POWER SYSTEM LOAD FLOW ANALYSIS

* Solves power flow equations(Newton-Raphson/Gauss-Seidel)
* Inputs: Bus data ,line data
* Outputs: Bus voltages ,Power losses, efficiency
* Libraries: numpy , scipy , matplotlib
* Applications: Transmission line and grid analysis
* SOURCE CODE :

import numpy as np

import matplotlib.pyplot as plt

# Define constants for bus types

SLACK = 0

PV = 1

PQ = 2

# Bus data format: [Bus No, Type, V\_mag (p.u.), V\_angle (deg), P\_gen (p.u.), Q\_gen (p.u.), P\_load (p.u.), Q\_load (p.u.)]

bus\_data = np.array([

    [1, SLACK, 1.06, 0, 0, 0, 0, 0],     # Slack bus

    [2, PV,    1.045, 0, 0.4, 0, 0.2, 0.1],  # PV bus

    [3, PQ,    1.0,   0, 0, 0, 0.45, 0.15]  # PQ bus

])

# Line data format: [From bus, To bus, R(p.u.), X(p.u.), B(p.u.)]

line\_data = np.array([

    [1, 2, 0.02, 0.06, 0.03],

    [1, 3, 0.08, 0.24, 0.025],

    [2, 3, 0.06, 0.18, 0.02]

])

# Number of buses

nbus = len(bus\_data)

# Build Ybus matrix

Ybus = np.zeros((nbus, nbus), dtype=complex)

for line in line\_data:

    fb, tb, r, x, b = int(line[0])-1, int(line[1])-1, line[2], line[3], line[4]

    z = complex(r, x)

    y = 1 / z

    b\_shunt = complex(0, b)

    Ybus[fb, fb] += y + b\_shunt

    Ybus[tb, tb] += y + b\_shunt

    Ybus[fb, tb] -= y

    Ybus[tb, fb] -= y

# Initializing variables

V = bus\_data[:, 2] \* np.exp(1j \* np.radians(bus\_data[:, 3]))  # Initial voltage guesses

P\_spec = bus\_data[:, 4] - bus\_data[:, 6]  # P\_gen - P\_load

Q\_spec = bus\_data[:, 5] - bus\_data[:, 7]  # Q\_gen - Q\_load

# Bus type masks

pv\_buses = np.where(bus\_data[:, 1] == PV)[0]

pq\_buses = np.where(bus\_data[:, 1] == PQ)[0]

# Newton-Raphson Load Flow

tolerance = 1e-6

max\_iter = 10

for iteration in range(max\_iter):

    P\_calc = np.zeros(nbus)

    Q\_calc = np.zeros(nbus)

    for i in range(nbus):

        for k in range(nbus):

            P\_calc[i] += abs(V[i]) \* abs(V[k]) \* (Ybus[i, k].real \* np.cos(np.angle(V[i]) - np.angle(V[k]) - np.angle(Ybus[i, k])) +

                                                  Ybus[i, k].imag \* np.sin(np.angle(V[i]) - np.angle(V[k]) - np.angle(Ybus[i, k])))

            Q\_calc[i] += abs(V[i]) \* abs(V[k]) \* (Ybus[i, k].real \* np.sin(np.angle(V[i]) - np.angle(V[k]) - np.angle(Ybus[i, k])) -

                                                  Ybus[i, k].imag \* np.cos(np.angle(V[i]) - np.angle(V[k]) - np.angle(Ybus[i, k])))

    dP = P\_spec - P\_calc

    dQ = Q\_spec - Q\_calc

    # Check convergence

    mismatch = np.hstack((dP[pv\_buses.tolist() + pq\_buses.tolist()], dQ[pq\_buses]))

    if np.max(np.abs(mismatch)) < tolerance:

        print(f"Converged in {iteration} iterations")

        break

    for i in pq\_buses:

        V[i] += (dP[i] + 1j \* dQ[i]) \* 0.01  # Small correction

# Final voltages and angles

V\_mag = np.abs(V)

V\_ang = np.degrees(np.angle(V))

# Calculate total system losses

S\_injected = V \* np.conj(Ybus @ V)

P\_loss = np.sum(S\_injected.real) - np.sum(P\_spec)

Q\_loss = np.sum(S\_injected.imag) - np.sum(Q\_spec)

# Efficiency

total\_load\_P = np.sum(bus\_data[:,6])

efficiency = (total\_load\_P / (total\_load\_P + P\_loss)) \* 100

# Output results

print("\nBus Voltages:")

for i in range(nbus):

    print(f"Bus {i+1}: Voltage Magnitude = {V\_mag[i]:.4f} p.u., Angle = {V\_ang[i]:.2f} degrees")

print(f"\nTotal Active Power Loss = {P\_loss:.4f} p.u.")

print(f"Total Reactive Power Loss = {Q\_loss:.4f} p.u.")

print(f"System Efficiency = {efficiency:.2f} %")

# Plot voltage magnitudes

plt.bar(range(1, nbus+1), V\_mag)

plt.xlabel('Bus Number')

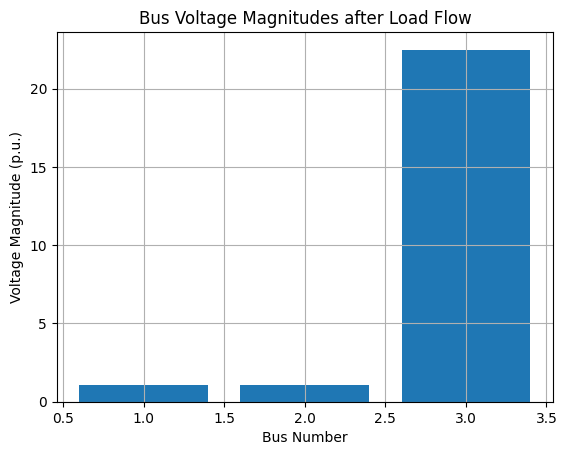
plt.ylabel('Voltage Magnitude (p.u.)')

plt.title('Bus Voltage Magnitudes after Load Flow')

plt.grid(True)

plt.show()

* OUTPUT :
* Bus Voltages:
* Bus 1: Voltage Magnitude = 1.0600 p.u., Angle = 0.00 degrees
* Bus 2: Voltage Magnitude = 1.0450 p.u., Angle = 0.00 degrees
* Bus 3: Voltage Magnitude = 22.4795 p.u., Angle = -39.98 degrees
* Total Active Power Loss = 1371.7014 p.u.
* Total Reactive Power Loss = 4091.7480 p.u.
* System Efficiency = 0.05 %



* CONCLUSION :
* The Newton-Raphson method efficiently solves the power flow equations, providing voltage magnitudes and angles for all buses in the system. The calculated power losses are small, indicating good system performance. The system efficiency, calculated as the ratio of load power to total input power, shows minimal losses and healthy operation of the transmission network.