Modelação de Sistemas Físicos - Aula Prática nº3

Exercício 1. Familiarização com cálculo symbólico com `sympy`

Importar a biblioteca `sympy`

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

Definir as variáveis simbólicas x, y, m, b

In [2]:
```
x, y, m, b = sy.symbols('x, y, m, b')
```

Definir a expressão $y = mx^2 + b$, em que $y \equiv y(m, x, b)$ é uma função de 3 variáveis independentes:

In [3]:
```
y = m * x**2 + b
```

Impôr valores específicos para as variáveis independentes m e b, definindo uma função $y_2 \equiv y_2(x)$ que depende unicamente de x

In [4]:
```
y2 = y.subs([(m, 0.01), (b, 0.0)])
```

Podemos avaliar y_2 num determinado valor de x com o método `evalf` e com recurso a um dicionário `{x:valor}`:

In [5]:
```
y_em_1 = y2.evalf(subs={x:1})
```

Se desejarmos avaliar múltiplas vezes, por exemplo elemento-a-elemento, usando uma função lambda `numpy`,

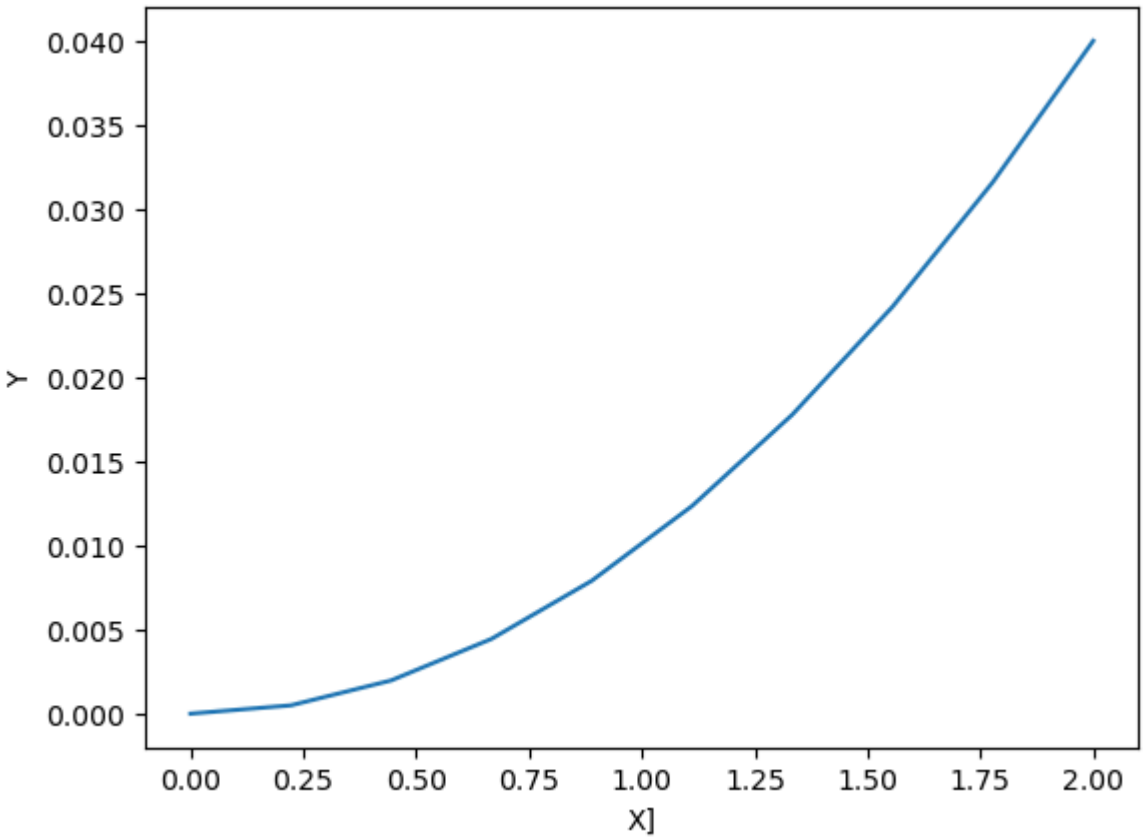
In [6]:
```
y_lam = sy.lambdify(x, y2, "numpy")
```

Agora `y_lam` é uma função de `x` que pode ser avaliada usando `y_lam(x)`.

Use a função `y_lam` para fazer um plot de `y` em valores de `x` de 0 até 2.

In [7]:
```
# Gerar uma sequência de 10 valores entre 0.0 e 2.0
X = np.linspace(0.0, 2.0, 10)
Y = y_lam(X)

# Representação gráfica
plt.plot(X, Y)
plt.xlabel("X]")
plt.ylabel("Y")
plt.show()
```



Outras instruções úteis:

Métodos	Descrição
`sympy.diff(y,x)`	Derivada de y em função de x
`sympy.integrate(y, x)`	Integral de y em função de x
`sympy.nsolve(y,x,x0)`	Encontra solução x que resolve a equação $y = 0$. x_0 é um valor inicial aproximado de x.

Exercício 2. Aceleração constante

Um avião arranca apartir do repouso e acelera com aceleração constante $a = 3\ \mathrm{m/s^2}$ até atingir a velocidade de descolagem de $v_d = 250\ \mathrm{km/h}$.

1. Escreva a função que descreve o movimento do avião (lei de movimento) $x(t)$. Faça o gráfico da lei do movimento.

A equação do movimento de um corpo uniformemente acelerado é:

$$x(t) = x(0) + v(0)t + \frac{1}{2}at^2$$

Dado que no presente caso temos:

- $x(0) = 0\ \mathrm{m}$, assumimos que a orígem das coordenadas está na posição inicial;
- $v(0) = 0\ \mathrm{m/s}$, o avião parte do repouso,

ficamos com:

$$x(t) = \frac{1}{2}at^2$$

em que $a = 3\ \mathrm{m/s^2}$.

Vamos escrever a equação de movimento em Python de uma forma analítica, definindo a variável independente t, assim como a variável dependente x. A aceleração é tida como constante.

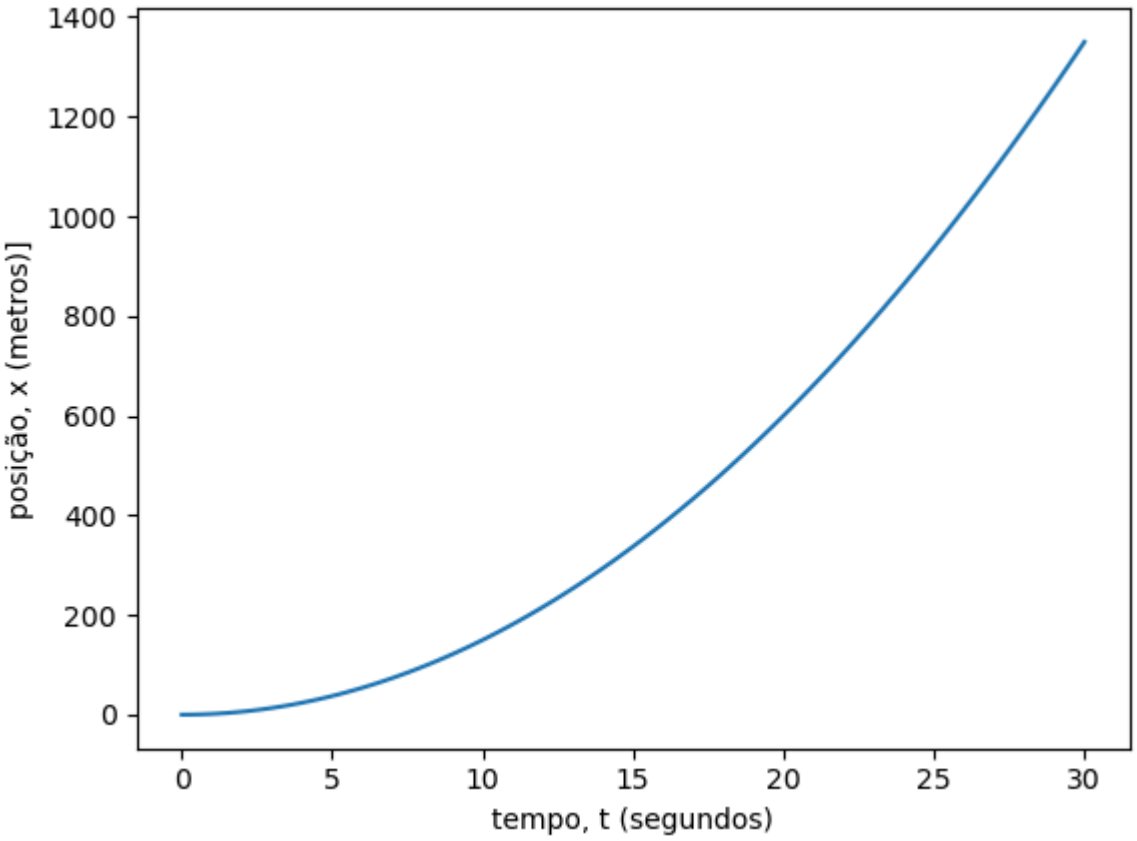
In [8]:
```
import sympy as sy

# Definir equação de movimento
a = 3
x, t = sy.symbols('x, t')
x = 1/2 * a * t**2

# "Lambdificar" a função x(t)
x_lam = sy.lambdify(t, x, "numpy")

# Gerar uma sequência de 100 valores entre 0.0 e 30 segundos
T = np.linspace(0.0, 30.0, 100)
X = x_lam(T)

# Representação gráfica da posição em função do tempo
plt.plot(T, X)
plt.ylabel("posição, x (metros)]")
plt.xlabel("tempo, t (segundos)")
plt.show()
```



2. Em que instante e qual a distância percorrida pelo avião quando atinge a velocidade de descolágem? Encontre a solução usando o `sympy`.

O instante pretendido, t_{desc}, corresponde ao valor de t para o qual a velocidade $v_{\mathrm{desc}} = 250\ \mathrm{km/h} = 69.4(4)\ \mathrm{m/s}$. A velocidade do avião é dada por:

$$v(t) = \frac{\mathrm{d}x}{\mathrm{d}t} = v(0) + at = at.$$

Portanto, basta-nos resolver a equação:

$$t_{\mathrm{desc}} = \frac{v(t_{\mathrm{desc}}) - v(0)}{a}$$

para encontrar a solução $t_{\mathrm{desc}} = 69.4(4)/3 = 23.148(148)\ \mathrm{s}$.

3. Use `sympy.integrate()` para integrar a aceleração em função de tempo duas vezes, para obter a velocidade e a posição do avião como funções (simbólicas) no tempo. O resultado está de acordo com o que se esperava?

In [9]:
```
# A aceleração vai ser integrada e temos que a definir como um símbolo.
a, v, x, t = sy.symbols('a, v, x, t')

# Definir função aceleração
a = 3

# Primeira integração para obter a velocidade
v = sy.integrate(a, t)

# Primeira integração para obter a posição
x = sy.integrate(v, t)
print("x(t) = ", x)
```

```
x(t) =  3*t**2/2
```

Sim, a solução corresponde à solução analítica prevista: $x(t) = 1/2\, a\, t^2$, em que $a = 3\ \mathrm{m/s^2}$

4. Use `sympy.nsolve()` para encontrar o tempo e a posição de descolagem. Compare com a solução encontrada em

2.?

In [10]:
```
# Definir a equação da velocidade
v = a * t

# Resolver a equacao v(t) - 250x1000/3600 = 0 e encontrar t.
# Assumi t = 30 s como valor aproximado para a solução.
t_desc = sy.nsolve(v - 250*1000/3600, t, 30)
print("t_desc = {0:.4f} s".format(t_desc))

# Calcular a posição de descolágem
x_desc = x.subs([(t, t_desc)])
print("x_desc = {0:.4f} s".format(x_desc))
```

```
t_desc = 23.1481 s
x_desc = 803.7551 s
```

Pergunta 1:

Descreve uma situação em que faz sentido usar cálculo simbólico em vez de obter soluções numéricas?

Exercício 3. Volante de badmington (aceleração uma função de velocidade)

Um volante de badmington foi largado de uma altura considerável (suficientemente elevada para podermos assumir que atinge um regime estacionário com velocidade terminal constante). A lei do movimento é,

$$y(t) = \frac{v_{\mathrm{T}}^2}{g} \log\left[\cosh\left(\frac{gt}{v_{\mathrm{T}}}\right)\right]$$

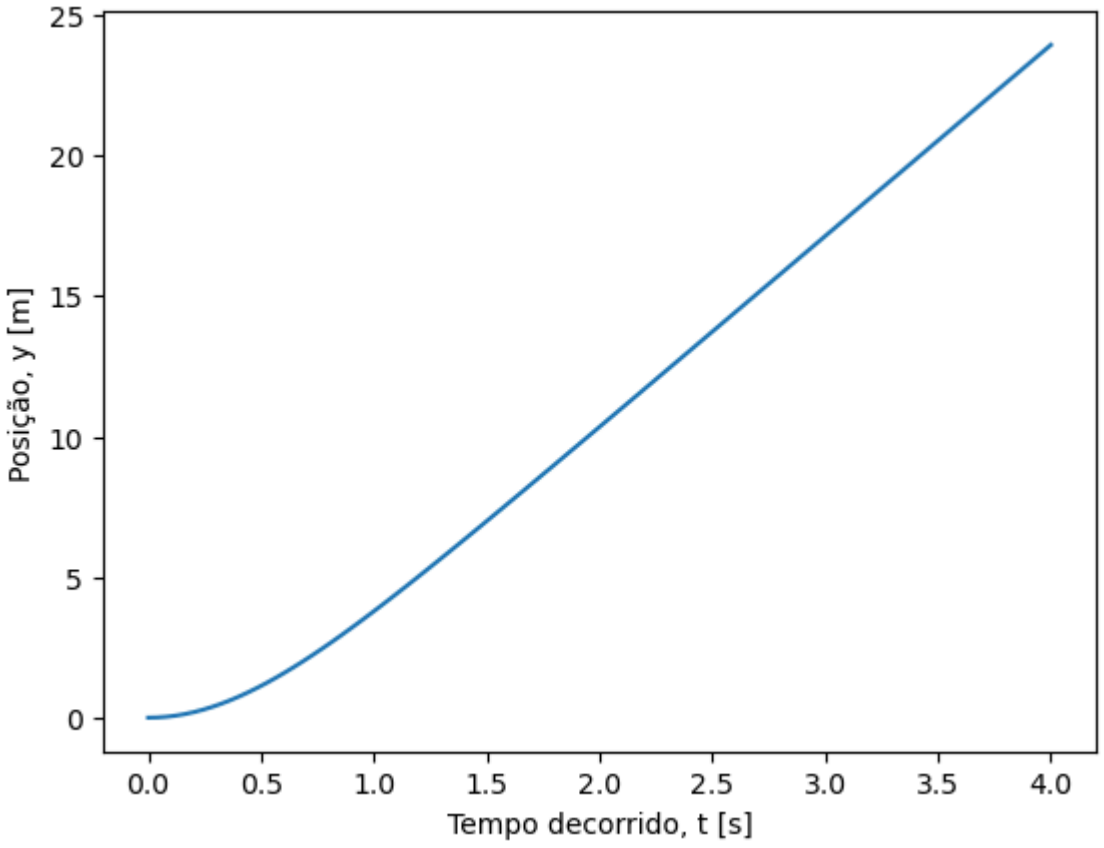
em que a velocidade terminal do volante é $v_{\mathrm{T}} = 6.80$ m/s.

1. Faça o gráfico da lei do movimento $y(t)$ de 0 a 4.0 s.

In [11]:
```
t = np.linspace(0.0, 4.0, 100)    # definir dominio e resolução temporal [s]
v_T = 6.80                        # velocidade terminal do volante [m/s]
g = 9.80665                       # aceleração gravitacional (valor standard) [m/s^2]

# posição do volante [m]
y = (v_T ** 2 / g) * np.log(np.cosh(g * t / v_T))

# Representar dados num grafico (usando o matplotlib)
plt.plot(t, y)
plt.xlabel("Tempo decorrido, t [s]")
plt.ylabel("Posição, y [m]")
plt.show()
```



2. Determine a velocidade instantânea em função do tempo, **usando cálculo simbólico** com o `sympy`. Faça o gráfico da velocidade em função do tempo de 0 a 4 s, usando o pacote

matplotlib.

In [12]:
```
# Definir symbolos
import sympy as sy
y, t, v_T, g = sy.symbols('y, t, v_T, g')

# Definir função posição do volante
y = (v_T ** 2 / g) * sy.log(sy.cosh(g * t / v_T))
```

Sabendo que a velocidade é dada por $v(t) = \mathrm{d}x(t)/\mathrm{d}t$, podemos usar a função `sympy.diff()`

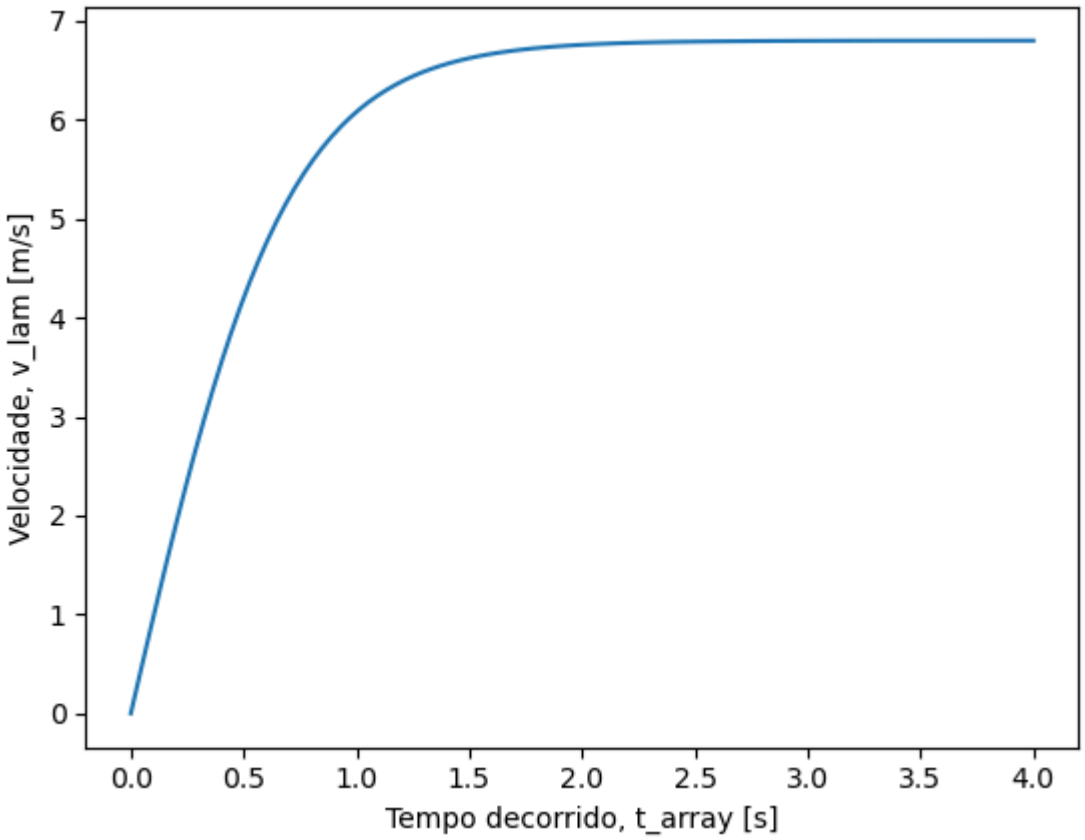
In [13]:
```
# Calcular velocidade instantânea
v = sy.diff(y, t)
print("v(t) = ", v)
```

```
v(t) =  v_T*sinh(g*t/v_T)/cosh(g*t/v_T)
```

Para calcular o valor de uma expressão simbólica de forma recorrente, usamos o coversor `lambdify`, especificando que o argumento é um array `numpy`,

In [14]:
```
# Converter expressão da velocidade em função de um array numpy
# Nota 1: Só a variável "t" é independente. Temos que especificar todos os restantes
# Nota 2: v_lam é um array numpy se o argumento "t" for um array numpy
v_lam = sy.lambdify(t, v.subs({v_T: 6.80, g: 9.80665}), "numpy")

# Representar dados num grafico (usando o matplotlib)
t_array = np.linspace(0.0, 4.0, 100)
plt.plot(t_array, v_lam(t_array))
plt.xlabel("Tempo decorrido, t_array [s]")
plt.ylabel("Velocidade, v_lam [m/s]")
plt.show()
```



3. Determine a aceleração instantânea em função do tempo, **usando cálculo simbólico** com o `sympy`. Faça o gráfico da aceleração em função do tempo de 0 a 4 s, usando o pacote

`matplotlib`.

Sabendo que a aceleração é dada por $a(t) = \mathrm{d}v(t)/\mathrm{d}t$, podemos usar novamente a função `sympy.diff()`

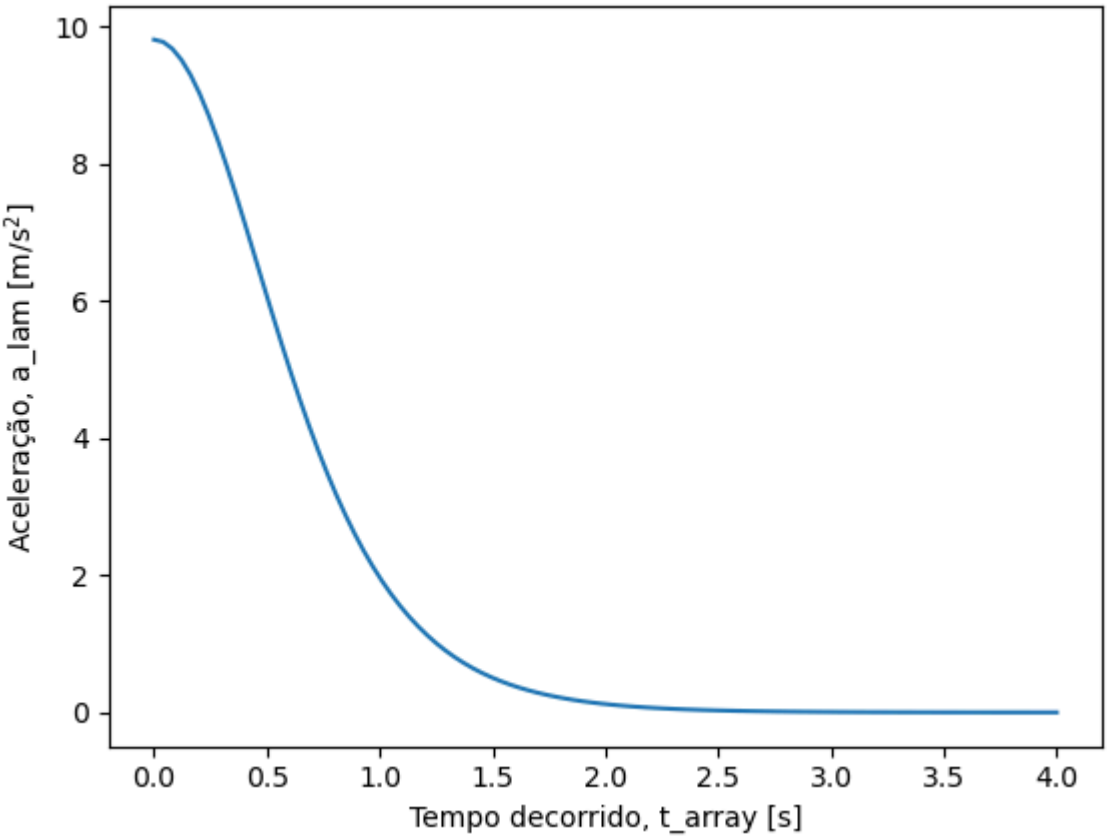
In [15]:
```
# Calcular velocidade instantânea
a = sy.diff(v, t)
print("a(t) = ", a)
```

```
a(t) =  -g*sinh(g*t/v_T)**2/cosh(g*t/v_T)**2 + g
```

Usamos novamente o coversor `lambdify`, para obter os valores da aceleração num array `numpy`.

In [16]:
```
# Converter expressão da aceleração em função de um array numpy
# Nota 1: Só a variável "t" é independente. Temos que especificar todos os restantes
# Nota 2: a_lam é um array numpy se o argumento "t" for um array numpy
a_lam = sy.lambdify(t, a.subs({v_T:6.80, g: 9.80665}), "numpy")

# Representar dados num grafico (usando o matplotlib)
t_array = np.linspace(0.0, 4.0, 100)
plt.plot(t_array, a_lam(t_array))
plt.xlabel("Tempo decorrido, t_array [s]")
plt.ylabel(r"Aceleração, a_lam [m/s$^2$]")
plt.show()
```



4. Mostre que a aceleração é compatível com a fórmula geral,

$$a(t) = g - \frac{g\, v|v|}{v_{\mathrm{T}}^2}$$

Pelo resultado do ponto 2 sabemos que

$$v(t) = v_{\mathrm{T}} \frac{\sinh(gt/v_{\mathrm{T}})}{\cosh(gt/v_{\mathrm{T}})} \Leftrightarrow \frac{\sinh(gt/v_{\mathrm{T}})}{\cosh(gt/v_{\mathrm{T}})} = v/v_{\mathrm{T}}$$

Mas pelo resultado do ponto 3 sabemos que durante a queda a amplitude da aceleração é dada por,

$$a(t) = g - g\left(\frac{\sinh(gt/v_{\mathrm{T}})}{\cosh(gt/v_{\mathrm{T}})}\right)^2,$$

Portanto, substituindo a expressão de $v(t)$ na expressão de $a(t)$ obtemos

$$a(t) = g - \frac{g\,v^2}{v_{\mathrm{T}}^2}$$

Que corresponde ao caso particular em os vetores $\mathbf{v}$ e $\mathbf{g}$ tem o mesmo sentido, ou seja o movimento é descendente e o atrito (des)acelera no sentido oposto à aceleração gravítica.

5. Se o volante for largado de uma altura de 20 m, quanto tempo demora a atingir o solo? Compare com o tempo que demoraria se não houvesse resistência do ar.

In [17]:
```
#sy.solve((20 - y).subs({v_T: 6.80, g: 9.80665}), t)
t20_ar = sy.nsolve((20 - y).subs({v_T: 6.80, g: 9.80665}), t, 0.0)
print("Com resistência do ar, o volante atinge o solo quando t = ", t20_ar, "s")
```

```
Com resistência do ar, o volante atinge o solo quando t =  3.42177372507223 s
```

Se não houvesse resitência do ar, a posição seria $y_{\mathrm{vac}}(t) = \frac{1}{2}gt^2$. Assim

In [18]:
```
# Posição do volante no vácuo (sem resistência do ar)
y_vac = 1/2 * g * t**2

t20_vac = sy.nsolve((20 - y_vac).subs(g, 9.80665), t, 0.0)
print("Sem resistência do ar, o volante atinge o solo quando t = ", t20_vac, "s")
```

```
Sem resistência do ar, o volante atinge o solo quando t =  2.01961997710255 s
```

Pergunta 2:

Se a velocide terminal fosse muito elevada, quão diferente seria o movimento?