Smart micro grids
Simulating the electrical components of a home

Smart homes are an important part of smart micro-grids. They provide information about electricity consumption that can be used for analysis of the grid and predictions of future power consumption. They enable active control of certain devices that can be used to regulate the grid or provide contingency services. They enable the implementation of energy saving management schemes, among many other very useful applications. In order to model a smart home, we must first model its electrical components.

Any residence makes use of a number of household appliances, lighting, and other electrically powered devices. Some of these devices require three-phases from the grid, e.g. washing machine, and some require only one phase from the grid, e.g. television. They have different consumption profiles and can be modeled in different ways. This document is an introduction to how these different devices might be modeled using Acumen.

The grid (Three-phase system)

The grid in Sweden is a three-phase system with 230 root-mean-square (*rms*) voltage, and 50Hz frequency. Each phase is offset from the others by $120°$ or $^{2\pi}/_{3}$ radians, as illustrated below.

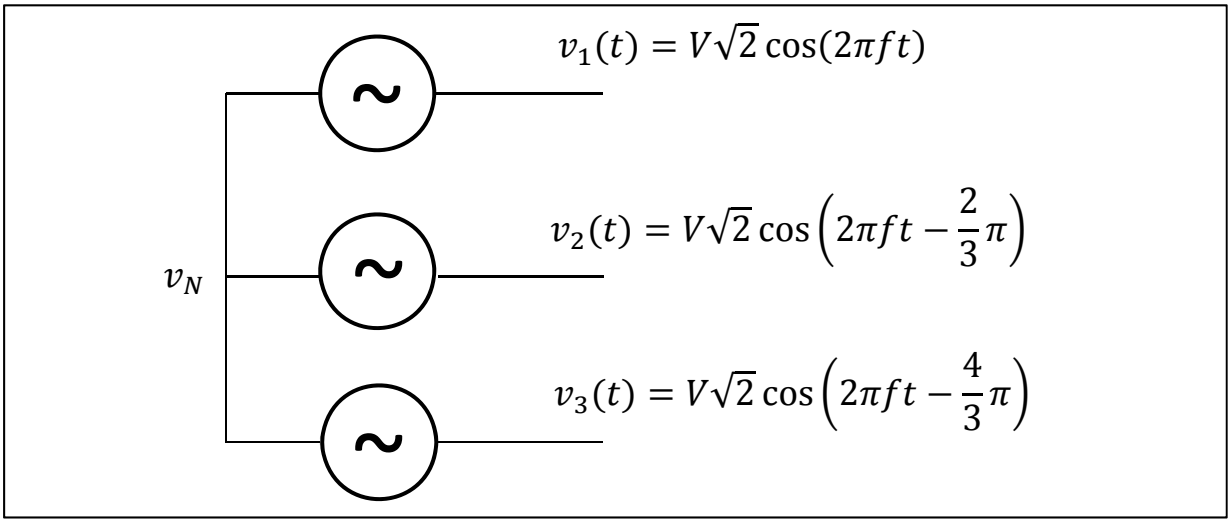

Fig1. Three-phase source (the grid).

We will represent *rms* values with capital letters and time varying values with small case letters. For example, V is the *rms* value of $v(t)$. Note that in a sinusoidal wave, the rms value of the wave is its peak value (amplitude) divided by square-root of two. That is, $V_{peak} = V\sqrt{2}$.

Complex numbers are commonly used to represent the voltages in a three-phase system, as shown below.

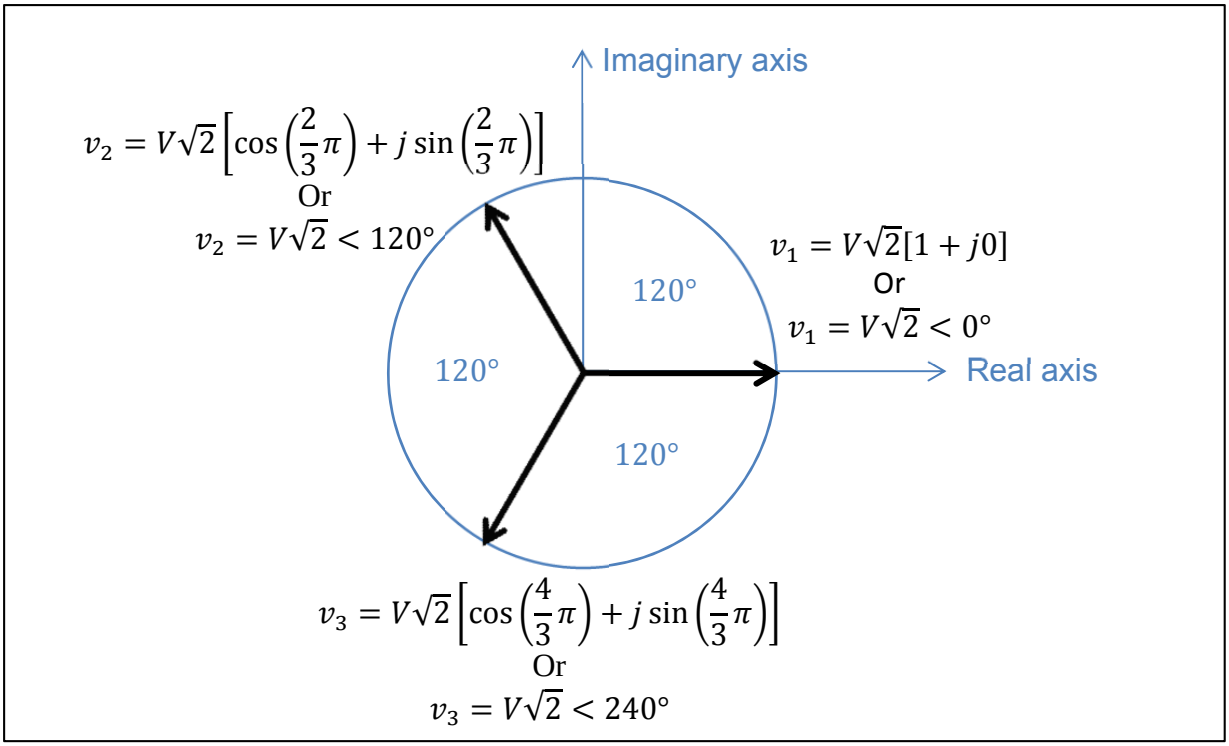

Fig 2. Complex number representation of three-phase systems.

Note that in an ideal system:

$$v_N = v_1 + v_2 + v_3$$
$$v_N = V\sqrt{2}\left[1 + \cos\left(\frac{2}{3}\pi\right) + \cos\left(\frac{4}{3}\pi\right) + j0 + j\sin\left(\frac{2}{3}\pi\right) + j\sin\left(\frac{4}{3}\pi\right)\right]$$
$$v_N = V\sqrt{2}[1 - 0.5 - 0.5 + j0.866 - j0.866] = 0$$

In practice, v_N is typically grounded to ensure that it is zero.

In Acumen we can create a class which represents the grid as follows.

Table 1

```
class ThreePhaseSourceIdeal(V, f)
  private t:=0; t':=1; v1:=0; v2:=0; v3:=0; end
  t' = 1;
  v1 = V*sqrt(2)*cos(2*pi*f*t);
  v2 = V*sqrt(2)*cos(2*pi*f*t - pi*2/3);
  v3 = V*sqrt(2)*cos(2*pi*f*t - pi*4/3);
end

class Main(simulator)
  private
    source := create ThreePhaseSourceIdeal(230, 50);
  end
  simulator.endTime = 1;
  simulator.timeStep = 0.0001;
end
```

You might want to double check that the sum of the phase voltages really is zero; or that the *rms* value of v_1 is 230 volts. In order to do so, we define:

$$v_N = v_1 + v_2 + v_3$$

$$V_1 = \sqrt{\left(\frac{1}{T}\int_0^T v_1(t)^2\,dt\right)}$$

Let's alter the previous code to incorporate this. We will add a class that computes the *rms* value of a signal, `CalculateRMS`. After running the code in Table 2, you can verify that `Vrms` is approximately 230 and that `vN` is virtually zero.

Table 2

```
class ThreePhaseSourceIdeal(V, f)
  private
    t:=0; t':=1; v1:=0; v2:=0; v3:=0; vN:=0;
    checkV1 := create CalculateRMS();
    V1:=0;
  end
  t' = 1;
  v1 = V*sqrt(2)*cos(2*pi*f*t);
  v2 = V*sqrt(2)*cos(2*pi*f*t - pi*2/3);
  v3 = V*sqrt(2)*cos(2*pi*f*t - pi*4/3);
  vN = v1 + v2 + v3;
  checkV1.time_series = v1;
  V1 = checkV1.rms;
end

class CalculateRMS()
  private
    t:=0; t':=0; dummy:=0; dummy':=0; time_series:=0; rms:=0;
  end
  t'=1;
  dummy' = time_series^2;
  rms = sqrt(dummy/t);
end

class Main(simulator)
  private
    source := create ThreePhaseSourceIdeal(230, 50);
  end
  simulator.endTime = 1;
  simulator.timeStep = 0.0001;
end
```

Single phase loads

We will consider two simple types of load: purely resistive (R), or inductive and resistive (RL) loads. Resistive loads only produce real power. These are typically associated with heating elements and incandescent lighting. Inductive loads introduce a reactive power component. Complex numbers are usually used to represent RL loads, as follows.

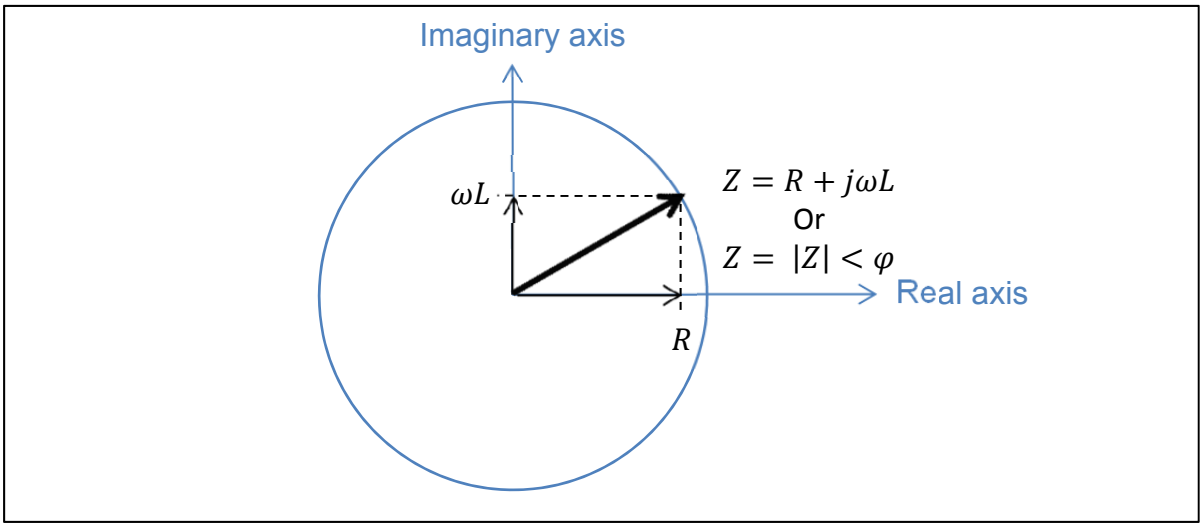

Fig 3. Complex number representation of resistive and inductive loads.

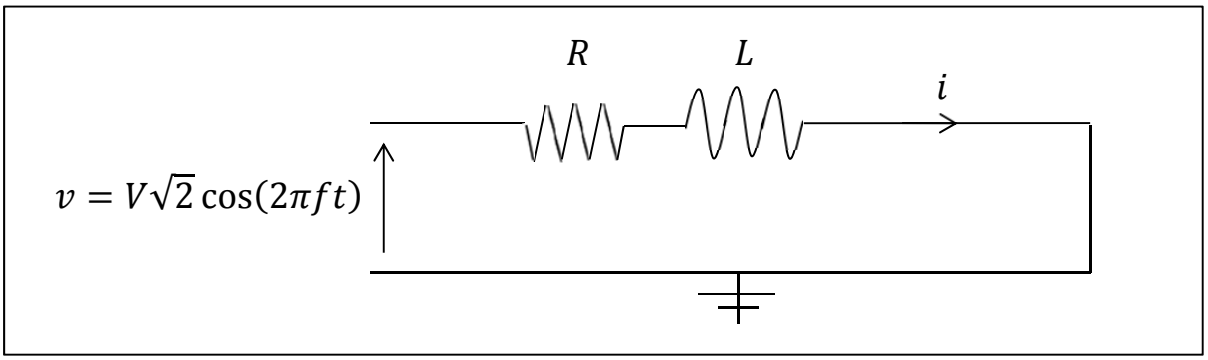

Fig 4. Single phase RL load, AC source.

In the case of an RL load, its current, i, and apparent power, S, are:

$$\frac{di}{dt} = \frac{v - Ri}{L}$$
$$S = P + jQ$$
$$S = \frac{V^2}{R} + j\frac{V^2}{\omega L}$$
$$\omega = 2\pi f$$

In the case of a purely resistive load:

$$i = \frac{v}{R}$$
$$S = P = \frac{V^2}{R}$$

The power factor (PF) is the ratio between active power and apparent power.

$$PF = \frac{P}{S} = cos\varphi$$
$$\varphi = \tan^{-1}\frac{\omega L}{R}$$

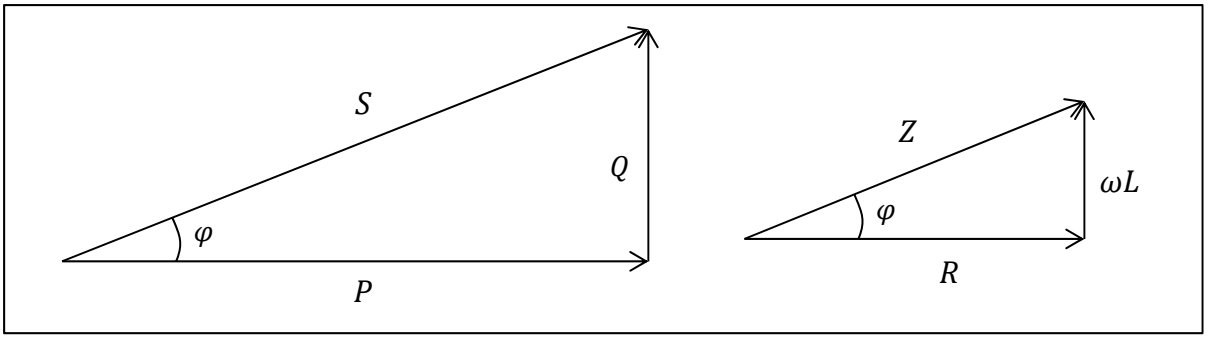

Fig 5. Vector representation of the power factor (PF).

In Acumen, a single phase load can be connected to phase 1 of the grid as shown in Table 3. Note that the class `SinglePhase_RL_Load` can be used to model a purely resistive load by setting `L=0`. The current, active power, reactive power and power factor are computed in this class.

Table 3

```
class ThreePhaseSourceIdeal(V, f)
  private t:=0; t':=1; v1:=0; v2:=0; v3:=0; end
  t' = 1;
  v1 = V*sqrt(2)*cos(2*pi*f*t);
  v2 = V*sqrt(2)*cos(2*pi*f*t - pi*120/180);
  v3 = V*sqrt(2)*cos(2*pi*f*t - pi*240/180);
end

class SinglePhase_RL_Load(R, L)
  private
    i:=0; i':=0; I:=0;
    v:=0; V:=0; f:=0;
    P:=0; Q:=0; PF:=0;
  end
  if L>0  i' = (v - R*i)/L;
    else i = v/R; end;
  I = V/R;
  P = (V^2/R)/100; //KW
  if L>0  Q = (V^2/2*pi*f*L)/100; end; //Kvar
  PF = P/sqrt(P^2 + Q^2);
end

class Main(simulator)
  private
    source := create ThreePhaseSourceIdeal(230, 50);
    load := create  SinglePhase_RL_Load(1, 0.01);
  end
  load.v = source.v1;
  load.V = source.V;
  load.f = source.f;
  simulator.endTime = 1;
  simulator.timeStep = 0.0001;
end
```

Three-phase loads

Any appliance that requires a three-phase source can be connected to the grid in two ways: in a Y (wye) configuration, or a Δ (delta) configuration. They can also be characterized as a balanced load, where each phase contains the same load; or an unbalanced load, where each phase may contain an arbitrary load. Household appliances are usually balanced; however, a malfunction may cause a load imbalance. The house as a whole can also be seen as an unbalanced load, since each phase will be connected to different combinations of appliances.

Loads in Y configuration

Figure 6 shows the schematics for an RL load in a Y configuration.

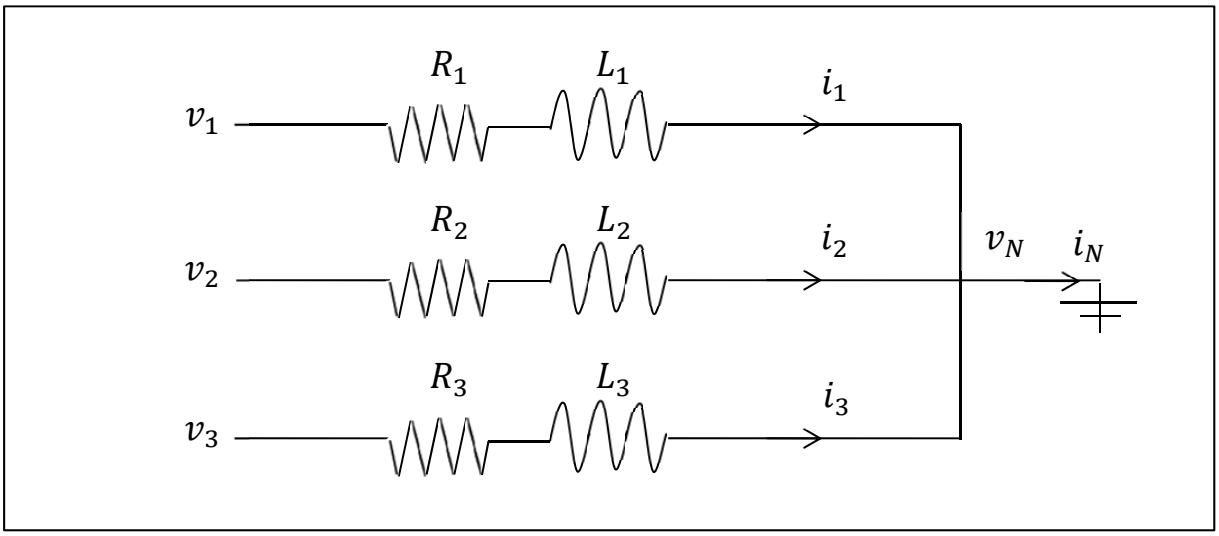

Fig 6. Three-phase RL load in Y configuration.

Note that the current in each phase, i_i, can be calculated as follows.

$$i_i = \frac{\sqrt{2}V < \theta_i}{Z_i < \varphi_i} = \frac{\sqrt{2}V}{Z_i} < (\theta_i - \varphi_i)$$
$$i_N = i_1 + i_2 + i_3$$

The total power consumed by this circuit is the sum of the powers consumed by each load.

$$S = S_1 + S_2 + S_3$$

The apparent power dissipated by each load, S_i, is a combination of its active, P_i, and reactive, Q_i, powers.

$$S_i = P_i + jQ_i$$
$$S_i = \frac{V^2}{R_i} + j\frac{V^2}{\omega L_i}$$

For balanced loads:

$$Z_1 = Z_2 = Z_3 = R + j\,\omega L$$
$$S = 3\frac{V^2}{R} + j3\frac{V^2}{\omega L}$$

For balanced and purely reactive loads:

$$S = P = 3\frac{V^2}{R}$$

The following Acumen code creates a three-phase source, a three-phase RL load in Y configuration and connects both elements in the `Main` class. Note that a purely resistive load can be modeled by `ThreePhase_RL_Load_WyeConnection` by setting the inductance values to zero.

Table 4

```
class ThreePhaseSourceIdeal(V, f)
  private t:=0; t':=1; v1:=0; v2:=0; v3:=0; end
  t' = 1;
  v1 = V*sqrt(2)*cos(2*pi*f*t);
  v2 = V*sqrt(2)*cos(2*pi*f*t - pi*2/3);
  v3 = V*sqrt(2)*cos(2*pi*f*t - pi*4/3);
end

class  ThreePhase_RL_Load_WyeConnection(R1, L1, R2, L2, R3, L3)
  private
    i1:=0; i2:=0; i3:=0; iN:=0;
    i1':=0; i2':=0; i3':=0;
    I1:=0; I2:=0; I3:=0;
    v1:=0; v2:=0; v3:=0;
    V:=0; f:=0;
    P:=0; Q:=0; Q1:=0; Q2:=0; Q3:=0; PF:=0;
  end
  if L1>0 i1' = (v1 - R1*i1)/L1;
    else i1 = v1/R1; end;
  if L2>0 i2' = (v2 - R2*i2)/L2;
    else i2 = v2/R2; end;
  if L3>0 i3' = (v3 - R3*i3)/L3;
    else i3 = v3/R3; end;
  I1 = V/sqrt(2*pi*f*L1^2 + R1^2);
  I2 = V/sqrt(2*pi*f*L2^2 + R2^2);
  I3 = V/sqrt(2*pi*f*L3^2 + R3^2);
  iN = i1+i2+i3;
  P = (V^2/R1 + V^2/R2 + V^2/R3)/100; //KW
  if L1>0 Q1 = V^2/2*pi*f*L1; end;
  if L2>0 Q2 = V^2/2*pi*f*L2; end;
  if L3>0 Q3 = V^2/2*pi*f*L3; end;
  Q = (Q1 + Q2 + Q3)/100; //Kvar
  PF = P/sqrt(P^2 + Q^2);
end

class Main(simulator)
  private
    source := create ThreePhaseSourceIdeal(230, 50);
    load   := create ThreePhase_RL_Load_WyeConnection(1, 0.01, 1,
```

```
0.01, 1, 0.01)
  end
  load.v1 = source.v1;
  load.v2 = source.v2;
  load.v3 = source.v3;
  load.f = source.f;
  load.V = source.V;
  simulator.endTime = 1;
  simulator.timeStep = 0.0001;
end
```

Loads in ∆ configuration

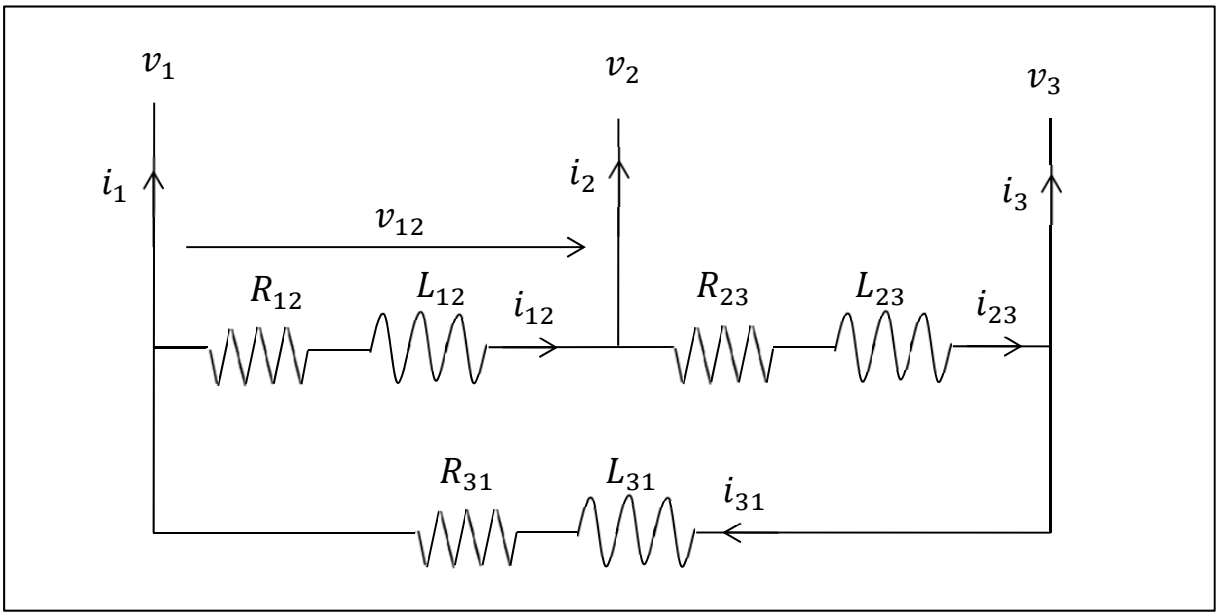

Fig 7. Three-phase RL load in ∆-connection

As previously, the total power is the sum of the powers dissipated by each load.

$$S = S_{12} + S_{23} + S_{31}$$
$$S_{ik} = P_{ik} + jQ_{ik}$$
$$S_{ik} = \frac{{V_{ik}}^2}{R_{ik}} + j\frac{{V_{ik}}^2}{\omega L_{ik}}$$
$$v_{ik} = v_i - v_k$$

Given that any two line voltages are $120°$ out of phase with each other, we can compute the line-to-line (or phase-to-phase) voltage between any two lines (or phases) as follows.

$$v_{ik} = \left(\sqrt{2}\,V\ <\theta\right) - \left(\sqrt{2}\,V\ <\theta + \frac{2\pi}{3}\right)$$
$$v_{ik} = \sqrt{2}\,V\ \left(\cos\theta + j\sin\theta - \cos\left(\theta + \frac{2\pi}{3}\right) - \sin\left(\theta + \frac{2\pi}{3}\right)\right)$$
$$v_{ik} = \sqrt{2}\,V\ \left(\cos\theta + j\sin\theta - \cos\theta\cos\frac{2\pi}{3} + \sin\theta\sin\frac{2\pi}{3} - j\sin\theta\cos\frac{2\pi}{3} - j\cos\theta\sin\frac{2\pi}{3}\right)$$

$$v_{ik} = \sqrt{2}\,V\left(\cos\theta + j\sin\theta + \frac{1}{2}\cos\theta + \frac{\sqrt{3}}{2}\sin\theta + j\frac{1}{2}\sin\theta - j\frac{\sqrt{3}}{2}\cos\theta\right)$$
$$v_{ik} = \sqrt{2}\,V\sqrt{3}\left(\frac{\sqrt{3}}{2}\cos\theta + \frac{1}{2}\sin\theta + j\frac{\sqrt{3}}{2}\sin\theta - j\frac{1}{2}\cos\theta\right)$$
$$v_{ik} = \sqrt{2}\,V\sqrt{3}\left(\cos\frac{\pi}{6}\cos\theta + \sin\frac{\pi}{6}\sin\theta + j\cos\frac{\pi}{6}\sin\theta - j\sin\frac{\pi}{6}\cos\theta\right)$$
$$v_{ik} = \sqrt{2}\,V\sqrt{3}\left(\cos\left(\theta - \frac{\pi}{6}\right) + j\sin\left(\theta - \frac{\pi}{6}\right)\right) = \sqrt{2}\,V\sqrt{3}\ <\theta - 30°$$

Therefore

$$S_{ik} = \frac{\left(V\sqrt{3}\right)^2}{R_{ik}} + j\frac{\left(V\sqrt{3}\right)^2}{\omega L_{ik}}$$

For balanced loads

$$S = \frac{9V^2}{R} + j\frac{9V^2}{\omega L}$$

For balanced and purely resistive loads

$$S = P = \frac{9V^2}{R}$$

Note that in a balanced system

$$I_i = \sqrt{3}\,I_{ik}$$

The Acumen code in Table 5 creates a three-phase source representing the grid, a RL load in Δ configuration, and connects them in the `Main` class. Although the *rms* line currents, I_i, are easy to compute for a balanced system, the equations for an unbalanced systems are little more complicated. However, we know that:

$$i_1 = i_{31} - i_{12}$$
$$i_2 = i_{12} - i_{23}$$
$$i_3 = i_{23} - i_{31}$$

The *rms* value of each line current can be approximated with the class `CalculateRMS` presented in Table 2. Note that in this example, $I_{ik} = 121$ volts, and I_i is approximately 210 volts, which is a little over $\sqrt{3}\,I_{ik} = 209.5$.

Table 5

```
class ThreePhaseSourceIdeal(V, f)
  private t:=0; t':=1; v1:=0; v2:=0; v3:=0; end
  t' = 1;
  v1 = V*sqrt(2)*cos(2*pi*f*t);
  v2 = V*sqrt(2)*cos(2*pi*f*t - pi*2/3);
  v3 = V*sqrt(2)*cos(2*pi*f*t - pi*4/3);
end
```

```
class  ThreePhase_RL_Load_DeltaConnection(R12, L12, R23, L23, R31,
L31)
  private
    i12:=0; i23:=0; i31:=0;
    I12:=0; I23:=0; I31:=0;
    i12':=0; i23':=0; i31':=0;
    i1:=0; i2:=0; i3:=0;
    I1:=0; I2:=0; I3:=0;
    v1:=0; v2:=0; v3:=0;
    V:=0; f:=0;
    P:=0; Q:=0; Q1:=0; Q2:=0; Q3:=0; PF:=0;
    rmsI1 := create CalculateRMS();
    rmsI2 := create CalculateRMS();
    rmsI3 := create CalculateRMS();
  end
  if L12>0 i12' = (v1 - v2 - R12*i12)/L12;
    else i12 = (v1 - v2)/R12; end;
  if L23>0 i23' = (v2 - v3 - R23*i23)/L23;
    else i23 = (v2 - v3)/R23; end;
  if L31>0 i31' = (v3 - v1 - R31*i31)/L31;
    else i31 = (v3 - v1)/R31; end;
  I12 =  (sqrt(3)*V)/sqrt(R12^2+(2*pi*f*L12)^2);
  I23 =  (sqrt(3)*V)/sqrt(R23^2+(2*pi*f*L23)^2);
  I31 =  (sqrt(3)*V)/sqrt(R31^2+(2*pi*f*L31)^2);
  i1 = i31 - i12;
  i2 = i12 - i23;
  i3 = i23 - i31;
  rmsI1.time_series = i1;
  rmsI2.time_series = i2;
  rmsI3.time_series = i3;
  I1 = rmsI3.rms;
  I2 = rmsI2.rms;
  I3 = rmsI3.rms;
  P = ((sqrt(3)*V)^2/R12 + (sqrt(3)*V)^2/R23 +
(sqrt(3)*V)^2/R31)/100; //KW
  if L12>0 Q1 = (sqrt(3)*V)^2/2*pi*f*L12; end;
  if L23>0 Q2 = (sqrt(3)*V)^2/2*pi*f*L23; end;
  if L31>0 Q3 = (sqrt(3)*V)^2/2*pi*f*L31; end;
  Q = (Q1 + Q2 + Q3)/100; //Kvar
  PF = P/sqrt(P^2 + Q^2);
end

class CalculateRMS()
  private
    t:=0; t':=0; dummy:=0; dummy':=0; time_series:=0; rms:=0;
  end
  t'=1;
  dummy' = time_series^2;
  rms = sqrt(dummy/t);
end

class Main(simulator)
  private
    source := create ThreePhaseSourceIdeal(230, 50);
    load   := create ThreePhase_RL_Load_DeltaConnection(1, 0.01, 1,
```

```
0.01, 1, 0.01)
  end
  load.v1 = source.v1;
  load.v2 = source.v2;
  load.v3 = source.v3;
  load.f = source.f;
  load.V = source.V;
  simulator.endTime = 1;
  simulator.timeStep = 0.0001;
end
```

Circuits and breakers

The wiring system of home is divided into a number of circuits. Each circuit may provide energy to a certain number of appliances. For safety reasons, each circuit is connected to a breaker, which can open the circuit in case of an elevated current due to a faulty appliance or short circuiting of the system.

The code in Table 6 creates a three-phase source, and a circuit with two three-phase RL–Y loads. In order to illustrate the operation of a breaker, we simulate a fault in phase 1 of `load1`. Note that when the fault occurs at $t \geq 0.1$ s, there is a surge in current that overpasses the limit of `Ibreak1` and triggers the breaker.

Table 6

```
class ThreePhaseSourceIdeal(V, f)
  private t:=0; t':=1; v1:=0; v2:=0; v3:=0; end
  t' = 1;
  v1 = V*sqrt(2)*cos(2*pi*f*t);
  v2 = V*sqrt(2)*cos(2*pi*f*t - pi*120/180);
  v3 = V*sqrt(2)*cos(2*pi*f*t - pi*240/180);
end

class  ThreePhase_RL_Load_WyeConnection(R1, L1, R2, L2, R3, L3)
  private
    i1:=0; i2:=0; i3:=0; iN:=0;
    i1':=0; i2':=0; i3':=0;
    I1:=0; I2:=0; I3:=0;
    v1:=0; v2:=0; v3:=0;
    V:=0; f:=0;
    P:=0; Q:=0; Q1:=0; Q2:=0; Q3:=0; PF:=0;
  end
  if L1>0 i1' = (v1 - R1*i1)/L1;
    else i1 = v1/R1; end;
  if L2>0 i2' = (v2 - R2*i2)/L2;
    else i2 = v2/R2; end;
  if L3>0 i3' = (v3 - R3*i3)/L3;
    else i3 = v3/R3; end;
  I1 = V/sqrt(2*pi*f*L1^2 + R1^2);
  I2 = V/sqrt(2*pi*f*L2^2 + R2^2);
  I3 = V/sqrt(2*pi*f*L3^2 + R3^2);
  iN = i1+i2+i3;
  P = (V^2/R1 + V^2/R2 + V^2/R3)/100; //KW
  if L1>0 Q1 = V^2/2*pi*f*L1; end;
```

```
  if L2>0 Q2 = V^2/2*pi*f*L2; end;
  if L3>0 Q3 = V^2/2*pi*f*L3; end;
  Q = (Q1 + Q2 + Q3)/100; //Kvar
  PF = P/sqrt(P^2 + Q^2);
end

class ThreePhaseCircuitAndBreaker(Ibreak1, Ibreak2, Ibreak3)
  private
    load1 := create ThreePhase_RL_Load_WyeConnection(100, 0.01, 100,
0.01, 100, 0.01);
    load2 := create ThreePhase_RL_Load_WyeConnection(50, 0.01, 50,
0.01, 50, 0.01);
    v1:=0; v2:=0; v3:=0; V:=0; f:=0;
    I1:=0; I2:=0; I3:=0; IN:=0;
    Pcircuit:=0; Qcircuit:=0;
    mode:="normal"; t:=0; t':=1;
  end

  t' = 1;
  if t > 0.1 load1.R1 = 5; end;

  I1 = load1.I1 + load2.I1;
  I2 = load1.I2 + load2.I2;
  I3 = load1.I3 + load2.I3;
  Pcircuit = load1.P + load2.P;
  Qcircuit = load1.Q + load2.Q;

  switch mode
    case "normal"
    load1.v1 = v1;
    load1.v2 = v2;
    load1.v3 = v3;
    load1.V = V;
    load1.f = f;
    load2.v1 = v1;
    load2.v2 = v2;
    load2.v3 = v3;
    load2.V = V;
    load2.f = f;
    if I1 >= Ibreak1 || I2 >= Ibreak2 || I3 >= Ibreak3  mode =
"break"; end;

    case "break"
    load1.v1 = 0;
    load1.v2 = 0;
    load1.v3 = 0;
    load1.V = 0;
    load1.f = 0;
    load2.v1 = 0;
    load2.v2 = 0;
    load2.v3 = 0;
    load2.V = 0;
    load2.f = 0;
  end
end
```

```
class Main(simulator)
  private
    source := create ThreePhaseSourceIdeal(230, 50);
    circuit1 := create ThreePhaseCircuitAndBreaker(10, 10, 10);
    Ptotal:=0; Qtotal:=0; PF:=0;
  end
  circuit1.v1 = source.v1;
  circuit1.v2 = source.v2;
  circuit1.v3 = source.v3;
  circuit1.V = source.V;
  circuit1.f = source.f;
  Ptotal = circuit1.Pcircuit;
  Qtotal = circuit1.Qcircuit;
  PF = Ptotal/sqrt(Ptotal^2+Qtotal^2);
  simulator.endTime = 0.5;
  simulator.timeStep = 0.0001;
end
```