Distribution Power Flow in GridLabD

IRW Group Meeting
June 18, 2010
Chengrui Cai
Nearly all objects within the powerflow module are derived from two primary objects which are node and link.

All objects within the powerflow module inherit two basic properties. These two properties are the phases of the object and the nominal voltage.

The phases property has a variety of valid inputs. These are:
- A - Phase A of a three phase connection
- B - Phase B of a three phase connection
- C - Phase C of a three phase connection
- D - Delta connected phases
  - this implies ABC, but explicitly specifying them is recommended
- N - Neutral phase
- G - Ground phase
- S - Split phase - this represents residential level wires (2 "hot" and 1 neutral wire)
Node:

Voltage A, B, C, AB, BC, CA
Current A, B, C
Power A, B, C
Shunt A, B, C
Bustype (PQ, SWING, PV)
Maximum voltage error
Busflag: HASSOURCE to indicate that this node can have distributed generation, e.g. PV
Reference bus: the node to sample frequency

Although all variables can be set, only bolded items (line to neutral voltage, the nominal voltage, node type and HASSOURCE) are recommended to set. The last one is used to indicate the node supporting PV
**Link:**

From
To
Power in and power out (sum of 3 phases)
Power losses
Power in & out A, Current in & out A, Power loss A
Power in & out B, Current in & out B, Power loss B
Power in & out C, Current in & out C, Power loss C
Status (Open or Closed)
Flow direction

Two nodes or one node and one load
Relationships between objects in GridLabD power flow module

- Overhead line
  - Overhead line Configuration
  - Overhead Line Spacing
    - Length
    - Overhead Line Conductor
- Underground line
  - Underground line Configuration
    - Length
    - Underground Line Spacing
      - Underground Line Conductor
- Link
- Transformer
  - Configuration
- Voltage Regulator
  - Configuration
- Relay
- Fuse
- Switch
Relationships between objects in GridLabD power flow module
**Line:**
The line object has two implementations: overhead and underground line. Line-based objects inherit properties from the link object. Two new properties are also added:
- Double **length**
- Object **configuration**

**Configuration:**
- Object conductor_A
- Object conductor_B
- Object conductor_C
- Object conductor_N

**Object spacing**
- Complex $z_{11} \sim z_{33}$ (3 x 3 impedance matrix)

**Spacing:**
- Double distance_AB, distance_BC, distance_CA, distance_AN, distance_BN, distance_CN
**Overhead Line:**

```plaintext
object overhead_line{
    phases "ABCN";
    name 701-802;
    from node_701;
    to load_802;
    length 125960;
    configuration line_config_A;
}
```

**Overhead Line Conductors:**

```plaintext
object overhead_line_conductor {
    name overhead_line_conductor_100;
    geometric_mean_radius .00446;
    resistance 1.12;
}
```
Transformer:

object transformer {
    name xfrmr_709_775;
    phases "ABCN";
    from node_709;
    to node_775;
    configuration xfrmr_config_400;
}

Transformer Configuration:

object transformer_configuration {
    name xfrm_config_400;
    connect_type DELTA DELTA;
    power_rating 500;
    primary_voltage 4800;
    secondary_voltage 480;
    resistance 0.09;
    reactance 1.81;
}
Load:

**Player** objects can be used to vary the load with time. Load objects provide a means to implement **constant current**, **constant power**, and **constant impedance losses** or **generation** into the system. The convention is a load is a positive quantity, so generation would need to be represented as a negative number.

```plaintext
object load {
    phases "ABCD";
    name 841;
    constant_current_C -0.586139+9.765222j;
    constant_impedance_B 221.915014+104.430595j;
    constant_power_A 42000.000000+21000.000000j;
    nominal_voltage 4800;
}
```
**Meter:**

Meters provide a measurement point for power and energy on the system at a specific point. Coupled with a recorder or collector, the meter object provides a method to determine how much power and energy have been used by downstream connections, as well as how much current is flowing through the meter object at the present time.

```plaintext
object meter {
    name Mtr1;
    phases ABC;
    nominal_voltage 4800.0;
}
```

**Measurement:**
Energy, power, demand, real and reactive power
3 phase voltage and current

**Note:**
measured_demand is the watts measurement of the peak power demand of downstream objects.
Case study: IEEE 4 nodes test feeder

Data:
Geometric mean radius of phase lines and the neutral line;
Resistance of lines in ohm / mile
Lines spacing
Transformer rating(6000kWA), connecting type(Wye-Wye)
Primary/secondary nominal voltage (12470/4160)
Impedance in per unit value
Initial value of node voltages
Load type (constant power) and the complex power
Case study of IEEE 4 nodes test feeder

Pseudo code:

1\textsuperscript{st} step : Load tape and powerflow module
2\textsuperscript{nd} step :
  
  Define two kinds of line conductor \{gmr and resistance\}
  Define spacing
  Define distribution line configuration \{conductors and spacing\}
3\textsuperscript{rd} step : Define transformer configuration
4\textsuperscript{th} step:
  
  Define lines, nodes and load to give the topology of the distribution system to gridlabD
5\textsuperscript{th} step:
  
  Define meter{
    
    give data to meter properties.
    define nesting load object{
      
      phases, nominal voltage, load type and complex power and so on
    }\n  
  define players to change load according to time
  define recorders to record specific data of the load (power, current and so on)
}
results:

**Input:** changing phase A load and constant phase B and C load

A phase:
1987-04-04 00:00:00, +1275000.000+790174.031j
+1h, +1375000.000+780174.031j
+1h, +1275000.000+790174.031j
+1h, +1375000.000+780174.031j
+1h, +1275000.000+790174.031j

B and C phase: +1800000.000+871779.789j and +2375000.000+780624.750j

**Output:** phase A voltage of load and phase A current of load

<table>
<thead>
<tr>
<th>timestamp, voltage_A</th>
<th>timestamp, measured_current_A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1987-04-04 00:00:00 PST, +2169.34-156.393j</td>
<td>1987-04-04 00:00:00 PST, +558.574-404.515j</td>
</tr>
<tr>
<td>1987-04-04 01:00:00 PST, +2150.94-189.024j</td>
<td>1987-04-04 01:00:00 PST, +602.726-415.68j</td>
</tr>
<tr>
<td>1987-04-04 02:00:00 PST, +2169.34-156.393j</td>
<td>1987-04-04 02:00:00 PST, +558.574-404.515j</td>
</tr>
<tr>
<td>1987-04-04 03:00:00 PST, +2150.94-189.024j</td>
<td>1987-04-04 03:00:00 PST, +602.726-415.68j</td>
</tr>
<tr>
<td>1987-04-04 04:00:00 PST, +2169.34-156.393j</td>
<td>1987-04-04 04:00:00 PST, +558.574-404.515j</td>
</tr>
</tbody>
</table>
Working plan for the first stage seaming work:

Scenario:
Only the day ahead price signal will be passed to retail customers
Short term simulation without changing contract types of retail customers
LSE can have the access to meter data from TDU
Price signal is forwarded by TDU to meters installed on retail customers
Barriers to overcome:

1. Modeling of TDU: how to properly define the input and output of a TDU in GridlabD

2. Construct an environment for TDU: how to add TDU to the existing powerflow module

3. Establish the relationship between TDU and other objects in powerflow module: How can they communicate with each other

4. How to expand the capability of the external input portals of GridlabD: To send and receive data via Ethernet.

5. How to control the execution of GridlabD: to synchronize GridlabD and AMES