Introdução à Computação em Física

PROGRAMAÇÃO SIMBÓLICA EM PYTHON

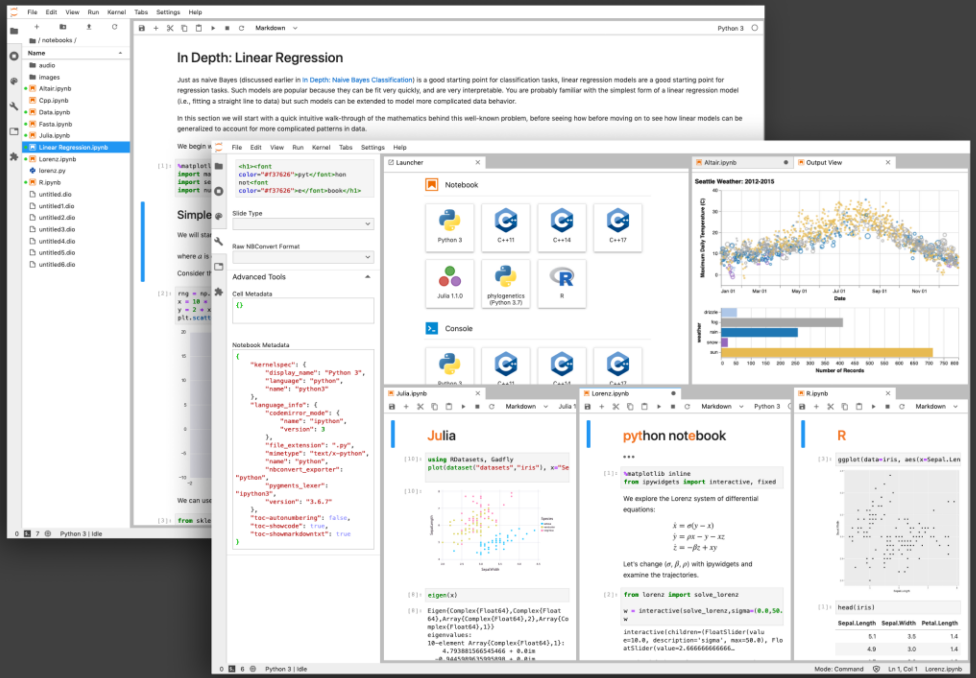
PROF. WALBER

Refs.: Symbolic Computation with Python and Sympy, D. Sandona (2021), https://docs.sympy.org/latest/tutorials/intro-tutorial/index.html

Antes de comerçarmos: Jupyter Lab (Notebook)

- Plataforma do tipo web que permite desenvolvimento de códigos em uma variedade de linguagens, inclusive python.
- Permite compartilhar um código e documenta-lo, na forma de um notebook (*.ipynb);

Disponível em: https://jupyter.org/



Motivações:

1. Seja uma partícula em movimento, que tem sua posição dada por:

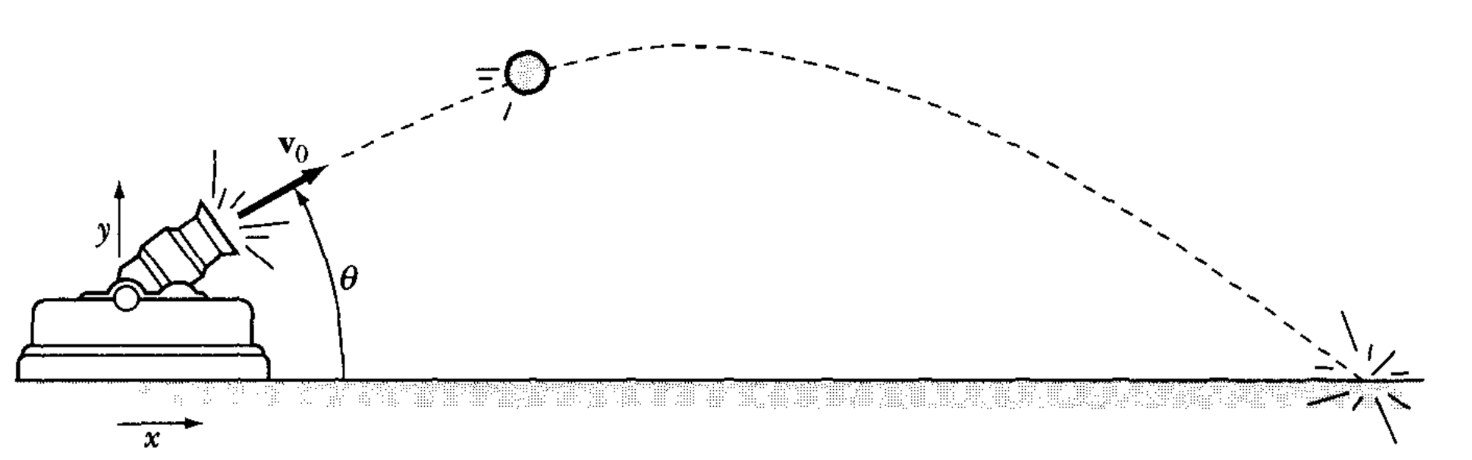
$$x(t) = x_0 + v_o t - \frac{g}{2}t^2$$

Como calcular velocidade e aceleração da partícula em um dado instante de tempo?

Seria possível ensinar o computador a fazer esse cálculo ?

Motivações:

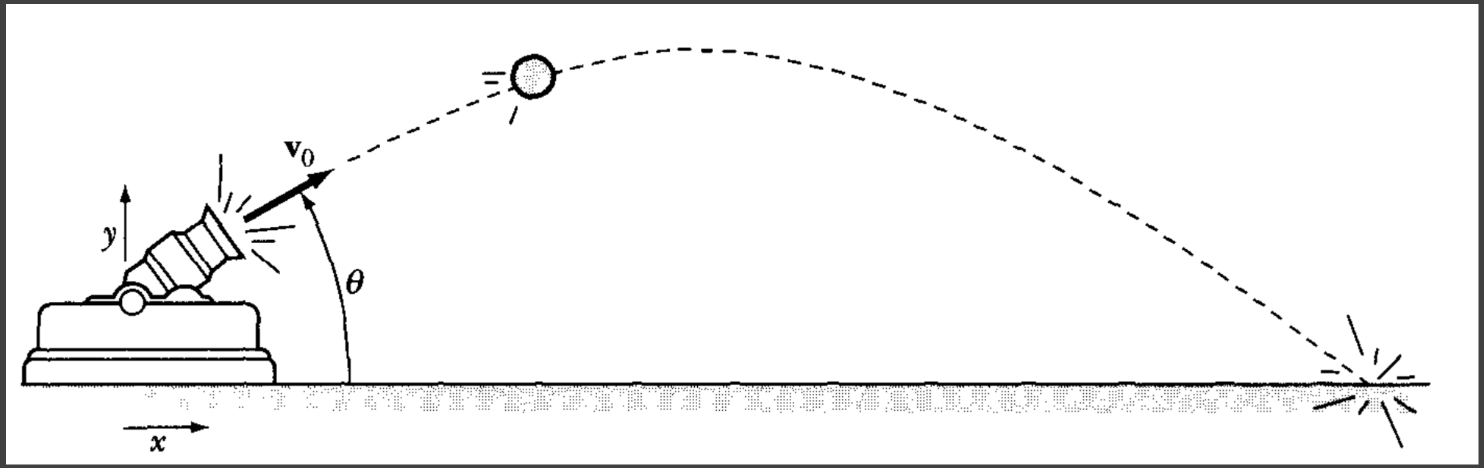
2. Seja o lançamento de um projetil (sem resistência do ar)



Como calcular a posição da partícula em um dado instante de tempo ?

Motivações:

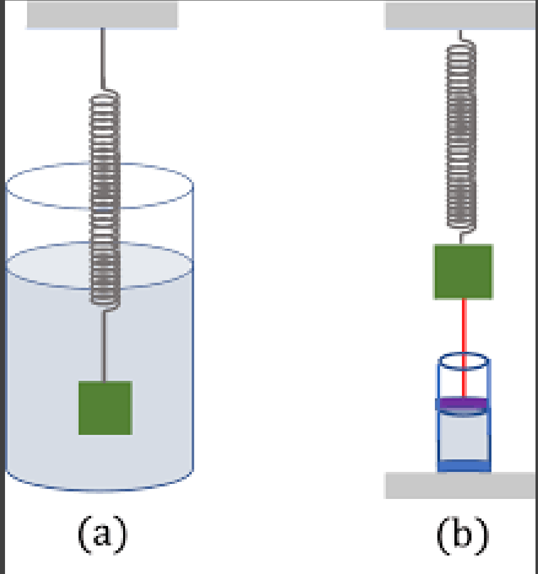
3. Seja o lançamento de um projetil (com resistência do ar)



Como calcular a posição da partícula em um dado instante de tempo considerando resistência proporcional a velocidade ?

Motivações:

4. Oscilação harmônica amortecida



$$\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + \omega_0^2 x = 0$$

Como calcular a posição da partícula em um dado instante de tempo considerando o amortecimento?

Programação Simbólica

- Possibilidade de computação com objetos matemáticos e expressões matemáticas (de maneira analítica ao invés de puramente numérica);

- Ferramenta que auxilia bastante na obtenção de soluções algébricas de problemas complexos, além de permitir a checagem dos resultados obtidos;

- Pode ser utilizada para simplificação de problemas mais complexos que necessitam de métodos numéricos;

- Nesse contexto, faremos uso do módulo SymPy (Symbolic Python):

Mais informações em: www.sympy.org

Elementos básicos

Programação Simbólica

- Sympy possui uma grande variedade de símbolos , e.g. I, pi, ∞, ; Além disso, os símbolos são utilizados como objetos Python, que por sua vez possuem atributos (nome, se é real ou imaginário, etc.);

Exemplo 2 Programação simbólica com SymPy. Nesse contexto iremos configurar o sistema de escrita para escrever no formato matemático mais apropriado.

In [1]:
```
import sympy
sympy.init_printing()
```

```
Vamos agora definir uma variável abstrata x através do símbolo de nome x
```

In [2]:
```
x = sympy.Symbol("x")
```

Podemos ainda mais atributos:

In [5]:
```
x = sympy.Symbol("x",real=True, positive=True)
```

Tal que ao pedir a resolução de algo simples, como:

In [4]:
```
sympy.sqrt(x**2)
```

Out[4]: $|x|$

Caso contrário:

In [6]:
```
y = sympy.Symbol("y")
sympy.sqrt(y**2)
```

Out[6]: $\sqrt{y^2}$

Alguns atributos

Assumption Keyword Arguments	Attributes	Description
`real, imaginary`	`is_real, is_imaginary`	Specify that a symbol represents a real or imaginary number.
`positive, negative`	`is_positive, is_negative`	Specify that a symbol is positive or negative.
`integer`	`is_integer`	The symbol represents an integer.
`odd, even`	`is_odd, is_even`	The symbol represents an odd or even integer.
`prime`	`is_prime`	The symbol is a prime number and therefore also an integer.
`finite, infinite`	`is_finite, is_infinite`	The symbol represents a quantity that is finite or infinite.

Tipos de variáveis, declarações e frações:

Exemplo 3 Podemos declarar um conjunto de variáveis de uma única vez, depois checar o tipo delas:

In [1]:
```
import sympy
sympy.init_printing()

a, b, c = sympy.symbols("a, b, c", real = True, positive = True)
d, e, f = sympy.symbols("d, e, f", real = True, negative = True)
```

In [2]:
```
a.is_real, b.is_positive, d.is_negative
```

Out[2]: (True, True, True)

Frações podem ser introduzidas como:

In [4]:
```
sympy.Rational(20,13)
```

Out[4]: $\frac{20}{13}$

In [5]:
```
sympy.Rational(2,5)
```

Out[5]: $\frac{2}{5}$

In [6]:
```
r1 = sympy.Rational(20,13)
r2 = sympy.Rational(2,5)
r1*r2
```

Out[6]: $\frac{8}{13}$

Algumas constantes e símbolos usuais:

Mathematical Symbol	SymPy Symbol	Description
π	`sympy.pi`	Ratio of the circumference to the diameter of a circle.
e	`sympy.E`	The base of the natural logarithm, $e = \exp(1)$.
γ	`sympy.EulerGamma`	Euler's constant.
i	`sympy.I`	The imaginary unit.
∞	`sympy.oo`	Infinity.

Funções:

```
**Exemplo 4**
Podemos definir funções  através do comando sympy.Function. Iremos definir uma função f(x) e g(x,y,z).
```

In [6]:
```python
import sympy
sympy.init_printing()

x, y, z = sympy.symbols("x, y, z")

f = sympy.Function("f")(x)
g = sympy.Function("g")(x,y,z)
f, g
```

Out[6]: $\left(f(x),\ g(x, y, z)\right)$

In [7]:
```python
g.free_symbols
```

Out[7]: $\{x, y, z\}$

In [8]:
```python
f.free_symbols
```

Out[8]: $\{x\}$

Expressões e operações associadas:

Exemplo 5 Neste exemplo iremos definir expressoes matematicas atraves dos objetos:

In [3]:
```
import sympy
sympy.init_printing()
x = sympy.Symbol("x")
A = 1 + 2*x**2 + 3*x**3 + 4*x**4
A
```

Out[3]: $4x^4 + 3x^3 + 2x^2 + 1$

In [4]:
```
A.args
```

Out[4]: $\left(1,\ 2x^2,\ 3x^3,\ 4x^4\right)$

In [6]:
```
B = 2*x**2 + 3*x**3 + 4*x**4
C = sympy.simplify(B)
C
```

Out[6]: $x^2 \left(4x^2 + 3x + 2\right)$

In [7]:
```
sympy.factor(C)
```

Out[7]: $x^2 \left(4x^2 + 3x + 2\right)$

In [8]:
```
D = x**2-1
sympy.factor(D)
```

Out[8]: $(x - 1)(x + 1)$

Cálculo de derivadas e integrais

Derivadas totais e parciais (funções não definidas):

```
**Exemplo 6**
As derivadas de uma função f(x) podem ser definidas como:
```

In [1]:
```python
import sympy
sympy.init_printing()
x = sympy.Symbol("x")
f = sympy.Function('f')(x)
sympy.diff(f,x)
```

Out[1]:

$$\frac{d}{dx} f(x)$$

In [2]:
```python
sympy.diff(f,x,x)
```

Out[2]:

$$\frac{d^2}{dx^2} f(x)$$

In [3]:
```python
sympy.diff(f,x,x,x)
```

Out[3]:

$$\frac{d^3}{dx^3} f(x)$$

In [4]:
```python
y = sympy.Symbol("y")
g = sympy.Function('g')(x,y)
g.diff(x,y)
```

Out[4]:

$$\frac{\partial^2}{\partial y \partial x} g(x, y)$$

In [5]:
```python
g.diff(x, 3, y, 2)
```

Out[5]:

$$\frac{\partial^5}{\partial y^2 \partial x^3} g(x, y)$$

Derivadas totais e parciais (funções definidas):

Exemplo 7 As derivadas de funções definidas ou até mesmo expressões podem ser feitas como o comando .diff():

In [1]:
```
import sympy
sympy.init_printing()
x = sympy.Symbol("x")
A = x**4 + x**3 + x**2 + x + 1
A.diff(x)
```

Out[1]: $4x^3 + 3x^2 + 2x + 1$

Ou até mesmo com os comandos .Derivative() e .doit()

In [6]:
```
f = sympy.Function('f')(x)
f = sympy.sin(x**3)*sympy.cos(x/2)
f
```

Out[6]: $\sin\left(x^3\right)\cos\left(\frac{x}{2}\right)$

In [7]:
```
d = sympy.Derivative(f,x)
d
```

Out[7]: $\frac{d}{dx}\sin\left(x^3\right)\cos\left(\frac{x}{2}\right)$

In [8]:
```
d.doit()
```

Out[8]: $3x^2\cos\left(\frac{x}{2}\right)\cos\left(x^3\right) - \frac{\sin\left(\frac{x}{2}\right)\sin\left(x^3\right)}{2}$

Derivadas totais e parciais (funções definidas):

In [12]:
```
g = sympy.Function('g')(x)
g = sympy.exp(sympy.cos(x))
g
```

Out[12]: $e^{\cos(x)}$

In [13]:
```
d2 = sympy.Derivative(g,x)
d2
```

Out[13]: $\frac{d}{dx} e^{\cos(x)}$

In [14]:
```
d2.doit()
```

Out[14]: $-e^{\cos(x)} \sin(x)$

In [18]:
```
y = sympy.Symbol("y")
z = sympy.Function('z')(x, y)
z = sympy.sin(x*y)*sympy.cos(x/2) + sympy.exp(x*y)
d3 = sympy.Derivative(z, x)
d3
```

Out[18]: $\frac{\partial}{\partial x}\left(e^{xy} + \sin(xy)\cos\left(\frac{x}{2}\right)\right)$

In [19]:
```
d3.doit()
```

Out[19]: $ye^{xy} + y\cos\left(\frac{x}{2}\right)\cos(xy) - \frac{\sin\left(\frac{x}{2}\right)\sin(xy)}{2}$

In [20]:
```
d4 = sympy.Derivative(z,y)
d4
```

Out[20]: $\frac{\partial}{\partial y}\left(e^{xy} + \sin(xy)\cos\left(\frac{x}{2}\right)\right)$

Derivadas totais e parciais (funções definidas):

Out[20]: $$\frac{\partial}{\partial y}\left(e^{xy} + \sin(xy)\cos\left(\frac{x}{2}\right)\right)$$

```
In [21]: d4.doit()
```

Out[21]: $$xe^{xy} + x\cos\left(\frac{x}{2}\right)\cos(xy)$$

Exemplo 8 Integrais também podem ser calculas, neste caso temos que usar a opção .integrate():

In [5]:
```
import sympy
sympy.init_printing()
a, b, x, y = sympy.symbols("a, b, x, y")
f = sympy.Function('f')(x)
sympy.integrate(f)
```

Out[5]:
$$\int f(x)\,dx$$

In [6]:
```
sympy.integrate(f, (x, a, b))
```

Out[6]:
$$\int_a^b f(x)\,dx$$

In [7]:
```
f = x**4 + x**3 + x**2 + x
sympy.integrate(f, (x, a, b))
```

Out[7]:
$$-\frac{a^5}{5} - \frac{a^4}{4} - \frac{a^3}{3} - \frac{a^2}{2} + \frac{b^5}{5} + \frac{b^4}{4} + \frac{b^3}{3} + \frac{b^2}{2}$$

In [8]:
```
sympy.integrate(f)
```

Out[8]:
$$\frac{x^5}{5} + \frac{x^4}{4} + \frac{x^3}{3} + \frac{x^2}{2}$$

In [9]:
```
sympy.integrate(f, (x, 2, 5))
```

Out[9]:
$$\frac{16407}{20}$$

Integrais

Exemplo 10
Somatórias e produtórias são definidas como .Sum() e .Product():

In [5]:
```python
import sympy
sympy.init_printing()
n = sympy.Symbol('n', integer=True)
S = sympy.Sum(1/(n**2), (n,1,sympy.oo))
S
```

Out[5]:

$$\sum_{n=1}^{\infty} \frac{1}{n^2}$$

In [6]:
```python
S.doit()
```

Out[6]:

$$\frac{\pi^2}{6}$$

In [9]:
```python
P = sympy.Product(n, (n,1,20))
P
```

Out[9]:

$$\prod_{n=1}^{20} n$$

In [10]:
```python
P.doit()
```

Out[10]: 2432902008176640000

Somatórias e produtorias:

Exemplo 11 Equações podem ser resolvidas com a opção .solve():

In [5]:
```
import sympy
sympy.init_printing()
x,y = sympy.symbols('x, y')
sympy.solve(x**2 + 2*x -3)
```

Out[5]: $[-3,\ 1]$

In [6]:
```
a, b, c = sympy.symbols("a, b, c")
sympy.solve(a*x**2 + b*x + c, x)
```

Out[6]:
$$\left[\frac{-b + \sqrt{-4ac + b^2}}{2a},\ -\frac{b + \sqrt{-4ac + b^2}}{2a}\right]$$

In [7]:
```
sympy.solve(sympy.sin(x)-sympy.cos(x),x)
```

Out[7]:
$$\left[-\frac{3\pi}{4},\ \frac{\pi}{4}\right]$$

In [8]:
```
eq1 = x + 2*y-1
eq2 = x-y+1
sympy.solve([eq1,eq2],[x,y], dict=True)
```

Out[8]:
$$\left[\left\{x : -\frac{1}{3},\ y : \frac{2}{3}\right\}\right]$$

Equações:

Atividades praticas

1. Seja a equação da posição de uma partícula em movimento:

$$x(t) = x_0 + v_o t - \frac{g}{2}t^2$$

Escreva um programa que calcule a velocidade e a aceleração da partícula.

2. Escreva um programa para encontrar as soluções das equações abaixo:

$$y^4 + 4y^3 + y + 10 = 0$$

$$tan(y) + y = 0$$

Reporte o ocorrido no último caso

Equações de movimento de partículas
(Problema do valor inicial)

Resolução do problema do valor inicial:

EDO - exemplo1 Vamos agora trabalhar com a solução de equações diferenciais ordinárias. Nesse contexto, iremos utilizar a opção .dsolve().

Seja por exemplo a situação de uma partícula abandonada do alto de uma torre. Da segunda lei de Newton sabemos que:

$m\frac{d^2y}{dt^2} = -mg$, ou seja, $\frac{d^2y}{dt^2} = -g$. Assim sendo, a velocidade fica sendo dada pela solução da equação

$\frac{dv}{dt} = -g$.

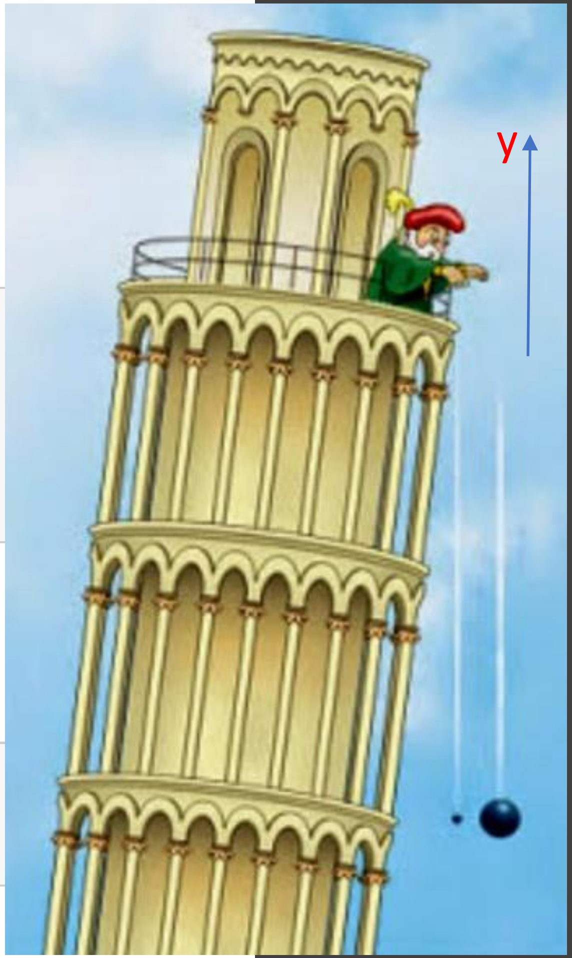
In [3]:
```python
import sympy
sympy.init_printing()
t, g = sympy.symbols("t, g")
v = sympy.Function('v')(t)
eq1 = sympy.Derivative(v, t) + g
sympy.dsolve(eq1, v)
```

Out[3]: $v(t) = C_1 - gt$

Já em relação a equação para a posição da partícula, temos que:

In [4]:
```python
y = sympy.Function('y')(t)
eq2 = sympy.Derivative(y,t,t) + g
sympy.dsolve(eq2,y)
```

Out[4]:
$$y(t) = C_1 + C_2t - \frac{gt^2}{2}$$



Resolução do problema do valor inicial (condições iniciais):

EDO - exemplo2

Sendo:

$m\frac{d^2y}{dt^2} = -mg$, ou seja, $\frac{d^2y}{dt^2} = -g$. Assim sendo, a velocidade fica sendo dada pela solução da equação

$\frac{dv}{dt} = -g$.

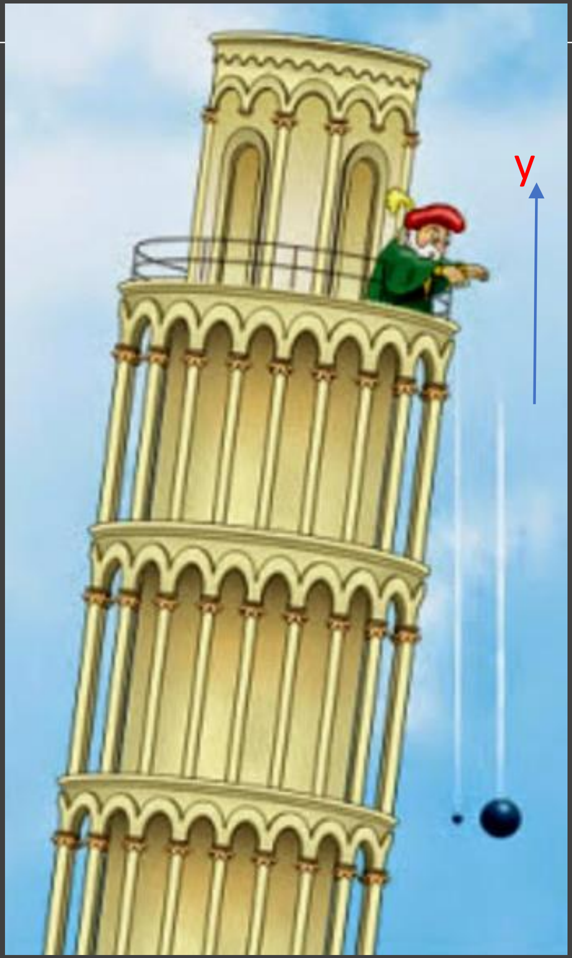
Condições de contorno: $v_0 = 0$ e $y(0) = H = 4m$, onde H é a altura da torre.

```
import sympy as sp
sp.init_printing()
t, g = sp.symbols("t, g")

y = sp.Function("y")

eqy = sp.Eq(y(t).diff(t,2),-g)
eqy
```

$$\frac{d^2}{dt^2}y(t) = -g$$

```
ysol = sp.dsolve(eqy, ics={y(0):4, y(t).diff(t,1).subs(t,0): 0})
ysol
```

$$y(t) = -\frac{gt^2}{2} + 4$$



Queda livre com resistência do ar:

EDO - exemplo3

No caso de um objeto de massa m em queda livre sob ação da força de resistência do ar $-bv$, temos a seguinte equação diferencial:

$m\frac{d^2y}{dt^2} = mg - b\frac{dy}{dt}$, ou seja,

$\frac{d^2y}{dt^2} + k\frac{dy}{dt} - g = 0$, onde $k = \frac{b}{m}$.

Para a velocidade, temos:

$$\frac{dv}{dt} + kv - g = 0$$

In [4]:
```
import sympy
sympy.init_printing()
t, g, k = sympy.symbols("t, g, k")

v = sympy.Function('v')(t)
edoV = sympy.Derivative(v,t) + k*v - g
edoV
```

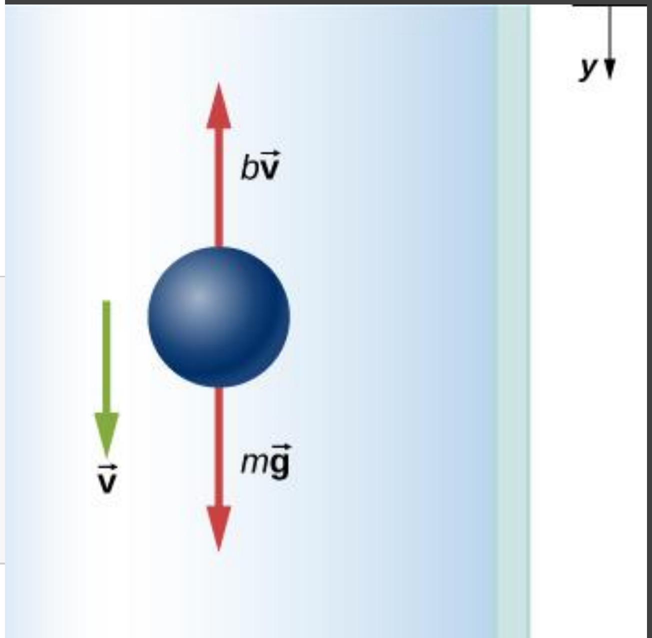
Out[4]:

$$-g + kv(t) + \frac{d}{dt}v(t)$$

In [7]:
```
eqv = sympy.dsolve(edoV,v)
eqv
```

Out[7]:

$$v(t) = \frac{g + e^{k(C_1 - t)}}{k}$$



Queda livre com resistência do ar:

In [17]:
```python
y = sympy.Function('y')(t)
edoY = sympy.Derivative(y,t,t) + k*sympy.Derivative(y,t)-g
edoY
```

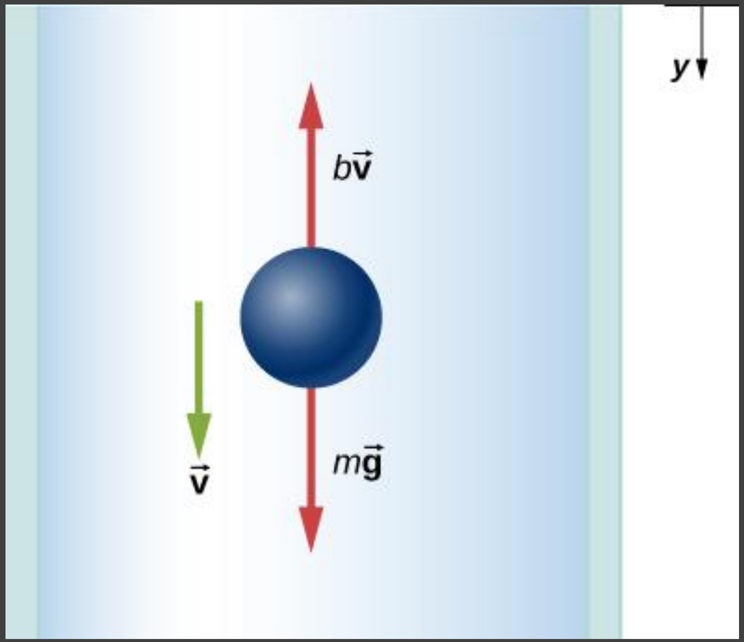
Out[17]:

$$-g + k\frac{d}{dt}y(t) + \frac{d^2}{dt^2}y(t)$$

In [19]:
```python
eqy = sympy.dsolve(edoY, y)
eqy
```

Out[19]:

$$y(t) = C_1 + C_2e^{-kt} + \frac{gt}{k}$$



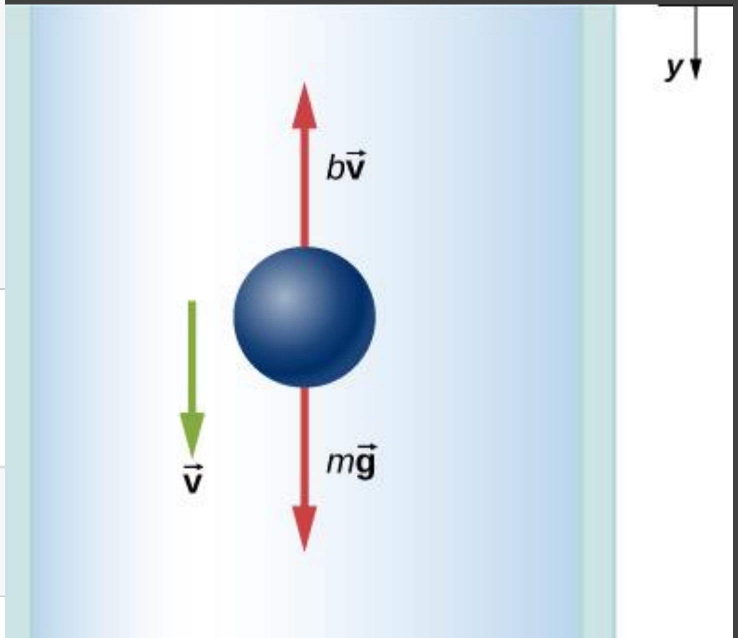
Queda livre com resistência do ar (com condições iniciais):

```python
import sympy as sp
sp.init_printing()
t, g, k = sp.symbols("t, g, k")

y = sp.Function("y")

eqy = sp.Eq(y(t).diff(t,2),g - k*y(t).diff(t,1))
eqy
```

$$\frac{d^2}{dt^2}y(t) = g - k\frac{d}{dt}y(t)$$

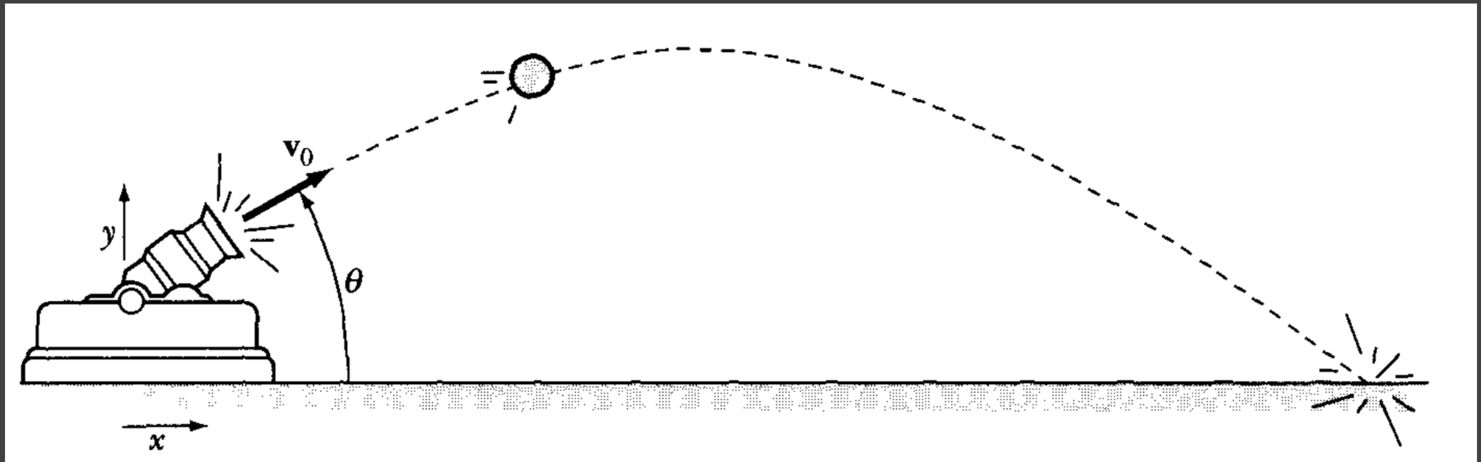
```python
ysol = sp.dsolve(eqy, ics={y(0):20, y(t).diff(t,1).subs(t,0): 0})
ysol
```

$$y(t) = \frac{gt}{k} + \frac{ge^{-kt}}{k^2} + \frac{-g + 20k^2}{k^2}$$



Atividades praticas

1. Escreva um programa para calcular a posição do projetil lançado com velocidade inicial v0 e ângulo θ



(a) Na ausência de resistência do ar
(b) Na presença de resistência do ar
(c) Faça um programa que escreva em um arquivo os valores de x(t),y(t) (duas colunas) para k = 0.08, 0.01 e 0 (theta = 60, v0 = 600 m/s) para tempos suficientes para o projetil tocar o chão. Plote os resultados.

Atividades praticas

2. Escreva um programa para obter x(t) do oscilador harmônico simples:

$$\frac{d^2x}{dt^2} + \omega_0^2 x = 0 \qquad \omega_0^2 = \frac{k}{m}$$

3. Escreva um programa para obter x(t) do oscilador harmônico amortecido:

$$\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + \omega_0^2 x = 0$$