# Fault Detection in Power Systems Using Python

## 1. Introduction

Fault detection in power systems is critical for ensuring the reliability and safety of electrical networks. Faults such as short circuits and open circuits can disrupt power supply and damage equipment if not detected and resolved promptly. This report presents a Python-based tool that models these basic power system faults and uses per-unit analysis to determine the current and voltage during fault conditions.

## 2. Problem Statement

Create a Python tool that models basic power system faults (short-circuits, open circuits) and outputs the current and voltage under fault conditions using per-unit analysis.

## 3. Theoretical Background

### 3.1 Power System Faults

Short-Circuit Faults: These occur when there is an unintended connection between two points of different potential, leading to excessive currents. Examples include phase-to-phase and phase-to-ground faults.

Open-Circuit Faults: These occur when a conducting path is broken, resulting in an interruption of the current flow. Examples include broken conductors or disconnected loads.

### 3.2 Per-Unit System

The per-unit system is a method of normalizing electrical quantities (such as voltage, current, impedance) by expressing them as fractions of a defined base value. It simplifies calculations, especially when dealing with different voltage levels. The per-unit values are calculated as follows:

$$\text{Per-Unit Voltage (V_pu)} = \frac{\text{Actual Voltage}}{\text{Base Voltage}}$$

$$\text{Per-Unit Current (I_pu)} = \frac{\text{Actual Current}}{\text{Base Current}}$$

$$\text{Per-Unit Impedance (Z_pu)} = \frac{\text{Actual Impedance}}{\text{Base Impedance}}$$

### 3.3 Fault Analysis Using Per-Unit System

During fault analysis, power system networks are modeled using their per-unit values, and the resulting equations are solved to determine the fault current and voltages at different points in the system.

## 4. Methodology

### Python-Based Tool Design

The Python tool is designed to perform the following tasks:

1. Input System Parameters: Gather input data such as base voltage, base power, line impedance, and load conditions.

2. Model Faults: Simulate short-circuit and open-circuit conditions.

3. Calculate Per-Unit Values: Convert system parameters into their per-unit equivalents.

4. Solve Fault Equations: Compute the per-unit currents and voltages using Ohm's law and Kirchhoff's laws.

5. Output Results: Display the calculated fault currents and voltages.

## 5. Code Implementation

Below is the Python code for modeling basic power system faults and computing fault conditions using per-unit analysis:

```
class PowerSystemFault:

    def __init__(self, base_voltage_kv, base_power_mva):

        # Base values

        self.base_voltage = base_voltage_kv  # in kV

        self.base_power = base_power_mva  # in MVA

        self.base_impedance = (self.base_voltage ** 2) / self.base_power  # Zbase = Vbase^2 / Sbase

    def per_unit_impedance(self, actual_impedance):

        # Convert actual impedance to per-unit

        return actual_impedance / self.base_impedance

    def short_circuit_fault(self, pre_fault_voltage_pu, fault_impedance_pu):

        # Fault current calculation (I_f = V_pre / Z_f)

        fault_current_pu = pre_fault_voltage_pu / fault_impedance_pu
```

```
        print(f"Short-circuit fault: Fault Current (pu) = {fault_current_pu}")
        return fault_current_pu
    def open_circuit_fault(self, pre_fault_voltage_pu):
        # For an open circuit, current becomes zero, voltage is unchanged (assumed balanced
condition)
        fault_current_pu = 0
        print(f"Open-circuit fault: Fault Current (pu) = {fault_current_pu}")
        return fault_current_pu
# Test harness to collect inputs from command line
def main():
    # User inputs
    base_voltage = float(input("Enter base voltage (kV): "))
    base_power = float(input("Enter base power (MVA): "))
     # Create the system
    system = PowerSystemFault(base_voltage, base_power)
     # Choose fault type
    fault_type = input("Enter fault type (short-circuit/open-circuit): ").strip().lower()
    pre_fault_voltage_pu = float(input("Enter pre-fault voltage (pu): "))
      if fault_type == "short-circuit":
        actual_impedance = float(input("Enter actual fault impedance (ohms): "))
        fault_impedance_pu = system.per_unit_impedance(actual_impedance)
        system.short_circuit_fault(pre_fault_voltage_pu, fault_impedance_pu)
      elif fault_type == "open-circuit":
        system.open_circuit_fault(pre_fault_voltage_pu)
    else:
```

```
    print("Invalid fault type entered.")

if __name__ == "__main__":

    main()
```

## 6. Results and Discussion

The Python tool allows users to simulate fault conditions and observe the behavior of current and voltage in the system under these conditions:

Short-Circuit Fault: Results in a high fault current due to the reduced impedance, causing significant stress on system components.

Open-Circuit Fault: Results in zero current flow, which can lead to a loss of power delivery to the load.

The use of per-unit analysis simplifies the calculations and helps in understanding the system's behavior under fault conditions, irrespective of different voltage levels.

## 7. Conclusion

The developed Python tool provides a simplified approach to analyzing basic power system faults using per-unit analysis. By converting system parameters to their per-unit equivalents, the tool can efficiently simulate fault conditions and determine the resulting fault currents and voltages. This tool can serve as a fundamental building block for more advanced fault detection and analysis in power systems.

## 8. Future Work

Extend the tool to analyze three-phase faults.

Integrate graphical representation of fault conditions.

Include real-world data to improve the accuracy of simulations.

## 9. References

Grainger, J.J., Stevenson, W.D. *Power System Analysis*. McGraw-Hill, 1994.

Kundur, P. *Power System Stability and Control*. McGraw-Hill, 1994.

Saadat, H. *Power System Analysis*. McGraw-Hill, 1999.