Python para Ciências Exatas

Prof: Dennis

#02: Gráficos de funções com matplotlib

https://wiki.python.org/moin/BeginnersGuide

https://matplotlib.org/

https://numpy.org/

O que faremos nessa aula:

- Neste curso, na aula anterior já vimos:
  - O Python, como abrir um ambiente de programação python online para criar nossos programas em python sem instalar nada ou mesmo de um celular ou tablet;
  - Como fazer um código muito simples em python;
  - Como criar um vetor e mostra-lo usando o “numpy”;

- Nesta aula veremos:
  - Como fazer um código muito simples para produzir um gráfico em algumas linhas;
  - Usaremos o numpy para os dados que serão graficados;
  - Faremos gráficos simples de funções;
  - Aprenderemos como melhorar o aspecto do gráfico, incluindo legendas, mudando as fontes e cores de linhas e pontos;

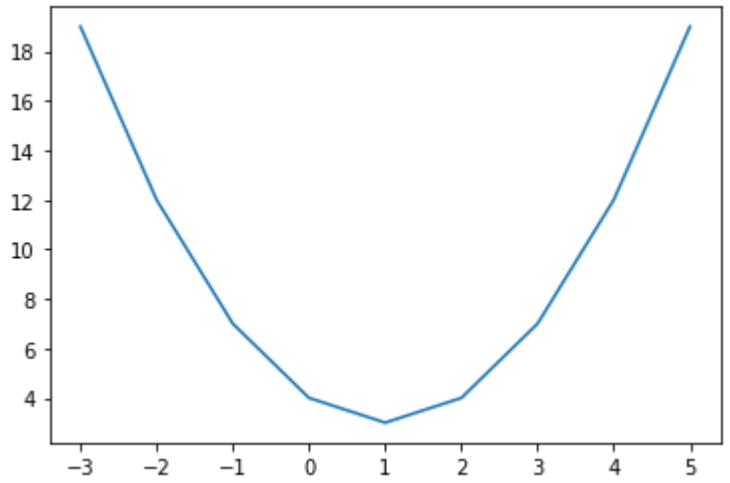
Meu primeiro gráfico em python

- Para transformar o python em uma poderosa ferramenta de produção de gráficos e mesmo animações usaremos a biblioteca "matplotlib.pyplot";
- Assim como o numpy (que transforma o python em um ferramenta matemática) essa biblioteca precisa ser importada para o código. Por padrão usaremos "plt" como apelido para essa biblioteca
- Para o nosso primeiro gráfico, digite o código a seguir:

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([-3, -2, -1, 0, 1, 2, 3, 4, 5])
y = x*x-2*x+4
plt.plot(x,y)
```

$$y = x^2 - 2x + 4$$

Uau, 5 linhas!



https://numpy.org/doc/stable/reference/generated/numpy.array.html
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html

Meu primeiro gráfico em python

- Vamos entender a função de cada linha:

```
import numpy as np
```
→ *importa a biblioteca numpy com o apelido de np*

```
import matplotlib.pyplot as plt
```
→ *importa a biblioteca matplotlib.pyplot com o apelido de plt*

```
x = np.array([-3, -2, -1, 0, 1, 2, 3, 4, 5])
```
→ *cria o vetor* $[-3\ -2\ -1\ 0\ 1\ 2\ 3\ 4\ 5]$ *e guarda na variável* "x"

```
y = x*x-2*x+4
```
→ *calcula* $x \cdot x - 2x + 4$ *e guarda na variável* "y"

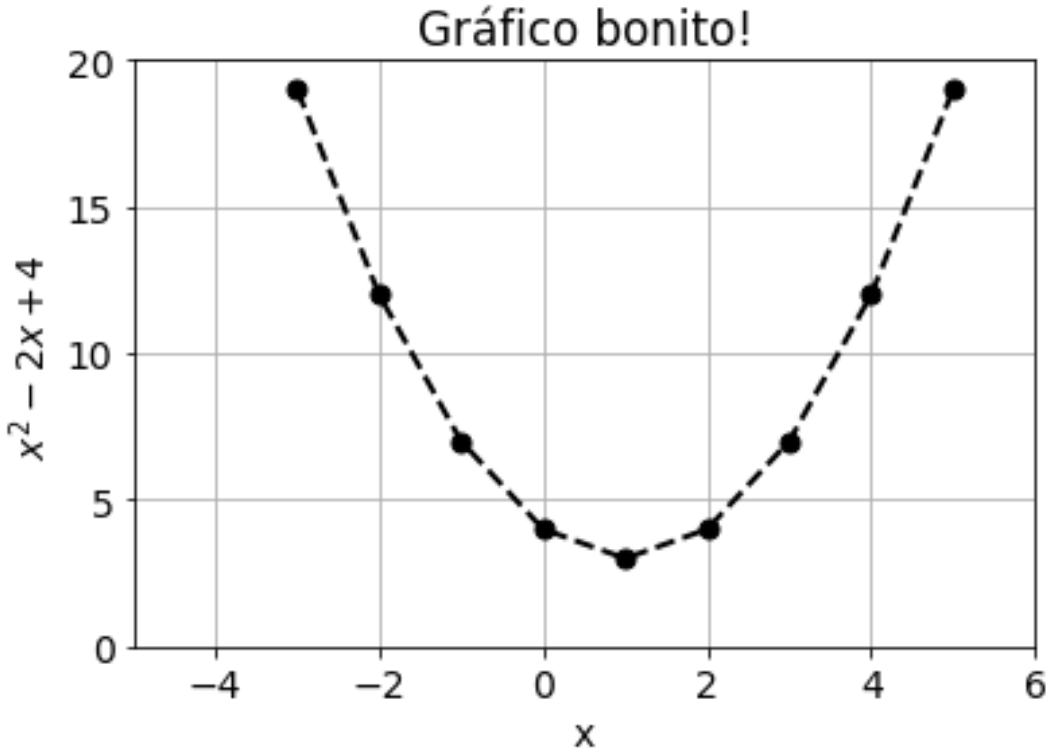
```
plt.plot(x,y)
```
→ *faz o gráfica de y por x, usando as condigurações padrão da função "plot"*

https://numpy.org/doc/stable/reference/generated/numpy.array.html
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html

Minhas primeiras configurações de gráficos

- Uma grande vantagem de fazer gráficos usando o python, é poder configura-lo como se deseja em linhas de código;
- Com um pequeno texto no final do comando plot vamos configurar as linhas e pontos do gráfico;
- Com alguns comando após do plot vamos configurar a área do gráfico;

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 14})

x = np.array([-3, -2, -1, 0, 1, 2, 3, 4, 5])
y = x*x-2*x+4
plt.plot(x,y,'o--k',linewidth=2,markersize=7)
plt.title('Gráfico bonito!')
plt.xlabel('x')
plt.ylabel('$x^2-2x+4$')
plt.xlim([-5, 6])
plt.ylim([0,20])
plt.grid()
```



https://stackoverflow.com/questions/3899980/how-to-change-the-font-size-on-a-matplotlib-plot

Minhas primeiras configurações de gráficos

- Vamos entender a função de cada linha nova:

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 14})
```
→ *fixa em 14 todas a fonte para tudo no gráfico*

```
x = np.array([-3, -2, -1, 0, 1, 2, 3, 4, 5])
y = x*x-2*x+4
plt.plot(x,y,'o--k',linewidth=2,markersize=7)
```
o texto 'o--k' : coloca uma marca redonda, linha tracejada e preta.
→ linewidth=2: faz a linha ter espessura 2, o padrão é 1.
markersize=7: faz a marca ter tamanho 7, o padrão é 6

```
plt.title('Gráfico bonito! ')
```
→ *coloca um título no gráfico*, "Gráfico bonito"

```
plt.xlabel('x')
```
→ *coloca uma legenda no eixo x, chamada "x"*

```
plt.ylabel('$x^2-2x+4$')
```
→ *coloca uma legenda no eixo y*, os $ fazem o texto ser interpretado como equação!

```
plt.xlim([-5, 6])
```
→ *define que o gráfico irá exibir o eixo x entre* -5 *e* 6.

```
plt.ylim([0,20])
```
→ *define que o gráfico irá exibir o eixo y entre* 0 *e* 20.

```
plt.grid()
```
→ *Coloca grade no gráfico*

Minhas primeiras configurações de gráficos

- Vamos olhar com um pouco mais de detalhe o texto ‘0—k’ no comando plot;

```
plt.plot(x,y,'o--k',linewidth=2,markersize=7)
```

- Esse texto deve seguir preferencialmente a sequencia [marca][linha][cor];
- Outras opções para esses parâmetros podem ser encontradas em:

https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot

'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
'8'	octagon marker
's'	square marker

'p'	pentagon marker
'P'	plus (filled) marker
'*'	star marker
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'X'	x (filled) marker
'D'	diamond marker
'd'	thin_diamond marker
'|'	vline marker
'_'	hline marker

character	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

Melhorando os dados da função

- Já vimos que fazer o gráfico de uma função é simples com um vetor criado pela função np.array();
- Mas com poucos pontos a função não fica suave;
- A solução é trocar o np.array() pelo np.arange() ou np.linspace():

```
import numpy as np


x1=np.arange(-3,5,2)  → gera o vetor: x1= [−3 −1 1 3], ou seja, inicia em −3, termina antes de 5 com passo 2
print('x1=',x1)


x2=np.arange(-3,5,1) → gera o vetor: x2= [−3 −2 −1 0 1 2 3 4], ou seja, inicia em −3, termina antes de 5 com passo 1
print('x2=',x2)


x3=np.linspace(-3,5,6)  → gera o vetor: x3=[−3. −1.4  0.2  1.8  3.4  5. ], ou seja, inicia em −3, termina em 5 com 6 pontos
print('x3=',x3)
```

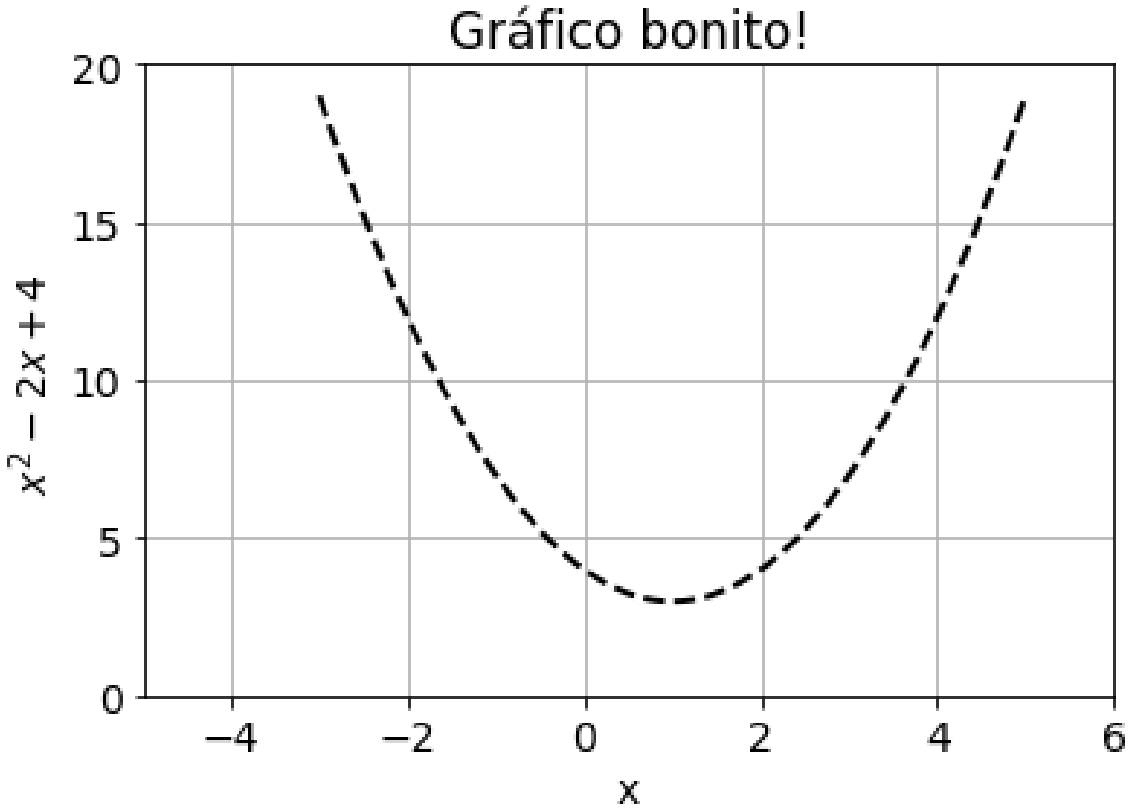
https://stackoverflow.com/questions/3899980/how-to-change-the-font-size-on-a-matplotlib-plot

Melhorando os dados da função

- Vamos usar o np.linspace() para deixar a função mais suave.

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 14})

x = np.linspace(-3,5,1000)
y = x*x-2*x+4
plt.plot(x,y,'--k',linewidth=2)
plt.title('Gráfico bonito!')
plt.xlabel('x')
plt.ylabel('$x^2-2x+4$')
plt.xlim([-5, 6])
plt.ylim([0,20])
plt.grid()
```

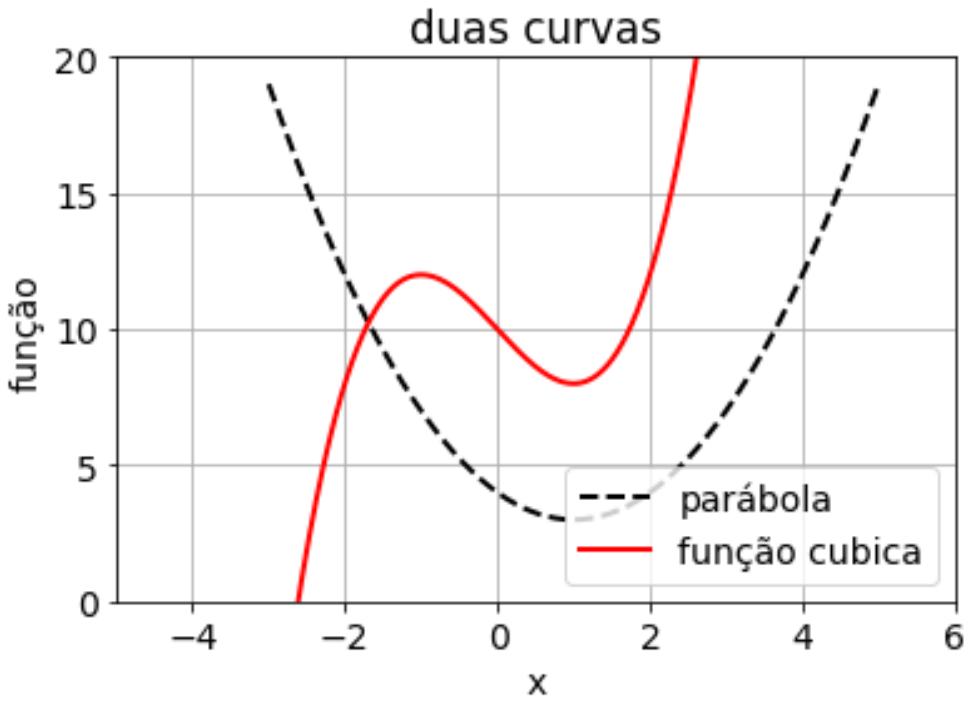
Essa curva agora tem 1000 pontos de resolução!



Duas ou mais curvas no mesmo gráfico

- Agora veremos como é simples colocar mais funções, ou curvas, em um mesmo gráfico;
- Para isso basta usar plt.plot() para cada curva:

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 14})

x = np.linspace(-3,5,1000)
y = x*x-2*x+4
z = x**3-3*x+10

plt.plot(x,y,'--k',label='parábola',linewidth=2)
plt.plot(x,z,'-r',label='função cubica',linewidth=2)
plt.title('duas curvas')
plt.xlabel('x')
plt.ylabel('função')
plt.xlim([-5, 6])
plt.ylim([0,20])
plt.legend(loc=4)
plt.grid()
```



Duas ou mais curvas no mesmo gráfico

- Agora veremos como é simples colocar mais funções, ou curvas, em um mesmo gráfico;
- Para isso basta usar plt.plot() para cada curva:

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 14})

x = np.linspace(-3,5,1000)
y = x*x-2*x+4
z = x**3-3*x+10    → usamos uma nova variável para a segunda função. lembre: em python x ** 3 é x^3

plt.plot(x,y,'--k',label='parábola',linewidth=2)  → o label =' parabola' será usado na legenda
plt.plot(x,z,'-r',label='função cubica',linewidth=2)  → a segunda curva será vermelha linha sólida
plt.title('duas curvas')
plt.xlabel('x')
plt.ylabel('função')
plt.xlim([-5, 6])
plt.ylim([0,20])
plt.legend(loc=4)   → o loc = 4 define a localização da legenda
plt.grid()
```

Localizações possíveis para a legenda:

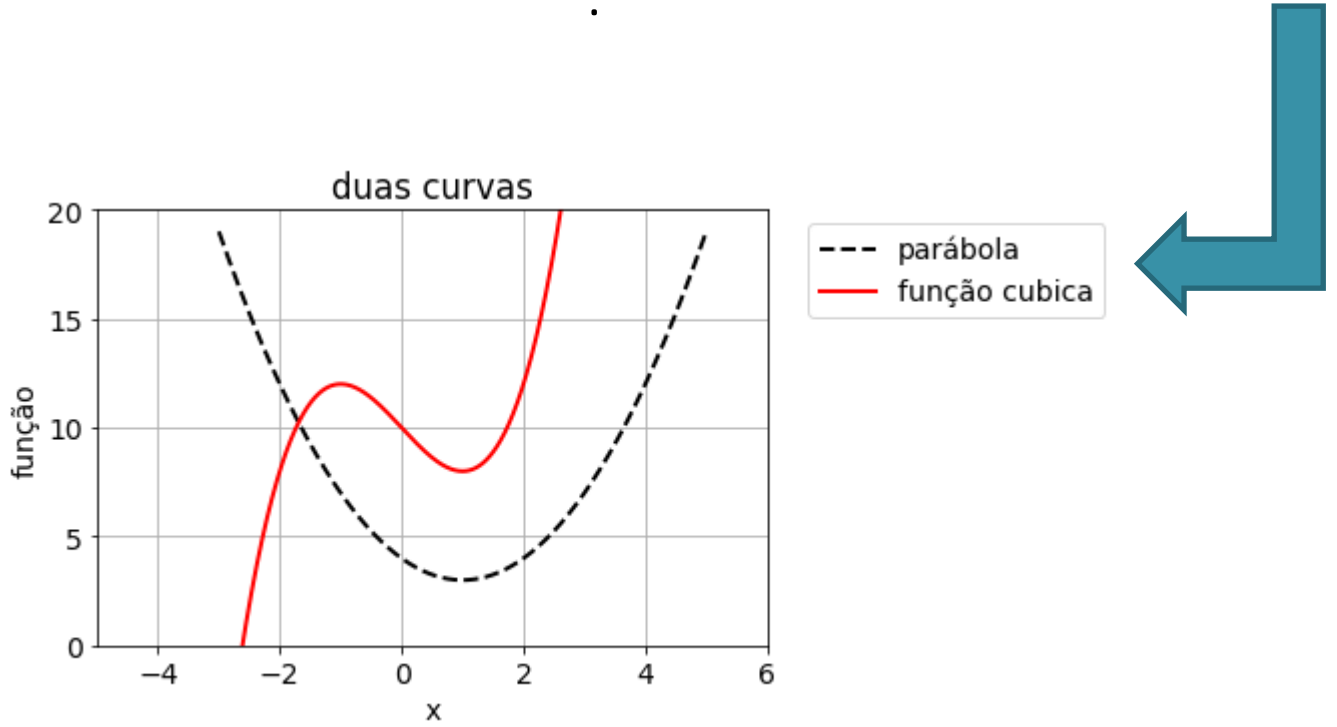
- Para localizar a legenda em algum lugar dentro a área do gráfico, use a tabela existente em:

https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.legend.html

- Mas se desejar a legenda fora da área do gráfico, use "bbox_to_anchor=(x0,y0)", como no commando de exemplo:

:
plt.legend(bbox_to_anchor=(1.04,1))
:

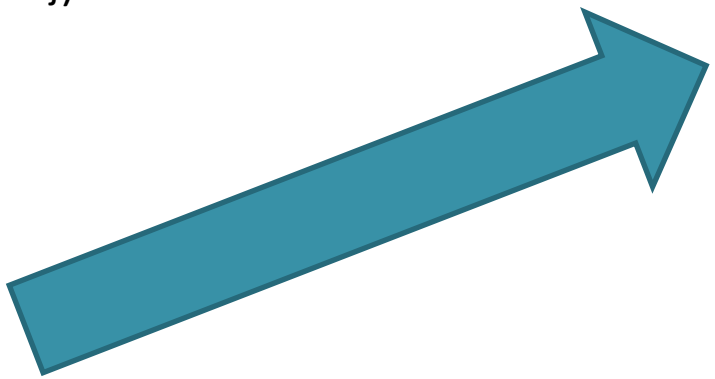
Location String	Location Code
'best'	0
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center'	9
'center'	10

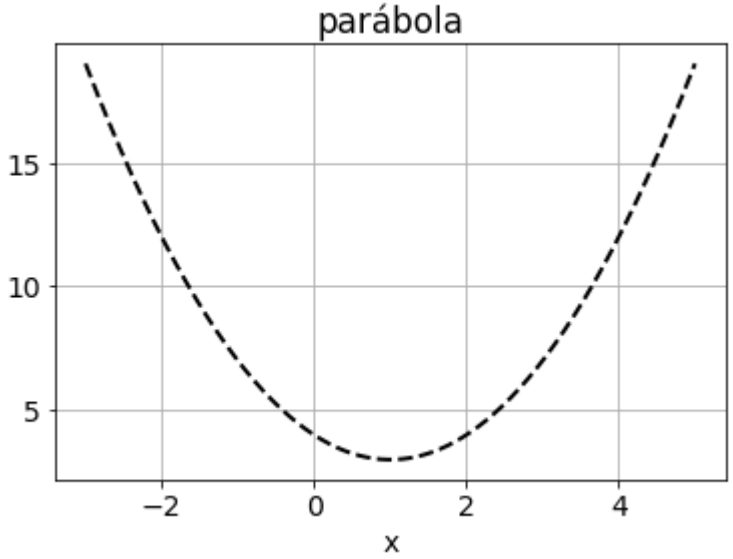


Duas ou mais figuras

- Agora veremos como criar mais figuras para que todas as curvas não caiam no mesmo gráfico usando o comando plt.figure()

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 14})

x = np.linspace(-3,5,1000)
y = x*x-2*x+4
z = x**3-3*x+10

plt.plot(x,y,'--k',linewidth=2)
plt.title('parábola')
plt.xlabel('x')
plt.grid()

plt.figure()
plt.plot(x,z,'-r',linewidth=2)
plt.title('função cubica')
plt.xlabel('x')
plt.grid()
```





Duas ou mais figuras

- Agora veremos como criar mais figuras para que todas as curvas não caiam no mesmo gráfico usando o comando plt.figure()

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 14})

x = np.linspace(-3,5,1000)
y = x*x-2*x+4
z = x**3-3*x+10

plt.plot(x,y,'--k',linewidth=2)
plt.title('parábola')
plt.xlabel('x')
plt.grid()
```

→ a primeira figura deve ser construído sem alterações.
Mas como não teremos 2 curvas, a legenda foi tirada
e um novo título incorporado.

```
plt.figure()
plt.plot(x,z,'-r',linewidth=2)
plt.title('função cubica')
plt.xlabel('x')
plt.grid()
```

→ o comando plt.figure() cria uma nova figura, daqui em diante qualquer plt.plot() será na nova figura.

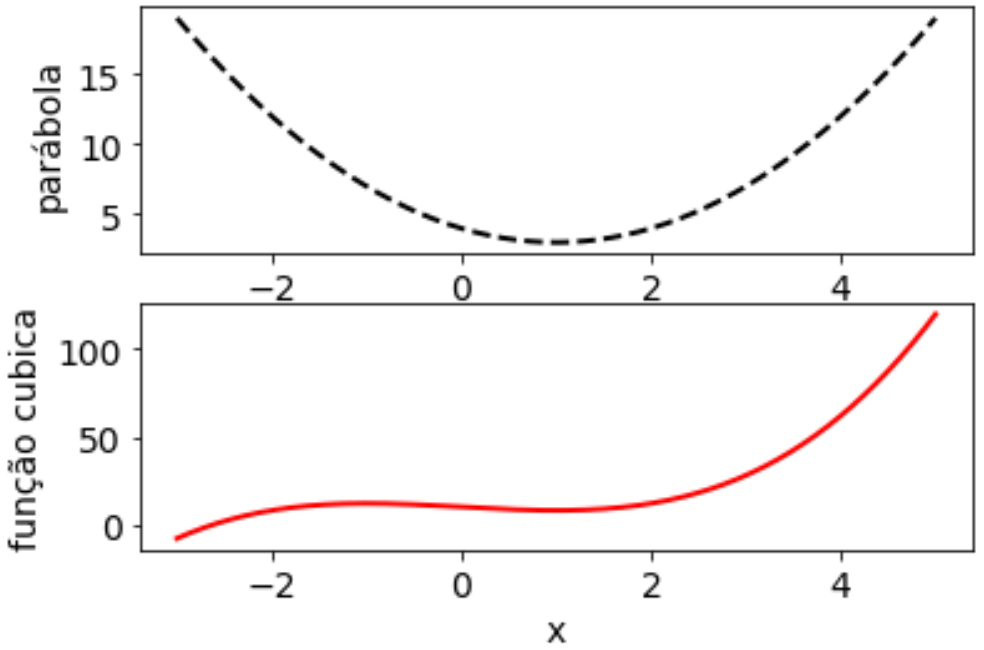
→ não se esqueça de que, a menos do tamanho da fonte,
tudo deve ser novamente configurado.

Dois ou mais gráficos na mesma figura: plt.subplot()

- Agora veremos como dividir uma mesma figura para inserir mais de um gráfico na mesma figura.
- Quem fara isso é o plt.subplot(n_linhas, n_colunas, qual_gráfico)
- Note que esse comando tem 3 parametros, que definem em quantas linhas em quantas colunas a figura será dividida.
- O último parâmetro de em qual das divisões será plotada a curva;

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 14})

x = np.linspace(-3,5,1000)
y = x*x-2*x+4
z = x**3-3*x+10
plt.subplot(2,1,1) → 2 linhas 1 coluna, gráfico 1
plt.plot(x,y,'--k',linewidth=2)
plt.ylabel('parábola')
plt.subplot(2,1,2) → 2 linhas 1 coluna, gráfico 2
plt.plot(x,z,'-r',linewidth=2)
plt.ylabel('função cubica')
plt.xlabel('x')
```

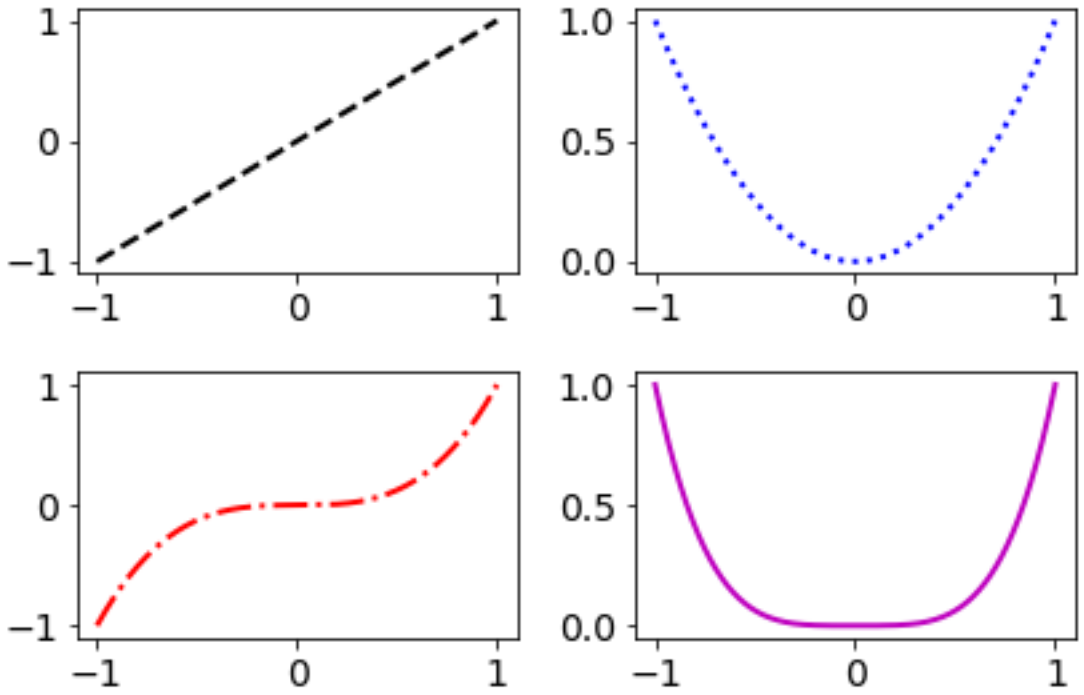
→ os plt.grid() foram tirados apenas por simplicidade



Dois ou mais gráficos na mesma figura: plt.subplot()

- Mais um exemplo, agora com 4 gráficos.

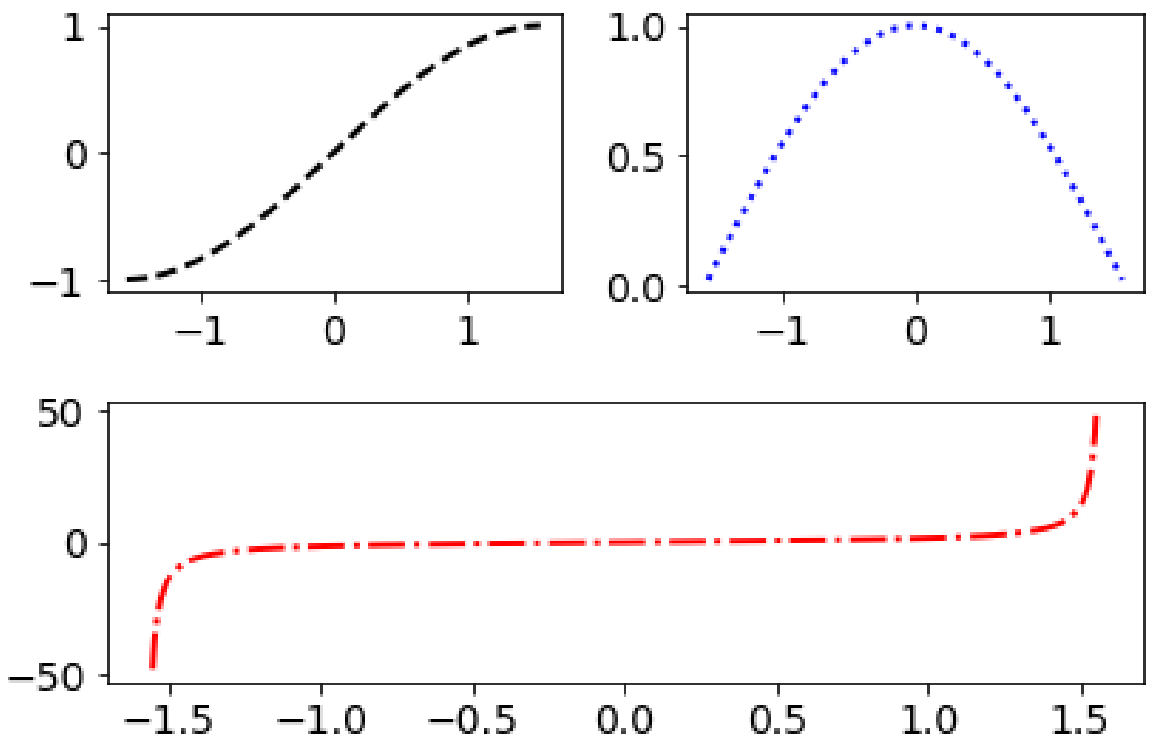
```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 14})

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

plt.subplot(2,2,1)
plt.plot(x,x,'--k',linewidth=2)
plt.subplot(2,2,2)
plt.plot(x,x**2,':b',linewidth=2)
plt.subplot(2,2,3)
plt.plot(x,x**3,'-.r',linewidth=2)
plt.subplot(2,2,4)
plt.plot(x,x**4,'-m',linewidth=2)
plt.tight_layout()
```



Dois ou mais gráficos na mesma figura: plt.subplot()

- Mais um exemplo, misturando as quantidades de linhas e colunas.

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 14})

x = np.linspace(-1.55,1.55,1000)

plt.subplot(2,2,1)
plt.plot(x,np.sin(x),'--k',linewidth=2)
plt.subplot(2,2,2)
plt.plot(x,np.cos(x),':b',linewidth=2)
plt.subplot(2,1,2)
plt.plot(x,np.tan(x),'-.r',linewidth=2)
plt.tight_layout()
```



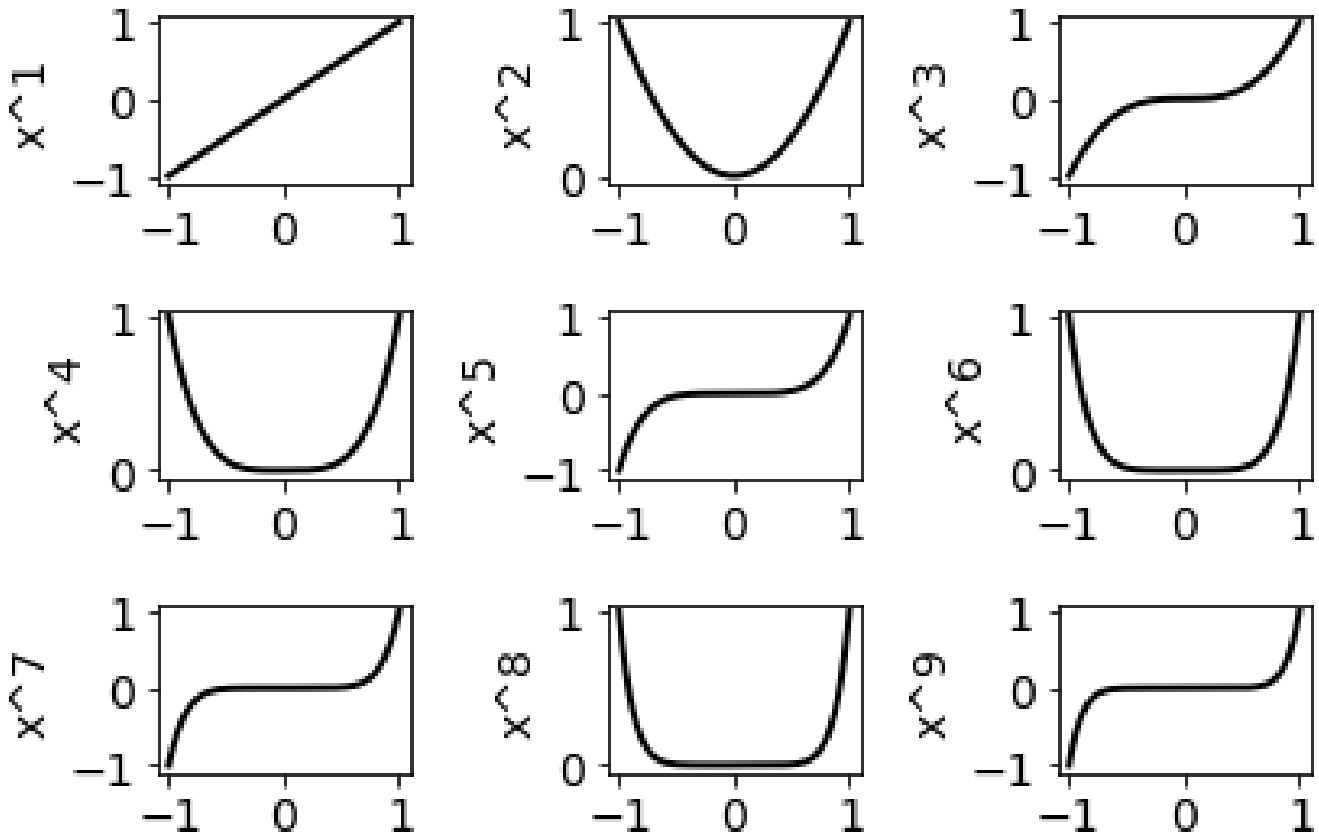
Uau, um resultado e tanto!

Experimente você também, mude o numero de linha e colunas do subplot!

Bonus 1: Misturando laço FOR com plt.subplot()

- Como um bônus vamos fazer um exemplo em que usamos um laço for para fazer 9 gráfico!

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 14})

x = np.linspace(-1,1,1000)
for i in range(1, 10):
  plt.subplot(3,3,i)
  plt.plot(x,x**i,'-k',linewidth=2)
  plt.ylabel('x^'+str(i))
plt.tight_layout()
```

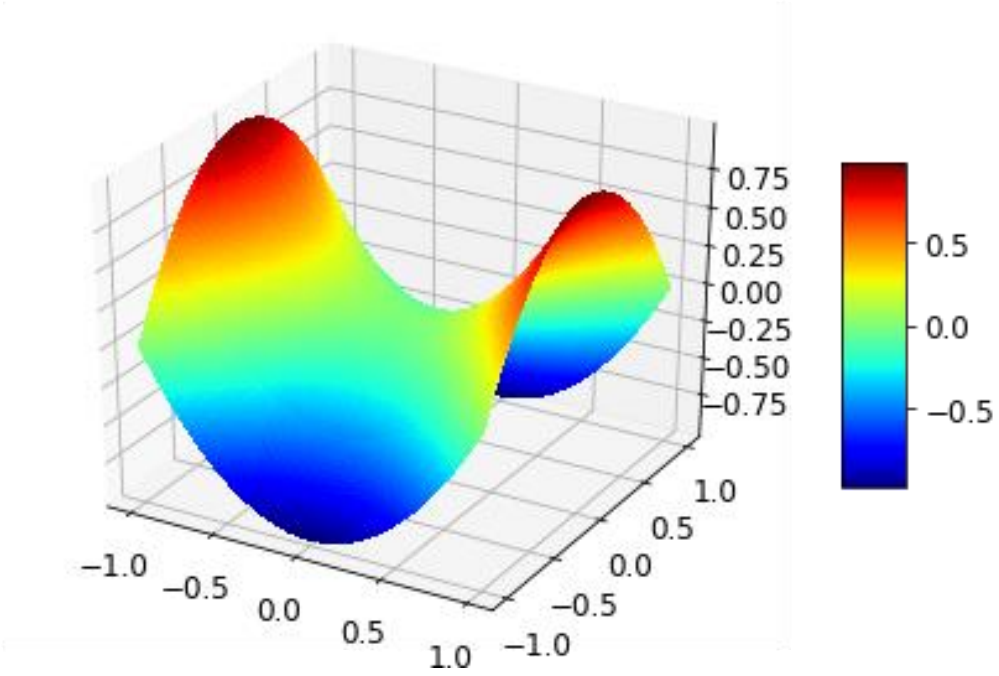


Poderoso não?

Bonus 2: Gráfico de função 3D

- Como um segundo bônus vamos fazer uma superfície:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

plt.rcParams.update({'font.size': 12})

x = np.linspace(-1,1,100)
y = np.linspace(-1,1,100)
X, Y = np.meshgrid(x, y)
Z=X**2-Y**2
fig = plt.figure()
ax = fig.gca(projection='3d')
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=plt.cm.
jet,
    linewidth=0, antialiased=False)
#ax.set_zlim(0, 100)
#ax.view_init(elev=45, azim=30)
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.tight_layout()
```

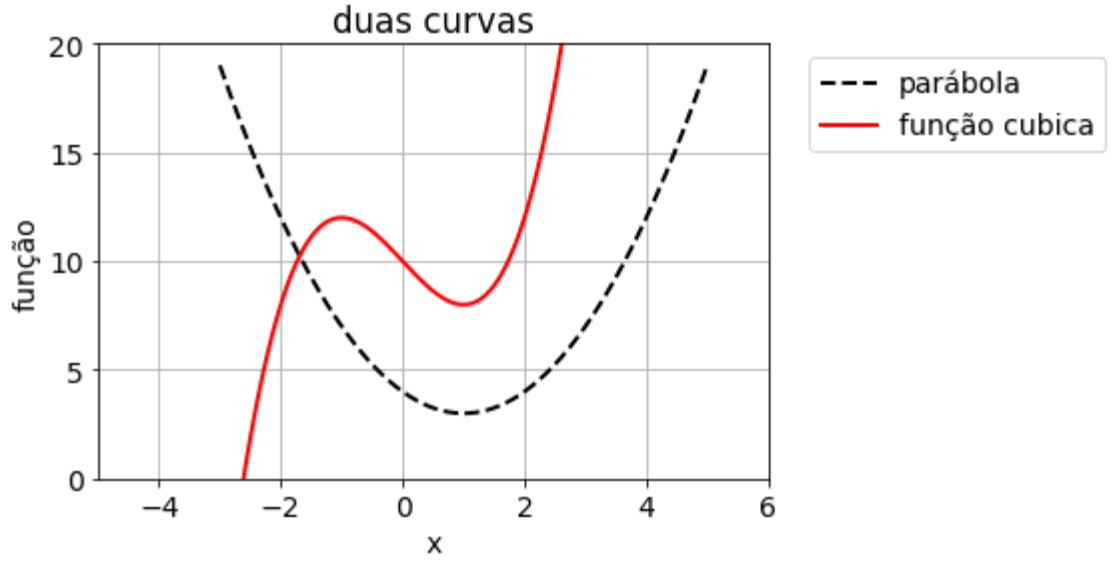


Uauu! Tão poucas linhas!

Vá em frente, mude a função de Z e veja outras superfícies!

Exercício 1:

- Refaça o gráfico abaixo em dois gráficos verticais na mesma figura (1 linha, 2 colunas), usando 5000 pontos na curva.



Exercício 2:

- Faça os gráficos das funções linear, seno e cosseno, de -1,55 á +1,55 no mesmo gráfico. Use as legendas apropriadas e coloque 100 pontos em cada curva;

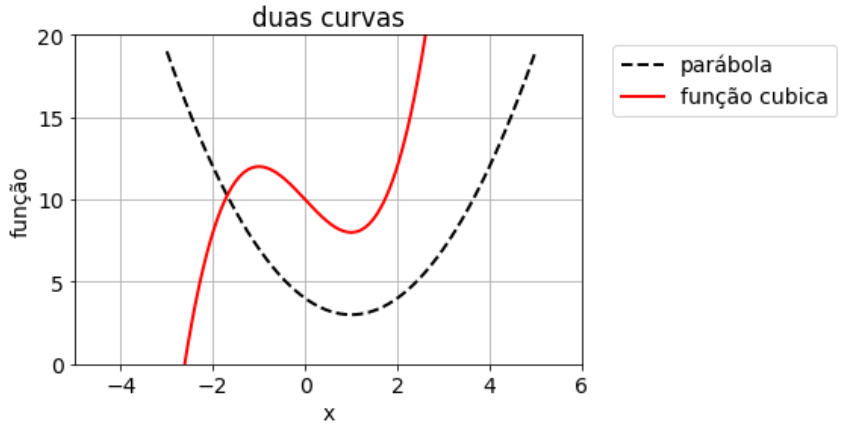
Exercício 3:

- Faça os gráficos de polinômios da tabela abaixo no mesmo gráfico, que ocupa a metade de cima da figura. Em seguida faça o gráfico separado de cada um em 4 “subplots” que ocupam a segunda metade de baixo da figura.

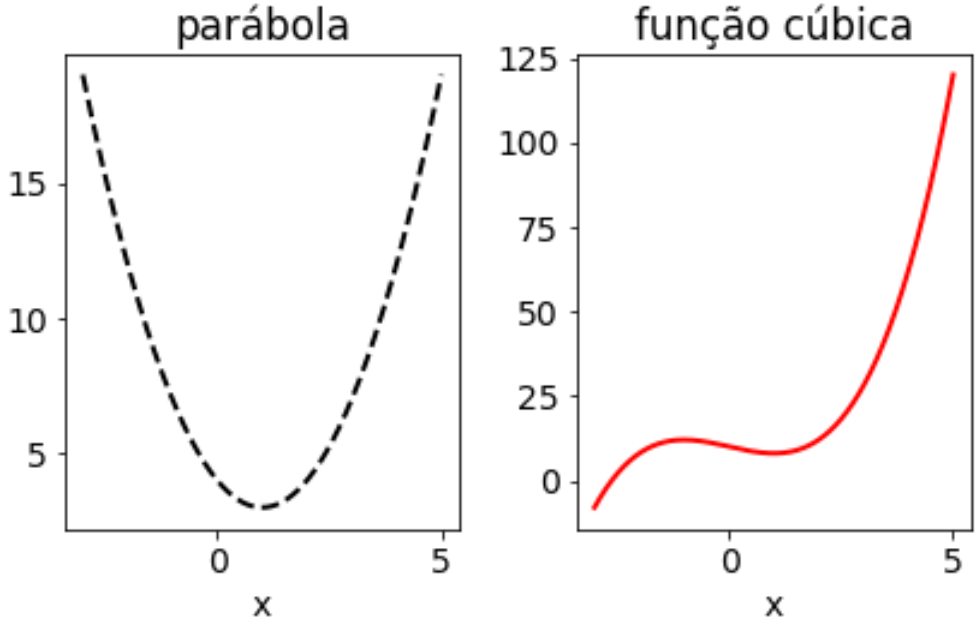
$y = 2x^4 - 4x^2 + 2$
$y = -3x^3 + 5\mathrm{x}$
$y = 2x^2 - 4x + 2$
$y = -3x + 5$

Resposta Exercício 1:

- Refaça o gráfico abaixo em dois gráficos verticais na mesma figura (1 linha, 2 colunas), usando 5000 pontos na curva.



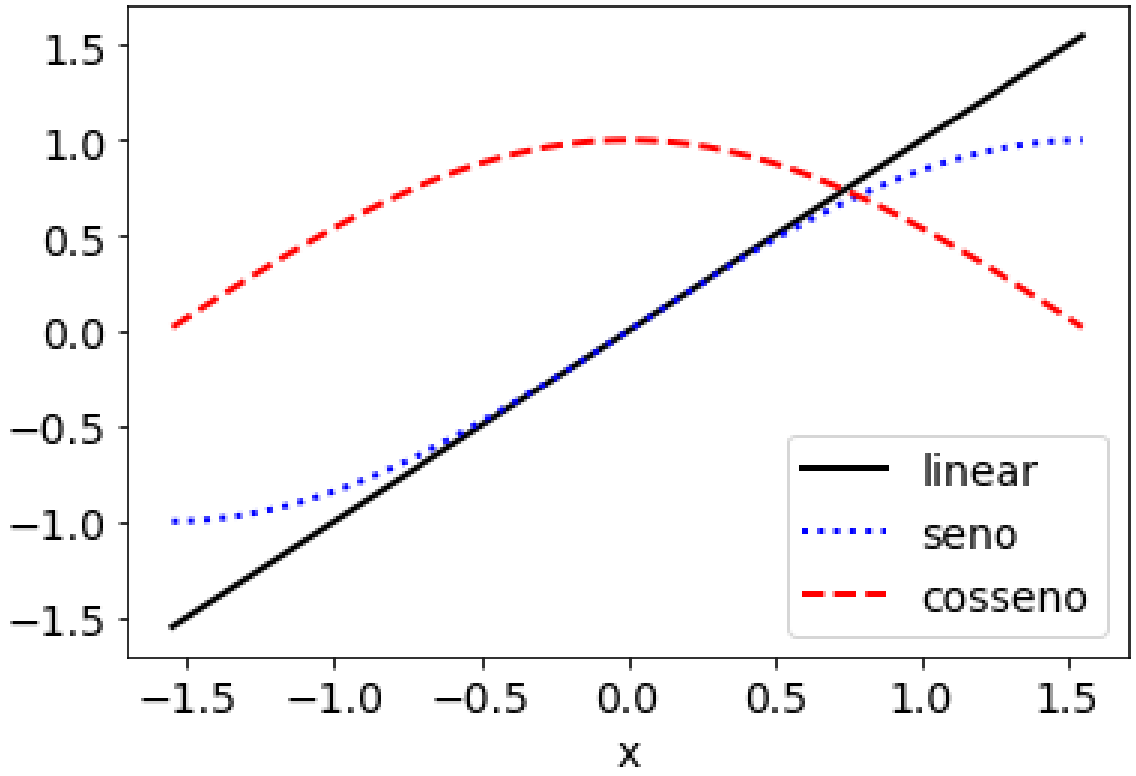

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 14})

x = np.linspace(-3,5,5000)
y = x*x-2*x+4
z = x**3-3*x+10
plt.subplot(1,2,1)
plt.plot(x,y,'--k',linewidth=2)
plt.title('parábola')
plt.xlabel('x')
plt.subplot(1,2,2)
plt.plot(x,z,'-r',linewidth=2)
plt.title('função cúbica')
plt.xlabel('x')
plt.tight_layout()
```



Resposta Exercício 2:

- Faça os gráficos das funções linear, seno e cosseno, de -1,55 á +1,55 no mesmo gráfico. Use as legendas apropriadas e coloque 100 pontos em cada curva;

```
#exercício 2
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 14})

x = np.linspace(-1.55,1.55,100)

plt.plot(x,x,'-k',label='linear',linewidth=2)
plt.plot(x,np.sin(x),':b',label='seno',linewidth=2)
plt.plot(x,np.cos(x),'--r',label='cosseno',linewidth=2)
plt.legend(loc=4)
plt.xlabel('x')
```

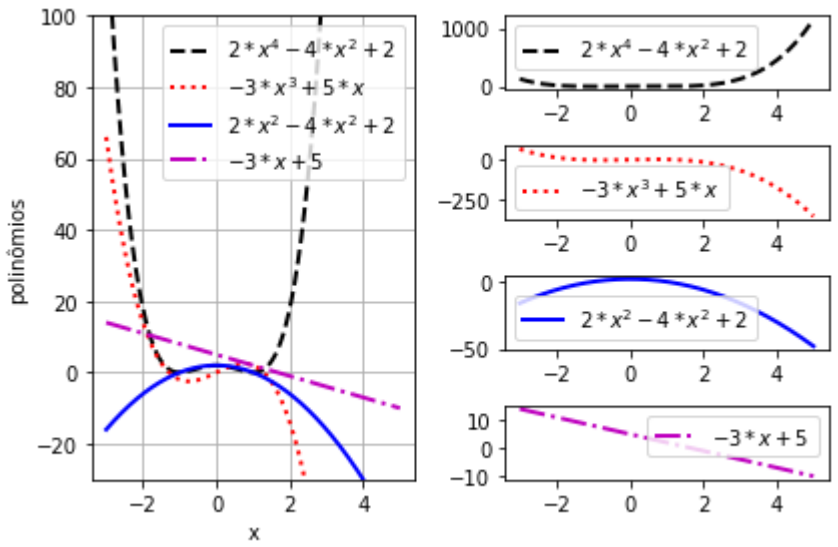


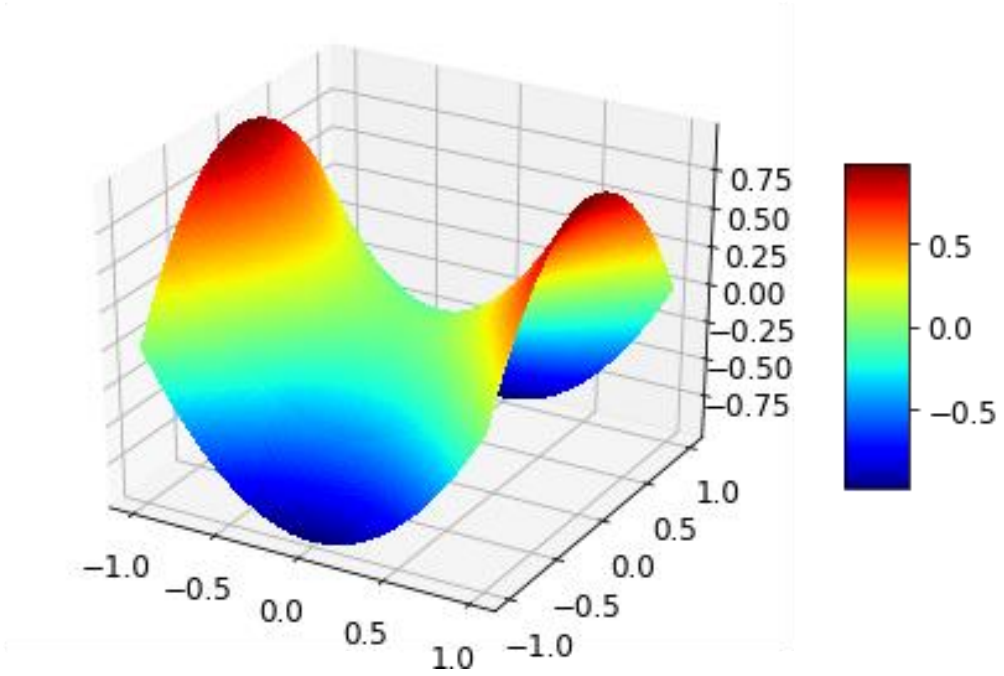
Resposta Exercício 3:

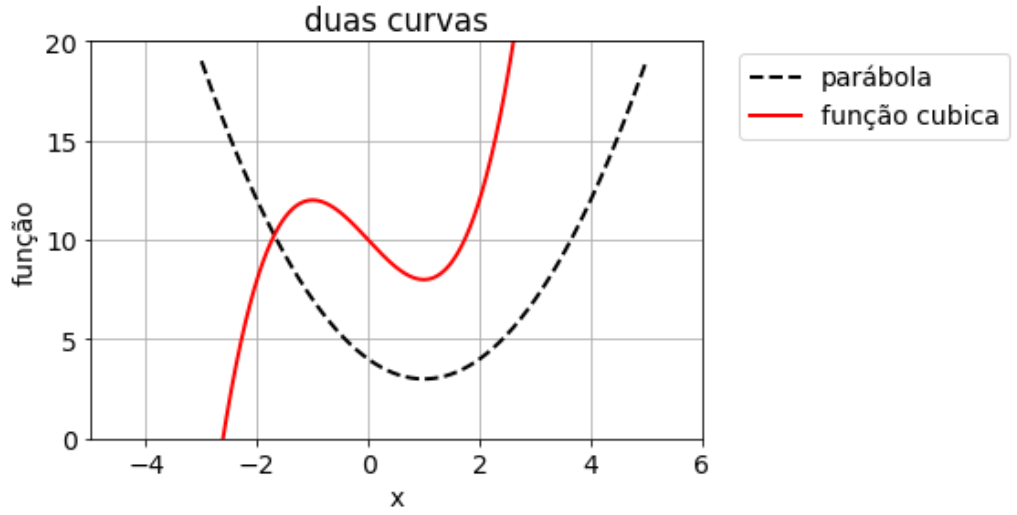
- Faça os gráficos de polinômios da tabela abaixo no mesmo gráfico, que deve ocupar a metade de à esquerda da figura. Em seguida faça o gráfico separado de cada um em 4 "subplots" que ocupam a segunda metade da direita da figura.

$y = 2x^4 - 4x^2 + 2$
$y = -3x^3 + 5x$
$y = 2x^2 - 4x + 2$
$y = -3x + 5$

```
#exercício 3
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 10})

x = np.linspace(-3,5,1000)
p1 = 2*x**4-4*x**2+2
p2 = -3*x**3+5*x
p3 = 2*x**2-4*x**2+2
p4 = -3*x+5
plt.subplot(1,2,1)
plt.plot(x,p1,'--k',label='$2*x^4-4*x^2+2$',linewidth=2)
plt.plot(x,p2,':r',label='$-3*x^3+5*x$',linewidth=2)
plt.plot(x,p3,'-b',label='$2*x^2-4*x^2+2$',linewidth=2)
plt.plot(x,p4,'-.m',label='$-3*x+5$',linewidth=2)
plt.legend()
plt.xlabel('x')

plt.ylabel('polinômios')
plt.ylim([-30,100])
plt.legend(loc=1)
plt.grid()

plt.subplot(4,2,2)
plt.plot(x,p1,'--k',label='$2*x^4-4*x^2+2$',linewidth=2)
plt.legend()
plt.subplot(4,2,4)
plt.plot(x,p2,':r',label='$-3*x^3+5*x$',linewidth=2)
plt.legend()
plt.subplot(4,2,6)
plt.plot(x,p3,'-b',label='$2*x^2-4*x^2+2$',linewidth=2)
plt.legend()
plt.subplot(4,2,8)
plt.plot(x,p4,'-.m',label='$-3*x+5$',linewidth=2)
plt.legend()
plt.tight_layout()
```



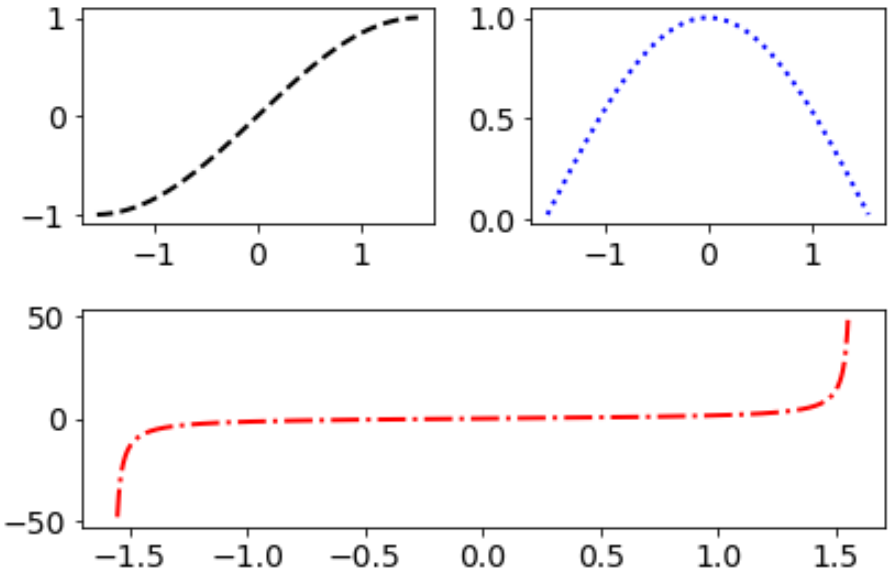




:

```
x = np.array([-3, -2, -1, 0, 1, 2, 3, 4, 5])
y = x*x-2*x+4
plt.plot(x,y,'o--k',linewidth=2,markersize=7)
plt.title('Gráfico bonito!’)
```

:



Python para exatas #02: Gráficos de funções com matplotlib