Power Factor Correction Program

Hani Al Jamal – Ali Yaseen – Chafic Labaki

American University of Beirut

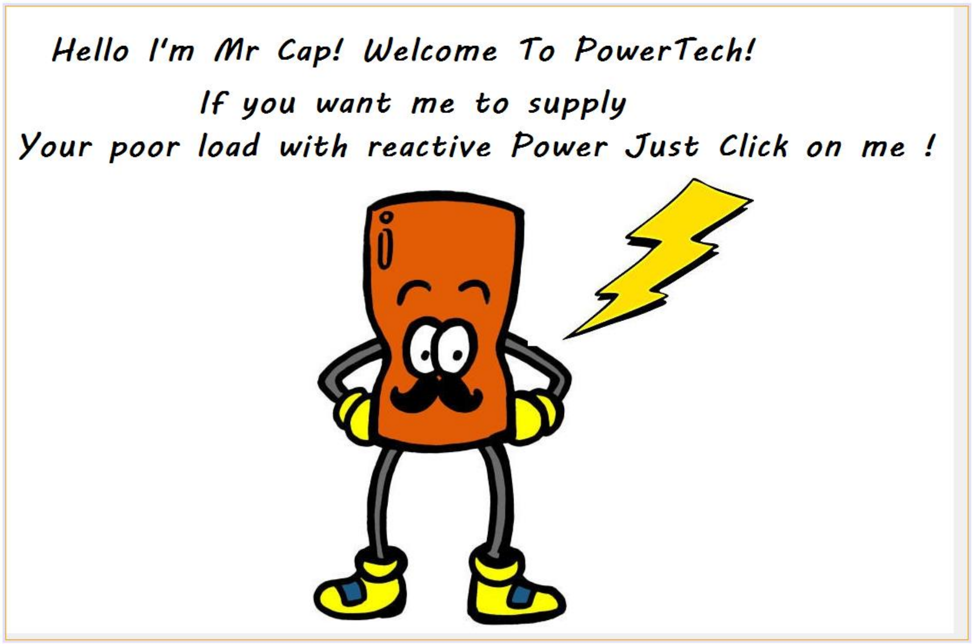
EECE 370

8 July 2019

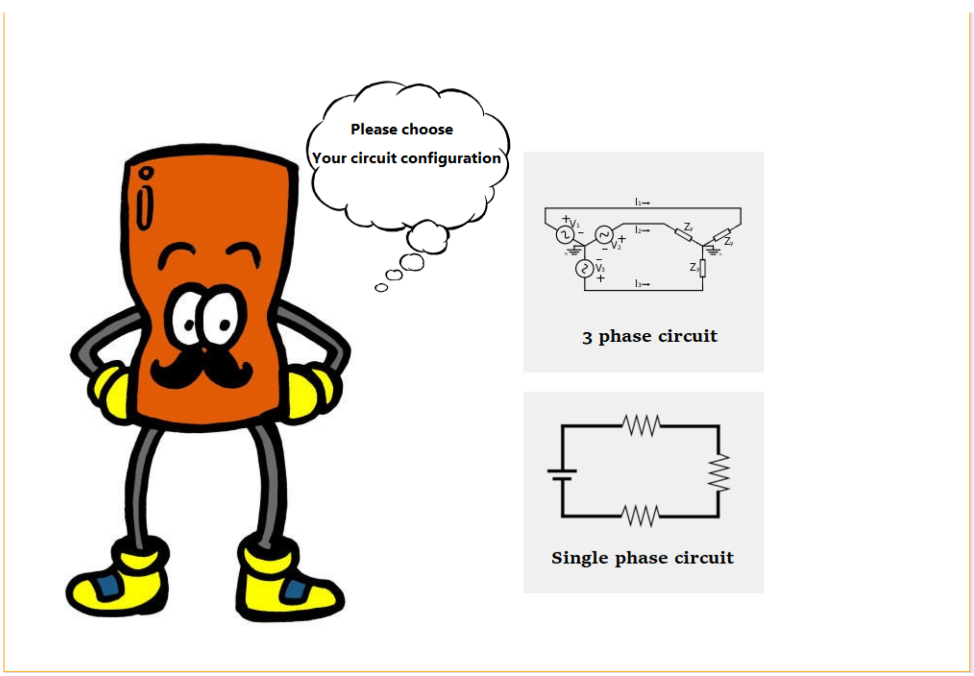
Overview:

The following program we have developed aims at calculating the needed capacitor size for power factor correction. It features a fun and easy to use graphical user interface that we developed using NetBeans and Java. The following shows how the program function:

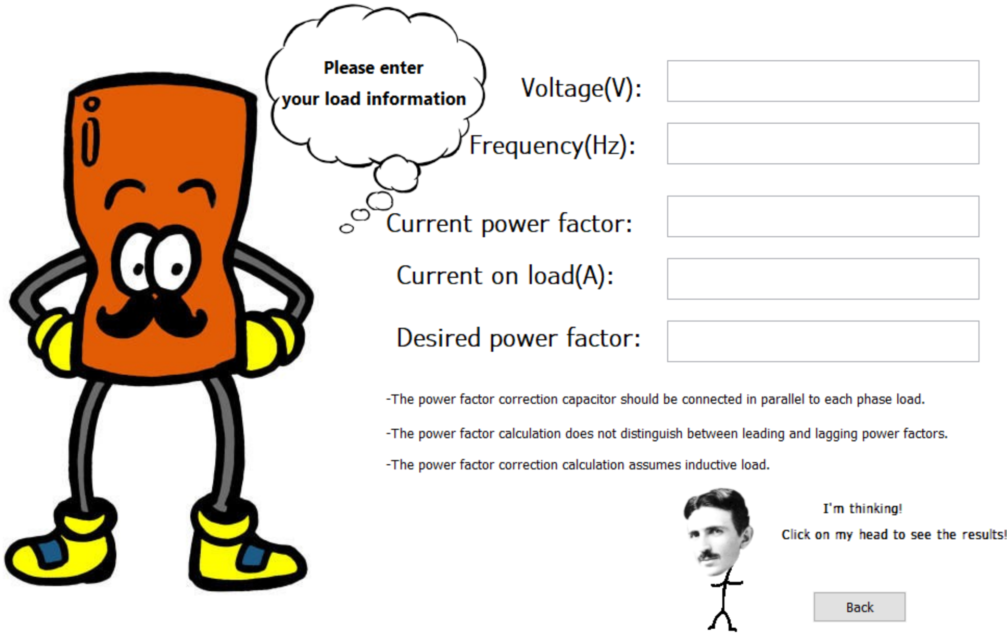
- Mr. Cap greets you on the home screen page:



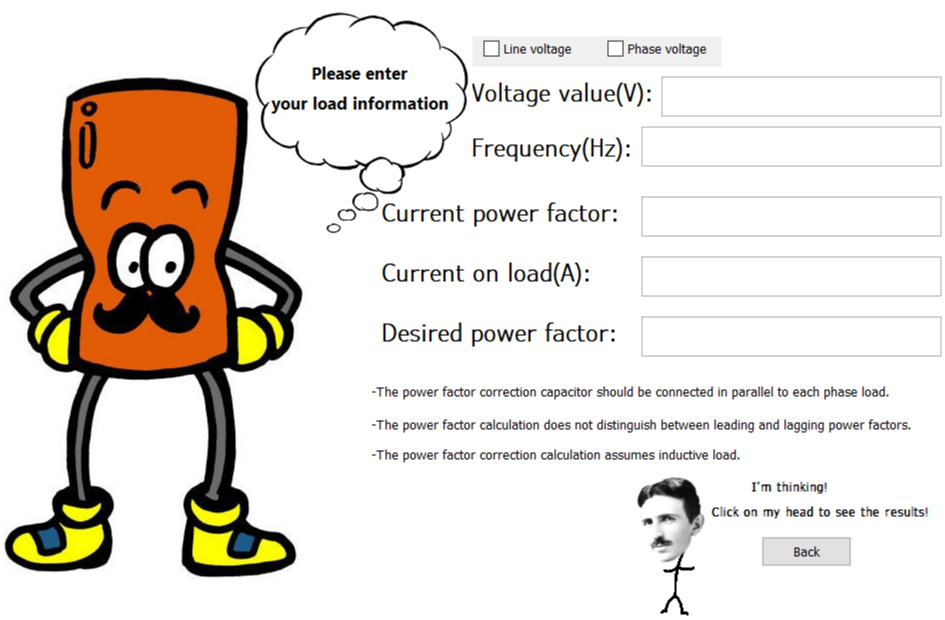
- Our program supports both single and 3 phase circuits:



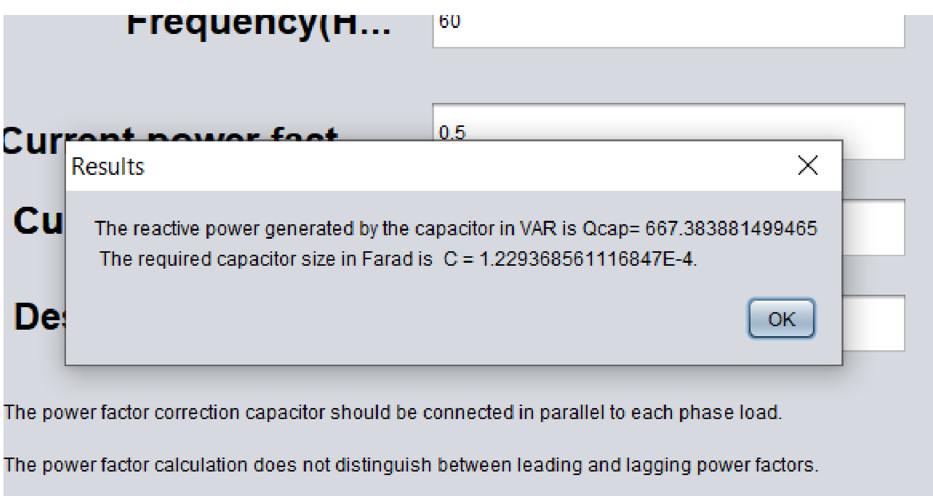
- Single phase circuit parameters to be entered by the user:



- 3 phase circuit parameters to be entered by the user with the option to choose between line or phase voltage:



- The output shows the reactive power required to be delivered by the capacitor and the capacitor size:



- The calculations the program performs are based on the following formulas and Java code:

1. Single Phase:

Power factor calculation:

$$PF = |\cos\varphi| = 1000 \times P_{(kW)} / (V_{(V)} \times I_{(A)})$$

Apparent power calculation:

$$|S_{(kVA)}| = V_{(V)} \times I_{(A)} / 1000$$

Reactive power calculation:

$$Q_{(kVAR)} = \sqrt{(|S_{(kVA)}|^2 - P_{(kW)}^2)}$$

Power factor correction capacitor's capacitance calculation:

$$S_{corrected\ (kVA)} = P_{(kW)} / PF_{corrected}$$

$$Q_{corrected\ (kVAR)} = \sqrt{(S_{corrected\ (kVA)}^2 - P_{(kW)}^2)}$$

$$Q_{c\ (kVAR)} = Q_{(kVAR)} - Q_{corrected\ (kVAR)}$$

$$C_{(F)} = 1000 \times Q_{c\ (kVAR)} / (2\pi f_{(Hz)} \times V_{(V)}^2)$$

Figure 1: Single Phase Formulas

```java
public void result( double v, double i, double f, double pfPrime, double pfTemp, double p) {

        double pf = pfTemp != 0.0 ? pfTemp : this.pfPrime(p, v, i);
        this.q = v * i * Math.sin( Math.acos( pf ) );
        double vipfPrime =  v * i * pf;
        this.sc = vipfPrime / pfPrime;
        this.qc = Math.sqrt( (sc*sc) - (vipfPrime*vipfPrime) );
        this.qcap = this.q-this.qc;
        this.c = this.qcap / ( 2 * Math.PI * f * v * v );}

    public double getQ(){ return this.q;}

    public double getSc() { return this.sc;}

    public double getQc() { return this.qc; }

    public double getQcap() { return this.qcap; }

    public double getC() { return this.c; }

    private double pfPrime(double p, double v, double i) {
        return p / (v*i);
    }
```

Figure 2: Single Phase Java Code

2. Three Phase (Line to Line voltage):

Power factor calculation:

$$PF = |\cos \varphi| = 1000 \times P_{(kW)} / (\sqrt{3} \times V_{L\text{-}L(V)} \times I_{(A)})$$

Apparent power calculation:

$$|S_{(kVA)}| = \sqrt{3} \times V_{L\text{-}L(V)} \times I_{(A)} / 1000$$

Reactive power calculation:

$$Q_{(kVAR)} = \sqrt{(|S_{(kVA)}|^2 - P_{(kW)}{}^2)}$$

Power factor correction capacitor's capacitance calculation:

$$Q_{c\ (kVAR)} = Q_{(kVAR)} - Q_{corrected\ (kVAR)}$$

$$C_{(F)} = 1000 \times Q_{c\ (kVAR)} / (2\pi f_{(Hz)} \times V_{L\text{-}L(V)}{}^2)$$

Figure 3: Three Phase (Line to Line Voltage) Formulas

3. Three Phase (line to neutral):

Power factor calculation:

$$PF = |\cos \varphi| = 1000 \times P_{(kW)} / (3 \times V_{L\text{-}N(V)} \times I_{(A)})$$

Apparent power calculation:

$$|S_{(kVA)}| = 3 \times V_{L\text{-}N(V)} \times I_{(A)} / 1000$$

Reactive power calculation:

$$Q_{(kVAR)} = \sqrt{(|S_{(kVA)}|^2 - P_{(kW)}{}^2)}$$

Power factor correction capacitor's capacitance calculation:

$$Q_{c\ (kVAR)} = Q_{(kVAR)} - Q_{corrected\ (kVAR)}$$

$$C_{(F)} = 1000 \times Q_{c\ (kVAR)} / (3 \times 2\pi f_{(Hz)} \times V_{L\text{-}N(V)}{}^2)$$

Figure 4: Three Phase (Line to Neutral) Formulas

```java
public void result2( boolean phase,double v, double i, double f, double pfPrime, double pfTemp, double p) {
    if(phase==true)
     {
        double pf = pfTemp != 0.0 ? pfTemp : this.pfPrime(p, v, i);

     this.q = v * i * Math.sin( Math.acos( pf ) );
     double vipfPrime =  v * i * pf;
     this.sc = vipfPrime / pfPrime;
     this.qc = Math.sqrt( (sc*sc) - (vipfPrime*vipfPrime) );
     this.qcap = this.q-this.qc;
     this.c = this.qcap / ( 2 * Math.PI * f * v * v );}

    else {
        double pf = pfTemp != 0.0 ? pfTemp : this.pfPrime(p, v, i);

             this.q = (v/Math.sqrt(3)) * i * Math.sin( Math.acos( pf ) );
             double vipfPrime =  (v/Math.sqrt(3)) * i * pf;
             this.sc = vipfPrime / pfPrime;
             this.qc = Math.sqrt( (sc*sc) - (vipfPrime*vipfPrime) );
             this.qcap = this.q-this.qc;
             this.c = this.qcap / ( 2 * Math.PI * f * (v/Math.sqrt(3)) * (v/Math.sqrt(3)) );}


    }
```

Figure 5:Three Phase (Line to Neutral and Line to Line) Java code.