Modelação de Sistemas Físicos - Aula Prática nº4

Método de Euler para resolução de equações diferenciais

O método de Euler é um procedimento de primeira ordem (o erro da solução é proporcional ao *passo*) para resolver equações diferenciais ordinarias a partir de condições iniciais (equações no tempo) ou fronteira (equações no espaço).

Suponhamos que temos a seguinte equação diferencial

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

O método de Euler começa pela discretização do domínio t em

$$t \longrightarrow \tau_0,\ \tau_1 = \tau_0 + \delta\tau,\ \tau_2 = \tau_0 + 2\delta\tau,\ \ldots$$

das funções

$$g \longrightarrow g_0 = g(\tau_0),\ g_1 = g(\tau_0 + \delta\tau),\ g_2 = g(\tau_0 + 2\delta\tau),\ \ldots$$

$$f \longrightarrow f_0 = f(\tau_0),\ f_1 = f(\tau_0 + \delta\tau),\ f_2 = g(\tau_0 + 2\delta\tau),\ \ldots$$

e da discretização da equação para

$$\left.\frac{\mathrm{d}f}{\mathrm{d}t}\right|_{t=\tau_i} \approx \left.\frac{\delta f}{\delta t}\right|_{t=\tau_i} = \frac{f(\tau_{i+1}) - f(\tau_i)}{\delta\tau} = \frac{f_{i+1} - f_i}{\delta\tau} \approx g_i.$$

em que $\delta\tau$ é o chamado **passo** da variável t.

No limite $\delta\tau \to 0$ (na prática $\delta\tau$ deve ser suficientemente pequeno) a expressão anterior converte-se numa igualdade, e podemos obter o valor de $f(\tau_{i+1}) = f_{i+1}$ a partir da solução no intervalo de tempo imediatamente anterior,

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

Note-se que o processo de solução é iterativo (a solução em τ_{i+1} depende da solução em τ_i). Assim, o início do procedimento requer o conhecimento de uma solução arbitraria em $f(\tau_i)$, referida como **condição inicial**, que pode (mas não tem que) ser f_0.

Note-se tambem que podemos *andar para trás* e obter a solução f_{i-1} a partir de f_i, nomeadamente,

$$f_{i-1} = f_i - g_{i-1}\delta\tau$$

Exemplo: Equação da velocidade de um corpo sujeito a uma aceleração $a(t)$

$$\frac{\mathrm{d}v}{\mathrm{d}t} = a(t)$$

discretizando a equação,

$$\frac{v(\tau_{i+1}) - v(\tau_i)}{\delta\tau} = a(\tau_i)$$

$$v_{i+1} = v_i + a_i\,\delta\tau$$

para o caso particular da aceleração ser constante, $a(t) \equiv a$, temos

$$v_{i+1} = v_i + a\,\delta\tau,$$

o que rapidamente se traduz para Python da seguinte forma

```
import numpy as np

t0 = 0.0                        # condição inicial, tempo [s]
v0 = 5.0                        # condição inicial, velocidade
[m/s]
dt = 0.001                      # passo [s]
tf = 10.0                       # limite do domínio, tempo
final [s]

a = 5.0                         # aceleração constant [m/s^2]

t = np.arange(t0, tf, dt)       # inicializar domínio [s]
v = np.empty(np.size(t))        # inicializar solução,
velocidade [m/s]

v[0] = v0
for i in range(np.size(t) - 1):
    v[i+1] = v[i] + a * dt
```

Pergunta 1: Como é que este código deve ser modificado para o caso de uma aceleração variável no tempo?

Problema 1

Um objeto pequeno é largado de uma altura elevada. Considere a queda livre, sem resistência do ar. Considere $g = 9.8\,\mathrm{m/s}^2$.

a) Qual a relação entre a velocidade e a aceleração instantânea?

A derivada da velocidade é iqual à aceleração instantânea:

$$\frac{\mathrm{d}v(t)}{\mathrm{d}t} = a(t)$$

No caso da queda livre de um objeto que parte do repouso, e assumindo a direção de queda como positiva, temos $a(t) = g$, em que g é a aceleração gravítica, ou seja,

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

em que

- $t_0 = 0\ \mathrm{s}$
- $v(t_0) = 0\ \mathrm{m/s}$

b) Construa um programa que determine a velocidade do objeto, usando o método de Euler, no intervalo de tempo $[0, 4\ \mathrm{s}]$. Qual a velocidade em $t = 3\ \mathrm{s}$?

In [1]:

```
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]
v0 = 0.0                        # condição inicial, velocidade [m/s]
dt = 0.1                        # passo [s]

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

t = np.arange(t0, tf, dt)       # inicializar domínio [s]
v = np.empty(np.size(t))        # inicializar solução, velocidade [m/s]

v[0] = v0
for i in range(np.size(t) - 1):
    v[i+1] = v[i] + g * dt

i3 = int((3 - t0) / dt)         # indice correspondente a t = 3 s
print("v(t=3) = ", v[i3], "m/s")
```

```
v(t=3) =  29.400000000000001 m/s
```

c) Repita a alínea anterior, com um passo 10 vezes menor.

In [2]:

```
dt = 0.01                       # passo [s]

t = np.arange(t0, tf, dt)       # inicializar domínio [s]
v = np.empty(np.size(t))        # inicializar solução, velocidade [m/s]

v[0] = v0
for i in range(np.size(t) - 1):
    v[i+1] = v[i] + g * dt

i3 = int((3 - t0) / dt)         # indice correspondente a t = 3 s
print("v(t = 3 s) = ", v[i3], "m/s")
```

```
v(t = 3 s) =  29.399999999999913 m/s
```

d) Compare o resultado obtido em b) e c) com o resultado exato. Que conclui?

O resultado exato é dado por

$$v(t) = v_0 + gt = 29.4\,\mathrm{m/s}$$

onde $g = 9.8\,\mathrm{m/s}^2$ e $v_0 = 0\,\mathrm{m/s}$.

A solução numérica é independent do passo porque v é linear em t!

In [3]:
```
0 + 9.8 * 3
```

Out[3]:
```
29.400000000000002
```

e) Construa um programa que determine a posição do objeto, usando o método de Euler, no intervalo de tempo $[0,\ 4s]$. Qual a posição no instante $t = 2\,\mathrm{s}$, se o objeto partiu da posição $x = 0\,\mathrm{m}$? Usa o passo de tempo usado em alínea b).

A velocidade instantânea do objeto é dada por

$$v(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}$$

que tem a seguinte forma discretizada,

$$x_{i+1} = x_i + v_i \delta\tau$$

em que v pode ser obtida como em b) e c).

In [4]:
```
t0 = 0.0                     # condição inicial, tempo [s]
tf = 4.0                     # limite do domínio, tempo final [s]
x0 = 0.0                     # condição inicial, posição [m]
v0 = 0.0                     # condição inicial, velocidade [m/s]
dt = 0.1                     # passo [s]

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

t = np.arange(t0, tf, dt)    # inicializar domínio [s]
x = np.empty(np.size(t))     # inicializar solução, posição [m]
v = np.empty(np.size(t))     # inicializar solução, velocidade [m/s]

x[0] = x0
v[0] = v0
for i in range(np.size(t) - 1):
    v[i+1] = v[i] + g * dt
    x[i+1] = x[i] + v[i] * dt

i2 = int((2 - t0) / dt)      # indice correspondente a t = 2 s
print("x(t = 2 s) = ", x[i2], "m")
```

```
x(t = 2 s) =  18.620000000000005 m
```

f) Repita a alínea anterior, com um passo 10 vezes menor.

In [5]:
```python
dt = 0.01                       # passo [s]

t = np.arange(t0, tf, dt)    # inicializar domínio [s]
x = np.empty(np.size(t))     # inicializar solução, posição [m]
v = np.empty(np.size(t))     # inicializar solução, velocidade [m/s]

x[0] = x0
v[0] = v0
for i in range(np.size(t) - 1):
    v[i+1] = v[i] + g * dt
    x[i+1] = x[i] + v[i] * dt

i2 = int((2 - t0) / dt)      # indice correspondente a t = 2 s
print("x(t = 2 s) = ", x[i2], "m")
```

```
x(t = 2 s) =  19.502000000000027 m
```

g) Compare o resultado obtido em e) e f) com o resultado exato. Que conclui?

O resultado exato é dado por

$$x(t) = \frac{1}{2}gt^2 = 19.6\,\mathrm{m/s}^2$$

onde $g = 9.8\,\mathrm{m/s}^2$ e $t = 2\,\mathrm{s}$.

A solução numérica depende do passo, e aproxima-se da solução exata à medida que diminuimos o passo!

In [6]:
```python
0.5 * 9.8 * 2**2
```

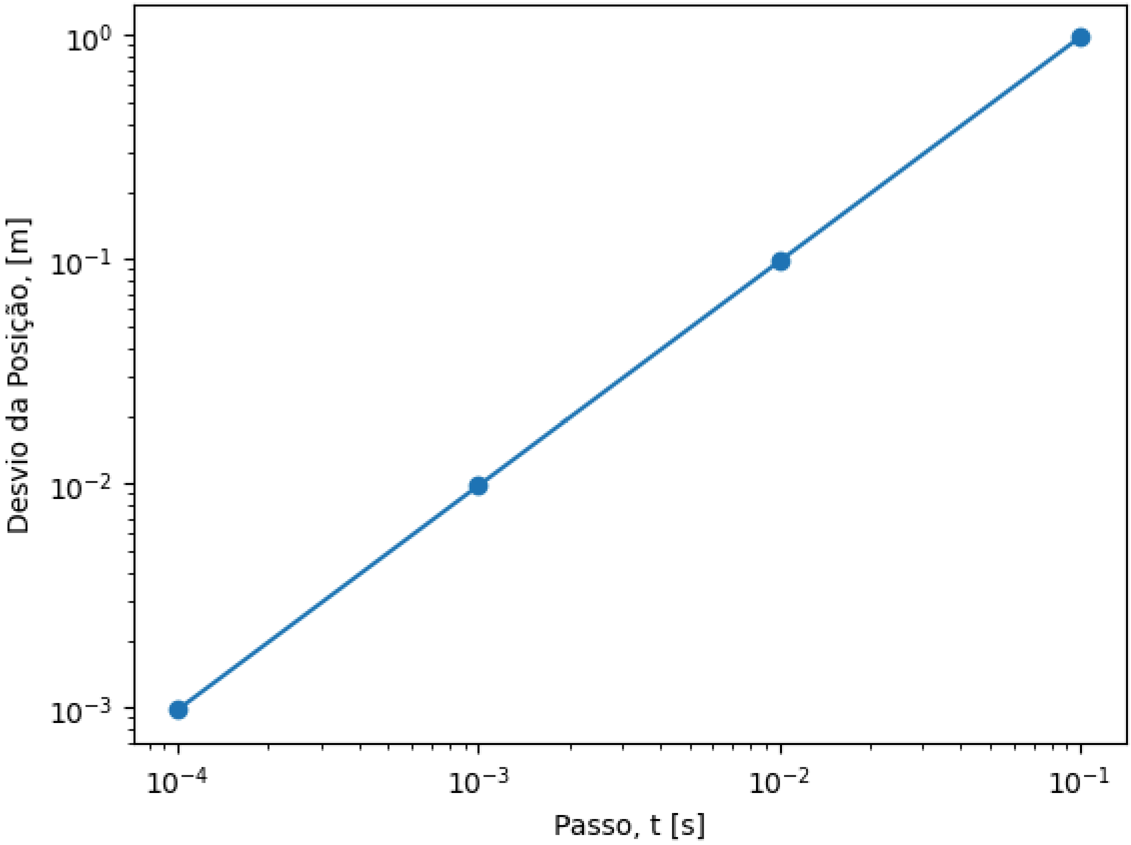
Out[6]:
```
19.6
```

h) Calcule novamente a posição no instante 2s, com o passo 10 vezes menor do que em alínea f). Faça o gráfico do desvio do valor aproximado com o valor exato em função do passo. Como varia o erro com o passo?

In [7]:
```python
dt = 0.001                      # passo [s]

t = np.arange(t0, tf, dt)    # inicializar domínio [s]
x = np.empty(np.size(t))     # inicializar solução, posição [m]
v = np.empty(np.size(t))     # inicializar solução, velocidade [m/s]
x_exata = 0.5 * g * t ** 2  # inicializar solução exata [m]

x[0] = x0
v[0] = v0
for i in range(np.size(t) - 1):
    v[i+1] = v[i] + g * dt
    x[i+1] = x[i] + v[i] * dt

i2 = int((2 - t0) / dt)      # indice correspondente a t = 2 s
print("x(t = 2 s) = ", x[i2], "m")
print("desvio = ", np.abs(x[i2] - x_exata[i2]), "m")
```

```
x(t = 2 s) =  19.590200000000138 m
desvio =  0.009799999999863473 m
```

In [8]:
```
passo =np.array([0.1, 0.01, 0.001, 0.0001])
desvio = np.array([0.98, 0.098, 0.0098, 0.00098])

# Representar desvio de x num grafico (usando o matplotlib)
plt.loglog(passo, desvio, 'o-')
plt.xlabel("Passo, t [s]")
plt.ylabel("Desvio da Posição, [m]")
plt.show()
```



O desvio varia linearmente com o passo.

Problema 2

Uma bola é lançada verticalmente para cima com a velocidade 10 m/s.

a) Encontre analiticamente a lei do movimento $y = y(t)$, se não considerar a resistência do ar.

$$y(t) = y_0 + v_0 t - \frac{1}{2} g t^2$$

Assumindo que a bola parte de $y_0 = 0\,\mathrm{m}$, ficamos com

$$y(t) = 10t - \frac{1}{2} g t^2 = t\left(10 - \frac{gt}{2}\right)$$

b) Qual a altura máxima e o instante em que ocorre, no caso da alínea a)?

A altura máxima ocorre quando a velocidade é nula, ou seja,

$$\frac{\mathrm{d}y}{\mathrm{d}t} = 10 - gt = 0$$

Assumindo $g = 9.8\,\mathrm{m/s}^2$ obtemos $t = 10/9.8 = 1.02040816\,\mathrm{s}$

A altura máxima ocorre a

$$y_{\mathrm{max}} = 10 \times \frac{10}{g} - \frac{1}{2} \times g \times \left(\frac{10}{g}\right)^2 = 5.1020408\,\mathrm{m}$$

In [9]:
```
10 * (10/9.8) - 1/2 * 9.8 * (10/9.8)**2
```

Out[9]: 5.10204081632653

c) Em que instante volta a passar pela posição inicial, no caso da alínea a)?

A bola vola a atingir a altura inicial quando $y(t) = 0$, ou seja,

$$y(t) = t\left(10 - \frac{gt}{2}\right) = 0$$

A condição anterior verifica-se quando $t = 0\,\mathrm{s}$ (obviamente), e também quando

$$10 - \frac{gt}{2} = 0 \Leftrightarrow t = \frac{20}{g} = 2.04081632\,\mathrm{s}$$

In [10]:
```
20/9.8
```

Out[10]: 2.0408163265306123

d) Resolva o problema da alínea a), considerando a resistência do ar. Resolva usando o método de Euler. A velocidade terminal da bola no ar é de 100 km/h.

Assumindo como positiva a direção de ascenção inicial da bola A, temos os seguintes regimes:

- A aceleração durante a ascenção é $-g - Dv^2$;
- A aceleração durante a descida é $-g + Dv^2$

Portanto, a equação da velocidade é,

$$\frac{\mathrm{d}v(t)}{\mathrm{d}t} = a(t) = -g - D\,v(t)\,|v(t)|$$

e a versão discretizada é data por,

$$v_{i+1} = v_i + a_i\,\delta\tau$$

em que

$a_i = -g - D\,v_i\,|v_i|$

e

$D = g/v_T^2$, e $v_T = 100\,\mathrm{km/h} = 100 * 1000\,\mathrm{m}/(3600\,\mathrm{s}) = 27.7(7)\,\mathrm{m/s}$.

A equação da posição é dada por

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = v(t)$$

e a versão discretizada é data por,

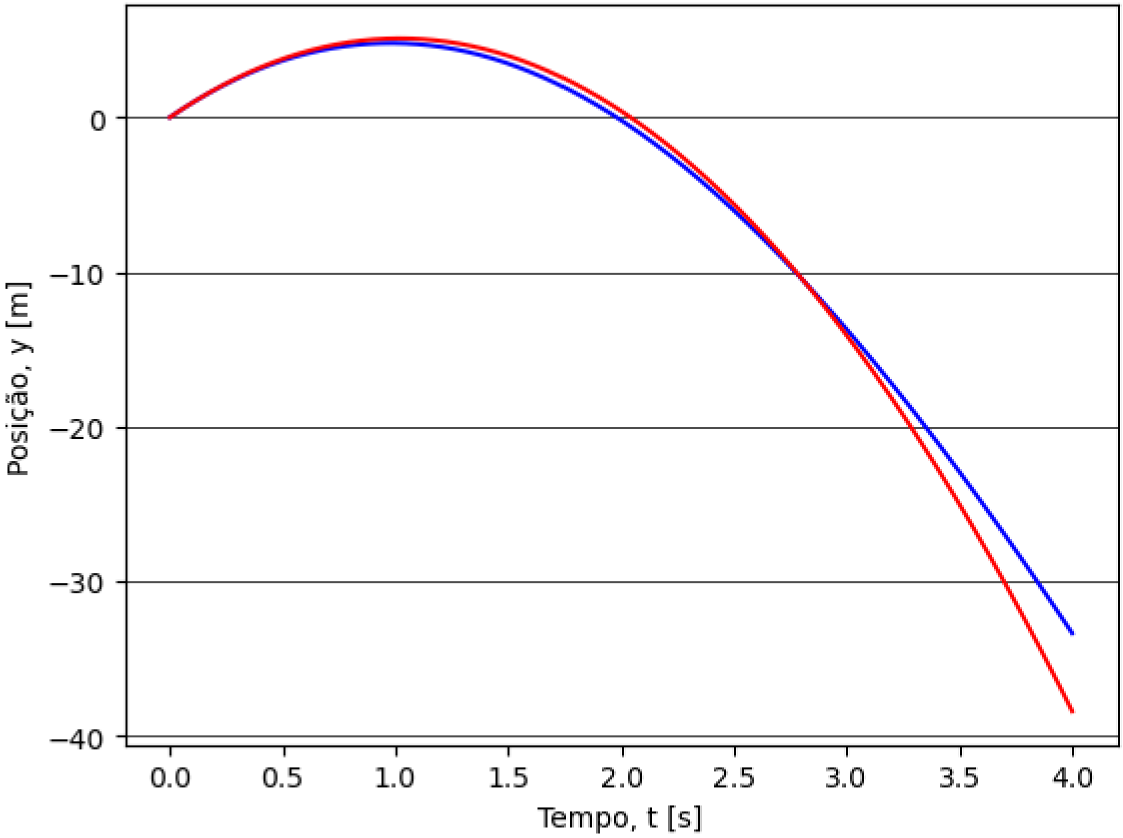
$$y_{i+1} = y_i + v_i\,\delta\tau$$

em que

- $t_0 = 0\,\mathrm{s}$
- $y(t=0) = 0\,\mathrm{m}$
- $v(t=0) = 10\,\mathrm{m/s}$

In [11]:
```
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]
y0 = 0.0                     # condição inicial, posição [m]
v0 = 10.0                    # condição inicial, velocidade [m/s]
vT = 100 * 1000 / 3600       # velocidade terminal [m/s]
dt = 0.0001                    # passo [s]

g = 9.8                      # aceleração gravítica [m/s^2]
D = g / vT ** 2              # parâmetro de resitência ao ar [m^-1]

t = np.arange(t0, tf, dt)    # inicializar domínio [s]
y = np.empty(np.size(t))     # inicializar solução, posição [m]
v = np.empty(np.size(t))     # inicializar solução, velocidade [m/s]
a = np.empty(np.size(t))     # inicializar solução, aceleração [m/s]

y[0] = y0
v[0] = v0

for i in range(np.size(t) - 1):
    a[i] = - g - D * v[i] * np.abs(v[i])
    v[i+1] = v[i] + a[i] * dt
    y[i+1] = y[i] + v[i] * dt

plt.plot(t, y, 'b-')
plt.plot(t, y0 + v0 * t - 0.5 * g * t**2, 'r-')
plt.grid(axis = "y", color = 'black', linewidth = 0.5)
plt.xlabel("Tempo, t [s]")
```

```python
plt.ylabel("Posição, y [m]")
plt.show()
```



e) Repita alíneas b) e c) nas condições de alínea d). Deve encontrara uma maneira numérica de estimar os instantes da altura máxima e do retorno ao posição inicial.

In [12]:

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

# indice e tempo para o qual volta a passar por y=0
izero = np.size(y) - np.size(y[y<0])
tzero = t[izero]
print("Tempo de rotorno à orígem, tzero = ", tzero, "s")
```

```
Tempo correspondente à altura máxima, tmax =  0.9795 s
Altura máxima, ymax =  4.797936730541468 m
Tempo de rotorno à orígem, tzero =  1.9792 s
```