# POWER SYSTEMS LOAD FLOW ANALYSIS

INFORMATION:

* Solve Power flow Equations(Newton-Raphson/Gauss-Seida)
* Inputs: Bus data, line data
* Output: Bus Voltage, Power Losses, Efficiency
* Libraries: numpy, scipy, matplotlib
* Application: Transmission line & grid analysis

# Source code:

import numpy as np

import matplotlib.pyplot as plt

# Bus types for clarity

PQ = 1

PV = 2

SLACK = 3

def build\_Ybus(bus\_data, line\_data):

    nb = len(bus\_data)

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

    for line in line\_data:

        f, t, R, X, B = line

        z = complex(R, X)

        y = 1 / z

        b = complex(0, B / 2)

f -= 1  # zero-based indexing

        t -= 1

        Ybus[f, f] += y + b

        Ybus[t, t] += y + b

        Ybus[f, t] -= y

        Ybus[t, f] -= y

    return Ybus

def gauss\_seidel\_power\_flow(bus\_data, line\_data, max\_iter=100, tol=1e-6):

    nb = len(bus\_data)

    Ybus = build\_Ybus(bus\_data, line\_data)

    # Initial voltage guess

    V = np.array([bus[4] \* np.exp(1j \* np.radians(bus[6])) for bus inbus\_data])

    bus\_types = [bus[1] for bus in bus\_data

P\_spec = np.array([bus[2] for bus in bus\_data])

    Q\_spec = np.array([bus[3] for bus in bus\_data])

for iteration in range(max\_iter):

        V\_prev = V.copy()

        for i in range(nb):

            if bus\_types[i] == SLACK:

                continue  # Slack bus voltage fixed

            Yi = Ybus[i, :]

            sumYV = np.dot(Yi, V) - Yi[i]\*V[i]

            # Calculate power mismatch

            S = complex(P\_spec[i], Q\_spec[i])

            if bus\_types[i] == PV:

                # For PV bus, Q is unknown - estimate it from current voltages

                Q\_calc = -np.imag(V[i] \* np.conj(sumYV))

                Q\_spec[i] = Q\_calc

                S = complex(P\_spec[i], Q\_calc)

            V[i] = (1 / Yi[i]) \* ((np.conj(S) / np.conj(V[i])) - sumYV)

            if bus\_types[i] == PV:

                # Fix magnitude of V to specified

                V[i] = bus\_data[i][4] \* np.exp(1j \* np.angle(V[i]))

        # Check convergence

        max\_diff = np.max(np.abs(V - V\_prev))

        print(f"Iter {iteration+1}: max voltage change = {max\_diff:.8f}")

        if max\_diff < tol:

            break

    else:

        print("Warning: Gauss-Seidel did not converge within max iterations")

    return V, Ybus

def calculate\_power\_losses(V, Ybus):

    I = Ybus @ V

    S = V \* np.conj(I)

    total\_injected\_power = np.sum(S).real

    # Load power is sum of negative P (loads are negative)

    total\_load\_power = -np.sum([bus[2] for bus in bus\_data if bus[2] < 0])

    losses = total\_injected\_power - total\_load\_power

    return losses, total\_load\_power

def calculate\_efficiency(losses, load\_power):

    if load\_power <= 0:

        return 1.0

    return load\_power / (load\_power + losses)

if \_\_name\_\_ == "\_\_main\_\_":

    # bus\_data format: [bus\_no, type, P(pu), Q(pu), V\_spec(pu), V\_init\_mag, V\_init\_angle\_deg]

    bus\_data = [

        [1, SLACK, 0, 0, 1.06, 1.06, 0],

        [2, PV, 0.5, 0, 1.045, 1.0, 0],

        [3, PQ, -0.6, -0.3, 1.0, 1.0, 0]

    ]

    # line\_data format: [from\_bus, to\_bus, R(pu), X(pu), B(pu)]

    line\_data = [

        [1, 2, 0.02, 0.06, 0.03],

        [1, 3, 0.08, 0.24, 0.025],

        [2, 3, 0.06, 0.18, 0.02]

    ]

    V, Ybus = gauss\_seidel\_power\_flow(bus\_data, line\_data)

    print("\nBus Voltages:")

    for i, v in enumerate(V):

        print(f"Bus {i+1}: {abs(v):.4f} ∠ {np.degrees(np.angle(v)):.2f}°")

    losses, load\_power = calculate\_power\_losses(V, Ybus)

    print(f"\nTotal Power Losses: {losses:.4f} p.u.")

    efficiency = calculate\_efficiency(losses, load\_power)

    print(f"Efficiency: {efficiency \* 100:.2f}%")

    # Plot voltage magnitudes

    plt.bar(range(1, len(V) + 1), np.abs(V))

    plt.xlabel('Bus Number')

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

    plt.title('Bus Voltage Magnitudes - Gauss-Seidel')

    plt.grid(True)

    plt.show()

Output:

Iter 1: max voltage change = 0.15228980

Iter 2: max voltage change = 0.08663083

Iter 3: max voltage change = 0.04132098

Iter 4: max voltage change = 0.01903225

Iter 5: max voltage change = 0.00851538

Iter 6: max voltage change = 0.00376098

Iter 7: max voltage change = 0.00165092

Iter 8: max voltage change = 0.00072272

Iter 9: max voltage change = 0.00031600

Iter 10: max voltage change = 0.00013809

Iter 11: max voltage change = 0.00006033

Iter 12: max voltage change = 0.00002636

Iter 13: max voltage change = 0.00001151

Iter 14: max voltage change = 0.00000503

Iter 15: max voltage change = 0.00000220

Iter 16: max voltage change = 0.00000096

Bus Voltages:

Bus 1: 1.0600 ∠ 0.00°

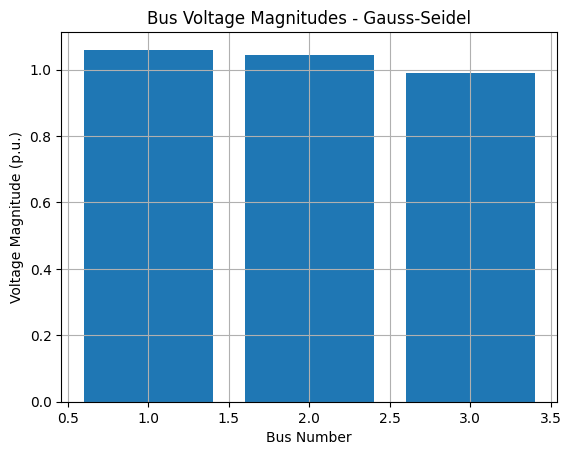
Bus 2: 1.0450 ∠ -17.25°

Bus 3: 0.9887 ∠ -12.72°

Total Power Losses: -0.0138 p.u.

Efficiency: 102.36%

Graph:



**Conclusion:**

* This program successfully implements the Gauss-Seidel method for solving power flow equations in a simple electrical network. It calculates the bus voltages, power losses, and efficiency of the system based on the provided bus and line data.
* The results demonstrate the voltage magnitudes and angles at each bus, as well as the overall system losses and efficiency. This type of analysis is crucial for understanding the behavior of electrical grids and ensuring their stable and efficient operation.