

# AULA 4

Interpolação.

Série 1D. Interpolação polynomial, Interpolação por troços:  
linear, splines.

Sensibilidade à amostra.

Geração de amostras aleatórias.

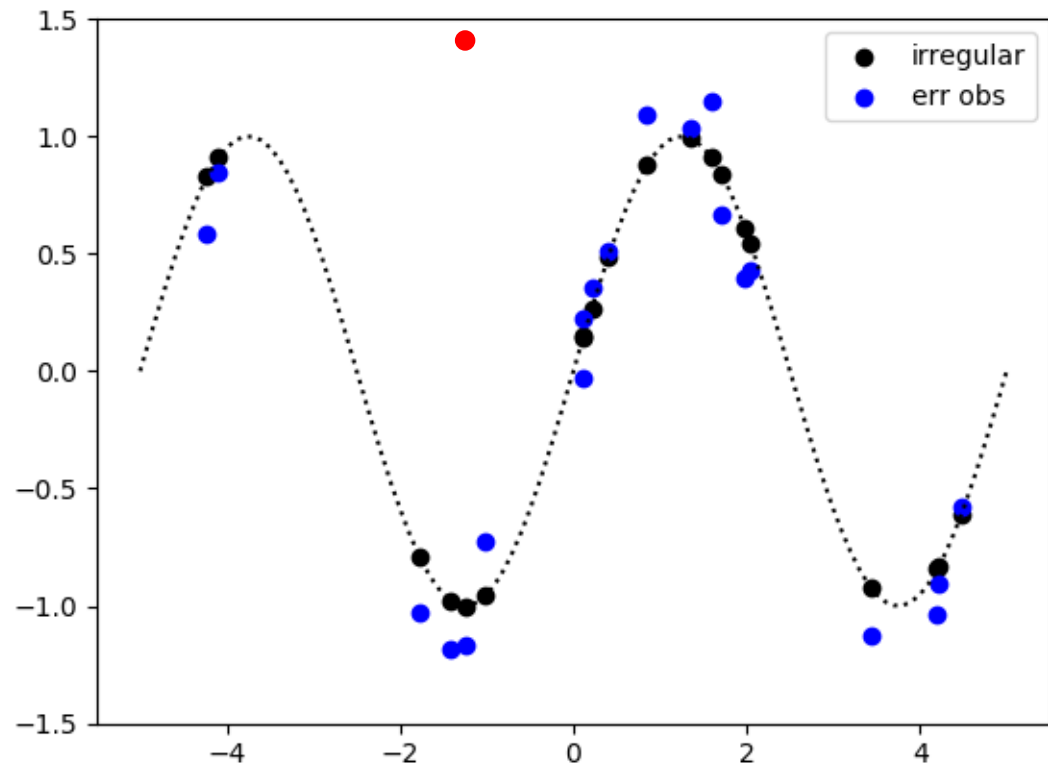
# Porquê interpolar?

Produzir uma amostra regularmente espaçada

Conhecer o valor esperado do sinal num ponto intermédio

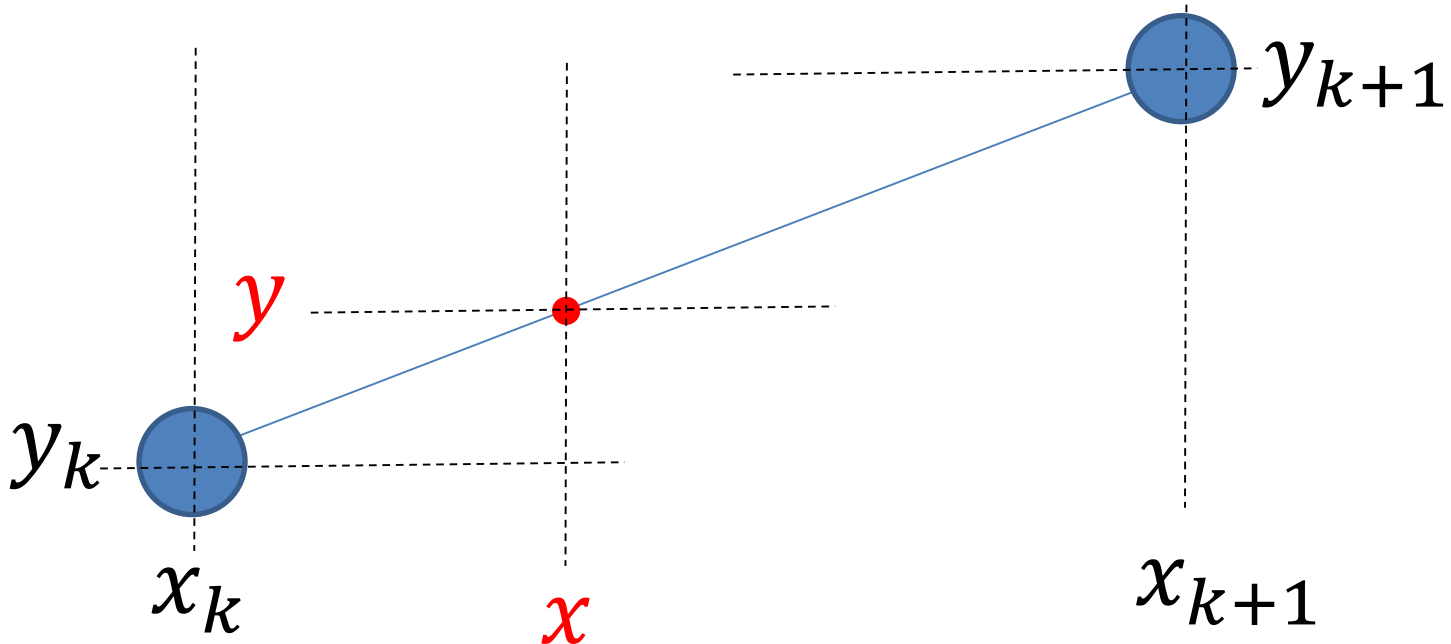
...

Pode ser preciso lidar com erro observacional (**ruído** e **outliers**)



# Interpolação linear por troços (linha quebrada)

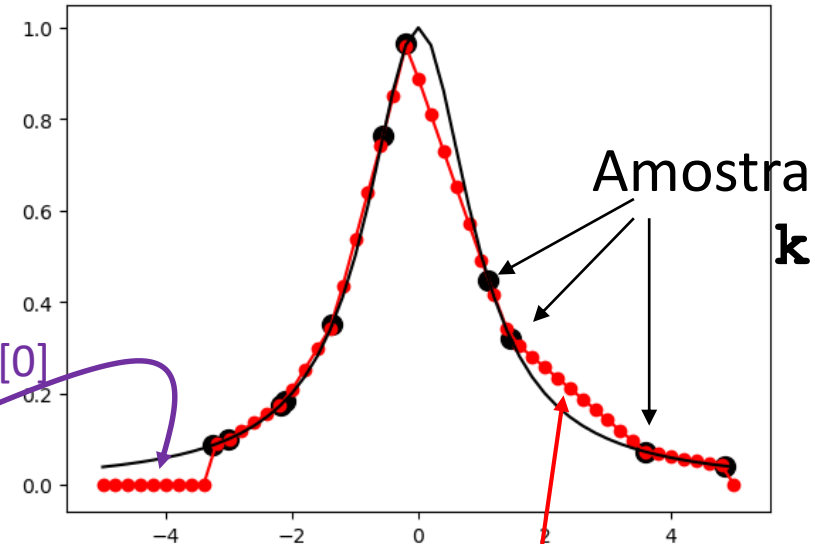
$$y = y_k + \frac{y_{k+1} - y_k}{x_{k+1} - x_k} (x - x_k)$$



# Interpolação linear

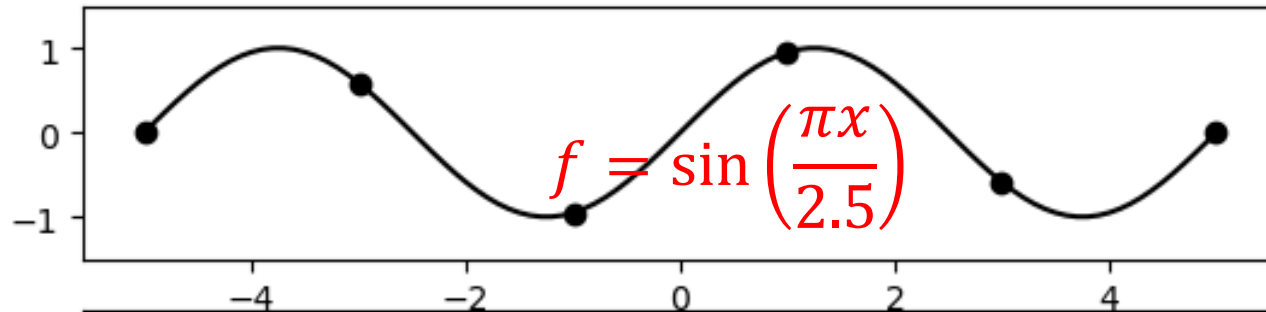
```
import numpy as np
import matplotlib.pyplot as plt
def interpL(x,y,xi):
    yi=np.zeros(xi.shape) #yi=np.nan
    xmax=x[len(x)-1]
    k=0; ki=0
    while xi[ki]<x[k] and ki<len(xi):
        ki=ki+1
    while ki <len(xi)-1 and xi[ki]<xmax:
        yi[ki]=y[k]+(y[k+1]-y[k])*(xi[ki]-x[k])/(x[k+1]-x[k])
        ki=ki+1
    while xi[ki]>x[k+1] and k<len(x)-2:
        k=k+1
    return yi
xplot=np.linspace(-5,5,51)
x=np.sort((np.random.sample(11)-0.5)*10)
y=1./(1+x**2)
f=interpL(x,y,xplot)
plt.scatter(x,y,color='black',s=100)
plt.plot(xplot,f,color='red')
plt.scatter(xplot,f,color='red')
plt.plot(xplot,1/(1+xplot**2),color='black')
```

Pontos  $x[0]$

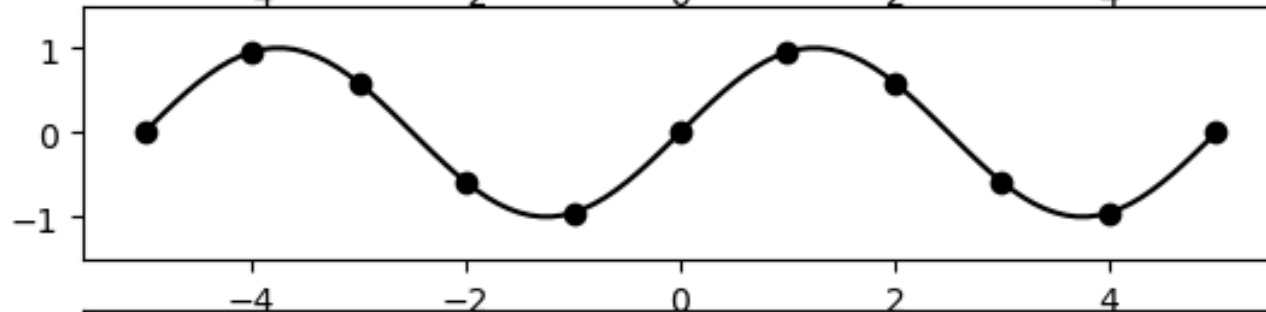


Interpolado  
 $k_i$

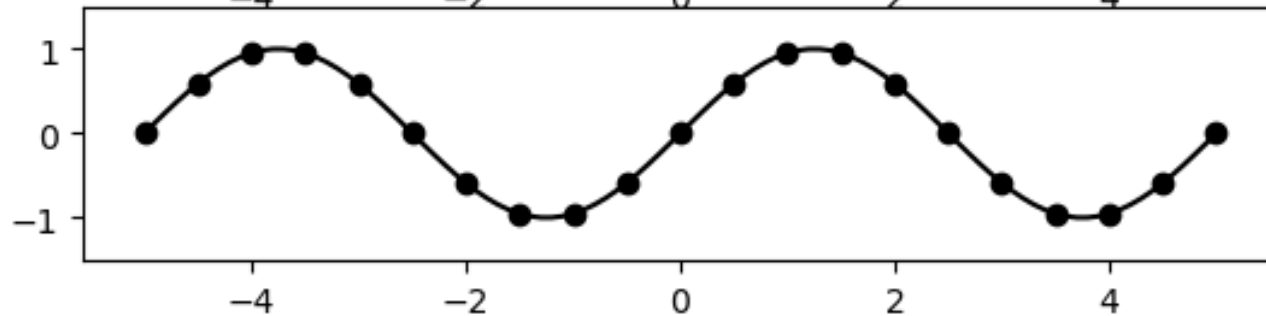
# Interpolando a partir de uma amostra regular



N=6



N=11

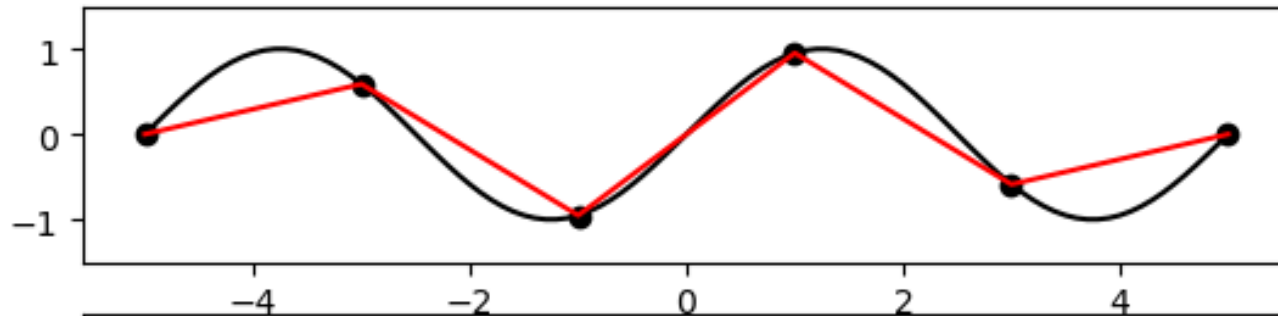


N=21

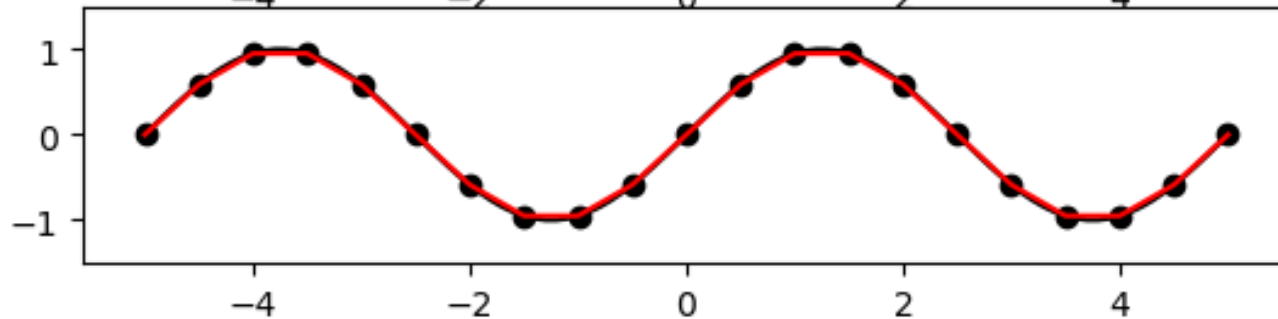
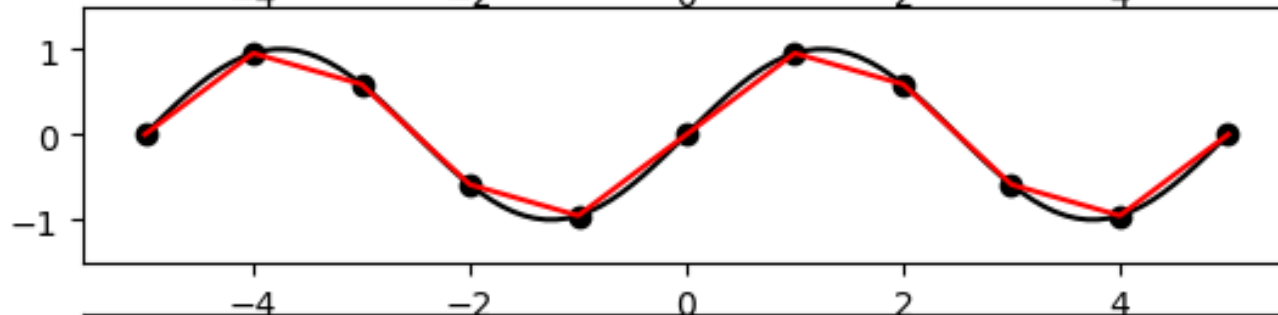
# Interpolação linear por troços (np.interp), podia ser interpL

```
import numpy as np
import matplotlib.pyplot as plt
xfine=np.linspace(-5,5,101)
yfine=np.sin(xfine*np.pi/2.5)
plt.close('all')
NN=[6,11,21];Na=len(NN)
for k in range(Na):
    N=NN[k]
    x=np.linspace(-5,5,N) #x amostra
    y=np.sin(x*np.pi/2.5) #y amostra
    yinter=np.interp(xfine,x,y);
    plt.subplot(3,1,k+1)
    plt.plot(xfine,yfine,color='black') #original
    plt.scatter(x,y,color='black') #amostra
    plt.plot(xfine,yinter,color='red') #interpolado
    plt.ylim(-1.5,1.5)
```

As linhas preta(original) e vermelha (interpolada) têm 101 pontos.



MAU



BOM

# Interpolação polynomial: n pontos, polinómio n-1

$$y = p_0x^{n-1} + \dots + p_{n-1}x^0$$

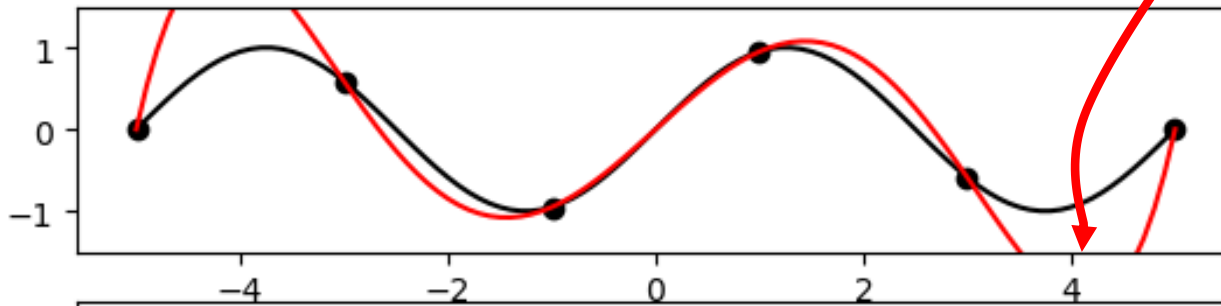
Indices python

```
import numpy as np
import matplotlib.pyplot as plt
xfine=np.linspace(-5,5,101)
yfine=np.sin(xfine*np.pi/2.5)
plt.close('all')
NN=[6,11,21];Np=len(NN)
for k in range(Np):
    N=NN[k]
    x=np.linspace(-5,5,N)
    y=np.sin(x*np.pi/2.5)
    p=np.polyfit(x,y,N-1) #p coeficientes do polinómio
    yinter=np.polyval(p,xfine) # avalia f nos pontos xplot
    plt.subplot(3,1,k+1)
    plt.plot(xfine,yfine,color='black')
    plt.scatter(x,y,color='black')
    plt.plot(xfine,yinter,color='red')
    plt.ylim(-1.5,1.5)
```

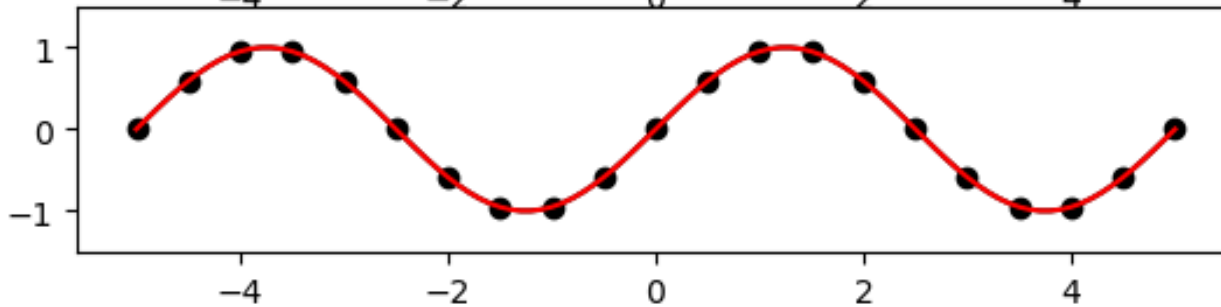
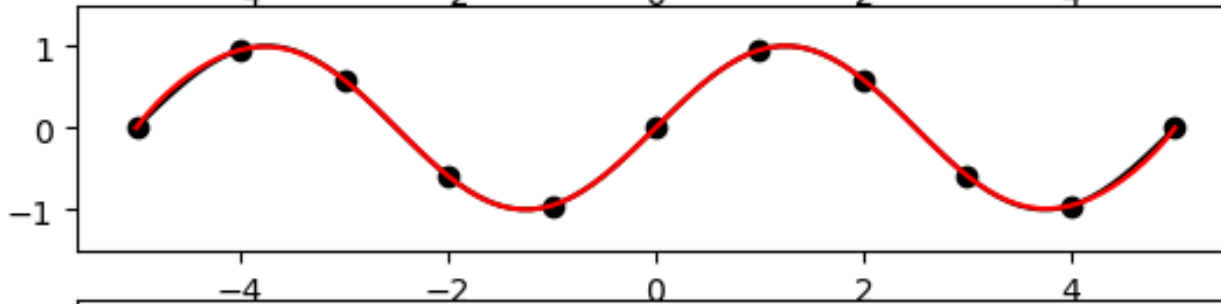


# Interpolação polinomial

fora do alcance



PROBLEMA



EXCELENTE

## np.polyval

Trata-se de um função numpy. Mas seria *fácil* escrever, por exemplo: (atenção aos índices python!)

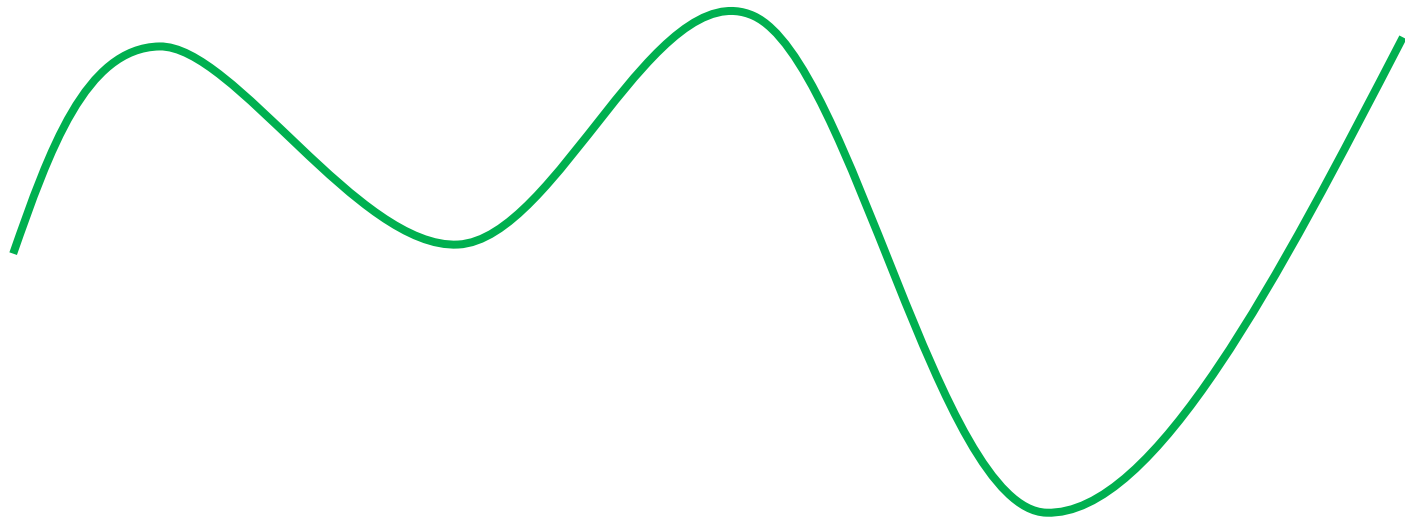
$$y = p_0x^{n-1} + \dots + p_{n-1}x^0$$

```
def polyval (p, x) :  
    n=len(p) #reta (grau 1) n=2  
    y=np.zeros(x.size)  
    for k in range(1, n+1) : #acaba em n!  
        y[k]=y[k]+p[k-1]*x**(n-k)  
    return y
```

# splines

Curvas suaves, cúbicas entre cada dois pontos, deriváveis, segundas derivadas contínuas.

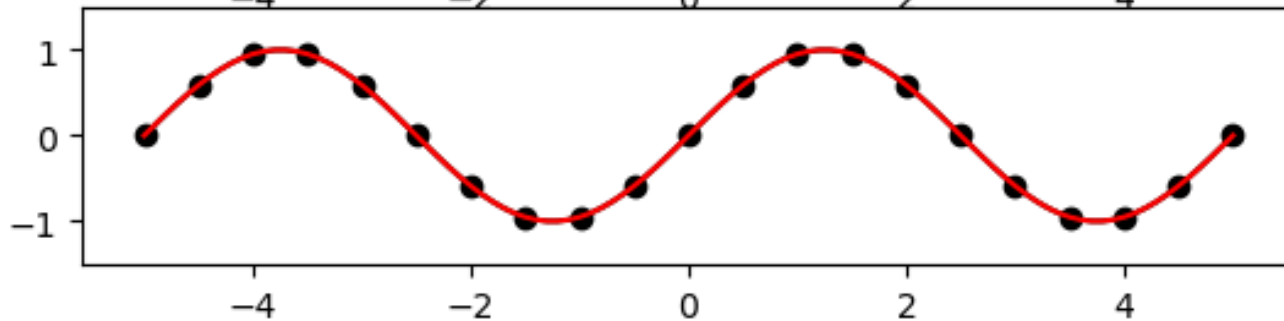
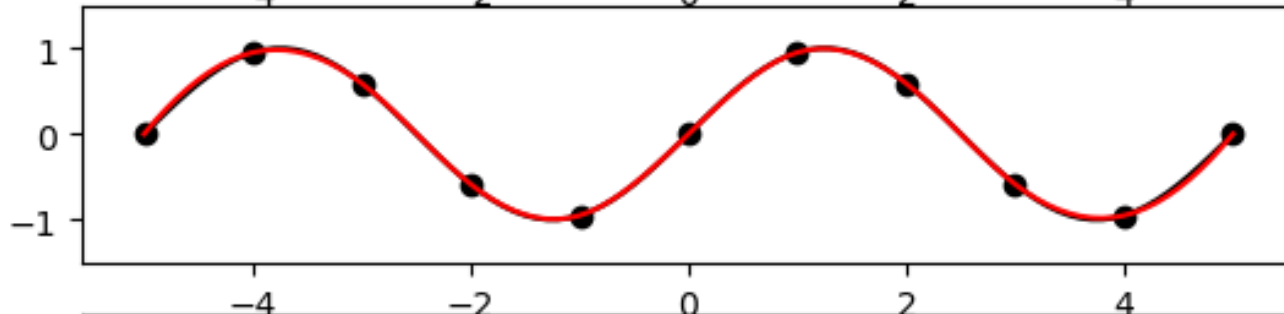
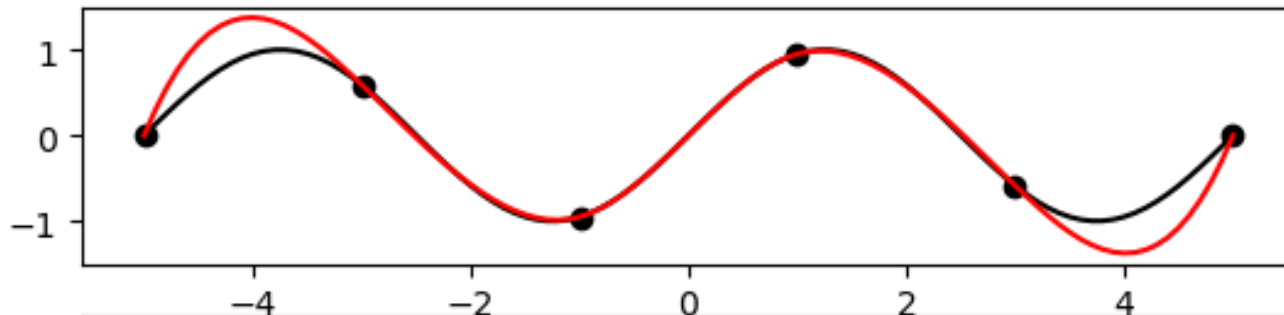
Usadas como interpoladoras nos programas gráficos (inc. Powerpoint)



# splines

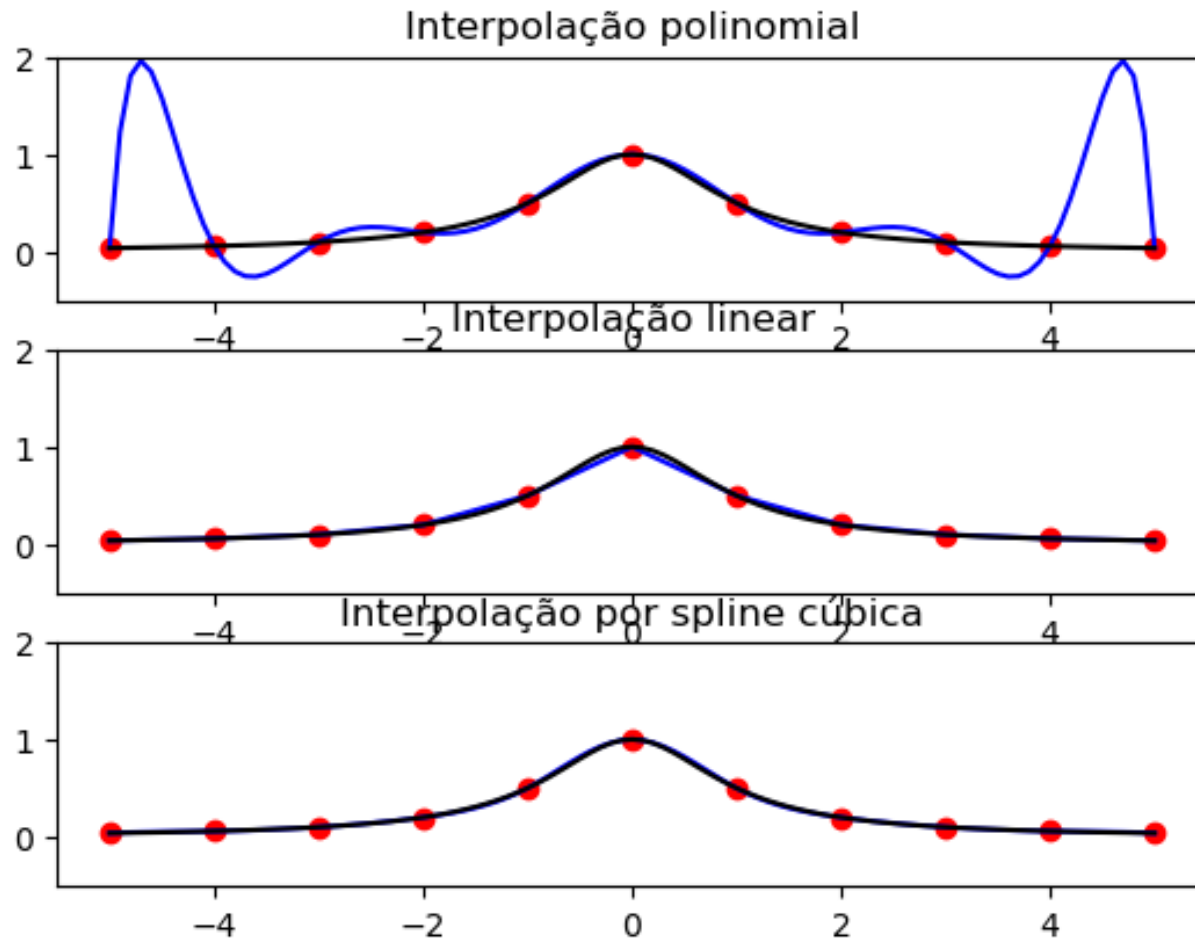
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import spline
xfine=np.linspace(-5,5,101)
yfine=np.sin(xfine*np.pi/2.5)
plt.close('all')
NN=[6,11,21];Np=len(NN)
for k in range(Np):
    N=NN[k]
    x=np.linspace(-5,5,N) y=np.sin(x*np.pi/2.5)
    yinter=spline(x,y,xfine,order=3) #order=1 linear
    plt.subplot(3,1,k+1)
    plt.plot(xfine,yfine,color='black')
    plt.scatter(x,y,color='black')
    plt.plot(xfine,yinter,color='red')
    plt.ylim(-1.5,1.5)
```

# Spline cúbica: uma boa solução por defeito



EXCELENTE

# Interpolando a função $y = \frac{1}{1+x^2}$



# Interpolação de uma amostra irregularmente espaçada

Necessário quando uma série de medidas regularmente espaçadas apresenta falhas.

O método de interpolação polinomial funciona (**mal**) sem alterações. Mas o seu resultado é **muito sensível** à localização dos pontos a interpolar.

# Amostra aleatória

```
np.random.sample(n)
```

amostra aleatória com  $n$  números no  
intervalo  $[0,1]$

```
(np.random.sample(n) - 0.5) * 10
```

vamostra aleatória com  $n$  números no  
intervalo  $[-5,5]$

```
np.sort((np.random.sample(n) - 0.5) * 10)
```

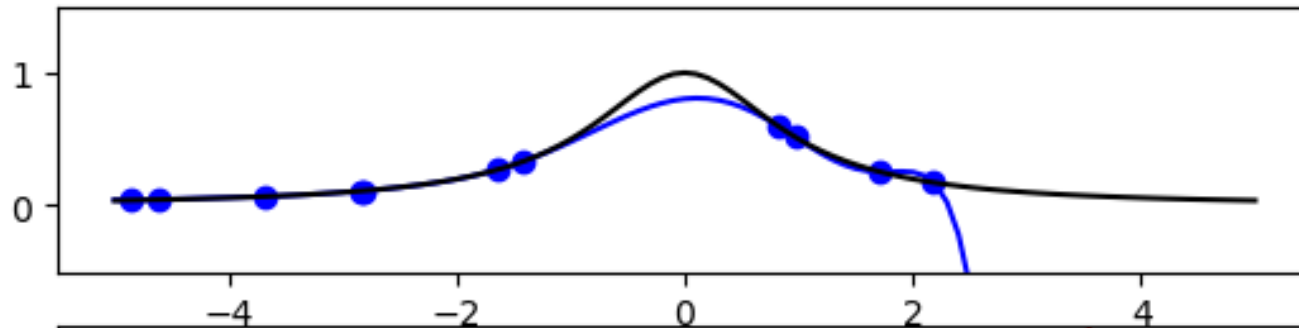
O mesmo, ordenado por ordem crescente



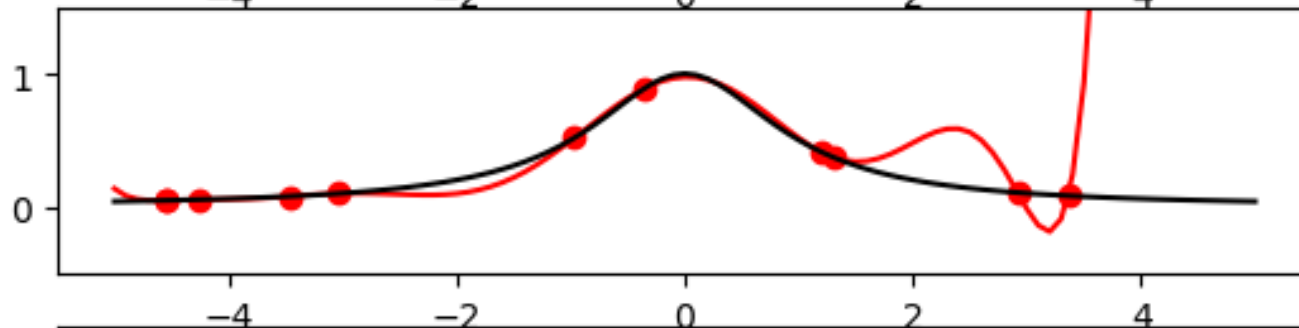
# Interpolação polinomial c/ amostra aleatória

```
import numpy as np
import matplotlib.pyplot as plt
xplot=np.linspace(-5,5,101) #original
cores=['blue', 'red', 'green'] #3 amostras
plt.close('all')
for k in range(len(cores)):
    x=np.sort((np.random.sample(11)-0.5)*10) #amostra
    N=len(x)-1 # escolha de grau 10 para polinómio
    y=1./(1+x**2) # função de Runge
    p=np.polyfit(x,y,N) #p coeficientes do polinómio
    f=np.polyval(p,xplot) # avalia f nos pontos xplot
    plt.subplot(len(cores),1,k+1);
    plt.scatter(x,y,color=cores[k])
    plt.plot(xplot,f,color=cores[k])
    plt.plot(xplot,1/(1+xplot**2),color='black')
    plt.ylim(-0.5,1.5)
```

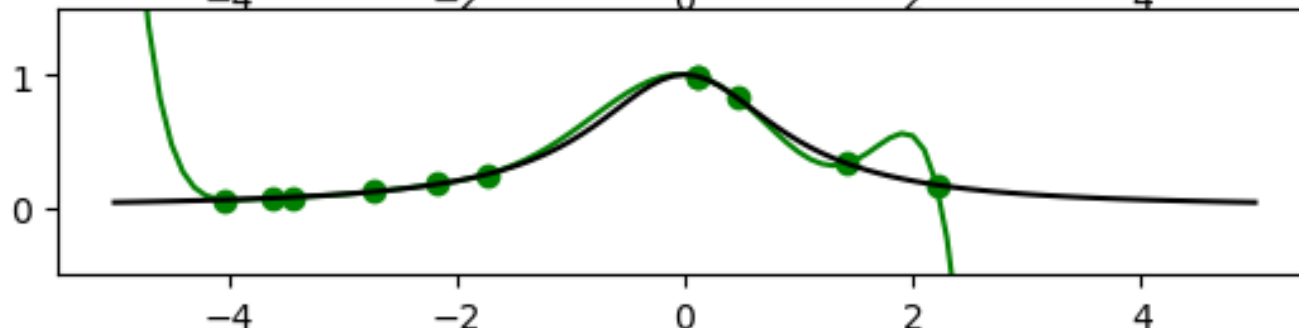
# Amostra aleatória, interpolação polinomial



N=11



N=11

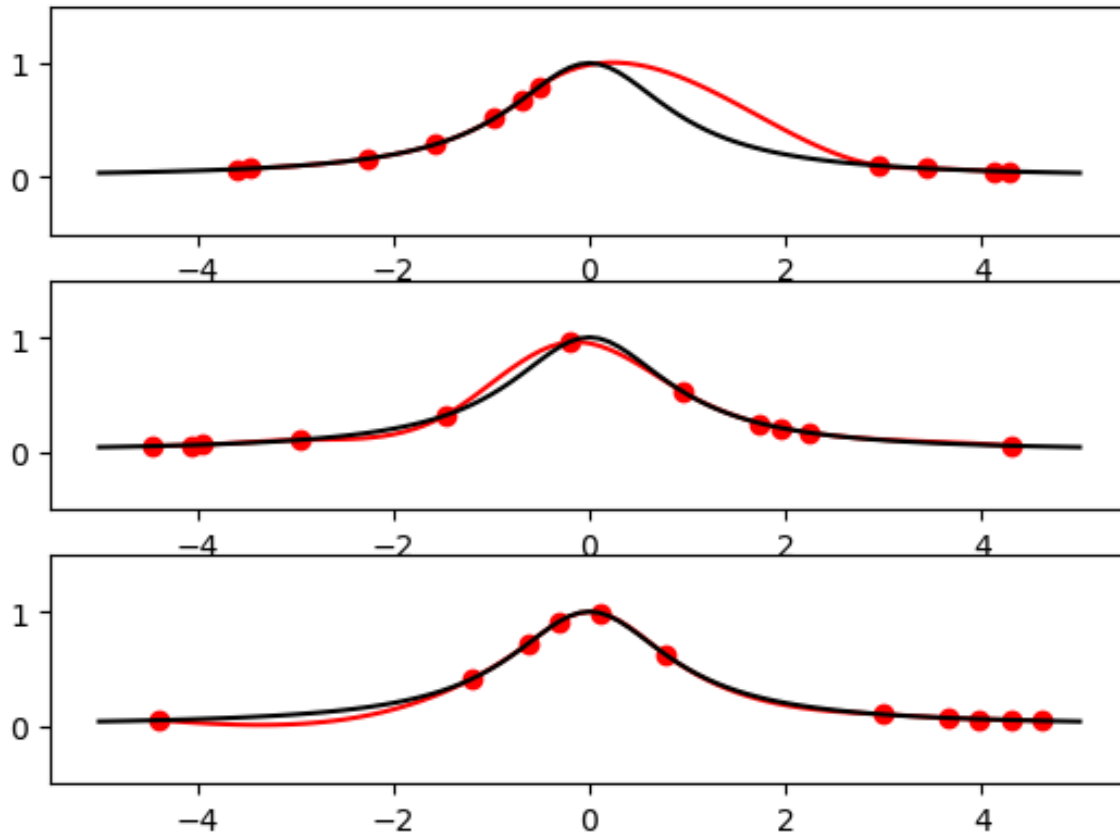


N=11

# Interpolação por troços (linear,spline): **griddata**

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import griddata
xplot=np.linspace(-5,5,101)
plt.close('all')
for k in range(3): #3 amostras
    x=np.sort(np.random.sample(11)-0.5)*10)
    y=1./(1+x**2)
    f=griddata(x,y,xplot,method='cubic')
    plt.subplot(3,1,k+1);
    plt.scatter(x,y,color='red')
    plt.plot(xplot,f,color='red')
    plt.plot(xplot,1/(1+xplot**2),color='black')
    plt.ylim(-0.5,1.5)
```

# Spline c/ griddata (...method='cubic')



# Interpolação

## Polinomial

- Suave (derivável) mas tende a ter oscilações espúrias
- Cria novos extremos (máximos e mínimos)
- Problemas na fronteira da série
- É muito sensível ao ruído (mesmo de pequena amplitude)

## Linear por troços (linha quebrada)

- Não cria novos extremos
- Pouco sensível ao ruído
- Não é suave. A 1ª derivada tem descontinuidades

## Splines (cúbica por troços)

- Suave, segundas derivadas contínuas
- Simula uma interpolação feita por um desenhador com uma “cobra”
- Pode criar novos máximos e mínimos
- Medianamente sensível ao ruído

# Antes de manipular dados

Representem-nos graficamente.

Pensem!

Existe um modelo físico subjacente?

Depois de interpolar, olhem de novo para os dados.

A interpolação introduziu oscilações excessivas?

Na falta de melhor: interpolação linear por troços é mais seguro...

# Projeto A

A1,...,A8/9 (grupo)

No Fenix, amanhã