Introducción a Matplotlib

FÍSICA COMPUTACIONAL 2020-2021

1. Introducción a Python

- Python es un lenguage de programación de **interpretado** (no necesita compilación)

- Debido su facilidad de uso y al gran número de bibliotecas disponibles, es **muy usado** en la actualidad en multitud de campos:
  - Ciencia de datos, machine learning, desarrollo web, visualización de datos…

- Al ser un lenguaje interpretado es mucho **menos eficiente** (más lento) que lenguajes compilados como C, C++ o Fortran. A cambio, es mucho más cómodo de usar

1. Python – Sintaxis básica

- No hay que declarar las variables ni asignarles tipos (se asignan de forma dinámica e implícita)
- Las líneas no terminan en “;” y comentarios con “#”
- Las operaciones básicas son similares a otros lenguajes

Condicionales

```
if (condicion1):
    comando1
    comando2
elif (condicion2):
    comando3
elif (condicion3):
    comando4
else:
    comando5
```

Bucles for

```
for var in range(0,10,1):
    comando1
    comando2

for var in [1, 6, “hola”]:
    comando1
    comando2
```

Bucles while

```
while condicion:
    comando1
    comando2
```

Otros comandos básicos:

`print(var)` ← Muestra la variable var

1. Instalación matplotlib

- Instalamos los paquetes necesarios de Python. En la terminal:

```
sudo apt-get update
sudo apt-get install python3-pip
pip3 install numpy scipy matplotlib jupyterlab
echo 'export PATH=PATH=$PATH:~/.local/bin' >> ~/.bash_profile
source ~/.bash_profile
```

- Descargamos el fichero de datos de prueba “dataset.txt”.

- Empezamos abriendo Python en la carpeta en la que tengamos el fichero de datos. Una vez en la carpeta usamos el comando: “python3”

1. Primeros pasos

- Importando dependencias

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

- Cargando datos desde un fichero de texto

```
data = np.loadtxt('dataset.txt')
x = data[: , 0]
y = data[: , 1]
z = data[: , 2]
```

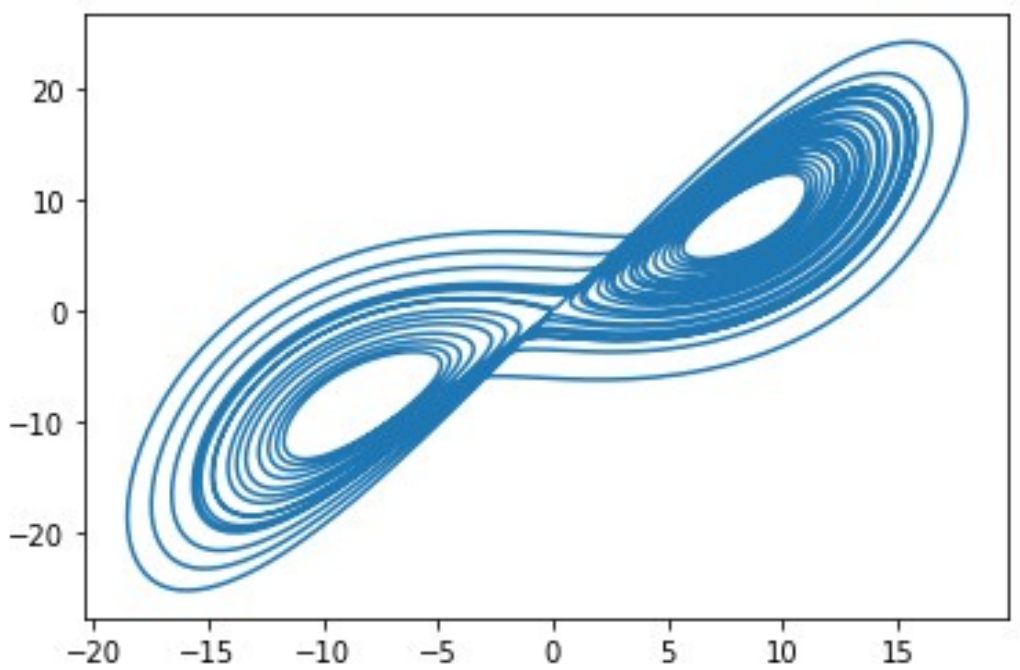
- Nuestra primera figura!

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(x, y)
```

- Para mostrar la figura

```
plt.show( )
```

Cada columna del fichero de texto contiene 6000 puntos correspondientes a una coordenada del atractor de Lorenz.

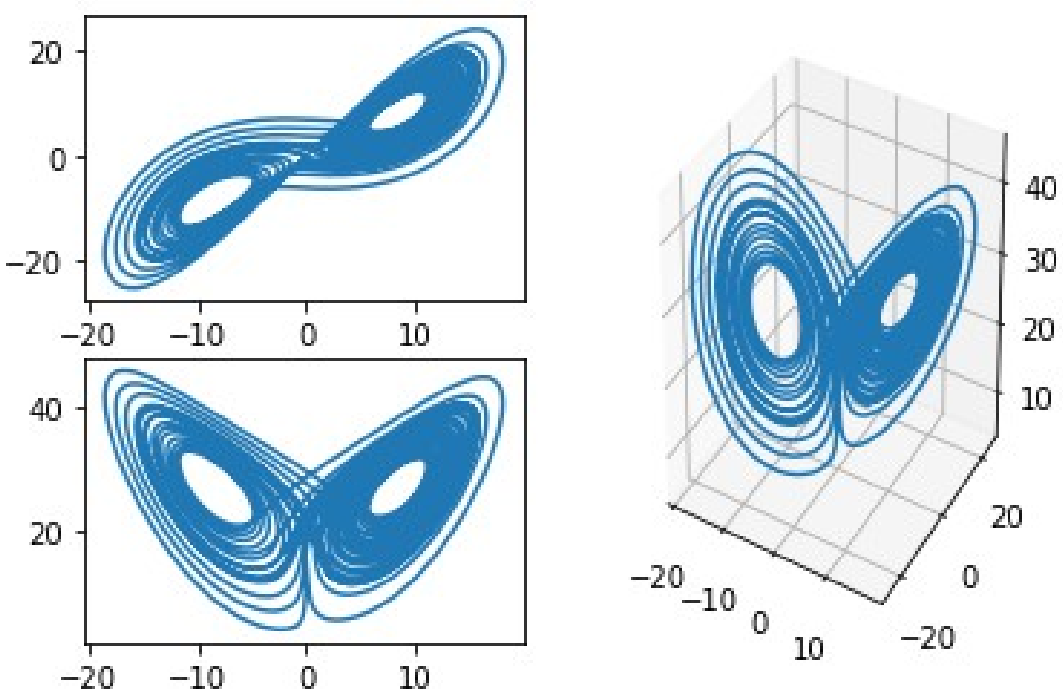


2. Creando subplots

- Nueva figura con el atractor completo y dos proyecciones:

```
fig = plt.figure()
ax1 = fig.add_subplot(“221”)
ax1.plot(x, y)
ax2 = fig.add_subplot(“223”)
ax2.plot(x, z)
ax3 = plt.subplot(“122”, projection='3d')
ax3.plot(x, y, z)
plt.show()
```

plt.subplot(n, m, i): crea los ejes en la posición i de un “grid” con n filas y m columnas.



2. Modificando las propiedades (I)

- Podemos acceder a las propiedades del gráfico mediante su variable correspondiente:

1. Borramos lo que hubiera en el primer y segundo subplots

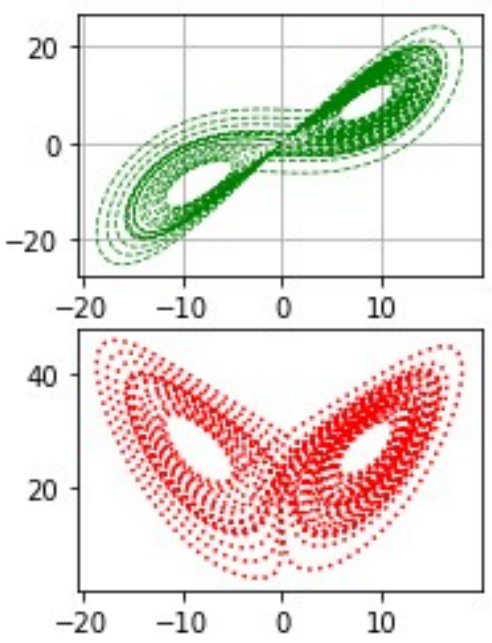
```
ax1.clear()
ax2.clear()
```

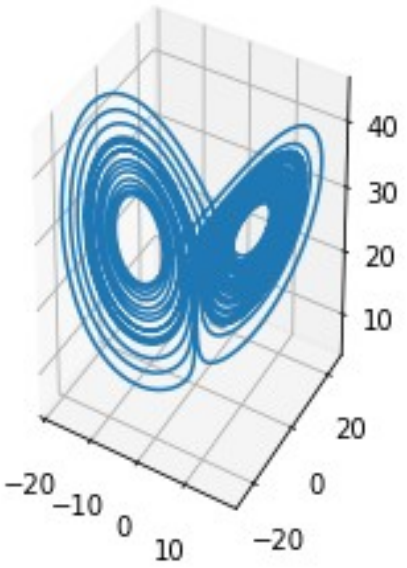
2. Cambiamos el estilo, grosor y color de las líneas

```
ax1.plot(x, y, linestyle=‘dashed’, linewidth=0.75, color='green’)
ax2.plot(x, z, linestyle=‘dotted’, linewidth=1.5, color=‘red’)
```

3. Añadimos un grid al primer subplot

```
ax1.grid(True)
```





2. Modificando las propiedades (II)

4. Cambiamos los límites del segundo subplot

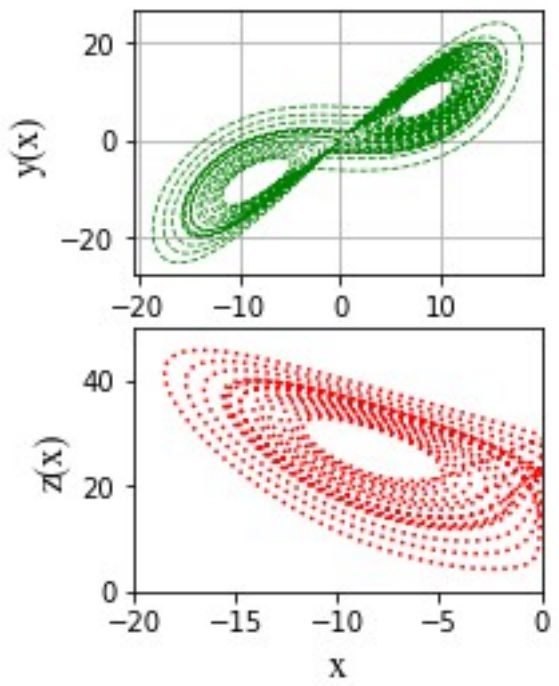
```
ax2.set_xlim(-20,0)
ax2.set_ylim(0,50)
```

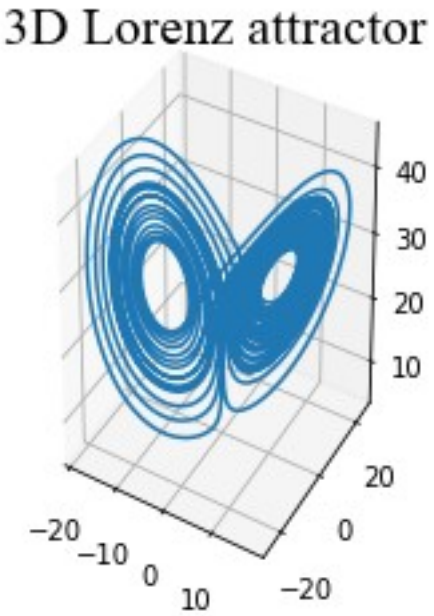
5. Añadimos etiquetas para los ejes x e y

```
ax1.set_ylabel("y(x)", fontsize=14, fontname="Times New Roman")
ax1.set_xlabel("x", fontsize=14, fontname="Times New Roman")
ax2.set_ylabel("z(x)", fontsize=14, fontname="Times New Roman")
ax2.set_xlabel("x", fontsize=14, fontname="Times New Roman")
```

6. Añadimos un título para la gráfica 3D

```
ax3.set_title("3D attractor", fontsize=20, fontname="Times New Roman")
```





¡Los ejes quedan demasiado juntos y se solapan!

3. Guardando en formato imagen

El siguiente comando, aplicado a la figura completa, ajusta automáticamente los ejes y el texto:

fig.tight_layout()

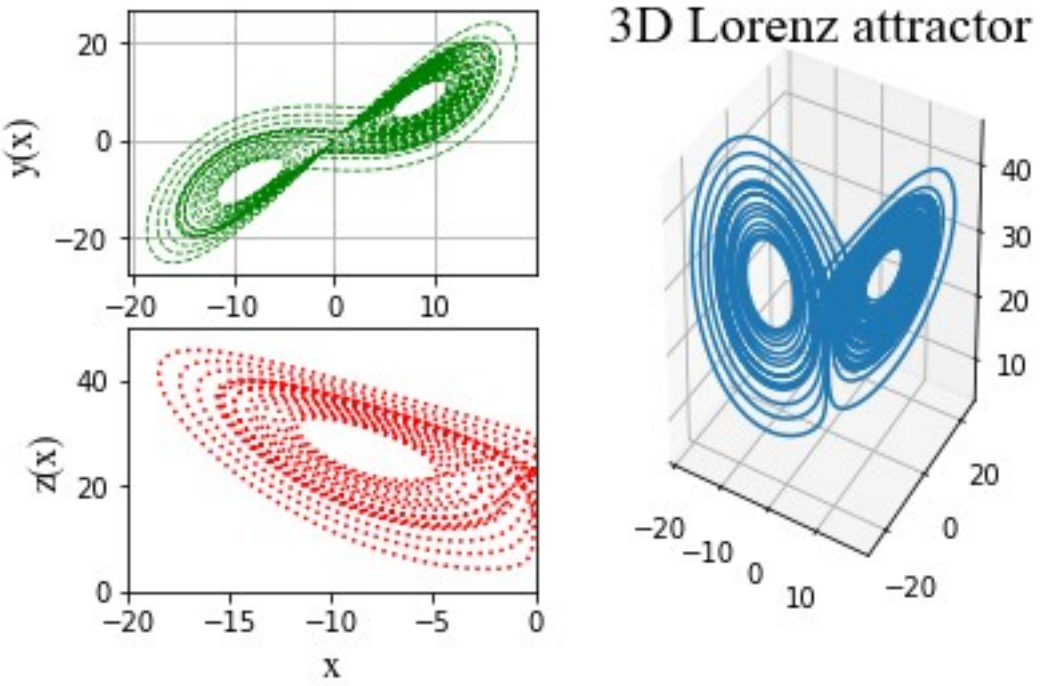
Para guardar nuestra gráfica con buena resolución:

fig.savefig('filename.png', dpi=300)

Alternativa: formato vectorial (.pdf o .svg)

fig.savefig('filename.svg')

fig.savefig(‘filename.pdf’)



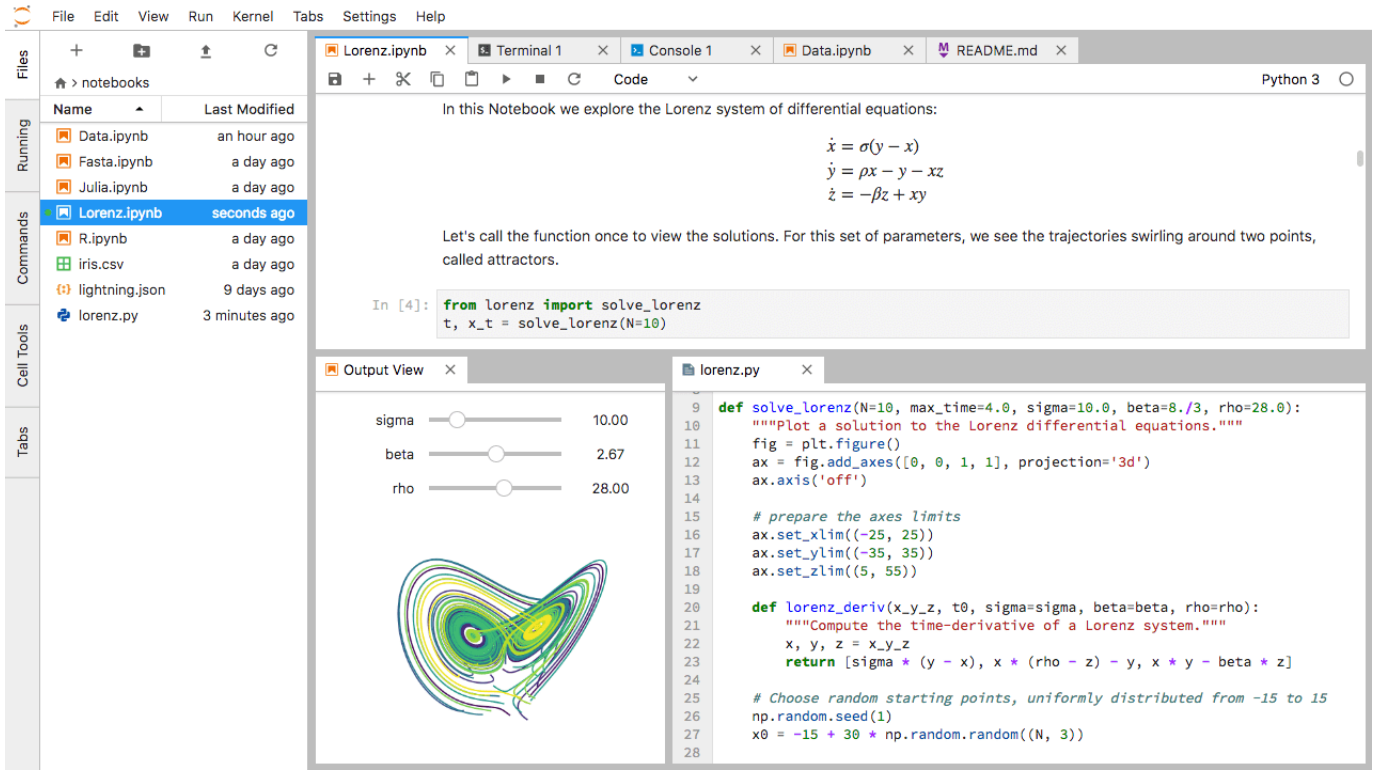
3. Alternativas a la terminal

Es muy útil usar la aplicación JupyterLab para visualizar el código y los resultados.

Para abrirlo, escribid en la terminal:

```
jupyter lab
```

(se abre en el navegador)



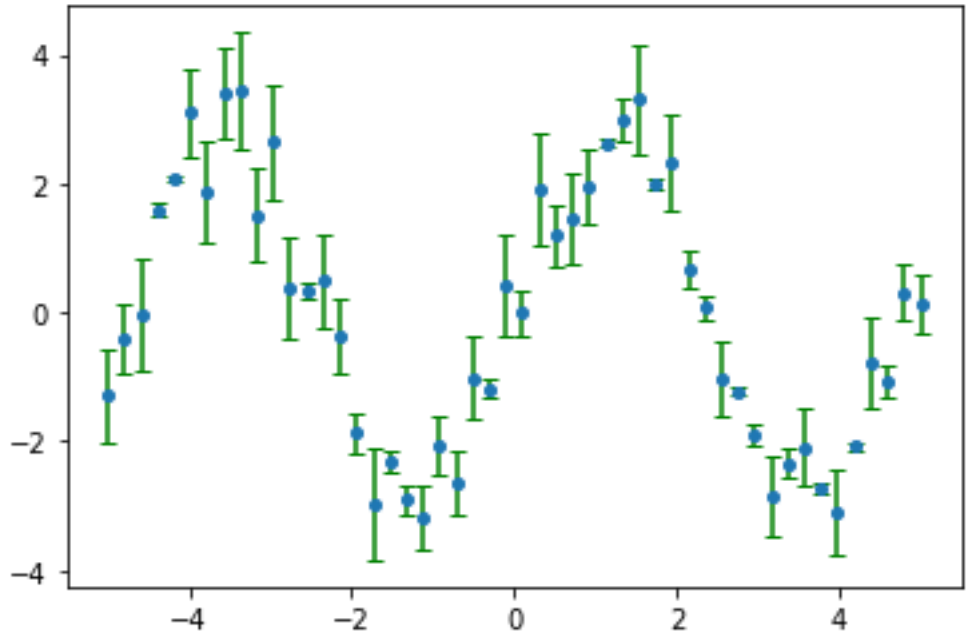
4. Ajustes y errores (I)

Generamos 50 puntos de sin(x) con “ruido”:

```
x = np.linspace(-5,5,50)   # 50 puntos equiespacidos entre -5 y 5
noise = 2*(np.random.rand(50) - 0.5)    # 50 números aleatorios en [-1,1]
y = 2.7*np.sin(1.3*x) + noise
```

Usamos la función *errorbar* para pintar puntos con barras de error

```
fig, ax = plt.subplots(1,1)
ax.errorbar(x, y, noise, linestyle='', capsize = 3, ecolor='g',
            marker='o', markersize=4, label='Data')
```



“label” permite definir el nombre de cada conjunto de datos en la leyenda.

4. Ajustes y errores (II)

Importamos dependencias y definimos la función a ajustar:

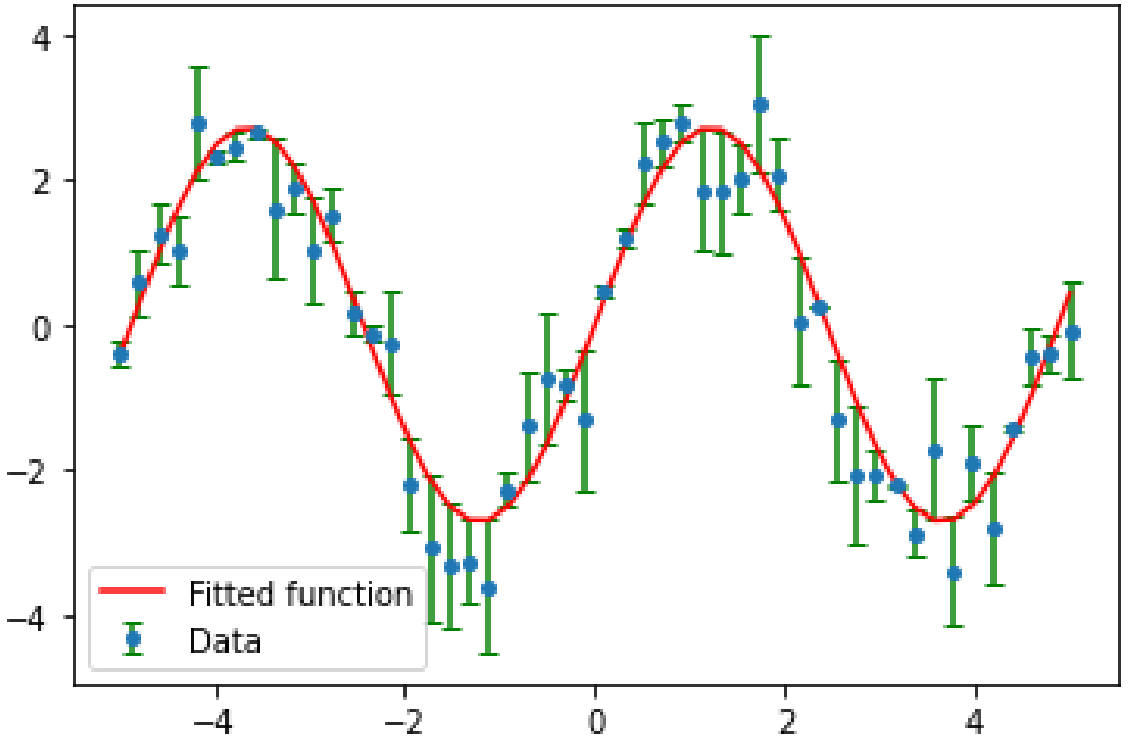
```
from scipy import optimize
def fit_func(x, a, b):
    return a * np.sin(b * x)
```

Ajustamos con curve_fit:

```
params, _ = optimize.curve_fit(fit_func, x, y)
print(params)
```

Representamos el ajuste y mostramos la leyenda

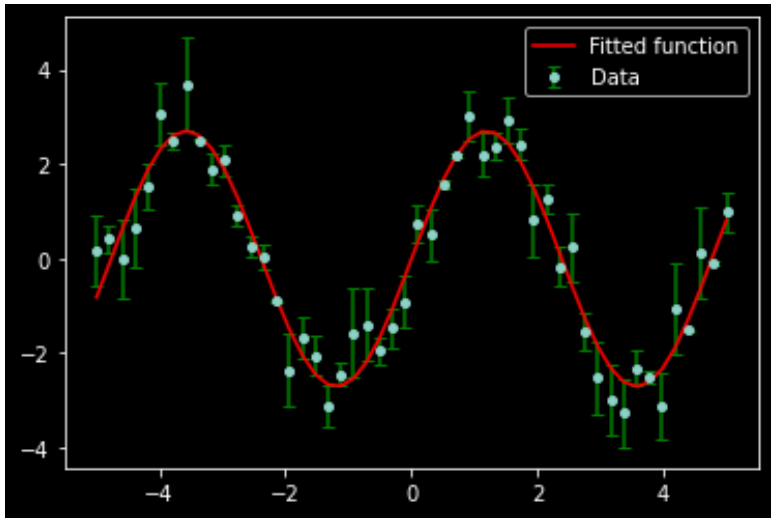
```
ax.plot(x, test_func(x, params[0], params[1]), color='r’,
        label='Fitted function')
ax.legend(loc='best')
```



params = [2.71611 1.28794]

5. Cambiando el estilo

Matplotlib incluye diferentes “estilos” que podemos elegir antes de empezar a graficar mediante el comando *style.use()*.



plt.style.use(‘dark_background’)

```
axes.labelsize: 10
axes.titlesize: 11
font.size: 10
legend.fontsize: 8

axes.grid: True
grid.color: gray
grid.alpha: 0.2

xtick.labelsize      : 9
ytick.labelsize      : 9
xtick.major.size     : 4
xtick.minor.size     : 2
xtick.minor.visible  : True

axes.spines.right    : False
axes.spines.top      : False
```

Además, podemos definir nuestros propios “estilos” en un fichero con extension **.mplstyle.patch**.
Haced print(plt.rcParams.keys()) para ver todas las opciones disponibles!

Páginas de interés

- Tutoriales sobre distintos aspectos de matplotlib:

  https://matplotlib.org/3.3.4/tutorials/index.html

- Una lista con todos los estilos disponibles por defecto:

  https://matplotlib.org/3.1.1/gallery/style_sheets/style_sheets_reference.html

- Galería con infinidad de ejemplos y su correspondiente código:

  https://matplotlib.org/stable/gallery/index.html