AULA 2

Números e operações. Erros de arredondamento e de truncatura. Gráficos simples. Cópia de objetos.

Números inteiros

Plain int: 32 bits, representa os números inteiros:

$$-2^{31} < N < +2^{31} - 1$$

A representação é **exata**, dentro do **alcance** representado.

Números float

$$x = m \times 2^{e-E}$$

m mantissa (**algarismos sinificativos**)

e expoente, E viés (**alcance**)

$$m, e, E \in \mathbb{Z}$$

Os números ***float*** representam um sub-conjunto dos racionais

Operações aritméticas

Operações elementares: `+ - * /`

Potência: `**`

Divisão inteira: `//`

As operações envolvem objetos (números) que podem ser de **tipo** diferente. O tipo de resultado depende do tipo dos números operados. Em geral o python dá o resultado no tipo de maior rank:

```
X=10 #inteiro
Y=2.5 #float
Z=X*Y #float
```

BigInt (só para inteiros escalares)

In[102]: x=2**128

In[103]: x

Out[104]: 340282366920938463463374607431768211456

(inteiro de 128 bit)

Y=2**1280

(inteiro de 1280 bit)

Mas:

z=2.0**1280 #OVERFLOW error

a=np.array([10]);b=a**10000 #OVERFLOW error

Erros

Overflow: fora do alcance

Erro de arredondamento:

In[116]: 1/3

Out[117]: 0.3333333333333333

Erro de truncatura: depende das aproximações do Código.

Instabilidade numérica resulta da interação entre o erro de arredondamento e o erro de truncatura.

Resolver $y = ax^2 + bx + c$

Parece fácil:

$$y = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Mas se $b^2 \gg 4ac$ há problemas…

$$-y^2 + 88550000y - 1 = 0$$

Solução (8 algarismos):

$$\begin{cases} y_1 = 88550000 \\ y_2 = 1.1293055 \times 10^{-8} \end{cases}$$

$$-y^2 + 88550000y - 1 = 0$$

```
from math import sqrt
a=-1.;b=88550000.;c=-1.;
y1=(-b-sqrt(b*b-4*a*c))/(2*a)
y2=(-b+sqrt(b*b-4*a*c))/(2*a)
print('y1=',y1,'y2=',y2)
```

```
y2= 88549999.99999999
y1= 1.4901161193847656e-08
```

$$\begin{cases} y_1 = 88550000 \\ y_2 = 1.1293055 \times 10^{-8} \end{cases}$$

Erro de 30%!

Fórmula alternativa

$$y = \frac{2c}{-b \mp \sqrt{b^2 - 4ac}}$$

```
y1a=2*c/(b+sqrt(b*b-4*a*c))
y2a=2*c/(b-sqrt(b*b-4*a*c))
print('y1=',y1a,'y2=',y2a)
```

```
y1= 67108864.0
y2= 1.1293054771315643e-08
```

$$\begin{cases} y_1 = 88550000 \\ y_2 = 1.1293055 \times 10^{-8} \end{cases}$$

Erro na outra raíz

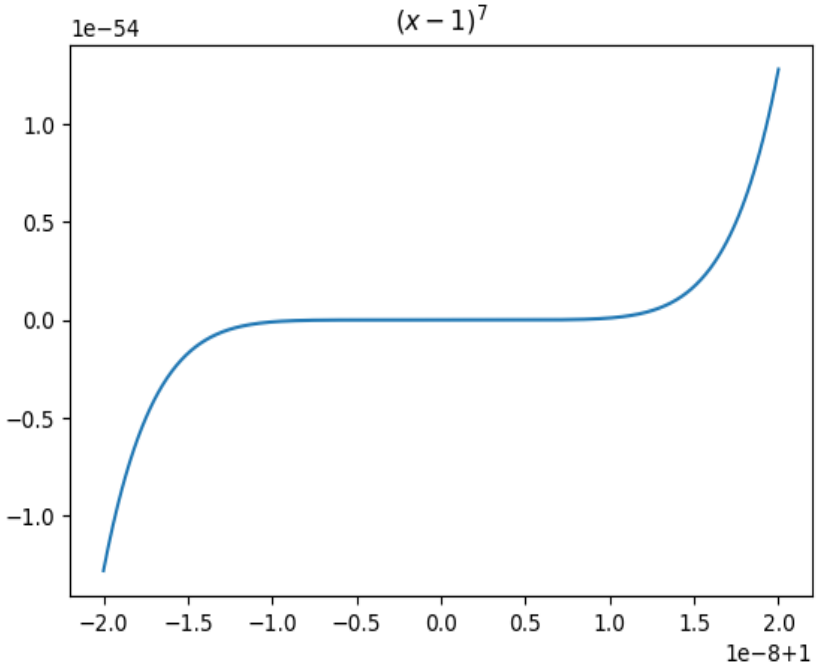
Solução exata

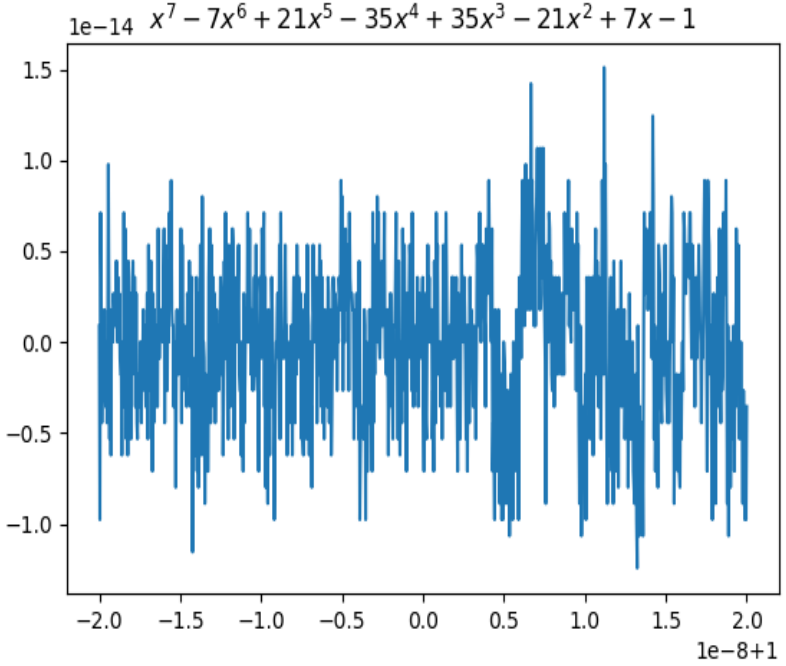
```
from sympy import Symbol,solve
y=Symbol('y')
solution=solve(a*y**2+b*y+c,y)
print('sympy solution=',solution)
```

```
sympy solution= [1.12930547713156e-8,
88550000.0000000]
```

Mais erros…

$$(x-1)^7 = x^7 - 7x^6 + 21x^5 - 35x^4 + 35x^3 - 21x^2 + 7x - 1)$$





Erro de 40 ordens de grandeza!

Gráficos e anotações

```
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(0.99999998,1.00000002,1000)
f1=(x-1)**(7)
f2=x**7-7*x**6+21*x**5-35*x**4+35*x**3-21*x**2+7*x-1

plt.close('all')
plt.subplot(2,1,1)
plt.plot(x,f1)
plt.title(r"$(x-1)^{7}$")
plt.subplot(2,1,2)
plt.plot(x,f2)
plt.title(r"$x^{7}-7x^{6}+21x^{5}-35x^{4}+35x^{3}-
21x^{2}+7x-1$")
```

a=b *vs* a=copy(b); list *vs* array

```
import numpy as np
import copy
a=[1,2,3];b=[1.,2.,3.]
d=copy.copy(a)
x=a;x[0]=10;d[0]=100
y=b*2 #b é uma list
c=np.array(b);z=c*2 #c é um array
print('a=',a,'b=',b,'c=',c,'d=',d,'\n','x=',x,'y=',y
,'z=',z)

>>a= [10, 2, 3] b= [1.0, 2.0, 3.0] c= [ 1.  2.  3.]
d= [100, 2, 3]
>>x= [10, 2, 3] y= [1.0, 2.0, 3.0, 1.0, 2.0, 3.0]
z= [ 2.  4.  6.]
```