pyladies

São Carlos - SP

Minicurso: simulações numéricas com Python

03 e 05 de Setembro de 2018

Não estamos sozinhas!

Apoio

raccoon
DIGITAL MARKETING

Um pouco de história

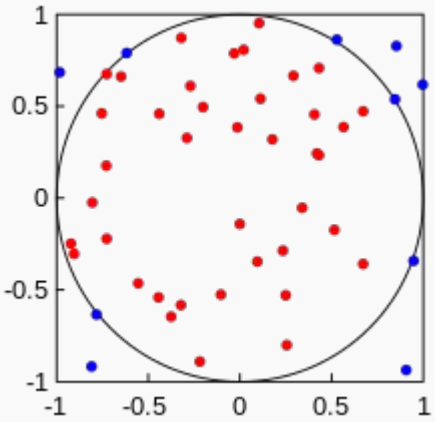
Uso da computação facilitando cálculos diversos …

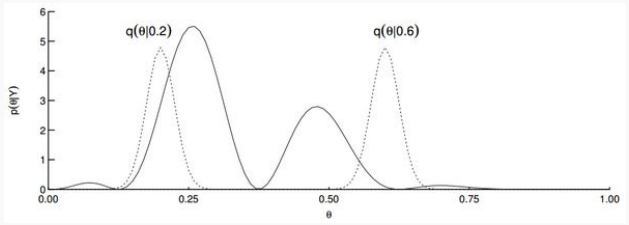
computador é uma calculadora mais eficiente, substituindo calculadoras humanas

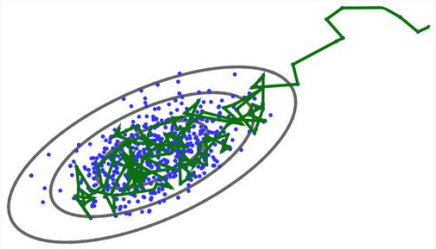
Um pouco de história

Projeto Manhattan

- déc. 60, Monte Carlo
- Algoritmo de Metrópolis



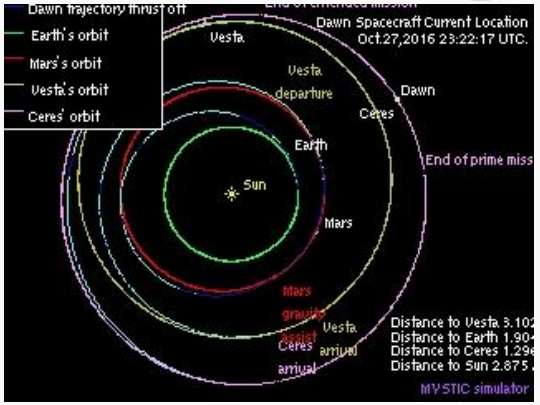




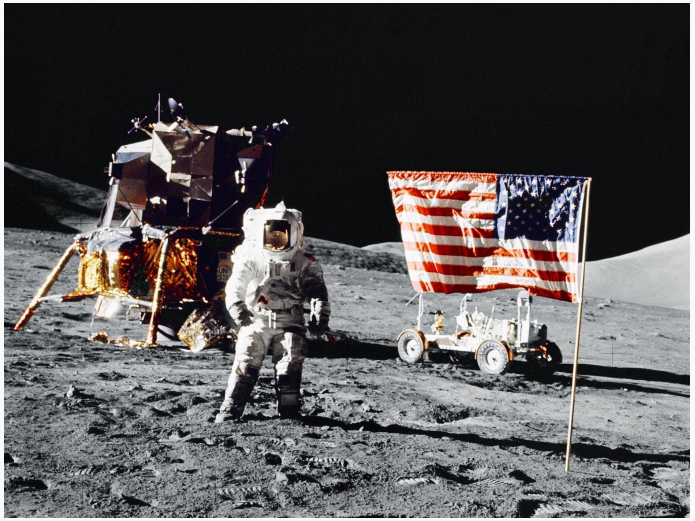
Um pouco de história

Cálculo de trajetórias na NASA

- missões espaciais
- (as “minas do Estrelas além do tempo”)



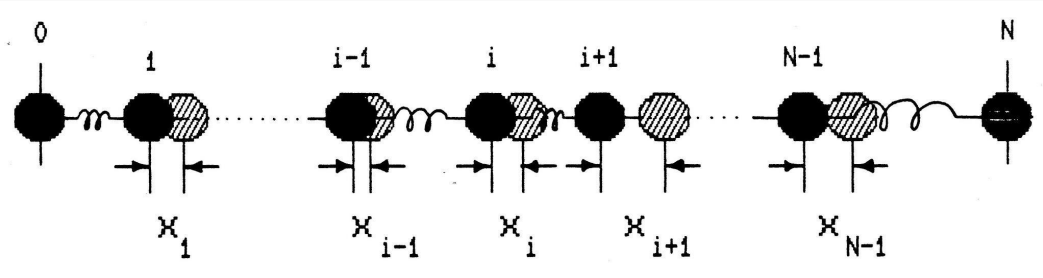
- missões Apolo, etc



Um pouco de história

1o experimento numérico

- 1953 Fermi-Pasta-Ulam-Tsingou (T a "mina" dos código)



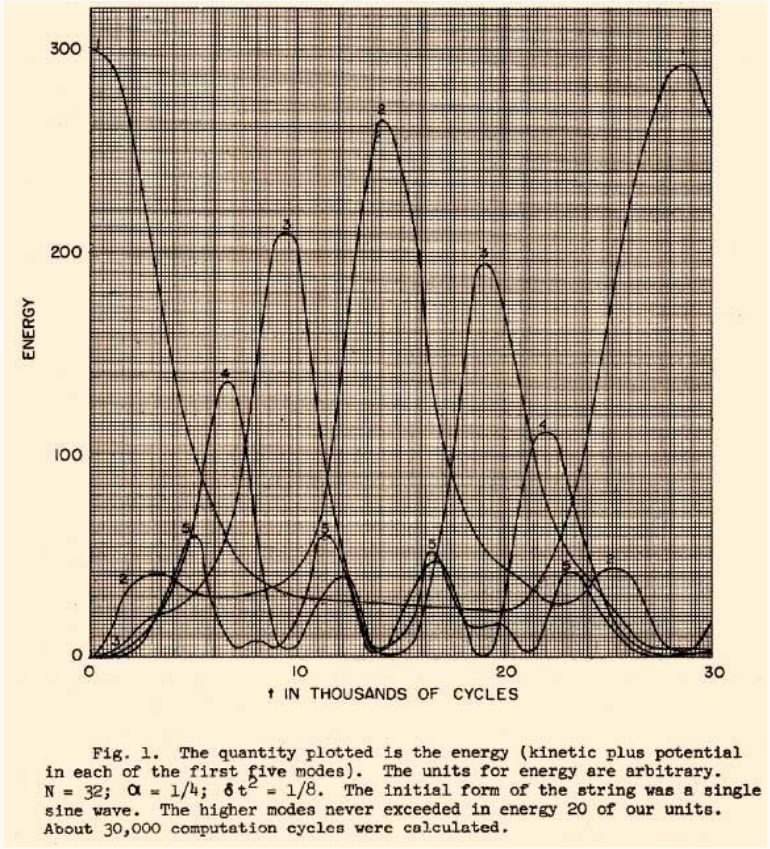


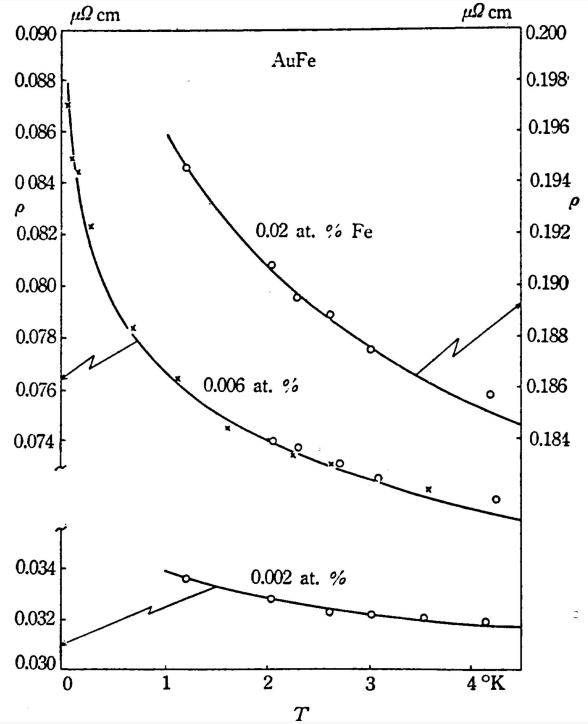
Fig. 1. The quantity plotted is the energy (kinetic plus potential in each of the first five modes). The units for energy are arbitrary. N = 32; $\alpha = 1/4$; $\delta t^2 = 1/8$. The initial form of the string was a single sine wave. The higher modes never exceeded in energy 20 of our units. About 30,000 computation cycles were calculated.

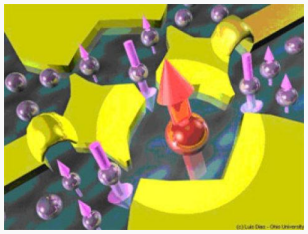
https://arxiv.org/abs/0801.1590

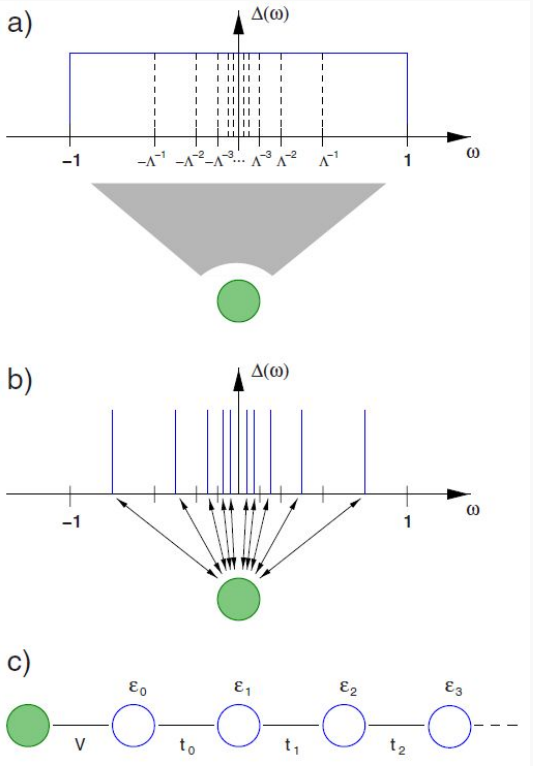
Um pouco de história

Cálculo numérico em física da matéria condensada

- 1964, Ken Wilson e o problema Kondo: o Grupo de Renormalização numérico





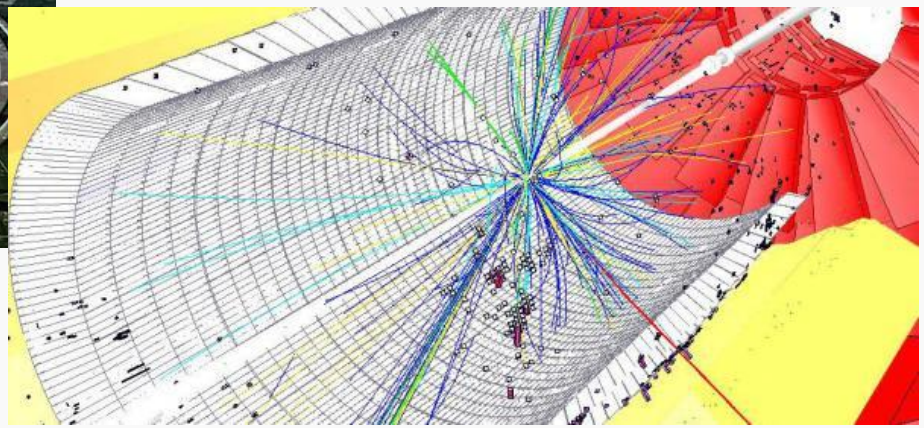


https://journals.aps.org/rmp/abstract/10.1103/RevModPhys.47.773

Um pouco de história

1998: Large Hadron Collider (LHC)

- simulações de colisões de partículas altamente energéticas
- análise de um ENORME conjunto de dados experimentais

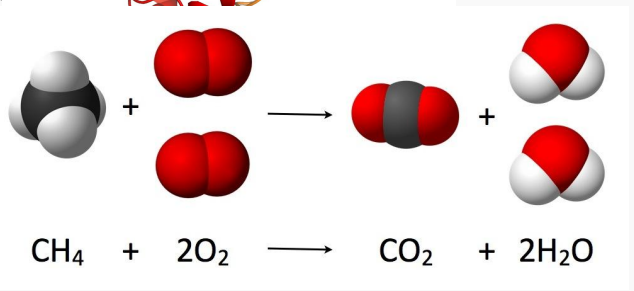


http://lhcathome.web.cern.ch/projects/test4theory/high-energy-physics-simulations

Um pouco de história

E hoje?

- Biologia: enovelamento de proteínas
- Química: reações, estrutura de materiais
- Física: todas as áreas! hot topic: computação quântica
- Medicina: espalhamento de doenças
- Sociologia: interações humanas
- Finanças: análise de risco
- etc etc etc ...

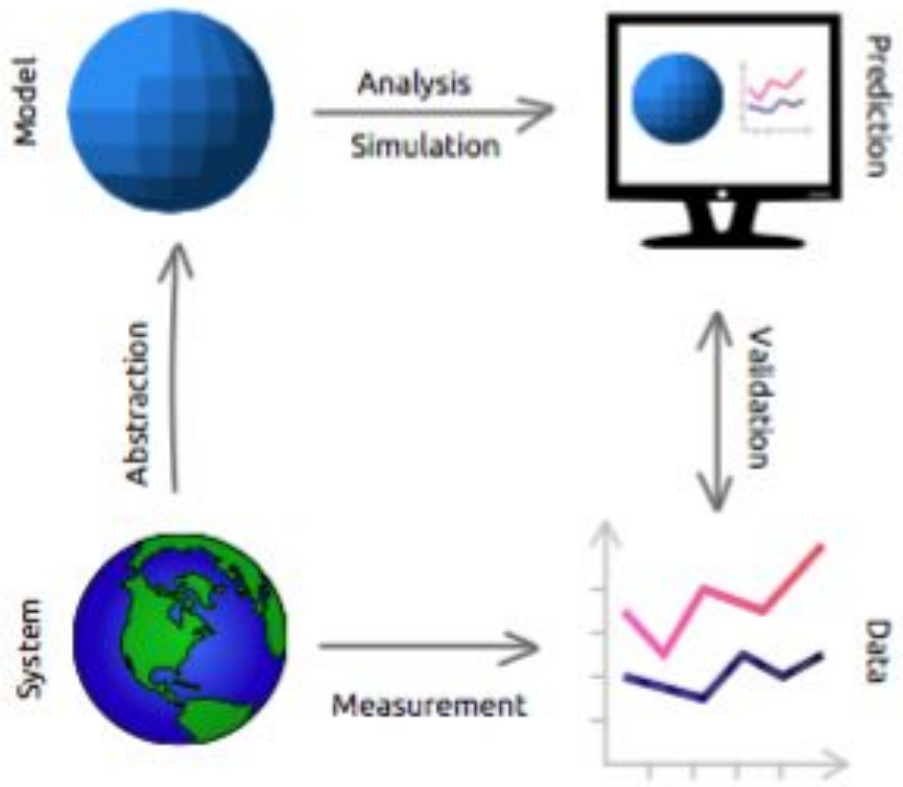


Pacotes científicos em python

- Numpy: básico para computação numérica (vetores, álgebra linear e matricial)
  http://www.numpy.org/
- SciPy: computação científica (cálculo numérico, integrais, ODE's, raízes)
  https://www.scipy.org/
- Matplotlib: visualização e gráficos (2D e 3D)
  https://matplotlib.org/
- Pandas: processamento e análise de dados (organização e estatística)
  https://pandas.pydata.org/
- Sympy: computação simbólica (trabalhando com símbolos **αβγδ**xyz)
  https://www.sympy.org/en/index.html
- Pint: biblioteca de unidades SI (kg, m, s)
  https://pint.readthedocs.io/en/latest/

Modelos

Modelos: simplificações de sistemas do mundo real e tradução em forma matemática (abstração)



- Análise
- Simulação
- Predição

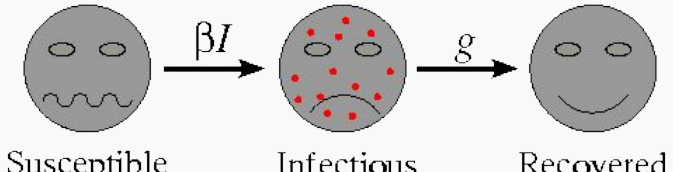
- Validação
- Coleta de dados
- Aperfeiçoamento

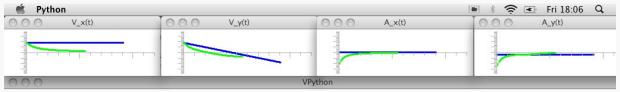
Simulações com Python

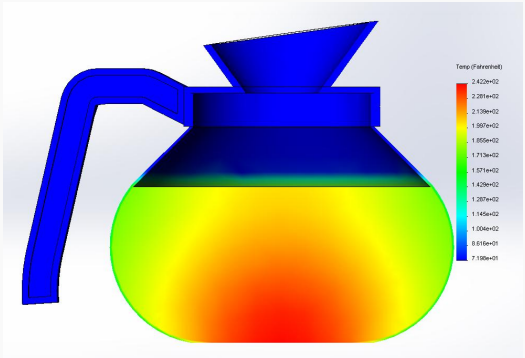
Excelente referência: ModSimPy (https://github.com/AllenDowney/ModSimPy)

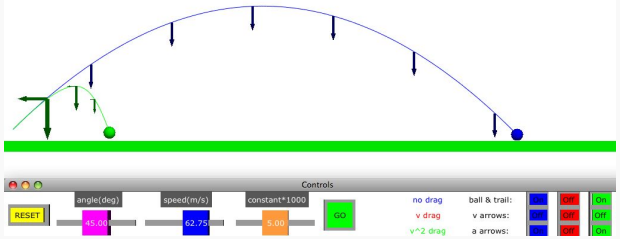
- módulo open source
- introdutório e pedagógico
- exemplos com jupyter notebooks
- interface para gráficos

jupyter









Ubuntu: para quem já possui python 3.6>=

```
python3 -m pip install --user
jupyter matplotlib pandas seaborn
pint scipy
```

No terminal:

jupyter notebook

https://bit.ly/2NferAQ

ModSimPy: requerimentos básicos

Python >=3.6

- Scipy (numpy, matplotlib, sympy)
- pint

Preferência de instalação:

ANACONDA®

https://anaconda.org/anaconda/python

Anaconda vem com todos os pacotes do Scipy

+ IPython
+ Jupyter Notebooks

Instalando o Anaconda

No Windows:

https://conda.io/docs/user-guide/install/windows.html

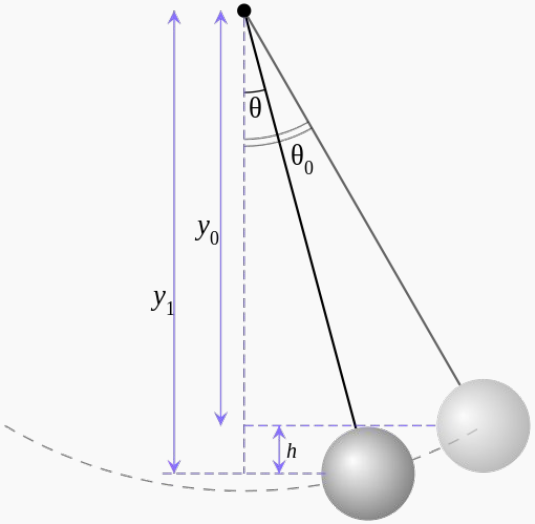
No Linux:

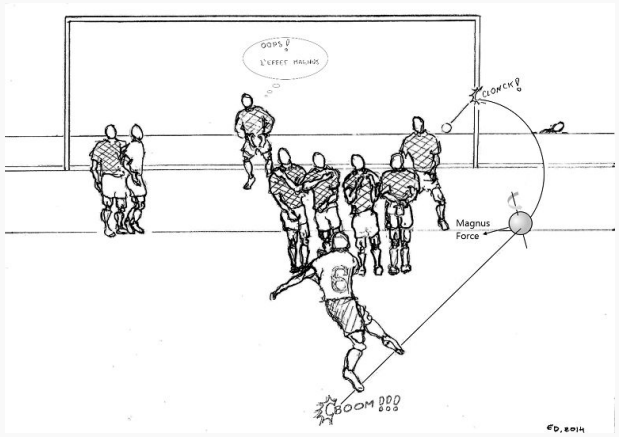
https://conda.io/docs/user-guide/install/linux.html

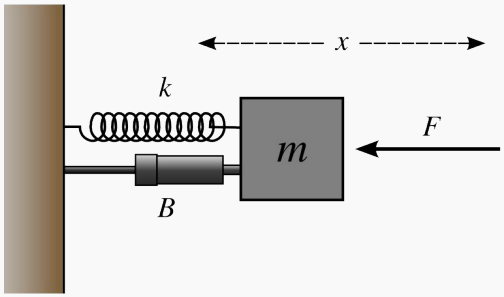
No MacOSX:

https://conda.io/docs/user-guide/install/macos.html

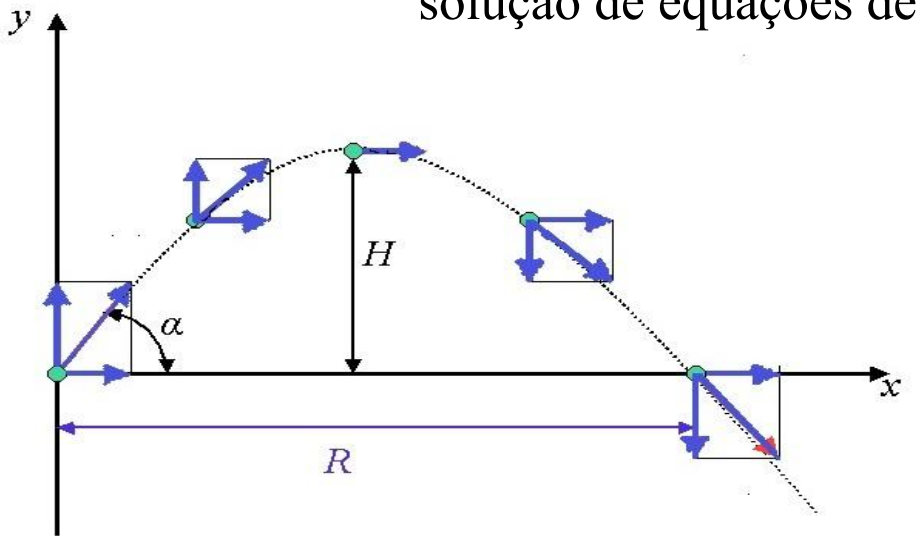
Simulações de mecânica clássica

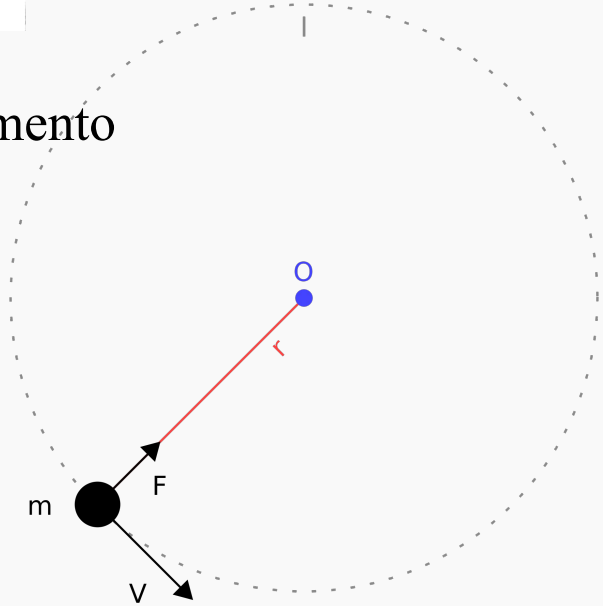






solução de equações de movimento





Instalando o pint

Via Anaconda:

Linux: no terminal

Windows: no Anaconda Prompt (pode requerer privilégios de adm)

digite

```
conda install -c conda-forge pint
```

Via Pip:

```
pip install -U pint
```

EX: bola caindo

Interesse: simular movimento de corpos

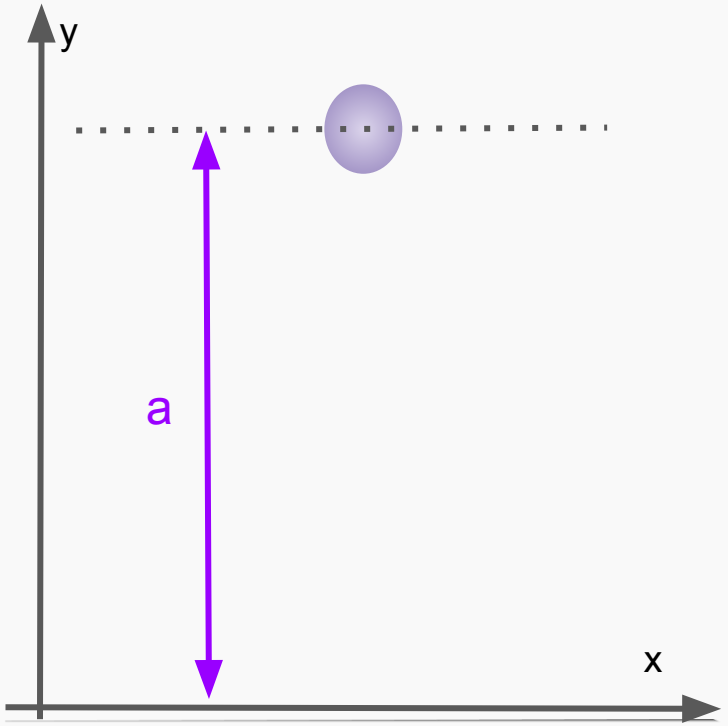
Ingredientes:

- corpo sob ação de forças
- equação diferencial governada pela 2 Lei de Newton
- solução da equação diferencial

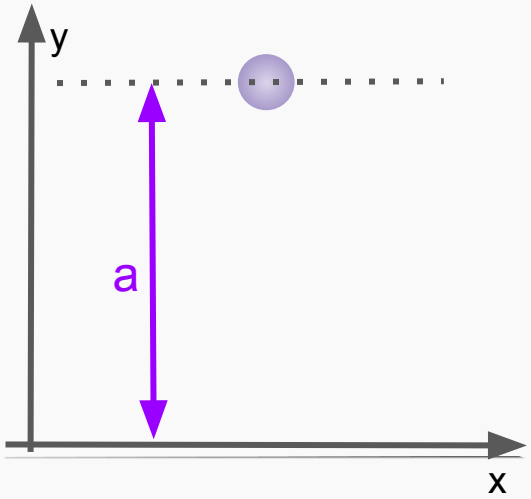
Ex: bola caindo sob ação da gravidade

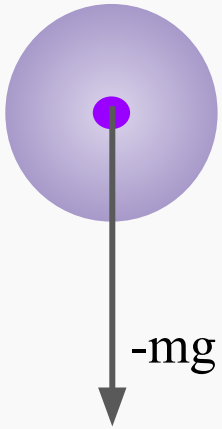
Abordagem:

- definição do modelo
- definição das forças
- definição da equação de movimento
- variáveis de interesse
- condições iniciais
- métodos de solução



EX: bola caindo





Lei de Newton na direção y

$$F_y = ma_y(t) = m\frac{d^2y}{dt^2}$$

Força em y:

$$F_y = -mg$$

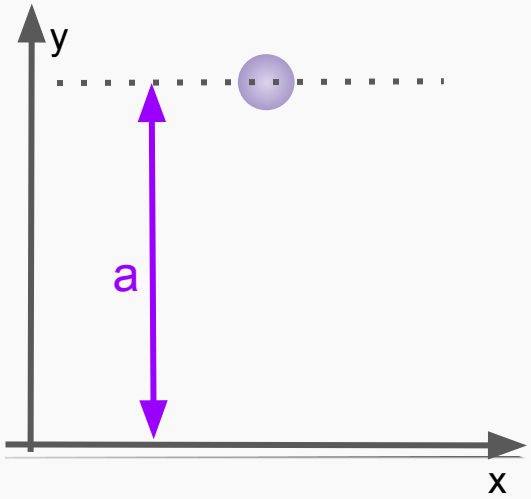
Equação de movimento:

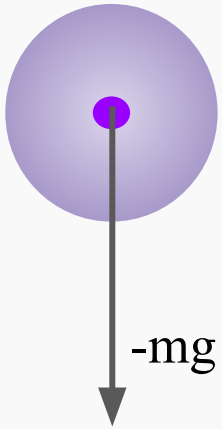
$$m\frac{d^2y}{dt^2} = -mg$$

y(t) é a função que satisfaz

$$\frac{d^2y}{dt^2} - g = 0$$

EX: bola caindo





Problema:

$$\frac{d^2y}{dt^2} - g = 0$$

Solução analítica:

variável auxilar v: velocidade

$$\frac{dy}{dt} = v$$

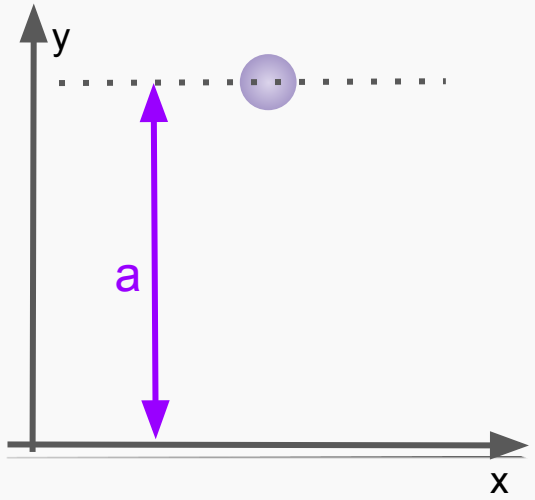
reescrevendo

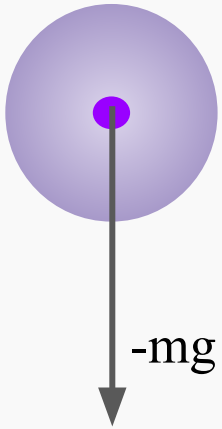
$$\frac{dv}{dt} = -g$$

integrando

$$\int_{t_0}^{t_f} \frac{dv}{dt} dt = -g \int_{t_0}^{t_f} dt \qquad v(t_f) - v(t_0) = -g(t_f - t_0)$$

EX: bola caindo





Integral indefinida

$$v(t) = v(t_0) - gt$$

Voltando à posição …

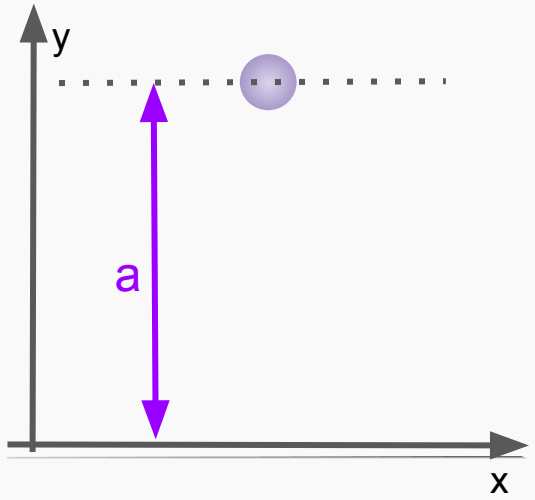
$$\frac{dy}{dt} = v(t)$$

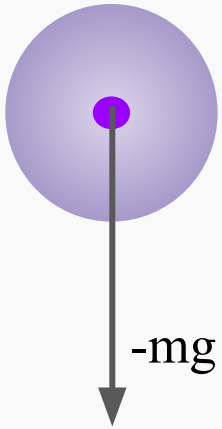
integrando novamente $\int \frac{dy}{dt} dt = \int v(t) dt$

substituindo

$$y(t) = y(t_0) + v_0 t - g\frac{t^2}{2}$$

EX: bola caindo





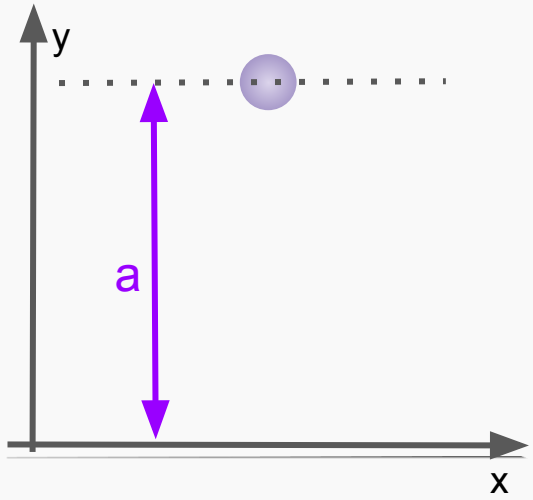
$$y(t) = y_0 + v_0 t - g\frac{t^2}{2}$$

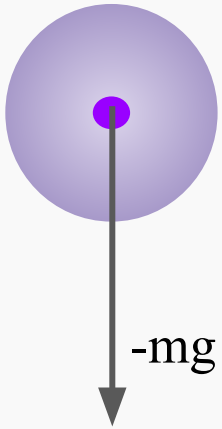
constantes dadas pelas condições iniciais

Ex: $y(0) = a\,m$
$v(0) = 0\,m/s$

estaríamos interessados em prever quanto tempo leva para a bola alcançar o chão (y(t) = 0) para um dada altura **a**

EX: bola caindo





OK: esse exemplo é simples e tem solução analítica!

MAS: e se tomássemos a bola como ela é (borracha cheia, deformável) e considerássemos a resistência do ar, rolamento, etc

modelo mais rebuscado e equações diferenciais de difícil solução:

SIMULAÇÕES NUMÉRICAS!

Resolvendo equações diferenciais numericamente

Qual o problema?

queremos determinar f(t) para y que obedece à eq.

onde g(t,y) é uma função qualquer do tempo e do próprio y $\frac{df}{dt} = g(t, f)$

EX: bola caindo

- queremos y(t)
- temos uma eq. de ordem 2
- mudamos de variável (de y(t) para v(t))
- a eq. para v(t) é de ordem 1!

NOTE: t é uma variável contínua e y(t) é uma função contínua!

Resolver ED numericamente requer trabalhar com números, isto é, trabalhar com variáveis discretas

Resolvendo equações diferenciais numericamente

Quais ingredientes:

- eq. diferencial para f(t): queremos f(t) e sabemos a taxa de variação g(f,t) de f com o tempo

$$\frac{df}{dt} = g(t, f)$$

- queremos calcular f entre $t = 0 \quad t = t_f$

- definimos um intervalo temporal total em N-1 intervalos de tamanho h

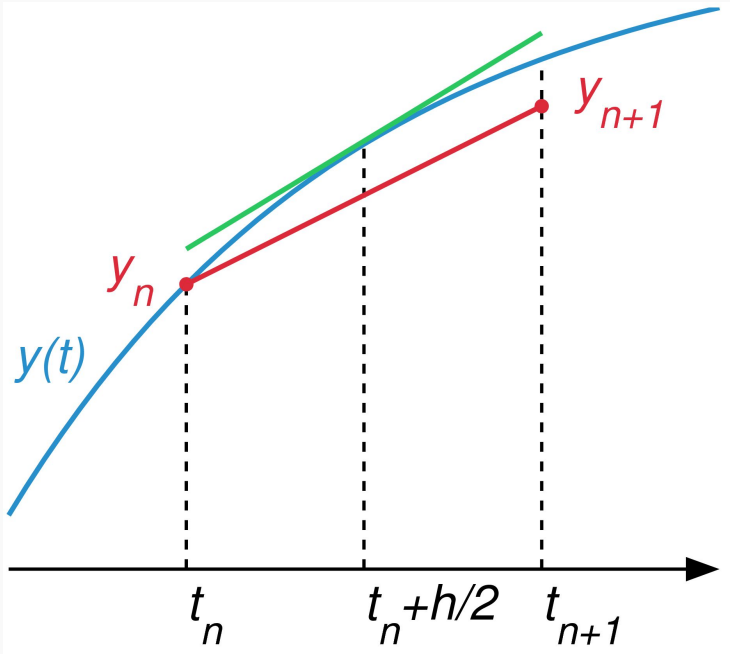
y(t) vai ser determinado para todo o conjunto de N pontos

$$y(t_0), y(t_0 + h), y(t_0 + 2h), y(t_0 + 3h), \ldots, y(t_0 + Nh)$$

Resolvendo equações diferenciais numericamente

Precisamos calcular o conjunto de pontos $t_n, f(t_n)$

considerando que f(t) obedece $\frac{df}{dt} = g(t, f)$



Métodos numéricos diversos …
um dos métodos mais usados é o **Runge Kutta de ordem 4**

Resolvendo equações diferenciais numericamente

Runge-Kutta 4

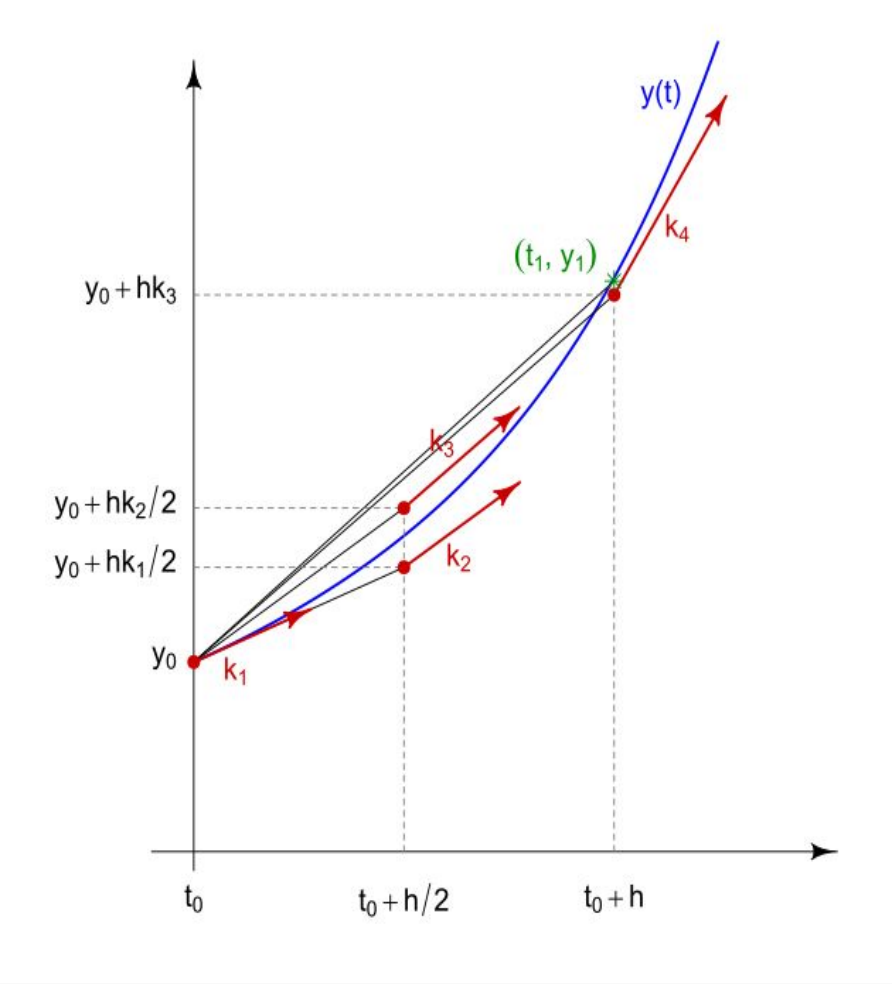
Problema: $\frac{dy}{dt} = f(t, y)$

Solução:

$$t_{n+1} = t_n + h$$

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = hf(t_n, y_n)$$
$$k_2 = hf(t_n + \frac{h}{2}, y_n + \frac{k_1}{2})$$
$$k_3 = hf(t_n + \frac{h}{2}, y_n + \frac{k_2}{2})$$
$$k_4 = hf(t_n + h, y_n + k_3)$$



Python e equações diferenciais

Suporte Python para equações diferenciais: **scipy.integrate**

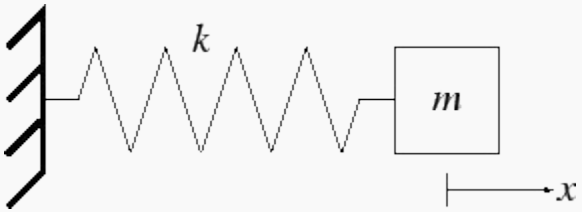
scipy.integrate.solve_ivp

$$\frac{dy}{dt} = f(t, y)$$
$$y(t = 0) = y_0$$

Runge Kutta de ordem 4 e outros

```
from scipy.integrate import solve_ivp

def funcao_dydt(t,y):
   return (y,f(t,y))

solve_ivp(funcao_dydt,[t_0,t_f],y0)
```

Warmp-up: Oscilador Harmônico



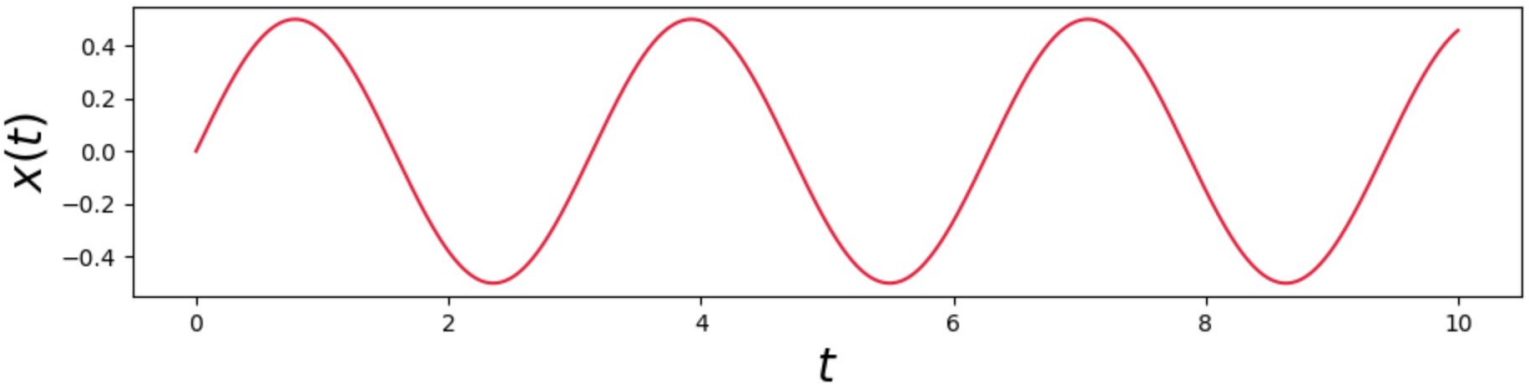
Lei de Newton:

$$\mathbf{F} = m\frac{d^2\mathbf{r}}{dt^2}$$

$$\frac{d^2x}{dt^2} = -\frac{k}{m}x$$

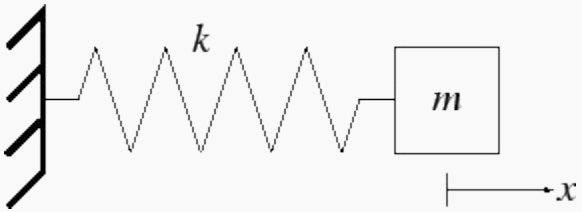
frequência de oscilação

$$\omega_0 = \sqrt{\frac{k}{m}}$$



bit.ly/2NferAQ

Warmp-up: Oscilador Harmônico



Equação de ordem 2: $\frac{d^2x}{dt^2} = -\omega_0^2 x$

Mudança de variável: $\frac{dx}{dt} = v$

Equação de ordem 1: $\frac{dv}{dt} = -\omega_0 x$

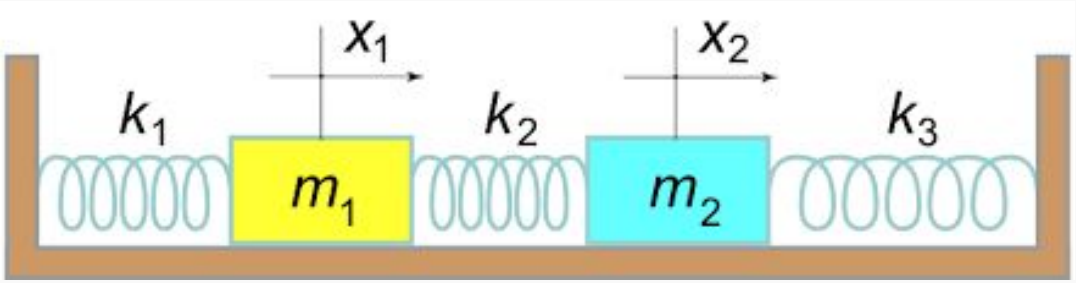
Identificando: $f(t,x) = -\omega_0 x$

$x = x(t)$

```
from scipy.integrate import
solve_ivp

def funcao_dvdt(t,x):
    return - omega2 * x

def funcao_d2xdt2(t, xv):
    return (xv[1],
funcao_dvdt(t,xv[0]))

xvt =
solve_ivp(funcao_d2xdt2,[t_0,t_f
],(x0,y0))

x_t = xvt.y[0]
v_t = xvt.y[1]
```

Osciladores acoplados

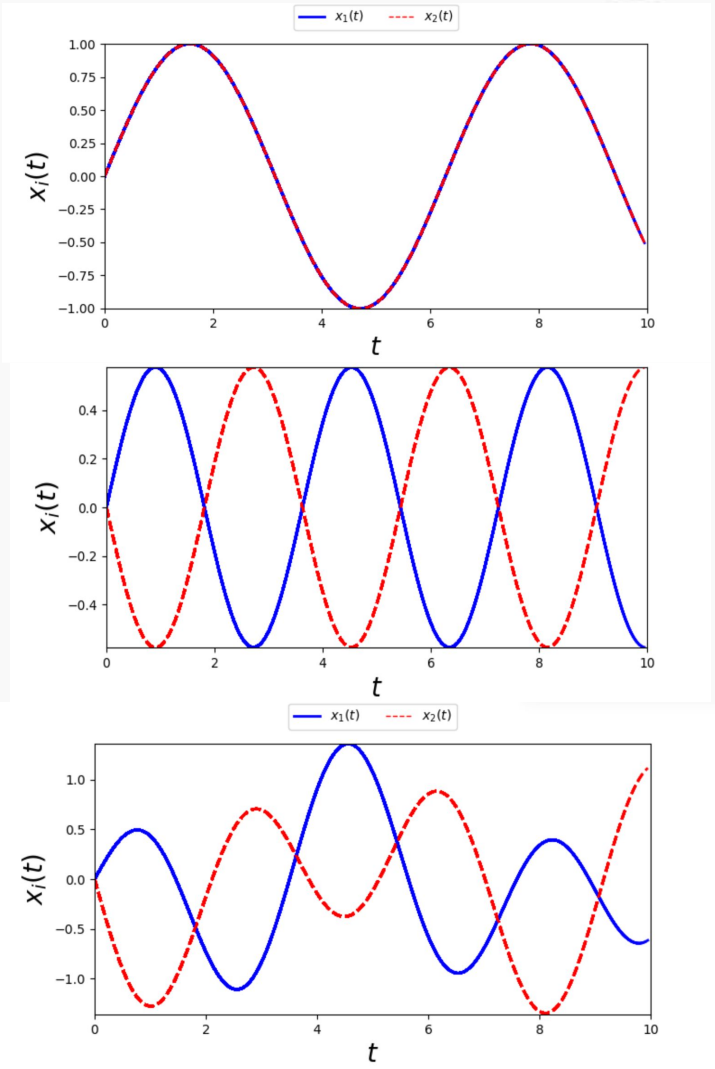
Complicando um pouquinho ...



$$m_2 \frac{d^2 x_2}{dt^2} = -k_2(x_2 - x_1) - k_3 x_2$$

$$m_1 \frac{d^2 x_1}{dt^2} = -k_2(x_1 - x_2) - k_1 x_1$$

ver exemplo …
https://www.dropbox.com/sh/zh8gy22phskg0fn/AAAZlQx3F0C5BlCzAKOAgIHYa?dl=0

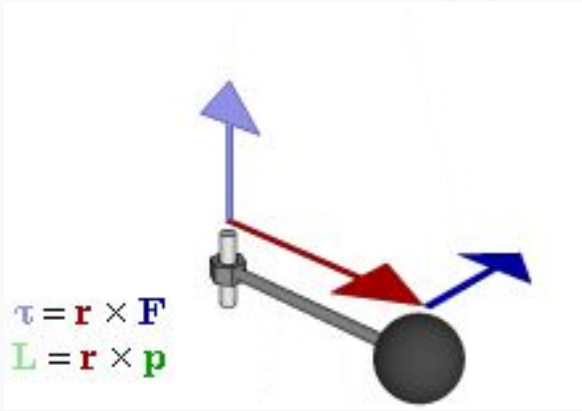


Rotações

Definições e quantidades importantes

Momento angular: $\mathbf{L} = \mathbf{r} \times \mathbf{p}$

Torque: $\tau \equiv \mathbf{r} \times \mathbf{F}$



Lei de Newton para rotações: $\mathbf{L} = I\omega\hat{\mathbf{u}}$

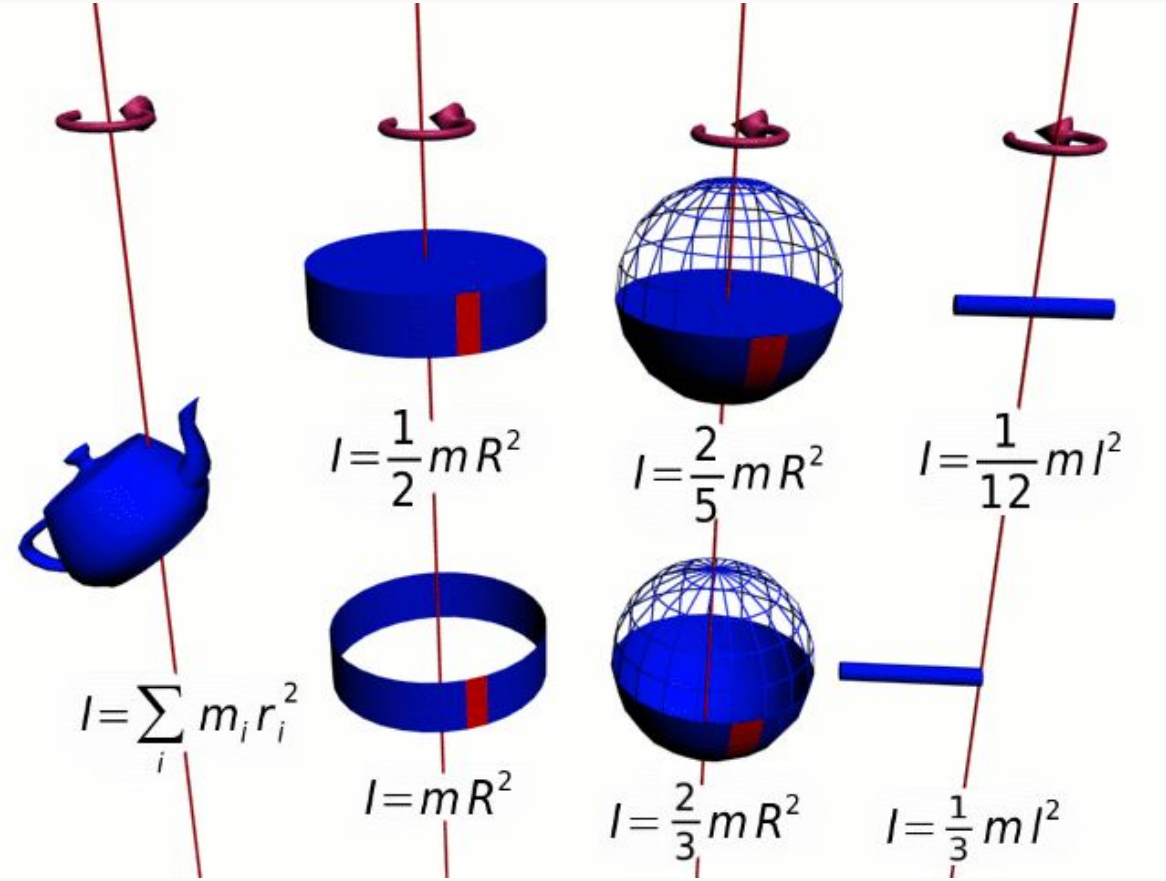
análogo da velocidade

análogo da massa

$$\frac{d\mathbf{L}}{dt} = \frac{dI}{dt}\omega + I\frac{d\omega}{dt}$$

Momento de inércia

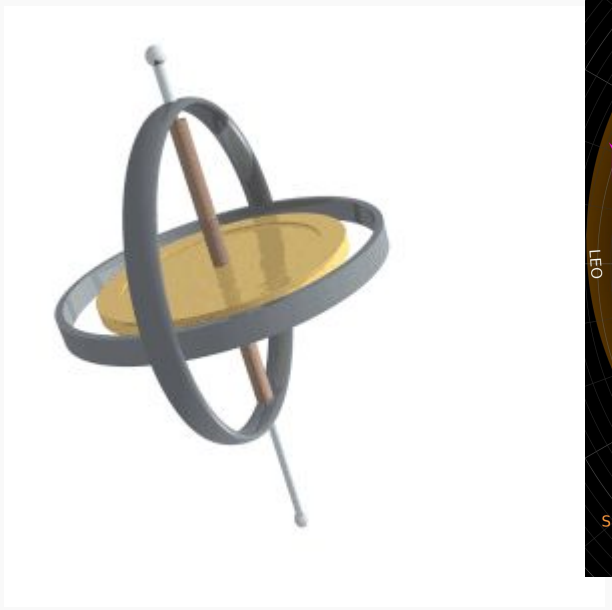
$$I = \iiint d\mathbf{r}\, \rho(\mathbf{r}) ||\mathbf{r}^2||$$

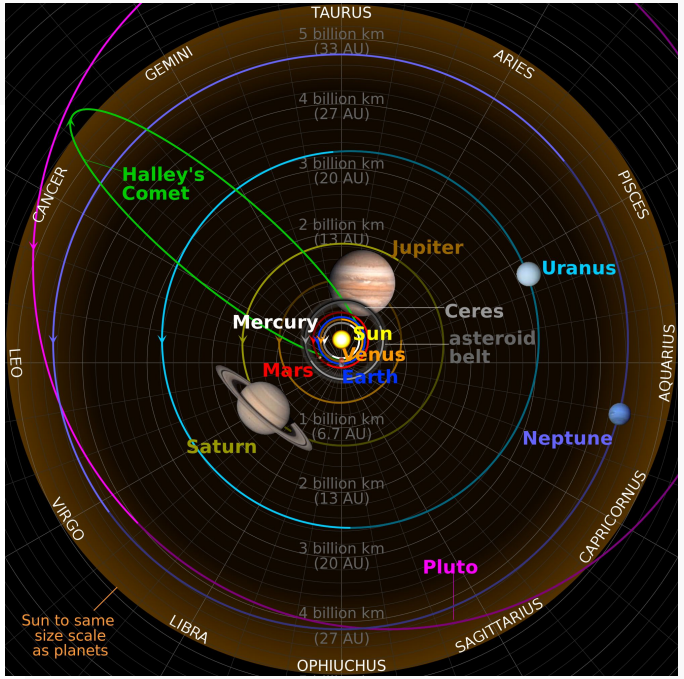


Conservação do momento angular

$$\frac{d\mathbf{L}}{dt} = \sum_i \tau_i$$

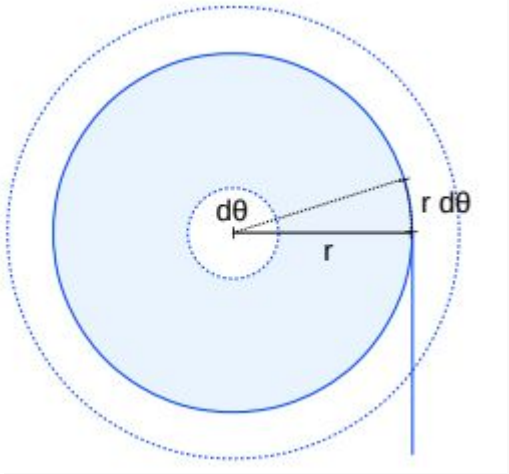
equação diferencial
para o movimento!





y
θ
k

O problema do papel higiênico: fabricação



$$\omega \equiv \frac{d\theta}{dt}$$

y comprimento enrolado

θ ângulo

k taxa de variação do raio

$$\frac{dy}{dt} = r\frac{d\theta}{dt}$$

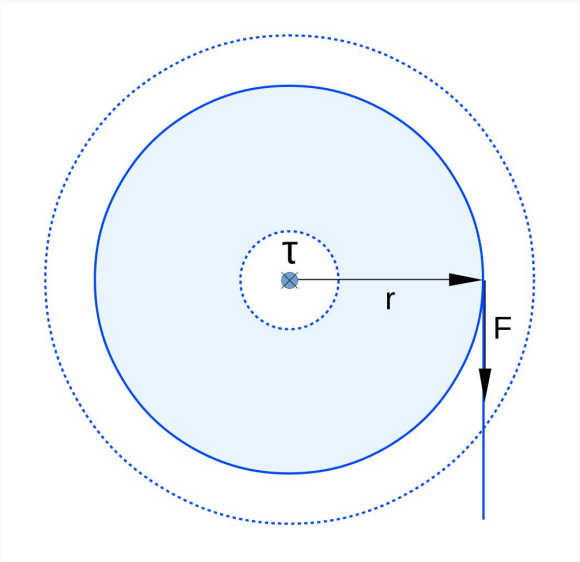
$$\frac{dr}{dt} = k\frac{d\theta}{dt}$$

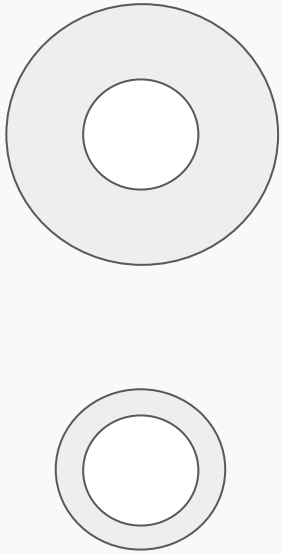
code rolling paper

O problema do papel higiênico: gato

sistema real :

sistema físico:

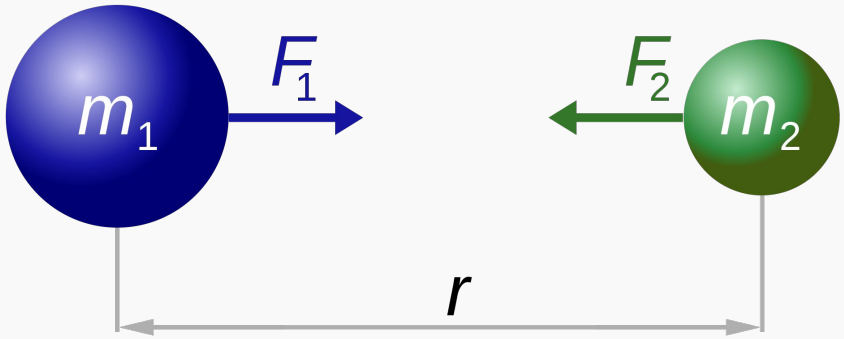




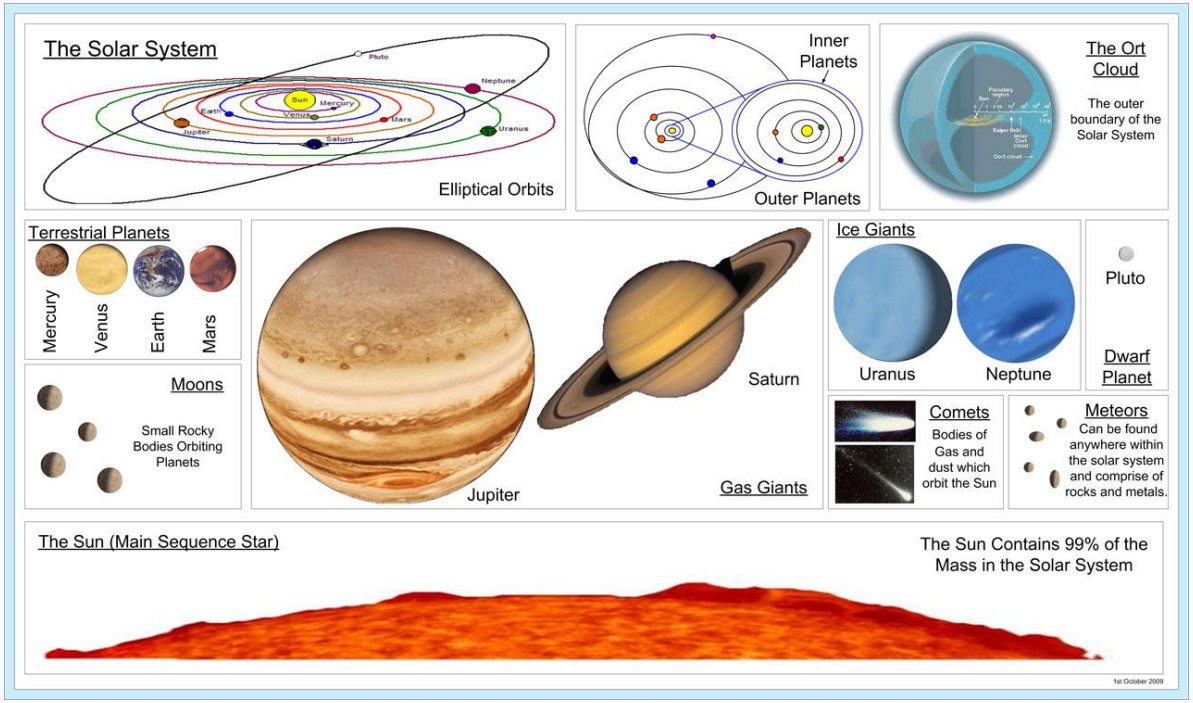
Gato faz força para puxar o papel
Força p/ rolamento <-> torque

code crazy cat paper

Gravitação



$$F = G\frac{Mm}{r^2}$$



Simulando um sistema planetário

pygame