Introdução à programação para ciência e engenharia em *Python*

Iuri Soter Viana Segtovich

<u>Parte 3: python científico</u>

Python científico

NumPy
Base N-dimensional array package

SciPy library
Fundamental library for scientific computing

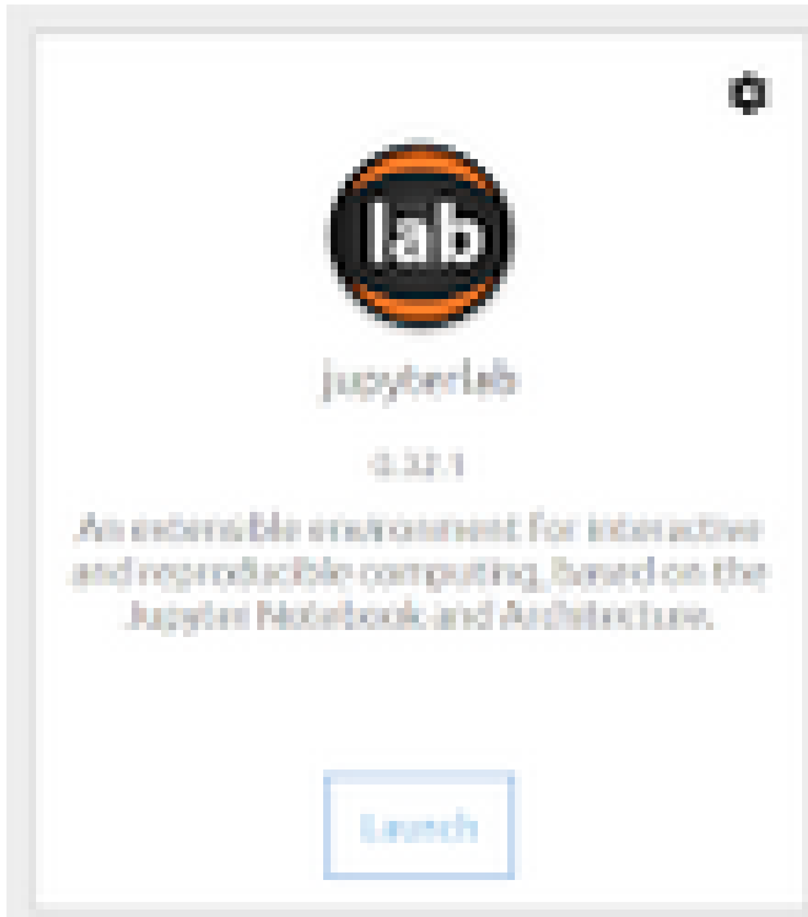
Matplotlib
Comprehensive 2D Plotting

IPython
Enhanced Interactive Console

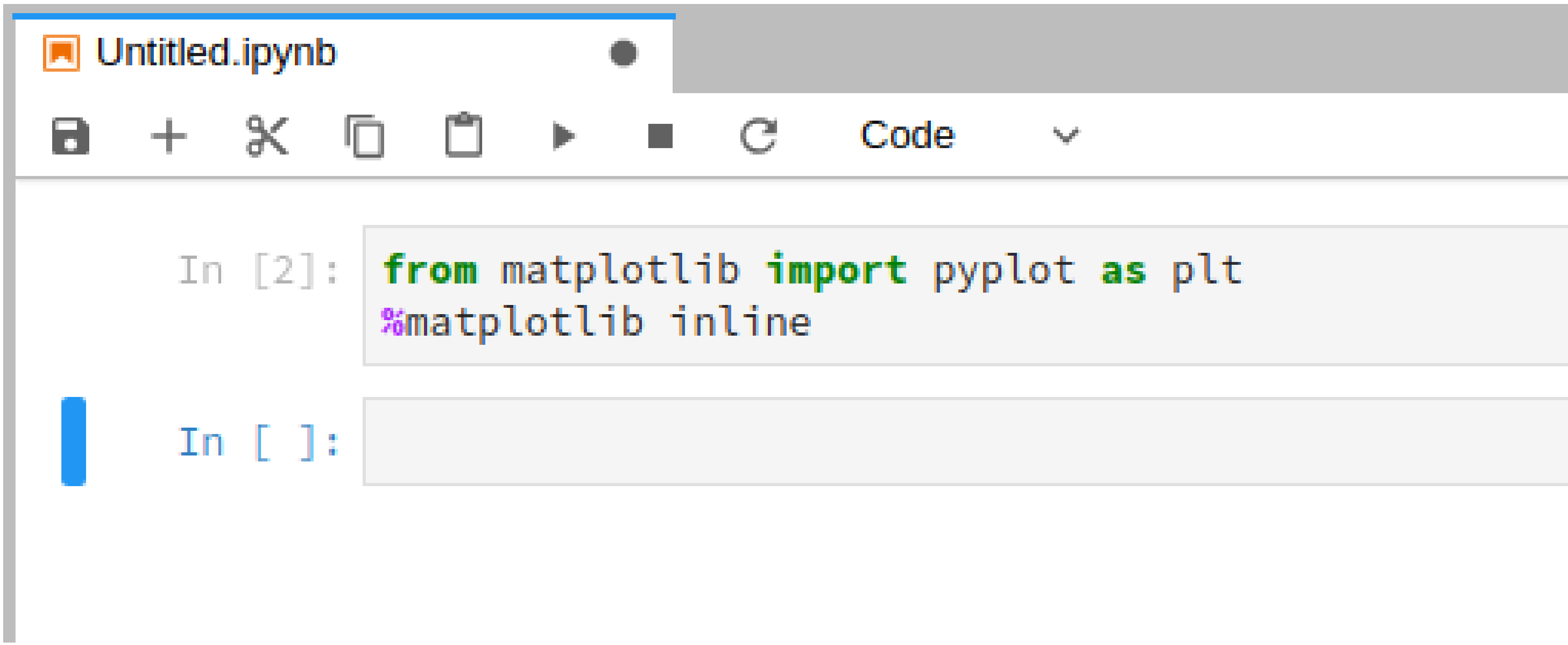
Sympy
Symbolic mathematics

pandas
Data structures & analysis

Anaconda navigator / Jupyter lab



matplotlib

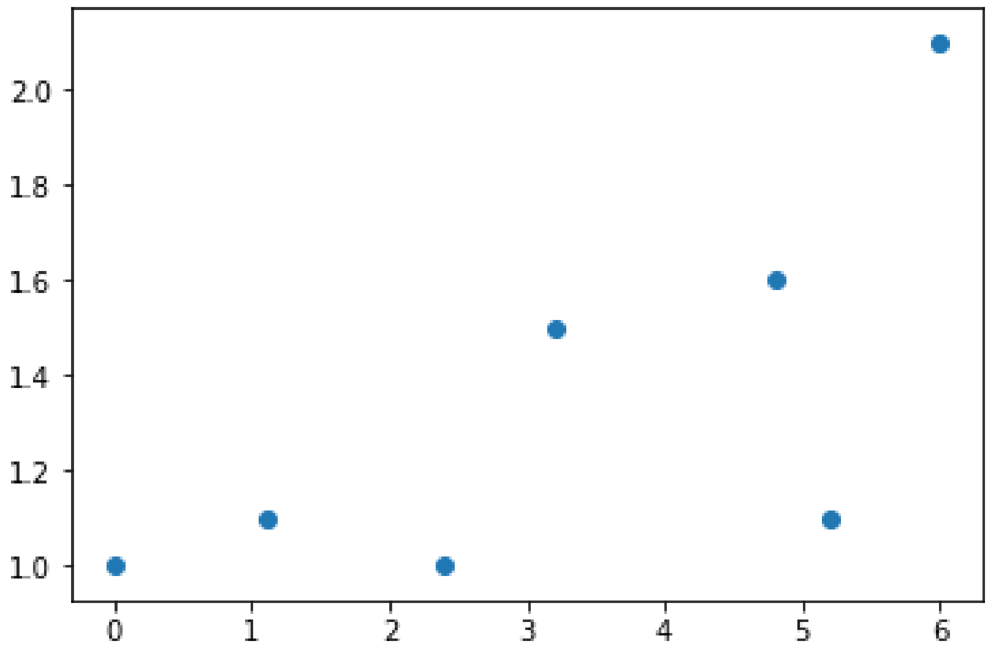

Untitled.ipynb
Code
In [2]:
from matplotlib import pyplot as plt
%matplotlib inline
In [ ]:

scatter

In [8]:
```
T= [0.0, 1.1, 2.4, 3.2, 4.8, 5.2, 6.0]
P= [1.0, 1.1, 1.0, 1.5, 1.6, 1.1, 2.1]
```

In [9]:
```
plt.scatter(T,P)
```

Out[9]: <matplotlib.collections.PathCollection at 0x7fc03364dda0>

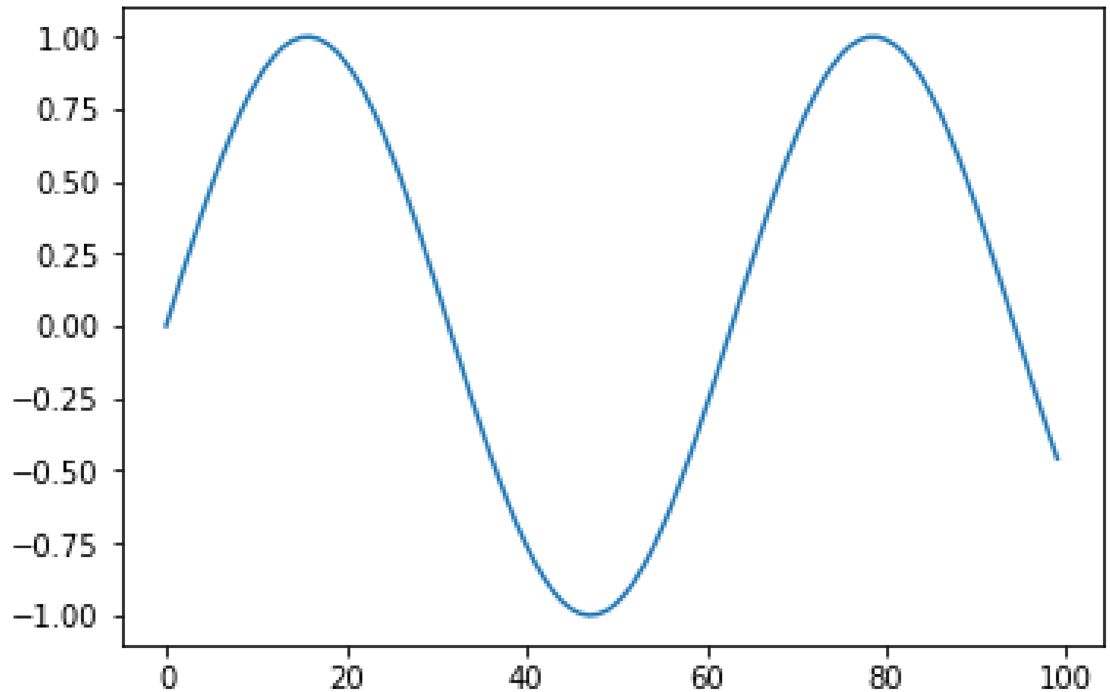


plot

In [18]:
```
x=list(range(100))
from math import sin
y=[sin(xi/10) for xi in x]
```

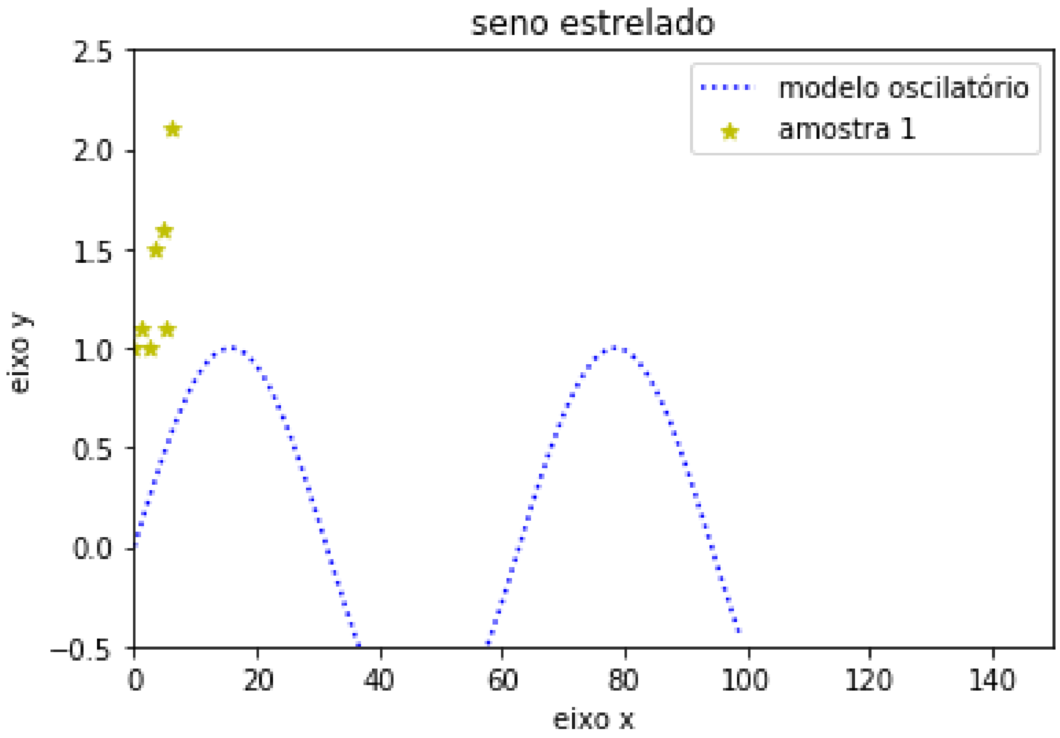
In [20]:
```
plt.plot(x,y)
```

Out[20]:
```
[<matplotlib.lines.Line2D at 0x7fc033411400>]
```



savefig

```
In [29]: plt.scatter(T,P,marker='*',color='y',label='amostra 1')
         plt.plot(x,y,ls=':',color='b', label='modelo oscilatório')
         plt.title("seno estrelado")
         plt.ylabel("eixo y")
         plt.xlabel("eixo x")
         plt.xlim(0,150)
         plt.ylim(-.5,2.5)
         plt.legend()
         plt.savefig("Figura1.png")
```

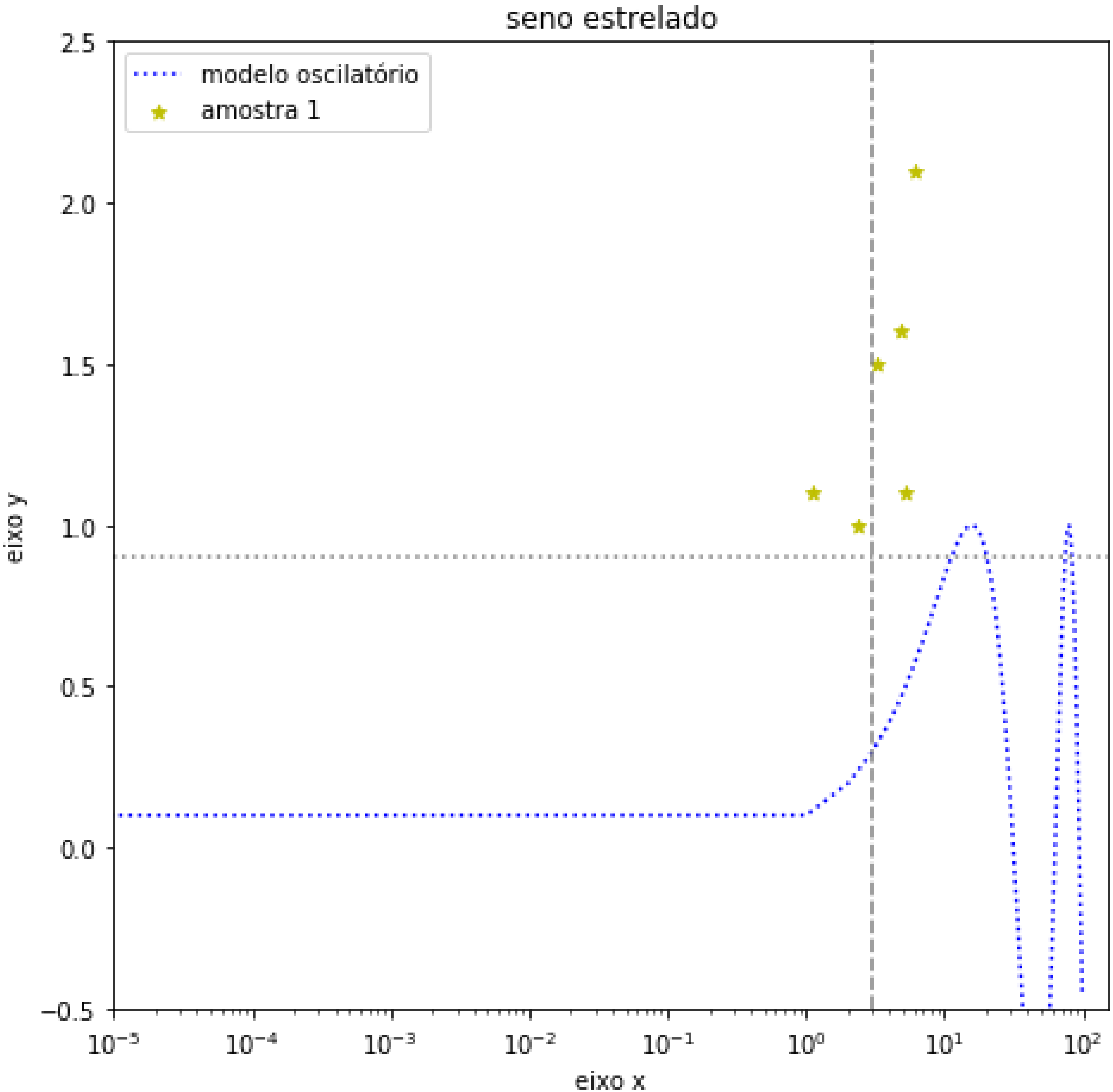


Multi-plot-Fu

https://static1.squarespace.com/static/530562f9e4b06fd3c041e221/t/5956de46099c01c37e5fa0b3/1498865226277/MatplotlibHotTopics.pdf

O caminho OOP

```
in_to_cm=0.393701 #0.393701 polegadas por centimetro
fig=plt.figure(figsize=(19*in_to_cm,19*in_to_cm)) #tamanho em polegadas
ax=fig.add_subplot(111)
ax.scatter(T,P,marker='*',color='y',label='amostra 1')
ax.plot(x,y,ls=':',color='b', label='modelo oscilatório')
ax.set_title("seno estrelado")
ax.set_ylabel("eixo y")
ax.set_xlabel("eixo x")
ax.set_xlim(1e-5,150)
ax.set_ylim(-.5,2.5)
ax.axhline(.9,color='gray',ls=':')
ax.axvline(3,color='grey',ls='--')
ax.set_xscale("log")
ax.legend(loc=2)
fig.savefig("Figura2.png")
fig.canvas.draw()
```



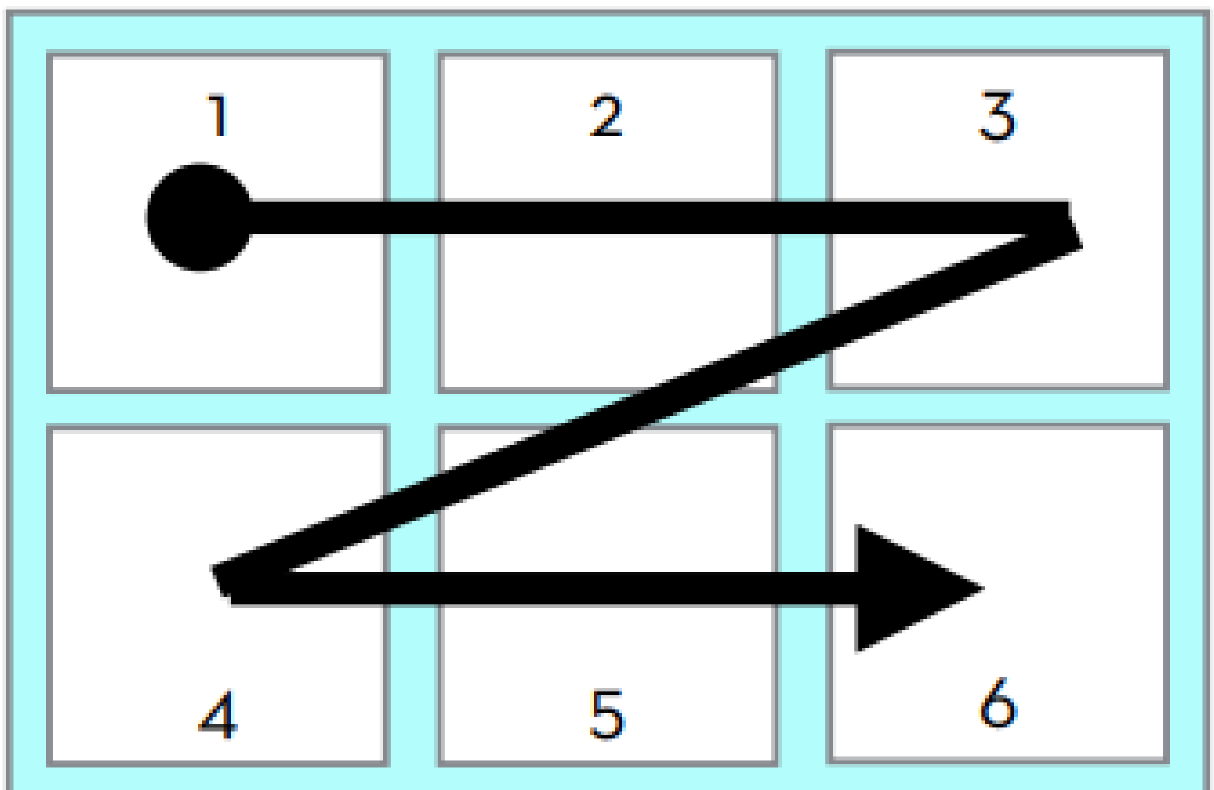
seno estrelado
modelo oscilatório
amostra 1
eixo y
eixo x
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}

```
fig = plt.figure()
ax = fig.add_subplot(111)
```

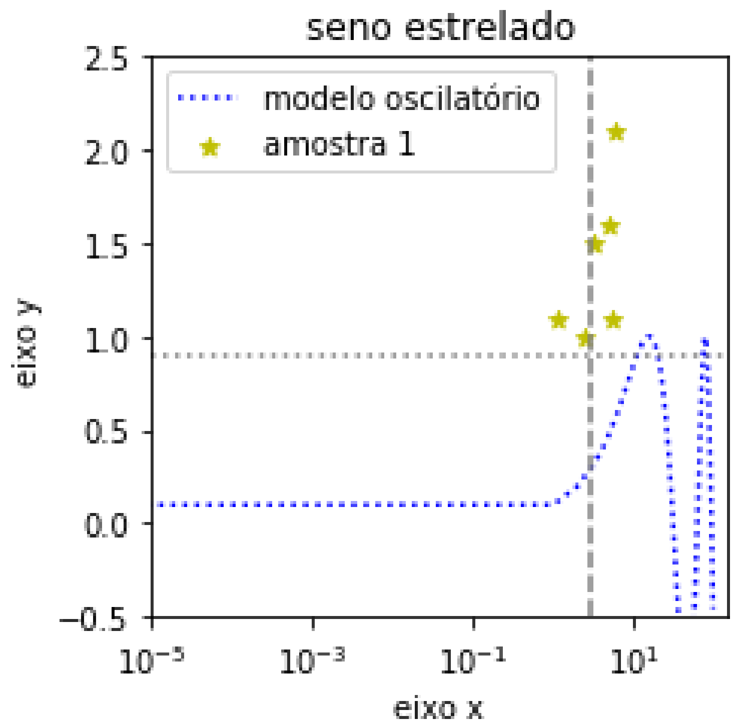
Number of Columns

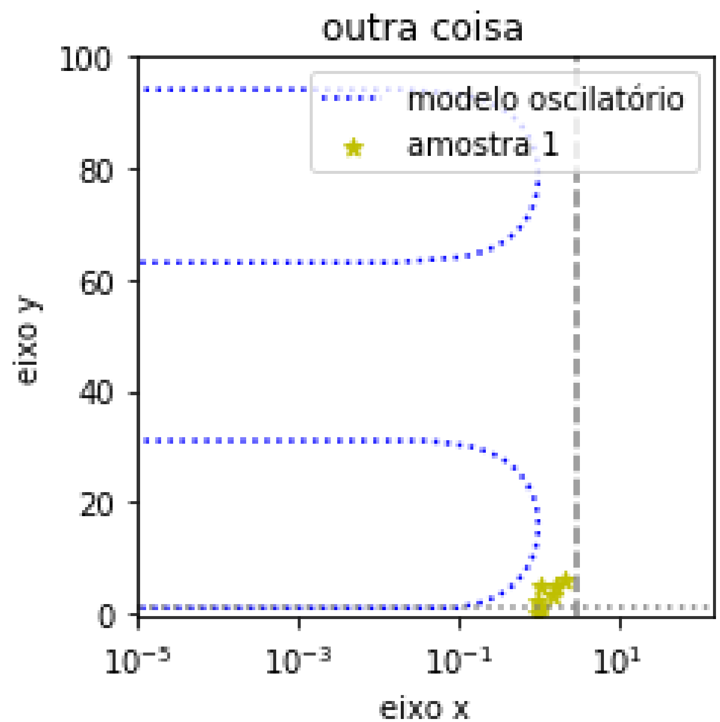
Number of Rows

Axes index in that grid



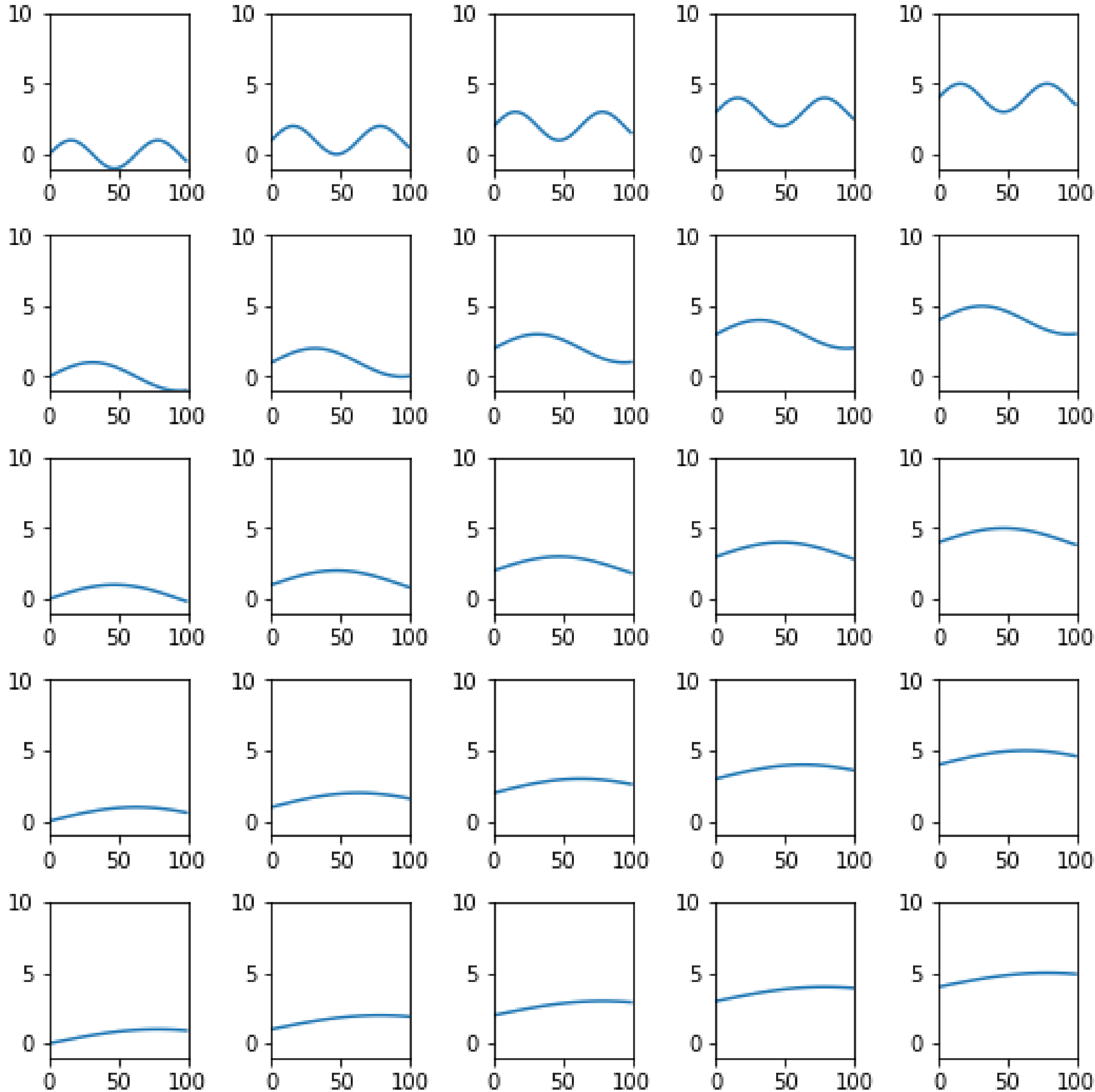
```python
in_to_cm=0.393701 #0.393701 polegadas por centimetro
fig=plt.figure(figsize=(19*in_to_cm,19*in_to_cm)) #tamanho em polegadas
ax_seno_estrelado=fig.add_subplot(221)
for ax in (ax_seno_estrelado,):
    ax.scatter(T,P,marker='*',color='y',label='amostra 1')
    ax.plot(x,y,ls=':',color='b', label='modelo oscilatório')
    ax.set_title("seno estrelado")
    ax.set_ylabel("eixo y")
    ax.set_xlabel("eixo x")
    ax.set_xlim(1e-5,150)
    ax.set_ylim(-.5,2.5)
    ax.axhline(.9,color='gray',ls=':')
    ax.axvline(3,color='grey',ls='--')
    ax.set_xscale("log")
    ax.legend(loc=2)
ax_outra_coisa=fig.add_subplot(224)
for ax in (ax_outra_coisa,):
    ax.scatter(P,T,marker='*',color='y',label='amostra 1')
    ax.plot(y,x,ls=':',color='b', label='modelo oscilatório')
    ax.set_title("outra coisa")
    ax.set_ylabel("eixo y")
    ax.set_xlabel("eixo x")
    ax.set_xlim(1e-5,150)
    ax.set_ylim(-.5,2.5)
    ax.axhline(.9,color='gray',ls=':')
    ax.axvline(3,color='grey',ls='--')
    ax.set_xscale("log")
    ax.legend(loc=2)
fig.savefig("Figura3.png")
fig.canvas.draw()
```


seno estrelado
modelo oscilatório
amostra 1
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
eixo y
10^{-5}
10^{-3}
10^{-1}
10^{1}
eixo x


outra coisa
modelo oscilatório
amostra 1
100
80
60
40
20
0
eixo y
10^{-5}
10^{-3}
10^{-1}
10^{1}
eixo x

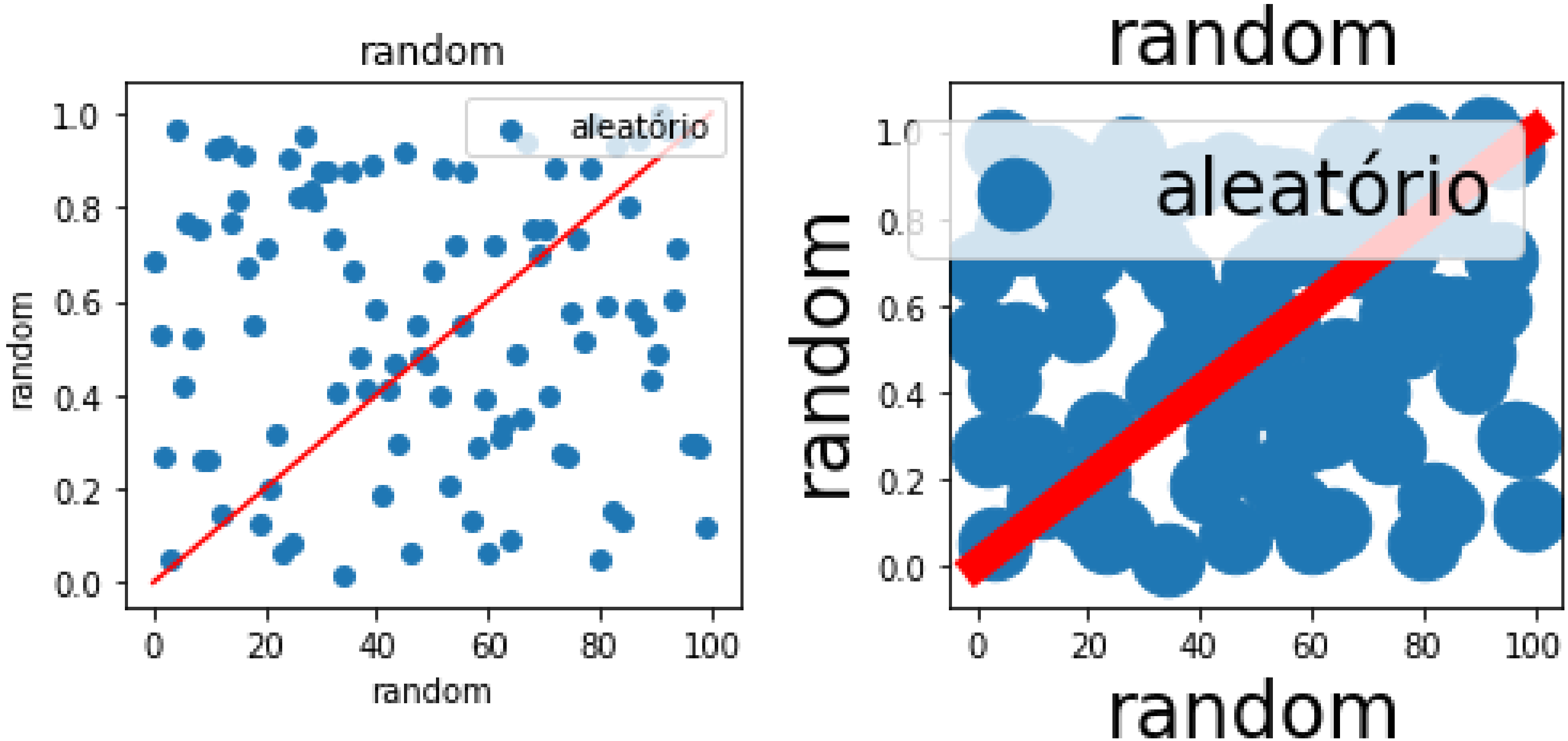
```python
T= [0.0, 1.1, 2.4, 3.2, 4.8, 5.2, 6.0]
P= [1.0, 1.1, 1.0, 1.5, 1.6, 1.1, 2.1]
x=list(range(100))

nx=5
ny=5
y=[[None for i in range(nx)] for j in range(ny)]
ax_varios=fig.subplots(nx,ny)
for i in range(nx):
    for j in range (ny):
        y[i][j]=[j+sin(xi/10/(i+1)) for xi in x]
```

```python
in_to_cm=0.393701 #0.393701 polegadas por centimetro
fig=plt.figure(figsize=(19*in_to_cm,19*in_to_cm)) #tamanho em polegadas
nx=5
ny=5
ax_varios=fig.subplots(nx,ny)
for i in range(nx):
    for j in range (ny):
        ax=ax_varios[i][j]
        ax.plot(x,y[i][j])
        ax.set_xlim(0,100)
        ax.set_ylim(-1,10)
fig.tight_layout()
fig.savefig("Figura4.png")
fig.canvas.draw()
```


10
5
0
0 50 100

fontsize, labelsize

```
in_to_cm=0.393701 #0.393701 polegadas por centimetro
fig=plt.figure(figsize=(18*in_to_cm,9*in_to_cm)) #tamanho em polegadas
ax1=fig.add_subplot(121)
ax1.scatter(xr,yr,label='aleatório')
ax1.plot([0,100],[0,1],color='r')
ax1.set_title("random")
ax1.set_xlabel("random")
ax1.set_ylabel("random")
ax1.legend(loc=1)
ax2=fig.add_subplot(122)
ax2.scatter(xr,yr,label='aleatório',s=500)
ax2.plot([0,100],[0,1],lw=10,color='r')
ax2.set_title("random",fontsize=24)
ax2.set_xlabel("random",fontsize=24)
ax2.set_ylabel("random",fontsize=24)
ax2.legend(loc=1,fontsize=24)
fig.tight_layout()
fig.savefig("Figura5.png")
fig.canvas.draw()
```



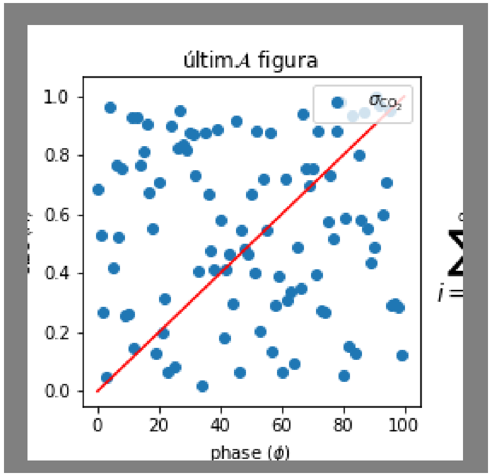
random
aleatório
random
random
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100
random
aleatório
random
random

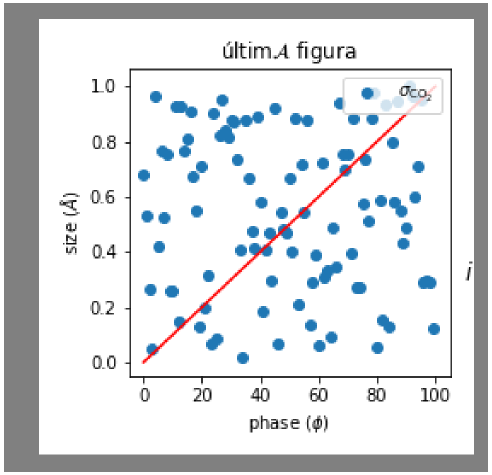
```python
in_to_cm=0.393701 #0.393701 polegadas por centimetro

fig1=plt.figure(figsize=(9*in_to_cm,9*in_to_cm)) #tamanho em polegadas
ax1=fig1.add_subplot(111)
ax1.scatter(xr,yr,label=r'$\sigma_\mathrm{CO_2}$')
ax1.plot([0,100],[0,1],color='r')
ax1.set_title(r'$\alpha_i > \beta_i$', fontsize=20)
ax1.text(110., 0.4, r'$\sum_{i=0}^\infty x_i+ \times \mathrm{sin}(2 \omega t)$',fontsize=20)
ax1.set_xlabel(r'phase $(\phi)$')
ax1.set_ylabel(r'size $(\AA)$')
ax1.set_title(r"últim$\mathcal{A}$ figura")
ax1.legend(loc=1)
fig1.savefig("Figura6")
fig1.canvas.draw()
```

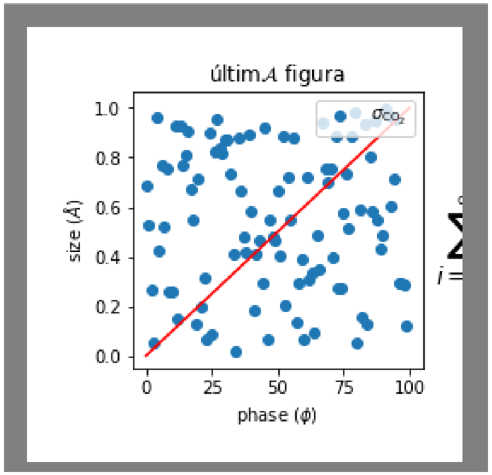
```python
fig1.tight_layout()
fig1.savefig("Figura6_tight_layout")
```

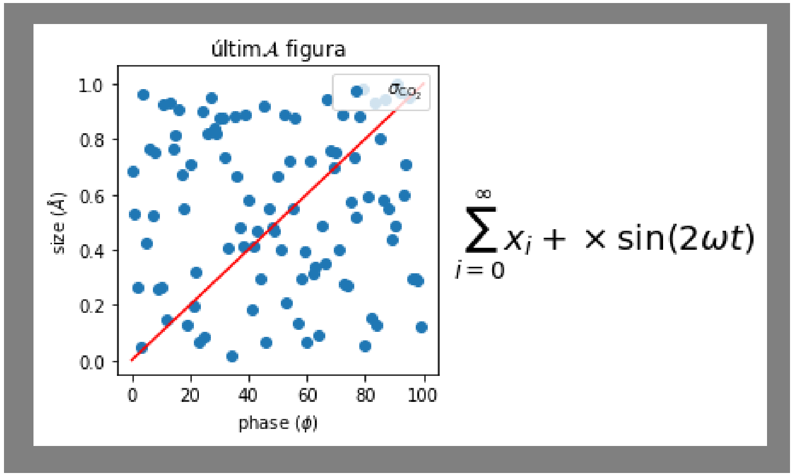
```python
fig1.savefig("Figura6_bbox_inches_tight.png",bbox_inches='tight')
```

```python
fig1.tight_layout(pad=2)
fig1.savefig("Figura6_pad_2.png")
```


últim$\mathcal{A}$ figura
σ_{CO_2}
phase (ϕ)


últim$\mathcal{A}$ figura
σ_{CO_2}
size ($\AA$)
phase (ϕ)


últim$\mathcal{A}$ figura
σ_{CO_2}
size ($\AA$)
phase (ϕ)


últim$\mathcal{A}$ figura
σ_{CO_2}
size ($\AA$)
phase (ϕ)
$\sum_{i=0}^{\infty} x_i + \times \sin(2\omega t)$

numpy

```
import numpy as np
```

numpy

```
import numpy as np
```

```
x=np.array([[11.,12.],[21.,22.])
x=np.array([[11.,12.],[21.,22.])
```

```
print(x)
print(y)
```

```
[ 1.  2.  3.]
[ 4.  5.  6.]
```

```
print(x+y)
print(x*y)
```

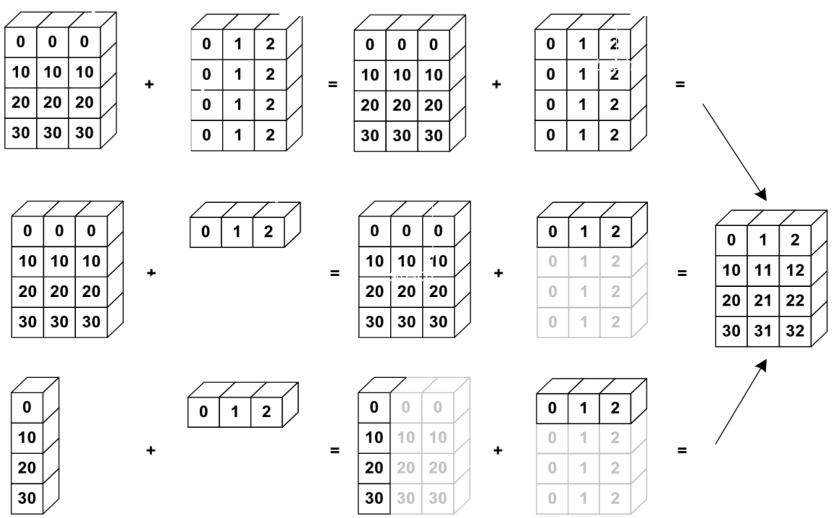
```
[ 5.  7.  9.]
[  4.  10.  18.]
```

```
print(1.*4.+2.*5.+3.*6.)
np.dot(x,y)
```

```
32.0

32.0
```

- Arrays
  - *arranjos de dados*
- Operações termo-a-termo

broadcasting



```
In [18]: A=np.array([[11.,12.],
                     [21.,22.]])
         B=np.array([[33.,44.],
                     [55.,66.]])
         x=np.array([[1.,2.]])
         y=np.array([[4.],
                     [5.]])
```

```
In [19]: print(A*B)

[[  363.   528.]
 [ 1155.  1452.]]
```

```
In [24]: print(A*x)

[[ 11.  24.]
 [ 21.  44.]]
```

```
In [26]: print(A*y)

[[  44.   48.]
 [ 105.  110.]]
```

```
In [27]: print(x*y)

[[  4.   8.]
 [  5.  10.]]
```

```
In [36]: print(x@A)
         print(A@y)

[[ 53.  56.]]
[[ 104.]
 [ 194.]]
```

numpy

- slice

```
In [66]: import numpy as np
```

```
In [67]: l1=[1,2,3]
         l2=l1[1:3]

         a1=np.array([1.,2.,3.])
         a2=a1[1:3]
```

```
In [68]: print(l1)
         print(l2)
         print(a1)
         print(a2)
```

```
[1, 2, 3]
[2, 3]
[ 1.  2.  3.]
[ 2.  3.]
```

```
In [69]: for i in range(3):
             l1[i]+=10
         print(l1)
         print(l2)
```

```
[11, 12, 13]
[2, 3]
```

```
In [70]: for i in range(3):
             a1[i]+=10
         print(a1)
         print(a2)
```

```
[ 11.  12.  13.]
[ 12.  13.]
```

numpy

In [3]:
```
import numpy as np
```

In [6]:
```
x=np.array([[11.,12.],[21.,22.]])
y=np.array([[33.,44.],[55.,66.]])
```

In [7]:
```
print(x)
print(y)
```

```
[[ 11.  12.]
 [ 21.  22.]]
[[ 33.  44.]
 [ 55.  66.]]
```

In [14]:
```
print(x)
print(x.T)
```

```
[[ 11.  12.]
 [ 21.  22.]]
[[ 11.  21.]
 [ 12.  22.]]
```

In [13]:
```
print(x*y)
print(np.matmul(x,y))
print(x@y)
```

```
[[  363.   528.]
 [ 1155.  1452.]]
[[ 1023.  1276.]
 [ 1903.  2376.]]
[[ 1023.  1276.]
 [ 1903.  2376.]]
```

- 2d array
- Matrix ops

linalg

Problema

$x + y = 1$

$x - 2y = 2$

Solução

$3y = -1$

$y = -\frac{1}{3}$

$x = 1 + \frac{1}{3}$

Matricialmente

$Av = b$

$v = A^{-1}b$

In [17]:
```
import numpy as np
A=np.array([[1.,1.],[1.,-2.]])
b=np.array([[1.,2.]]).T
```

In [18]:
```
print(b) #em pé
```
```
[[ 1.]
 [ 2.]]
```

In [19]:
```
A**-1
```

Out[19]:
```
array([[ 1. ,  1. ],
       [ 1. , -0.5]])
```

In [20]:
```
from numpy import linalg as la
Ainv=la.inv(A)
```

In [21]:
```
Ainv@A
```

Out[21]:
```
array([[  1.00000000e+00,   1.11022302e-16],
       [  0.00000000e+00,   1.00000000e+00]])
```

In [23]:
```
v=Ainv@b
print(v)
```
```
[[ 1.33333333]
 [-0.33333333]]
```

In [24]:
```
v=la.solve(A,b)
print(v)
```
```
[[ 1.33333333]
 [-0.33333333]]
```

numpy

In [22]:
```
import numpy as np
```

In [23]:
```
x=np.linspace(0,10,10)
xx=np.linspace(0,10,100)
```

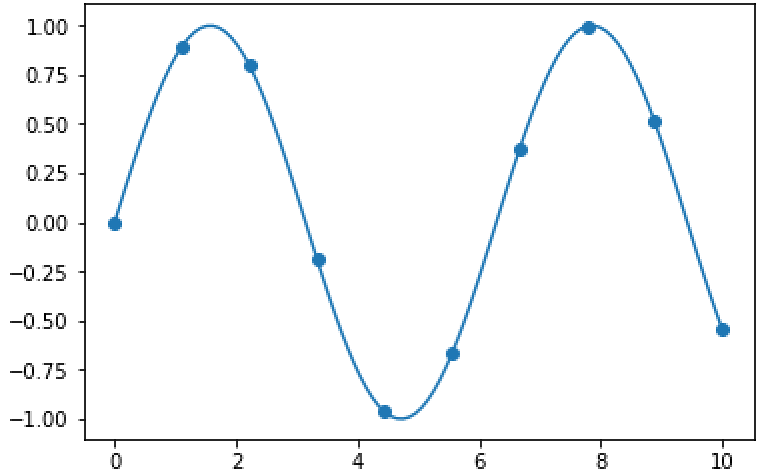
In [24]:
```
print(x)
```

```
[ 0.          1.11111111  2.22222222  3.33333333  4.44444444
  5.55555556  6.66666667  7.77777778  8.88888889 10.        ]
```

In [25]:
```
y=np.sin(x)
yy=np.sin(xx)
```

In [26]:
```
plt.scatter(x,y)
plt.plot(xx,yy)
```

Out[26]: [<matplotlib.lines.Line2D at 0x7f7c6f2819b0>]



- linspace

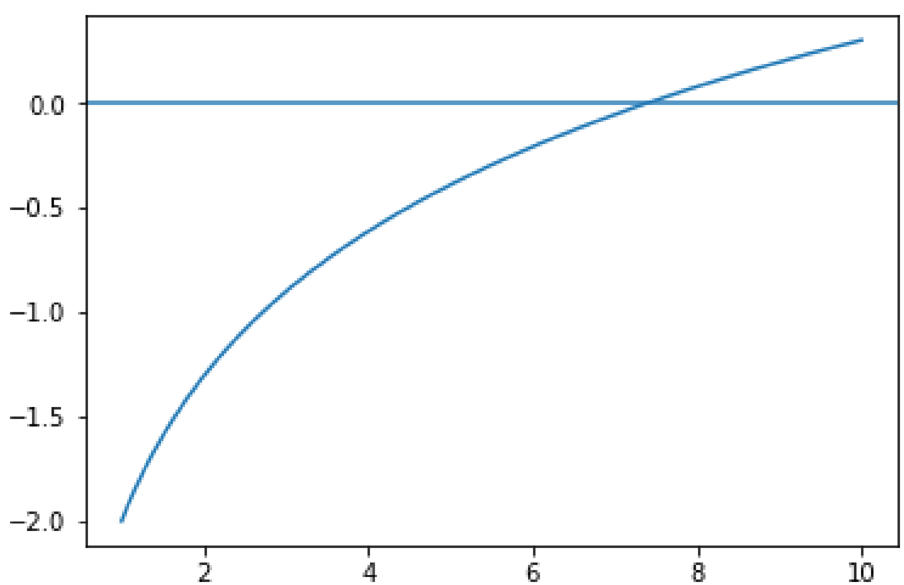
scipy

Exercise bisect

In [100]:
```python
def f(x):
    return np.log(x)-2.
```

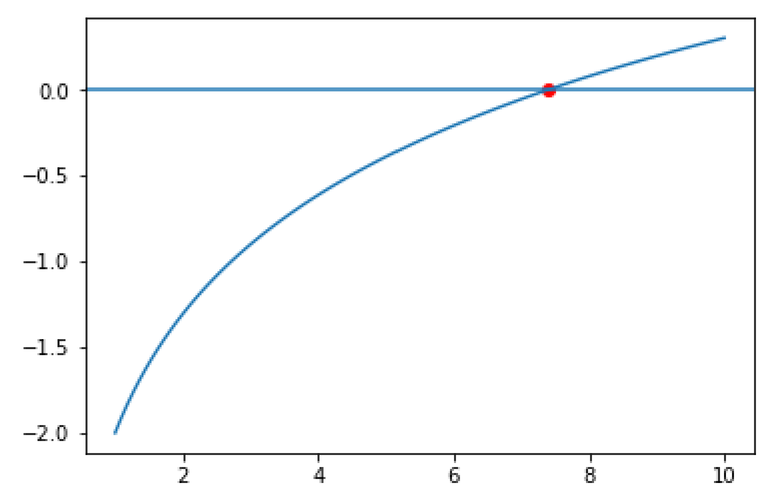
In [98]:
```python
x=np.linspace(1,10,100)
```

In [101]:
```python
y=f(x)
```



In [113]:
```python
x0=2.
x1=20.
f0=f(x0)
f1=f(x1)

tol=1e-5
fn=float("inf")
print(fn)

while abs(fn) > tol:
    xn=(x0+x1)/2
    fn=f(xn)
    if fn>0:
        x1=xn
    else:
        x0=xn

print(xn,fn)
```

```
inf
7.389068603515625 1.69231011737e-06
```



Scipy solver

```
In [100]: def f(x):
              return np.log(x)-2.
```

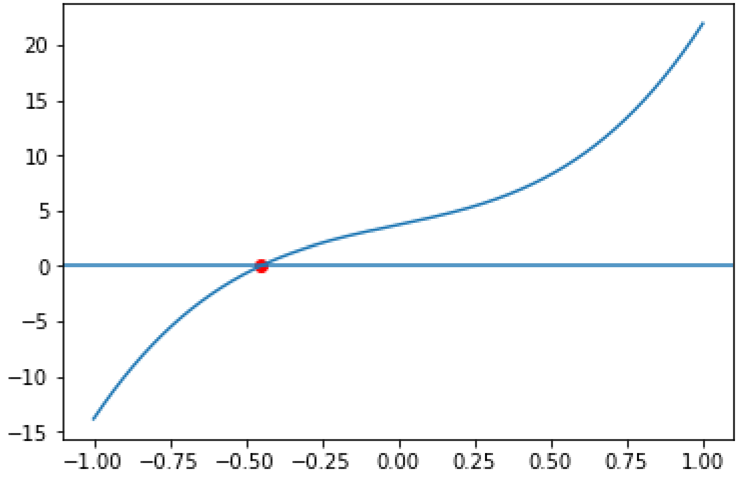
```
In [106]: from scipy import optimize as opt
          xsol=opt.fsolve(func=f,x0=2.)
```

Scipy roots

```
import numpy as np
from scipy import roots
```

```
from matplotlib import pyplot as plt
%matplotlib inline
```

```
a=12
b=.34
c=5.9
d=3.7

def y(x):
    return a*x**3+b*x**2+c*x+d

x=np.linspace(-1,1,1000)

plt.plot(x,y(x))
plt.axhline(0)

raizes=roots((a,b,c,d))

for raiz in raizes:
    if np.isreal(raiz):
        plt.scatter(raiz.real,y(raiz.real),color='r')
```



Aplication psat antoine

Exercise minimize

Scipy fmin

Application param

Exercise trapez

Scipy integrate

Application particle dist

- Generate random
- Plot hist
- Def gaussian
- integrate

Exercise numdiff

Scipy numdiff

Sympy symdiff

Mais Sympy mgaitan

Exercise euler

Scipy ode

Application reator

Pandas

Numba (mgaitan)

- # refs:
- >- python cheat https://www.cheatography.com/davechild/cheat-sheets/python/pdf/
- >- jupyter cheat https://www.cheatography.com/weidadeyue/cheat-sheets/jupyter-notebook/pdf/
- >- numpy manual https://docs.scipy.org/doc/numpy/reference/generated/numpy.genfromtxt.html
- >- scipy manual https://docs.scipy.org/doc/scipy/reference/optimize.html
- >- matplotlib examples https://matplotlib.org/examples/pylab_examples/simple_plot.html