AIN SHAMS UNIVERSITY
FACULTY OF ENGINEERING
International Credit Hours Programs (ICHEP)
Mechatronics and Automation

Computational Intelligence (CSE473)

Lab 4 Report

Name	ID	Section
Mahmoud Elsayd Abdelqader Labib Eldwakhly	21P0017	1

Submitted to Dr Hossam Hassan & Eng. Ali Ahmed

Fall 2025

1. Introduction

In this assignment, our goal is to solve a system of three nonlinear equations:

$$g_1(x_1, x_2, x_3) = 3x_1 - \cos(x_2 x_3) - 0.5 = 0$$
$$g_2(x_1, x_2, x_3) = x_1^2 - 81(x_2 + 0.1)^2 + \sin(x_3) + 1.06 = 0$$
$$g_3(x_1, x_2, x_3) = e^{-x_1 x_2} + 20x_3 + \frac{10\pi - 3}{3} = 0$$

Since these equations are nonlinear, solving them analytically is difficult. Instead, we convert the problem into a nonlinear least-squares minimization problem.

We define the objective:

$$F(\mathbf{x}) = \frac{1}{2}(g_1^2 + g_2^2 + g_3^2)$$

Minimizing F drives each equation $g_i(\mathrm{x})$ to zero.

We solve this using SciPy's least squares method (Levenberg–Marquardt algorithm), which is widely used for nonlinear systems, curve fitting, and inverse problems.

2. Mathematical Approach

2.1 Converting to a Least-Squares Problem

Instead of solving:

$$g_i(\mathbf{x}) = 0$$

we minimize:

$$F(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{3} g_i(\mathbf{x})^2$$

This converts root-finding → optimization.

If the global minimum satisfies $F(\mathbf{x}) = 0$, then:

$$g_1 = g_2 = g_3 = 0$$

and we have a solution.

3. Optimization Method Used

We use:
method = 'lm'
which refers to Levenberg–Marquardt, a hybrid between:

- Gradient descent (stable)
- Gauss–Newton (fast near solution)

LM is the best choice for small systems of nonlinear equations when no bounds are required.

4. Code Implemented

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

# ------------------------------------------------------------
# 1. Define the nonlinear equations g1, g2, g3
# ------------------------------------------------------------
def g(x):
    x1, x2, x3 = x
    g1 = 3*x1 - np.cos(x2*x3) - 0.5
    g2 = x1**2 - 81*(x2 + 0.1)**2 + np.sin(x3) + 1.06
    g3 = np.exp(-x1*x2) + 20*x3 + (10*np.pi - 3)/3
    return np.array([g1, g2, g3])

# ------------------------------------------------------------
# 2. Objective function (least squares minimization)
# ------------------------------------------------------------
def F(x):
    return g(x)
```

```
def F(x):
    return g(x)

# ------------------------------------------------------------
# 3. Initial guess
# ------------------------------------------------------------
x0 = np.array([0.1, 0.1, -0.1])

# ------------------------------------------------------------
# 4. Solve using SciPy Levenberg-Marquardt method
# ------------------------------------------------------------
res = least_squares(F, x0, method='lm')

print("Solution x =", res.x)
print("Residuals g(x) =", res.fun)
print("Cost (1/2 * sum g^2) =", res.cost)
print("Solver Message:", res.message)

#
```

```python
# ------------------------------------------------------------
# 5. 3D Surface Plot of g1(x1, x2) at solved x3
# ------------------------------------------------------------
x1_vals = np.linspace(-1, 2, 80)
x2_vals = np.linspace(-1, 1, 80)
X1, X2 = np.meshgrid(x1_vals, x2_vals)

x3_sol = res.x[2]
G1 = 3*X1 - np.cos(X2 * x3_sol) - 0.5

fig = plt.figure(figsize=(10,6))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X1, X2, G1, cmap='viridis', alpha=0.8)
ax.scatter(res.x[0], res.x[1], 0, color="r", s=80, label="Solution")
ax.set_xlabel("x1")
ax.set_ylabel("x2")
ax.set_zlabel("g1")
plt.title("3D surface of g1(x1, x2) at solved x3")
plt.legend()
plt.show()
```

5. Results