GridLAB-D Tutorial – Session 2
Powerflow Module

David P. Chassin
Summer 2016
dchassin@slac.stanford.edu

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Powerflow Overview

Powerflow: fundamental power system analysis tool.

- Problem statement:
  "For any system, given the voltages of the generators, the characteristics of power lines, and the voltage response of loads, determine the voltage magnitudes and angles at all nodes on the system."

Solution allows for the calculation of derived quantities

- All line currents
- All power flows throughout the system

Powerflow Overview

SLAC

Transmission Systems

- Originally limited to transmission systems
- Many assumptions made in early flow solvers, e.g.,
  - *Positive sequence phase model*
  - *Polynomial load response to voltage*

Distribution Systems

- Stability & voltage sag wasn't a problem → powerflow not done
- Initially based on simple circuit equivalents
- Sequential components used but found inappropriate.
- Recent work has focused on more complete modeling
  - *Full unbalanced 3-phase models are now the standard*
  - *Composite load model is becoming more prevalent*
  - *Distributed generation and voltage control a growing problem*

Distribution System Modeling

SLAC

Traditionally analysis performed for transmission

- Distribution analysis growing in last 20 years
- Emerging technologies designed to operate in distribution

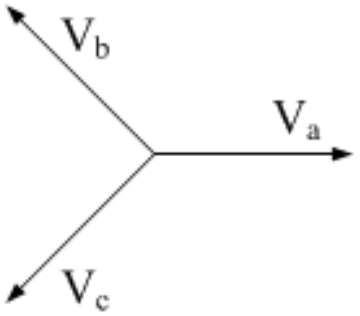
Distribution modeling similar to transmission modeling

- More detail in equipment models
- Single and double phase lines exist
- Distribution systems are often radial
- The R/X values in distribution are much higher
- Loads are unbalanced

What Does “Unbalanced” Mean?

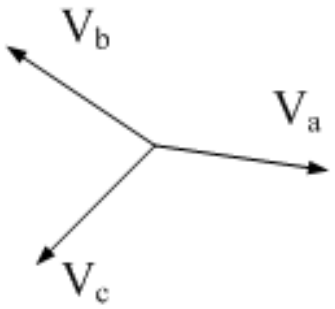
SLAC

The voltage at the source is balanced:



V_a=1V @0°=1+0j
V_b=1V @120°=-.5+.866j
V_c=1V @-120°=-.5+-.866j

The voltages downstream are unbalanced:



V_a=1V @-10°=.984+-.174j
V_b=.97V @110°=.332+.912j
V_c=1.03V @-133°=-.702+-.753j

Types of Distribution System Equipment

SLAC

Very common in distribution systems:

- Overhead lines
- Underground cables
- Transformers
- Voltage regulators
- Shunt capacitors

Others that may be found:

- Switches
- Reclosers
- Fuses
- Lightning protection

Transmission System

Distribution System

System Representations

Positive Sequence Model

- Assumed 3-phase balanced
- Single-phase equivalent models are used
- Performance advantage

Limits accuracy of model

- No phase-specific behavior

Three-phase Model

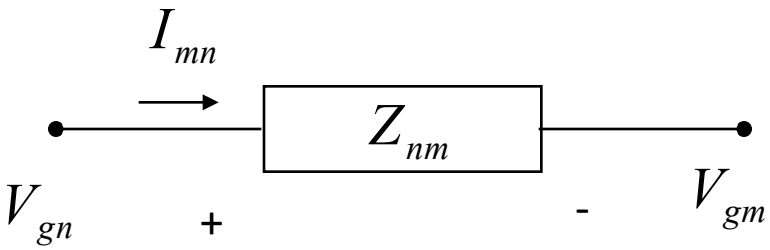
- Each phase explicit
- Model includes neutral
- Electromagnetic coupling

Allows accurate modeling

- Partial phase line topology
- Split-phase segments
- Single-phase loads
- Two-phase loads
- Phase-specific measurement, control design and actuation

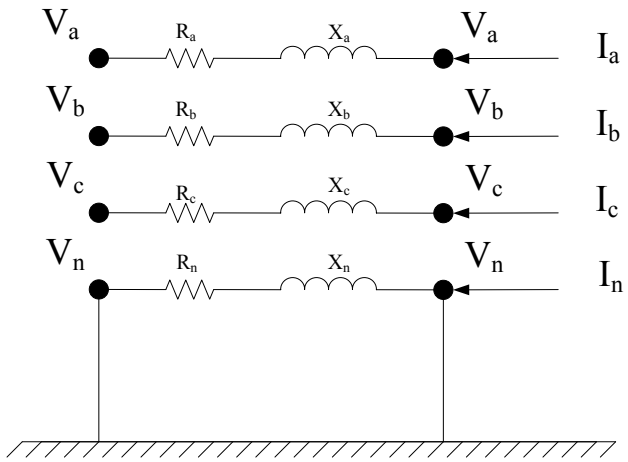
Modeling comparisons

Positive Sequence



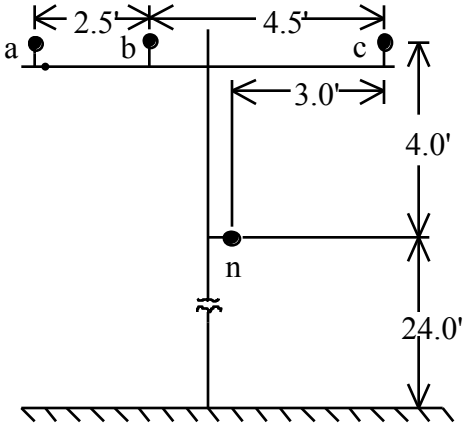
$$V_{gn} = V_{gm} + Z_{nm} I_{nm}$$

Unbalanced



$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix}_{to} = \begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix}_{from} + \begin{bmatrix} Z_{aa} & Z_{ab} & Z_{ac} \\ Z_{ba} & Z_{bb} & Z_{bc} \\ Z_{ca} & Z_{cb} & Z_{cc} \end{bmatrix} * \begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix}$$

Three-phase Unbalanced



Phase Conductor: 336,400 26/7

GMR = 0.0244 ft., Resistance = 0.306 Ω/mile, Diameter = 0.721 inch

Neutral Conductor: 4/0 6/1 ACSR

GMR = 0.00814 ft., Resistance = 0.592 Ω/mile, Diameter = 0.563 inch

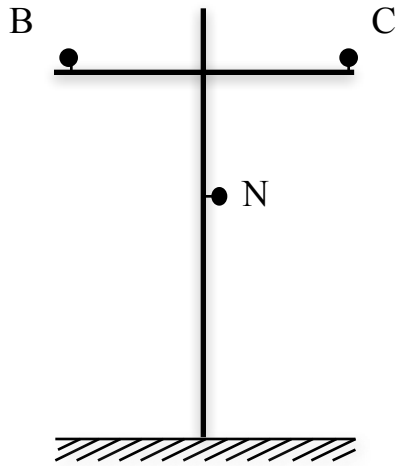
$$\begin{matrix} Z_{aa} & Z_{ab} & Z_{ac} \\ Z_{ba} & Z_{bb} & Z_{bc} \\ Z_{ca} & Z_{cb} & Z_{cc} \end{matrix}$$

Physical System → Parameterized Values → Values for Numeric Simulation

Overhead Line Code

Object Code

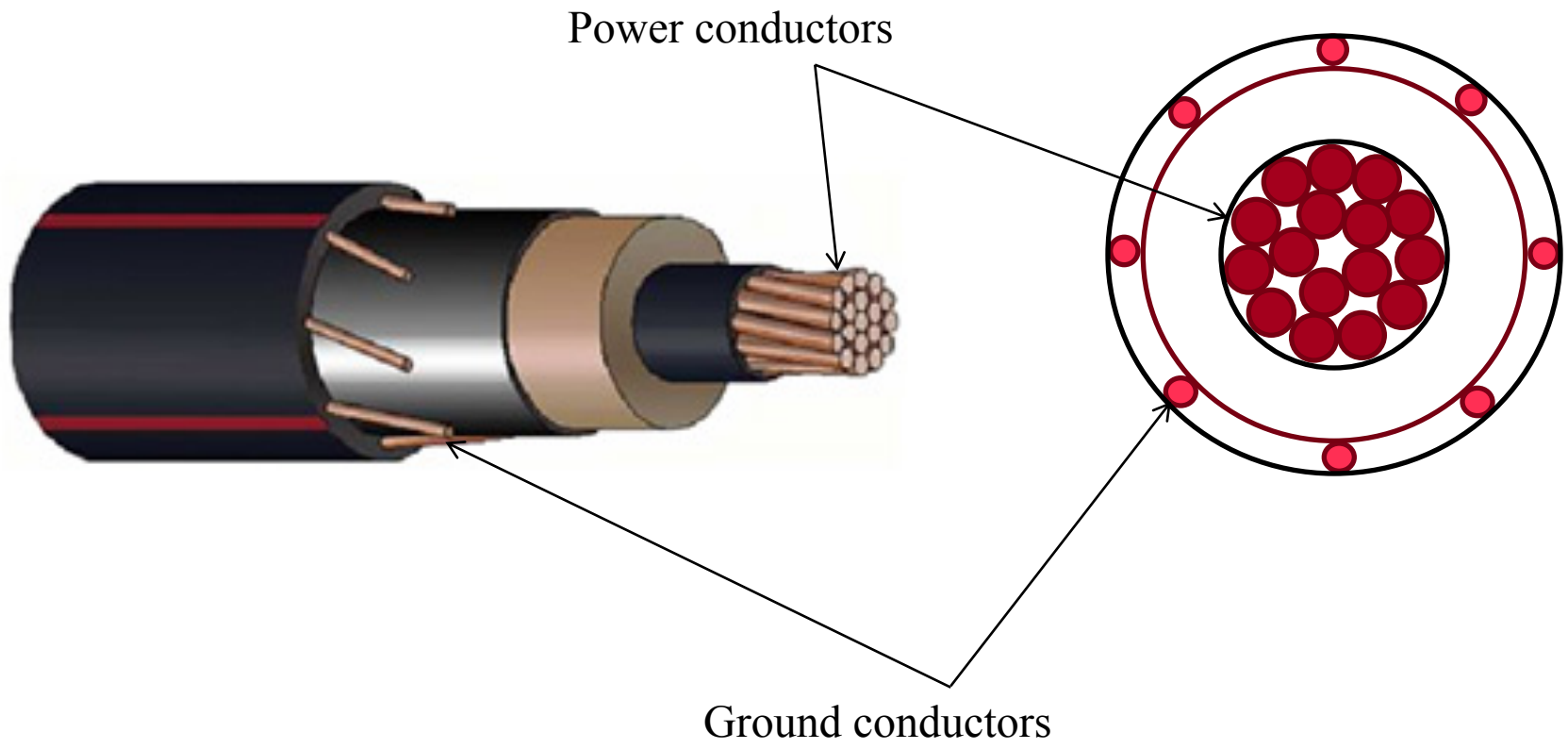
```
object overhead_line {
     phases "BCN";
     name ohl_632-645;
     from node_632;
     to load_645;
     length 500 ft;
     configuration line_config_603;
}
```



Configuration Code

```
object line_configuration {
            name line_config_603;
            conductor_B ohl_conductor_6030;
            conductor_C ohl_conductor_6030;
            conductor_N ohl_conductor_6030;
            spacing line_spacing_505603;
}
// Phase Conductor for 603, 604, 605: 1/0 ACSR
object overhead_line_conductor {
            name ohl_conductor_6030;
            geometric_mean_radius 0.004460;
            resistance 1.120000;
}
object line_spacing {
            name line_spacing_505603;
            distance_BC 7.0 ft;
            distance_CN 5.657 ft;
            distance_BN 5.0 ft;
}
```

Underground Cables



Underground Cable Code

Object Code

```
object underground_line {
      phases "AN";
      name ugl684_652;
      from node_684;
      to load_652;
      length 800;
      configuration line_config_607;
}
```

Configuration Code

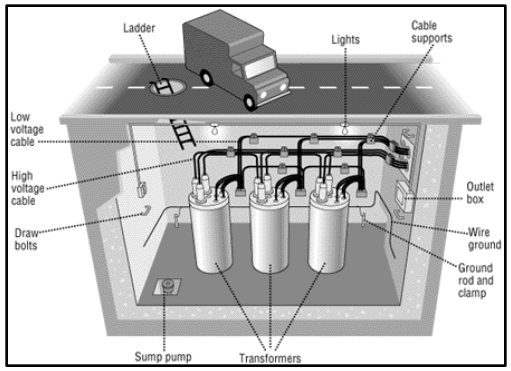
```
object line_configuration {
               name line_config_607;
               conductor_A ugl_conductor_6070;
               conductor_N ugl_conductor_6070;
               spacing line_spacing_520;
}
// Phase Conductor for 607: 1/0 AA,TS N: 1/0 Cu
object underground_line_conductor {
               name ugl_conductor_6070;
               outer_diameter 1.060000;
               conductor_gmr 0.011100;
               conductor_diameter 0.368000;
               conductor_resistance 0.970000;
               neutral_gmr 0.011100;
               neutral_resistance 0.970000;
               neutral_diameter 0.0640837;
               neutral_strands 6.000000;
               shield_gmr 0.000000;
               shield_resistance 0.000000;
}
object line_spacing {
               name line_spacing_520;
               distance_AN 0.0833 ft;
}
```

Service Transformers

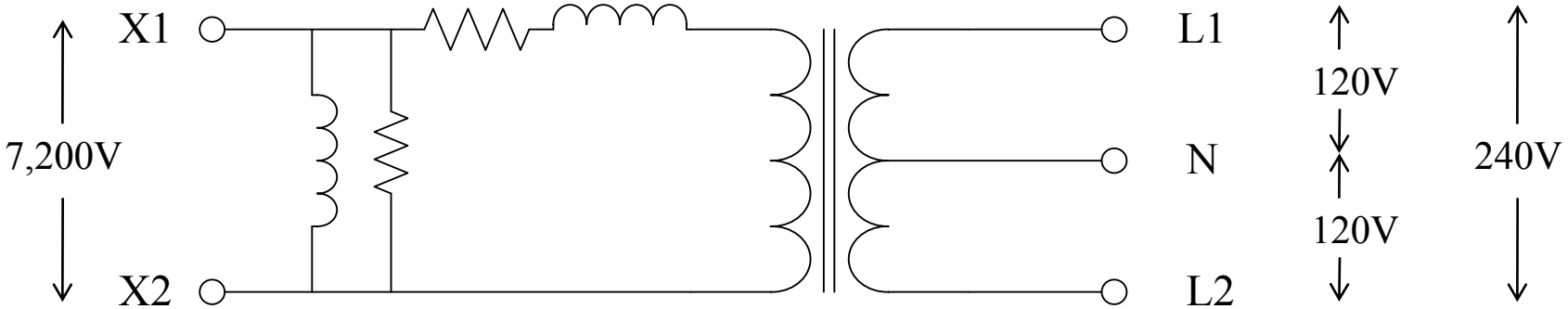
Poletop

Padmount

Underground Vault



Service Transformer Model



Transformer Code

Object Code

```
object transformer {
            groupid Distribution_Trans;
            configuration tc100100B;
            name f5_T21400253B;
            from f5_L2691959;
            to f5_X2691959B;
            nominal_voltage 7200;
            phases BS;
}
```

Configuration Code

```
object transformer_configuration {
            name tc100100B;
            connect_type SINGLE_PHASE_CENTER_TAPPED;
            install_type POLETOP;
            primary_voltage 7200.0;
            secondary_voltage 120.0;
            power_rating 100.0;
            powerB_rating 100.0;
            impedance 0.006+0.0136j;
            // Only used in Wye-Wye and CTTF
            impedance1 0.012+0.0204j;
            impedance2 0.012+0.0204j;
            shunt_impedance 259200+103680j;
}
```

Voltage Regulators

SLAC

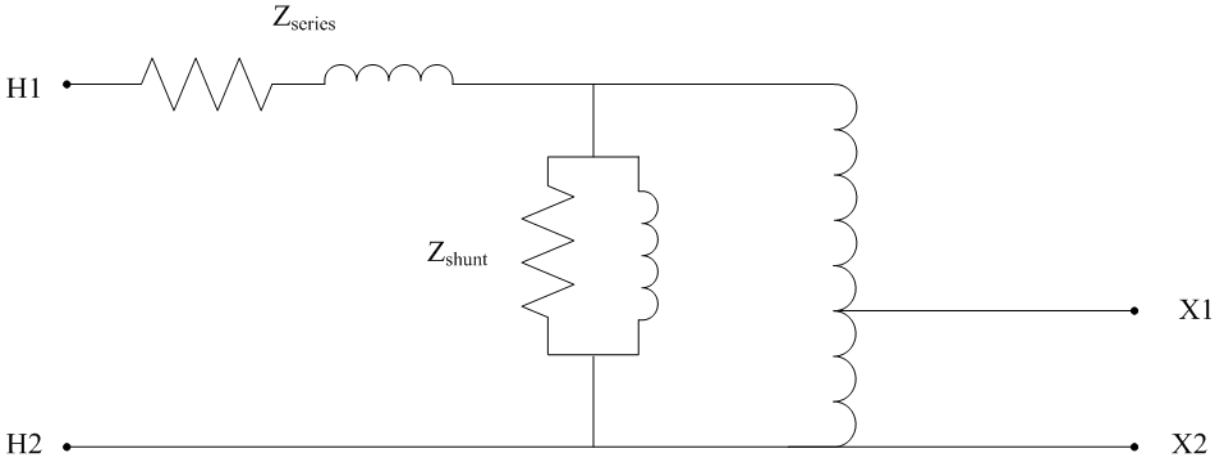
Voltage regulators adjust the voltage on a branch

- Compensates for heavy (or light) loading
- Controls voltage drop (or rise) that results from load changes

Voltage Regulators

SLAC

Two winding auto transformers with a variable tap ratio



- Note: GridLAB-D represents voltage regulators as ideal transformers (no losses)

Voltage Regulator Code

Object Code

```
object regulator {
            name reg1501491;
            phases "ABC";
            from node_150;
            to node_1491;
            configuration regulator_config_15;
}
```

Configuration Code

```
object regulator_configuration {
            name regulator_config_15;
            connect_type WYE_WYE;
            band_center 120.000;
            band_width 2.0;
            time_delay 30.0;
            raise_taps 16;
            lower_taps 16;
            current_transducer_ratio 700;
            power_transducer_ratio 20;
            compensator_r_setting_A 3.0;
            compensator_x_setting_A 7.5;
            compensator_r_setting_B 3.0;
            compensator_x_setting_B 7.5;
            compensator_r_setting_C 3.0;
            compensator_x_setting_C 7.5;
            CT_phase "ABC";
            PT_phase "ABC";
            regulation 0.10;
            Type A;
            tap_pos_A 7;
            tap_pos_B 7;
            tap_pos_C 7;
            Control MANUAL;  // OUTPUT_VOLTAGE,
                             // LINE_DROP_COMP or REMOTE NODE
}
```

Shunt Capacitors

Capacitor Code

Object Code

```
object capacitor {
            phases ABCN;
            name CAP1;
            pt_phase ABCN;
            parent 675;
            phases_connected ABCN;
            control VOLT;
            voltage_set_high 2700.0;
            voltage_set_low 2250.0;
            capacitor_A 0.10 MVAr;
            capacitor_B 0.10 MVAr;
            capacitor_C 0.10 MVAr;
            control_level INDIVIDUAL;
            time_delay 300.0;
            dwell_time 0.0;
            switchA CLOSED;
            switchB CLOSED;
            switchC CLOSED;
            nominal_voltage 2401.7771;
}
```

- Control modes available: **`MANUAL`**, **`VOLT`**, **`VAR`**, **`VARVOLT`** and **`CURRENT`**
- **`pt_phase`** determines the phases to monitor for controls
- **`nominal_voltage`** determines the impedance value
- **`control_level`** may also be **`BANK`**
- **`lockout_time`** allows minimum time between control operations
- **`remote_sense`**/**`remote_sense_B`** may be used for remote sensing operations

End Use Load Modeling

End use loads can be modeled in a variety of ways depending on the purpose.

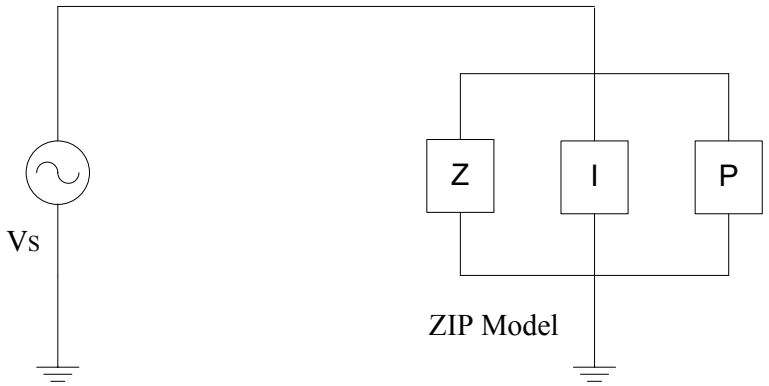
- Time-invariant load models (ZIP models)
- Time-variant load models
- Physical/performance load models (motor/power electronics)
- Multi-state/control response load models
- Dynamic response models (Deltamode)

Note: Most of the loads in GridLAB-D are represented in the residential module, but the basic theory is described here.

Time-invariant load models

Used for static power flow solution

- Used for capacity planning: peak load is examined
- Traditional ZIP load model

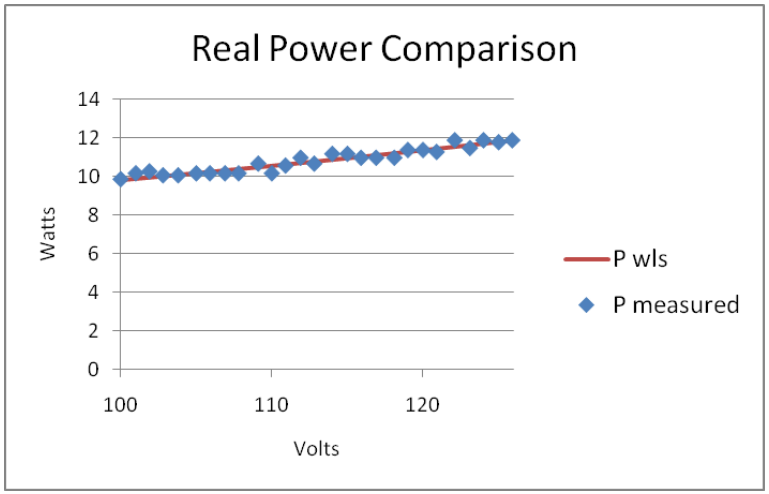


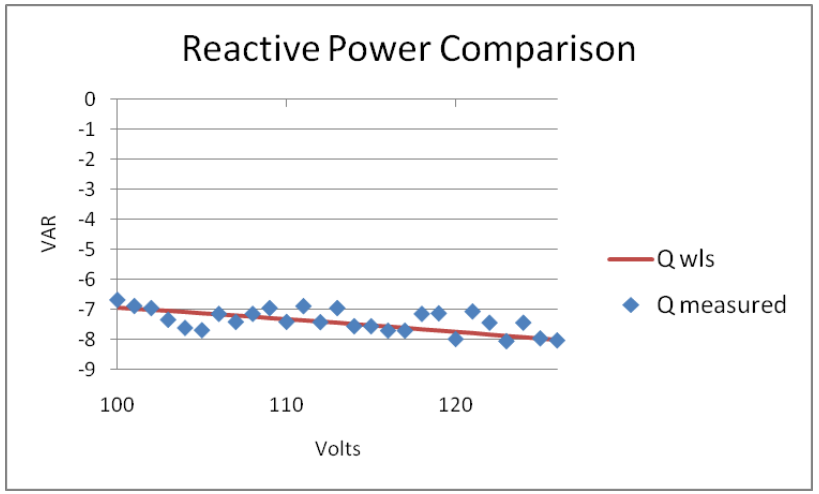
$$P_i = \frac{|V_a|^2}{|V_n|^2} S_n Z_{\%} \cos(Z) + \frac{|V_a|}{|V_n|} S_n I_{\%} \cos(I) + S_n P_{\%} \cos(P)$$

$$Q_i = \frac{|V_a|^2}{|V_n|^2} S_n Z_{\%} \sin(Z) + \frac{|V_a|}{|V_n|} S_n I_{\%} \sin(I) + S_n P_{\%} \sin(P)$$

$$1 = Z_{\%} + I_{\%} + P_{\%}$$

ZIP Load Model for 13W CFL





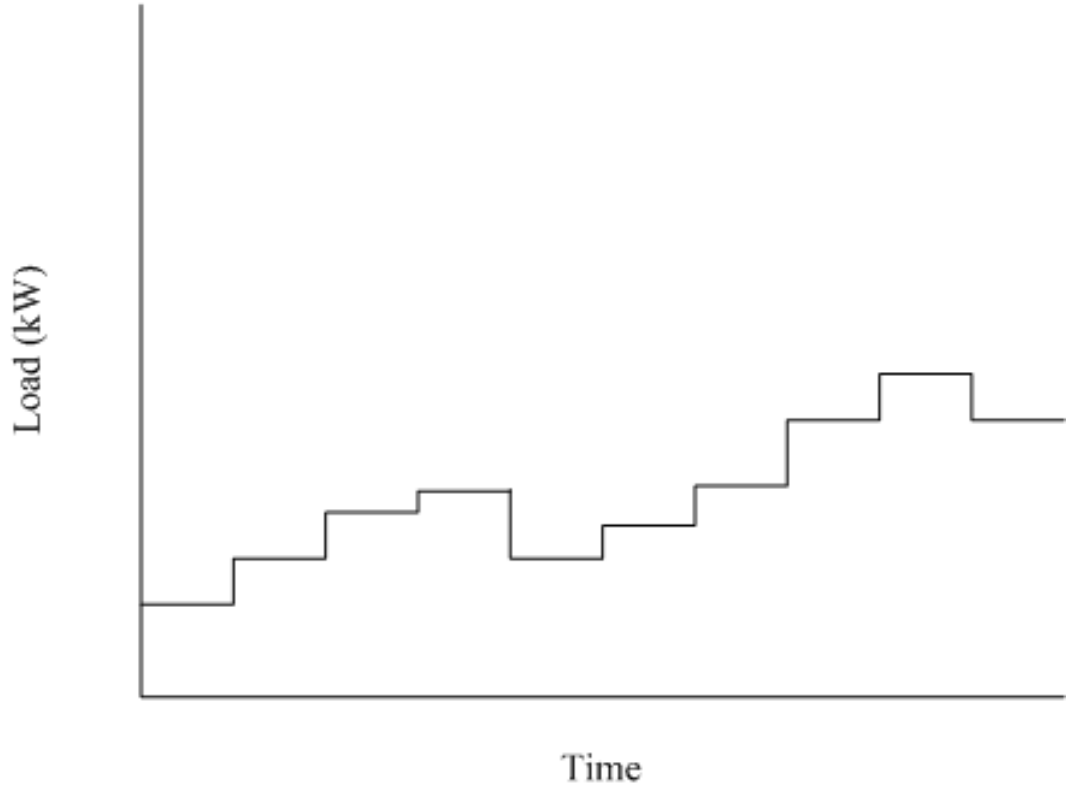
	ZIP Values					
	Z_%	I_%	P_%	Z_pf	I_pf	P_pf
CFL-13W	40.85%	0.67%	58.49%	-0.88	0.42	-0.78

Time-variant load models

SLAC

Often referred to as load profiles or load shapes

- Can be power or ZIP values that change with time
- Often comes from measured data with aggregated value



ZIP Load Code

Object Code

```
object ZIPload {
      groupid plugload;
      base_power plug1*2.828958;
      power_fraction 0.100000;
      impedance_fraction 0.800000;
      current_fraction 0.100000;
      power_pf 0.950000;
      current_pf 0.950000;
      impedance_pf 0.950000;
};
```

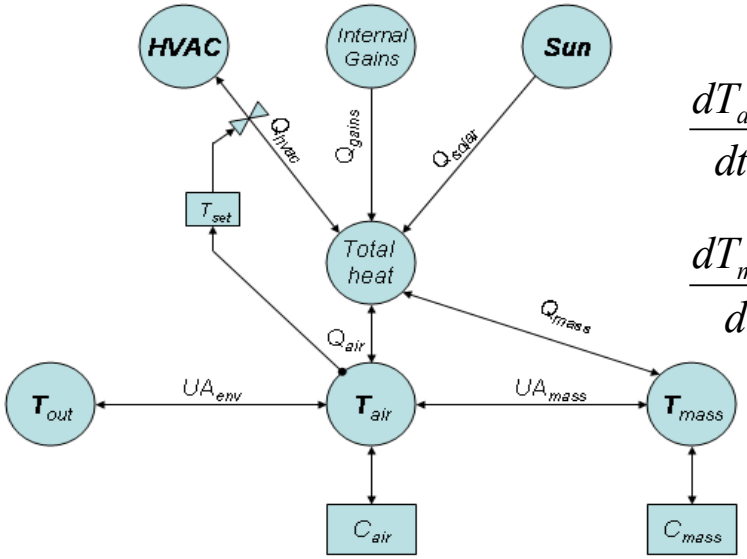
Schedule Code

```
schedule plug1 {
      0-23  0 * * * 0.1605;
      24-59 0 * * * 0.1381;
      0-30  1 * * * 0.1297;
      31-59 1 * * * 0.1289;
      0-34  2 * * * 0.1254;
      35-59 2 * * * 0.1301;
      0-29  3 * * * 0.1255;
      30-59 3 * * * 0.1276;
      0-18  4 * * * 0.1292;
      19-44 4 * * * 0.1399;
      45-59 4 * * * 0.1422;
      0-14  5 * * * 0.1438;
      15-35 5 * * * 0.1621;
      36-59 5 * * * 0.1806;
      0-14  6 * * * 0.1981;
      ...
}
```

Physical Load Models

Load models based on physical process

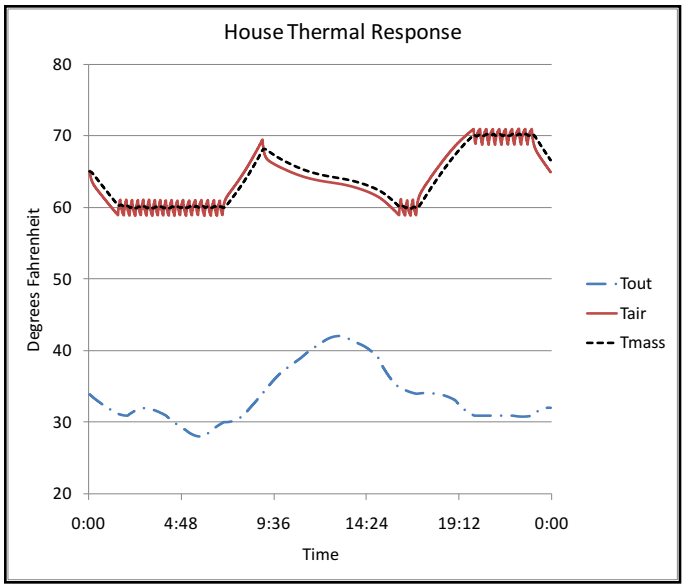
- HVAC, energy storage, thermal processes, etc.
- Usually time-variant model
- Can be significantly more complicated than ZIP models
- Usually cannot be modeled using a single LTI model
- Provide detail necessary to consider actual load behavior

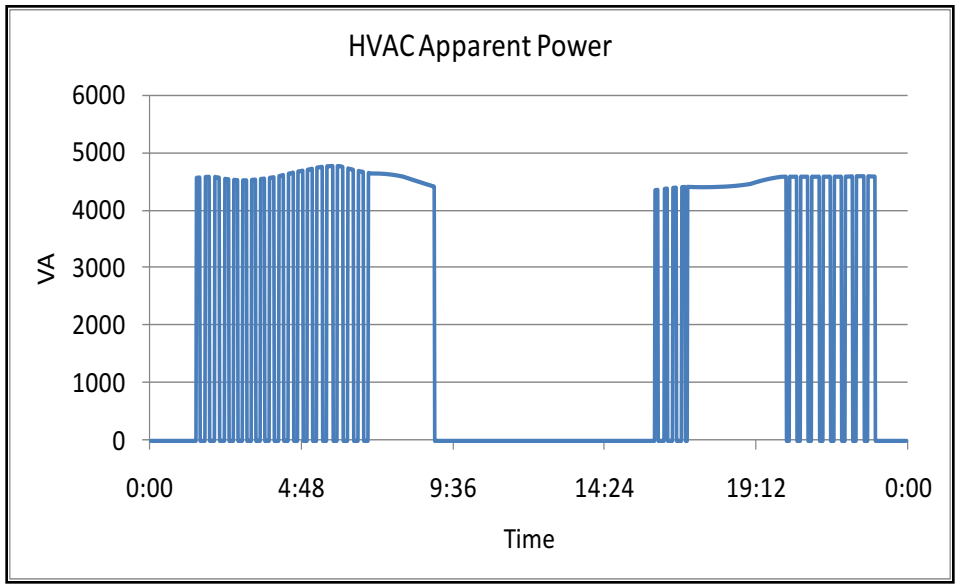


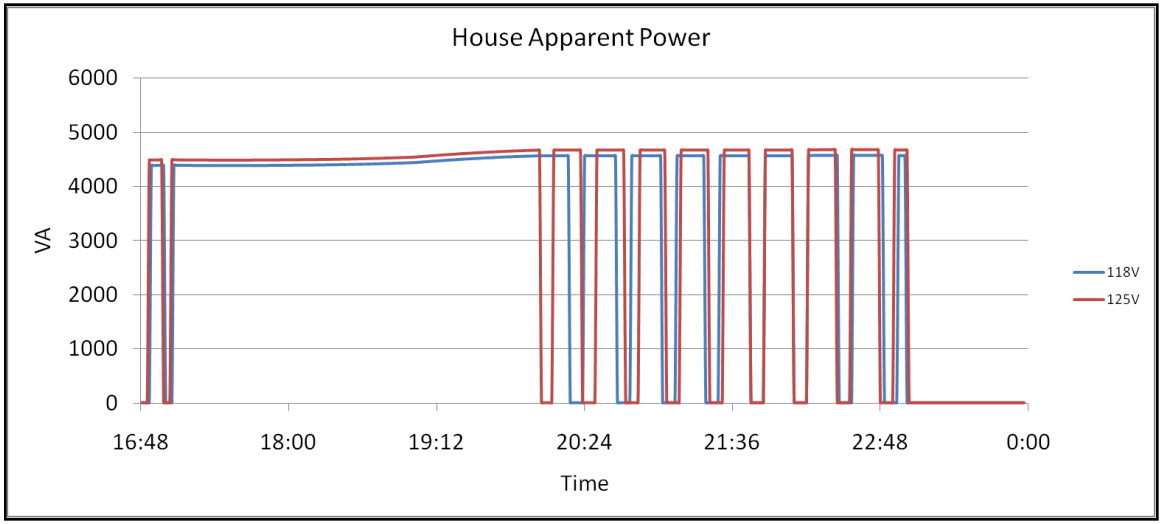
$$\frac{dT_{air}}{dt} = \frac{1}{C_{air}}\left[T_{mass}UA_{mass} \quad T_{air}\left(UA_{env}+UA_{mass}\right)+Q_{air}+T_{out}UA_{env}\right]$$

$$\frac{dT_{mass}}{dt} = \frac{1}{C_{mass}}\left[UA_{mass}\left(T_{air}+T_{mass}\right)+Q_{mass}\right]$$

Physical Load Model – Residential HVAC







Multi-State Physical Load Models

SLAC

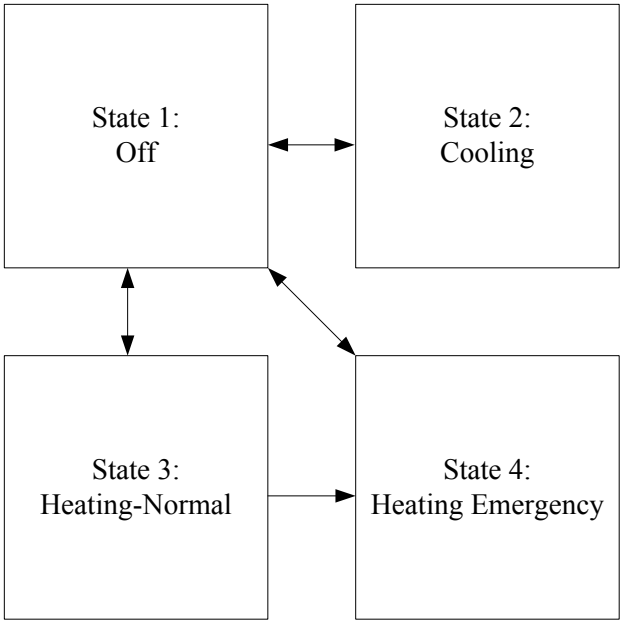
Allow for more than one state of operation

- Usually requires a control system to govern state dynamics
- Almost always results in non-LTI model
- Necessary for modern appliances & demand response

HVAC systems are a good example:

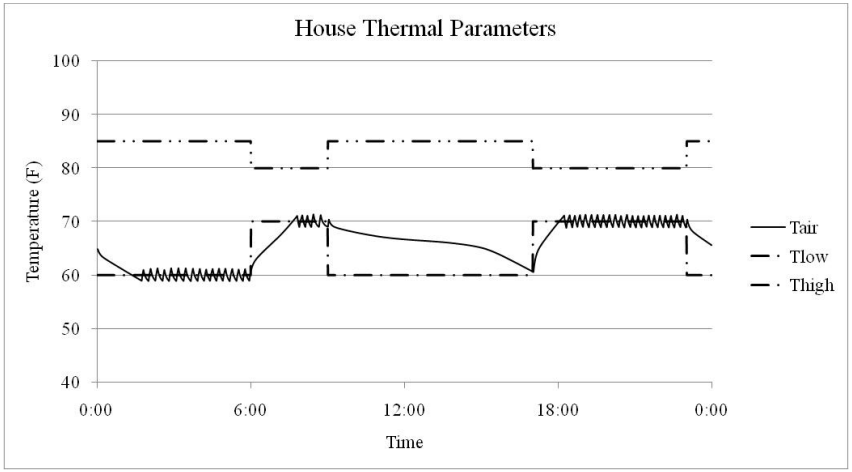
- State 1: Off
- State 2: Cooling
- State 3: Heating
- State 4: Auxiliary Heating

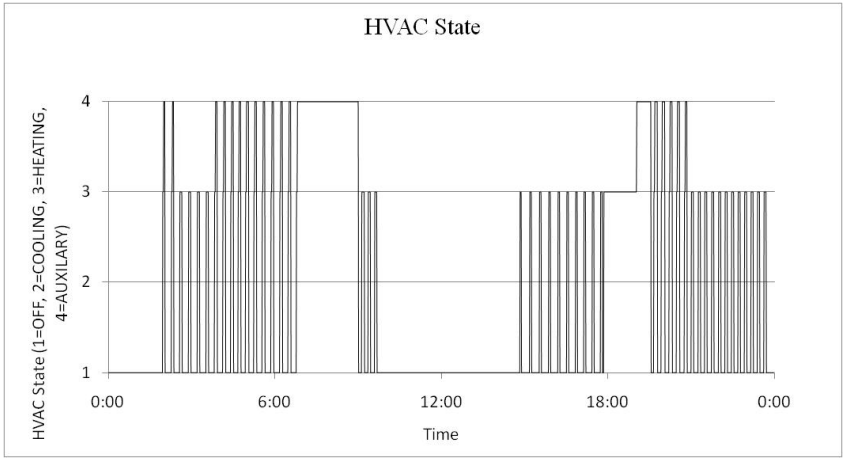
Multi-State Physical Load Models

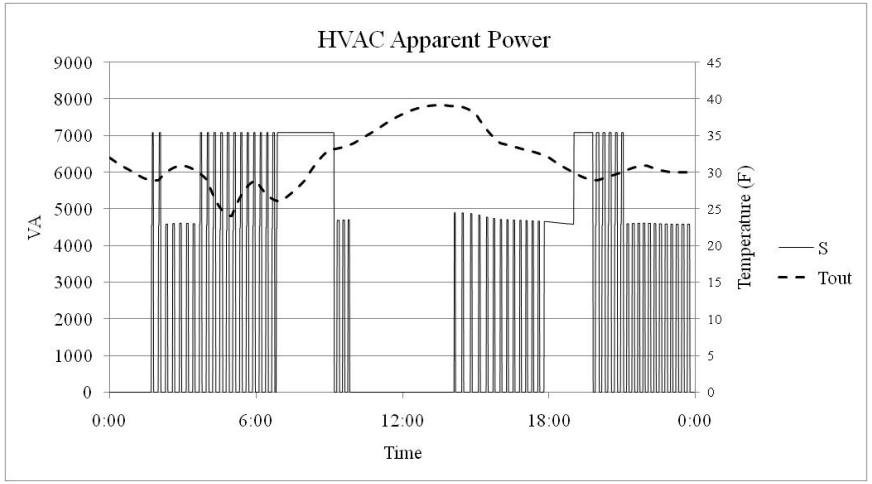


From State	To State	Transition Rule
1	2	$T_{air} > (T_{high} + DB_{high})$
1	3	$T_{air} < (T_{low} - DB_{low})$
1	4	$T_{air} < (T_{low} - DB_{low})$ & $T_{out} < T_{aux}$
2	1	$T_{air} < (T_{high} - DB_{high})$
3	1	$T_{air} > (T_{low} + DB_{low})$
3	4	$T_{out} < T_{aux}$
4	1	$T_{air} > (T_{low} + DB_{low})$

Multi-State Physical Load Models







Distribution System
Power Flow Solution Methods

Given the source voltage and end use load, determine the voltage magnitude and angle at every node in the system.

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Two methods available

SLAC

Forward/Backward Sweep method

- Fast, efficient, and robust to initial conditions
- Cannot solve meshed systems
- Reference:
  - *W. H. Kersting, "Distribution System Modeling and Analysis, 2nd Edition", CRC Press, 2007.*

Newton-Raphson Method

- Can solve meshed systems
- Requires Jacobian
- Fast converges for good initial values, otherwise, may fail to converge
- References:
  - *P. A. N. Garcia, J. L. R. Pereira, S. Carneiro Jr., V. M. Da Costa, and N. Martins, "Three-Phase Power Flow Calculations using the Current Injection Method", IEEE Transaction on Power Systems, vol. 15, issue 4, pp. 508-514, May 2000.*
  - *Araujo, L.R.; Penido, D.R.R.; Carneiro, S.; Pereira, J.L.R.; Garcia, P.A.N., "A Comparative Study on the Performance of TCIM Full Newton versus Backward-Forward Power Flow Methods for Large Distribution Systems", in Proc. 2006 IEEE PES Power Systems Conference and Exposition, pp. 522-526.*

Forward/Backward Sweep (FBS) method

SLAC

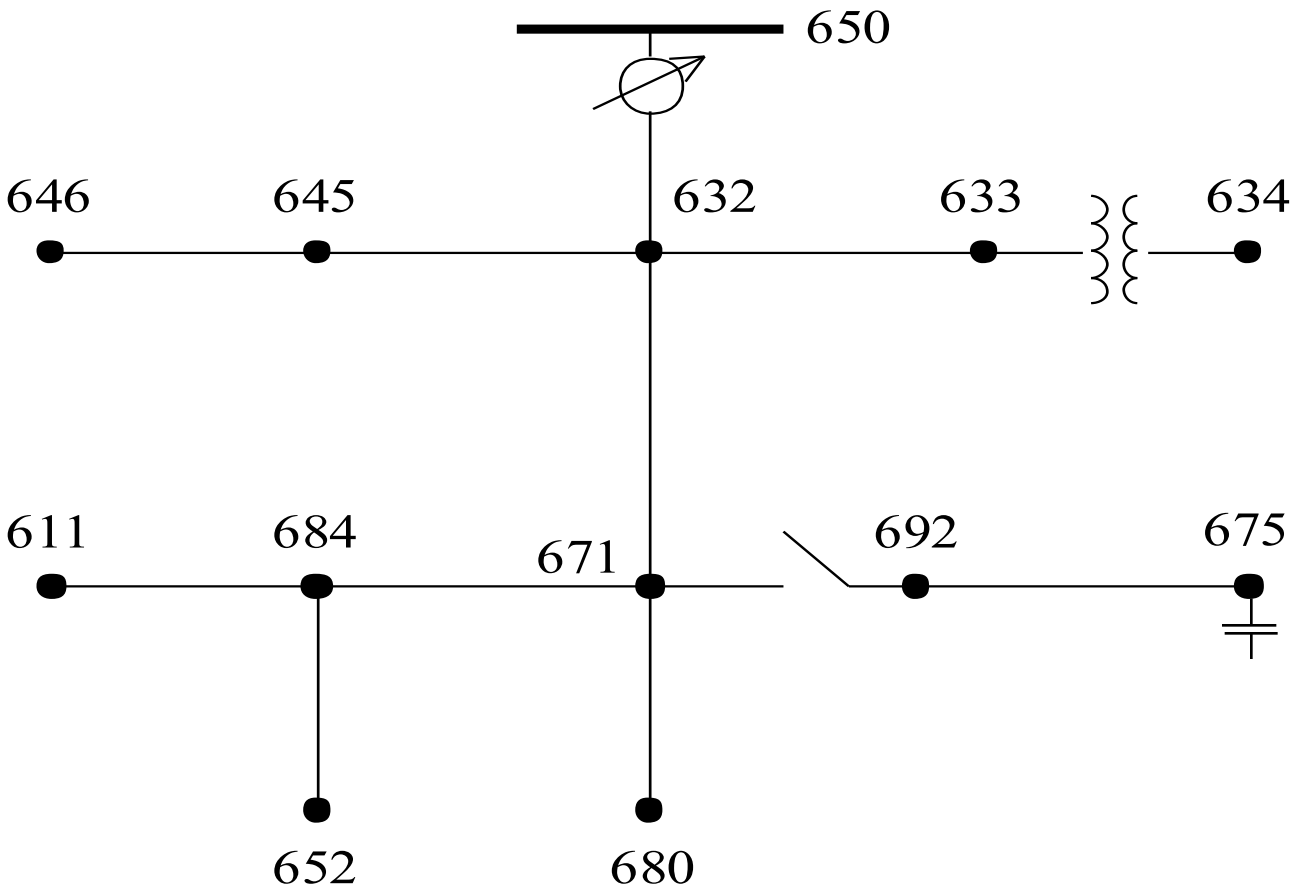
FBS method also called ladder solver

- Among earliest methods used for unbalanced power flow.
- Very efficient algorithm for radial systems
- Cannot solve looped systems
- Some components may make system appear non-radial
  - *Backflows from DG are allowed*
  - *Lateral switching may reverse currents*

Overall very good solution method, but has known limitations

- Deltamode and reliability modules don’t support FBS solver

Forward/Backward Sweep Example



Backward Sweep

SLAC

Starting with an assumed or known voltage and calculate the current to each load in the system

Node 634 load:
  Phase a: 160 kW; 110 kVar
  Phase b: 120 kW; 90 kVar
  Phase c: 120 kW; 90 kVar

Assume:
  Phase V_a: +2401 +0j
  Phase V_b: -1200 -2080j
  Phase V_c: -1200 +2080j

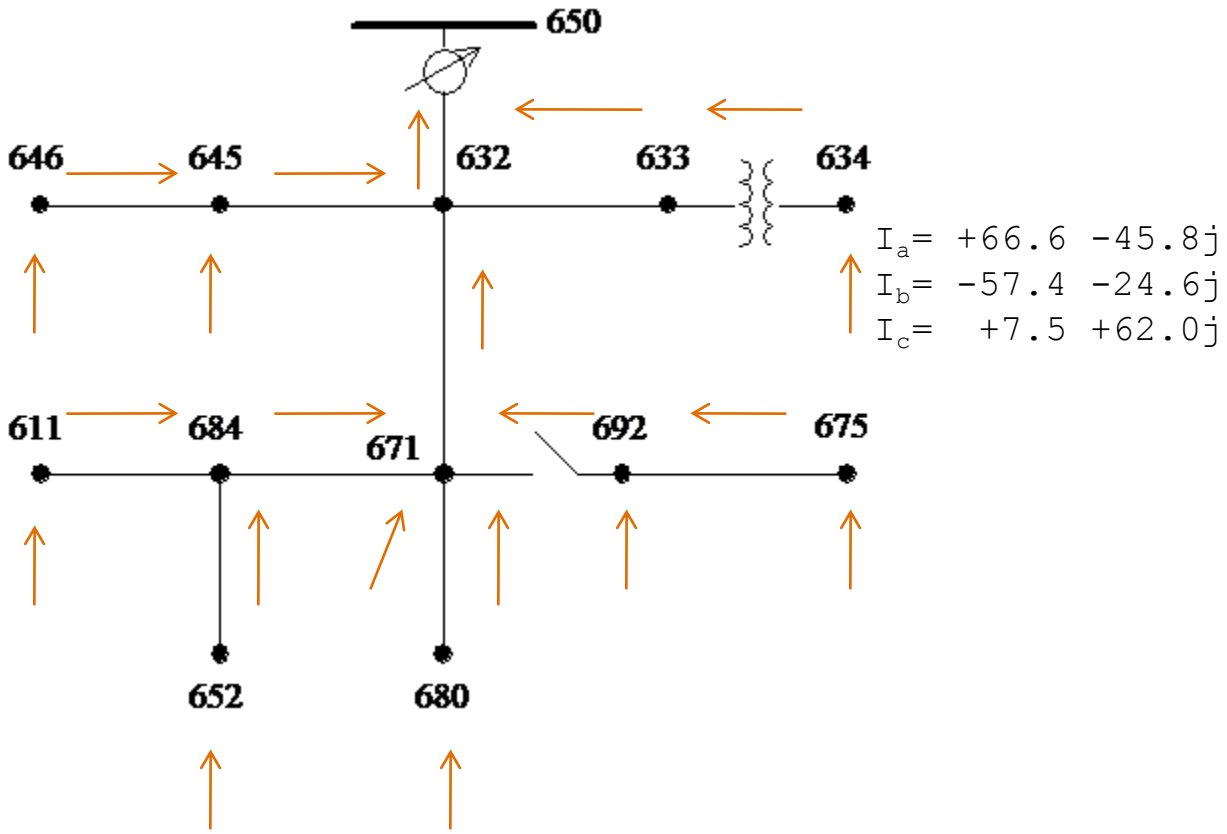
Given: $I=(S/V)^*$

Yields:
  I_a= +66.6 -45.8j
  I_b= -57.4 -24.6j
  I_c= +7.5 +62.0j

Backward Sweep

SLAC

Sum the currents at each node in the system until the current at the source is determined.



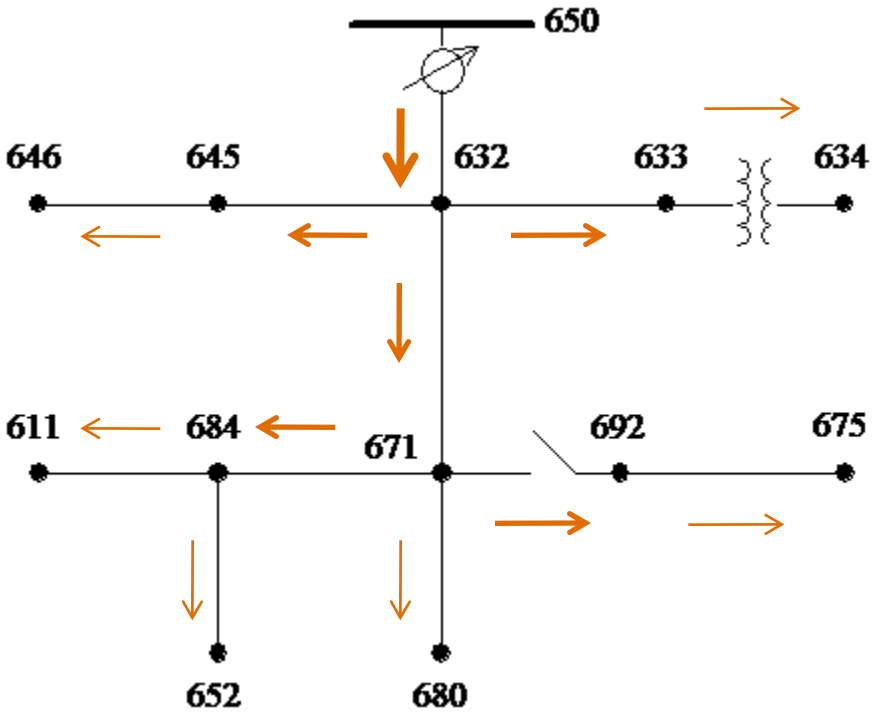
Forward Sweep

SLAC

Use currents to determine voltage drops on segments

- Voltage drop depends on load of other phases

  → complete per phase calculation must be used

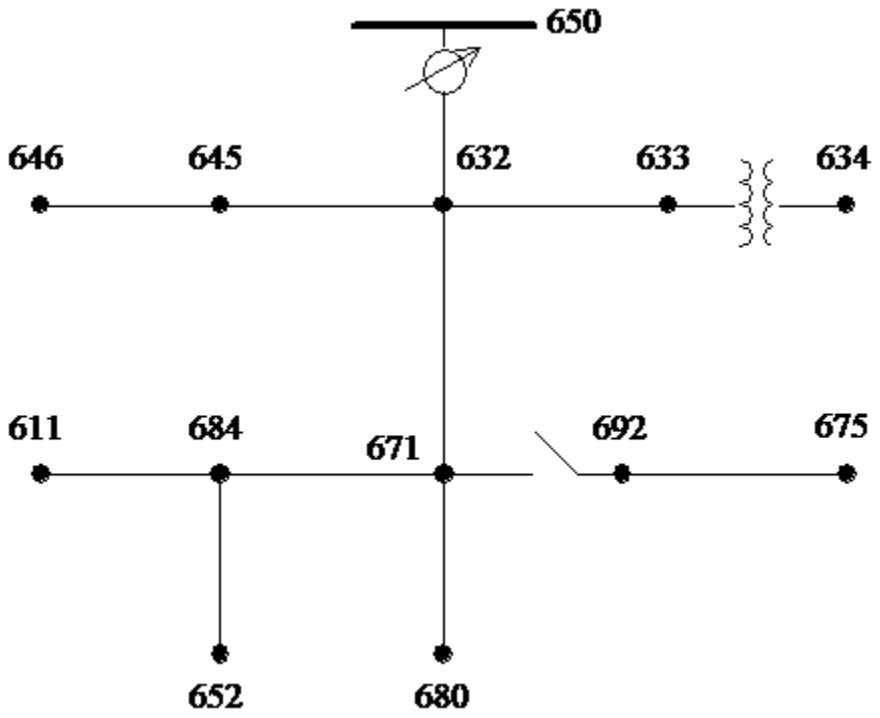


$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix}_n = \begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix}_m + \begin{bmatrix} Z_{aa} & Z_{ab} & Z_{ac} \\ Z_{ba} & Z_{bb} & Z_{bc} \\ Z_{ca} & Z_{cb} & Z_{cc} \end{bmatrix} * \begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix}$$

Recalculate Load

SLAC

Recalculate new current draw using new voltages for each load and perform another sweep



Newton-Raphson Method

SLAC

Developed for positive sequence transmission analysis

- Adapted for 3-phase unbalanced distribution analysis
- Based on the real/reactive power injections at each node
- Power injections used to update voltage magnitude and angle
- Requires the calculation and inversion of a Jacobian
- Computationally intensive

GridLAB-D implementation features

- Automatically constructs/updates Jacobian from models
- Uses previous results as initial condition → fast convergence

Newton-Raphson Method

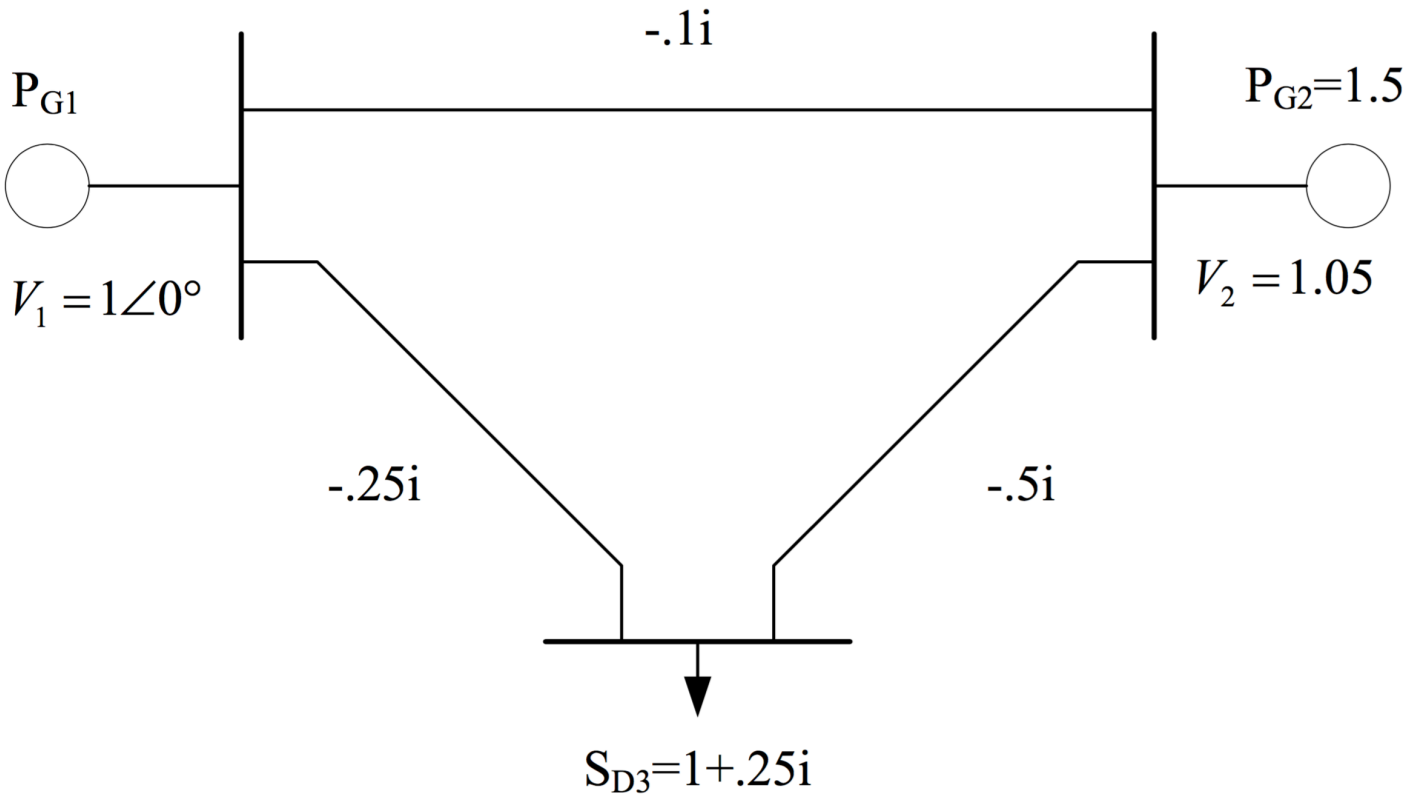
$$\Delta P_i = -P_i + \sum_{k=1}^{N} |V_i||V_k|\left[G_{ik}\cos(\theta_{ik}) + B_{ik}\sin(\theta_{ik})\right] = 0$$

$$\Delta Q_i = -Q_i + \sum_{k=1}^{N} |V_i||V_k|\left[G_{ik}\sin(\theta_{ik}) + B_{ik}\cos(\theta_{ik})\right] = 0$$

$$\begin{bmatrix} \Delta\theta \\ \Delta|V| \end{bmatrix} = -J^{-1}\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} \qquad J = \begin{bmatrix} \frac{\delta\Delta P}{\delta\theta} & \frac{\delta\Delta P}{\delta|V|} \\ \frac{\delta\Delta Q}{\delta\theta} & \frac{\delta\Delta Q}{\delta|V|} \end{bmatrix}$$

P_i	the real power injection at node i
Q_i	the reactive power injection at node i
G_{ik}	real portion of the admittance between node i and k
B_{ik}	imaginary portion of the admittance between node i and k
V_i	the voltage magnitude at node i
δ_i	the voltage angle at node i

Newton-Raphson Example (Transmission Level Example)



Newton Raphson Procedure

$$f(x) = P_i(x) - P_i$$

$$J(x) = \begin{bmatrix} \frac{\partial P}{\partial \theta} & \frac{\partial P}{\partial V} \\ \frac{\partial Q}{\partial \theta} & \frac{\partial Q}{\partial V} \end{bmatrix}$$

$$[J(x)] \begin{bmatrix} \Delta\theta \\ \Delta V \end{bmatrix} = \begin{bmatrix} \Delta P(x) \\ \Delta Q(x) \end{bmatrix}$$

Newton Raphson Procedure

$$\begin{bmatrix} \theta^{n+1} \\ V^{n+1} \end{bmatrix} = \begin{bmatrix} \theta^{n} \\ V^{n} \end{bmatrix} - \left[J\left(x^{n}\right)\right]^{-1} \begin{bmatrix} \Delta P\left(x^{n}\right) \\ \Delta Q\left(x^{n}\right) \end{bmatrix}$$

$$\begin{bmatrix} \theta^{n+1}{}_{2} \\ \theta^{n+1}{}_{3} \\ V^{n+1}{}_{3} \end{bmatrix} = \begin{bmatrix} \theta^{n}{}_{2} \\ \theta^{n}{}_{3} \\ V^{n}{}_{3} \end{bmatrix} - \begin{bmatrix} \frac{\partial P_2}{\partial \theta_2} & \frac{\partial P_2}{\partial \theta_3} & \frac{\partial P_2}{\partial V_3} \\ \frac{\partial P_3}{\partial \theta_2} & \frac{\partial P_3}{\partial \theta_3} & \frac{\partial P_3}{\partial V_3} \\ \frac{\partial Q_3}{\partial \theta_2} & \frac{\partial Q_3}{\partial \theta_3} & \frac{\partial Q_3}{\partial V_3} \end{bmatrix}^{-1} \begin{bmatrix} P_2\left(x_n\right) - 1.5 \\ P_3\left(x_n\right) - 1 \\ Q_3\left(x_n\right) - .25 \end{bmatrix}$$

N-R Power Flow Results

SLAC

	Initial	1	2	3	4
Theta 2	0	5.546556	5.593182	5.593524	5.593524
Theta 3	0	-7.43451	-7.88595	-7.89849	-7.89849
Voltage 3	1	0.9745	0.95735	0.956977	0.956977

Newton-Raphson Method (TCIM)

SLAC

- Instead of voltage magnitudes and angles, TCIM calculates the real and reactive components
- Voltage magnitudes and angle are then calculated from the components

$$\Delta I_{rk} = \frac{P_k^{sp} V_{rk} + Q_k^{sp} V_{mk}}{V_{rk}^2 + V_{mk}^2} - \sum_{i=1}^{n} \left(G_{ki} V_{ri} - B_{ki} V_{mi} \right) = 0$$

$$\Delta I_{mk} = \frac{P_k^{sp} V_{mk} - Q_k^{sp} V_{rk}}{V_{rk}^2 + V_{mk}^2} - \sum_{i=1}^{n} \left(G_{ki} V_{mi} - B_{ki} V_{ri} \right) = 0$$

$$\begin{bmatrix} \Delta V_{rk} \\ \Delta V_{mk} \end{bmatrix} = -J^{-1} \begin{bmatrix} \Delta I_{mk} \\ \Delta I_{rk} \end{bmatrix} \qquad J = \begin{bmatrix} \frac{\delta \Delta I_{mk}}{\delta V_{rk}} & \frac{\delta I_{mk}}{\delta V_{mk}} \\ \frac{\delta \Delta I_{rk}}{\delta V_{rk}} & \frac{\delta \Delta I_{rk}}{\delta V_{mk}} \end{bmatrix}$$

Time-series Simulations

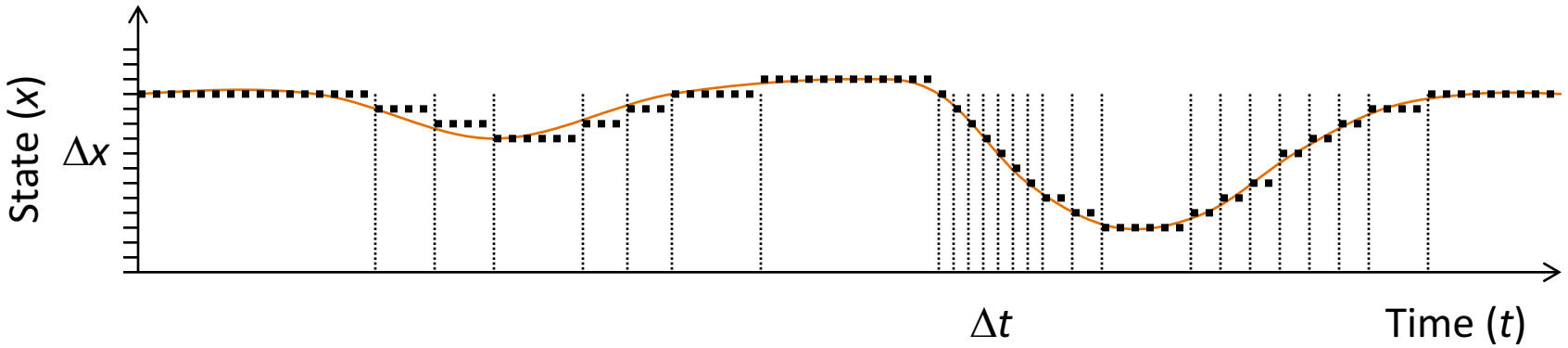
SLAC

Time series simulations necessary for certain analyses

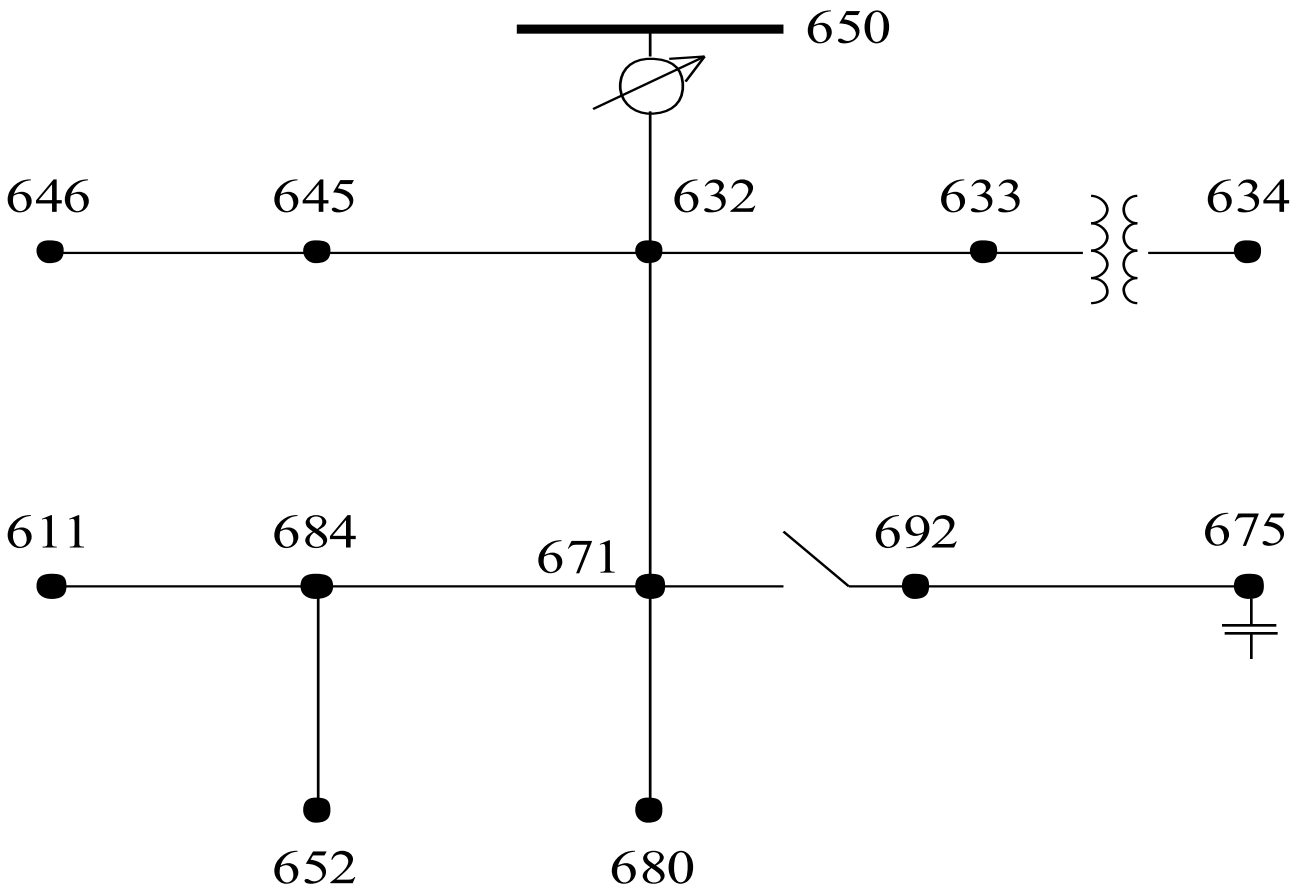
- Allows focus on “non-peak capacity” conditions
- Simulations are series of steady-state power flow solutions

Commonly called “quasi-steady” simulation

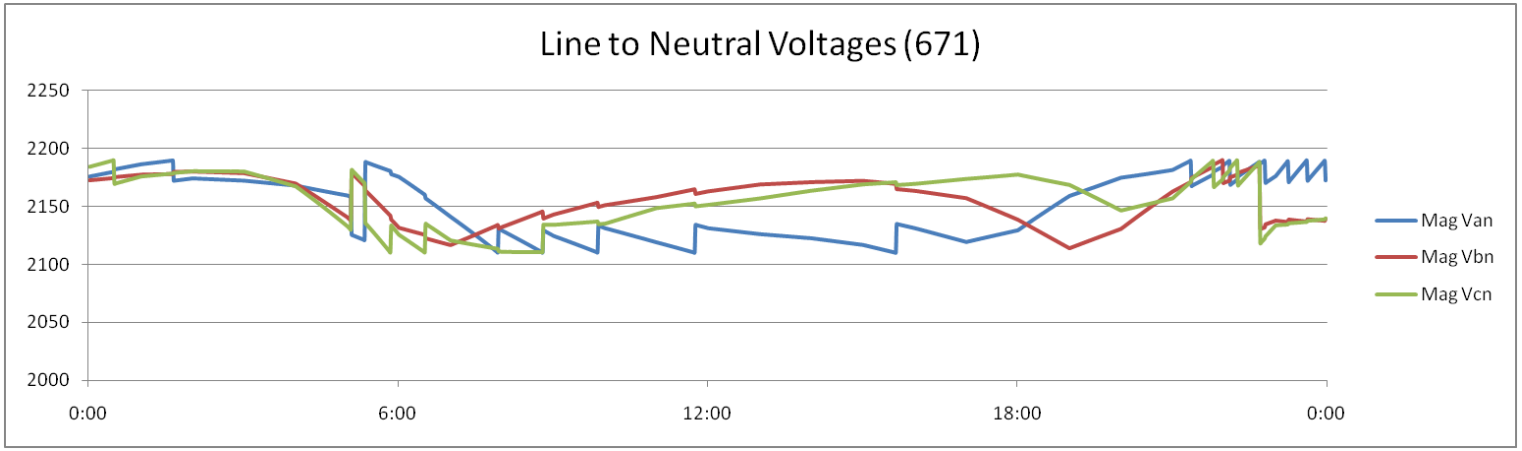
- Differ from dynamic or electromagnetic simulations

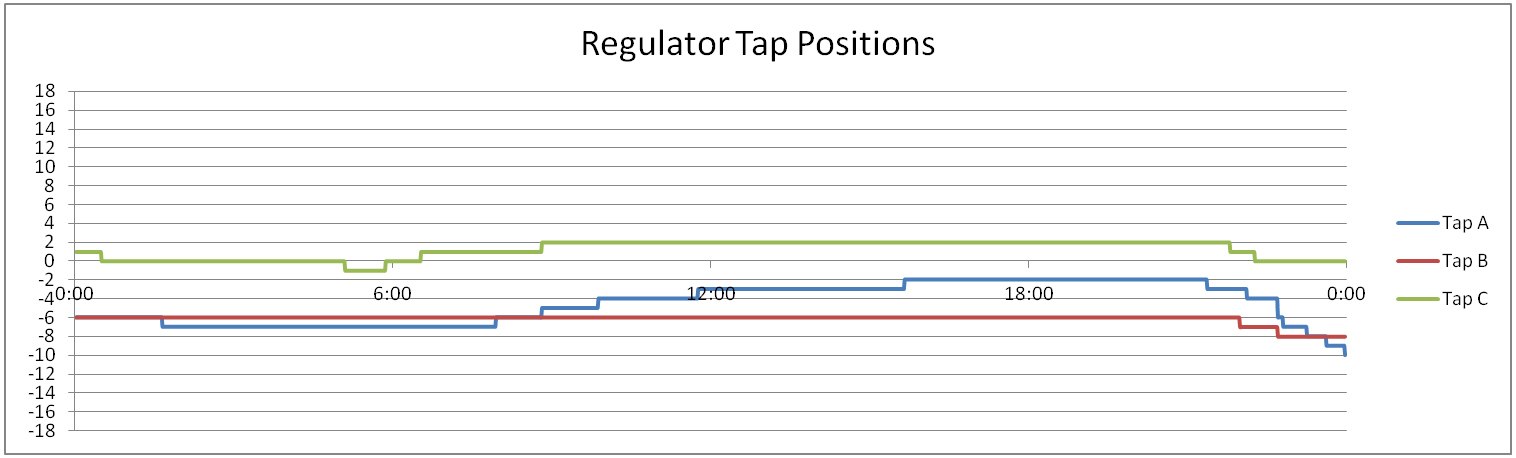


Modified IEEE 13 Node Test System



Time-series Simulations (II)





How does it work?

SLAC

Distribution system handled as network (nodes & links)

Forward-Backward Sweep (FBS)

- Voltage (nodes) and current (links) passed in sweeps
- Sweeps repeated each time step until zero voltage “error”

Newton-Raphson (NR)

- Admittance matrix built from links
- Used to calculate node voltages given current injections
- Iterations performed by SuperLU until zero voltage “error”

Iterations by GridLAB-D handles control algorithms

Powerflow module directives

SLAC

Preferred method depends on application

- As implemented, FBS is faster
- As implemented, NR handles meshed systems, reconfiguration algorithms, and reliability

Both methods support all objects in all other modules

- Exceptions for FBS:
  - *Reconfiguration, reliability, and deltamode (dynamics).*

```
module powerflow {
        solver_method FBS; // or NR; GS is deprecated as V3.0
        default_maximum_voltage_error 1e-6; // defines convergence criteria
        NR_iteration_limit 50; // max SuperLU iterations per GLD iteration
        NR_superLU_procs 1; // max processors assigned to SuperLU
        }
```

Objects in Powerflow Module

SLAC

Two main object types: node and link

- Most powerflow objects inherit properties from node and link

Library objects

- Specify configurations that are used multiple times
  - *line_configuration*
  - *line_spacing*
  - *transformer_configuration*
  - *regulator_configuration*
- Provide “library” values to compute model parameters
- Not used directly in constructing solutions

Abstract node and link objects

SLAC

FBS calculation done at level of node and link objects

- Node Objects

```
Capacitor                  Billdump                  Volt_var_control
Load                       Voltdump                  Currdump
Meter/triplex_meter        Motor (experimental)      Emissions
PQLoad                     Frequency_gen
Substation
```

- Link Objects

```
Fuse                       Overhead_line             Series_reactor
Switch_object              Underground_line          Regulator
Relay                      Triplex_line              Transformer
Recloser
Sectionalizer
```

Node objects

SLAC

```
class node {
  complex voltage_A[V];  // used to initialize voltage solution
  complex voltage_B[V];  // then tracks voltage over time
  complex voltage_C[V];
  complex voltage_AB[V]; // used for delta voltage connections
  complex voltage_BC[V];
  complex voltage_CA[V];
  complex current_A[A];
  complex current_B[A];
  complex current_C[A];
  complex power_A[W];
  complex power_B[W];
  complex power_C[W];
  complex shunt_A[S];
  complex shunt_B[S];
  complex shunt_C[S];
  set phases; // A,B,C,D,N or S
  enum bustype; // PQ (default), PV, SWING
  double maximum_voltage_error;
}
```

Loads

SLAC

```
class load {
      // also inherits properties from node object
      complex constant_impedance_A[Ohm]; // Z_A
      complex constant_impedance_B[Ohm]; // Z_B
      complex constant_impedance_C[Ohm]; // Z_C
      complex constant_current_A[A];     // I_A
      complex constant_current_B[A];     // I_B
      complex constant_current_C[A];     // I_C
      complex constant_power_A[VA];      // P_A
      complex constant_power_B[VA];      // P_B
      complex constant_power_C[VA];      // P_C
}
```

May be used to define general purpose load classes

- Residential, industrial, commercial, agricultural, mixed class

Meters

SLAC

Meters are the interaction point between the powerflow module and the commercial and generator modules (often used on loads)

```
class meter {
         // also inherits properties from node object
        double measured_real_energy[Wh];
        double measured_reactive_energy[Vah];
        complex measured_power[VA]; // also measured_power_A, _B, _C
        double measured_real_power[W];
        double measured_reactive_power[W];
        complex measured_voltage_A[V];
        complex measured_voltage_B[V];
        complex measured_voltage_C[V];
        complex  measured_current_A[A];
        complex measured_current_B[A];
        complex measured_current_C[A];
}
```

Triplex meters

SLAC

Triplex meters are interaction point between powerflow module and residential module

- Billing information may be extracted using bill_mode.

```
Class triplex_meter {
        // also inherits properties from node object
        double measured_energy[Wh];
        double measured_power[W];
        double measured_real_power[W];
        double measured_reactive_power[W];
        complex measured_voltage_1[V];
        complex measured_voltage_2[V];
        complex measured_voltage_N[V];
        complex measured_current_1[A];
        complex measured_current_2[A];
        complex measured_current_N[A];
}
```

Link objects

SLAC

No “generic” link object (unlike node)

- underground_line and triplex_line follow the same format
- “to” and “from” define the two nodes that the link connects
- configuration is defined by a separate “library” object
- configuration and length determine all relevant matrices for powerflow calculations

```
class overhead_line {
        // inherited from link objects
        char64 name;
        set phases; // A,B,C,D,N or S
        object from;
        object to;

        // Specific to line objects
        double length[ft];
        object configuration;
}
```

Lines

SLAC

All lines may be specified directly using a z-matrix (not the best way to do it)

- Matrix symmetry checks are not performed
- Negative resistance gives a warning but are accepted

```
class line_configuration {
        complex z11[Ohm/mile]; // line impedances
        complex z12[Ohm/mile];
        complex z13[Ohm/mile];
        complex z21[Ohm/mile];
        complex z22[Ohm/mile];
        complex z23[Ohm/mile];
        complex z31[Ohm/mile];
        complex z32[Ohm/mile];
        complex z33[Ohm/mile];
        double c11[nF/mile]; // shunt capacitances
        double c12[nF/mile];
        double c13[nF/mile];
        // etc.
}
```

Transformers

Specific transformer parameters are in library object

- Parameters are in `transformer_configuration`
- Allows definition of many transformer types

```
class transformer {
      // inherits from link object
      object from;
      object to;
      double power_in[W];
      double power_out[W];
      object configuration;
}
```

Transformer configurations

```
class transformer_configuration {
        enumeration connection_type;
        enumeration install_type;
        // continued ...
        }
```

Connection types

- WYE_WYE
- DELTA_DELTA
- DELTA_GWYE
- SINGLE_PHASE
- SINGLE_PHASE_CENTER_TAPPED

Install types

- POLETOP
- PADMOUNT
- VAULT

Transformer configuration (cont.)

SLAC

```
object transformer_configuration {
        // ... continued
        double primary_voltage[V];
        double secondary_voltage[V];
        double power_rating[kVA];
        double power_rating_A[kVA];
        double power_rating_B[kVA];
        double power_rating_C[kVA];
        double resistance[pu.Ohm];
        double reactance[pu.Ohm];
        complex impedance[pu.Ohm];
        complex impedance1[pu.Ohm]; // specifies individual
        complex impedance2[pu.Ohm]; // center taps impedances
        complex shunt_impedance[pu.Ohm]; // center tap and
                                         // wye-wye shunt loss
}
```

Regulators

SLAC

Regulator parameter definitions

- Specified by `regulator_configuration`
- `tap_A` etc. specify initial tap positions or manual control

`sense_node` used only for specific modes

- Specifies remote locations to observe voltage

```
class regulator {
      // inherited from link object
      object configuration;
      int16 tap_A;
      int16 tap_B;
      int16 tap_C;
      object sense_node;
}
```

Objects in powerflow Module

SLAC

```
class regulator_configuration {
            enumeration connect_type; //WYE_WYE is the only operational type at this time
            double band_center[V];
            double band_width[V];
            double time_delay[s];
            double dwell_time[s];
            double raise_taps;
            double lower_taps;
            double current_transducer_ratio[pu]; // used in line drop compensation
            double power_transducer_ratio[pu];
            double compensator_r_setting_A[V];
            double compensator_r_setting_B[V];
            double compensator_r_setting_C[V];
            double compensator_x_setting_A[V];
            double compensator_x_setting_B[V];
            double compensator_x_setting_C[V]; // used in line drop compensation mode
            string CT_phase;
            string PT_phase;
            double regulation;
            enumeration control; // MANUAL, OUTPUT_VOLTAGE, LINE_DROP_COMP, or REMOTE_NODE
            enumeration type; // A or B
            enumeration control_level; // INDIVIDUAL or BANK
            int tap_pos_A;
            int tap_pos_B;
            int tap_pos_C;
}
```

Node-to-Node Connection

SLAC

Connected via a link object

```
object overhead_line {
     name ovl_12;
     phases "ABCD";
     from node_1; // name or class:id
     to node_2; // name or class:id
     length 2000;
     configuration lineconfig_300;
}
```

Connected via parent/child relationship (length = 0)

```
object node {
     name node_2;
     phases "ABCD";
     parent node_1;
}
```

Feeder Example

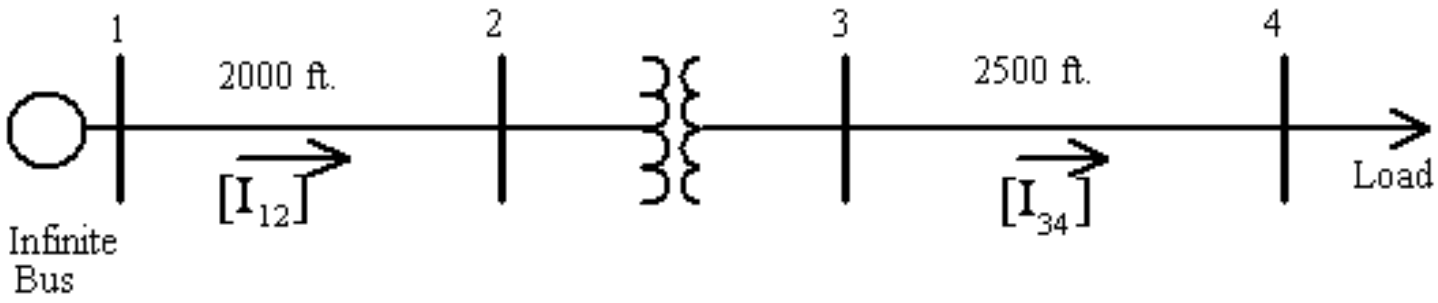
SLAC

IEEE 4 Node Feeder

- Presented as a system of nodes and links

Use FBS and NR method to solve

- Compare the solutions



Powerflow Output

3-ph voltage at each node, including inherited objects

- Power/current in/out
  - *Each phase*
  - *Each link,*
  - *Includes inherited objects*
- Additional information available for specific objects,
  - *Tap positions*
  - *Switch states*
- Log of events
  - *Output to KML if specified*

Sample Static Powerflow Output

SLAC

Load object: (node)

Name	constant_power_A	constant_power_B	constant_power_C	voltage_A	voltage_B	voltage_C
(load_11) (#11)	1.8e+006+871780j W	1.8e+006+871780j W	1.8e+006+871780j W	1918.94-271.698j V	-1266.24-1576.4j V	-697.555+1873.95j V

Transformer object: (link)

Name	configuration	status	From	to	power_in	power_out	phases
(transformer_8) (#8)	transformer_config_4	CLOSED	node7	node9	+7.17672e+006 W	+6.81855e+006 W	ABCD

Time-Series Powerflow Output

SLAC

Time-series gives a solution for each time step

- Series of .xml files is not practical:
  - *1 year at 1 minute resolution yields 525,600 solutions*

Use recorders to output desired values to .csv files.

These can then be post processed or plotted.

Exercise 1

Load one IEEE 4-node test feeder, run it, and examine the output file.

1. Run the feeder with different transformer configurations and note the effects.

Exercise 2

Load the IEEE 37-node test feeder, which will be used in other exercises.

1. There is a missing line. Find the missing line and fix the model to include the line.
2. Replace this line with a Delta-Delta transformer.
3. Connect an additional load via a node-to-node connection (e.g., without a link object).

Exercise 3

Load the IEEE 13-node test feeder, with building loads already attached.

1. Examine the interface between the powerflow module and the residential module at the triplex meter objects.