Introdução à Física Computacional II (4300318)

Prof. André Vieira
apvieira@if.usp.br
Sala 3120 – Edifício Principal

Aula 1

Solução numérica de equações diferenciais: métodos de Euler e de Runge-Kutta (2ª. ordem)

Ferramentas

- Iniciem os computadores no ‘linux’
- Localizem o ambiente de desenvolvimento ‘IDLE’
- Na *shell* que for aberta, cliquem em ‘File → New File’
- Na janela de edição que for aberta (não na shell!), cliquem em ‘Save as’ e escolham um nome para o arquivo que termine em ‘.py’, tal como ‘teste.py’.

Equações diferenciais ordinárias

$$\frac{d^n x}{dt^n} = f\left(\frac{d^{n-1} x}{dt^{n-1}}, \frac{d^{n-2} x}{dt^{n-2}}, \ldots, \frac{dx}{dt}, x, t\right)$$

- Essa é a forma mais comum em física, com a **variável independente** t representando o tempo e a **variável dependente** x representando, por exemplo, a posição ou a velocidade de uma partícula.

Equações diferenciais ordinárias

$$\boxed{\frac{d^n x}{dt^n} = f\left(\frac{d^{n-1} x}{dt^{n-1}}, \frac{d^{n-2} x}{dt^{n-2}}, \ldots, \frac{dx}{dt}, x, t\right)}$$

- Exemplos:

$$m\frac{dv}{dt} = -mg - \rho v$$

$$m\frac{d^2 x}{dt^2} + C\left(\frac{d x}{dt}\right)^2 = -k x + F(t)$$

- O que significa “resolver” uma equação diferencial ordinária? O que é preciso conhecer?

O método de Euler

- Vamos nos restringir por enquanto às equações diferenciais ordinárias de primeira ordem:

$$\boxed{\frac{dx}{dt} = f(x,t).}$$

- Se conhecermos o valor de x em um certo instante t, podemos escrever seu valor um curto intervalo de tempo h depois:

$$x(t+h) = x(t) + h\frac{dx}{dt} + O(h^2) = x(t) + hf(x,t) + O(h^2).$$

O método de Euler

- Vamos nos restringir por enquanto às equações diferenciais ordinárias de primeira ordem:

$$\boxed{\frac{dx}{dt}=f(x,t).}$$

- O **método de Euler** consiste em desprezar os termos não lineares em h e simplesmente calcular iterativamente

$$x(t+h)=x(t)+hf(x,t),$$

partindo de uma certa condição inicial

$$x(t_0)=x_0.$$

O método de Euler

- Vamos nos restringir por enquanto às equações diferenciais ordinárias de primeira ordem:

$$\boxed{\frac{dx}{dt}=f(x,t).}$$

- O método de Euler produz portanto uma lista de valores de x em tempos discretos, separados pelo **passo temporal** h:

$$\{x(t_0),x(t_0+h),x(t_0+2h),\cdots,x(t_0+nh),\cdots\}$$

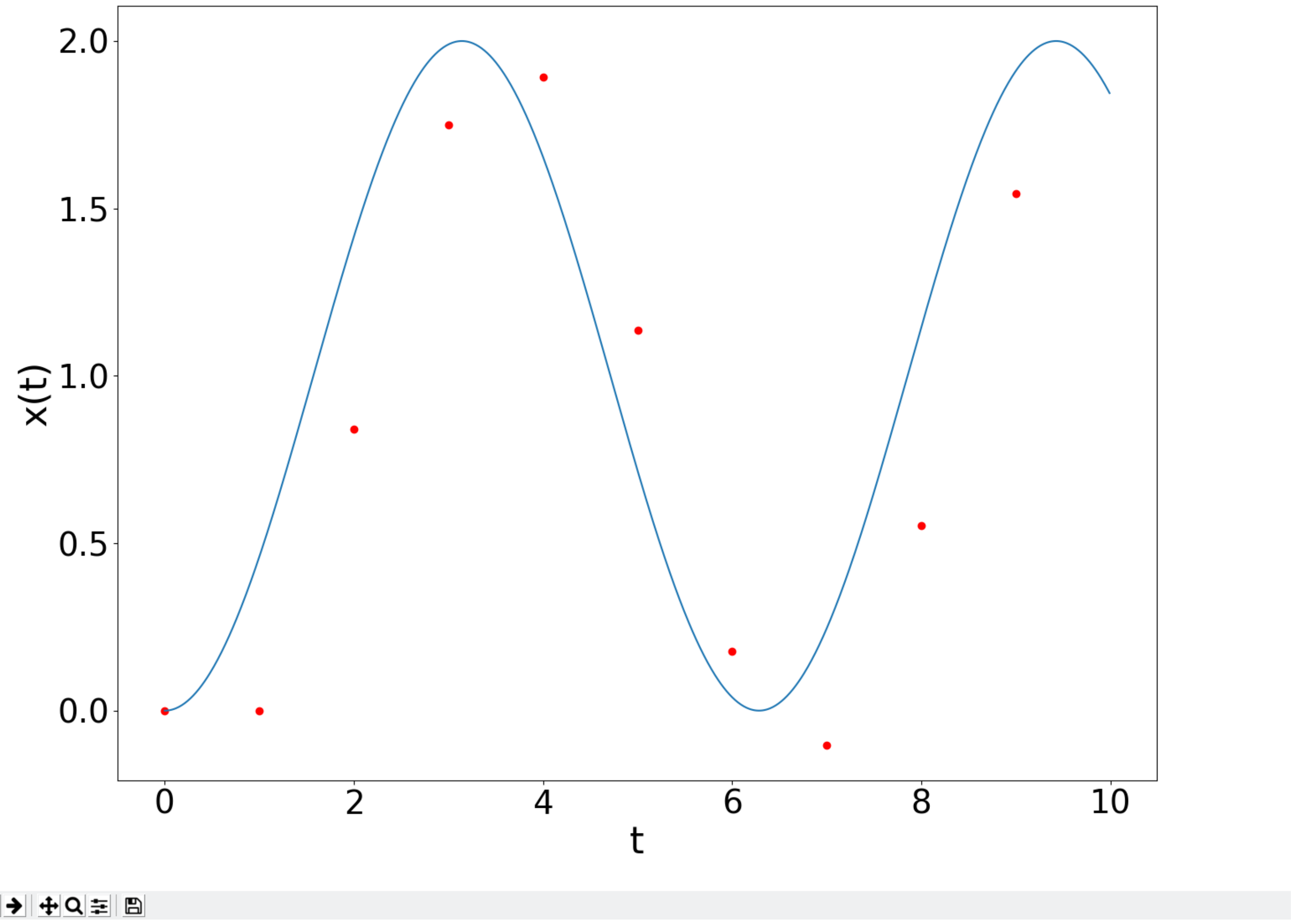
Exemplo 1

$$\frac{dx}{dt} = \text{sen}(t)$$

```python
from math import sin,cos
from numpy import arange
import matplotlib.pyplot as plt

def f(x,t):
    return sin(t)

a = 0.0             # Início do intervalo
b = 10.0            # Final do intervalo
N = 10              # Número de passos da sol. por Euler
h = (b-a)/N         # Tamanho de um passo dessa solução
x = 0.0             # Condição inicial, ou seja, x(a)

N_exato = 1000      # Número de pontos para a sol. exata
h_exato = (b-a)/N_exato

x_exato = []
t_exato = arange(a,b,h_exato)
for t in t_exato:
    x_exato.append(cos(a)+x-cos(t))

t_euler = arange(a,b,h)
x_euler = []
for t in t_euler:
    x_euler.append(x)
    x += h*f(x,t)

plt.plot(t_euler,x_euler,'r.',t_exato,x_exato)
plt.xlabel("t")
plt.ylabel("x(t)")
plt.show()
```

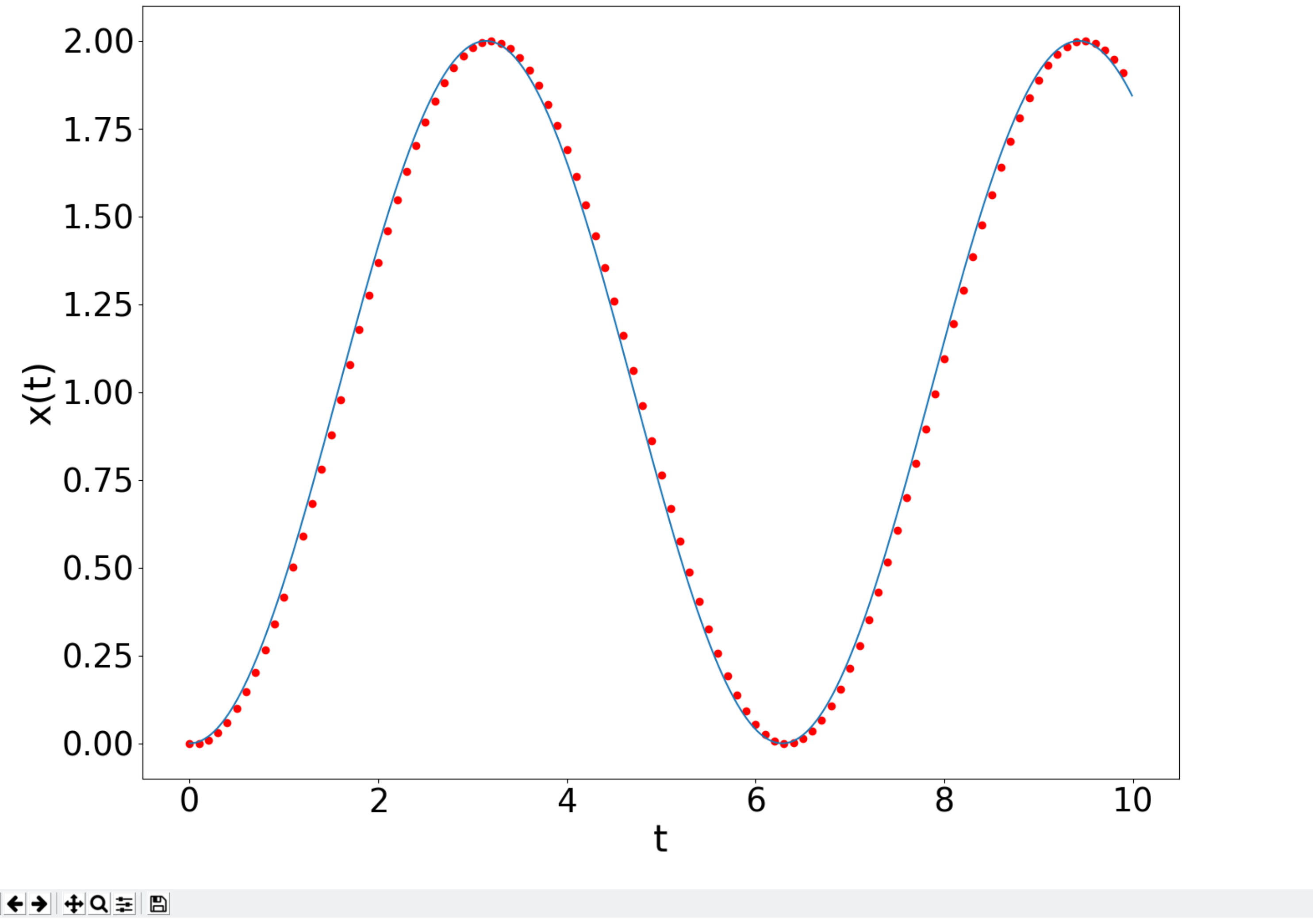
Exemplo 1

N = 10



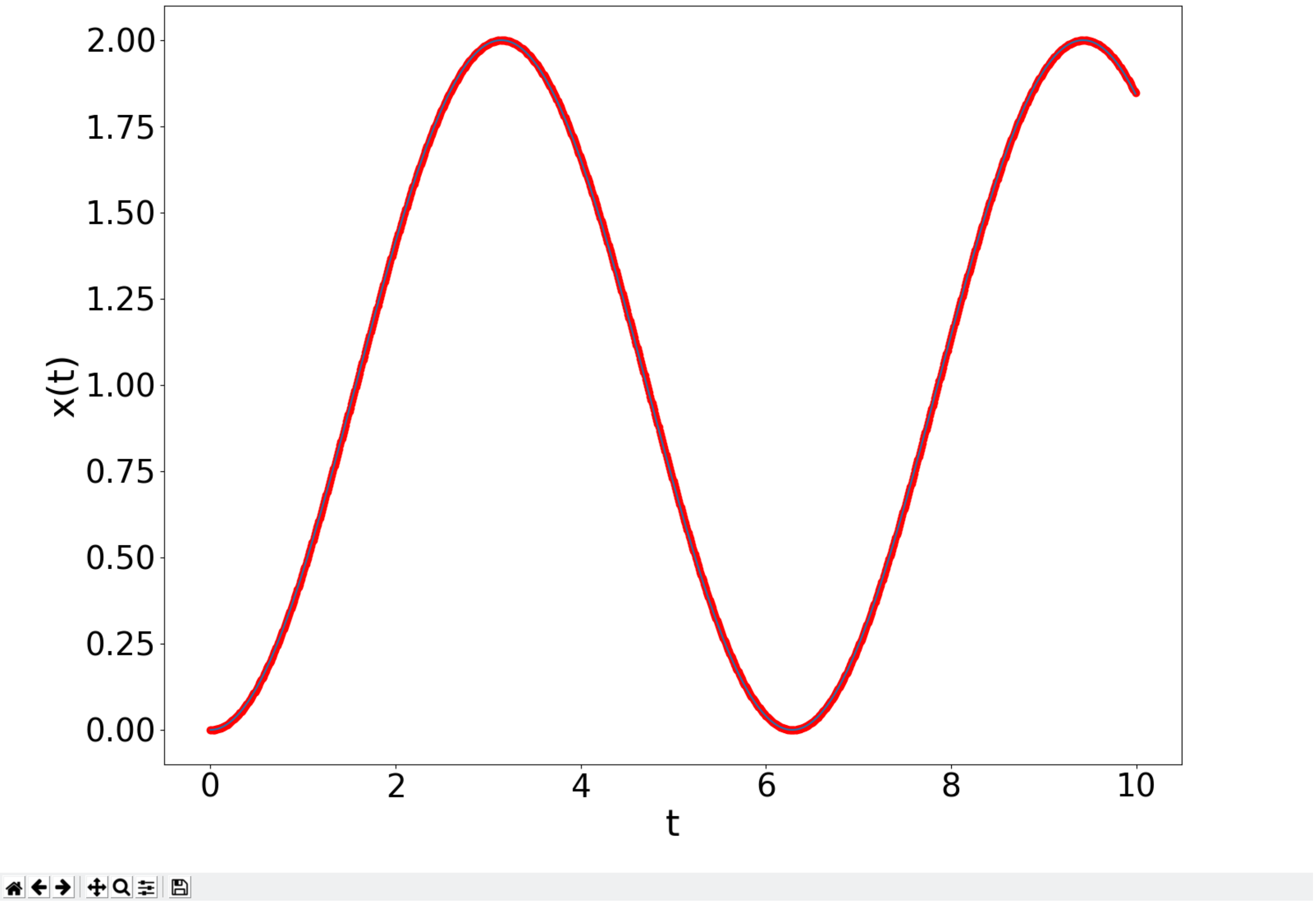
Exemplo 1

N = 100



Exemplo 1

N = 1000



O método de Euler

- O erro associado a cada passo no método de Euler é aproximadamente

$$\frac{1}{2}h^2\frac{d^2x}{dt^2},$$

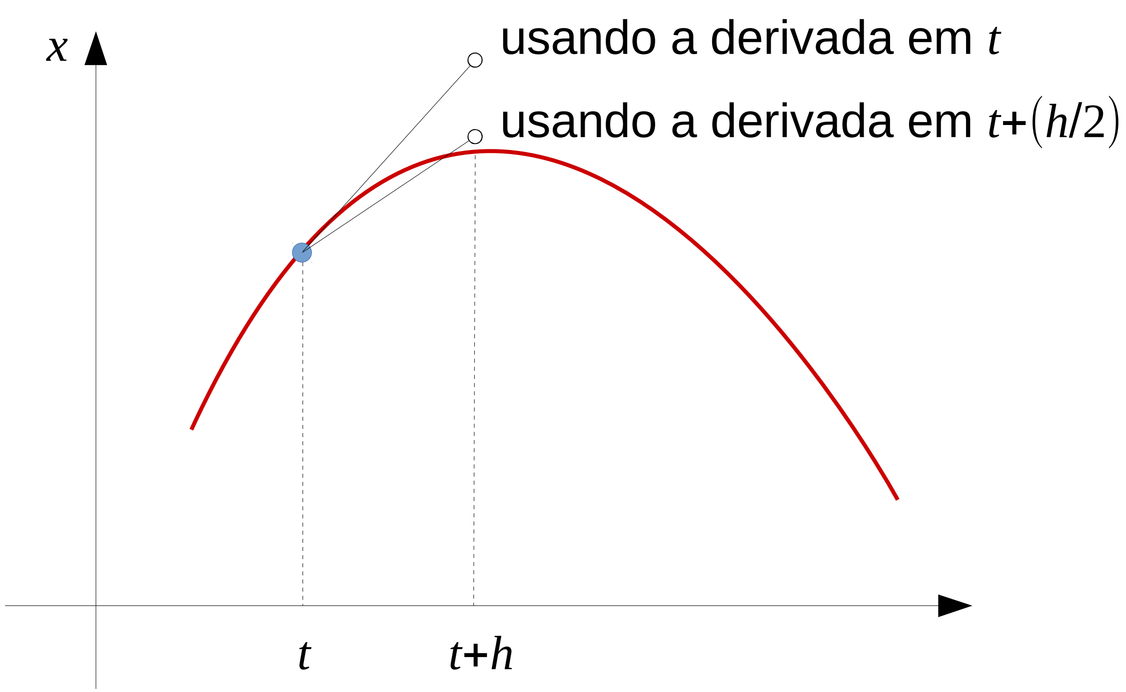
de forma que o erro cumulativo deve ser linear no tamanho do passo:

$$\sum_{k=0}^{N-1}\frac{1}{2}h^2\left(\frac{d^2x}{dt^2}\right)_{t=t_k}=\frac{1}{2}h\sum_{k=0}^{N-1}h\left(\frac{df}{dt}\right)_{t=t_k}\simeq\frac{1}{2}h\int_a^b dt\,\frac{df}{dt}$$

$$=\frac{1}{2}h\left[f\left(x\left(b\right),b\right)-f\left(x\left(a\right),a\right)\right]$$

O método de Runge–Kutta (2ª ordem)

- Enquanto o método de Euler calcula a função em $t+h$ extrapolando a partir da derivada calculada em t, o método de Runge–Kutta de segunda ordem utiliza a derivada calculada no ponto médio entre t e $t+h$.



O método de Runge–Kutta (2ª ordem)

- Queremos

$$k_1 = h f(x_n, t_n),$$
$$k_2 = h f\left(x_n + \tfrac{1}{2} k_1, t_n + \tfrac{1}{2} h\right),$$
$$x_{n+1} = x_n + a k_1 + b k_2,$$

com $t_n \equiv t_0 + nh$, $x_n \equiv x(t_n)$, de tal forma que o erro cometido em um passo seja **<u>cúbico</u>** em h. Mas

$$x_{n+1} = x_n + h\left(\frac{dx}{dt}\right)_{t_n} + \tfrac{1}{2} h^2 \left(\frac{d^2 x}{dt^2}\right)_{t_n} + O(h^3)$$

e

$$\frac{d^2 x}{dt^2} = \frac{df(x,t)}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial t} = f\frac{\partial f}{\partial x} + \frac{\partial f}{\partial t},$$

O método de Runge–Kutta ($2^{\underline{a}}$ ordem)

e assim

$$x_{n+1} = x_n + \left[hf + \tfrac{1}{2}h^2 \left(f\frac{\partial f}{\partial x} + \frac{\partial f}{\partial t} \right) \right](x_n, t_n) + O(h^3),$$

enquanto

$$k_2 = h\left[f + \tfrac{1}{2}k_1\frac{\partial f}{\partial x} + \tfrac{1}{2}h\frac{\partial f}{\partial t} \right](x_n, t_n) + O(h^3).$$

Portanto,

$$x_{n+1} = x_n + ak_1 + bh\left[f + \tfrac{1}{2}k_1\frac{\partial f}{\partial x} + \tfrac{1}{2}h\frac{\partial f}{\partial t} \right](x_n, t_n) + O(h^3).$$

O método de Runge–Kutta (2ª ordem)

$$x_{n+1}=x_n+\left[hf+\tfrac{1}{2}h^2\left(f\frac{\partial f}{\partial x}+\frac{\partial f}{\partial t}\right)\right](x_n,t_n)+O(h^3)$$

$$x_{n+1}=x_n+\left[(a+b)hf+bh^2\left(\tfrac{1}{2}f\frac{\partial f}{\partial x}+\tfrac{1}{2}\frac{\partial f}{\partial t}\right)\right](x_n,t_n)+O(h^3)$$

Comparando as potências de h:

$$a+b=1,\quad b=1\quad\Rightarrow\quad a=0$$

Finalmente

$$\boxed{\begin{gathered}k_1=hf(x_n,t_n)\\ k_2=hf\left(x_n+\tfrac{1}{2}k_1,t_n+\tfrac{1}{2}h\right)\\ x_{n+1}=x_n+k_2\end{gathered}}$$

O método de Runge–Kutta (2ª ordem)

$$x_{n+1}=x_n+\left[hf+\tfrac{1}{2}h^2\left(f\frac{\partial f}{\partial x}+\frac{\partial f}{\partial t}\right)\right](x_n,t_n)+O(h^3)$$

$$x_{n+1}=x_n+\left[(a+b)hf+bh^2\left(\tfrac{1}{2}f\frac{\partial f}{\partial x}+\tfrac{1}{2}\frac{\partial f}{\partial t}\right)\right](x_n,t_n)+O(h^3)$$

Comparando as potências de h:

$$a+b=1,\quad b=1\quad\Rightarrow\quad a=0$$

Finalmente

$$\boxed{\begin{gathered}k_1=hf(x_n,t_n)\\ k_2=hf\left(x_n+\tfrac{1}{2}k_1,t_n+\tfrac{1}{2}h\right)\\ x_{n+1}=x_n+k_2\end{gathered}}$$

O erro em um só passo é cúbico. Mas e quanto ao erro <u>cumulativo</u>?

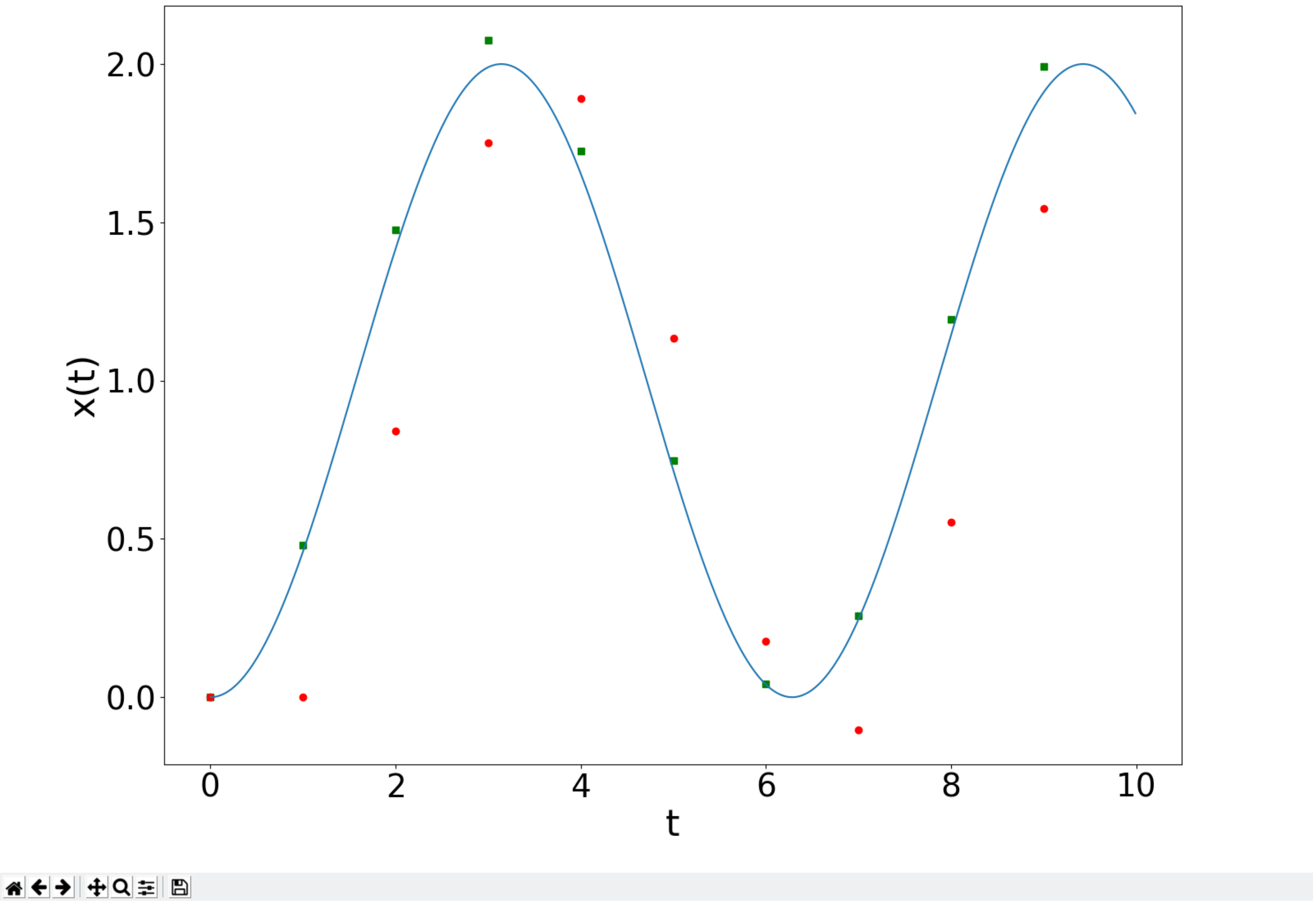
Exemplo 2

$$\frac{dx}{dt} = \mathrm{sen}(t)$$

```
from math import sin,cos
from numpy import arange
import matplotlib.pyplot as plt

def f(x,t):
    return sin(t)

a = 0.0             # Início do intervalo
b = 10.0            # Final do intervalo
N = 10              # Número de passos da sol. por Runge-Kutta
h = (b-a)/N         # Tamanho de um passo dessa solução
x = 0.0             # Condição inicial, ou seja, x(a)

N_exato = 1000      # Número de pontos para a sol. exata
h_exato = (b-a)/N_exato

x_exato = []
t_exato = arange(a,b,h_exato)
for t in t_exato:
    x_exato.append(cos(a)+x-cos(t))

t_rk2 = arange(a,b,h)
x_rk2 = []
for t in t_rk2:
    x_rk2.append(x)
    k1 = h*f(x,t)
    k2 = h*f(x+0.5*k1,t+0.5*h)
    x += k2
```

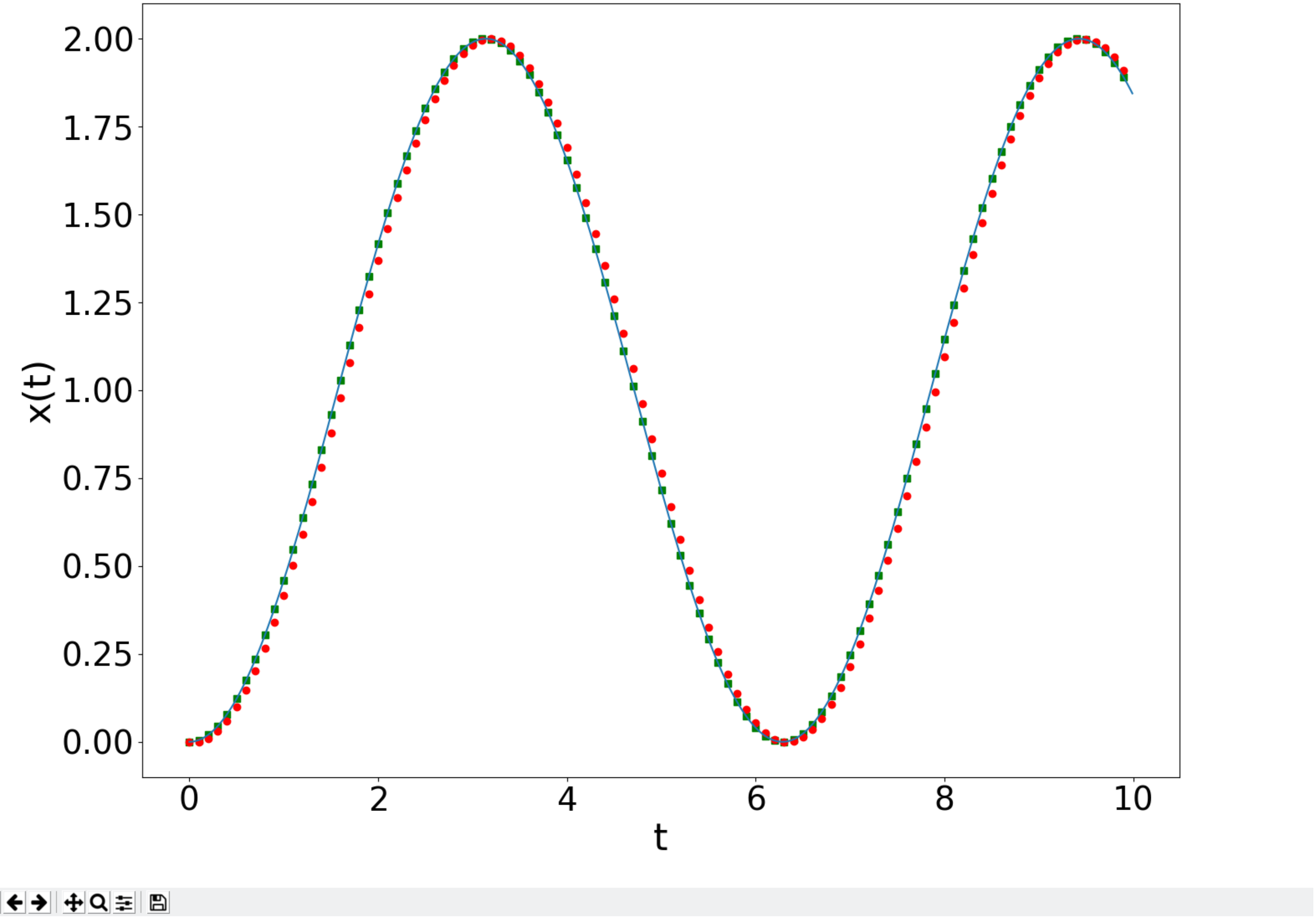
Exemplo 2

N = 10



Exemplo 2

N = 100



Exercício no moodle

O objetivo deste exercício é verificar numericamente as previsões para a dependência dos erros associados aos métodos de Euler e de Runge-Kutta de segunda ordem com o tamanho do passo de integração.

Escreva um programa que integre numericamente a equação diferencial

$$\frac{dx}{dt} = -2\exp(-\gamma t)\left(\gamma\cos t + \operatorname{sen} t\right),$$

no intervalo entre $t = 0$ e $t = 10$, utilizando tanto o método de Euler quanto o método de Runge-Kutta de segunda ordem. Utilize como condição inicial $x(0) = 0$. Além de realizar essas integrações numéricas, seu programa deve calcular o erro médio associado ao cálculo, definido em cada caso como

$$\text{erro} \equiv \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}\left[x_{\text{num}}(t_n) - x_{\text{exato}}(t_n)\right]^2},$$

em que $t_n = nh$, sendo h o passo de integração, enquanto $x_{\text{num}}(t_n)$ e $x_{\text{exato}}(t_n)$ representam, respectivamente, os valores de $x(t)$ calculados pela integração numérica e pela solução exata até $t = t_n$. Aqui, N representa o número máximo de passos de tempo utilizados na integração, que se relaciona ao passo de integração neste caso segundo $N = 10/h$. A solução exata da equação diferencial, como pode ser facilmente verificado por substituição direta ou por integração analítica, é

$$x_{\text{exato}}(t) = x(0) - 2 + 2\exp(-\gamma t)\cos t.$$

Em seu programa, crie listas para armazenar os valores dos erros associados aos dois métodos para $N \in \{8, 16, 32, 64, 128, 256, 512, 1024\}$, e as utilize para fazer um gráfico log-log que mostre a dependência desses erros com N (e, consequentemente, com o tamanho do passo de integração h). Você deve obter dependências do tipo

$$\text{erro} \propto h^{\alpha} \propto N^{-\alpha},$$

com valores diferentes do expoente α para os dois métodos. Teste seu programa para os valores $\gamma = 0$ e $\gamma = 1$. Os valores do expoente α dependem de γ?