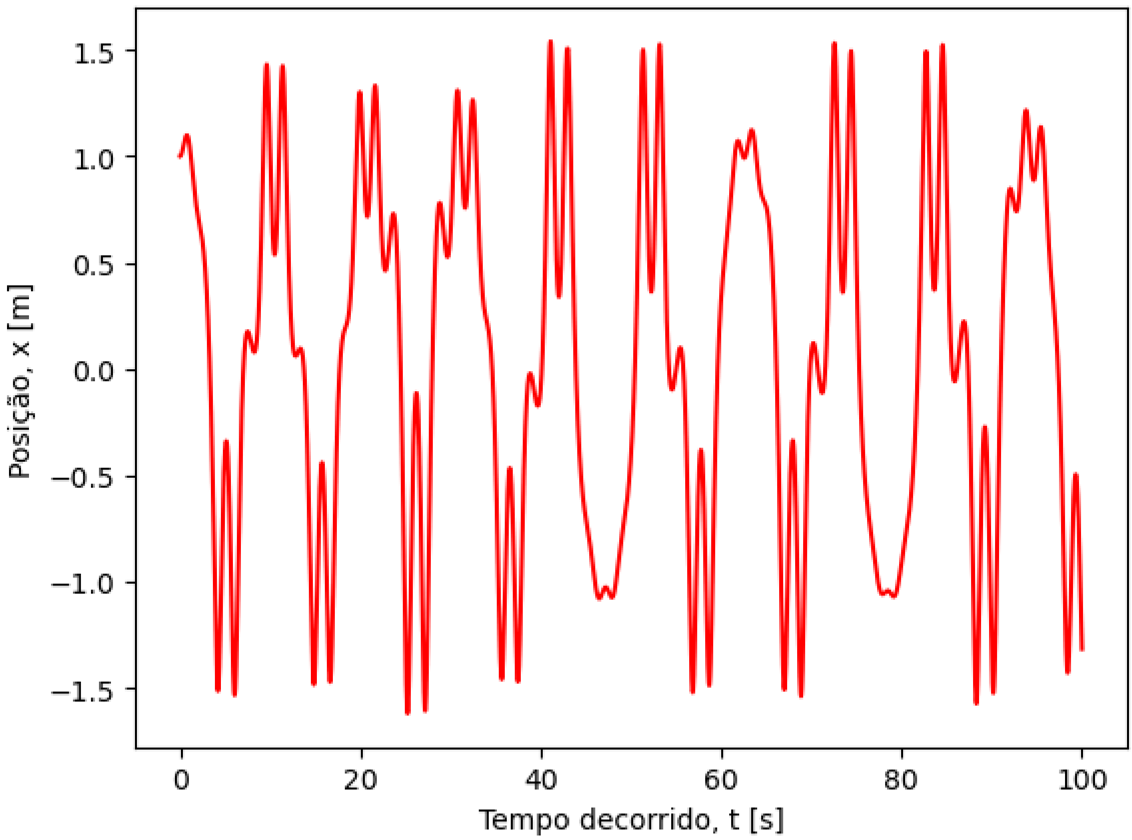
In [8]:
```python
print("x(t = 50 s) = {0:.4f} m".format(x50[2]))
```

```
x(t = 50 s) = 0.8482 m
```

c) Calcule novamente a lei do movimento, agora até $t = 100$ s, no caso em que a velocidade inicial é nula e a posição inicial é $x = 1.0001$ m. O que se observa?

In [9]:
```python
import numpy as np
import matplotlib.pyplot as plt

t0 = 0.0                         # condição inicial, tempo [s]
tf = 100.0                       # limite do domínio, tempo final [s]
dt = 0.001                       # passo [s]

x0 = 1.0001                      # condição inicial, posição inicial [m]
v0 = 0.0                         # condição inicial, velocidade inicial [m/s]

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

# inicializar solução
a = np.zeros(np.size(t))         # aceleração [m/s2]
v = np.zeros(np.size(t))         # velocidade [m/s]
```

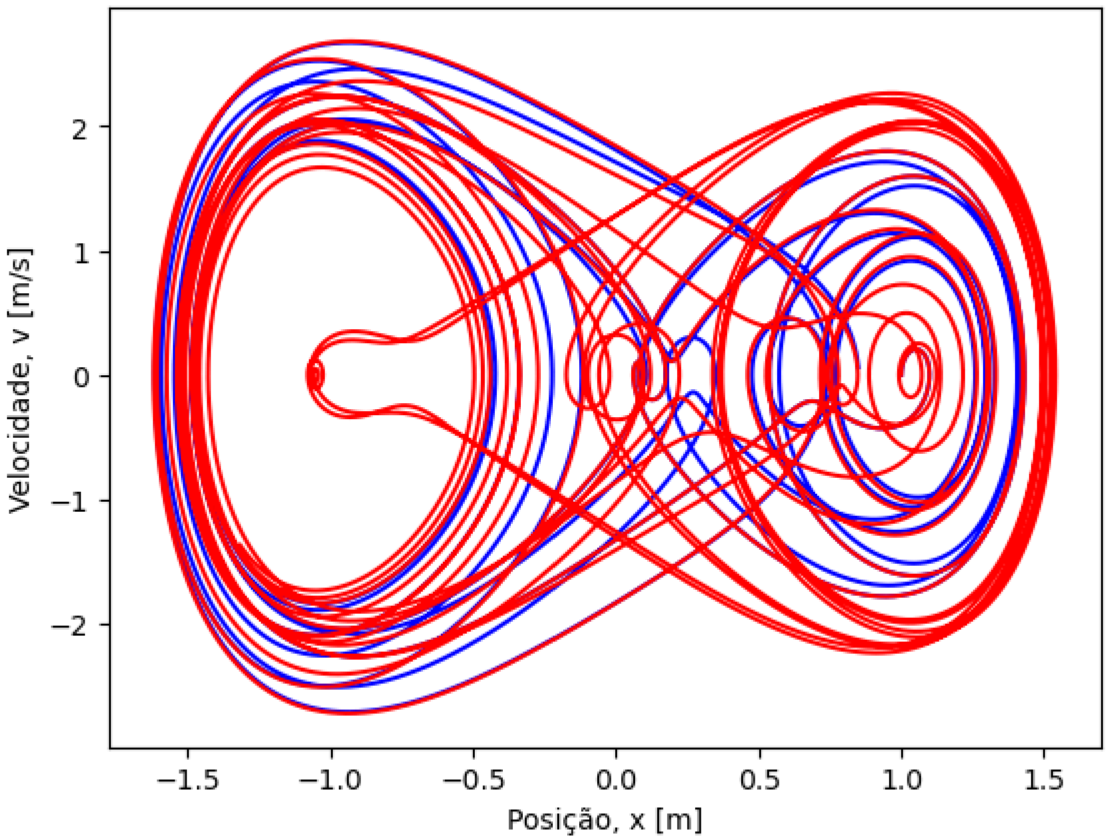
```python
x = np.zeros(np.size(t))          # posição [m]
x[0] = x0
v[0] = v0

# Não é necessário definir novamente as funções da aceleração e
# do método RK4

# percorrer o domínio temporal e resolver a equação
# de movimento usando o método RK4
for i in range(np.size(t) - 1):
    x[i + 1], v[i + 1] = rk4_x_v(t[i], x[i] ,v[i] ,a_res , dt)


plt.plot(t, x, 'r-')
plt.xlabel("Tempo decorrido, t [s]")
plt.ylabel("Posição, x [m]")
plt.show()

# Guardamos a velocidade e a posição para mais tarde
vB = v
xB = x
```



Podemos observar que o movimento é caótico, existe uma *estrutura* que não se repete exatamente ao longo do tempo.

d) Faça o gráfico das trajetórias no espaço de fase ($v(t)$ em função de $x(t)$). O que se observa?

In [10]:
```python
plt.plot(xA, vA, 'b-', xB, vB, 'r-')
plt.xlabel("Posição, x [m]")
plt.ylabel("Velocidade, v [m/s]")
plt.show()
```



Observações:

- As duas trajetórias iniciam com uma fase praticamente identica, mas divergem após alguns segundos;
- Uma pequena alteração na posição inicial induz alterações drásticas na equação de movimento;
- O corpo oscila em torno duas zonas ("atratores") localizadas em $x = \pm 1$;
- Na posição dos atratores ($x = \pm 1$) a velocidade é máxima;
- Longe dos atratores a velocidade é mínima (analogia com osciladores/pêndulos).

Pergunta 2:

Este sistema será sempre caótico, quaisquer que sejam os parâmetros?