Guía de laboratorio 4: Resolución de Sistemas de Ecuaciones No Lineales.

1. Introducción

Los sistemas de ecuaciones no lineales aparecen en diferentes áreas de la ciencia, la ingeniería, la biología, economía y física. A diferencia de los sistemas lineales, la mayoría de estos no pueden resolverse de forma analítica cerrada, por lo que es necesario aplicar métodos numéricos iterativos. En esta práctica se resolverán distintos sistemas no lineales de 2 y 3 variables empleando el método de Newton-Raphson multivariable y la función *fsolve* de *scipy* Python, además de representar gráficamente las soluciones en el caso de dos variables.

2. Objetivos

- Comprender la formulación de sistemas de ecuaciones no lineales en 2 y 3 variables.
- Implementar modelos computacionales en Python para resolver sistemas no lineales.
- Graficar los resultados obtenidos en sistemas de 2 variables utilizando Matplotlib.
- Analizar la convergencia de los métodos y la influencia de las condiciones iniciales.

3. Materiales y Recursos

- Computadora personal o laboratorio de cómputo.
- Python 3.10+ instalado (preferible en Anaconda).
- Librería *Numpy*, *Matplotlib*, *Scipy* instalada (pip install *nombre librería*).
- Editor de código (Sublime Text, Visual Studio Code o Jupyter Notebook).
- Opcional (crear un ambiente virtual)

4. Ejercicios y Desarrollo

Se plantearon 10 sistemas de ecuaciones no lineales, 5 con dos variables y 5 con tres variables. Para cada sistema se definieron las funciones correspondientes en *Python* y se resolvieron utilizando la función *fsolve* de la librería *scipy*. En el caso de los sistemas con dos variables, se realizaron gráficas de las curvas que representan las ecuaciones, de manera que las soluciones aparecen como puntos de intersección.

4.1 Sistemas de Ecuaciones No lineales de 2 variables

1. $x^2 + y^2 = 4, \quad e^x + y = 1$

2. $\sin(x) + y^2 = 1, \quad x^2 + y = 2$

3. $x^3 - y = 0, \quad x + y^2 = 4$

4. $e^x + y = 3, \quad x^2 + y^2 = 5$

5. $\ln(x + 2) + y = 1, \quad x^2 - y = 2$

a. **Realizar Pseudocódigo**
b. **Codificar en Python**

Ejercicio 1

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fsolve

def ej1(vars):
    x, y = vars
    return [x**2 + y**2 - 4, np.exp(x) + y - 1]

# Resolver
sol1 = fsolve(ej1, [0.5, 0.5])
print("Ejercicio 1:", sol1)

# Graficar
x = np.linspace(-3, 3, 400)
y = np.linspace(-3, 3, 400)
X, Y = np.meshgrid(x, y)

plt.contour(X, Y, X**2 + Y**2 - 4, [0], colors='r')
plt.contour(X, Y, np.exp(X) + Y - 1, [0], colors='b')
plt.plot(sol1[0], sol1[1], 'go')
plt.title("Ejercicio 1")
plt.xlabel("x"); plt.ylabel("y")
plt.grid(True); plt.show()
```

Ejercicio 2

```
#EJERCICIO 2

def ej2(vars):
    x, y = vars
    return [np.sin(x) + y**2 - 1, x**2 + y - 2]

# Resolver
sol2 = fsolve(ej2, [1, 1])
print("Ejercicio 2:", sol2)

# Graficar
plt.contour(X, Y, np.sin(X) + Y**2 - 1, [0], colors='r')
plt.contour(X, Y, X**2 + Y - 2, [0], colors='b')
plt.plot(sol2[0], sol2[1], 'go')
plt.title("Ejercicio 2")
plt.xlabel("x"); plt.ylabel("y")
plt.grid(True); plt.show()
```

Ejercicio 3

```python
#EJERCICIO 3

def ej3(vars):
    x, y = vars
    return [x**3 - y, x + y**2 - 4]

# Resolver
sol3 = fsolve(ej3, [1, 1])
print("Ejercicio 3:", sol3)

# Graficar
plt.contour(X, Y, X**3 - Y, [0], colors='r')
plt.contour(X, Y, X + Y**2 - 4, [0], colors='b')
plt.plot(sol3[0], sol3[1], 'go')
plt.title("Ejercicio 3")
plt.xlabel("x"); plt.ylabel("y")
plt.grid(True); plt.show()
```

NOTA: Complete la codificación en los ejercicios 4 y 5

4.2 Sistemas de Ecuaciones No lineales de 3 variables

6. $x^2 + y + z = 4, \quad x + y^2 + z = 5, \quad x + y + z^2 = 6$

7. $e^x + y + z = 7, \quad x^2 + y^2 = 4, \quad z - y = 1$

8. $x^2 + y^2 + z^2 = 9, \quad xy = 2, \quad x + z = 3$

9. $\sin(x) + y = 2, \quad y^2 + z = 3, \quad x + z^2 = 4$

10. $x^2 + y = 1, \quad y^2 + z = 2, \quad z^2 + x = 3$

c. **Realizar Pseudocódigo**
d. **Codificar en Python**

Ejercicio 6

```python
# EJERCICIO 6 (SIN USAR FUNCION DEF)
import numpy as np
from scipy.optimize import fsolve

sistema = lambda v: [
    v[0]**2 + v[1] + v[2] - 4,
    v[1]**2 + v[2] + v[0] - 5,
    v[2]**2 + v[0] + v[1] - 6
]

sol = fsolve(sistema, [1, 1, 1])
print("Ejercicio 6 - solución aproximada (x,y,z) =", sol)
```

Ejercicio 7

```python
# EJERCICIO 7 (SIN USAR FUNCION DEF)
import numpy as np
from scipy.optimize import fsolve

sistema = lambda v: [
    np.exp(v[0]) + v[1]**2 + v[2] - 7,
    v[0] + np.exp(v[1]) + v[2]**2 - 8,
    v[0]**2 + v[1] + np.exp(v[2]) - 9
]

sol = fsolve(sistema, [1, 1, 1])
print("Ejercicio 7 - solución aproximada (x,y,z) =", sol)
```

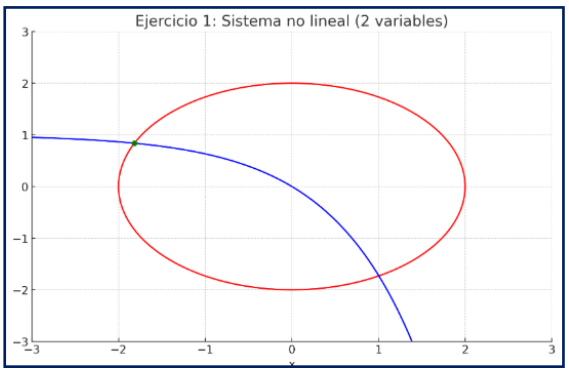
Ejercicio 8

```python
# EJERCICIO 8 (SIN USAR FUNCIÓN DEF)
import numpy as np
from scipy.optimize import fsolve

sistema = lambda v: [
    np.sin(v[0]) + v[1] + v[2]**2 - 3,
    v[0] + np.cos(v[1]) + v[2] - 2,
    v[0]**2 + v[1]**2 + v[2] - 4
]

sol = fsolve(sistema, [1, 1, 1])
print("Ejercicio 8 - solución aproximada (x,y,z) =", sol)
```

NOTA: Complete la codificación en los ejercicios 9 y 10 (usando función DEF)

4.3 Resultados (Máximo 10 líneas en texto)

4.4 Pegar las 5 gráficas generadas de las ecuaciones No lineales de 2 variables (Que encajen en los recuadros.



4.5 Conclusiones (Máximo en 10 líneas)