Notas de aula

Introdução à Física Computacional

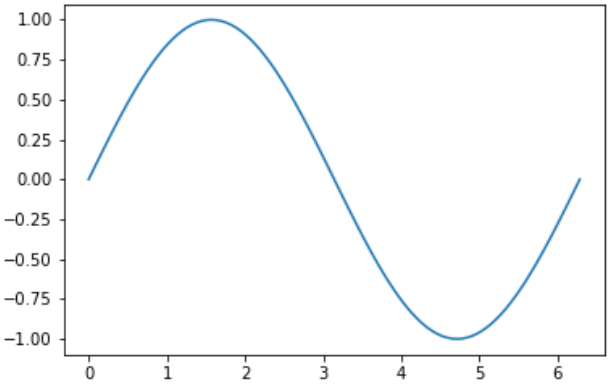
Prof. Gerson – UFU – 2019

Atendimento:

- Sala 1A225
- Email: gersonjferreira@ufu.br
- Webpage: http://gjferreira.wordpress.com
- Horário: sextas-feiras 16:00 – 16:50

Matplotlib

Já vimos a estrutura básica de um plot em matplotlib:




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

x = np.linspace(0, 2*np.pi, 101)
y = np.sin(x)

plt.plot(x, y) # veja o que acontece se trocar por plt.scatter
plt.show()
```

- Veremos a estrutura de bibliotecas externas (imports) mais adiante.
- A biblioteca numpy permite usar as funções matemáticas em vetores.

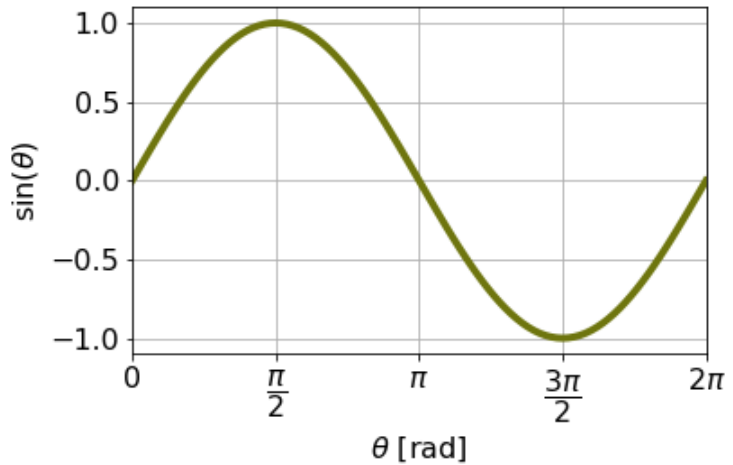
  Aqui, x é um vetor de 101 pontos de 0 a 2π.

  Em y, calculamos o seno para cada ponto de x, e retorna-se um vetor.
- O comando plot do pyplot combina x e y em pares ordenados para fazer o gráfico.

Melhorando a figura

Desafio: ler a documentação! (read the docs!!!)

Objetivo: fazer a figura ficar como esta abaixo



1- mudar cor e espessura da linha

2- definir limites do eixo x de 0 a 2π exatamente

3- exibir grade

4- aumentar o tamanho da fonte

5- labels usando Latex

6- salvar o arquivo automaticamente em PNG

Link para documentação do pyplot:
https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html?highlight=plot#module-matplotlib.pyplot

Código das figuras estarão disponiveis nos próximos slides depois da aula.

Código da figura anterior

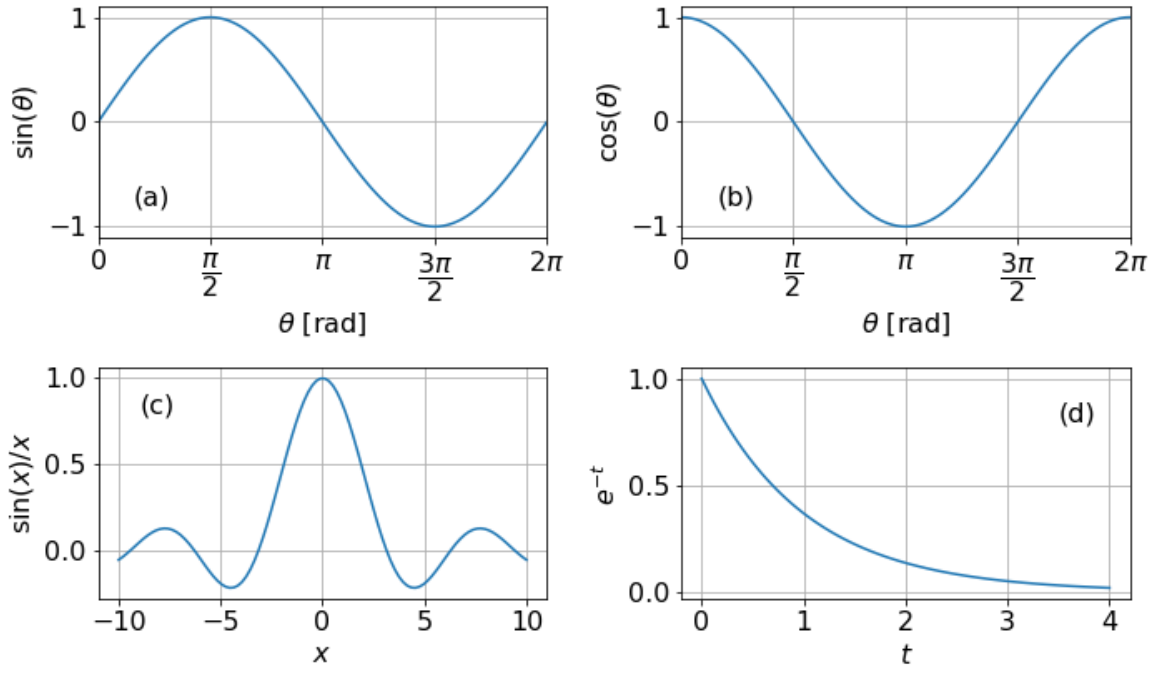
```
import numpy as np
import matplotlib.pyplot as plt
from numpy import pi

x = np.linspace(0, 2*pi,101)
y = np.sin(x)

plt.rcParams.update({'font.size': 16})
plt.plot(x, y, lw=4, c='xkcd:olive')
plt.xlabel(R"$\theta$ [rad]")
plt.ylabel(R"$\sin(\theta)$")
plt.xlim([0, 2*pi])
plt.xticks([0, pi/2, pi, 3*pi/2, 2*pi], ["0", R"$\dfrac{\pi}{2}$", R"$\pi$", R"$\dfrac{3\pi}{2}$", R"$2\pi$"])
plt.grid()
plt.tight_layout()
plt.savefig("seno.png")
plt.show()
```

O que faz o comando abaixo?

plt.tight_layout()

Vários paineis

Veja documentação dos comandos do pyplot: subplot, text



Dica: para mudar o tamanho da figura, use: plt.figure(figsize=(10,6))

→ mais detalhes : documentação ;-)

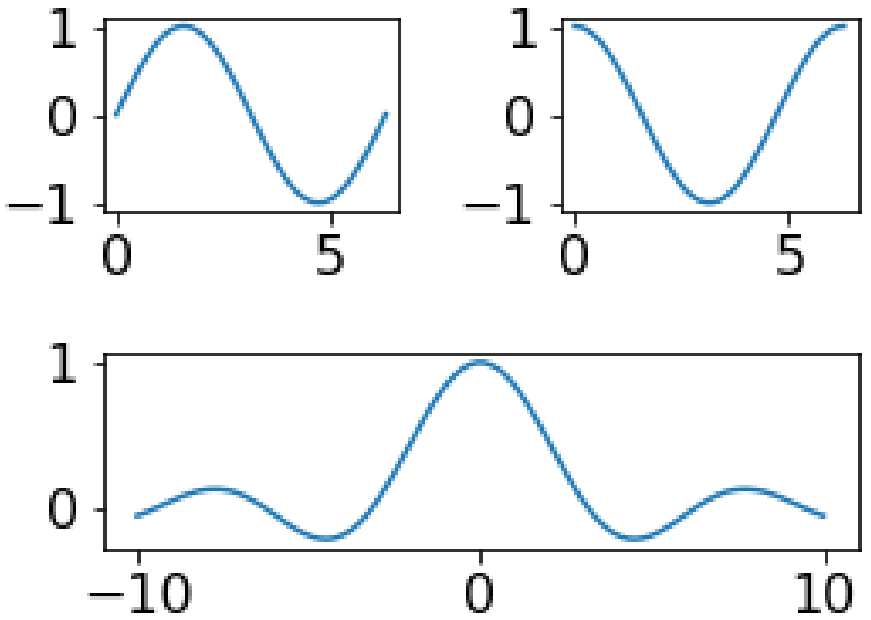
Código da figura anterior

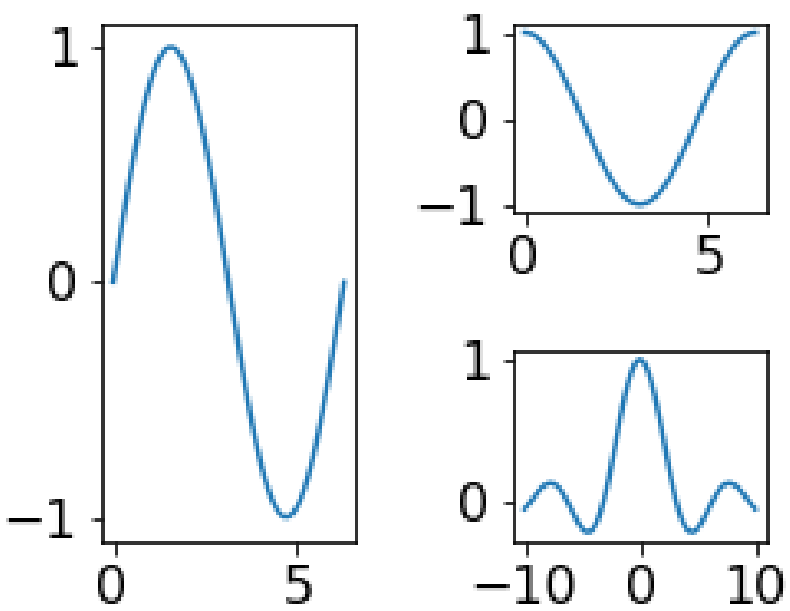
```
import numpy as np
import matplotlib.pyplot as plt
from numpy import pi

plt.figure(figsize=(10,6))
plt.rcParams.update({'font.size': 16})


plt.subplot(2,2,1)
x = np.linspace(0, 2*pi, 101)
y = np.sin(x)
plt.plot(x, y)
plt.text(0.5, -0.8, "(a)")
plt.grid()
plt.xlabel(R"$\theta$ [rad]")
plt.ylabel(R"$\sin(\theta)$")
plt.xlim([0,2*pi])
plt.xticks([0, pi/2, pi, 3*pi/2, 2*pi], ["0", R"$\dfrac{\pi}{2}$", R"$\pi$", R"$\dfrac{3\pi}{2}$", R"$2\pi$"])


plt.subplot(2,2,2)
x = np.linspace(0, 2*pi, 101)
y = np.cos(x)
plt.plot(x, y)
plt.text(0.5, -0.8, "(b)")
plt.grid()
plt.xlabel(R"$\theta$ [rad]")
plt.ylabel(R"$\cos(\theta)$")
plt.xlim([0,2*pi])
plt.xticks([0, pi/2, pi, 3*pi/2, 2*pi], ["0", R"$\dfrac{\pi}{2}$", R"$\pi$", R"$\dfrac{3\pi}{2}$", R"$2\pi$"])
```

```
plt.subplot(2,2,3)
x = np.linspace(-10, 10, 100)
y = np.sin(x)/x
plt.plot(x, y)
plt.text(-9, 0.8, "(c)")
plt.grid()
plt.ylabel(R"$\sin(x)/x$")
plt.xlabel(R"$x$")


plt.subplot(2,2,4)
x = np.linspace(0, 4, 101)
y = np.exp(-x)
plt.plot(x, y)
plt.text(3.5, 0.8, "(d)")
plt.grid()
plt.xlabel(R"$t$")
plt.ylabel(R"$e^{-t}$")


plt.tight_layout()
plt.savefig("paineis.png")
plt.show()
```

Vários paineis – extendendo paineis

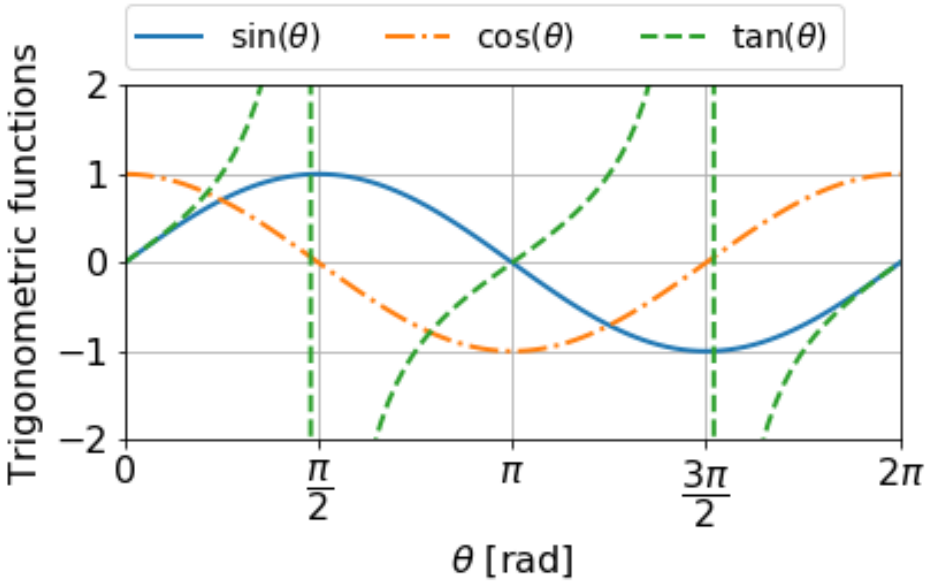
Agora tente fazer as figuras ficarem dispostas destas formas:





Legenda

→ plote as três funções trigonométricas e as indice na legenda



Dica:
O parametro loc pode receber palavras-chave:

exemplo: loc="upper center"

Ou posição explicita com relação ao painel

exemplo: loc=(0,1)

Note que a posição aqui é tratada de forma diferente da do comando plt.text(...)

Formato dos arquivos

Para salvar as figuras em arquivos, usamos o comando

```
plt.savefig("nome_do_arquivo.png")
```

Python automaticamente entende o formato desejado pela extensão:

→ opções usuais: PNG, PDF, SVG

PNG	PDF	SVG
Portable Network Graphics	Portable Document Format	Scalable Vector Graphics
Imagem salva em pedaços com informações dos pixels, como uma foto.	Formato vetorial para documentos, mas pode ser usado para imagens.	Formato vetorial especializado em imagens. Ideal para compor figuras mais complexas.
Prós: tamanho pequeno, fácil compartilhar	Prós: parcialmente editável. Fácil de abrir compartilhar.	Prós: resolução escalável, fácil de editar (inkscape, illustrator, …).
Contra: não pode ser editado. Resolução fixa.	Contra: tamanho do arquivo.	Contra: tamanho do arquivo.

Editando a figura externamente

→ Salve alguma das figuras anteriores em formato SVG

→ Use o programa Inkscape para abri-la, e

troque a cor das linhas
mova elementos de lugar
adicione setas
troque a ordem dos paineis
…

→ Para aprender mais sobre inkscape:

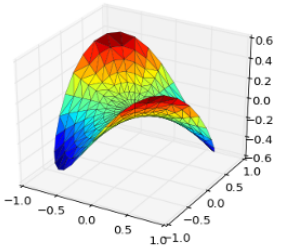
https://inkscape.org/learn/

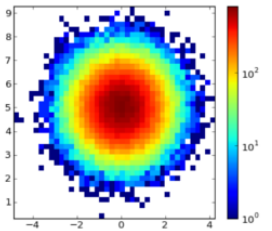
Perguntas / sugestões?

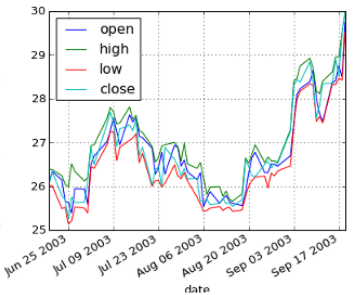
Vimos elementos básicos do matplotlib.pyplot

Há muito mais:

→ figuras 3D
→ animações (gif, mp4)
→ campos vetoriais
→ …







Veremos algumas destas funcionalidades ao longo do curso.

O que você gostaria de tentar fazer com as figuras?