Modelação de Sistemas Físicos - Aula Prática nº10

Realização e resolução de problemas sobre:

- Cap. 5: Potência

- Cap. 6: Osciladores

Problemas do Capítulo 5: Dinâmica de um ciclista

Uma ciclista com massa $m = 65\ \mathrm{kg}$ consegue produzir continuamente uma potência $P = 300\ \mathrm{W}$. Determine a evolução temporal da velocidade da ciclista, ao fim de $t = 100\ \mathrm{s}$, se partir com um empurrão que lhe confere uma velocidade inicial $v_0 = 1\ \mathrm{m/s}$?

Considere as seguintes condições:

- O coeficiente de resistência de um piso liso de alcatrão é de $\mu = 0.004$;
- O coeficiente de resistência do ar é $C_{\mathrm{res}} = 0.9\ \mathrm{s}$;
- A área frontal efetiva da ciclista é $A = 0.30\ \mathrm{m}^2$;
- A densidade do ar é $\rho_{\mathrm{ar}} = 1.225\ \mathrm{kg/m}^3$.

a) Qual a sua velocidade terminal?

b) Ao fim de quanto tempo atinge 90% da sua velocidade terminal?

Considere a seguinte descrição física do problema:

$$\begin{cases} F & = F_{\mathrm{cic}} - \frac{C_{\mathrm{res}}}{2} A \rho_{\mathrm{ar}} v^2 - \mu N \\ F_{\mathrm{cic}} & = P/v \\ N & = mg \end{cases}$$

em que F é a força resultante (ao longo da horizontal), F_{cic} é a força de propulsão imprimida pela ciclista, e N é a *força normal* que resulta do contacto com o solo.

Solução

Podemos usar o método de Euler para descrever a dinâmica do movimento da ciclista, para assim calcular a velocidade terminal, e estimar o instante de tempo t_{90} no qual $v(t = t_{90}) = 0.9 v_{\mathrm{T}}$.

A força resultante é dada por,

$$F = \frac{P}{v} - \frac{C_{\mathrm{res}}}{2} A \rho_{\mathrm{ar}} v^2 - \mu N,$$

ou seja, a aceleração instantânea é

$$a = \frac{P}{mv} - \frac{C_{\mathrm{res}}}{2m} A \rho_{\mathrm{ar}} v^2 - \mu g,$$

em que consideramos $P = 300\ \mathrm{W}$, $v(t=0) = 1\ \mathrm{m/s}$, $\mu = 0.004$, $C_{\mathrm{res}} = 0.9$, $A = 0.30\ \mathrm{m}^2$, $\rho_{\mathrm{ar}} = 1.225\ \mathrm{kg/m}^3$.

Assim, usando o método de Euler,

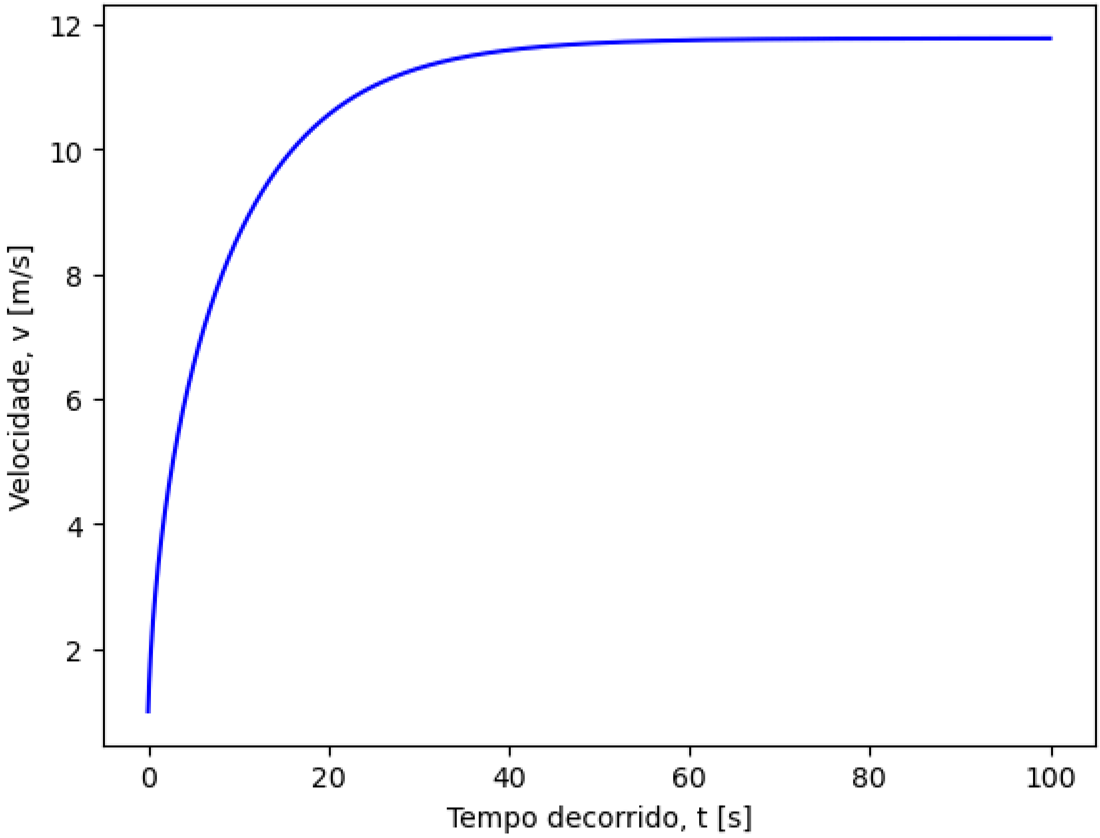
$$a_i = \frac{P}{m v_i} - \frac{C_{\mathrm{res}}}{2m} A \rho_{\mathrm{ar}} v_i^2 - \mu g$$

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

$$r_{i+1} = r_i + v_i\,\delta\tau$$

In [1]:

```
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 = 0.01                         # passo [s]

r0 = 0.0                          # condição inicial, posição inicial [m]
v0 = 1.0                          # condição inicial, velocidade inicial [m/s]

g = 9.8                           # aceleração gravítica [m/s^2]
P = 300.0                         # Potência de propulsão [W]
mu = 0.004                        # coeficiente de resistência do piso []
C_res = 0.9                       # coeficiente de resistência do ar [s]
rho_ar = 1.225                    # densidade do ar [kg/m3]
A = 0.3                           # área frontal efetiva da ciclista [m2]
m = 65.0                          # massa da ciclista (+ bicicleta) [kg]

# 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]
r = np.zeros(np.size(t))          # posição [m]
r[0] = r0
v[0] = v0

# método de Euler
for i in range(np.size(t) - 1):
    a[i] = P / (m * v[i]) - C_res * A * rho_ar * v[i] ** 2 / (2 * m) - mu * g
    v[i + 1] = v[i] + a[i] * dt
    r[i + 1] = r[i] + v[i] * dt

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



a) A velocidade terminal da ciclista é dada por,

In [2]:
```
v_T = v[-1]
print("Velocidade terminal, v_T = {0:.4f} m/s".format(v_T))
```

```
Velocidade terminal, v_T = 11.7744 m/s
```

b) Ao fim de quanto tempo atinge 90% da sua velocidade terminal?

Podemos estimar o instante de tempo t_{90} filtrando os elementos do array `v[t0,...,tN-1]` que são inferiores a $0.9 v_{\mathrm{T}}$,

In [3]:
```
v_filt = v[v < 0.9 * v_T]
i90 = np.size(v_filt)
t90 = t[i90]
print("Instante de tempo no qual a velocidade é 90% de v_T, t90 = {0:.2f} s".format(t
print("A velocidade correspondente a 90% de v_T é {0:.2f} s".format(0.9 * v_T))
print("A velocidade v(t90) é {0:.2f} m/s".format(v[i90]))
```

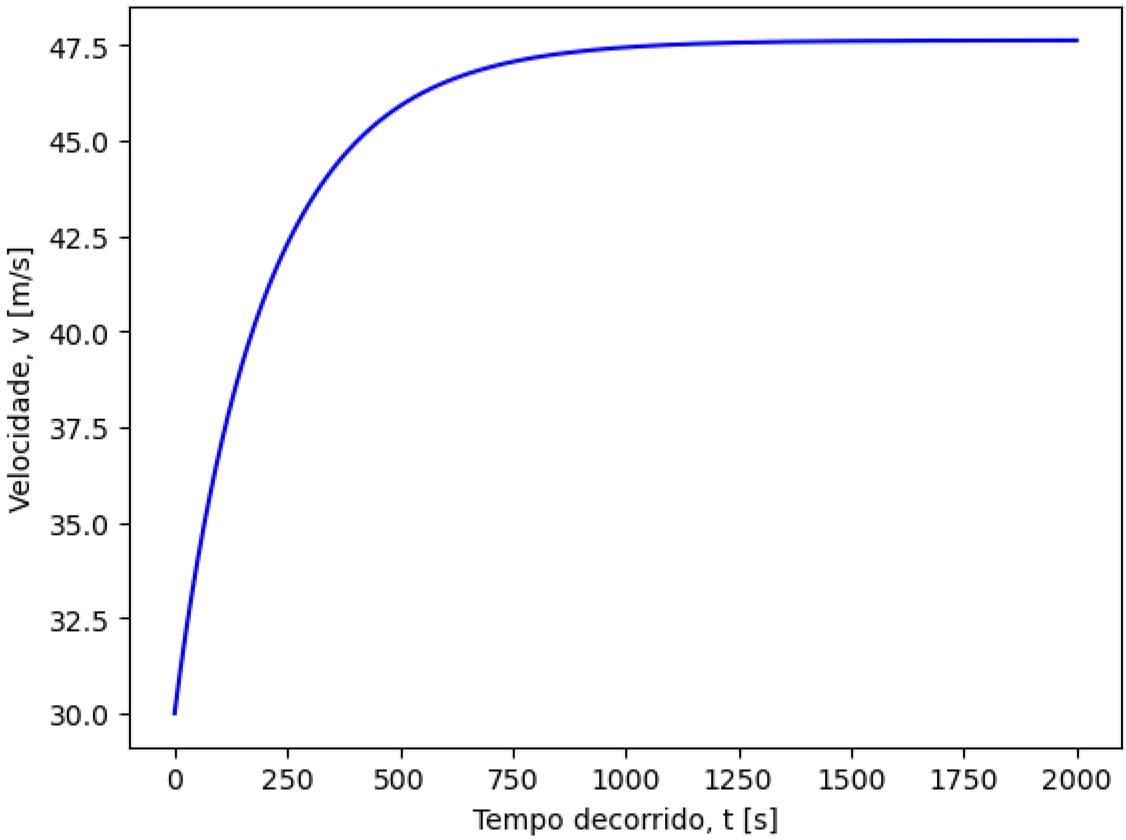
```
Instante de tempo no qual a velocidade é 90% de v_T, t90 = 20.42 s
A velocidade correspondente a 90% de v_T é 10.60 s
A velocidade v(t90) é 10.60 m/s
```

A ciclista do problema anterior segue logo atrás de um carro com carenagem aerodinâmica, tal que a força de resistência do ar experimentada pelo ciclista é reduzida em 99%.

Considerando uma velocidade inicial de 30 m/s, calcule a velocidade terminal nas novas condições.

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt

t0 = 0.0                          # condição inicial, tempo [s]
tf = 2000.0                        # limite do domínio, tempo final [s]
dt = 0.01                         # passo [s]

r0 = 0.0                          # condição inicial, posição inicial [m]
```

```
# Aqui consideramos a nova velocidade inicial
v0 = 30.0                          # condição inicial, velocidade inicial [m/s]

g = 9.8                            # aceleração gravítica [m/s^2]
P = 300.0                          # Potência de propulsão [W]
mu = 0.004                         # coeficiente de resistência do piso []
C_res = 0.9                        # coeficiente de resistência do ar [s]
rho_ar = 1.225                     # densidade do ar [kg/m3]
A = 0.3                            # área frontal efetiva da ciclista [m2]
m = 65.0                           # massa da ciclista (+ bicicleta) [kg]

# 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]
r = np.zeros(np.size(t))           # posição [m]
r[0] = r0
v[0] = v0

# método de Euler
for i in range(np.size(t) - 1):
    # aqui reduzimos a contribuição da resistência do ar
    # para aceleração a 1% do valor original
    a[i] = P / (m * v[i]) - 0.01 * C_res * A * rho_ar * v[i] ** 2 / (2 * m) - mu * g
    v[i + 1] = v[i] + a[i] * dt
    r[i + 1] = r[i] + v[i] * dt

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

v_T = v[-1]
print("Velocidade terminal, v_T = {0:.4f} m/s".format(v_T))
```




```
Velocidade terminal, v_T = 47.6238 m/s
```

Problema Capítulo 6 Movimento de um pêndulo simples

Uma massa suspensa do teto por um fio de comprimento $L = 1\,\mathrm{m}$ oscila à volta da sua posição de equilíbrio, expressa por $\theta = 0\,\mathrm{rad}$, de acordo com a equação diferencial,

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -\frac{g}{L}\sin\theta$$

onde $g = 9.8\,\mathrm{m/s}^2$ é a aceleração gravítica.

Pendulo simples

a) Simule o movimento do pêndulo usando o método de Euler-Cromer durante 10 s. A massa inicia com ângulo $\theta = 10^\circ$ e com velocidade nula.

b) Para ângulos de oscilação pequenos, a equação diferencial pode ser approximada a

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -\frac{g}{L}\theta,$$

cuja solução analítica é

$$\theta(t) = A\cos\left(\sqrt{\frac{g}{L}}t + \phi\right).$$

Compare esta solução com a solução numérica obtida na alínea a).

c) Repita a comparação com ângulos iniciais de 20° e 30°. As soluções numéricas e teóricas são diferentes?

Solução da alínea a)

Este problema é resolvido em coordenadas angulares, i.e., usando:

- a aceleração angular, $\gamma = \mathrm{d}^2\theta/\mathrm{d}t^2 = \mathrm{d}\omega/\mathrm{d}t$;
- a velocidade angular, $\omega = \mathrm{d}\theta/\mathrm{d}t$;
- o ângulo (posição angular), θ

Sabendo que a aceleração instantânea é dada por

$$\gamma = -\frac{g}{L}\sin\theta.$$

O método de Euler-Cromer em coordenadas angulares é dado por,

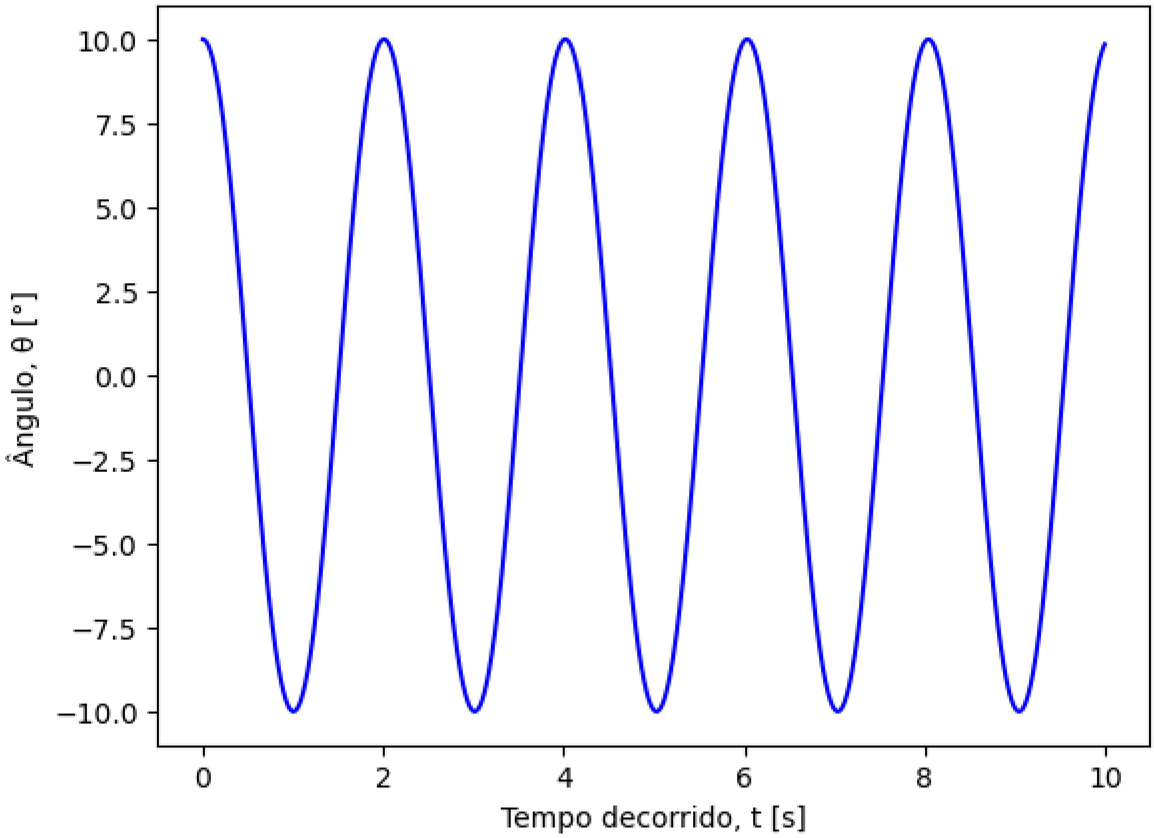
$$\gamma_i = -g\,\sin\theta_i/L$$

$$\omega_{i+1} = \omega_i + \gamma_i\,\delta\tau$$

$$\theta_{i+1} = \theta_i + \omega_{i+1}\,\delta\tau$$

In [5]:
```
import numpy as np
import matplotlib.pyplot as plt

t0 = 0.0                            # condição inicial, tempo [s]
```

```
tf = 10.0                       # limite do domínio, tempo final [s]
dt = 0.001                      # passo [s]

θ0 = 10.0 * np.pi / 180         # condição inicial, ângulo inicial [rad]
ω0 = 0.0                        # condição inicial, velocidade angular inicial [rad

L = 1.0                         # comprimento do fio [m]

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

# inicializar solução
γ = np.zeros(np.size(t))        # aceleração angular [rad/s2]
ω = np.zeros(np.size(t))        # velocidade angular [rad/s]
θ = np.zeros(np.size(t))        # ângulo [rad]
θ[0] = θ0
ω[0] = ω0

# método de Euler
for i in range(np.size(t) - 1):
    γ[i] = - g / L * np.sin(θ[i])
    ω[i + 1] = ω[i] + γ[i] * dt
    θ[i + 1] = θ[i] + ω[i + 1] * dt

plt.plot(t, θ * 180 / np.pi, 'b-')
plt.xlabel("Tempo decorrido, t [s]")
plt.ylabel("Ângulo, θ [°]")
plt.show()
```



Podemos verificar que usando a aproximação para o período de oscilação $T = 2\pi\sqrt{L/g}$, obtemos um valor consistente com a figura,

In [6]:
```
T = 2 * np.pi * np.sqrt(L / g)
print("Período de oscilação, T = {0:.2f} s".format(T))
```

```
Período de oscilação, T = 2.01 s
```

Solução da alínea b)

Comparemos então a solução analítica, válida apenas para amplitudes de oscilação pequenos,

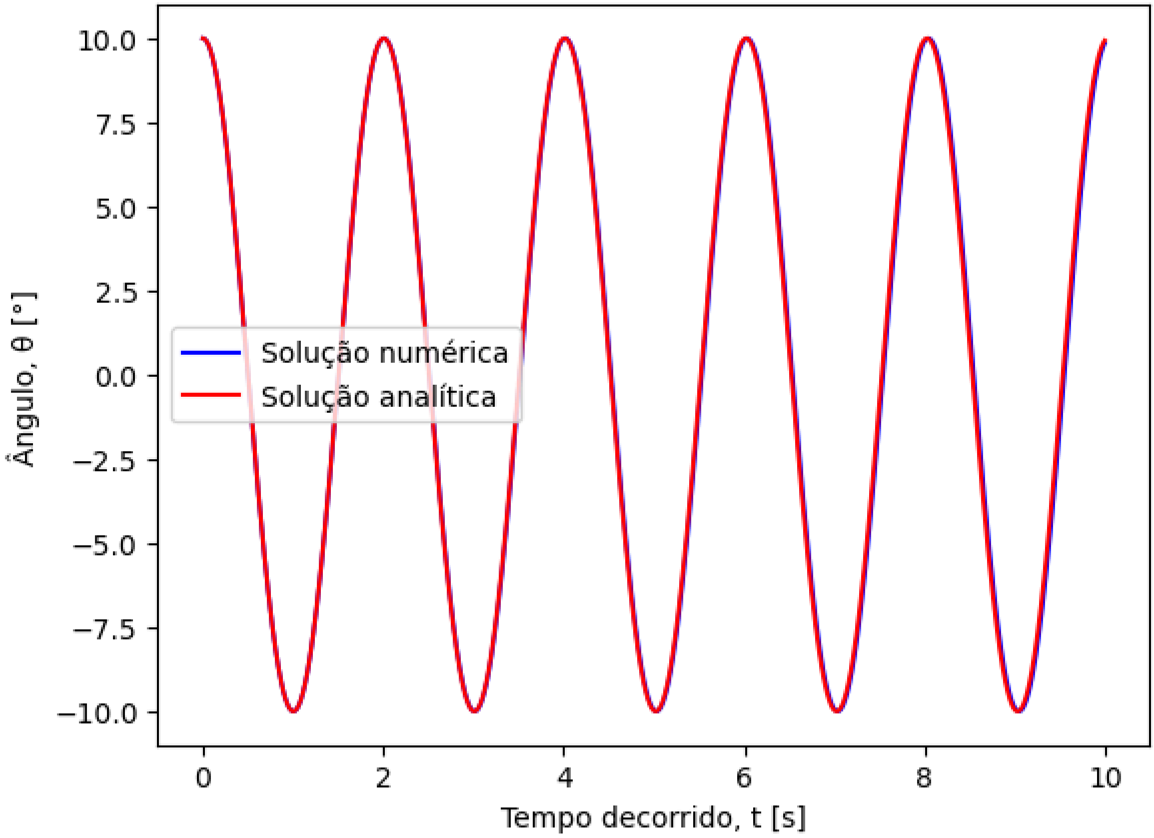
$$\theta_{\mathrm{a}}(t) = A\cos\left(\sqrt{\frac{g}{L}}t + \phi\right),$$

com a solução numérica $\theta(t)$ obtida na alínea anterior.

Notemos que:

- A constante de fase $\phi = 0$. Só assim, o pendulo inicia o movimento com amplitude máxima e velocidade nula.
- A amplitude $A = 10^\circ = 10\,\pi/180\,\mathrm{rad}$.

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

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

θ0 = 10.0 * np.pi / 180           # condição inicial, ângulo inicial [rad]
ω0 = 0.0                          # condição inicial, velocidade angular inicial [rad

L = 1.0                           # comprimento do fio [m]

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

# inicializar solução
γ = np.zeros(np.size(t))          # aceleração angular [rad/s2]
ω = np.zeros(np.size(t))          # velocidade angular [rad/s]
θ = np.zeros(np.size(t))          # ângulo [rad]
θ[0] = θ0
ω[0] = ω0

# método de Euler
for i in range(np.size(t) - 1):
    γ[i] = - g / L * np.sin(θ[i])
    ω[i + 1] = ω[i] + γ[i] * dt
    θ[i + 1] = θ[i] + ω[i + 1] * dt

# Solução analítica
A = 10 * np.pi / 180                      # amplitude [rad]
φ = 0.0                                   # constante de fase [rad]
θ_a = A * np.cos(np.sqrt(g / L) * t + φ)  # equação do movimento [rad]

plt.plot(t, θ * 180 / np.pi, 'b-', t, θ_a * 180 / np.pi, 'r-')
plt.xlabel("Tempo decorrido, t [s]")
plt.ylabel("Ângulo, θ [°]")
plt.legend(['Solução numérica', 'Solução analítica'])
plt.show()
```

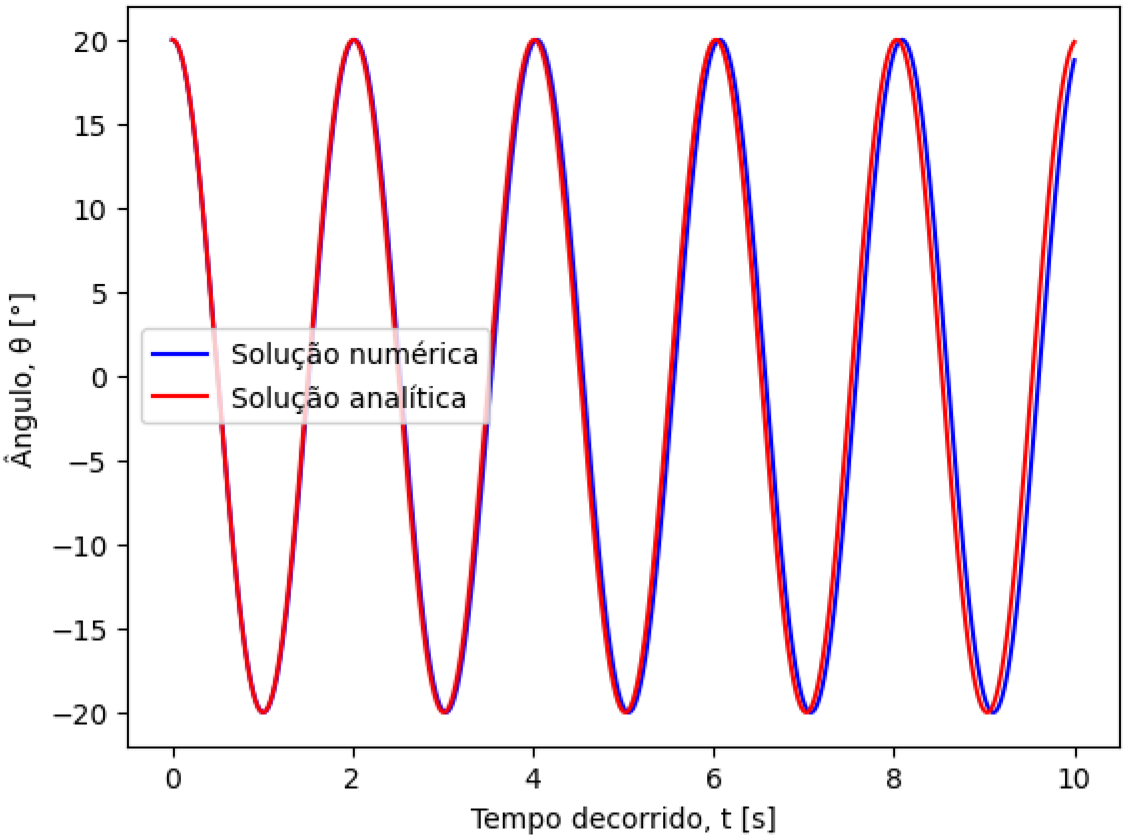


Se aumentarmos a amplitude para $A = 20^\circ = 20\,\pi/180\,\mathrm{rad}$,

In [8]:

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

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

θ0 = 20.0 * np.pi / 180             # condição inicial, ângulo inicial [rad]
ω0 = 0.0                            # condição inicial, velocidade angular inicial [rad

L = 1.0                             # comprimento do fio [m]

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

# inicializar solução
γ = np.zeros(np.size(t))            # aceleração angular [rad/s2]
ω = np.zeros(np.size(t))            # velocidade angular [rad/s]
θ = np.zeros(np.size(t))            # ângulo [rad]
θ[0] = θ0
ω[0] = ω0

# método de Euler
for i in range(np.size(t) - 1):
    γ[i] = - g / L * np.sin(θ[i])
    ω[i + 1] = ω[i] + γ[i] * dt
    θ[i + 1] = θ[i] + ω[i + 1] * dt

# Solução analítica
A = 20 * np.pi / 180                        # amplitude [rad]
φ = 0.0                                     # constante de fase [rad]
θ_a = A * np.cos(np.sqrt(g / L) * t + φ)    # equação do movimento [rad]

plt.plot(t, θ * 180 / np.pi, 'b-', t, θ_a * 180 / np.pi, 'r-')
plt.xlabel("Tempo decorrido, t [s]")
plt.ylabel("Ângulo, θ [°]")
```

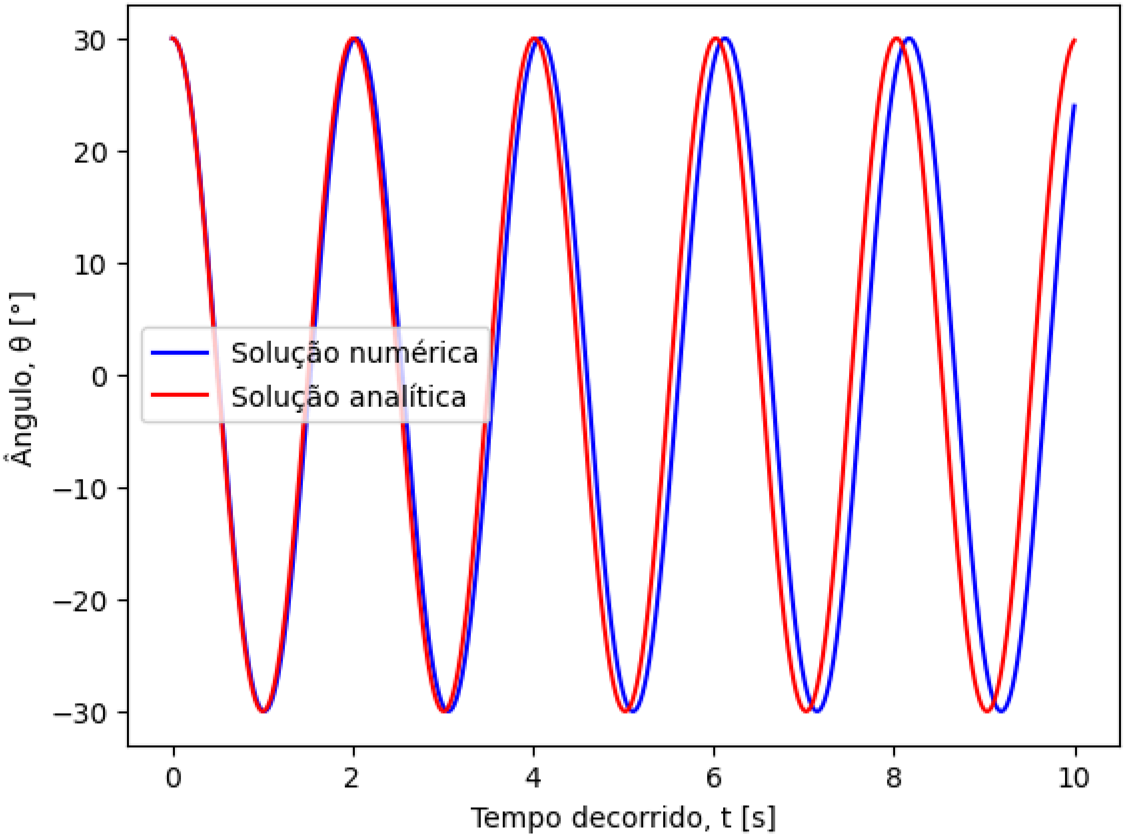
```
plt.legend(['Solução numérica', 'Solução analítica'])
plt.show()
```



E finalmente, aumentando a amplitude para $A = 30^\circ = 30\,\pi/180\,\mathrm{rad}$,

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

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

θ0 = 30.0 * np.pi / 180           # condição inicial, ângulo inicial [rad]
ω0 = 0.0                          # condição inicial, velocidade angular inicial [rad

L = 1.0                           # comprimento do fio [m]

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

# inicializar solução
γ = np.zeros(np.size(t))          # aceleração angular [rad/s2]
ω = np.zeros(np.size(t))          # velocidade angular [rad/s]
θ = np.zeros(np.size(t))          # ângulo [rad]
θ[0] = θ0
ω[0] = ω0

# método de Euler
for i in range(np.size(t) - 1):
    γ[i] = - g / L * np.sin(θ[i])
    ω[i + 1] = ω[i] + γ[i] * dt
    θ[i + 1] = θ[i] + ω[i + 1] * dt

# Solução analítica
A = 30 * np.pi / 180                      # amplitude [rad]
φ = 0.0                                   # constante de fase [rad]
θ_a = A * np.cos(np.sqrt(g / L) * t + φ)  # equação do movimento [rad]

plt.plot(t, θ * 180 / np.pi, 'b-', t, θ_a * 180 / np.pi, 'r-')
```

```
plt.xlabel("Tempo decorrido, t [s]")
plt.ylabel("Ângulo, θ [°]")
plt.legend(['Solução numérica', 'Solução analítica'])
plt.show()
```



Conclusão:

Verificamos que de facto a solução analítica

$$\theta_{\mathrm{a}}(t) = A\cos\left(\sqrt{\frac{g}{L}}t + \phi\right),$$

é uma solução aproximada da equação diferencial

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -\frac{g}{L}\sin\theta.$$

Esta última pode ser resolvida com uma precisão arbitrária usando o método de Euler-Cromer.

In [ ]: