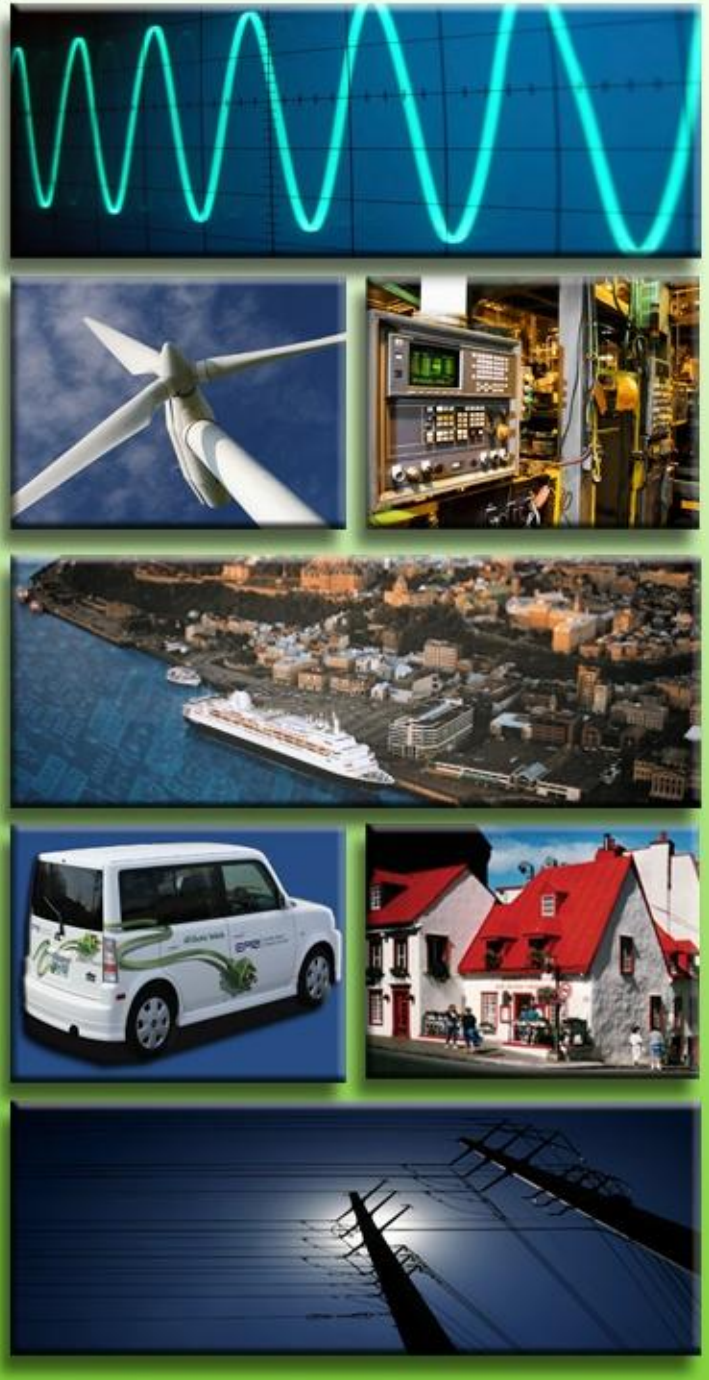


EPRI | ELECTRIC POWER RESEARCH INSTITUTE

# OpenDSS

## Open Source Distribution System Simulator

# Training Workshop

**Roger C. Dugan**
**Sr. Technical Executive, EPRI**

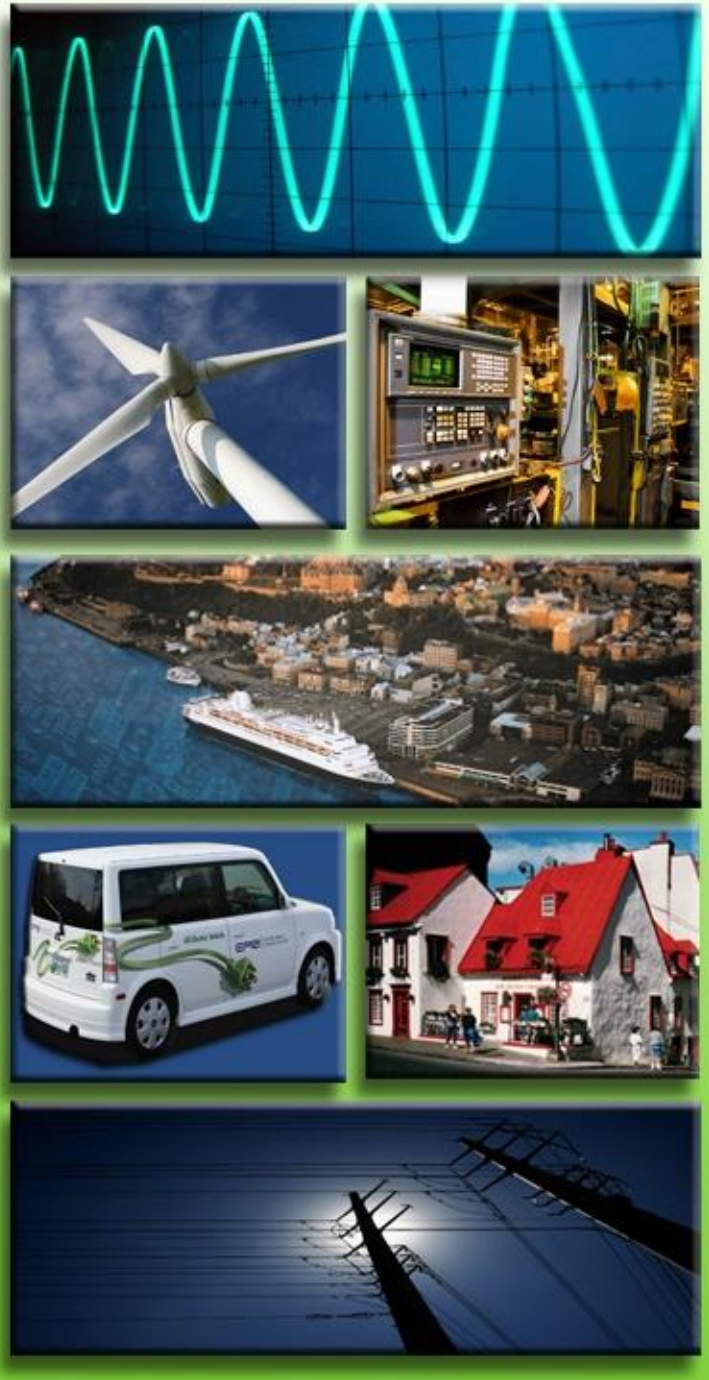
**Alumni House, Georgia Tech**

**Atlanta, GA**

Nov 15-16, 2011

# Agenda

| | |
|---|---|
| 8:30 AM - 9:00 AM | **Introduction to Open Distribution System Simulator** |
| 9:00 AM - 9:30 AM | **Introduction to Uses and Applications** |
| 9:30 AM - 10:00 AM | **Getting Started** |
| 10:00 AM - 10:15 AM | **Break** |
| 10:15 AM - 10:45 AM | **Open Distribution System Simulator Architecture and Circuit Modeling Basics** |
| 10:45 AM - 11:00 AM | **Power Flow and Harmonic Solution Modes** |
| 11:00 AM - 12:00 PM | **Scripting Basics** |
| 12:00 PM - 1:00 PM | **Lunch Break** |
| 1:00 PM - 2:30 PM | **Scripting for Larger Circuits** |
| 2:45 PM - 3:30 PM | **Loadshapes and Smart Grid Simulations** |
| 3:30 PM - 4:00 PM | **Intrinsic Simulation Modes** |
| 4:00 PM | **Adjourn for the Day** |



EPRI | ELECTRIC POWER RESEARCH INSTITUTE

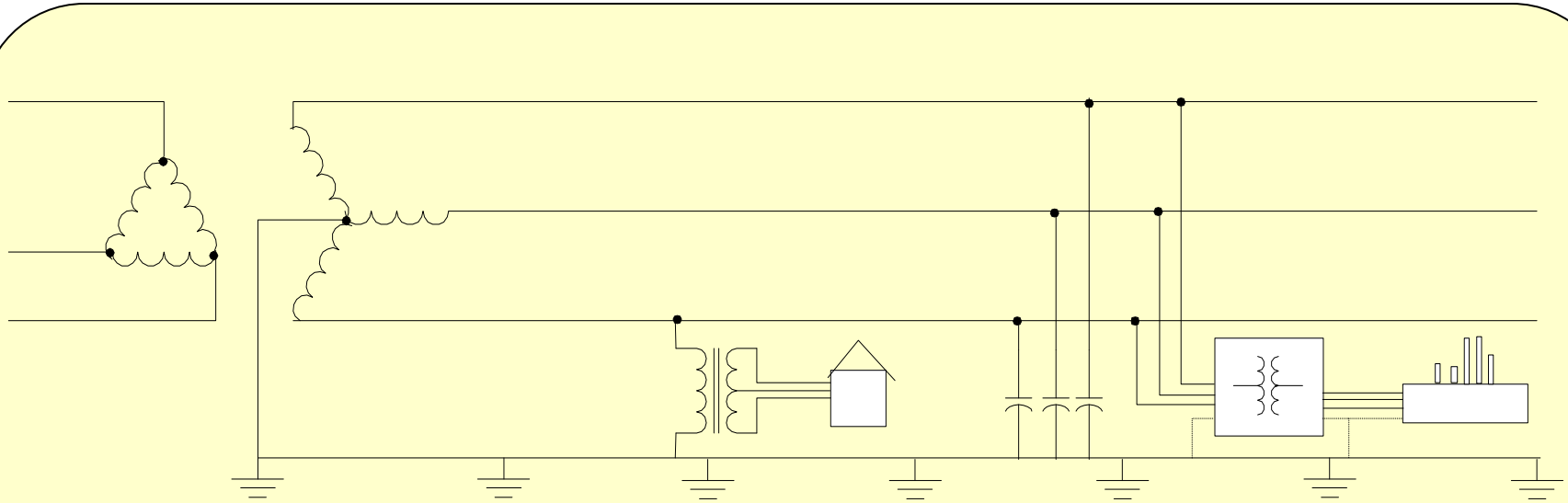
# Introduction to OpenDSS

# What is the OpenDSS?

- Script-driven, frequency-domain electrical circuit simulation tool

- Specific models for:
  - Supporting **utility distribution system** analysis
  - Unbalanced, multi-phase North American power distribution systems
    - As well as European-style systems
      - These typically have a simpler structure

# Typical North American Distribution System



**Typical 4-wire multi-grounded neutral system**

**Unigrounded/Delta 3-wire also common on West Coast**

# What is the OpenDSS? (cont'd)

- Heritage
  - **Harmonics solvers** rather than **power flow**
    - Gives OpenDSS extraordinary distribution system modeling capability
  - Simpler to solve power flow problem with a harmonics solver than vice-versa
- Supports all rms steady-state (i.e., frequency domain) analyses commonly performed for utility distribution system planning
  - And many new types of analyses
  - Original purpose: DG interconnection analysis

# What is the OpenDSS? (cont'd)

- What it Isn't
  - An *Electromagnetic* transients solver (Time Domain)
    - It can solve *Electromechanical transients*
      - Frequency Domain => "Dynamics"
      - All solutions are in ***phasors*** (complex math)
  - Not a Power Flow program
  - Not a radial circuit solver
    - Does meshed networks just as easily
  - Not a distribution data management tool
    - It is a simulation engine designed to work with data extracted from one or more utility databases

# Distribution System Simulator (DSS)

- Designed to simulate utility distribution systems
  - In arbitrary detail
  - For most types of analyses related to distribution planning.

- It performs its analysis types in the *frequency domain*,
    - Power flow,
    - Direct linear circuit solution,
    - Harmonics, and
    - Dynamics.

  - It does NOT perform electromagnetic transients (*time domain*) studies.

# Time- and Location-Dependent Benefits

- The OpenDSS was designed from the beginning to capture both
  - **Time-specific benefits** and
  - **Location-specific benefits**
- Needed for
  - DG analysis
  - Renewable generation
  - Energy efficiency analysis
  - PHEV and EV impacts
  - Other proposed capacity enhancements that don't follow typical loadshapes

# Time- and Location-Dependent Benefits

- Traditional distribution system analysis programs
  - Designed to study **peak loading** conditions
  - Capture mostly **location-specific** benefits
  - Ignores time; Assumes resource is available
  - This gets the wrong answer for many DG, energy efficiency, and Smart Grid analyses

- Must do **time sequence analysis** to get the right answer
  - Over distribution planning area

# What are the Key Features?

- See Documentation for more details
  - http://electricdss.svn.sourceforge.net/viewvc/electricdss/Doc/
  - Main page in Wiki:
  - sourceforge.net/apps/mediawiki/electricdss/index.php?title=Main_Page
- Designed to allow expansion indefinitely
  - Impossible to anticipate everything users will want to do
  - COM interface allows easier customization

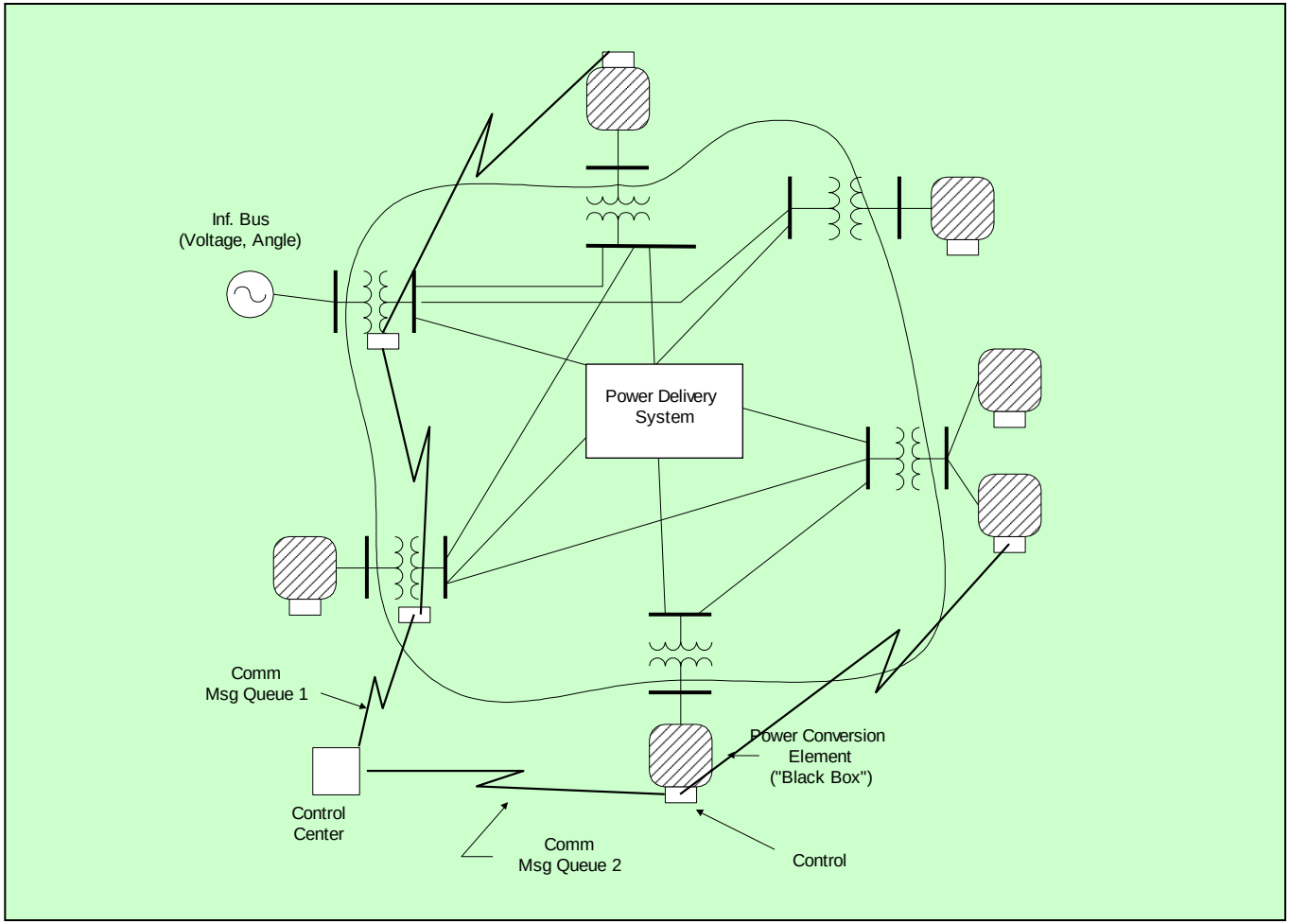
# Built-in Solution Modes

- Snapshot (static) Power Flow
- Direct (non-iterative)
- Daily mode (default: 24 1-hr increments)
- Yearly mode (default 8760 1-hr increments)
- Duty cycle (1 to 5s increments)
- Dynamics (electromechanical transients)
- Fault study
- Monte carlo fault study
- Harmonic
- Custom user-defined solutions

# Controls

- A key feature is that controls are modeled separately from the devices being controlled
  - Capacitors
  - Regulators/tapchangers
- Control Modes
  - Static
    - Power flows with large time steps
  - Time
    - Control queue employed to delay actions
    - Control acts when time is reached
  - Event

# Overall Model Concept



# User Interfaces Currently Implemented

- A stand-alone executable program that provides a text-based interface (multiple windows)
  - Some graphical output is also provided.
  - No graphical input is provided.
- An **in-process COM server** (for Windows) that supports driving the simulator from user-written programs.
  - (An out-of-process COM server is under development to support execution from 64-bit programs.)
- **Program 174** funders have access to *DGScreener*

# What is the Value for New/Current Users?

- System Loss studies
  - Annual simulation
- Assessment of volt-var control strategies
  - Advanced & new VVO models
- Evaluation of tap-changer operations
- Dynamic analysis of distribution system performance
- Combined transmission and distribution system studies
  - “holistic” assessment of power system response

# Other Value

- New capabilities can be added faster
- Special capabilities for DG
  - Evaluation of impact on losses
  - Impact on capacity
  - Risk of unserved energy analysis
  - Voltage rise/fluctuations/flicker
  - Interaction with regulator controls and capacitor switching
  - High penetration 1-phase DG
  - Short circuit contribution
  - Backfeed analysis (w/ transformer connections)
  - Islanding

# Other Value (cont'd)

- Users can model nearly any unbalanced condition accurately,
  - Unbalance power flow
  - Open-conductor faults
  - Unusual transformer configurations
  - Stray voltage (neutral-to-earth voltage)
  - Transmission overbuild falling into distribution
- Users can control the simulation from their own software.
  - Using COM interface

# Validation of OpenDSS

- EPRI routinely checks OpenDSS power flow results against CYMDIST, SYNERGEE, and WindMil programs after converting data sets for various projects.
- The OpenDSS program has been benchmarked against all the IEEE Test Feeders
  - (http://ewh.ieee.org/soc/pes/dsacom/testfeeders/).
  - OpenDSS was used to develop the NEV test feeder and the 8500-node test feeder.
  - Being used to develop the DG Protection test feeder and the large urban network test feeder.
- For the EPRI Green Circuits project, computed load characteristics were calibrated against measured data.
- Recently, the new *PVSystem* model was compared to actual measurements with excellent results.

# Plan for Future Work/Enhancements

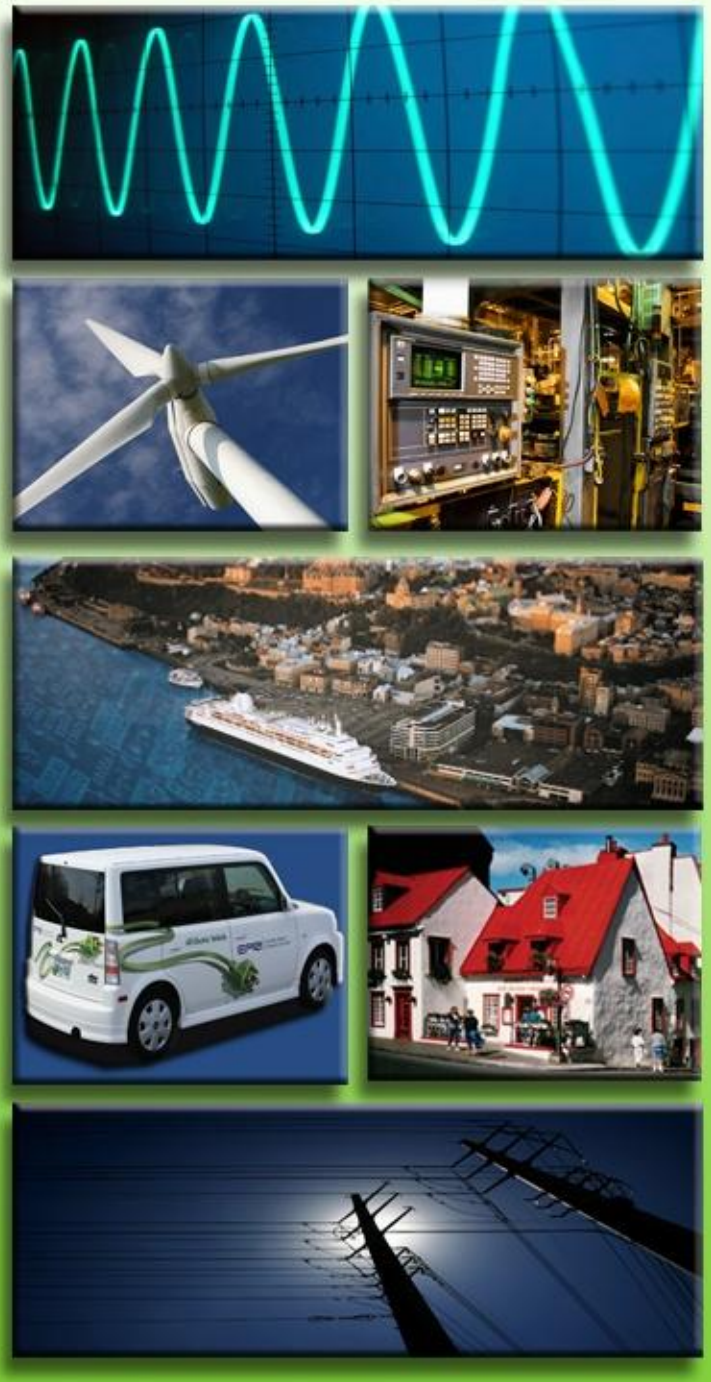
- In 2010-11, much work in solar PV generation modeling.
  - The *PVSystem* model was added recently
  - Enhancements are expected to result from the US DOE High-penetration solar PV study and the EPRI Distributed PV study.
- Storage models were developed in 2009-2010 for the EPRI Smart Grid demos and will continue through 2012
  - Dynamics model development underway
- Several projects underway to develop more mature dynamics models.
  - (much opportunity for participation from partners)
- There has been much interest in a more friendly graphical user interface for users who do not wish to learn either the scripting or programming interface.
  - A few users have already developed their own
  - Multiple proposals are under consideration.

# Plan for Future Work/Enhancements

- There is a continuing effort to promote interfacing to the **Common Information Model (CIM).**
  - The CIM is the common data interchange mechanism
- We plan to exploit **high performance desktop computing** to make it feasible to study distribution systems of at least 100,000 nodes and perhaps as many as 1,000,000 nodes.
  - This will involve revamping algorithms within the program to enable the application of parallel processing (multiple CPUs) to planning problems.
- Improved distribution state estimation functions
- Improved Distribution Management Systems (DMS) functions
- Reorganization of the source code
  - to better support platforms other than Windows and
  - to permit operation on 64-bit platforms.

# Why Open Source?

- EPRI has made the OpenDSS open source to:
  - Cooperate with various Smart Grid open source efforts
    - Gridlab-D (from PNL), for example
  - To encourage new advancements in distribution system analysis
  - To promote grid modernization/Smart Grid efforts by providing researchers and consultants with a tool to evaluate advanced concepts
  - Expand the pool of Smart Grid technology resources available to EPRI members



EPRI ELECTRIC POWER RESEARCH INSTITUTE

# Introduction to OpenDSS

## Uses and Applications

Some examples to inspire you…

# What can OpenDSS be used for?

- Simple power flow (unbalanced, n-phase)
- Daily loading simulations
- Yearly loading simulations
- Duty cycle simulations
  - Impulse loads
    - Rock crushers
    - Car crushers
  - Renewable generation
- DG
  - Interconnection studies/screening
  - Value of service studies (risk based)
  - Solar PV voltage rise/fluctuation
  - Wind power variations impact
  - Hi-penetration solar PV impacts
  - Harmonic distortion
  - Dynamics/islanding

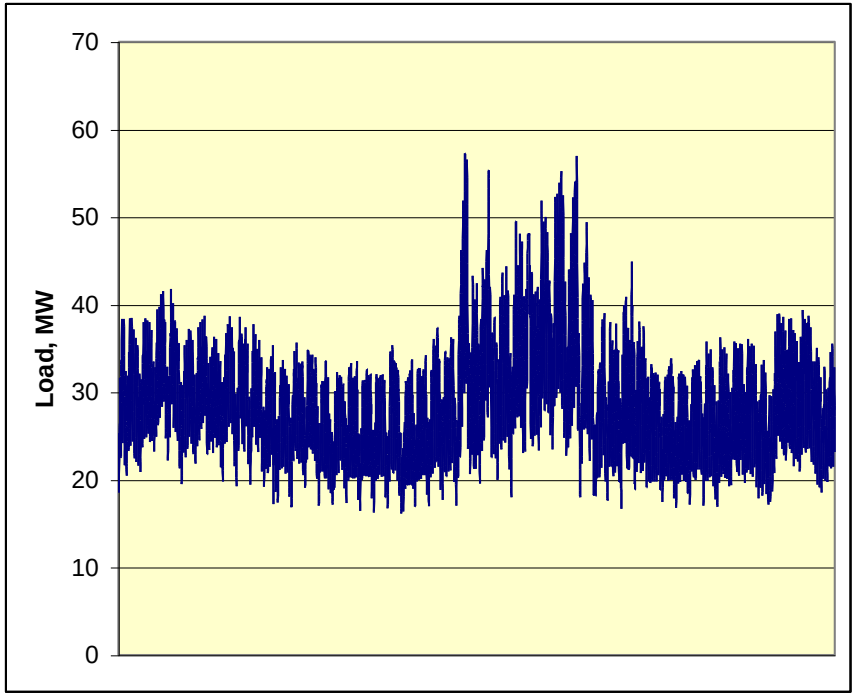
# What can OpenDSS be used for? (cont'd)

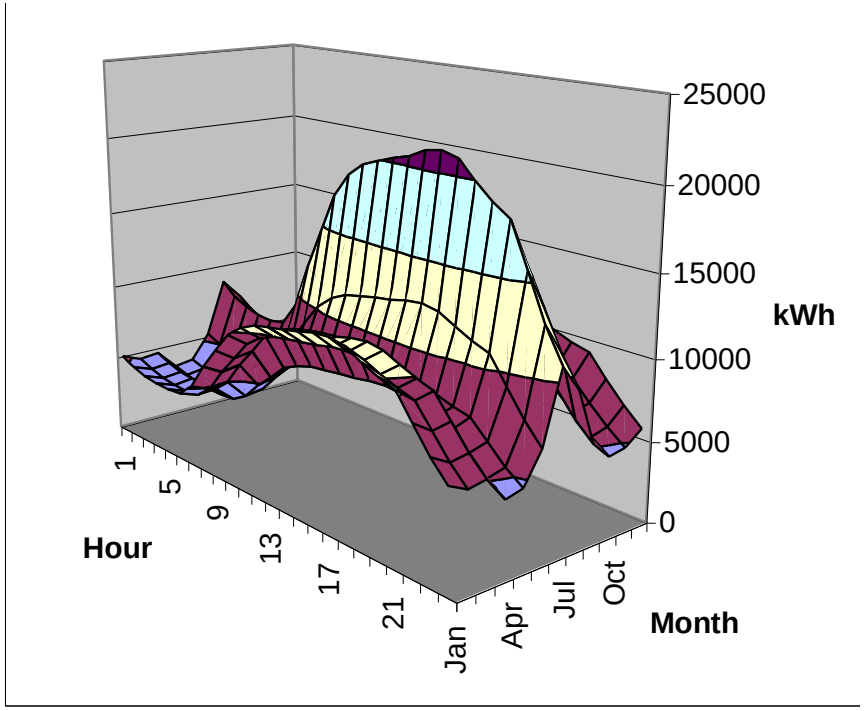
- Hybrid simulation of communications and power networks
- Geomagnetically-Induced Current (GIC) flow (solar storms)
- Power delivery loss evaluations (EPRI Green circuits program - > 80 feeders)
- Voltage optimization
- PEV/PHEV impact simulations
- Community energy storage (EPRI Smart Grid Demo)
- High-frequency harmonic/interharmonic interference
- Various unusual transformer configurations
- Transformer frequency response analysis
- Distribution automation control algorithm assessment
- Impact of tankless electric water heaters
- Wind farm collector simulations
- Wind farm interaction with transmission
- Wind generation impact on capacitor switching and regulator/LTC tapchanger operations
- Protection system simulation
- Open-conductor fault conditions
- Circulating currents on transmission skywires
- Ground voltage rise during faults on lines
- Stray voltage simulations
- Industrial load harmonics studies/filter design
- Distribution feeder harmonics analysis, triplen harmonic filter design

- And many more ….

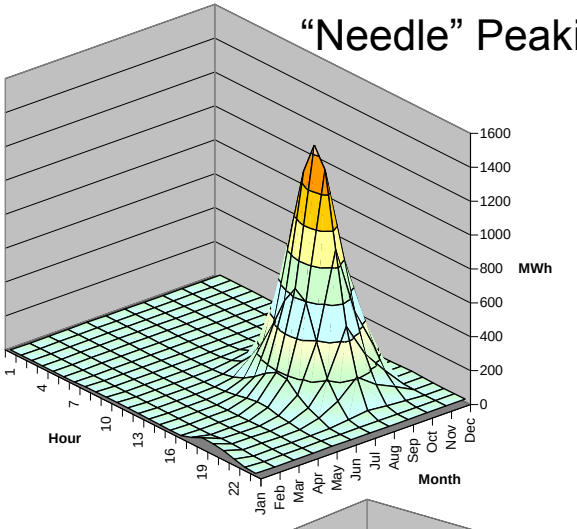
# Computing Annual Losses

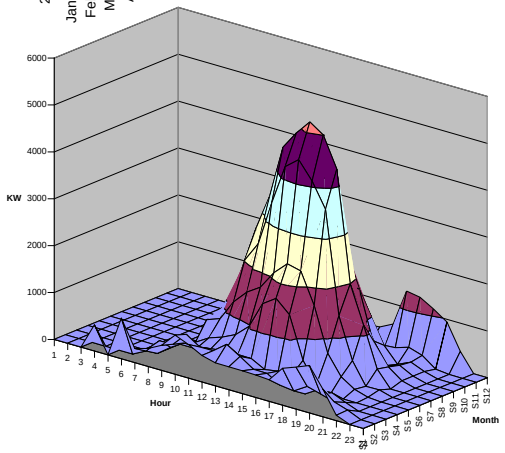
Peak load losses are not necessarily indicative of annual losses



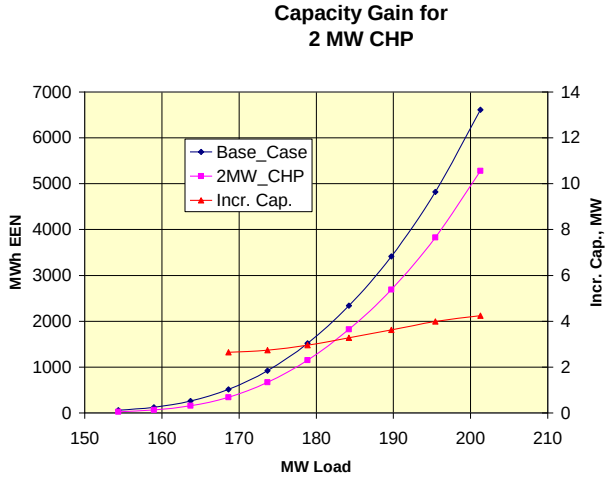



# Using DSS to Determine Incremental Capacity of DG



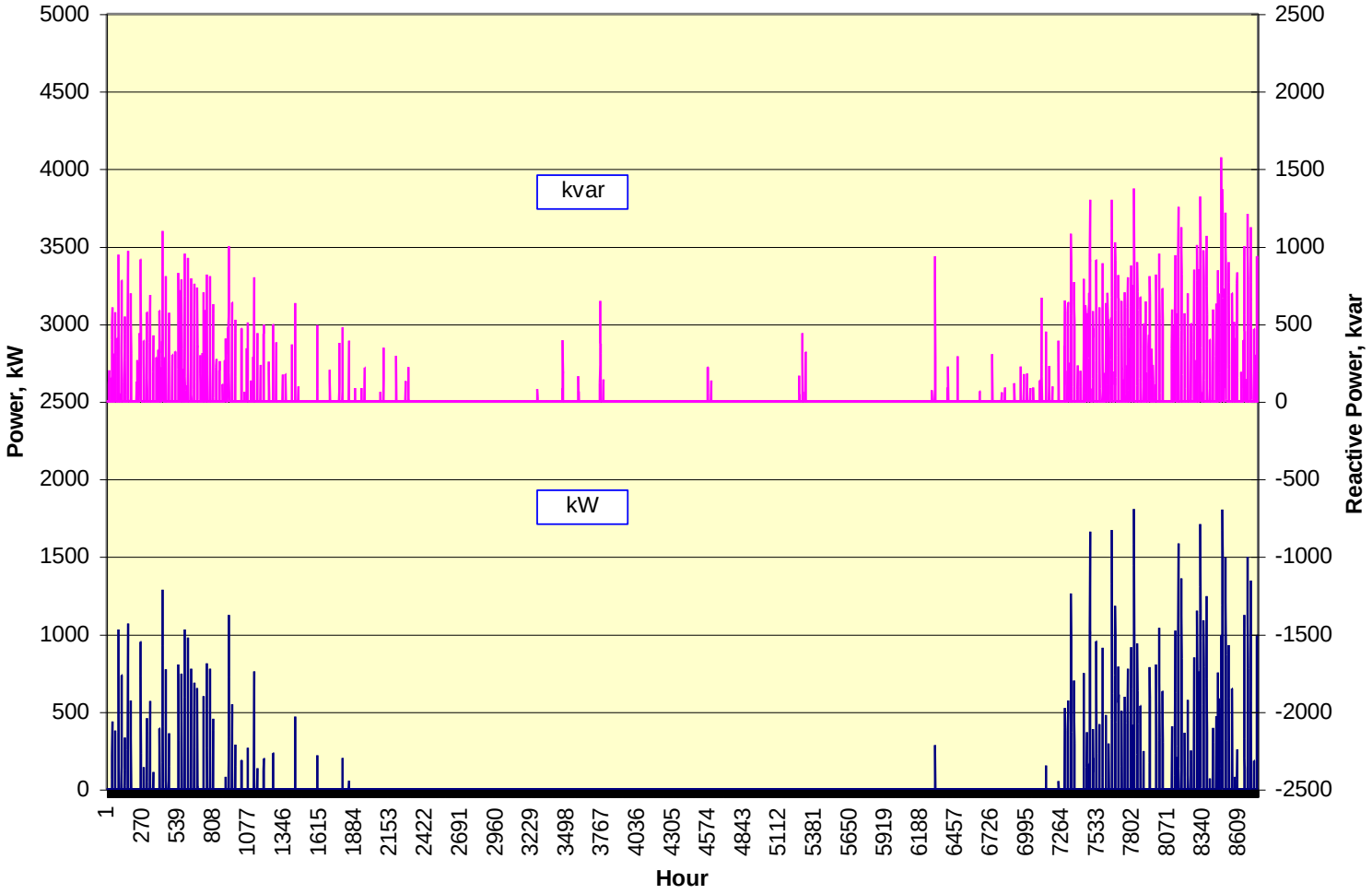




Broad Summer Peaking System

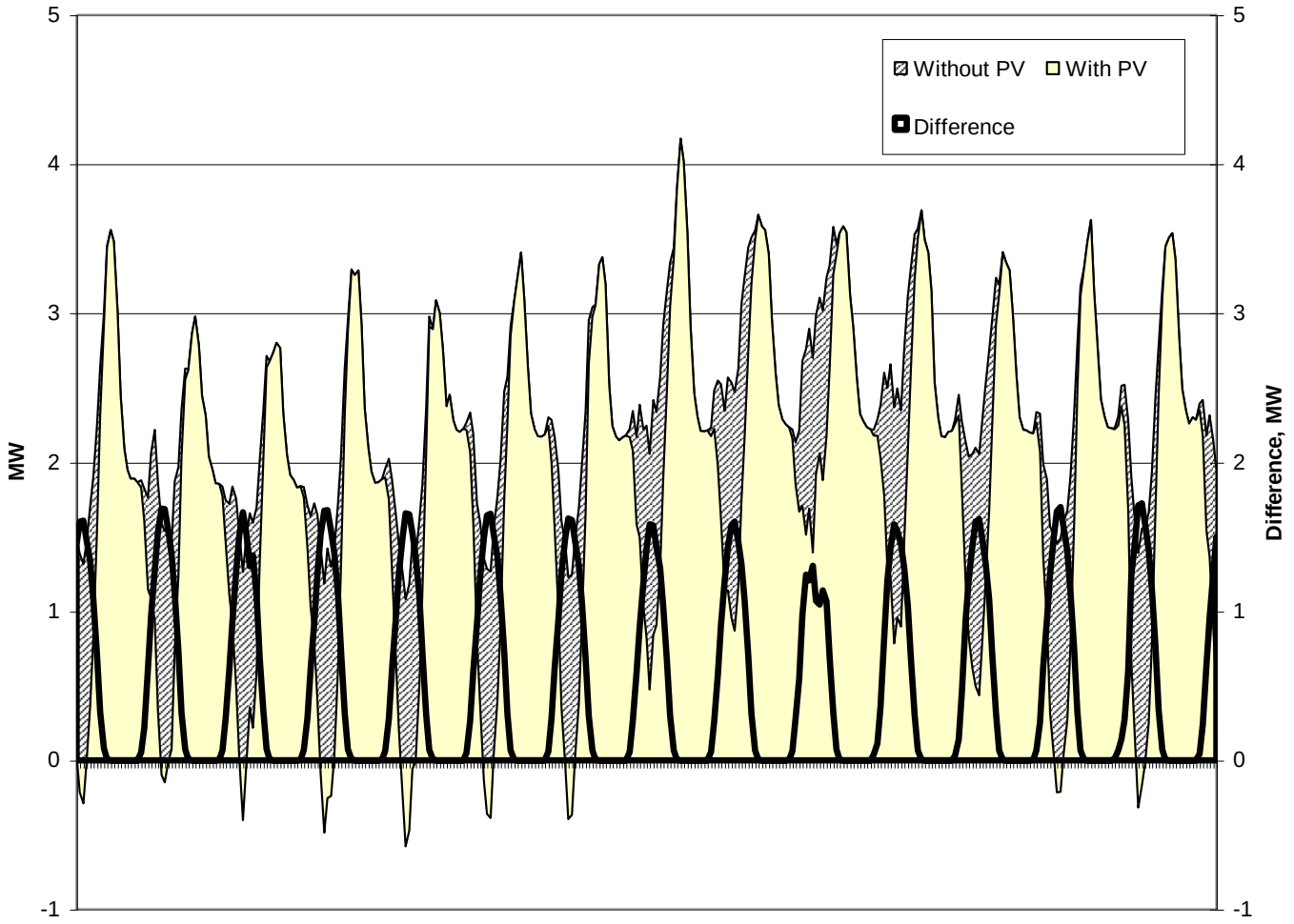



"How much more power can be served at the same risk of unserved energy?"

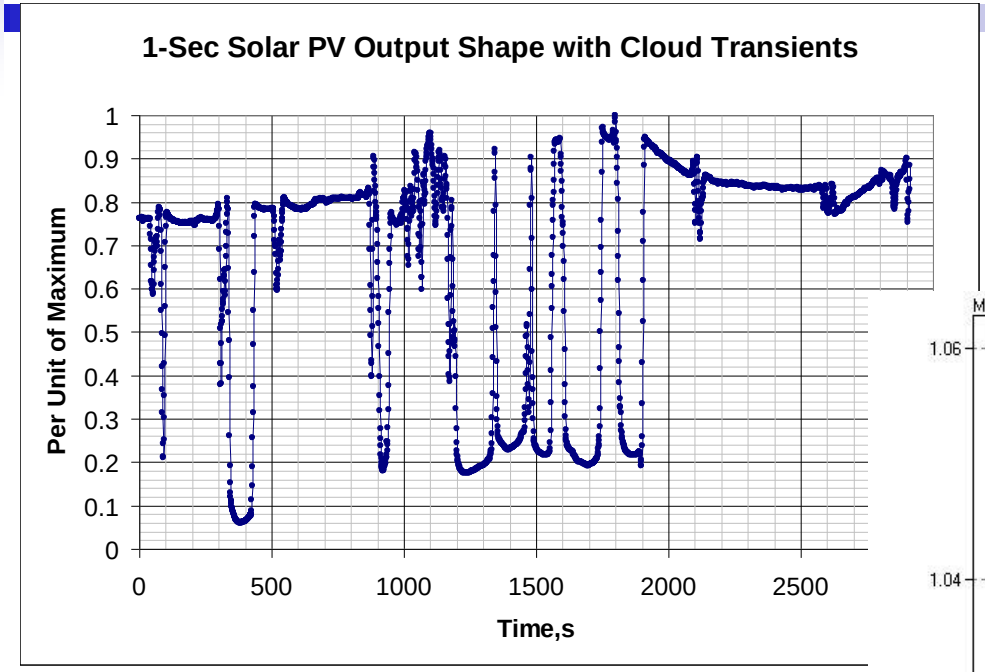
# DG Dispatch

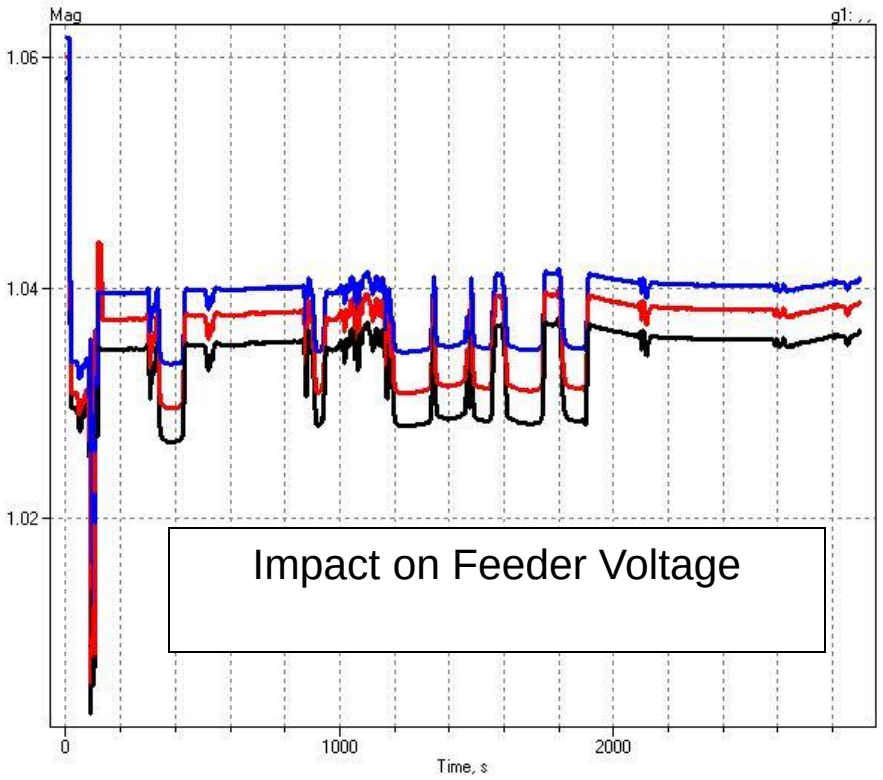


# Solar PV Simulation

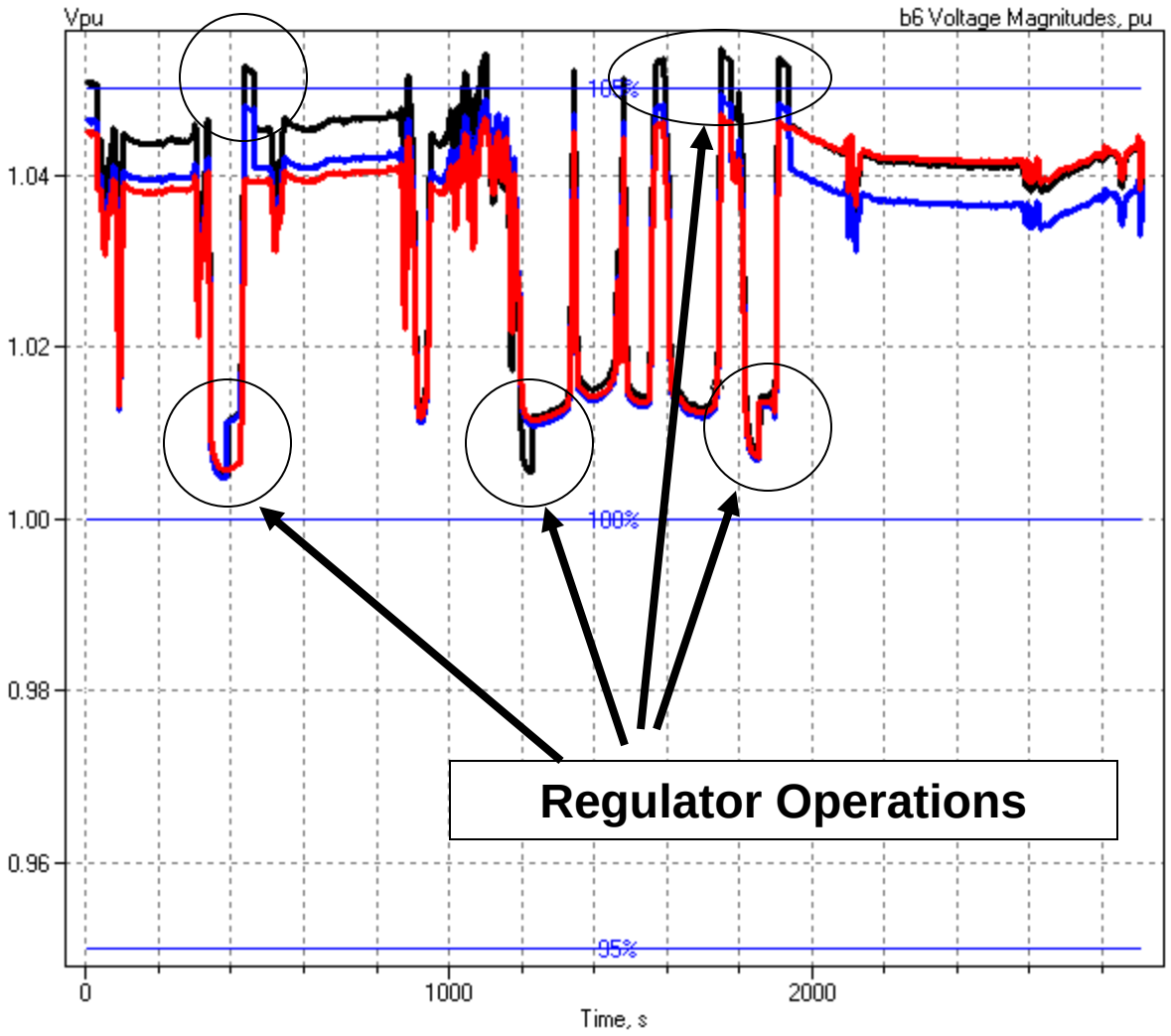


# 1-sec Solar Data – Cloud Transients

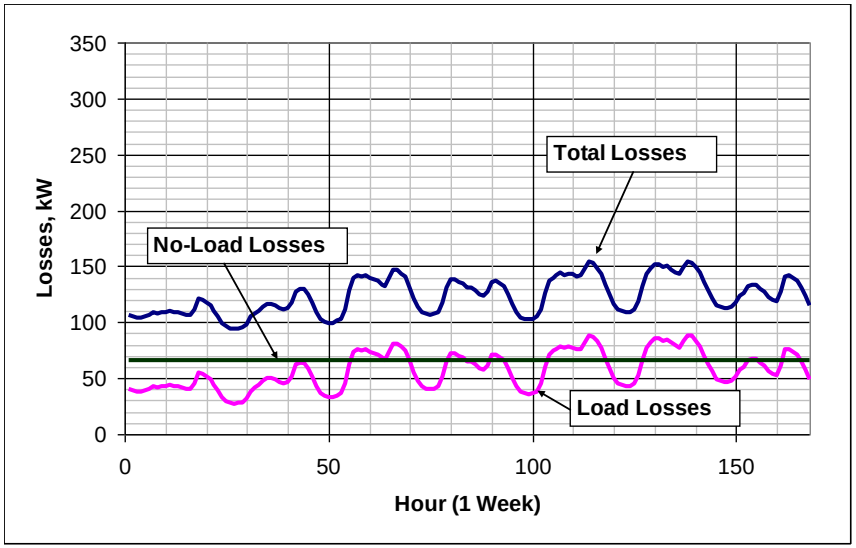


# 1-sec Solar Data (2010)

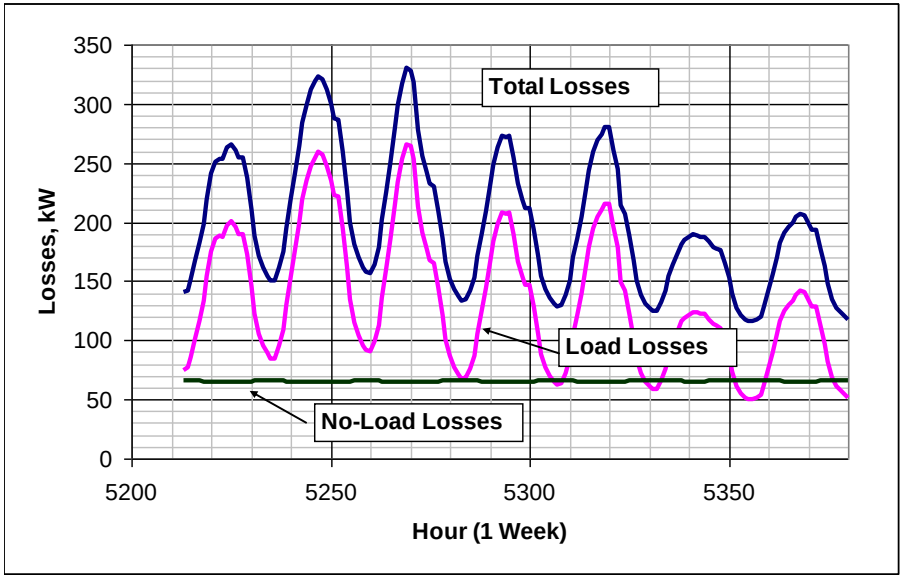


# Power Distribution Efficiency

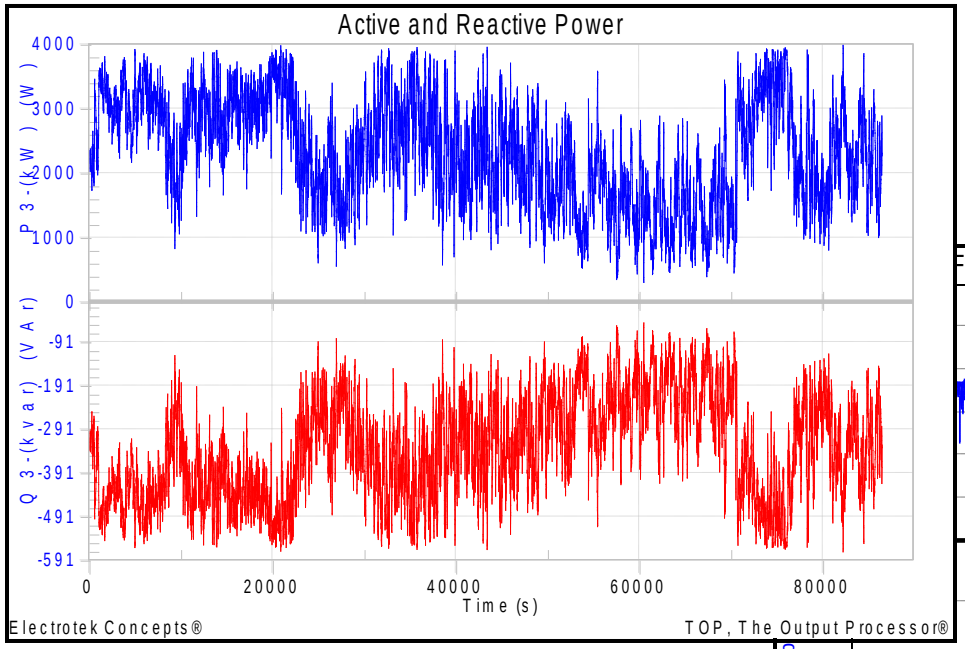


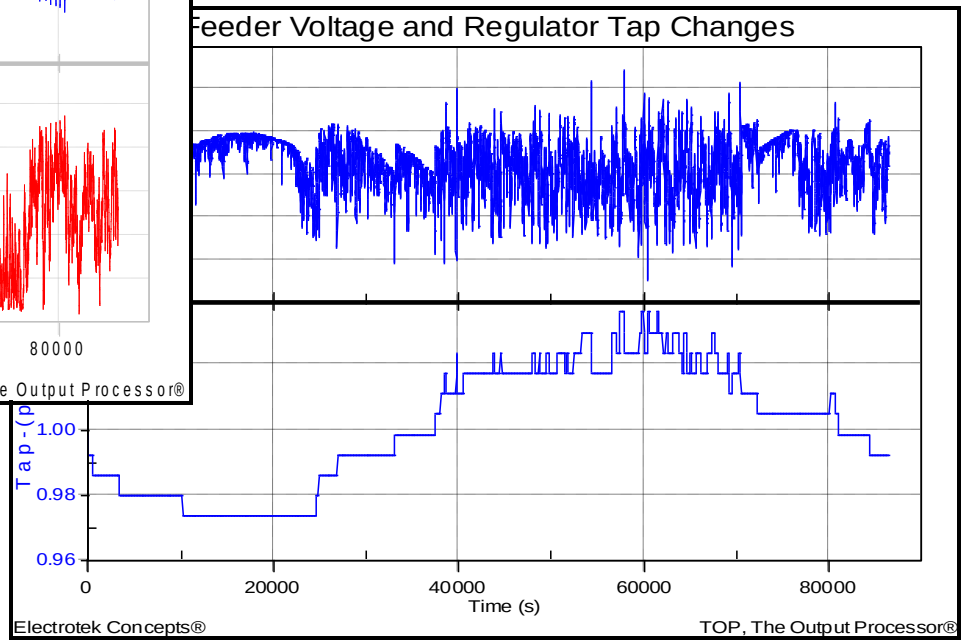

Light Load Week

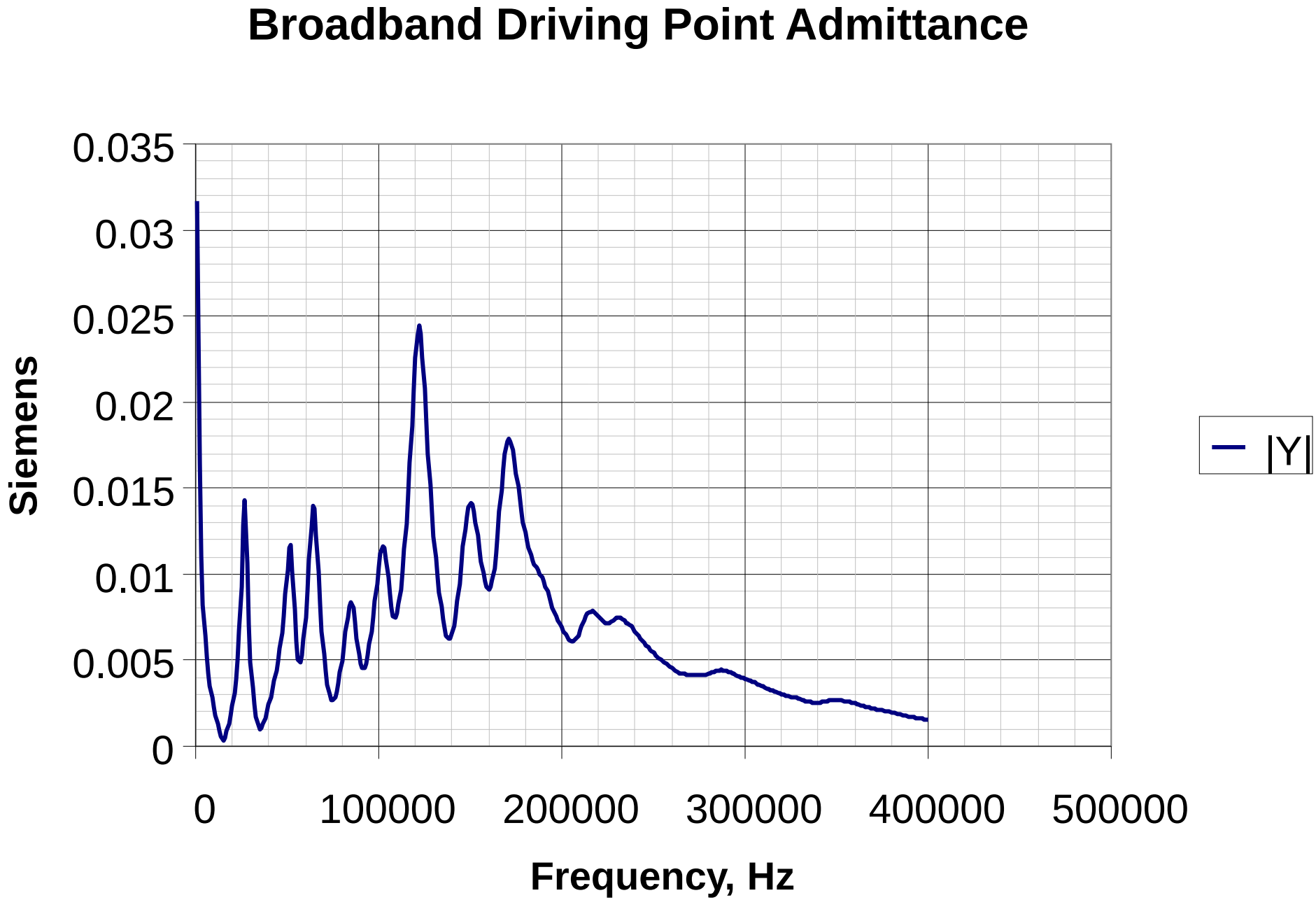
Peak Load Week



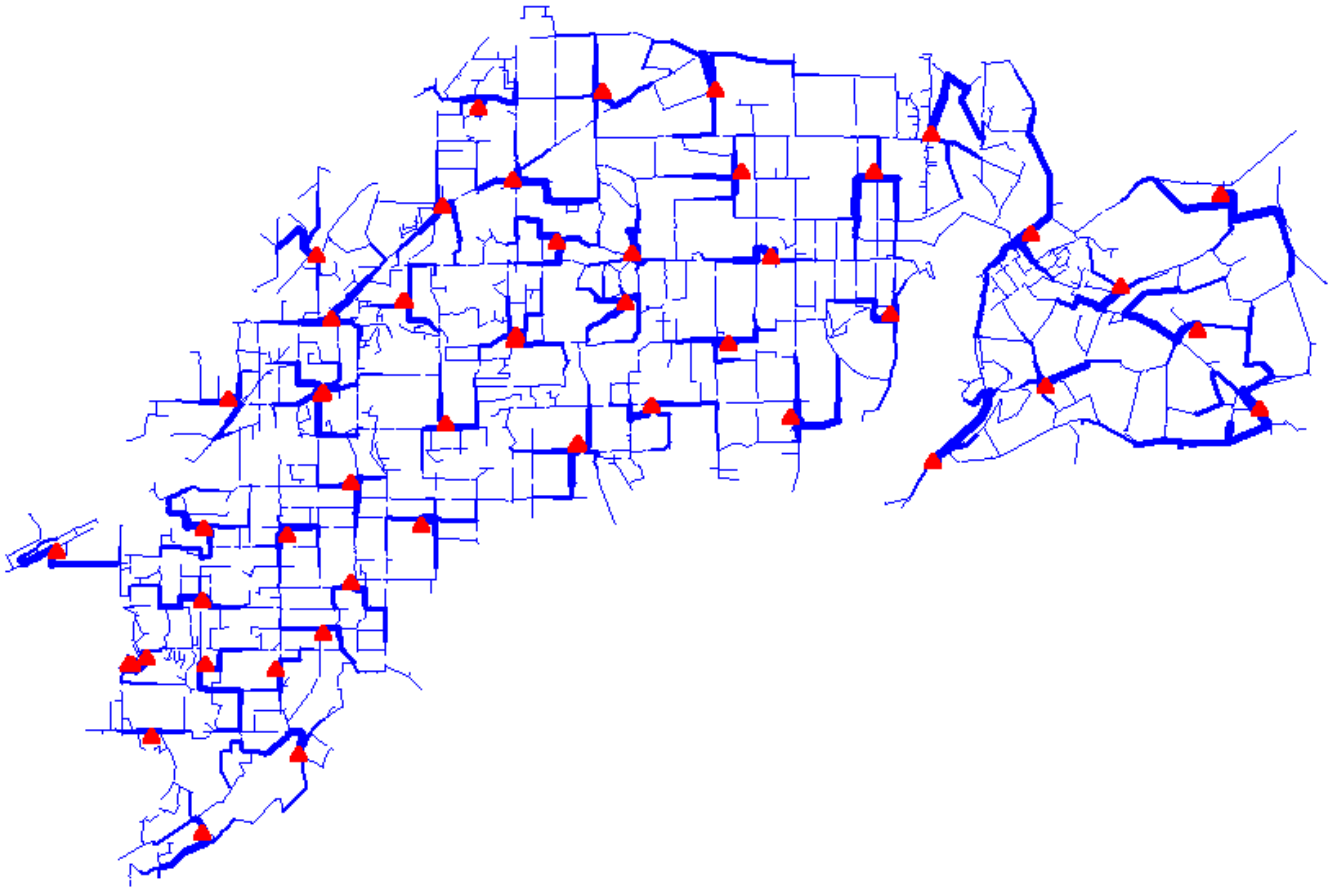
# Wind Plant 1-s Simulation



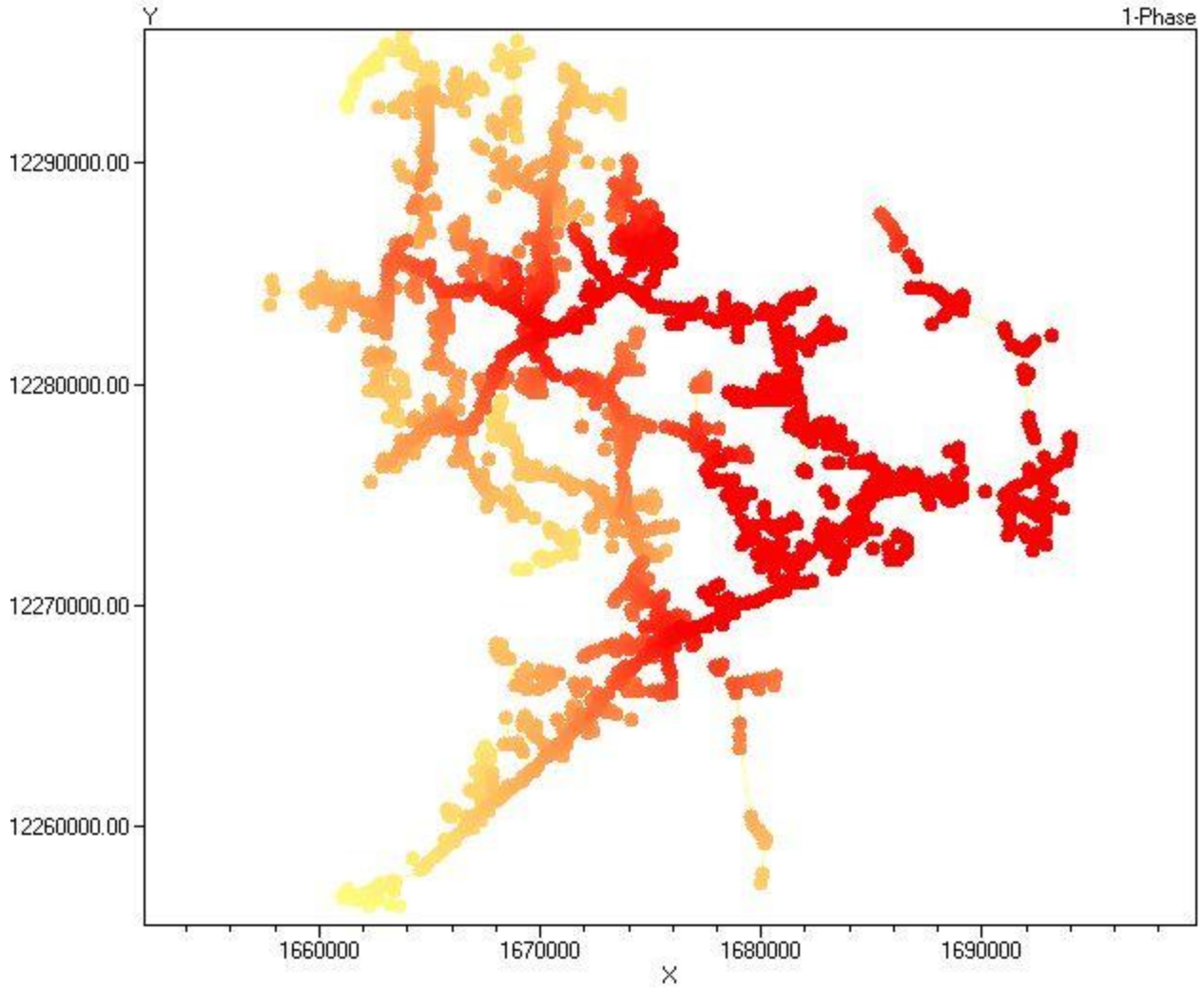




Broadband Driving Point Admittance
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
Siemens
0
100000
200000
300000
400000
500000
Frequency, Hz
|Y|

# Power Flow Visualization



# Special Displays

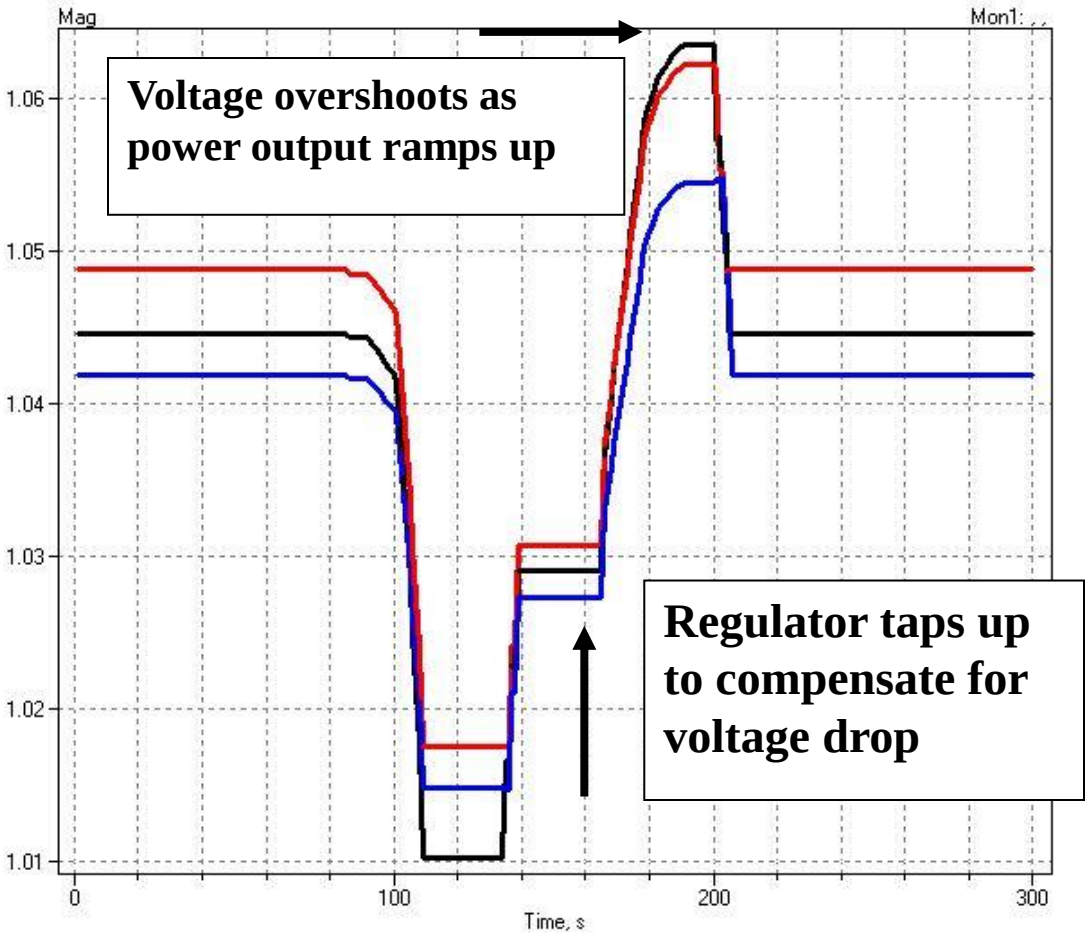



Fault Current Magnitudes

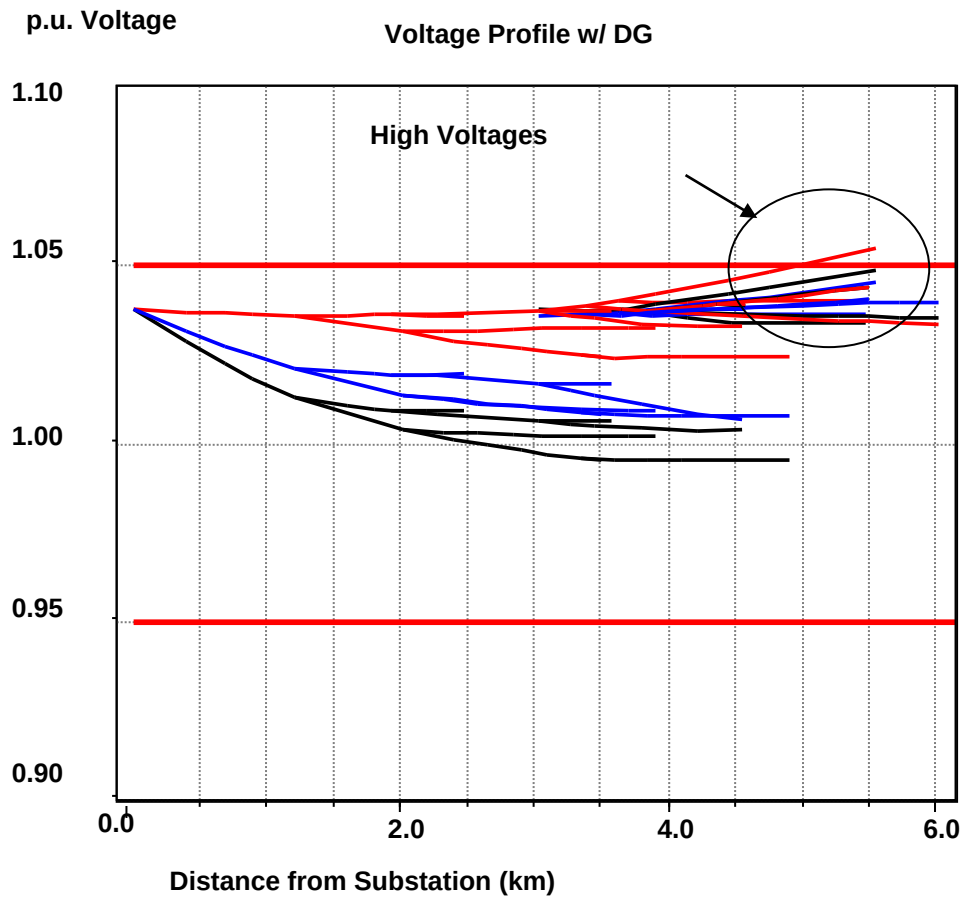
# Needs Envisioned by EPRI

- Sequential time simulation
- Meshed network solution capability
- Better modeling of Smart Grid controllers
- Advanced load and generation modeling
- High phase order modeling ( >3 phases)
  - Stray voltage (NEV), crowded ROWs, etc.
- Integrated harmonics
  - NEV requires 1st and 3rd
- User-defined (scriptable) behavior
- Dynamics for DG evaluations

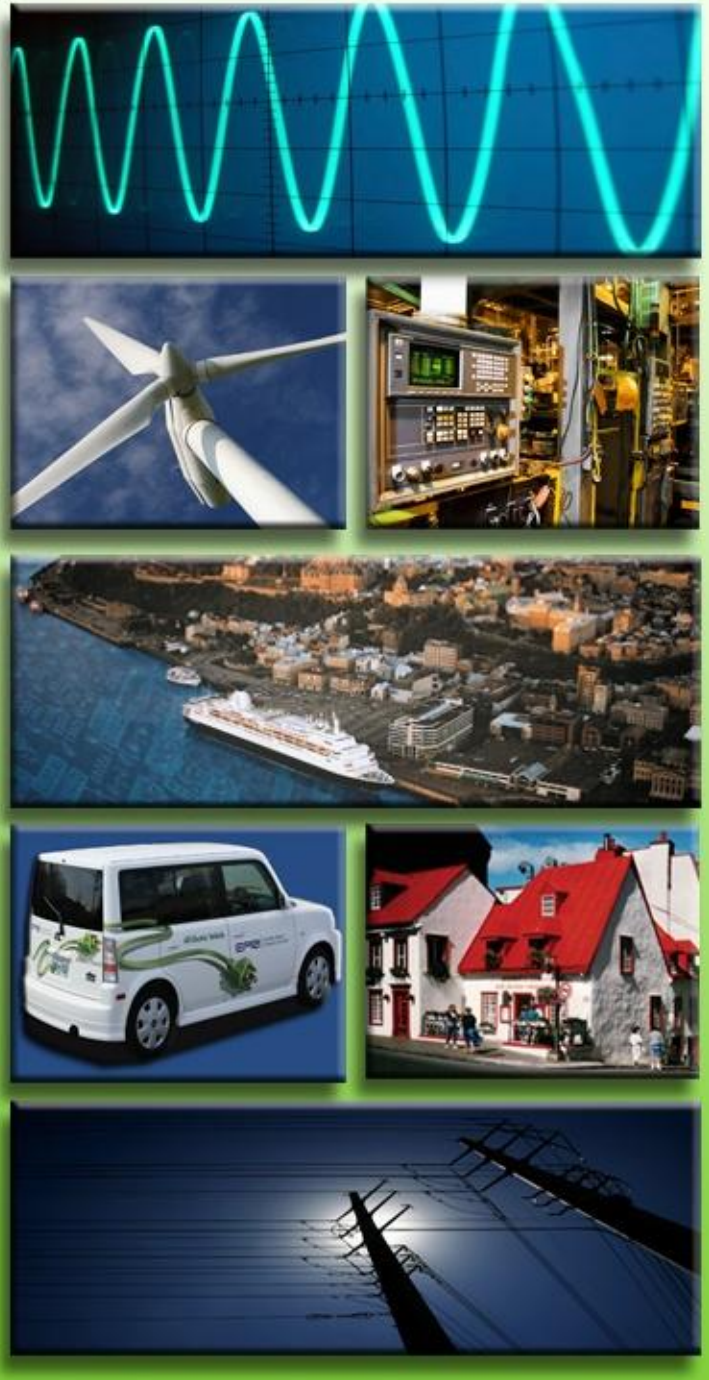
# Example of an Expected DG Problem



# Root of Problem




**Distribution Systems designed for voltage DROP, not voltage RISE.**



EPRI | ELECTRIC POWER RESEARCH INSTITUTE

# Getting Started:

# Installation and Basic Usage

# SourceForge.Net Links for OpenDSS

- OpenDSS Download Files:
  - **http://sourceforge.net/projects/electricdss/files/**
- Main Page in Wiki
  - **http://sourceforge.net/apps/mediawiki/electricdss/index.php?title=Main_Page**
- Top level of Main  Repository
  - **http://electricdss.svn.sourceforge.net/viewvc/electricdss/**
- Source Code
  - **http://electricdss.svn.sourceforge.net/viewvc/electricdss/Source/**
- Examples
  - **http://electricdss.svn.sourceforge.net/viewvc/electricdss/Distrib/Examples/**
- IEEE Test Cases
  - **http://electricdss.svn.sourceforge.net/viewvc/electricdss/IEEETestCases/**

# Wiki Home Page

page | discussion | view source | history

## Main Page

The **Distribution System Simulator (DSS)** is a comprehensive electrical system simulation tool for electric utility distribution systems. The **OpenDSS** is being provided as an open source program to the electric power system analysis community at large by the Electric Power Research Institute ([[1]]) under a BSD license to cooperate with other entities involved in the Smart Grid, or grid modernization, efforts.

The OpenDSS is implemented as both a standalone EXE program and as a COM DLL. The DLL is designed as an in-process server to be driven from a variety of existing software platforms for highly customized types of distribution system analysis. The EXE version provides a mulitple-window user interface to assist users in constructing and executing scripts. The DSS basically supports all rms steady-state (frequency domain) analyses commonly performed on electric power distribution systems, such as power flow, harmonic analysis and fault current calculations. In addition, it supports many new types of analyses that are designed to meet future needs, many of which are being dictated by the deregulation of US utilities and the formation of distribution companies worldwide. Many of the features were originally driven by distributed generation analysis needs. More recently, features have been added to enhance the study of energy efficiency, stray voltages, and distribution state estimation. The DSS is designed to be indefinitely expandable so that it can be more easily modified to meet future needs (see the Indmach012 model for an example of this expandability).

Through the COM interface, the user is capable of performing all the functions of the simulator, including definition of the model data. Thus, the DSS is entirely independent of any database or text file circuit definition. It can be driven entirely from a MS Office tool through VBA, for example, or from any other 3rd party analysis program (e.g., Matlab Interface) that can handle COM. One way to think of the DSS is as an object-oriented database of power system circuit data that can perform various common distribution system analysis tasks. The COM interface contains a text-based command interface as well as numerous COM interface methods and properties for accessing many of the parameters and functions of the simulator's models. Through the command line interface, users can prepare scripts to do several functions in sequence. The input may be redirected to a text file to accomplish the same effect as macros and also provide some database-like characteristics.

### News and Notes

The Free Pascal Compiler (FPC) command-line version has been fixed and updated on Sourceforge. Oct 14, 2010 [2] Should update automatically if you are using SVN.

Storage and Storage Controller Documentation Update Oct 25, 2010 [3]

Don't have a compiler to build the source to keep up with the latest changes? Get a zip file with the latest build (pre-release) from Roger Dugan's mirror site: [4] This is usually updated within minutes of posting the latest source code.

Did you know there were some typical Loadshapes available for use in studies? [5]

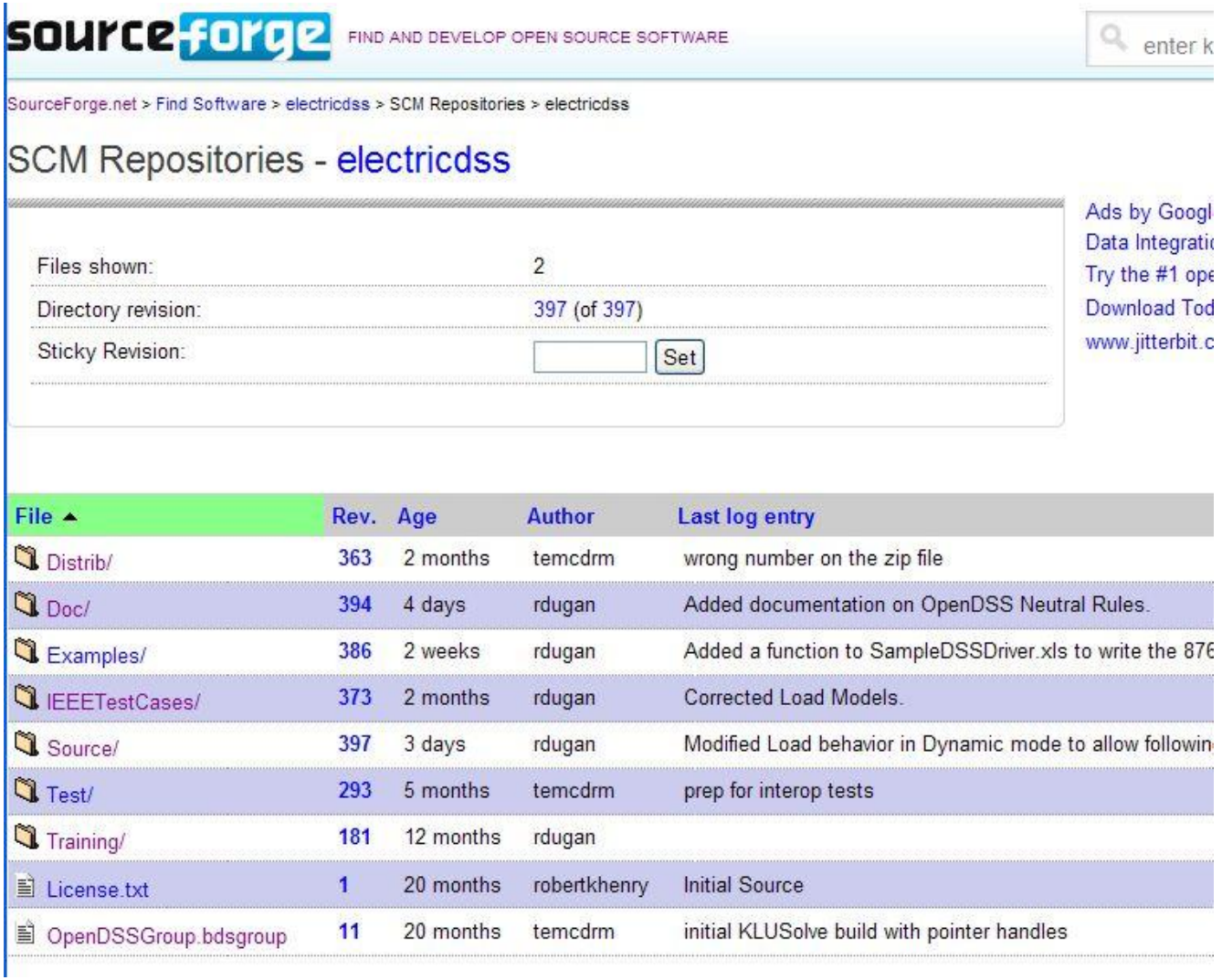
### See also

What is unique about OpenDSS?

Tech Notes

Questions and Answers (FAQ)

# Repository on SourceForge.Net



# Release Versions Vs. Source Code

- Release versions are posted irregularly on SourceForge
- You can keep up with the latest changes by accessing the source code and building the latest version

- OR -

- Latest builds are also posted on various sites
  - (see Wiki News and Notes)
  - EPRI Program 180 Collaboration site

- Compilers
  - Delphi 2010 - This is what we use for development
    - Delphi 2007 and 2009 also worked last time we tried
  - Free Pascal (with limitations) -*www.**freepascal**.org*

# Accessing the SourceForge.Net Source Code Repository with TortoiseSVN

- Install a TortoiseSVN client from Tortoisesvn.net/downloads.
- Recommendation:
  - From the TortoiseSVN General Settings dialog and click the last check box, to use "_svn" instead of ".svn" for local working directory name.

    Then, to grab the files from SourceForge by:

    1 - create a clean directory such as "c:\opendss"

    2 - **right-click** on it and choose "SVN Checkout..." from the menu

    3 - the repository URL is

    **http://electricdss.svn.sourceforge.net/svnroot/electricdss**

    (Change the checkout directory if it points somewhere other than what you want.)

EPRI ELECTRIC POWER RESEARCH INSTITUTE

# Program Installation

# Program Files (as of November 2011)

- OpenDSS.EXE — Standalone EXE
- **OpenDSSEngine.DLL — In-process COM server**
- KLUSolve.DLL — Sparse matrix solver
- DSSgraph.DLL — DSS graphics output

- Copy these files to the directory (folder) of your choice
  - Typically `c:\OpenDSS` *or* `c:\Program Files\OpenDSS`

# Registering the COM Server

- If you intend to drive OpenDSS from another program, you will need to register the COM server
  - Some programs require this !!
  - If you are sure you will only use OpenDSS.EXE, you can skip this step
    - You can come back and do it at any time

- In DOS, or command, window, change to the folder where you installed it and type:

**Regsvr32 OpenDSSEngine.DLL**

# Registering the COM Server – Windows 7

- Special Instructions for Window 7
  - (and probably Windows Vista, too)
  - See Q&A on the Wiki site
  - http://sourceforge.net/apps/mediawiki/electricdss/index.php?title=How_Do_I_Register_the_COM_Server_DLL_on_Windows_7%3F
- All Programs > Accessories
- right-click on the **Command Prompt** and select **Run as Administrator**.
- Then change to my OpenDSS folder and type in
  - "regsvr32 OpenDSSEngine.DLL" or
  - "RegisterDSSEngine", which runs the Bat file.

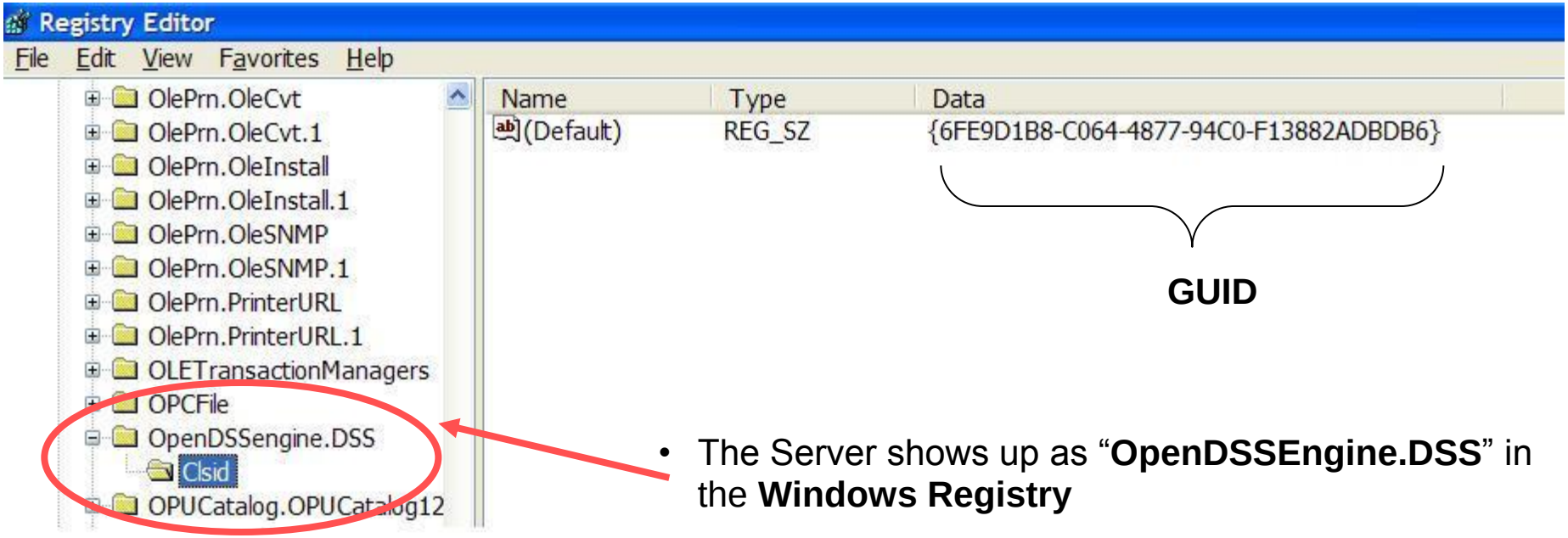
# Registering the COM Server – Windows 7

- Another Process for Windows 7
  - per Andy Keane- UC Dublin

  - 1. Right click on the desktop and select new -> shortcut
  - 2. For location of item just type ‘cmd’
  - 3. This will create a new shortcut to a command prompt on your desktop
  - 4. Right click on the new shortcut and select ‘Properties’
  - 5. On the shortcut tab select ‘Advanced’
  - 6. Tick the ‘Run as Administrator’ box
  - 7. Press Apply/OK and that command prompt will always be run as administrator allowing the user to use regsvr32

# So What Did this Do?

## Windows Registry Entry

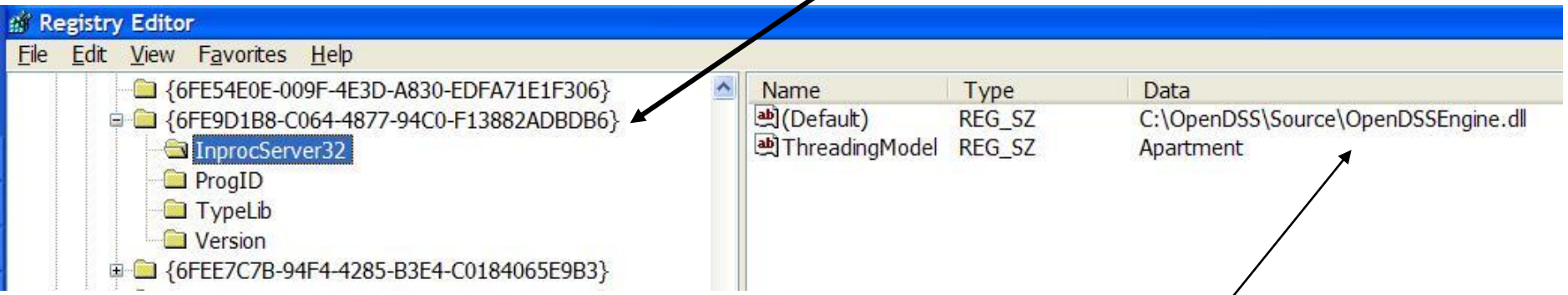



- The Server shows up as "**OpenDSSEngine.DSS**" in the **Windows Registry**

**The OpenDSS is now available to any program on the computer**

# The GUID References the DLL File ….

**If you look up the GUID in RegEdit**



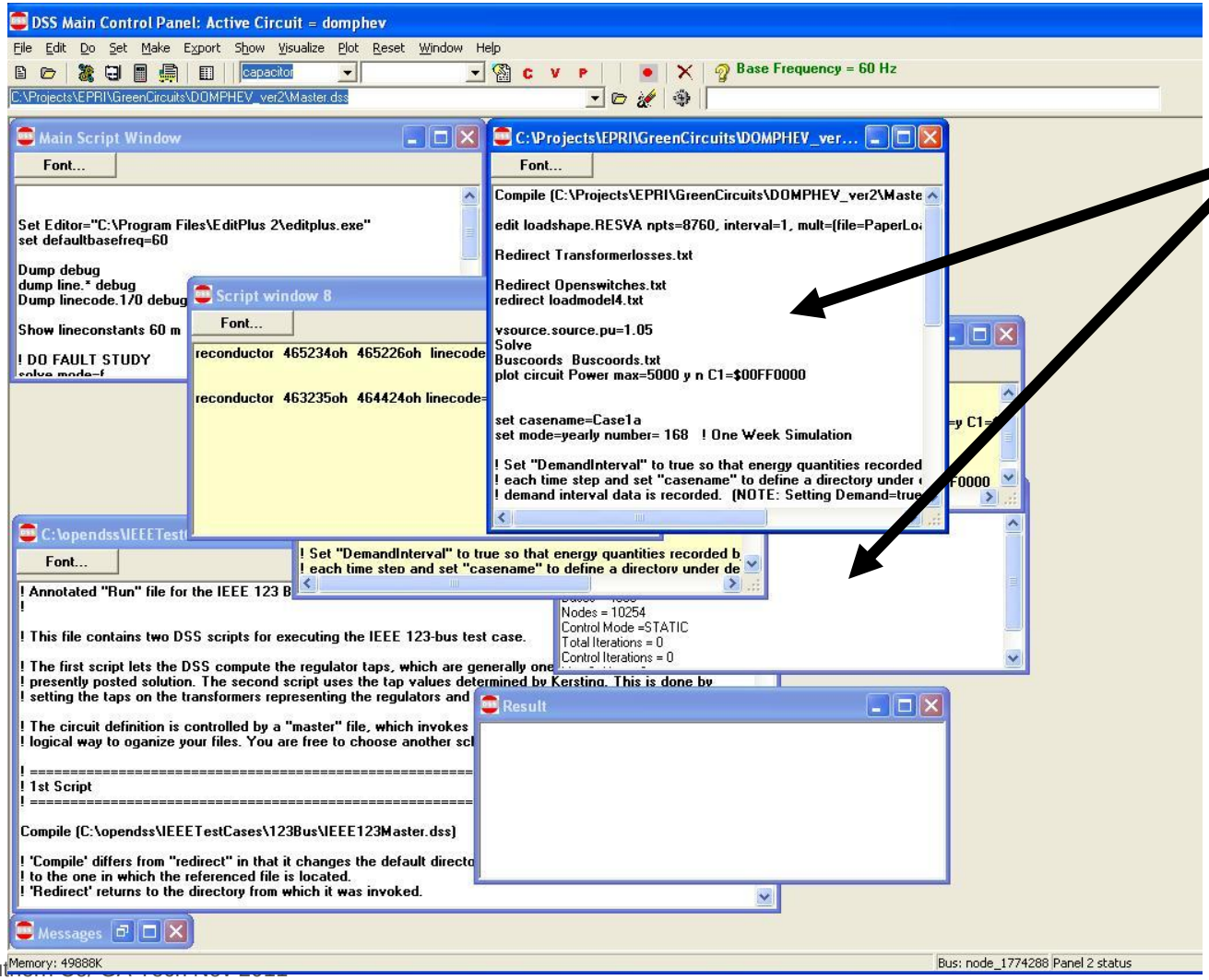

**Points to OpenDSSEngine.DLL**
**(In-process server, Apartment Threading model)**

# Accessing the COM Server

Examples of accessing the COM server in various languages

- In MATLAB:
  - `DSSobj = actxserver('OpenDSSEngine.DSS');`
- In VBA:
  - ```
    Public DSSobj As OpenDSSEngine.DSS
    Set DSSobj = New OpenDSSEngine.DSS
    ```
- In Dephi
  - `{Import Type Library}`
  - `DSSObj := coDSS.Create;`
- In PYTHON:
  - ```
    self.engine =
    win32com.client.Dispatch("OpenDSSEngine.DSS")
    ```

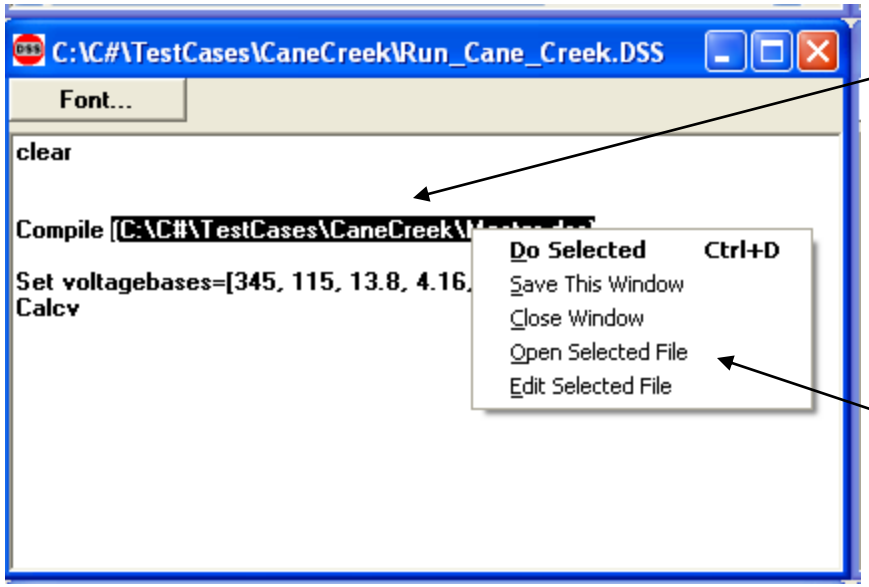
# OpenDSS Standalone EXE User Interface




Multiple script windows

Any script window may be used at any time.

# Executing Scripts in the Stand-alone EXE



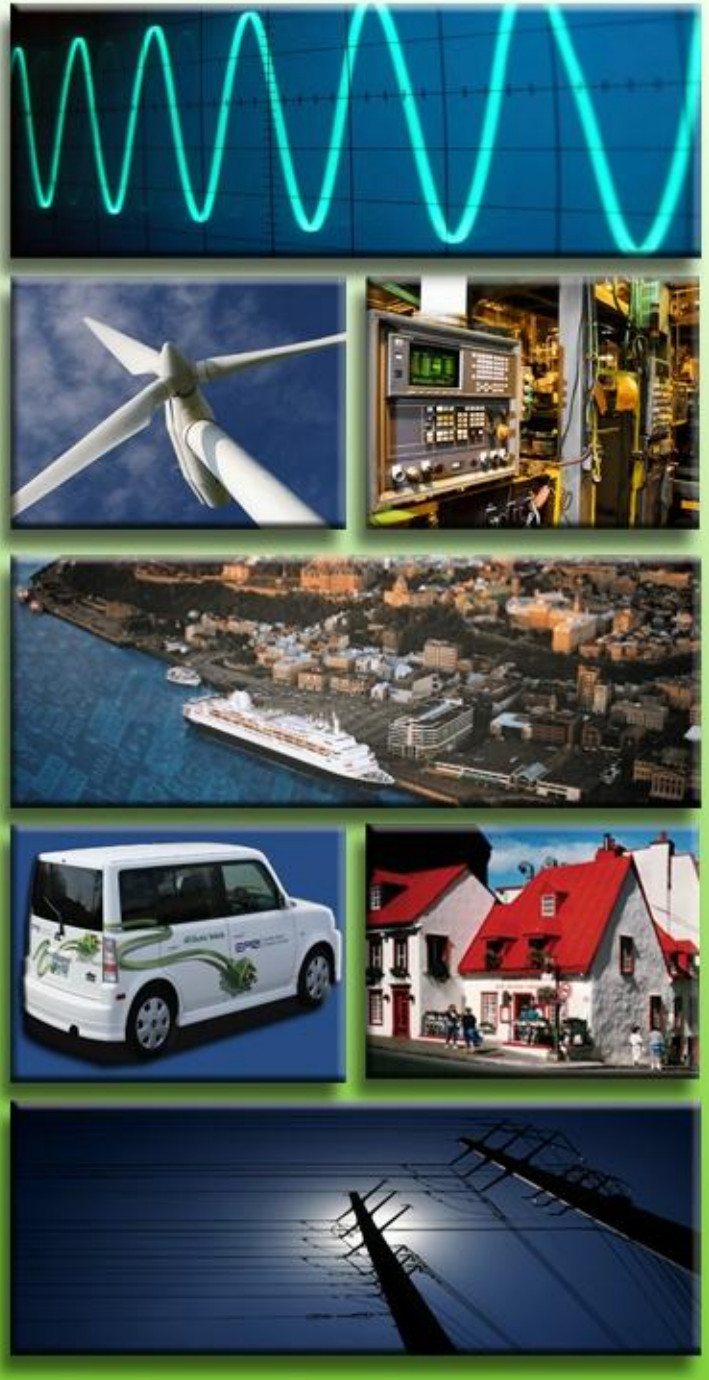

**Select all or part of a line**

**Right-Click to get this pop-up menu**

**DSS executes selected line or opens selected file name**

**Any script window may be used at any time.**

# Questions So Far?

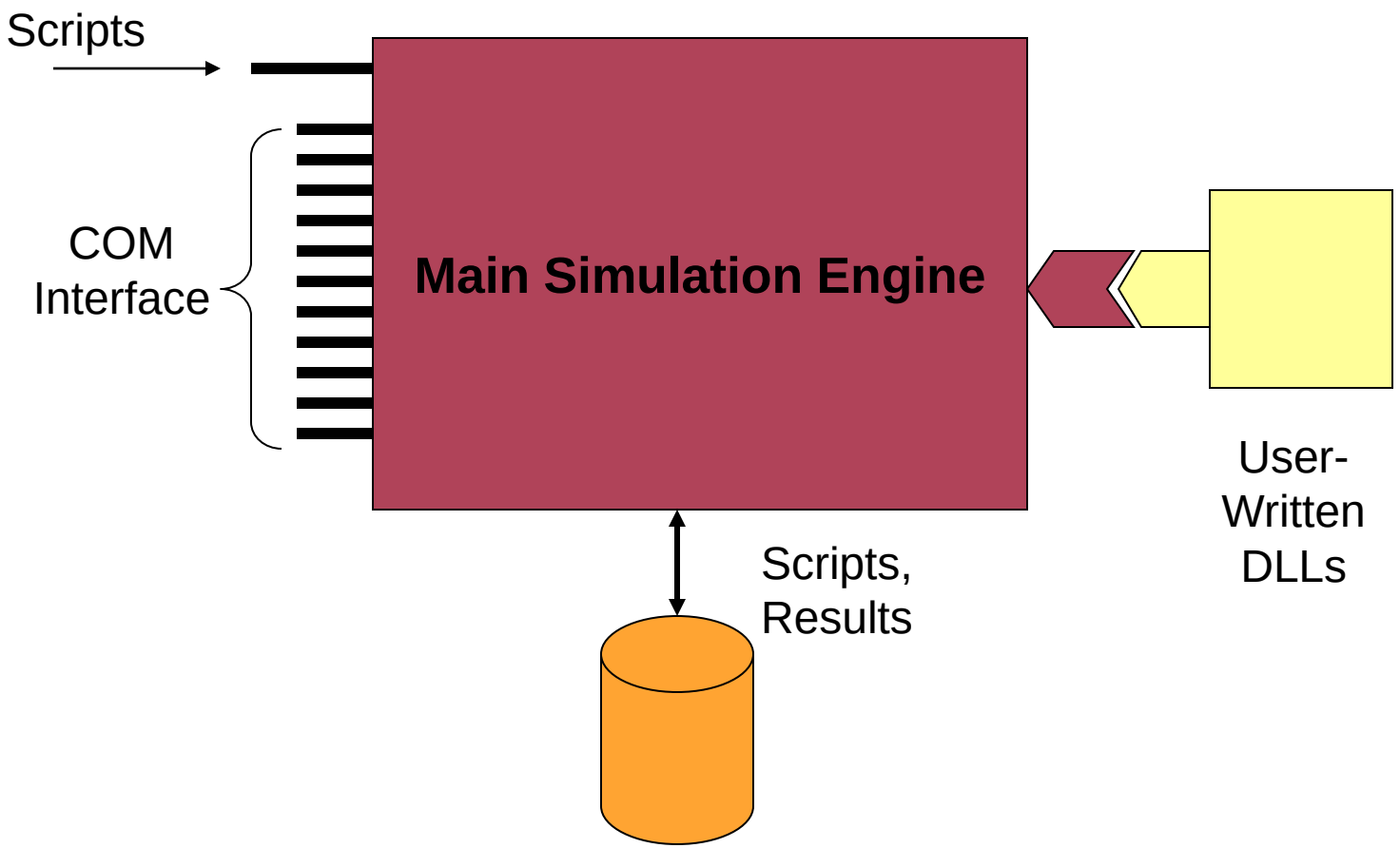


EPRI | ELECTRIC POWER RESEARCH INSTITUTE

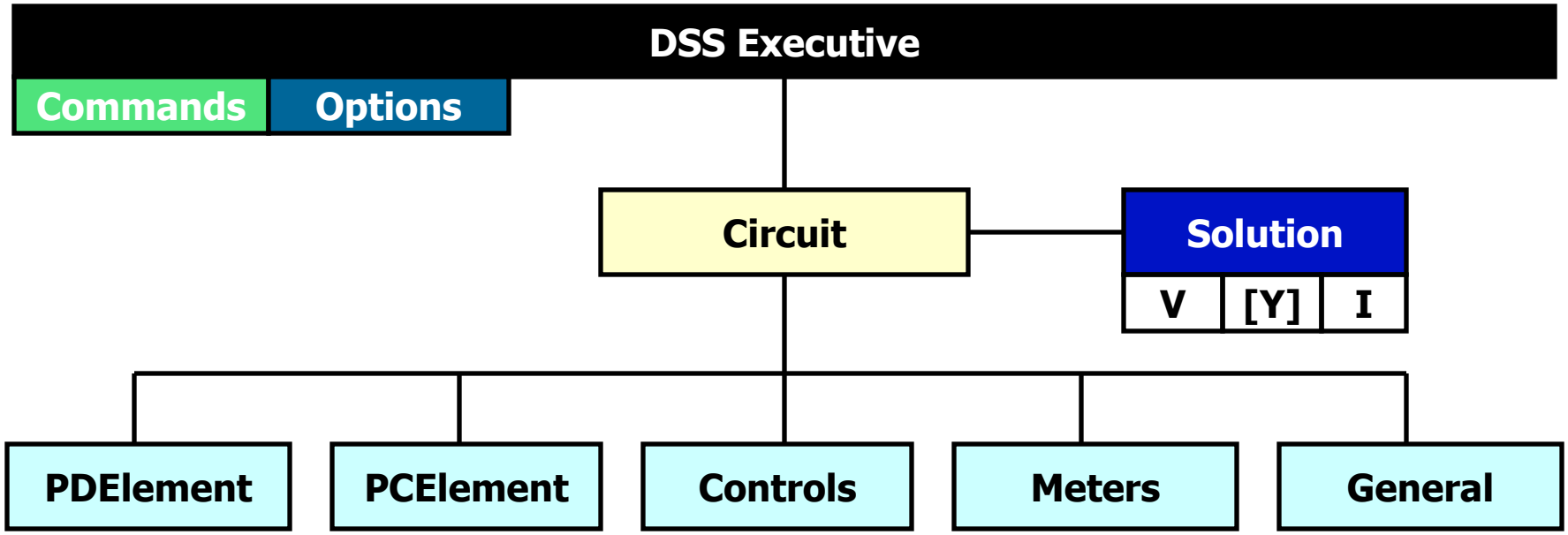
# OpenDSS Architecture and Circuit Modeling Basics

Description of basic circuit elements, bus models, etc.

# DSS Structure



# DSS Object Structure




**Line**
**Transformer**
**Capacitor**
**Reactor**

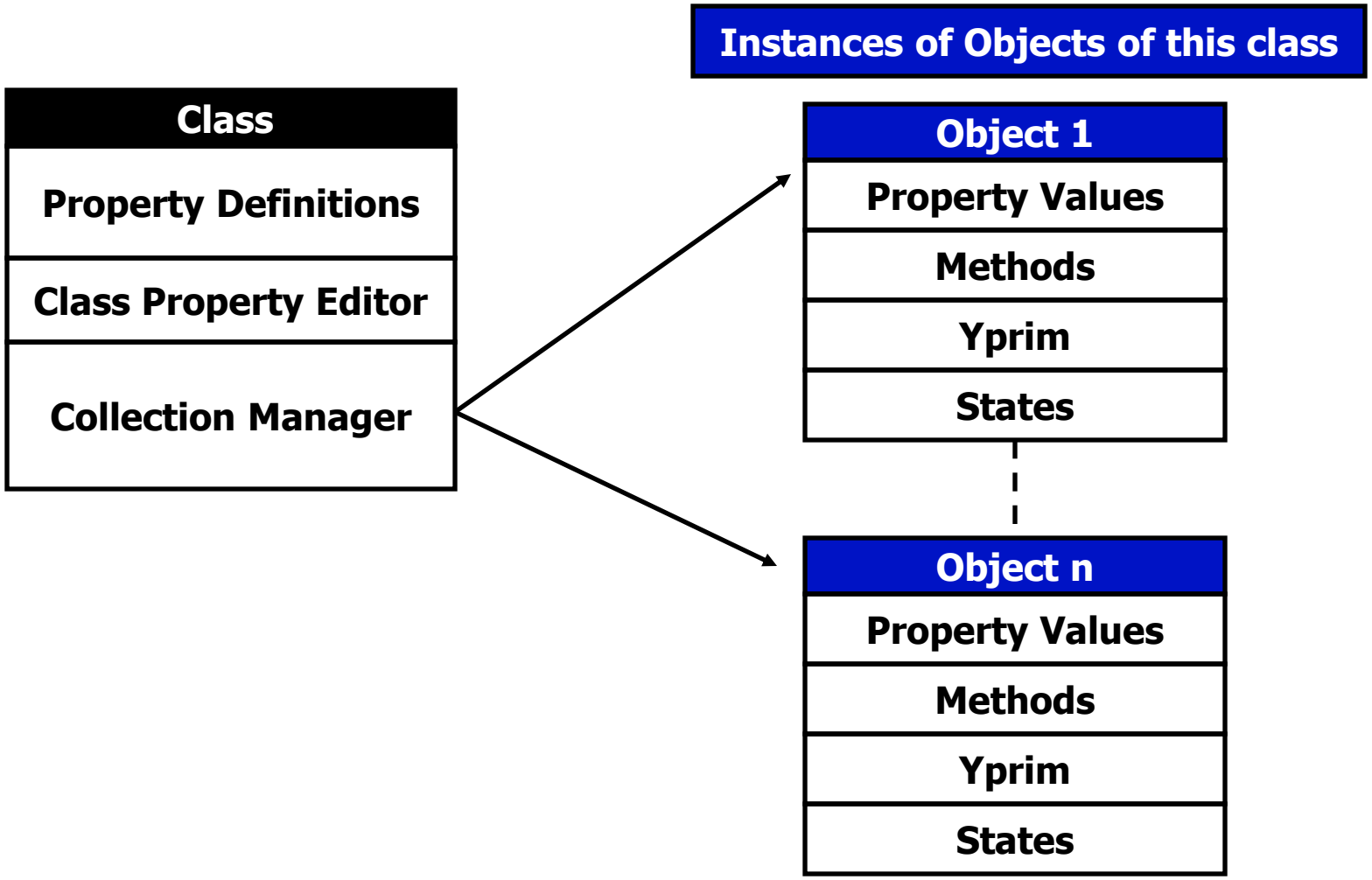
**Load**
**Generator**
**Vsource**
**Isource**
**Storage**

**RegControl**
**CapControl**
**Relay**
**Reclose**
**Fuse**

**Monitor**
**EnergyMeter**
**Sensor**

**LineCode**
**LineGeometry**
**WireData**
**LoadShape**
**GrowthShape**
**Spectrum**
**TCCcurve**
**XfmrCode**

# DSS Class Structure



# Models Currently Implemented (April 2011)

- POWER DELIVERY ELEMENTS
  - CAPACITOR (Series and shunt capacitors; filter banks)
  - LINE (All types of lines, cables)
  - REACTOR (Series and shunt reactors)
  - TRANSFORMER (multi-phase, multi-winding transformer models)
- POWER CONVERSION ELEMENTS
  - GENERATOR (General generator models)
  - LOAD (General load models)
  - PVSYSTEM (Solar PV system with panel and inverter)
  - STORAGE (Generic storage element models)
- CONTROL ELEMENTS
  - CAPCONTROL (Capacitor bank control; various types)
  - FUSE (Controls a switch, modeling fuse TCC behavior)
  - GENDISPATCHER (A specialized controller for dispatching DG)
  - RECLOSER (Controls a switch, modeling recloser behavior)
  - REGCONTROL (Standard 32-step regulator/LTC control)
  - RELAY (Controls a switch, modeling various relay behaviors)
  - STORAGECONTROLLER (Implementation of AEP's hub controller)
  - SWTCONTROL (one way to control switches during simulations)
  - VSOURCE (2-terminal multiphase voltage source, thevinen equivalent)

# Models Currently Implemented (April 2011)

- GENERAL DATA
  - CNDATA (Concentric neutral cable data)
  - GROWTHSHAPE (Growth vs year)
  - LINECODE (Line and cable impedances, matrices or symmetrical components)
  - LINEGEOMETRY (Line geometry data)
  - LINESPACING (spacing data for LINEGEOMETRY)
  - LOADSHAPE (Load shape data)
  - PRICESHAPE (Price shape data)
  - SPECTRUM (Harmonic spectra)
  - TCC_CURVE (TCC curves)
  - TSDATA (Tape shield cable data)
  - TSHAPE (Temperature shape data)
  - WIREDATA (Wire parameters, GMR, etc.)
  - XFMRCODE (Transformer type definitions)
  - XYCURVE (Generic x-y curves)
- METERS
  - ENERGYMETER (Captures energy quantities and losses)
  - MONITOR (Captures selected quantities at a point in the circuit)
  - SENSOR (Simple monitor used for state estimation)
- OTHER
  - FAULT (1 or more faults can be placed anywhere in the circuit)
  - ISOURCE (Multi-phase current source)
  - VSOURCE (2-terminal multiphase voltage source, thevinen equivalent)

# Input Data Requirements

- The OpenDSS was designed for a **research or consulting** environment where input data might come from a variety of sources.
- The program can accept many common forms of data for describing impedances, loading, and topology of distribution systems for planning analysis.
- The OpenDSS scripting language is designed to require minimal translation from other formats of distribution data.
- The program can accept more detailed data for lines, transformers, etc. than the standard data when they are available.

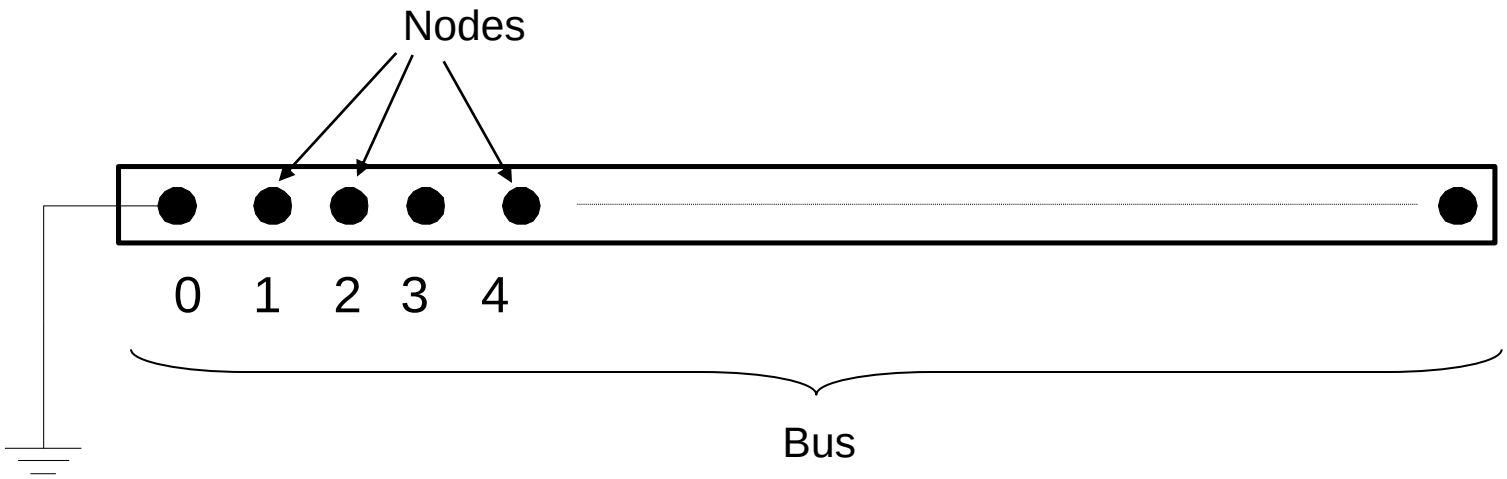
# Advanced Types of Data

- Harmonic spectra for harmonic analysis,
- Various curves as a function of time for sequential time simulations:
- Load shapes (e.g., AMI  P, Q data),
- Price shapes,
- Temperature shapes,
- Storage dispatch curves,
- Growth curves.
- Efficiency curves for PV inverters,
- Voltage dependency exponents for loads,
- Capacitor control settings,
- Regulator control settings,
- Machine data.

# Circuit Modeling Basics

Some things that might be a bit different than other power system analysis tools you may have used.

# DSS Bus Model (Bus ≠ Node)




Referring to Buses and Nodes (A Bus has 1 or more Nodes)

**Bus1=*BusName.1.2.3.0***

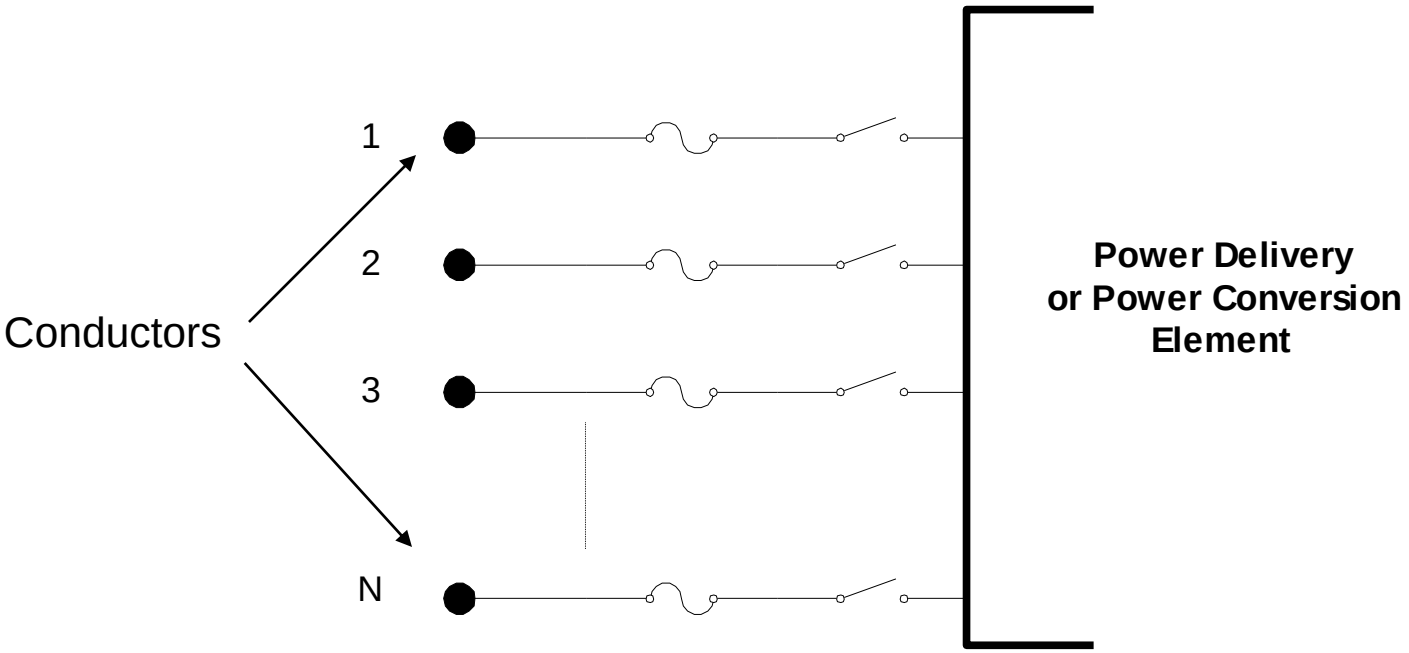
(This is the default for a 3-phase circuit element)

Shorthand notation for taking the default

**Bus1=*BusName***

*Note:* Sometimes this can bite you (e.g. – Transformers, or capacitors with ungrounded neutrals)

# DSS Terminal Definition

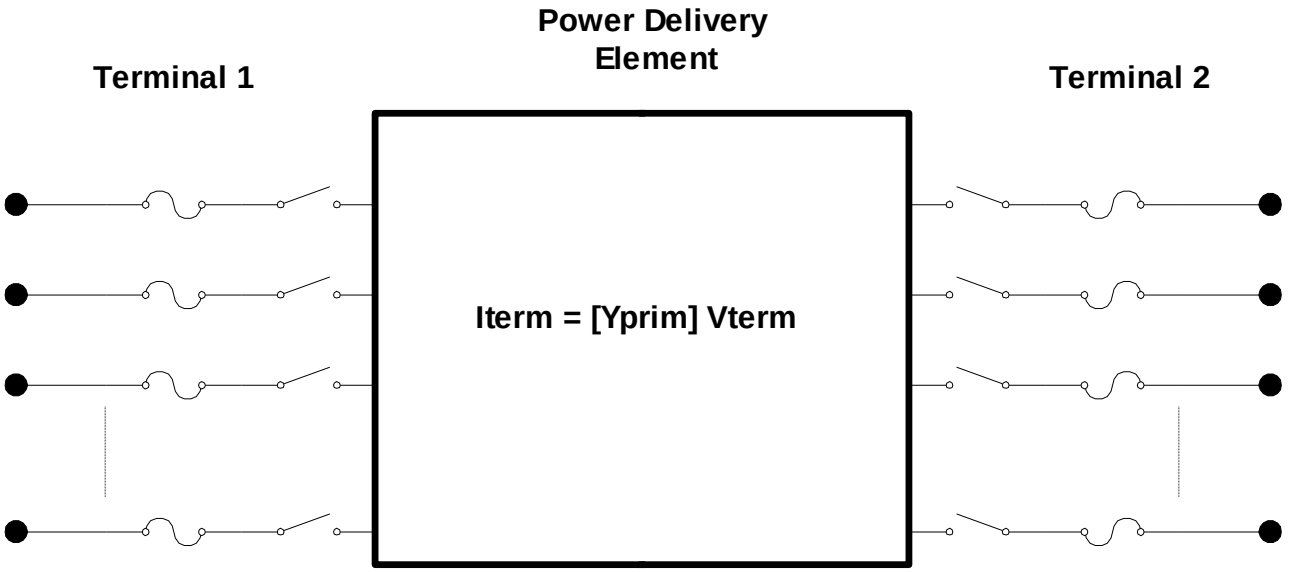



Circuit Elements have one or more *Terminals* with 1..N conductors.

*Conductors* connect to *Nodes* at a *Bus*

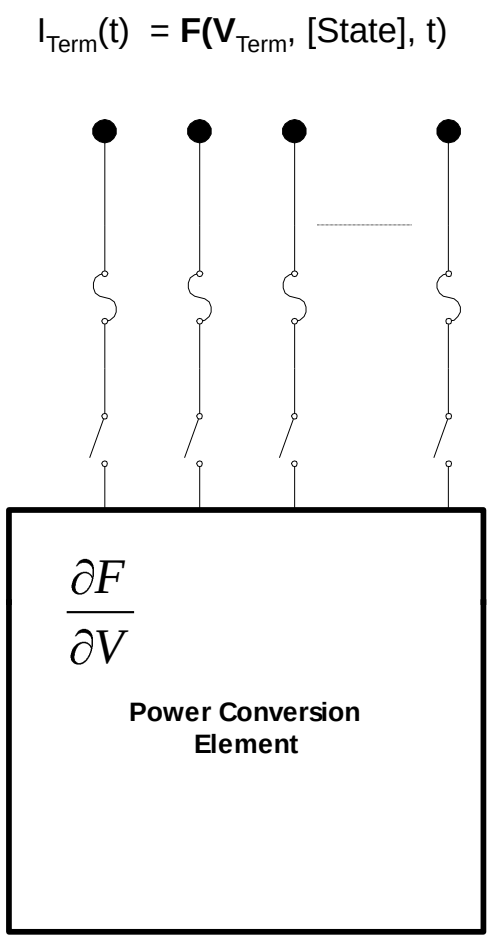
Each *Terminal* connects to one and only one *Bus*

# Power Delivery Elements




PD Elements are Generally Completely Described by *[Yprim]*

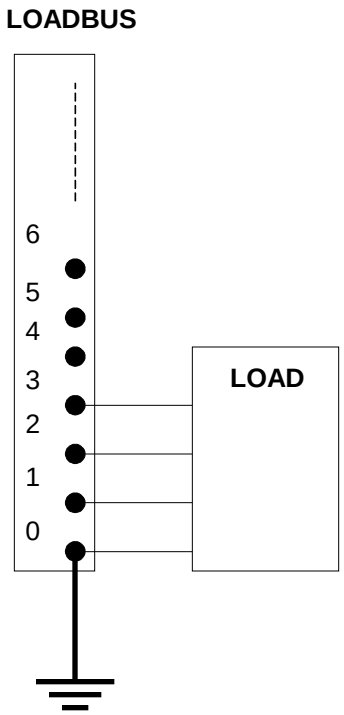
# Power Conversion Elements



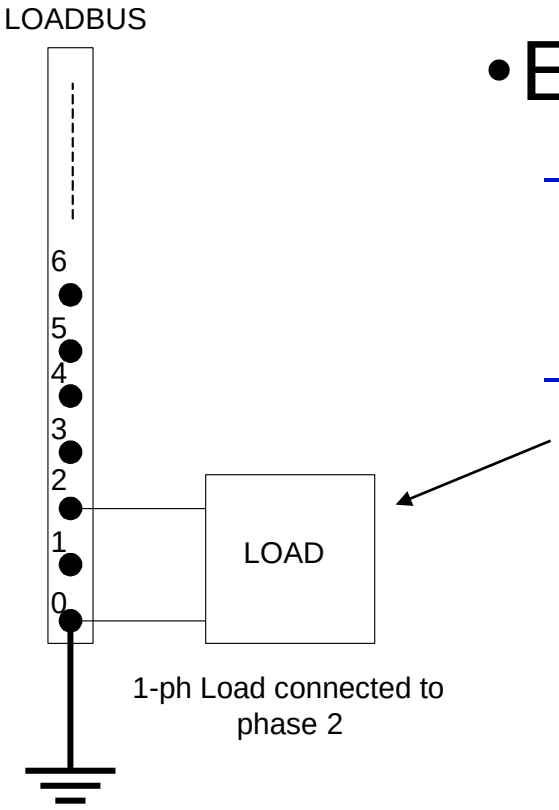

- Power Conversion (PC) elements are typically connected in “shunt” with the Power Delivery (PD) elements
- PC Elements may be nonlinear
- Described some function of V
  - May be linear
  - e.g., Vsource, Isource
- May have more than one terminal, but typically one
  - Load, generator, storage, etc.

# Specifying Bus Connections

- Shorthand (implicit)
  - **New Load.LOAD1 Bus1=LOADBUS**
    - Assumes standard 3-phase connection by default



# Specifying Bus Connections




- Explicit
  - **New Load.LOAD1 Bus1=LOADBUS.1.2.3.0**
    - Explicitly defines which node
  - **New Load.1-PHASELOAD Phases=1 Bus1=LOADBUS.2.0**
    - Connects 1-phase load to Node 2 and ground

1-Phase Load Example

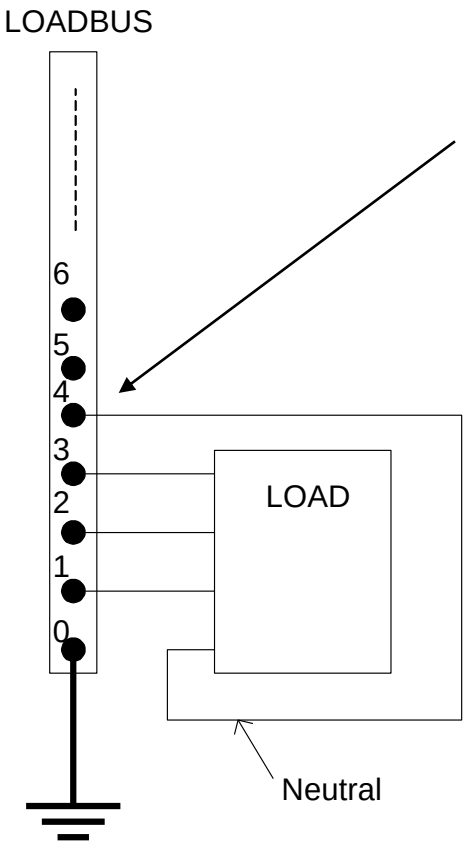
# Specifying Bus Connections

- Default Bus templates
  - Node connections assumed if not explicitly declared
  - Element declared Phases=1
    - **... LOADBUS.1.0.0.0.0.0.0.0.0. ...**
  - Element declared Phases=2
    - **... LOADBUS.1.2.0.0.0.0.0.0.0. ...**
  - Element declared Phases=3
    - **... LOADBUS.1.2.3.0.0.0.0.0.0. ...**

# Specifying Bus Connections

## Ungrounded-Wye Specification

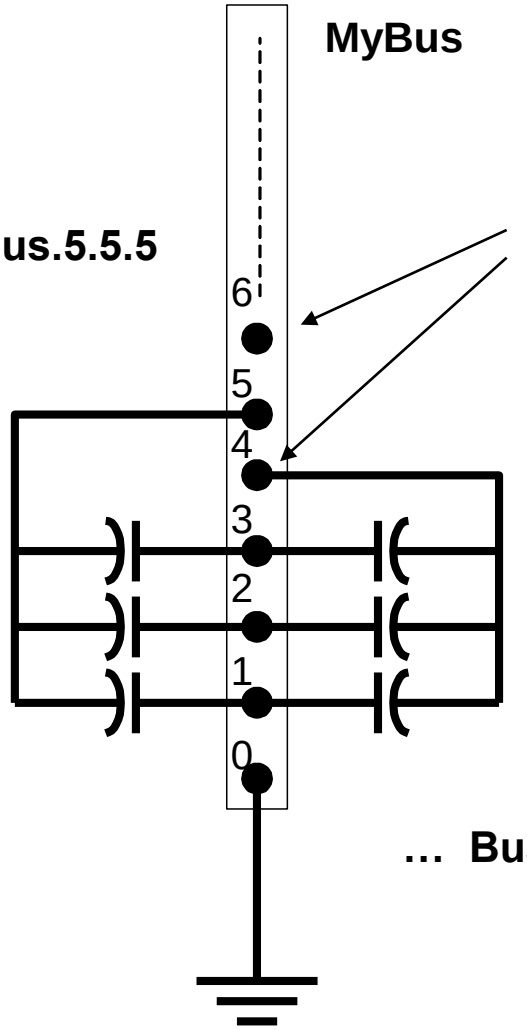
- **Bus1=LOADBUS.1.2.3.4** (or some other unused Node number)




Voltage at this Node explicitly computed (just like any other Node)

# Possible Gotcha: Specifying Two Ungrounded-Wye Capacitors on Same Bus

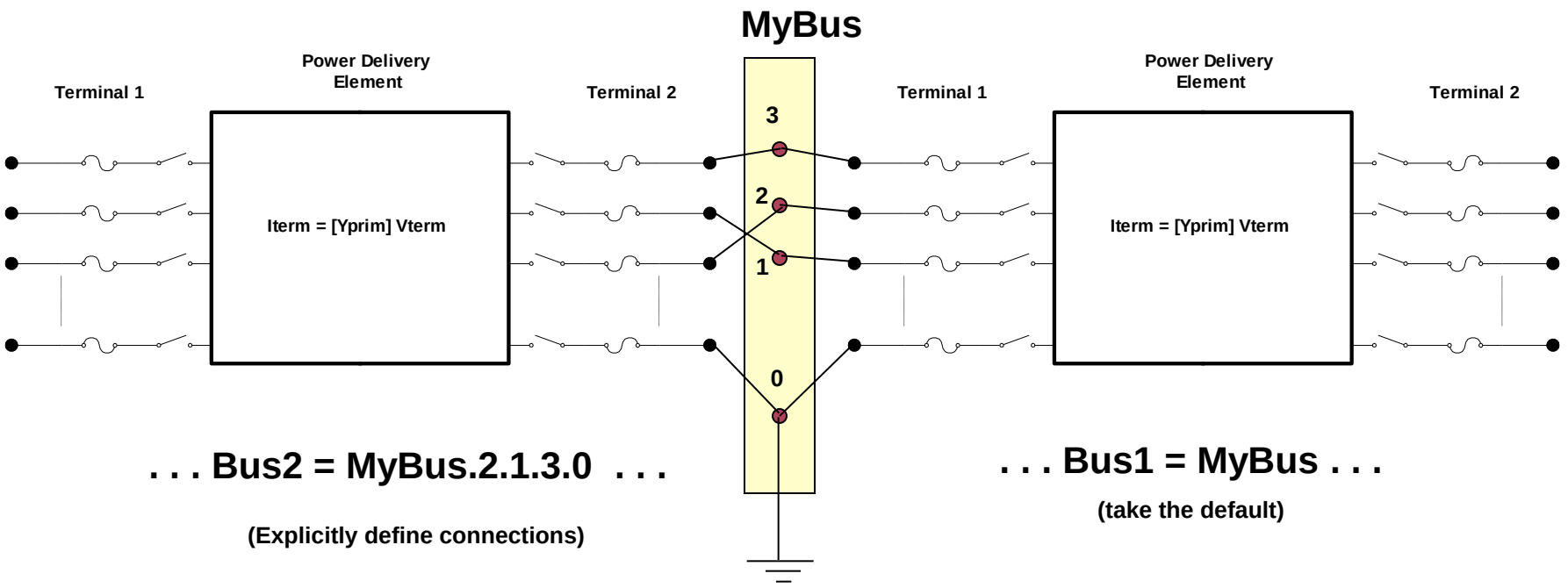
… Bus1=MyBus.1.2.3 Bus2=MyBus.5.5.5




Neutrals are not connected to each other in this specification!

… Bus1=MyBus Bus2=MyBus.4.4.4

# Circuit Element Conductors are Connected to the Nodes of Buses

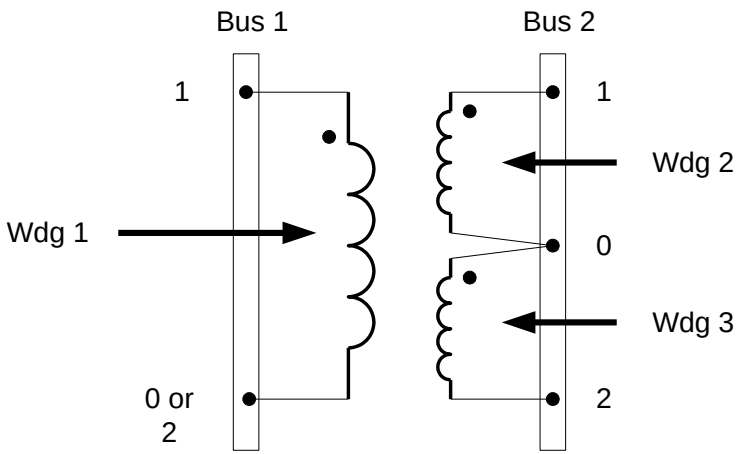



DSS Convention: A *Terminal* can be connected to only one *Bus*.
You can have any number of *Nodes* at a bus.

# Example: Connections for 1-Phase Residential Transformer Used in North America

```
! Line-to-Neutral Connected 1-phase Center-tapped transformer
New  Transformer.Example_1-ph  phases=1  Windings=3
! Typical impedances for small transformer with interlaced secondaries
~ Xhl=2.04    Xht=2.04    Xlt=1.36    %noloadloss=.2
! Winding Definitions
~ wdg=1   Bus=Bus1.1.0   kV=7.2    kVA=25   %R=0.6   Conn=wye
~ wdg=2   Bus=Bus2.1.0   kV=0.12   kVA=25   %R=1.2   Conn=wye
~ Wdg=3   Bus=Bus2.0.2   kV=0.12   kVA=25   %R=1.2   Conn=wye
```

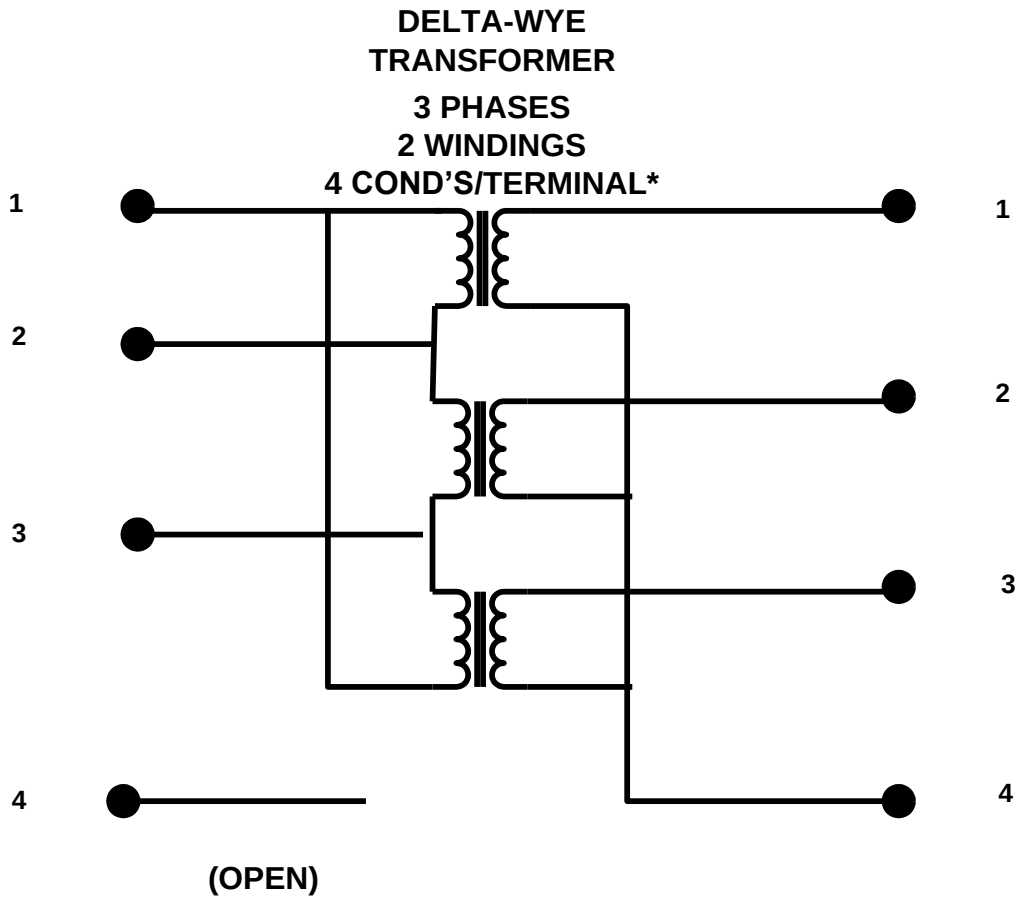
Note: You may use *XfmrCode* to define a library of transformer definitions that are used repeatedly (like *LineCode* for Line elements)



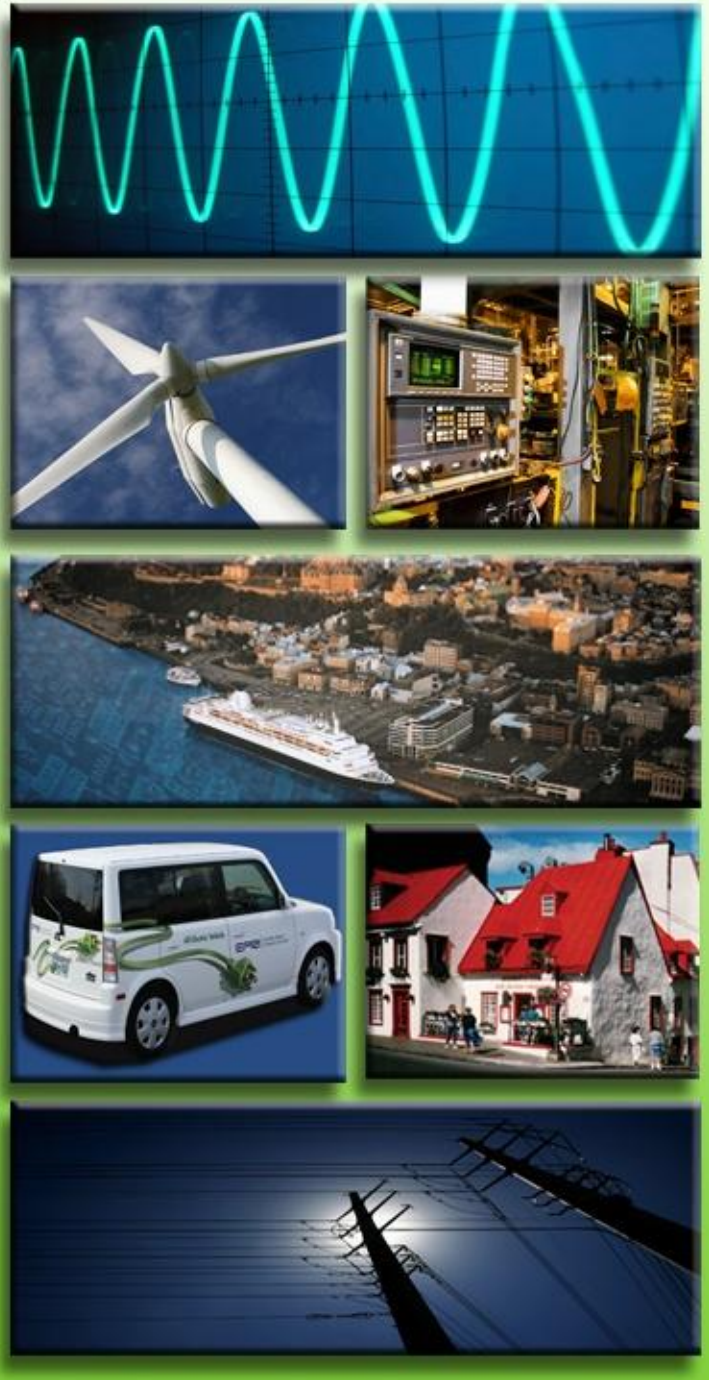

Center-Tapped 1-Phase Transformer Model

# All Terminals of a Circuit Element Have Same Number of Conductors

3-Phase Transformer




*** MUST HAVE THE SAME NUMBER OF CONDUCTORS FOR EACH TERMINAL**



EPRI | ELECTRIC POWER RESEARCH INSTITUTE

# Power Flow and Harmonics Solution Mode

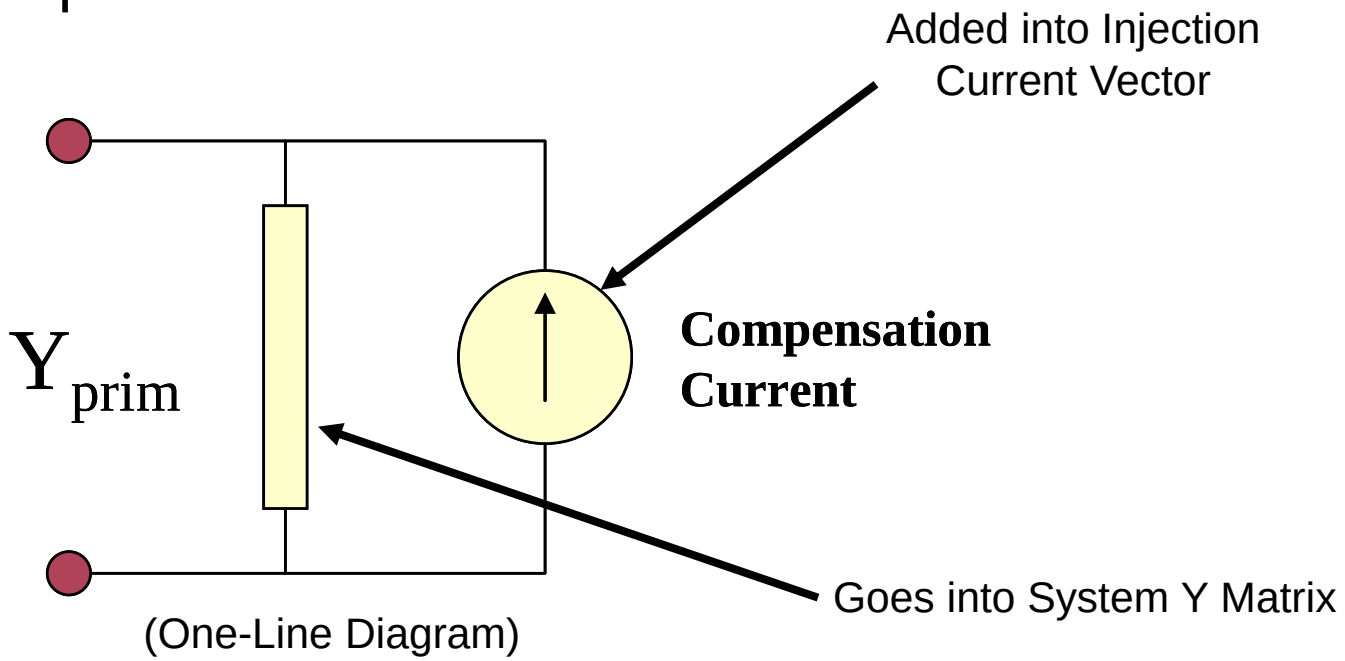
"How to make a harmonics solver solve the power flow problem"

# Solving the Power Flow

- Once the circuit model is connected properly the next step is to **Solve** the base power flow
- PC elements (i.e., Loads) are usually **nonlinear**
- Loads are linearized to a Norton equivalent based on nominal 100% rated voltage.
  - Current source is "**compensation current**"
  - Compensates for the nonlinear characteristic
- A *fixed point* iterative solution algorithm is employed for most solutions
- This method allows for flexible load models and is robust for most distribution systems

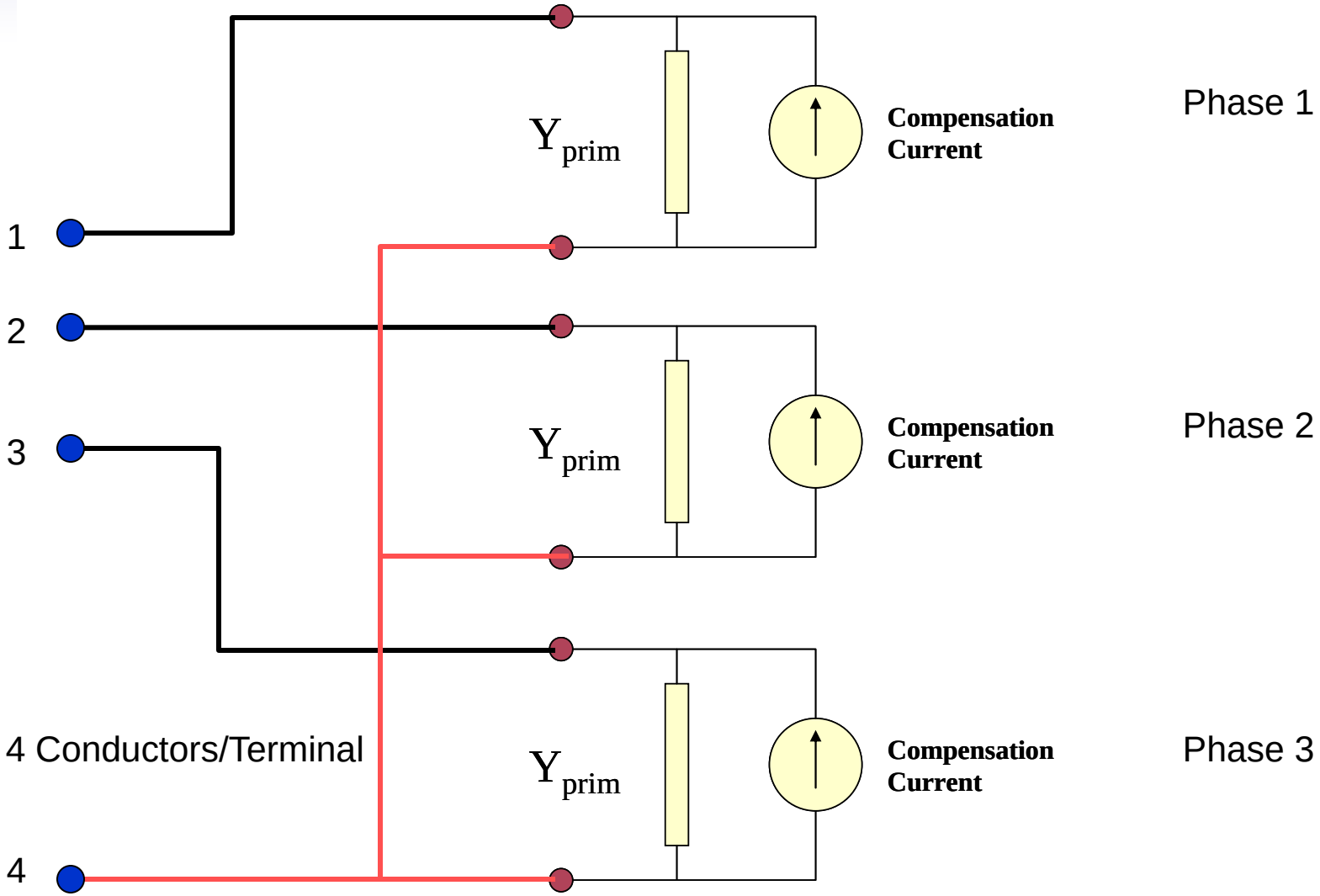
# Load (a PC Element)

General Concept

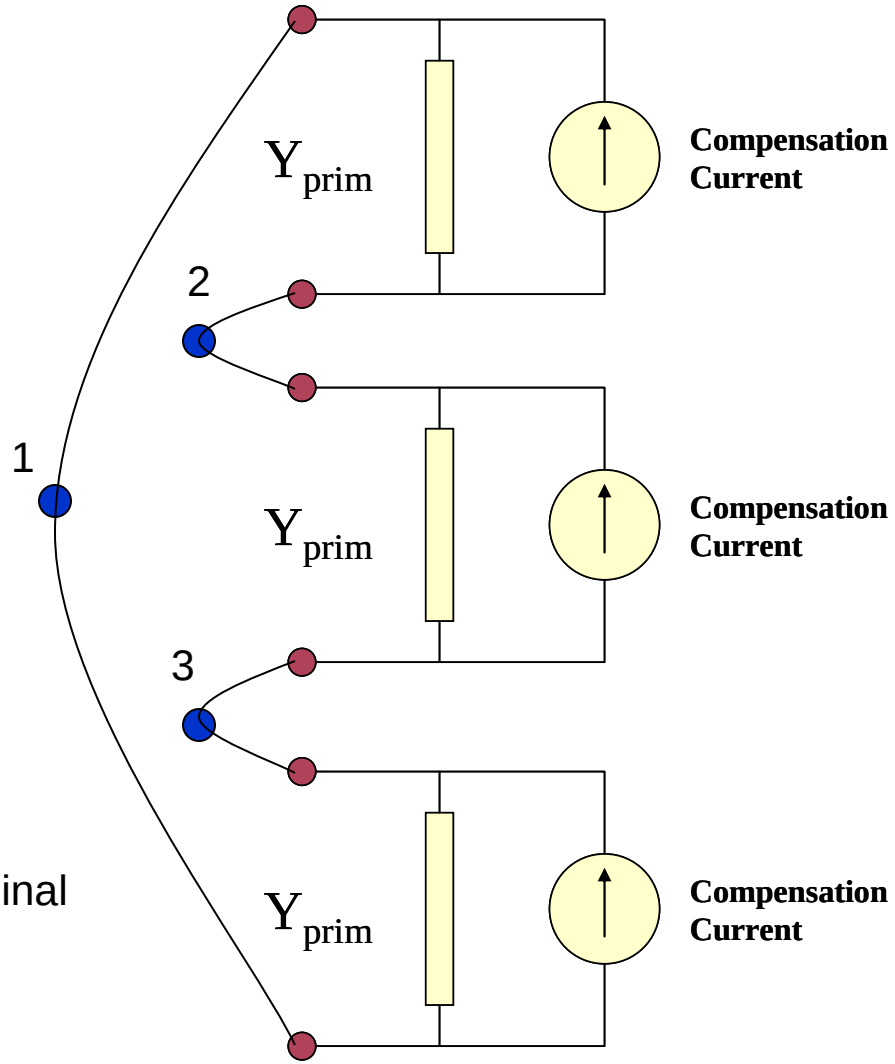



Most Power Conversion (PC) Elements are Modeled Like This

# Load - 3-phase Y connected



# Load - 3-phase Delta connected




3 Conductors/Terminal

# Load Models (Present version)

1:Standard constant P+jQ load. (Default)

2:Constant impedance load.

3:Const P, Quadratic Q (like a motor).

4:Nominal Linear P, Quadratic Q (feeder mix).
Use this with CVRfactor.

5:Constant Current Magnitude

6:Const P, Fixed Q

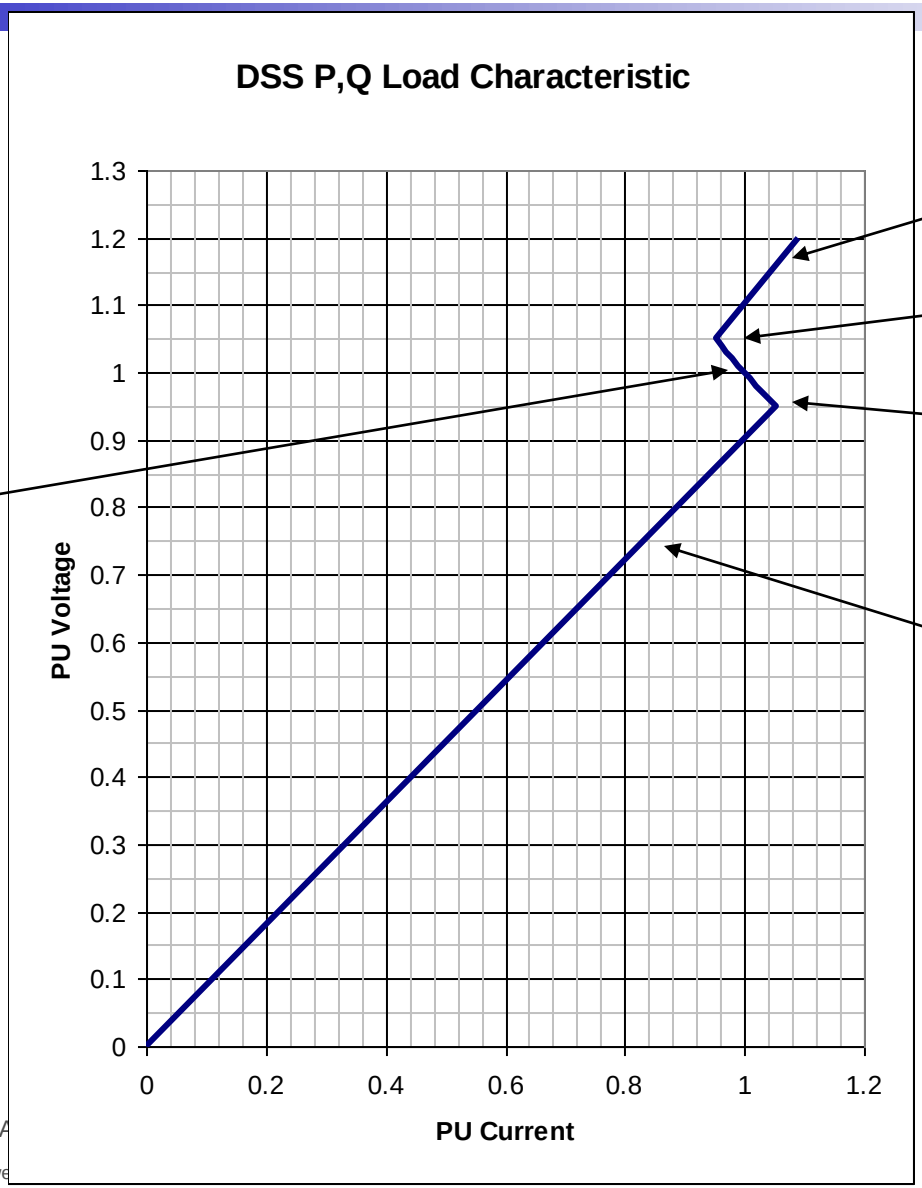
7:Const P, Fixed Impedance Q

8: Special ZIP load model (new)

# Standard P + jQ (constant power) Load Model

- When the voltage goes out of the normal range for a load the model reverts to a linear load model
  - This generally guarantees convergence
    - Even when a fault is applied
  - Script to change break points to +/- 10%:
    - Load.Load1.Vmaxpu=1.10
    - Load.Load1.Vminpu=0.90
  - Note: to solve some of the IEEE Radial Test feeders and match the published results, you have to set Vminpu to less than the lowest voltage published (usually about 0.80 per unit)

# Standard P + jQ Load Model (Model=1)




Const Z

105%

(Defaults*)

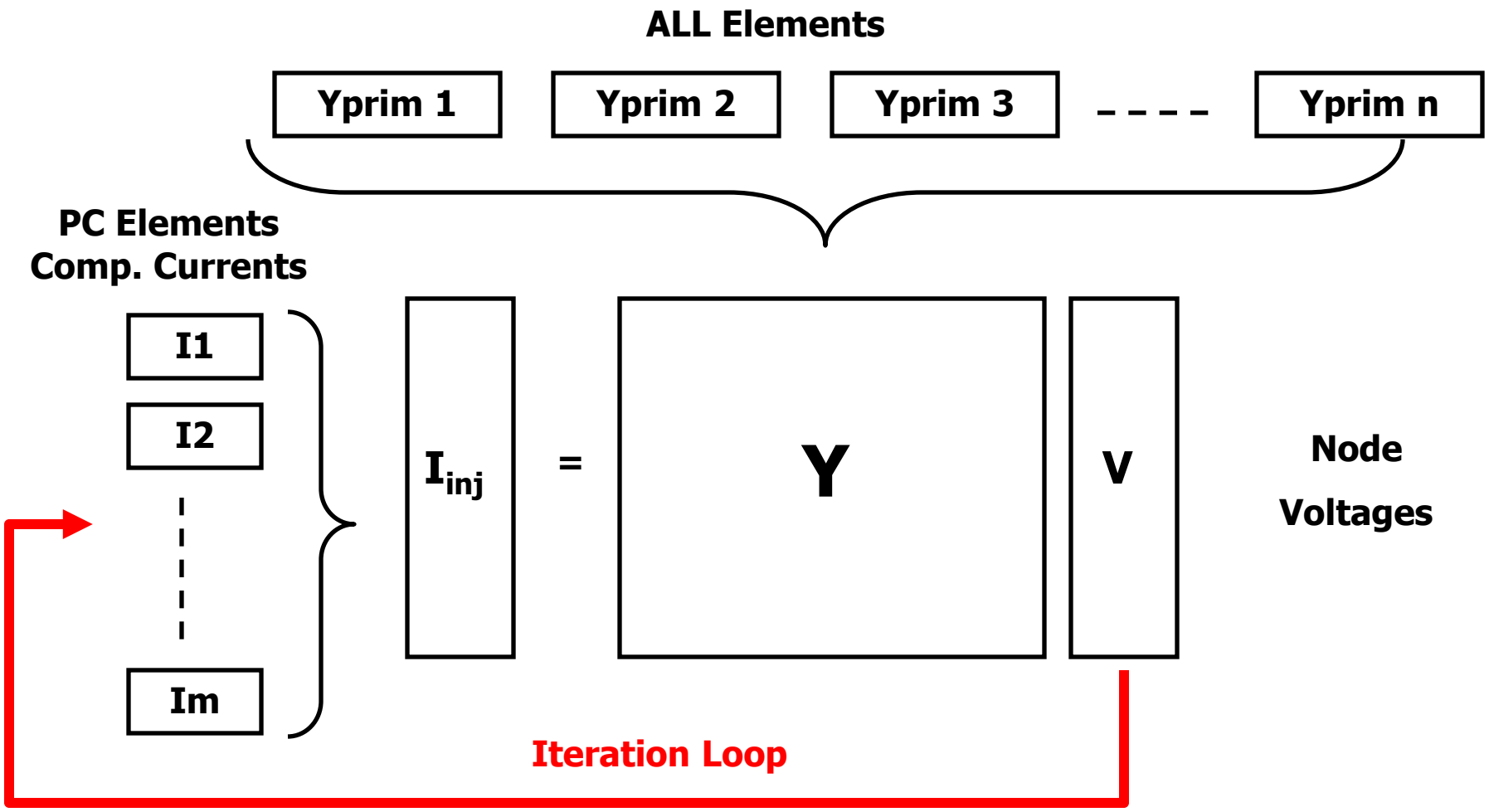
95%

|I| = |S/V|

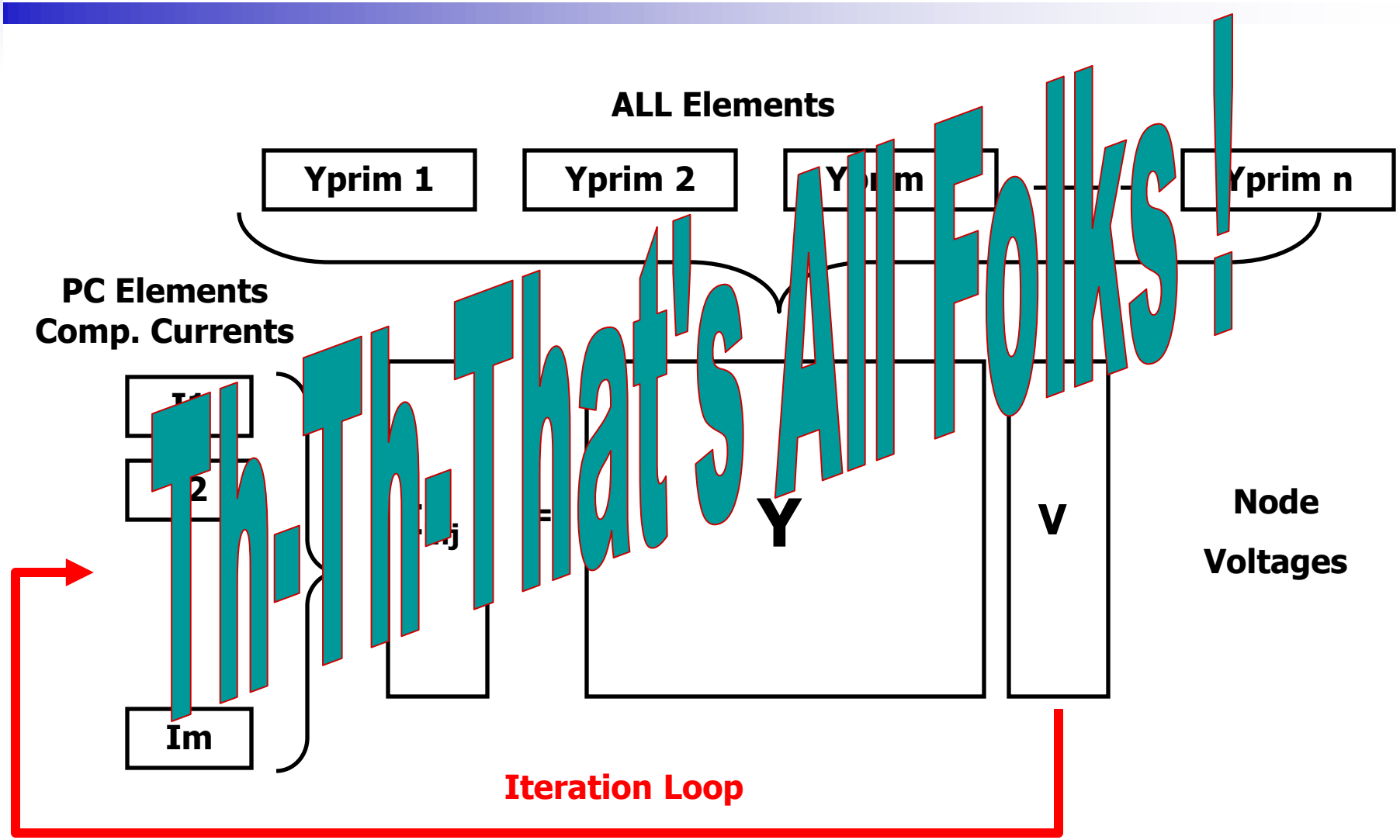
Const Z

* Change by setting *Vminpu* and *Vmaxpu* Properties

# Putting it All Together



# Putting it All Together



# Solving the Power Flow …

- This solution method requires that the first guess at the voltages be close to the final solution
  - Not a problem for daily or yearly simulations
    - Present solution is a good initial guess at next time step
  - First solution is often most difficult
- The solution initialization routine in OpenDSS accomplishes this with ease in most cases
- Method works well for arbitrary unbalances
  - For conditions that are sensitive, a *Newton* method is provided that is more robust, but slower.
  - Not to be confused with “Newton-Raphson Power Flow”

# Solution Speed

- Distribution systems generally converge very well
  - Many transmission systems, also
- The OpenDSS program seems to be on par with the speed of faster commercial programs
- Solution method is designed to run annual simulations
- Our philosophy:
  - ***Err on the side of running more power flow simulations***
    - Don't worry about the solution time until it proves to be a problem
    - This reveals more information about the problem

EPRI ELECTRIC POWER RESEARCH INSTITUTE

# Harmonic Solution

# Solving for Harmonic Flows

- The OpenDSS solution algorithm heritage
  - McGraw-Edison Steady-State Analysis Program (MESSAP) 1977-78
  - McGraw-Edison Harmonic Analysis Program (MEHAP) 1979-1982
  - McGraw/Cooper V-Harm® program (1984)
  - Electrotek Concepts SuperHarm® program (1990-92)
  - Electrotek DSS (1997)
- Harmonic solution has been an integral part of the program since its inception

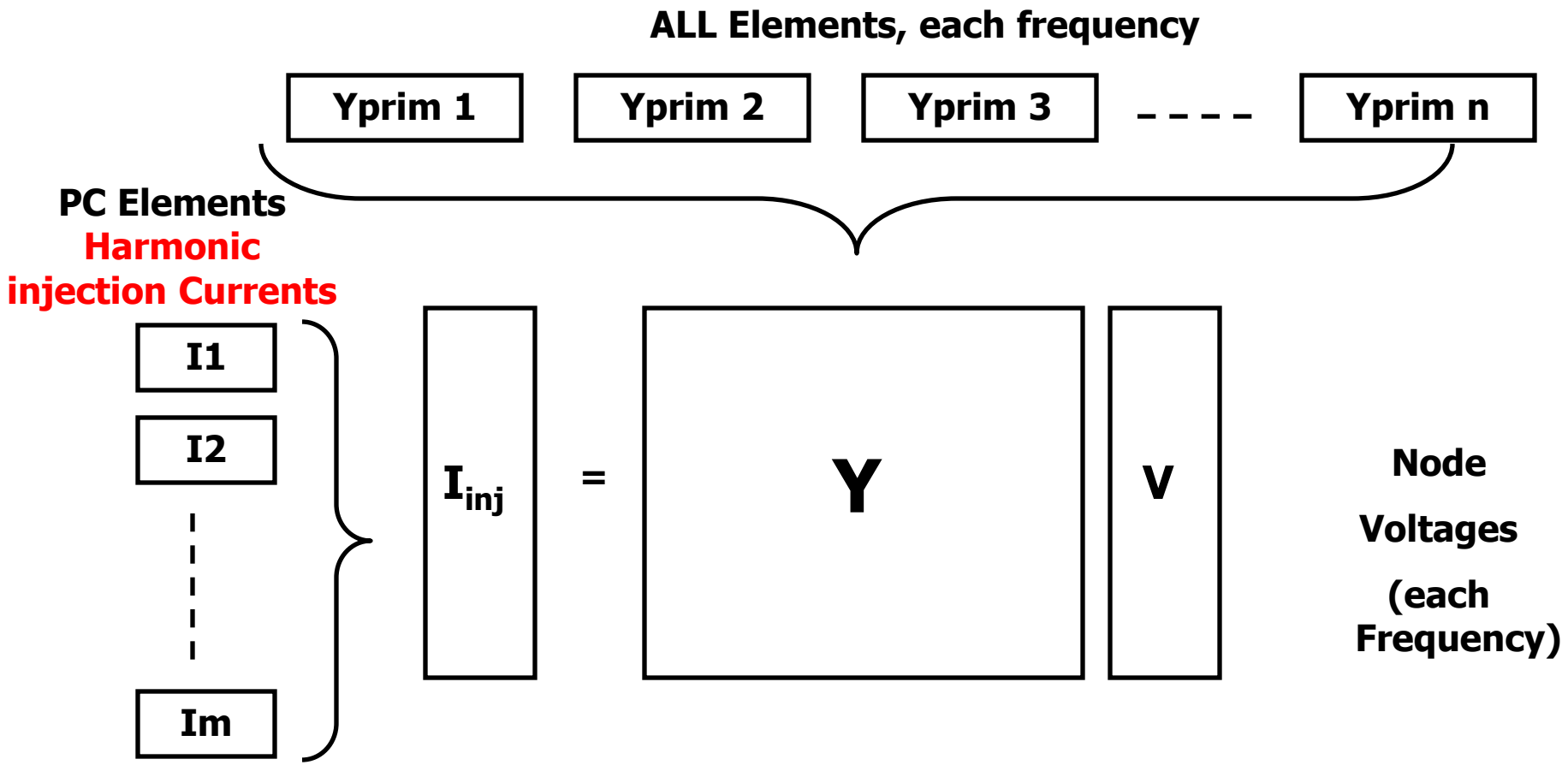
# Harmonic Solution

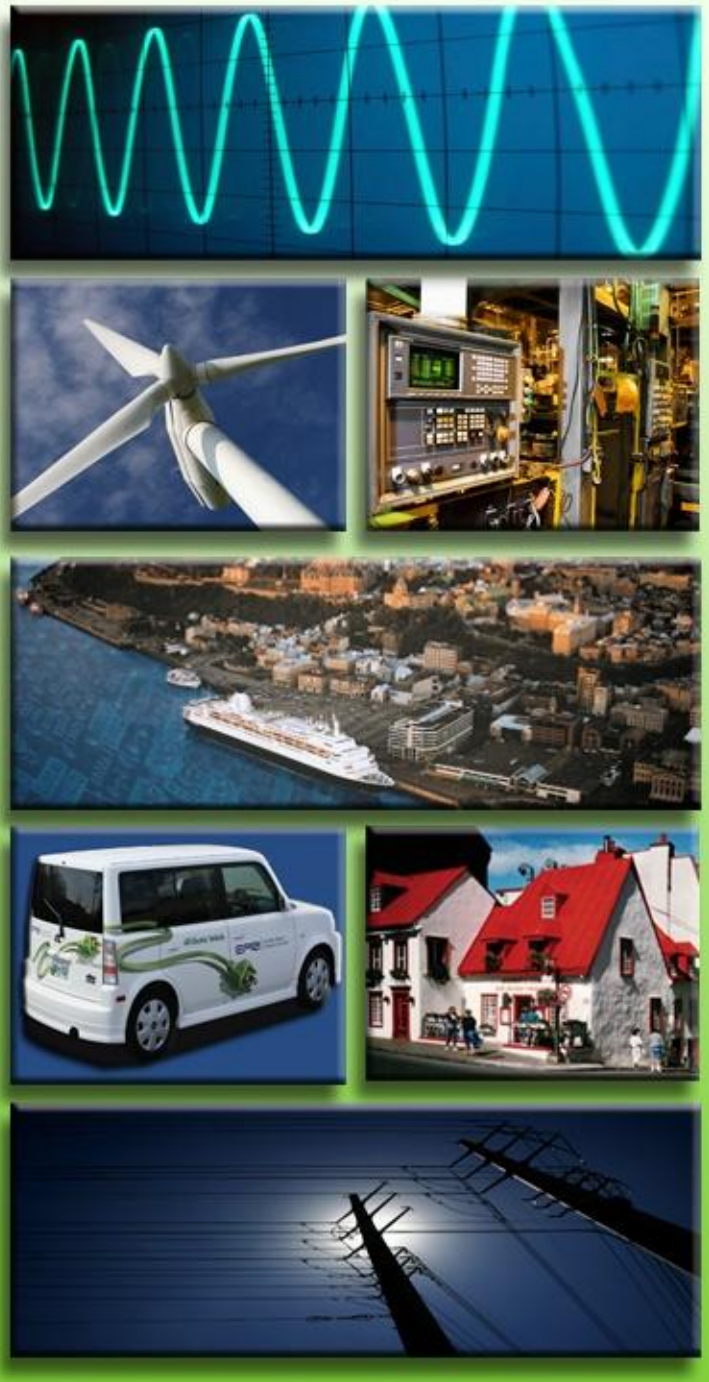
- All Load elements have 'Spectrum=default'
  - The default spectrum is a typical 3-phase ASD spectrum
  - Redefine *Spectrum.Default* if you want something else
    - Edit spectrum.default …
- Harmonic solution requires a converged power flow solution to initialize all the harmonic sources
  - Do "*Solve mode=direct"* if convergence is difficult to achieve
- **Typical script for Harmonics Solution**
  - `Solve  ! Snapshot power flow`
  - `Solve mode=harmonics`

# Harmonic Solution Mode

- Solve the power flow (fundamental frequency)
- Harmonic currents are initialized to match the fundamental frequency solution
- "Solve Mode=Harmonics"
  - Program solves at each frequency
  - Non-iterative (direct solution)
  - Harmonic currents assumed invariant
  - Y matrix is rebuilt for each frequency
- (Make sure you have Monitors where you want to see results because answers will come fast.)

# Putting it All Together - Harmonics





EPRI ELECTRIC POWER RESEARCH INSTITUTE

# Scripting Basics

Syntax and how to build circuit models.

Simple Model

# Scripting

- OpenDSS is a scriptable solution engine
- Scripts
  - Series of commands
  - From text files
  - From edit forms in OpenDSS.EXE
  - From another program through COM interface
    - e. g., This is how you would do looping
- Scripts define circuits
- Scripts control solution of circuits
- Scripts specify output, etc.

# Command Syntax

- *Command parm1, parm2 parm3 parm 4 ….*

- Parameters may be positional or named (tagged).
- If named, an "**=**" sign is expected.
  - ***Name=value*** *(this is the named form)*
  - ***Value*** *(value alone in positional form)*
- *For example, the following two commands are equivalent:*
  - `New Object="Line.First Line" Bus1=b1240 Bus2=32 LineCode=336ACSR, …`
  - `New "Line.First Line", b1240 32 336ACSR, …`

Comma or white space

# Delimiters

- Array or string delimiter pairs: **[ ] , { },( ),“ “,‘ ‘**
- Matrix row delimiter: **|**
- Value delimiters: **,** (comma)
  any white space (tab or space)
- Class, Object, Bus, or Node delimiter: **.** (period)
- Keyword / value separator: **=**
- Continuation of previous line: **~** (More)
- Comment line: ***//***
- In-line comment: **!**
- Query a property: **?**

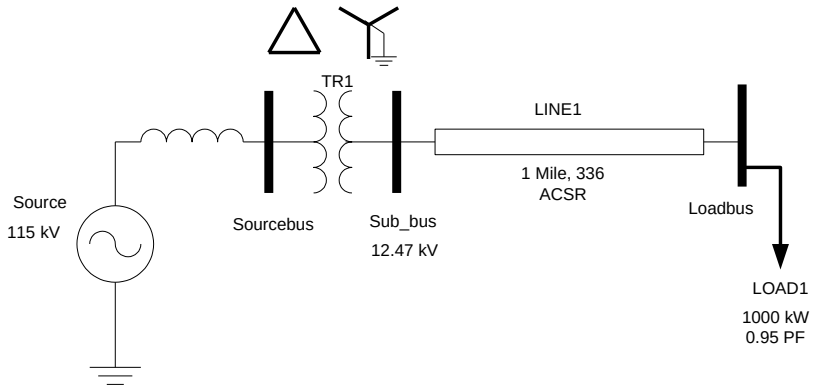
# Array and Matrix Parameters

- Array
  - **kvs = [115, 6.6, 22]**
  - **kvas=[20000  16000 16000]**

- Matrix
  - ***(3x3 matrix)***
    - **Xmatrix=[1.2  .3  .3 | .3  1.2  3 | .3  .3  1.2]**
  - ***(3x3 matrix – lower triangle)***
    - **Xmatrix=[ 1.2  | .3 1.2  | .3  .3  1.2 ]**

# An Example

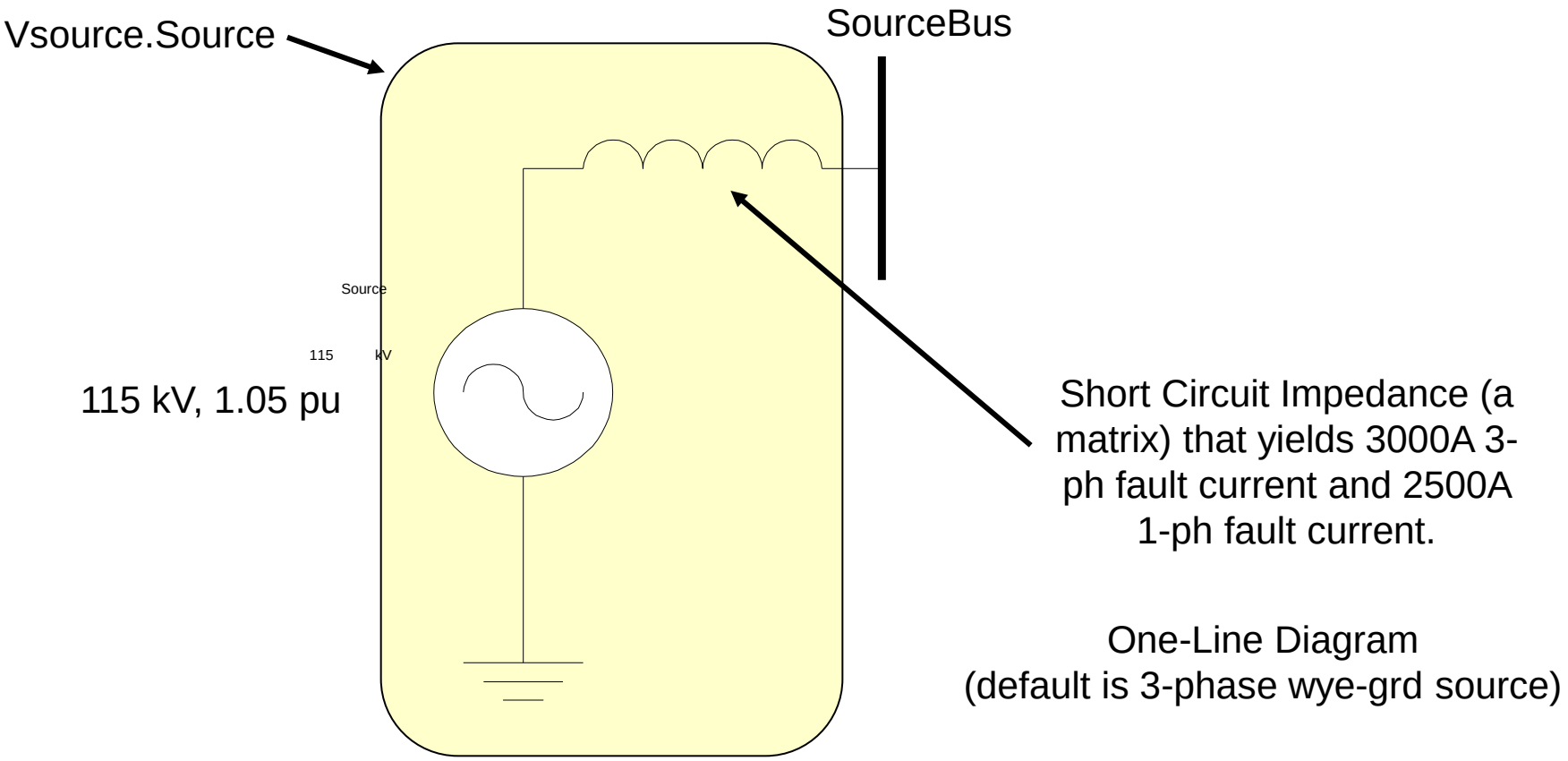
# A Basic Script (Class Exercise)




```
New Circuit.Simple     ! Creates voltage source  (Vsource.Source)
Edit Vsource.Source BasekV=115 pu=1.05  ISC3=3000  ISC1=2500  !Define source V and Z
New Transformer.TR1 Buses=[SourceBus, Sub_Bus] Conns=[Delta Wye] kVs= [115 12.47]
~ kVAs=[20000 20000] XHL=10
New Linecode.336ACSR R1=0.058 X1=.1206 R0=.1784 X0=.4047 C1=3.4 C0=1.6 Units=kft
New Line.LINE1 Bus1=Sub_Bus Bus2=LoadBus Linecode=336ACSR Length=1 Units=Mi
New Load.LOAD1 Bus1=LoadBus kV=12.47 kW=1000 PF=.95
Solve
Show Voltages
Show Currents
Show Powers kVA elements
```

# Circuit

- **New Circuit.Simple ! (Vsource.Source is active circuit element)**
- **Edit Vsource.Source BasekV=115 pu=1.05 ISC3=3000 ISC1=2500**

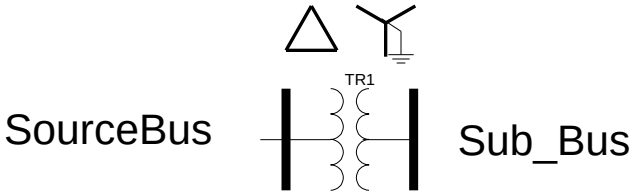


# Vsource Element Note

- Vsource is actually a ***Two-terminal Device***
  - 2nd terminal defaults to connected to ground (0V)
  - But you can connect it between any two buses
    - Comes in handy sometimes

- Conceptually a Thevinen equivalent
  - Short circuit equivalent
  - Actually converted to a Norton equivalent internally

# 20 MVA Substation Transformer




## Defining Using Arrays

**New Transformer.TR1 Phases=3 Windings=2**

**~ Buses=[SourceBus, Sub_Bus]**

**~ Conns=[Delta Wye]**

**~ kVs= [115 12.47]**

**~ kVAs=[20000 20000]**

**~ XHL=10**

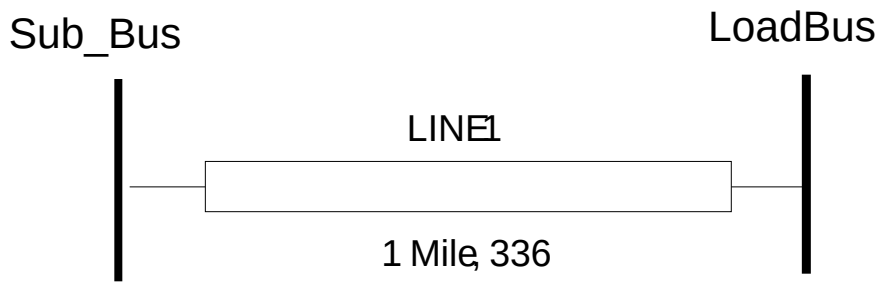
## Defining Winding by Winding

**New Transformer.TR1 Phases=3 Windings=2 XHL=10**

**~ wdg=1 bus=SourceBus Conn=Delta kV=115 kVA=20000**

**~ wdg=2 bus= Sub_Bus Conn=wye kV=12.47 kVA=20000**

# The Line



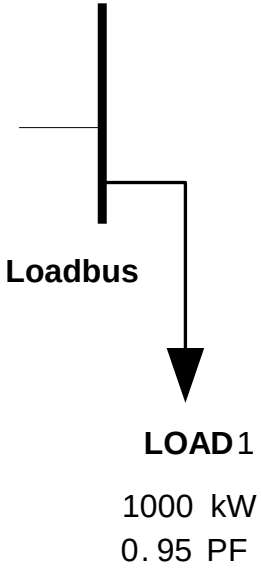

**New Linecode.336ACSR R1=0.058 X1=.1206 R0=.1784 X0=.4047 C1=3.4 C0=1.6 Units=kft**

**New Line.LINE1 Bus1=Sub_Bus Bus2=LoadBus Linecode=336ACSR Length=1 Units=Mi**

Line objects may also be defined by **Geometry** or **matrix** properties.

**(Rmatrix=… Xmatrix=… Cmatrix=…)**

# The Load




**New Load.LOAD1 Bus1=LoadBus kV=12.47 kW=1000 PF=.95**

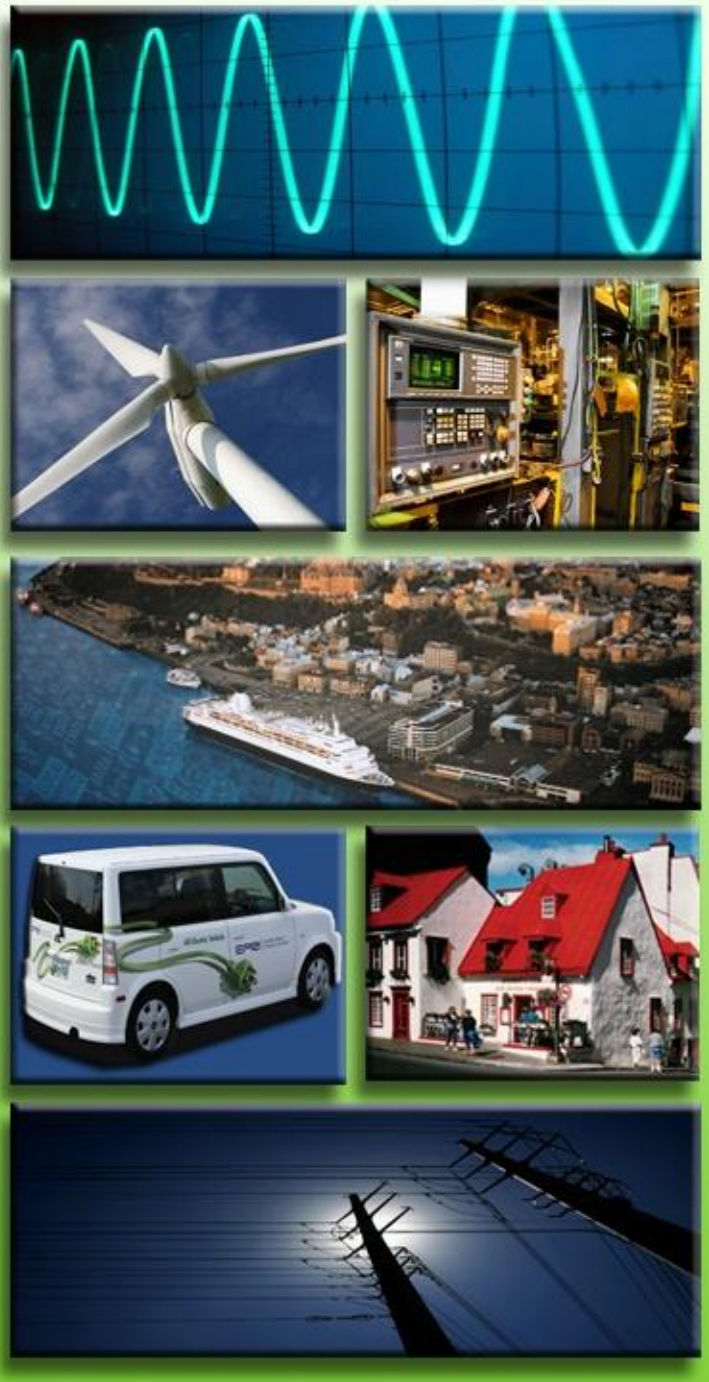
For 3-phase loads, use L-L kV and total kW

For 1-phase loads, typically use L-N kV and total kW unless L-L-connected; Then use L-L kV.

# Solving and Showing Results Reports

- **Solve**
  - **Show summary** (power flow summary)
- **Show Voltages**
- **Show Currents**
- **Show Powers kVA elements**

- Also
  - Export … (creates CSV files)
  - Plot …



EPRI ELECTRIC POWER RESEARCH INSTITUTE

# Scripting for Larger Circuits

How to organize scripts for larger problems

Examination of how the IEEE 8500-Node Test Feeder model is organized

# Scripting Large Circuits

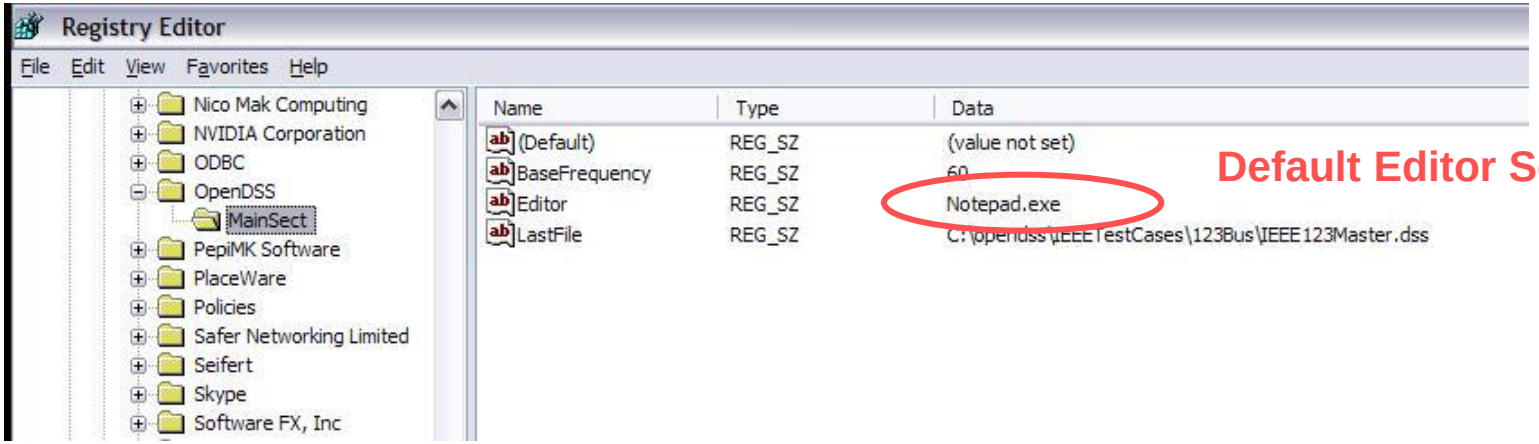
- For small circuits, it is often sufficient to put all the scripts in a single file
  - Many of the IEEE test feeder examples are mostly in a single file
- When you have large amounts of data, a more disciplined approach is recommended
- Redirect Command
  - Redirects the input to another file
  - Returns to home directory
- Compile Command
  - Same as Redirect except repositions home directory

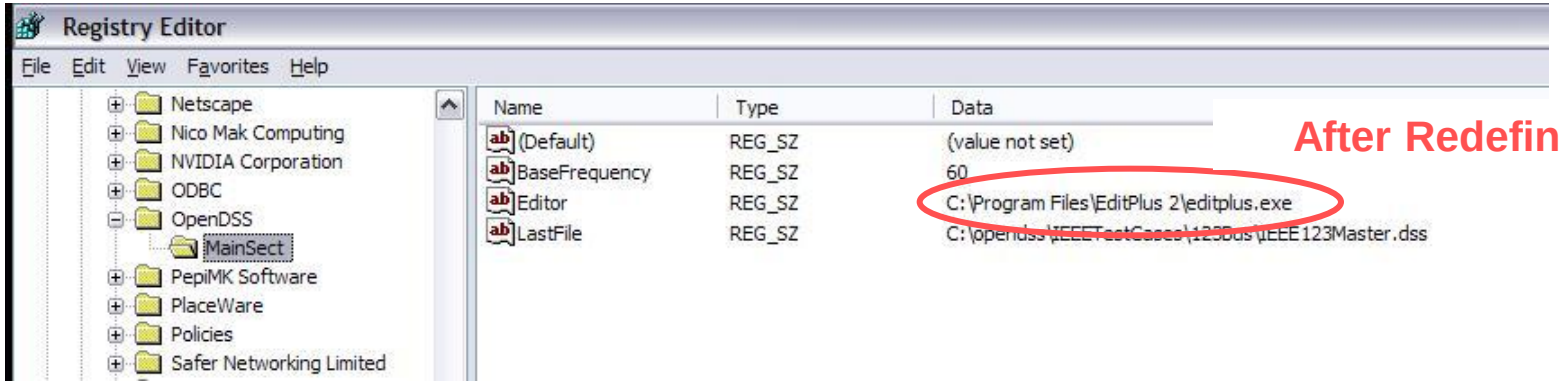
# Organizing Your Main Screen

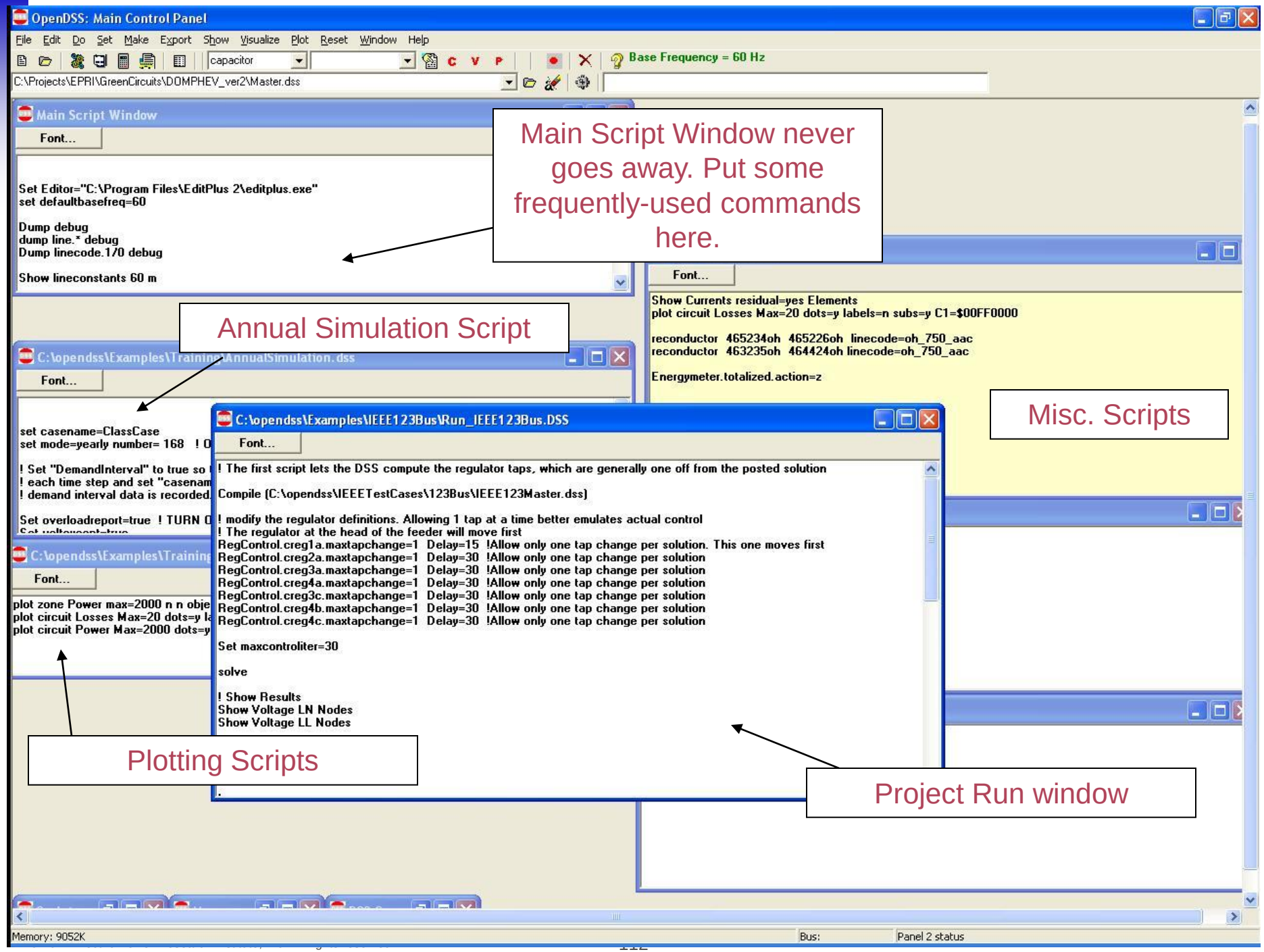
- OpenDSS.exe saves all windows on the main screen
  - They appear where you left them when you shut down
  - The next time you start up, you can resume your work
- Values are saved in a file (*OpenDSS.ini*)
  - in the OpenDSS.exe folder
    - Note: You can update the program simply by copying in new exe and dll files.
  - Do not overwrite the ".ini" file if you want to preserve your workspace
    - However, if the .ini file gets corrupted, you may simply delete it.

# OpenDSS Registry Entries

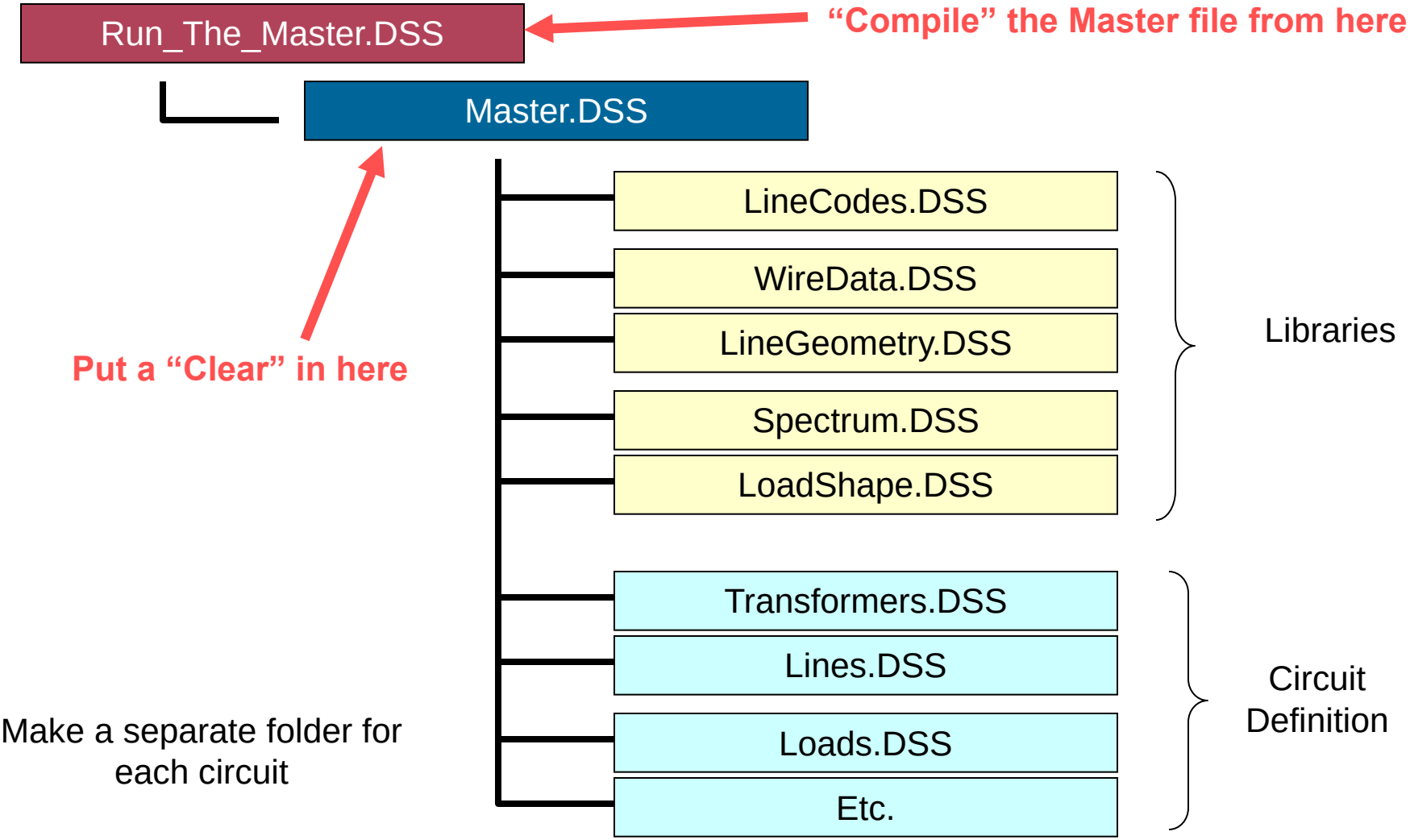
- Certain persistent values are saved to the Windows Registry upon exiting the program



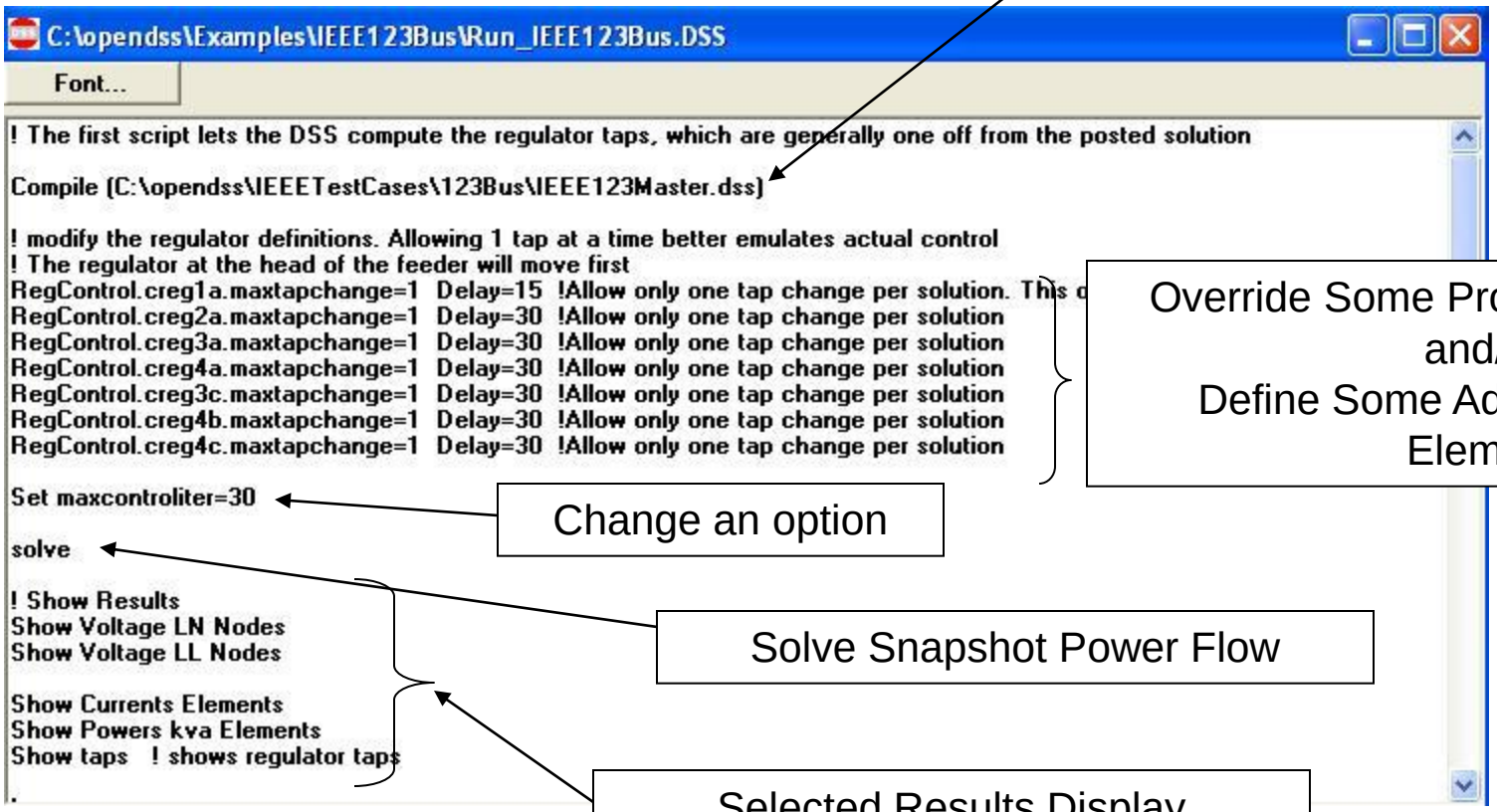

OpenDSS: Main Control Panel
File Edit Do Set Make Export Show Visualize Plot Reset Window Help
capacitor
Base Frequency = 60 Hz
C:\Projects\EPRI\GreenCircuits\DOMPHEV_ver2\Master.dss
Main Script Window
Font...
Set Editor="C:\Program Files\EditPlus 2\editplus.exe"
set defaultbasefreq=60
Dump debug
dump line.* debug
Dump linecode.170 debug
Show lineconstants 60 m
Main Script Window never goes away. Put some frequently-used commands here.
Annual Simulation Script
C:\opendss\Examples\Training\AnnualSimulation.dss
Font...
set casename=ClassCase
set mode=yearly number= 168 ! O
! Set "DemandInterval" to true so
! each time step and set "casenam
! demand interval data is recorded.
Set overloadreport=true ! TURN O
C:\opendss\Examples\Training
Font...
plot zone Power max=2000 n n obje
plot circuit Losses Max=20 dots=y l
plot circuit Power Max=2000 dots=y
Plotting Scripts
C:\opendss\Examples\IEEE123Bus\Run_IEEE123Bus.DSS
Font...
! The first script lets the DSS compute the regulator taps, which are generally one off from the posted solution
Compile (C:\opendss\IEEETestCases\123Bus\IEEE123Master.dss)
! modify the regulator definitions. Allowing 1 tap at a time better emulates actual control
! The regulator at the head of the feeder will move first
RegControl.creg1a.maxtapchange=1 Delay=15 !Allow only one tap change per solution. This one moves first
RegControl.creg2a.maxtapchange=1 Delay=30 !Allow only one tap change per solution
RegControl.creg3a.maxtapchange=1 Delay=30 !Allow only one tap change per solution
RegControl.creg4a.maxtapchange=1 Delay=30 !Allow only one tap change per solution
RegControl.creg3c.maxtapchange=1 Delay=30 !Allow only one tap change per solution
RegControl.creg4b.maxtapchange=1 Delay=30 !Allow only one tap change per solution
RegControl.creg4c.maxtapchange=1 Delay=30 !Allow only one tap change per solution
Set maxcontroliter=30
solve
! Show Results
Show Voltage LN Nodes
Show Voltage LL Nodes
Project Run window
Font...
Show Currents residual=yes Elements
plot circuit Losses Max=20 dots=y labels=n subs=y C1=$00FF0000
reconductor 465234oh 465226oh linecode=oh_750_aac
reconductor 463235oh 464424oh linecode=oh_750_aac
Energymeter.totalized.action=z
Misc. Scripts
Memory: 9052K
Bus:
Panel 2 status

# A Common Sense Structuring of Script Files



# Organizing Run Scripts

Based on 123-bus Test Feeder



# Organizing Your Master File

```
Clear

New Circuit.ExampleCircuit BasekV=138 pu=1.05 MVASC3 = 2000  MVASC1=2000

! Master file examples

! Library files
Redirect LineCode.dss
Redirect LoadShape.dss
Redirect GrowthShape.dss
Redirect TCC_Curve.dss
Redirect Spectrum.dss

! Circuit element descriptions are in a subdirectory "Feeders"
Redirect Feeders\Transformers.dss
Redirect Feeders\Branches.dss
Redirect Feeders\Loads.dss
Redirect Feeders\Capacitors.dss

Set Voltagebases=(69, 12.1, 4.16, 0.48)     ! define legal voltage bases
calcv        ! Abbrev for CalcVoltageBases

! Buses exit now so define coordinates
Buscoords buscoords.txt     ! Load bus x,y coor

! Define energy meters after voltage bases so
Redirect EnergyMeter.dss

! Don't do Solve here ... better to do it in Run File
```

So Compile Doesn't Fail

General Library Data

Circuit Elements for this Model

Let OpenDSS Define the Voltage Bases

(You can do this explicitly with SetkVBase command)

EPRI ELECTRIC POWER RESEARCH INSTITUTE

# Example:

# IEEE 8500-Node Test Feeder

# Main Part of "Run" File

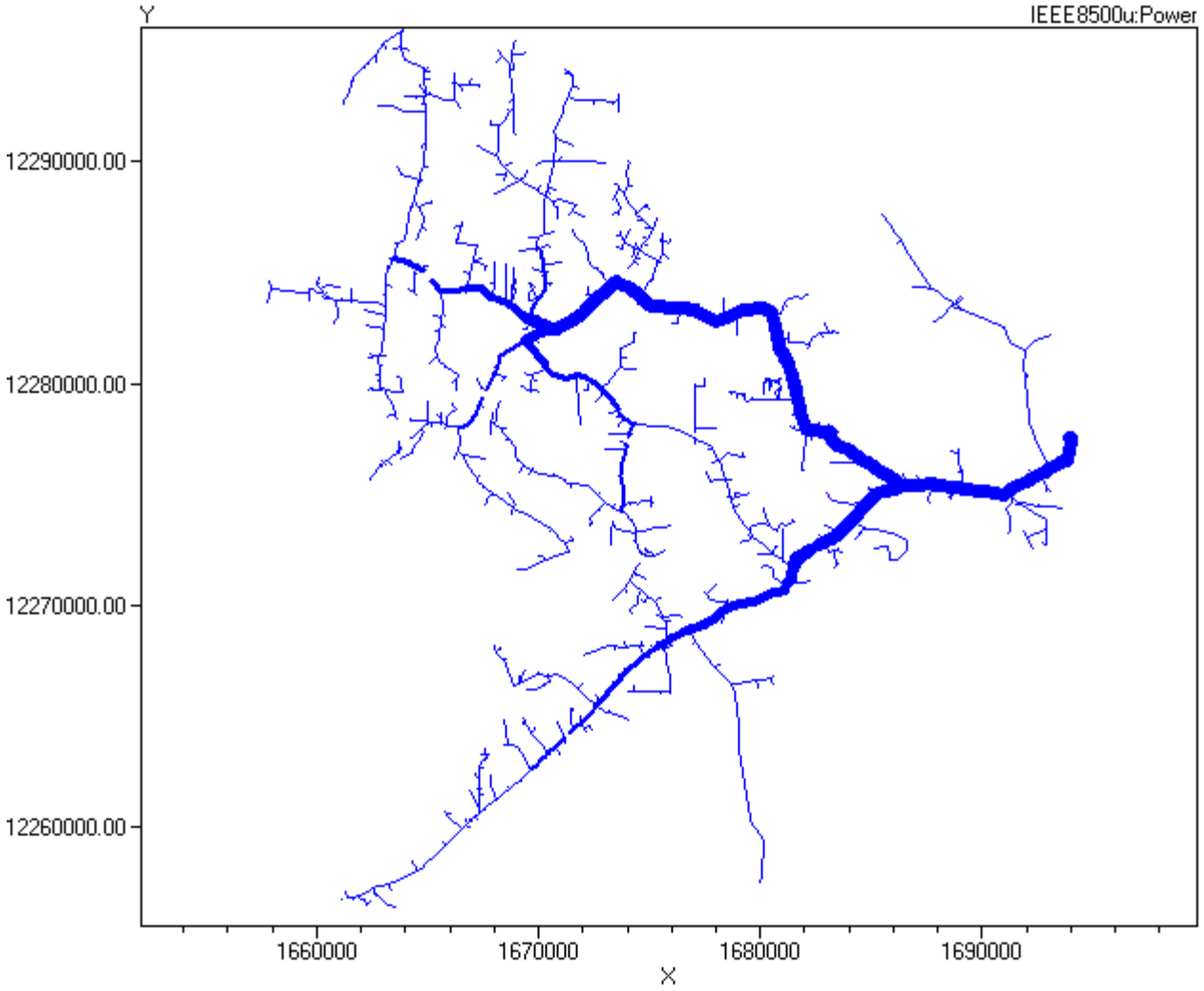
1. Compile base circuit description
2. Add an energymeter not in base description
3. Change an option
4. Solve

```
Compile (C:\DSSdata\IEEETest\8500Node\Master-unbal.dss)

New Energymeter.m1 Line.ln5815900-1 1
  ! Put an Energymeter at the head of the feeder

Set Maxiterations=20
   ! Sometimes the solution takes more than the default 15 iterations


Solve
```

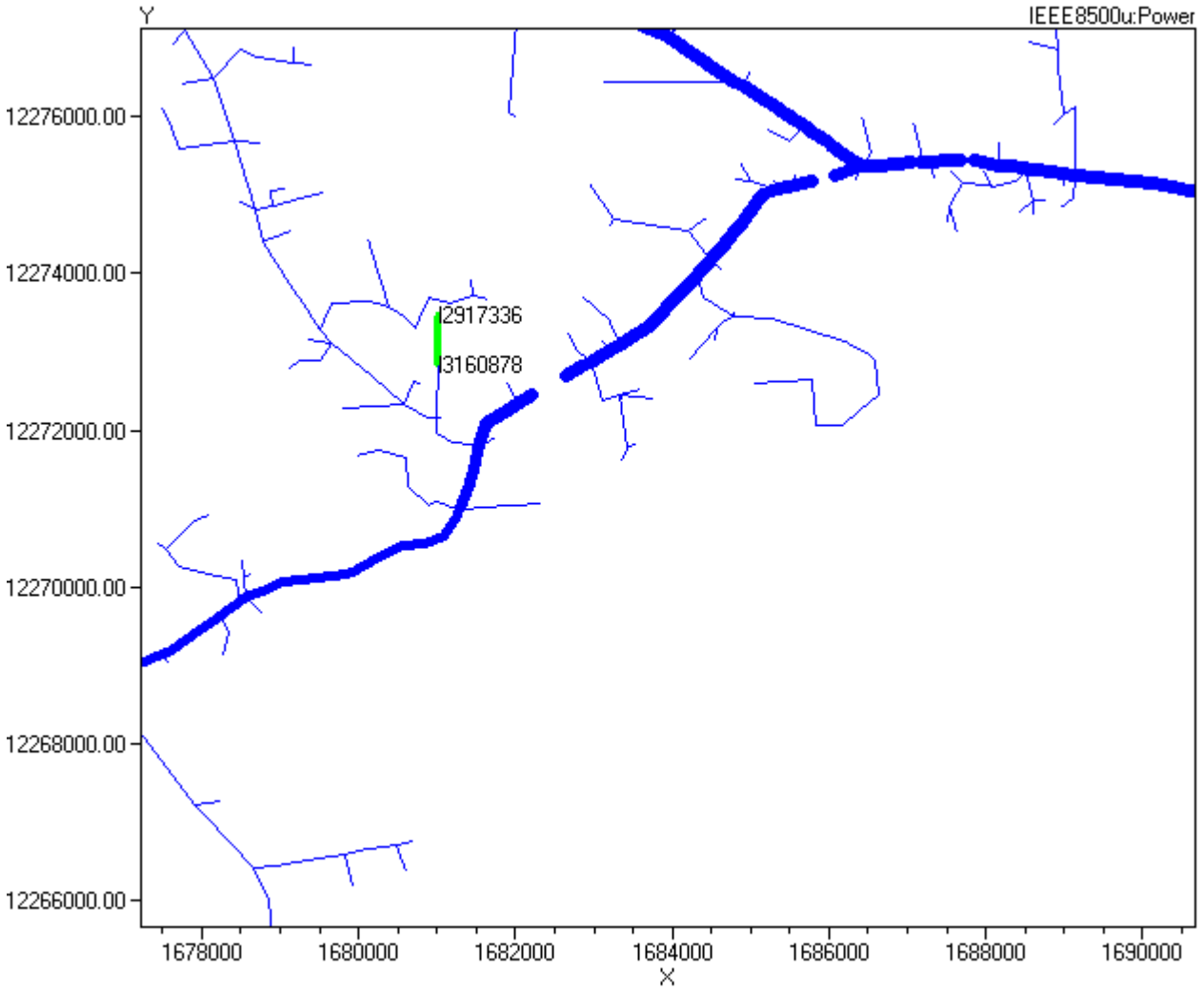
# The Master File

```
Clear

New Circuit.IEEE8500u

! Make the source stiff with small impedance
~ pu=1.05  r1=0  x1=0.001  r0=0  x0=0.001

Redirect  LineCodes2.dss
Redirect  Triplex_Linecodes.dss

Redirect  Lines.dss
Redirect  Transformers.dss
Redirect  LoadXfmrs.dss    ! Load Transformers
Redirect  Triplex_Lines.dss
Redirect  UnbalancedLoads.dss
Redirect  Capacitors.dss
Redirect  CapControls.dss
Redirect  Regulators.dss

! Let DSS estimate the voltage bases
Set voltagebases=[115, 12.47,  0.48, 0.208]
Calcvoltagebases     ! This also establishes the bus list

! Load in bus coordintes now that bus list is established
Buscoords  Buscoords.dss
```

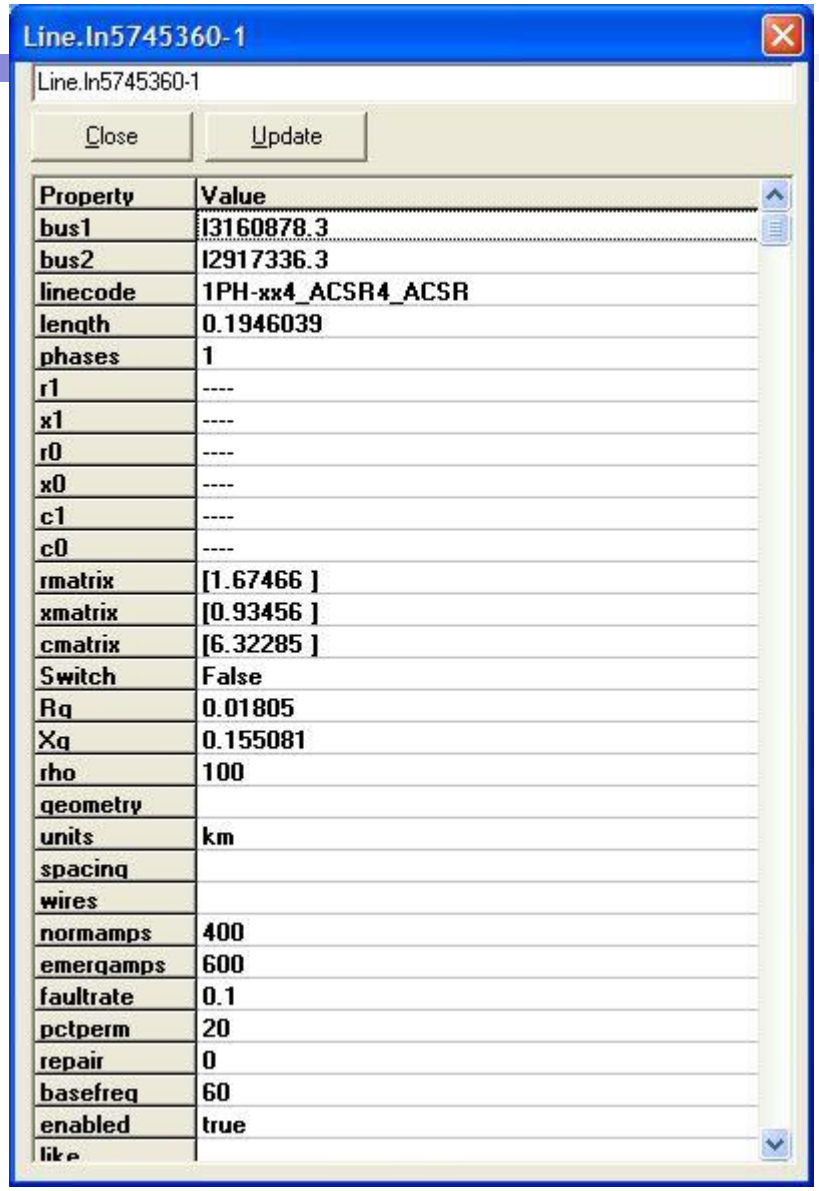
# Power Flow Solution Plot



# Selecting a Branch from the Plot (Zoomed)

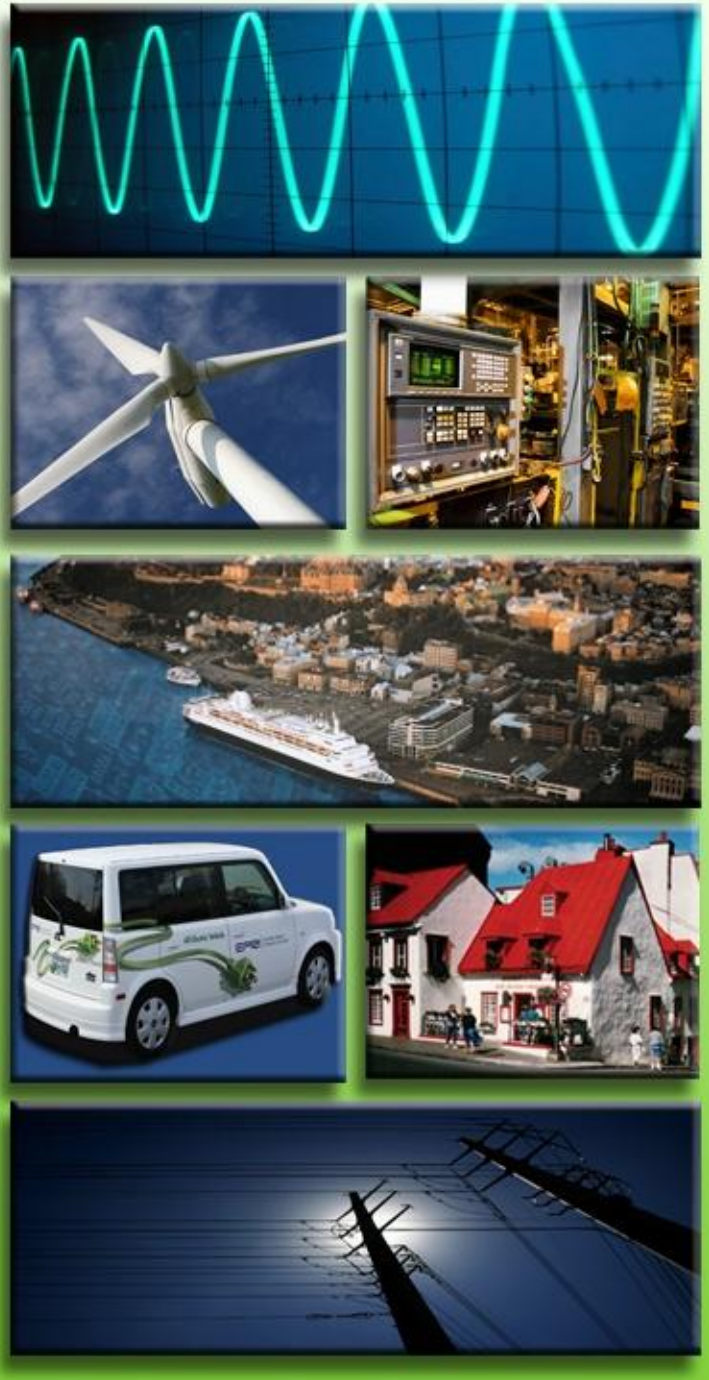


# Right-click and select Properties …



EPRI ELECTRIC POWER RESEARCH INSTITUTE

# Exercise the 8500-Node Test Feeder …



EPRI | ELECTRIC POWER RESEARCH INSTITUTE

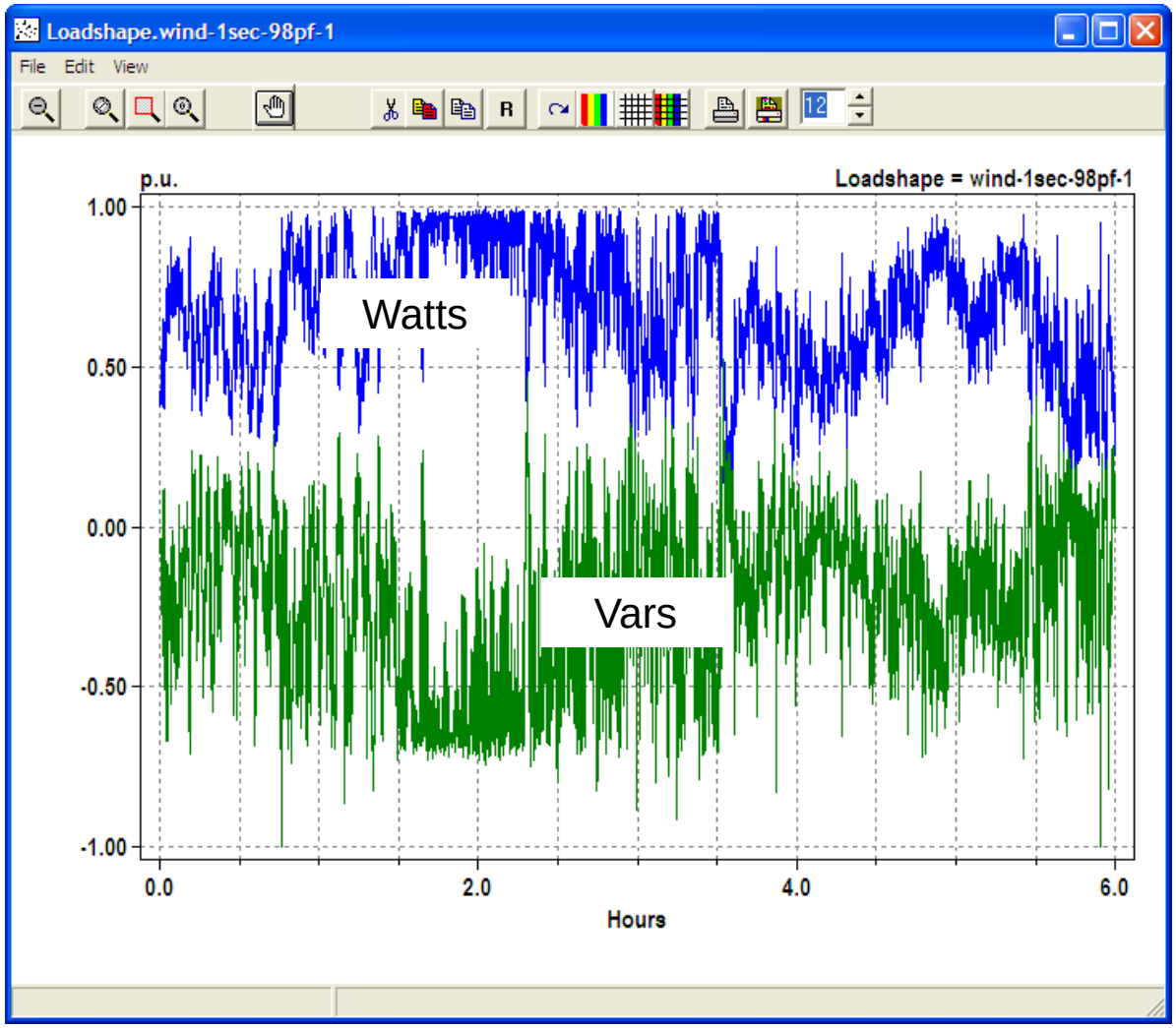
# Loadshapes and Smart Grid Simulation

How to use the OpenDSS Loadshape capability and simulate various time-series phenomena related to Smart Grid simulations.

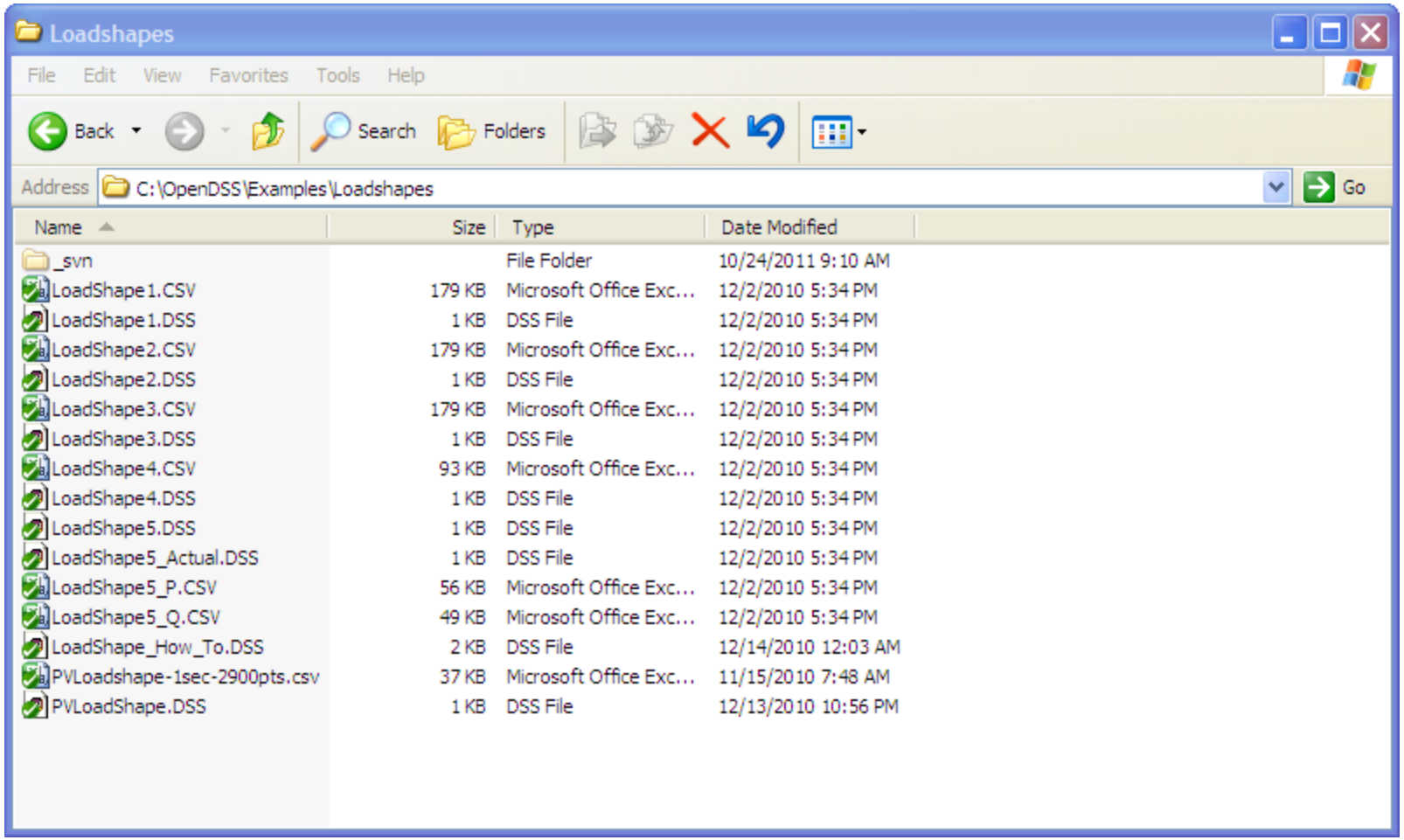
# Loadshapes

- Key feature of OpenDSS
- Sequential time simulation is required for many Smart Grid analyses
- Basic time-varying modes that use Loadshape objects
  - Daily (nominally 24 h, 1 h steps)
  - Yearly (nominally 8760 h, 1 h steps)
  - Dutycycle (nominal step size 1 s .. 5 m)
    - (used for wind and solar)

# Example Loadshape for Wind Turbine Output



# Example Loadshapes Provided in Examples Folder




| Name | Size | Type | Date Modified |
|---|---|---|---|
| _svn | | File Folder | 10/24/2011 9:10 AM |
| LoadShape1.CSV | 179 KB | Microsoft Office Exc... | 12/2/2010 5:34 PM |
| LoadShape1.DSS | 1 KB | DSS File | 12/2/2010 5:34 PM |
| LoadShape2.CSV | 179 KB | Microsoft Office Exc... | 12/2/2010 5:34 PM |
| LoadShape2.DSS | 1 KB | DSS File | 12/2/2010 5:34 PM |
| LoadShape3.CSV | 179 KB | Microsoft Office Exc... | 12/2/2010 5:34 PM |
| LoadShape3.DSS | 1 KB | DSS File | 12/2/2010 5:34 PM |
| LoadShape4.CSV | 93 KB | Microsoft Office Exc... | 12/2/2010 5:34 PM |
| LoadShape4.DSS | 1 KB | DSS File | 12/2/2010 5:34 PM |
| LoadShape5.DSS | 1 KB | DSS File | 12/2/2010 5:34 PM |
| LoadShape5_Actual.DSS | 1 KB | DSS File | 12/2/2010 5:34 PM |
| LoadShape5_P.CSV | 56 KB | Microsoft Office Exc... | 12/2/2010 5:34 PM |
| LoadShape5_Q.CSV | 49 KB | Microsoft Office Exc... | 12/2/2010 5:34 PM |
| LoadShape_How_To.DSS | 2 KB | DSS File | 12/14/2010 12:03 AM |
| PVLoadshape-1sec-2900pts.csv | 37 KB | Microsoft Office Exc... | 11/15/2010 7:48 AM |
| PVLoadShape.DSS | 1 KB | DSS File | 12/13/2010 10:56 PM |

# How to Define

- Clear

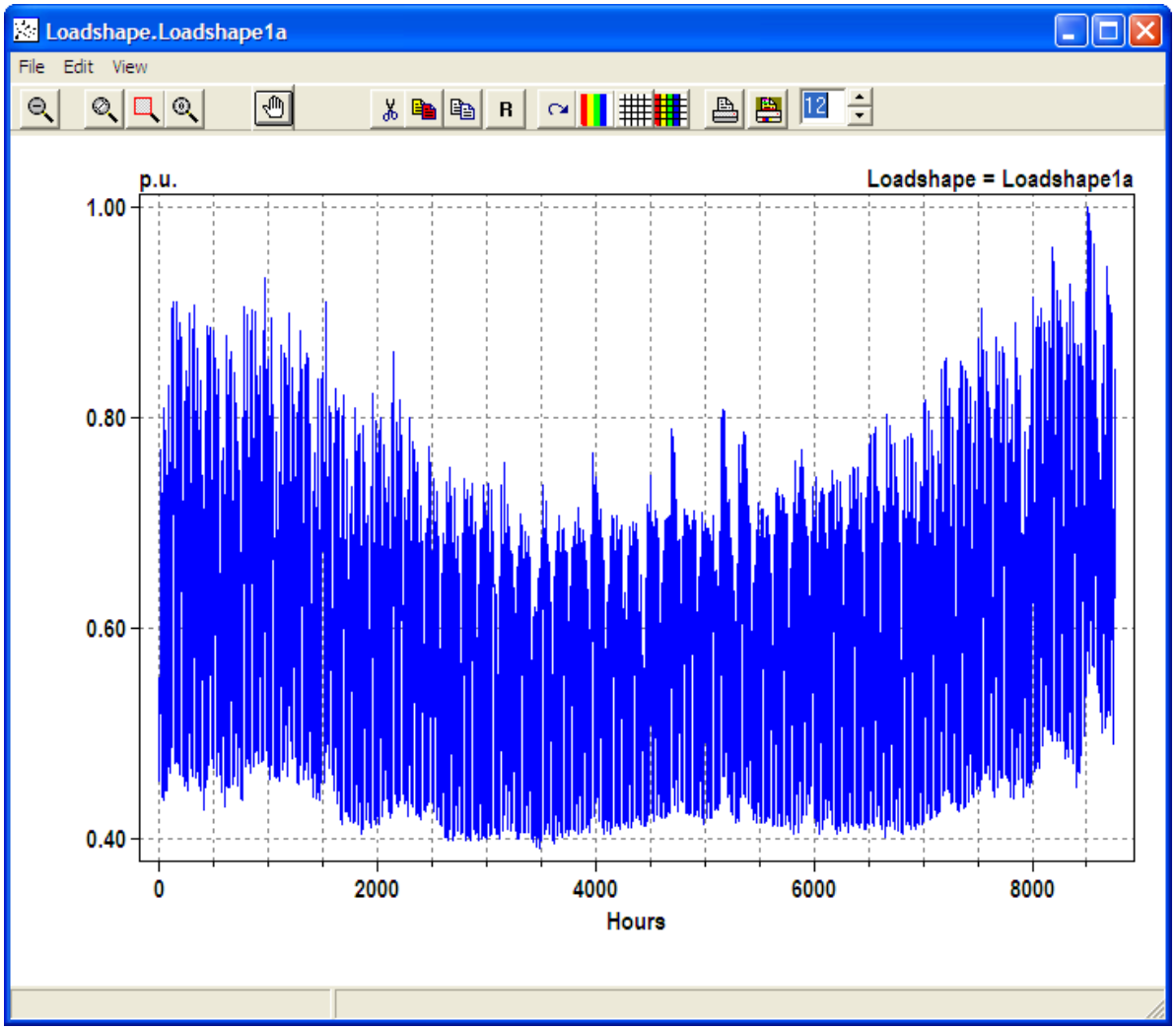
- ! Example scripts for loading and plotting loadshapes out of the loadshape library

- ! You have to have a circuit defined to load in loadshapes.
- New Circuit.LoadshapeExamples

- ! directly ...
- New "LoadShape.LoadShape1a" npts=8760 interval=1.0 mult=(File=LoadShape1.csv)
- Plot Loadshape Object=Loadshape1a  ! execute this to prove you got it

- ! or using Redirect
- Redirect Loadshape1.DSS  ! Load in Loadshape 1
- Plot Loadshape Object=Loadshape1

# Example Yearly LoadShape



# Loadshape Interpolation

- The OpenDSS LOADSHAPE class uses two different types of interpolation depending on it is defined
- Fixed interval data.
  - Default. INTERVAL property defaults to 1 hour.
  - You can set it to another value or to 0.
    - The SINTERVAL and MINTERVAL properties facilitate defining intervals in second or minutes.
  - INTERVAL > 0: fixed interval data
    - CSV files --one numeric value per line.
    - Interpolation algorithm assumes the value REMAINS CONSTANT over the entire interval
    - The HOUR array property is ignored

# Loadshape Interpolation, Cont'd

- For LINEAR INTERPOLATION between the points, define INTERVAL=0.
  - Then both the time and multiplier values for the loadshape using the HOUR, MULT, and QMULT array properties.
- Alternatively, you may use the CSVFILE, DBLFILE, or SNGFILE properties.
  - Enter both the time in hours and the multiplier values.
    - A CSV file would have two values per line separated by a comma or whitespace.
- The variable interval interpolation could be a little bit slower than the fixed interval data because there is more work to do to compute the factor.

# Importing Packed Binary Files

- For simulations, such as AMI, required large volumes of Loadshapes to be imported, using packed binary files can save time
- Standard CSV or TXT file
  - Mult=[file=myfile.txt]
- File of doubles
  - Mult=[dblfile=myfile.dbl]
- File of singles
  - Mult=[sngfile=myfile.sng]

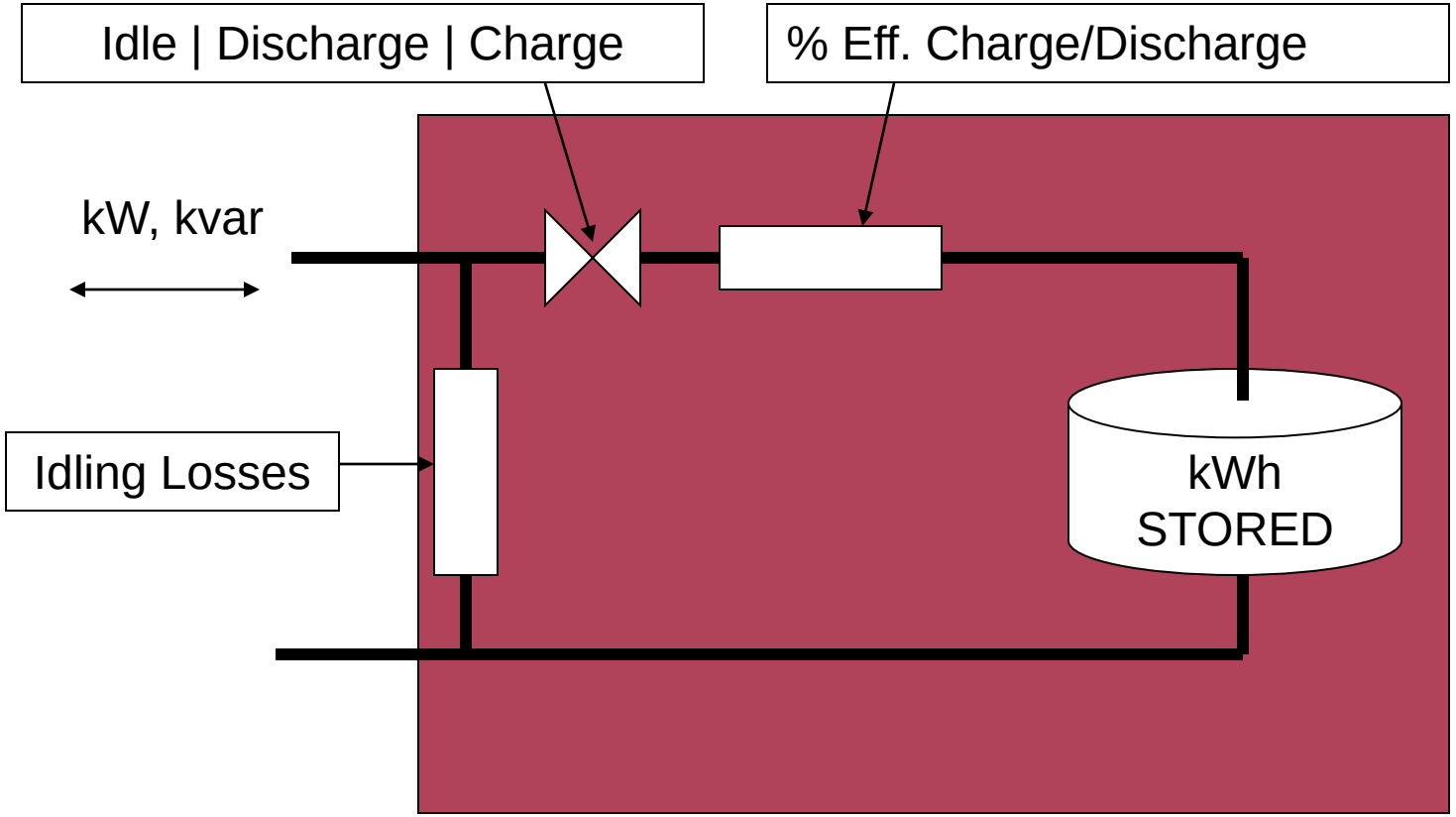
# Recent (2011) Enhancements to Defining Array Properties Using CSV files

- **Syntax:**
  - **mult=[File=myMultiColumnFile.CSV, Column=n, Header=Yes/No]**
- Allows use of multicolumn CSV files with a single header row.
- Example
- `New Loadshape.Ramp2 npts=4000 sInterval=1 mult=(file=MultiChannelTest.csv, column=3, header=yes)`
  - Imports the 3rd column from the file, skipping the header row

EPRI ELECTRIC POWER RESEARCH INSTITUTE

# Example STORAGE Simulations with Load Following Triggers

# Storage Element Model in OpenDSS

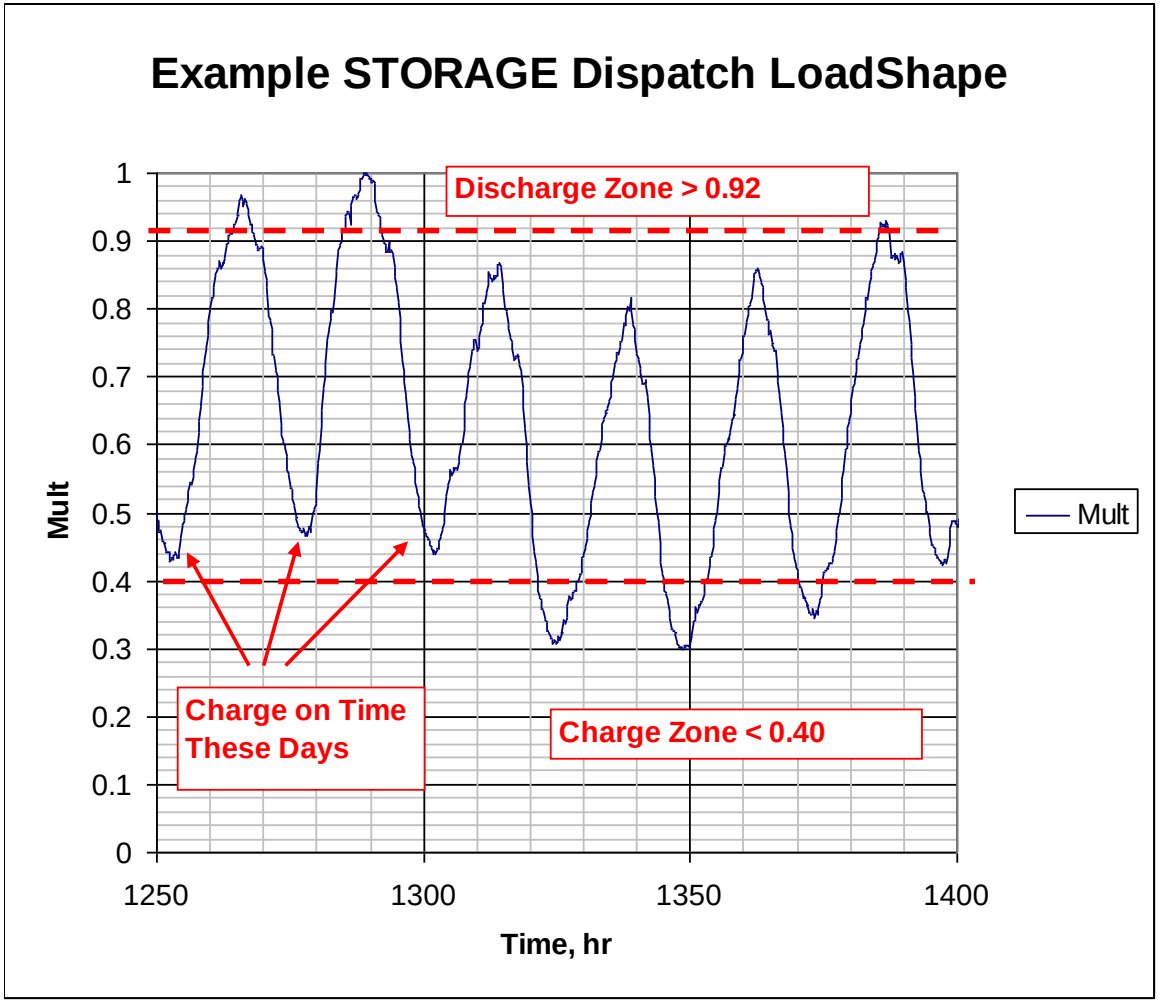



Other Key Properties

% Reserve
kWhRated
kWhStored
%Stored
kWRated

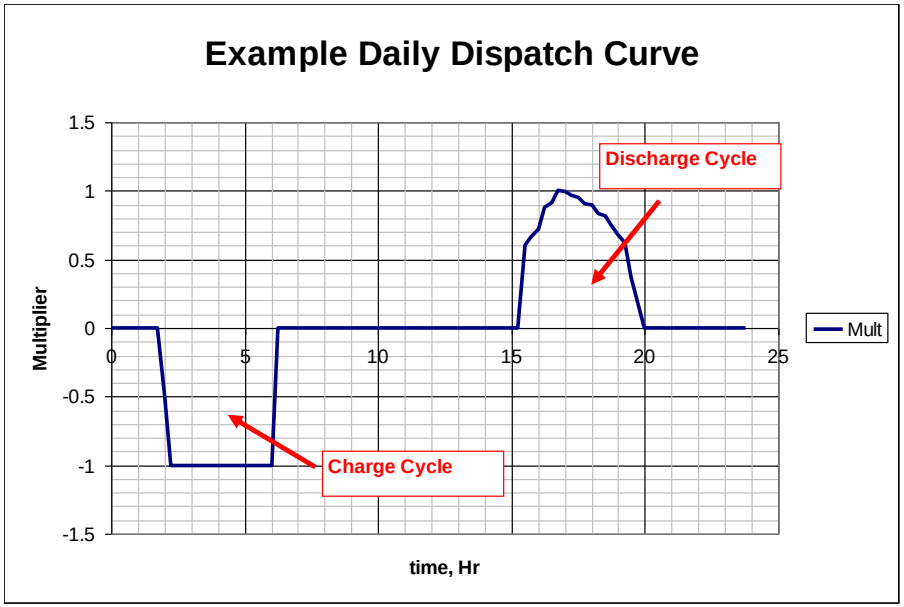
etc.

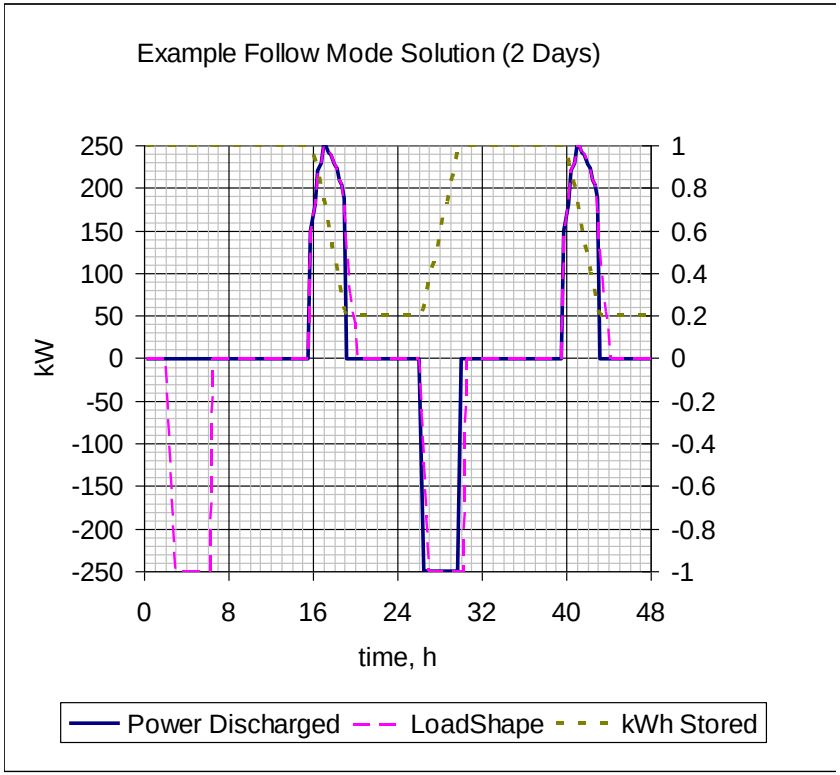
# Basic Charge/Discharge Model for Storage Model



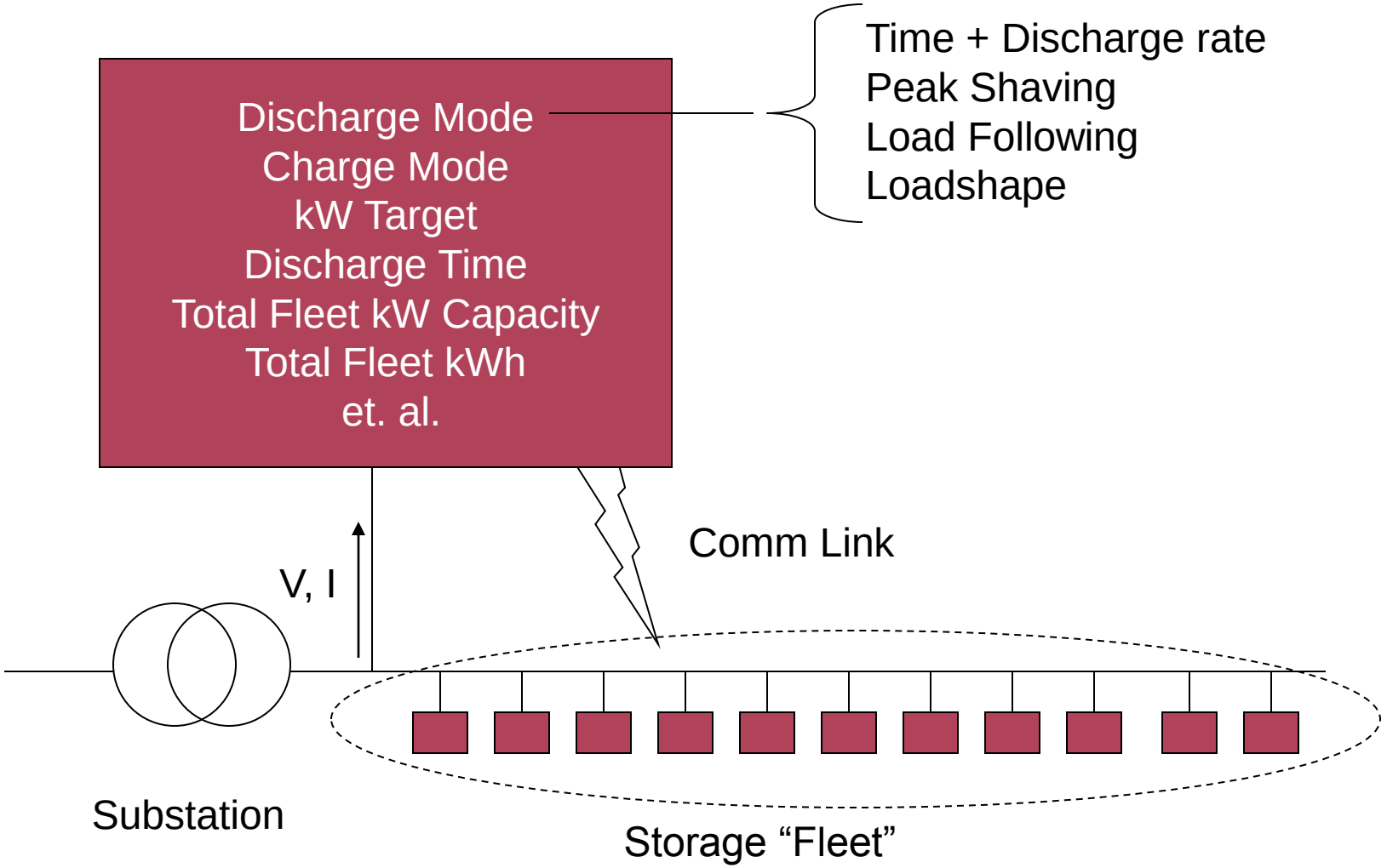
# Follow Mode

Charge/Discharge driven proportionately to a predefined curve



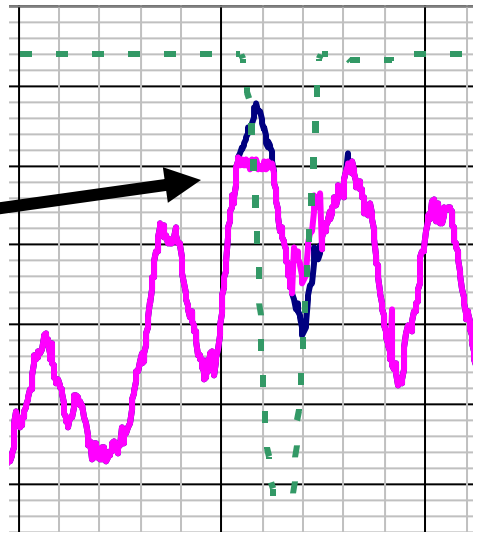



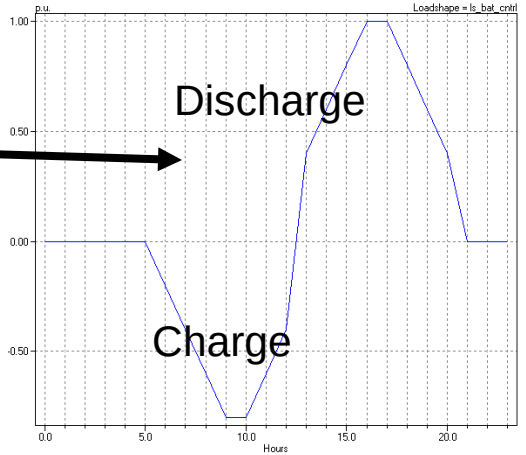
# StorageController Element in OpenDSS (CES Hub)

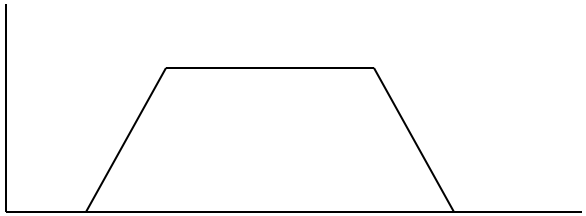


# Hub Dispatch Modes Defined for AEP Demo

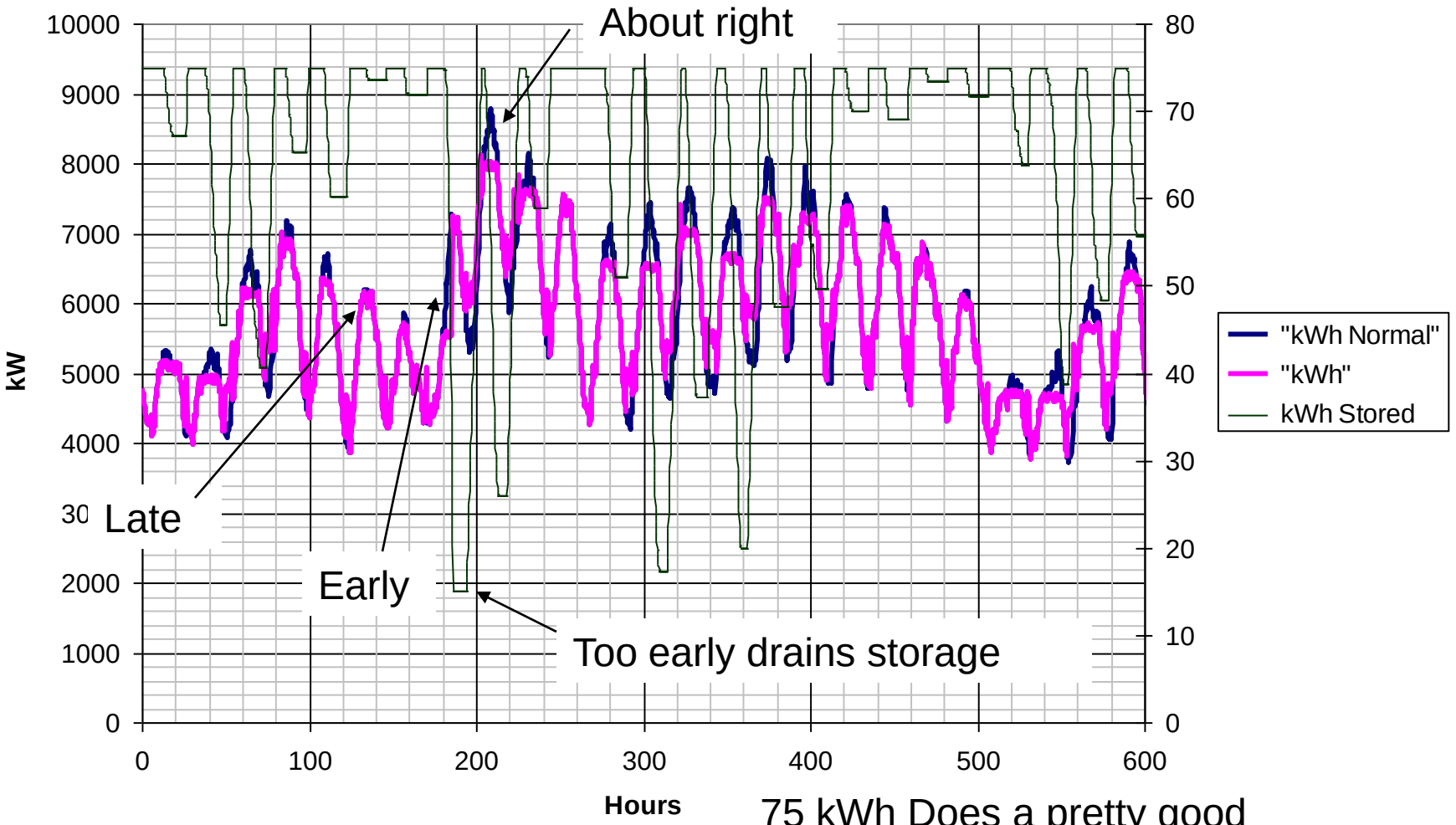
- **Time**
  - Turn ON at specific time
- **Peak Shave**
  - Limit to a value
- **Follow**
  - Time trigger and then shave
- **Loadshape**
- **Schedule**





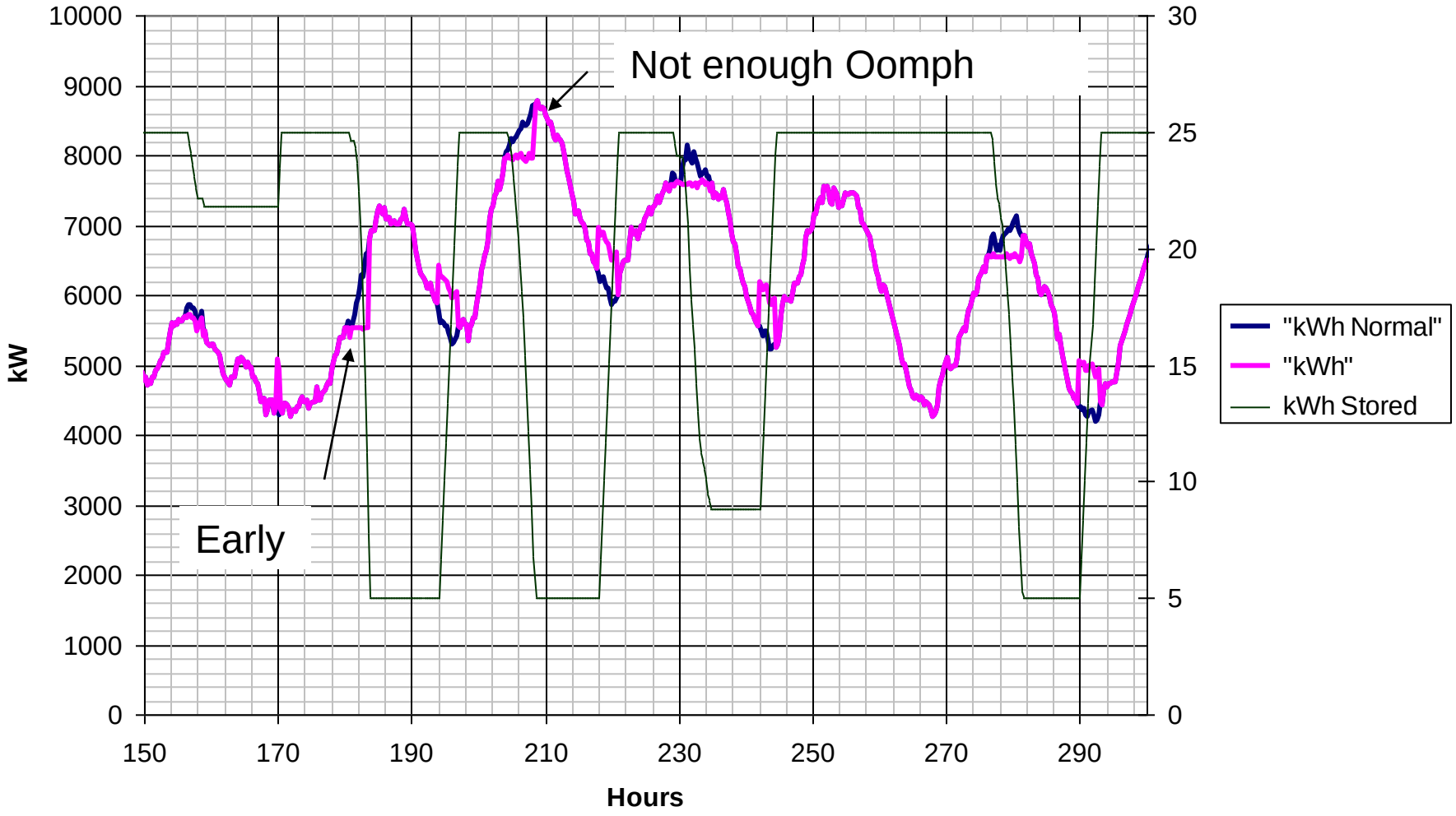



## Load Shapes With and Without Storage
## Discharge Trigger @ Noon, 75 kWh Storage, 30% charge @ 2AM



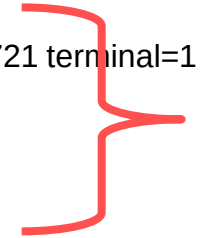

75 kWh Does a pretty good job of clipping the peaks unless triggered too early

## Load Shapes With and Without Storage
## Discharge Trigger @ Noon, 25 kWh Storage, 30% charge @ 2AM



# Run Script for Storage Simulation

```
Compile  Master.DSS
Redirect AllocateLoadsandMeters.DSS

BusCoords BUSCOORDS.CSV
BusCoords buscoordsCES.DSS      !  COORDINATES OF CES LOCATIONS
Set maxcontroliter=20

! ****** ADD STORAGE **************************
redirect CES.DSS
Redirect Set_For_75kWh.DSS

! DEFINE STORAGE CONTROLLER

New StorageController.CESmain  element=line.568_4921721 terminal=1
  ~ kWTarget=7500  PFTarget=0.98
  ~ %ratecharge=30
  ~ eventlog=y
  ~ modedischarge=follow

! SPECIAL MONITORS
New monitor.Store       Storage.jo0211000173 1 mode=1 ppolar=no
New monitor.StoreVars Storage.jo0211000173 1 mode=3

solve
Set Casename=StorageOn75

redirect annualscript.dss
```



**Controller Definition**

# File: AllocateLoadsandMeters.DSS

```
New load.phaseaparallelload kV=7.2 phases=1 bus1=parallelfeedersbus.1 kW=3788.767
   kVAR=1415.917 yearly=phaseaparallel
New load.phasebparallelload kV=7.2 phases=1 bus1=parallelfeedersbus.2 kW=3761.407
   kVAR=1353.185 yearly=phasebparallel
New load.phasecparallelload kV=7.2 phases=1 bus1=parallelfeedersbus.3 kW=3805.81
   kVAR=1356.857 yearly=phasecparallel

New energymeter.feeder element=Line.138_4921721 terminal=1 option=R
   peakcurrent=(359.3,426.25,420.8)

New monitor.FeederPQ element=Line.138_4921721 term=1 mode=1 ppolar=no
New monitor.FeederVI  element=Line.138_4921721 term=1 mode=0 residual=yes

New monitor.660_4921721-1A_PQ element=transformer.660_4921721-1A term=2 mode=1 ppolar=no
New monitor.660_4921721-1B_PQ element=transformer.660_4921721-1B term=2 mode=1 ppolar=no
New monitor.660_4921721-1C_PQ element=transformer.660_4921721-1C term=2 mode=1 ppolar=no

!  redirect AllocationFactors.Txt
allocateloads
```

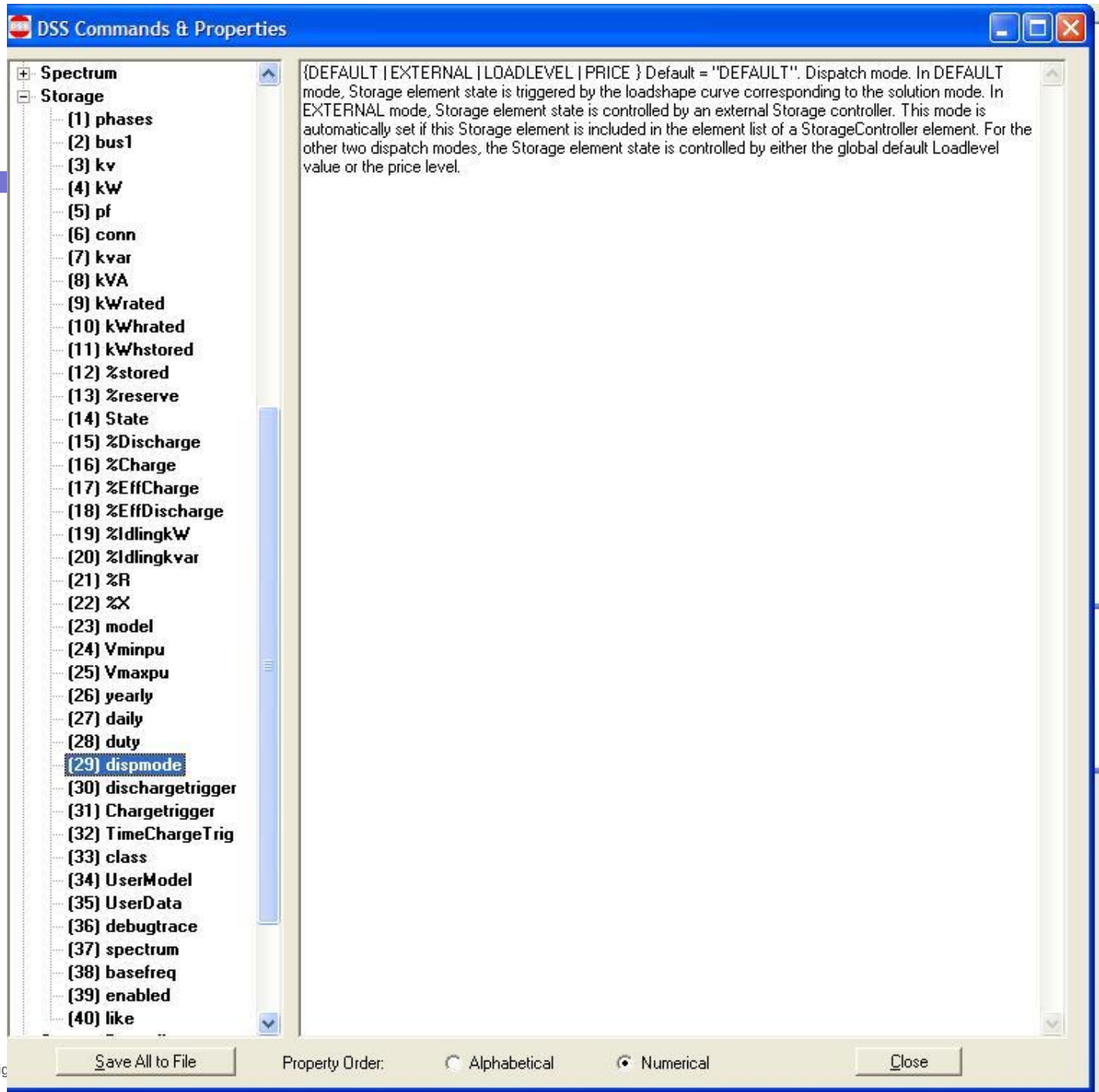
# File: BusCoordsCES.DSS

```
! secondarybus      coord      coord
X_107_4921721_JO0235000030      975232   2491529
X_107_4921721_JO0235000030      975232   2491529
X_107_4921721_JO0235000030      975232   2491529
X_107_4921721_JO0235000030      975232   2491529
X_111_4921721_JO0235000585      977756   2489482
X_119_4921721_JO0211000186      975832   2495110
X_123_4921721_JO0235000080      975519   2491522
X_247_4921721_JO0235000562      978172   2490503
X_247_4921721_JO0235000562      978172   2490503
X_247_4921721_JO0235000562      978172   2490503
X_247_4921721_JO0235000562      978172   2490503
X_247_4921721_JO0235000562      978172   2490503
X_247_4921721_JO0235000562      978172   2490503
X_299_4921721_JO0235000357      976825   2489395
X_328_4921721_JO0211000173      975154   2492117
X_349_4921721_JO0235000323      976528   2490432
X_349_4921721_JO0235000323      976528   2490432
…..
```

# File: Set_For_75kWh.DSS

```
Set Casename=StorageOn75
Storage.jo0235001304.kwhrated=75
Storage.jo0235000257.kwhrated=75
Storage.jo0235000265.kwhrated=75
Storage.jo0235000268_1.kwhrated=75
Storage.jo0235000268_2.kwhrated=75
Storage.jo0235000269_1.kwhrated=75
Storage.jo0235000269_2.kwhrated=75
Storage.jo0235000272_1.kwhrated=75
Storage.jo0235000272_2.kwhrated=75
Storage.jo0235000274.kwhrated=75
Storage.jo0235000583.kwhrated=75
Storage.jo0235001512.kwhrated=75
Storage.jo0235000618.kwhrated=75
Storage.jo0235000593.kwhrated=75
Storage.jo0235000591.kwhrated=75
Storage.jo0235000540_1.kwhrated=75
Storage.jo0235000540_2.kwhrated=75
Storage.jo0235001498.kwhrated=75
Storage.jo0235000570.kwhrated=75
Storage.jo0235000572.kwhrated=75
Storage.jo0235000575.kwhrated=75
….
```

# Storage Element Properties



# Script for a Storage Element Definition (From CES.DSS)

```
New storage.JO0235001304 phases=1
~ bus1=X_68_4921721_JO0235001304.1
~ yearly=Phasealoadshape
~ kV=0.24 kwrated=25
~ pf=1.0
~ kwhrated=25
~ state=IDLING
~ DischargeTrigger=0.8
~ ChargeTrigger=0.6
```

# Simple Annual Simulation Script With Demand Interval Data Capture Turned On

- ! BASIC YEARLY SIMULATION SCRIPT

- ! DO PART OF A YEAR
- set mode=yearly number=2784 stepsize=0.25h

- Set overloadreport=true  ! TURN OVERLOAD REPORT ON
- set voltexceptionreport = true
- set demand=true
- set DIVerbose=true

- Set Year=1
- solve

- closeDI  ! Always do this to close DI files

EPRI ELECTRIC POWER RESEARCH INSTITUTE

# Distribution Efficiency Studies

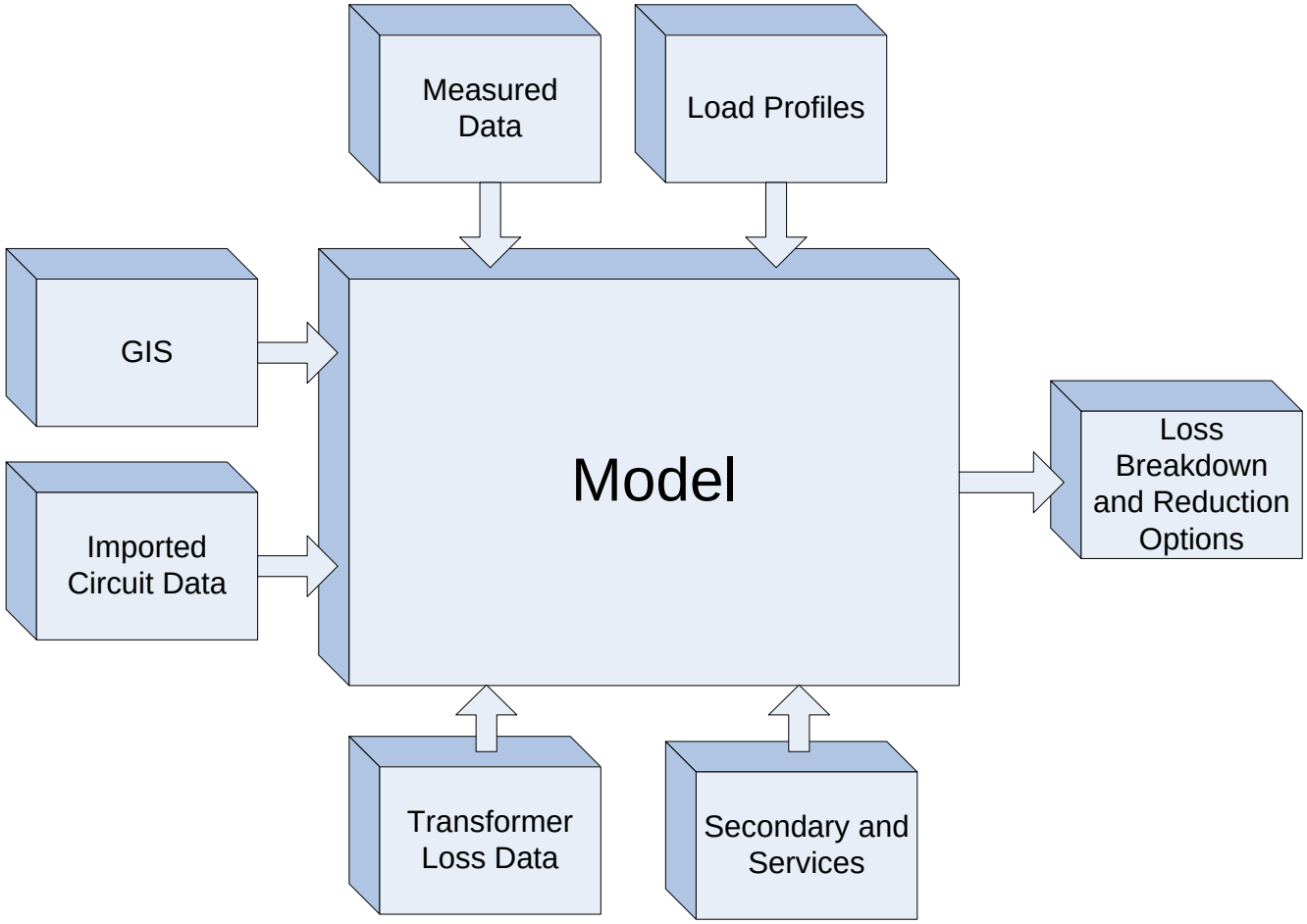
# Modeling/Simulation Analytical Approach

- Develop high-fidelity distribution feeder models
  - Models developed from native electrical model formats and GIS.
  - Substation transformer (Primary lines (3-phase mains and 1-phase laterals)
  - Distribution transformers
  - Secondaries/services
  - Voltage regulation controls (LTCs, regulators, capacitors)
- Spatial and temporal variation of circuit loads
  - Allocated demand to individual customers
  - Annual load shape
- Dynamic simulation of full electrical model serving loads through annual load cycle

# Modeling/Simulation Analytical Approach, cont.

- Distribution system losses are relatively low compared to the total load demand.
- It is challenge to the analyst and the analysis tools to model the system in sufficient detail to accurately quantify the possible savings.

# Model Development Procedure



# Capacitor Controls

- OpenDSS contains many control features for capacitors
  - Current
  - Voltage
  - kvar
  - PF
  - Time
  - etc.
- Capacitor may be operated independently
- Time-Sequencing may be required when multiple capacitors banks are used on a single-feeder
- Multiple control schemes can be used annually

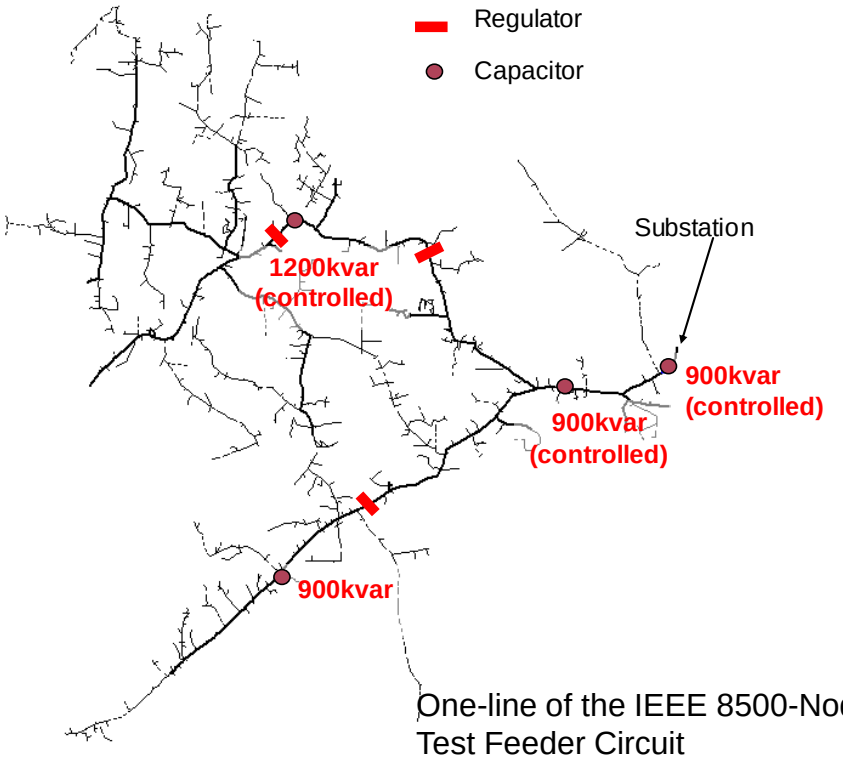

# Per-Phase Capacitor Control

**New Capacitor. CAPBank2 bus1=R20185 kv=12.47 kvar=900 conn=wye** } **Three-Phase Capacitor**

**or**

**New Capacitor.CAPBank2A Bus1=R20185.1 kv=7.2 kvar=300 phases=1 conn=wye**
**New Capacitor.CAPBank2B Bus1=R20185.2 kv=7.2 kvar=300 phases=1 conn=wye**
**New Capacitor.CAPBank2C Bus1=R20185.3 kv=7.2 kvar=300 phases=1 conn=wye**

**Three Single-Phase Capacitors**




One-line of the IEEE 8500-Node Test Feeder Circuit

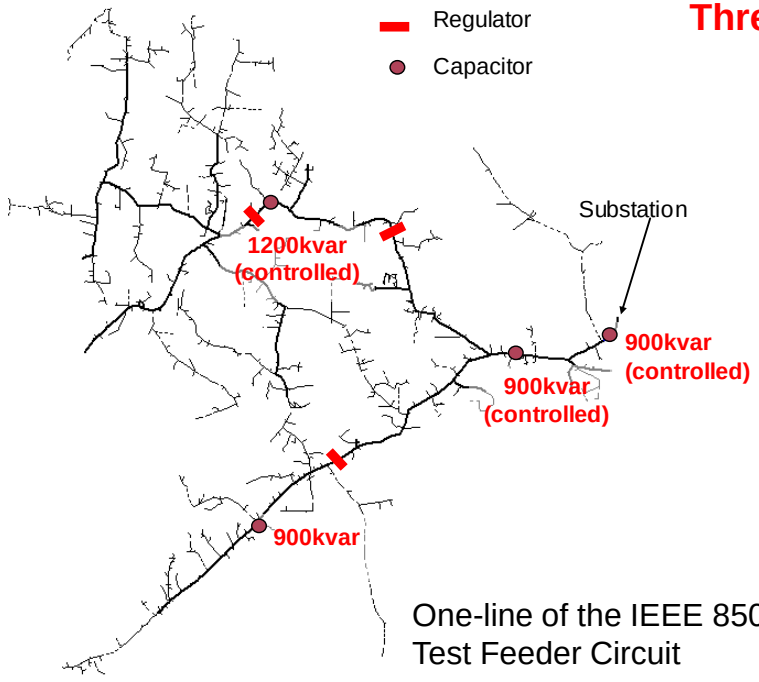
- Three single-phase capacitors defined.
- The controlled capacitor banks monitor each phase separately and operates the respective capacitor on the same phase.

# Per-Phase Capacitor Control, cont.

**New CapControl.CAPBank2A_Ctrl Capacitor=CAPBank2A element=line.CAP_1A terminal=1 type=kvar ptratio=1 ctratio=1**
**~ ONsetting=150 OFFsetting=-225 VoltOverride=Y Vmin=7110 Vmax=7740 Delay=100 Delayoff=100**

**New CapControl.CAPBank2B_Ctrl Capacitor=CAPBank2B element=line.CAP_1B terminal=1 type=kvar ptratio=1 ctratio=1**
**~ ONsetting=150 OFFsetting=-225 VoltOverride=Y Vmin=7110 Vmax=7740 Delay=101 Delayoff=101**

**New CapControl.CAPBank2C_Ctrl Capacitor=CAPBank2C element=line.CAP_1C terminal=1 type=kvar ptratio=1 ctratio=1**
**~ ONsetting=150 OFFsetting=-225 VoltOverride=Y Vmin=7110 Vmax=7740 Delay=102 Delayoff=102**




One-line of the IEEE 8500-Node Test Feeder Circuit

**Three Independently Controlled Single-Phase Capacitors**

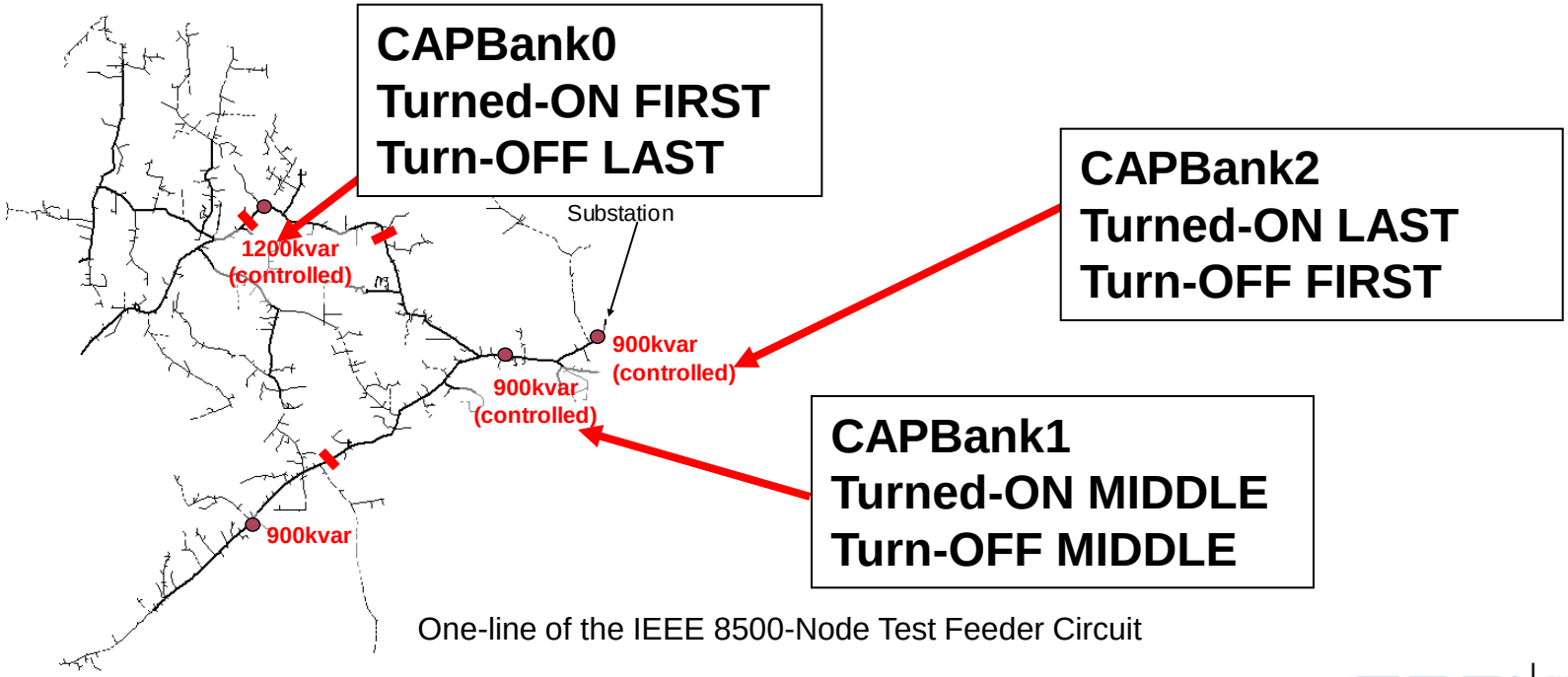
- The capacitor is controlled to switches ON when the reactive power flow in the line is 50% of the capacitor size and switches OFF when the flow is 75% of the capacitor size in the reverse direction.
- Each controlled capacitor also includes voltage override where the capacitor turns ON at 0.9875pu and turns OFF at 1.075pu.

# Capacitor-Control Timing

**New CapControl.CAPBank2A_Ctrl Capacitor=CAPBank2A element=line.CAP_1A terminal=1 type=kvar ptratio=1 ctratio=1 ~ ONsetting=150 OFFsetting=-225 VoltOverride=Y Vmin=7110 Vmax=7740 Delay=100 Delayoff=80**

**New CapControl.CAPBank1A_Ctrl Capacitor=CAPBank1A element=line.CAP_2A terminal=1 type=kvar ptratio=1 ctratio=1 ~ONsetting=150 OFFsetting=-225 VoltOverride=Y Vmin=7110 Vmax=7740 Delay=90 Delayoff=90**

**New CapControl.CAPBank0A_Ctrl Capacitor=CAPBank0A element=line.CAP_3A terminal=1 type=kvar ptratio=1 ctratio=1 ~ONsetting=200 OFFsetting=-300 VoltOverride=Y Vmin=7110 Vmax=7740 Delay=80 Delayoff=100**




One-line of the IEEE 8500-Node Test Feeder Circuit

Sc

# Capacitor Control – Multiple Control Methods

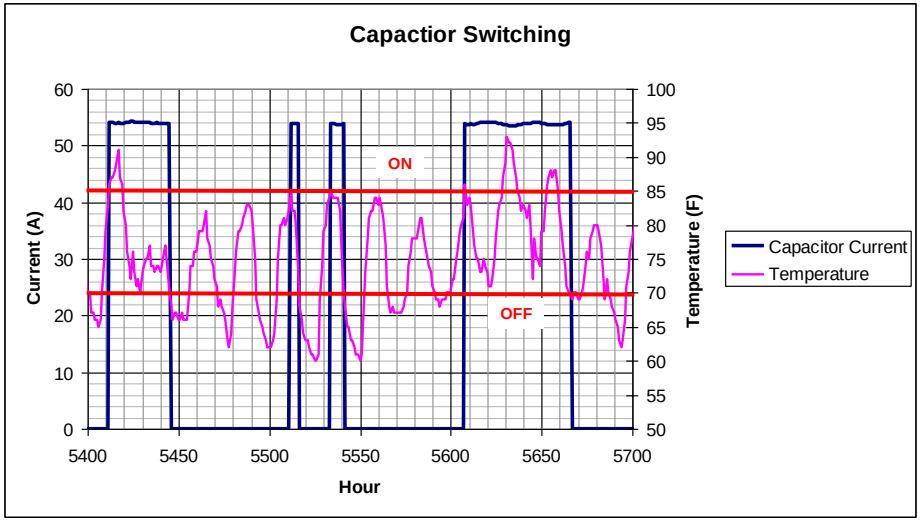
**CapControl.JA-86271_Con_Temp_Summer.Enabled = no**
**CapControl.JA-86271_Con_Temp_Winter.Enabled = yes**
**Solve**

**Set number = 2951 !!!! Next 4 Months**
**CapControl.JA-86271_Con_Temp_Summer.Enabled = yes**
**CapControl.JA-86271_Con_Temp_Winter.Enabled = no**
**solve**

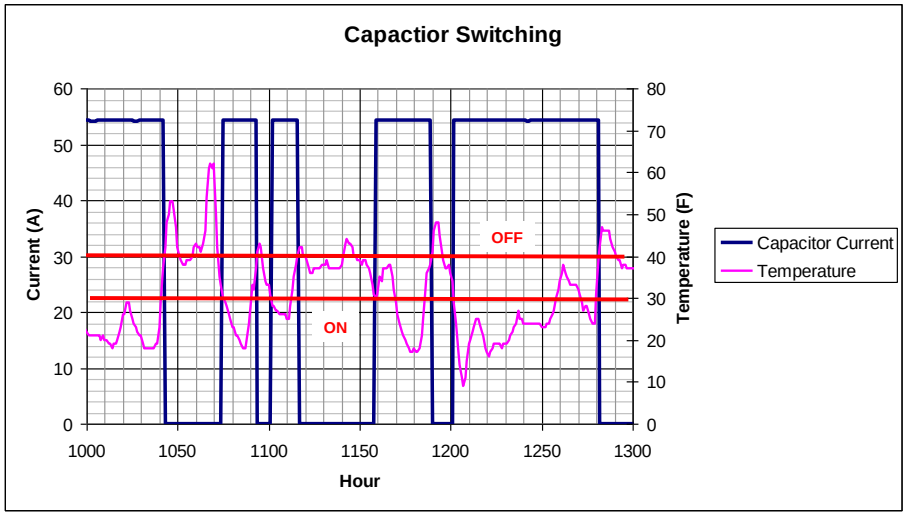
**set number = 2591 !!!! Next 4 Months**
**CapControl.JA-86271_Con_Temp_Summer.Enabled = no**
**CapControl.JA-86271_Con_Temp_Winter.Enabled = yes**
**solve**

**Enable and Disable Capacitor Control**

## Summer Switching (May 15 to September 15)




## Non-Summer Switching (September 15 to May 15)

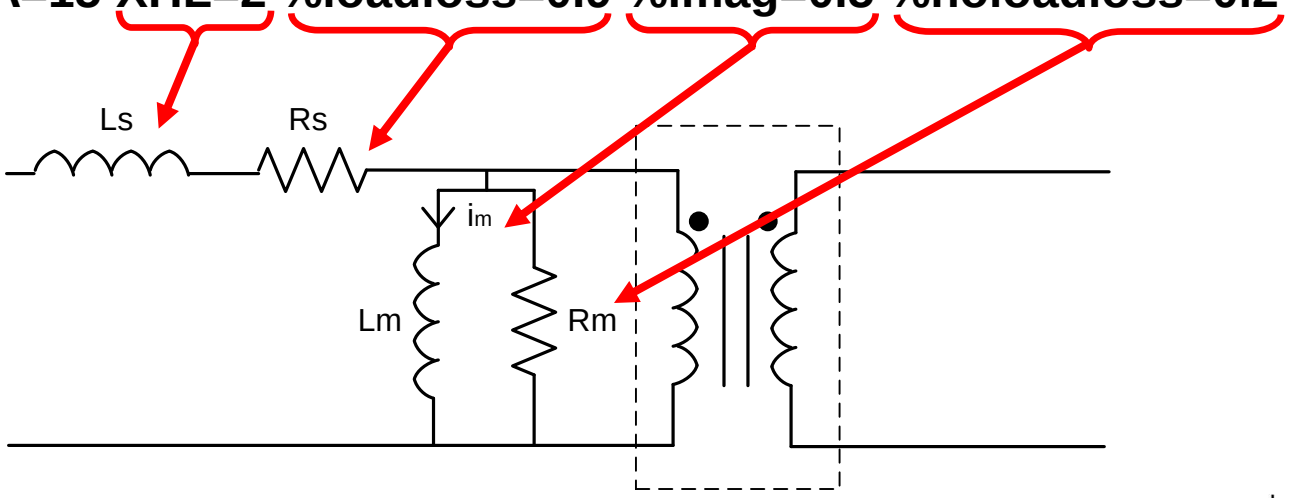


# Modeling Transformers

- Non-ideal transformer modeling
  - Voltage Dependent (Core Loss, Magnetizing Current)
  - Load Dependent (Leakage Reactance, Series Loss)

**!DSS Script:**

**New Transformer.XX phases=1 wdg=1 bus=Y.1 kv=7.2 kVA=15 wdg=2 bus=Z.1 ~ kv=0.24 kVA=15 XHL=2 %loadloss=0.9 %imag=0.5 %noloadloss=0.2**

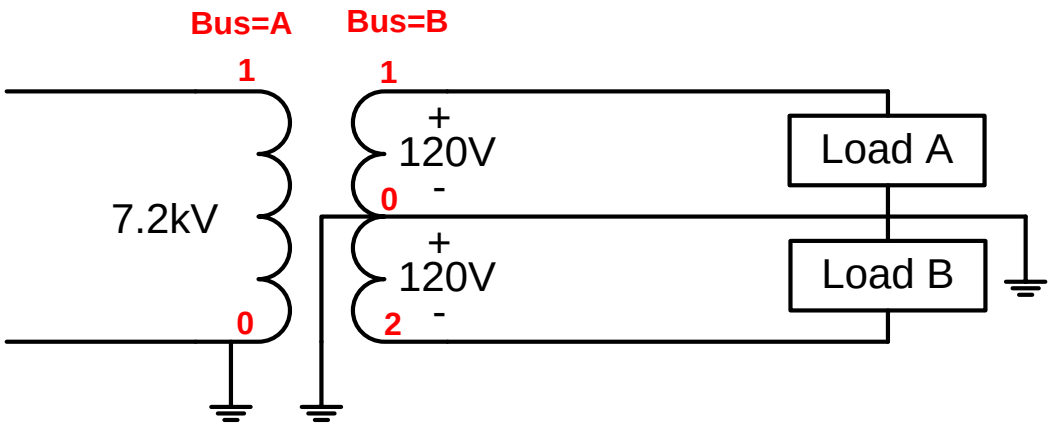


# Modeling Split-Phase Transformers

- OpenDSS allows for a split-phase transformer to be modeled.


**!DSS Script:**

```
New Transformer.D4D2SA phases=1 Windings=3
~Xhl=2.04 Xht=2.04 Xlt=1.36 %noloadloss=0.2
~wdg=1 bus=A.1 kv=7.2 kVA=15 %r=0.5
~wdg=2 bus=B.1.0 kv=0.12 %r=0.6 conn=delta
~wdg=3 bus=B.0.2 kv=0.12 %r=0.6 conn=delta
```




**Note:**
- Xhl: reactance winding 1 to winding 2(H-L)
- Xht: reactance winding 1 to winding 3(H-T)
- Xlt: reactance winding 2 to winding 3(L-T)

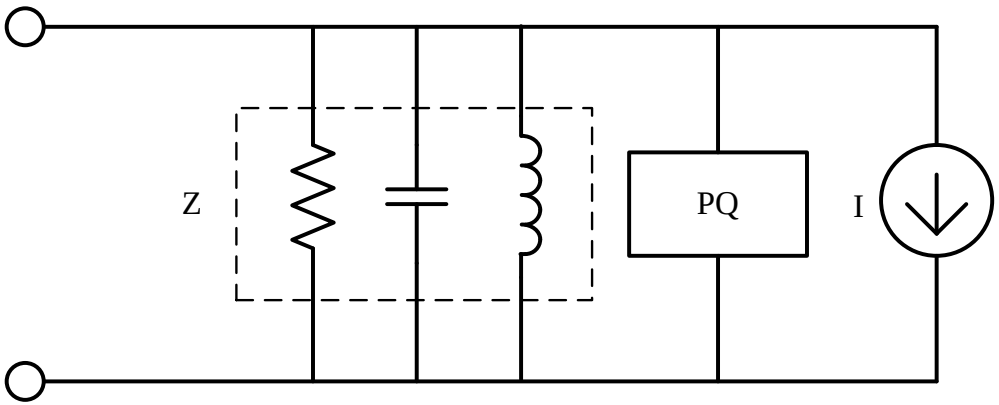
**Warning:**
"%loadloss" overwrites "%r" for windings 1 and 2 only.

# Load Modeling

- Model 4 includes a voltage dependent element which allows for the study of voltage optimization.

**!DSS Script:**

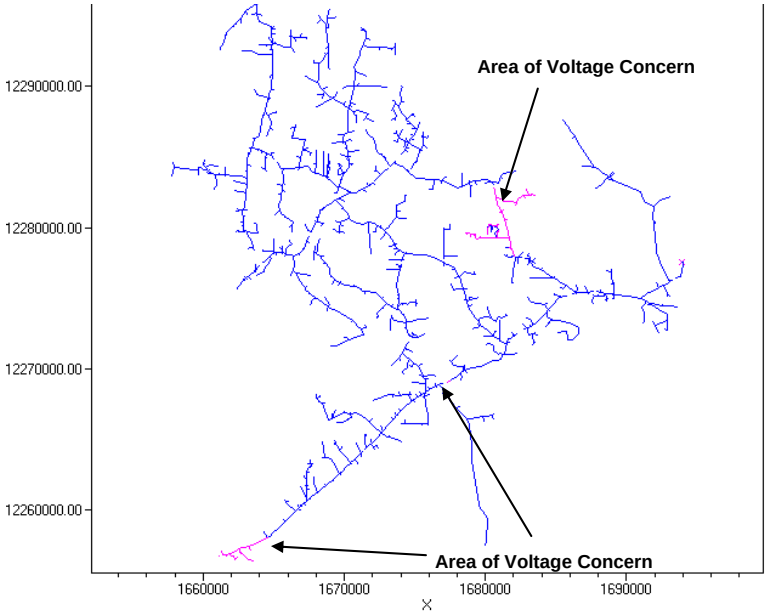
**New load.X_1 phases=1 bus1=1_sec_x.2 kV=0.24 xfkVA=50 pf=0.91 status=variable ~ model=4 CVRwatts=0.8 CVRvars=3 conn=wye yearly=LoadshapeB**




Approximate Equivalent Circuit

CVRw factor = %ΔP / %ΔV
Where:
%ΔP = Change in Power
%ΔV = Change in Voltage

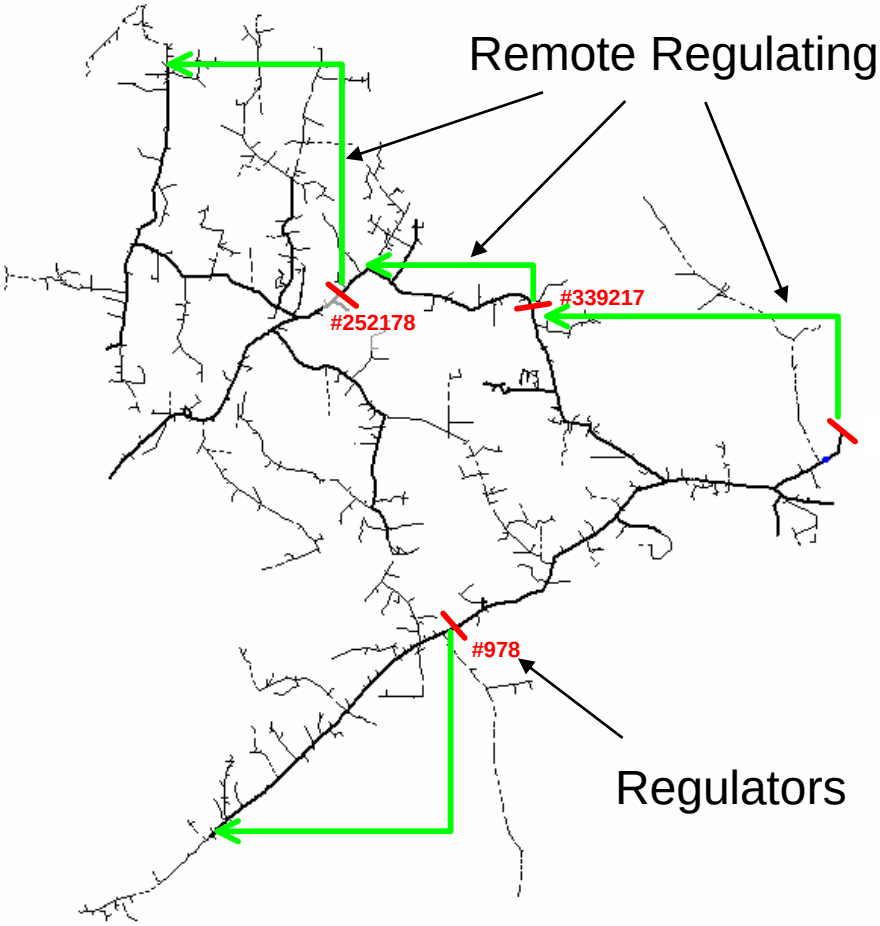
# Voltage Regulation Modeling




**!DSS Script:**

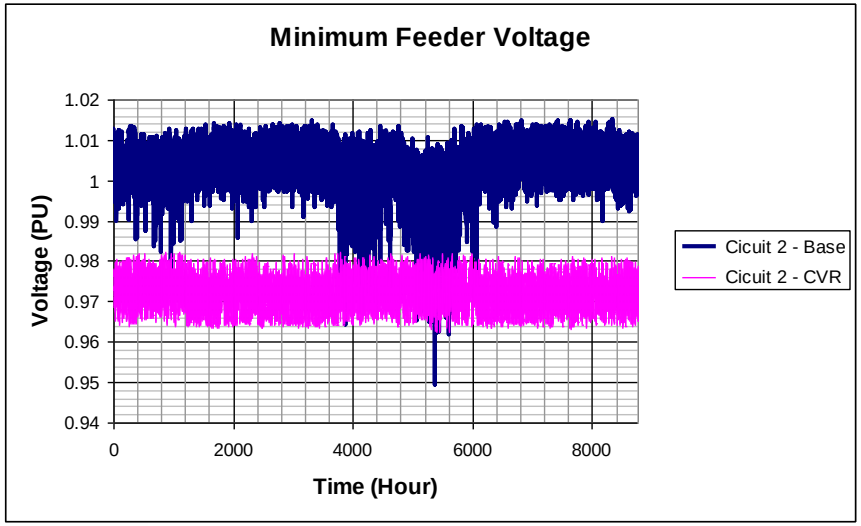
```
New Transformer.VREG2_A phases=1  windings=2  buses=(regxfmr_190.1, 190.1)  conns=(wye, wye)  kvs=(7.2, 7.2)
~ kvas=(10000, 10000)  xhl=0.001 %loadloss=.01 Wdg=2 Maxtap=1.1 Mintap=0.9 ppm=0
New Transformer.VREG2_B phases=1  windings=2  buses=(regxfmr_190.2, 190.2)  conns=(wye, wye)  kvs=(7.2, 7.2)
~ kvas=(10000, 10000)  xhl=0.001 %loadloss=.01 Wdg=2 Maxtap=1.1 Mintap=0.9 ppm=0
New Transformer.VREG2_C phases=1  windings=2  buses=(regxfmr_190.3, 190.3)  conns=(wye, wye)  kvs=(7.2, 7.2)
~ kvas=(10000, 10000)  xhl=0.001 %loadloss=.01 Wdg=2 Maxtap=1.1 Mintap=0.9 ppm=0

New RegControl.VREG2_A transformer=VREG2_A  winding=2 vreg=125  ptratio=60 band=2
New RegControl.VREG2_B transformer=VREG2_B  winding=2 vreg=125  ptratio=60 band=2
New RegControl.VREG2_C transformer=VREG2_C  winding=2 vreg=125  ptratio=60 band=2
```

# Voltage Regulation – Remote Monitoring




One-line of the IEEE 8500-Node Test Feeder Circuit

- Remote regulating can be used in control voltage
- Define “Bus= ” in RegControl to monitor remote bus.
- This remote regulation can maintain voltage in areas of concern when CVR is implemented

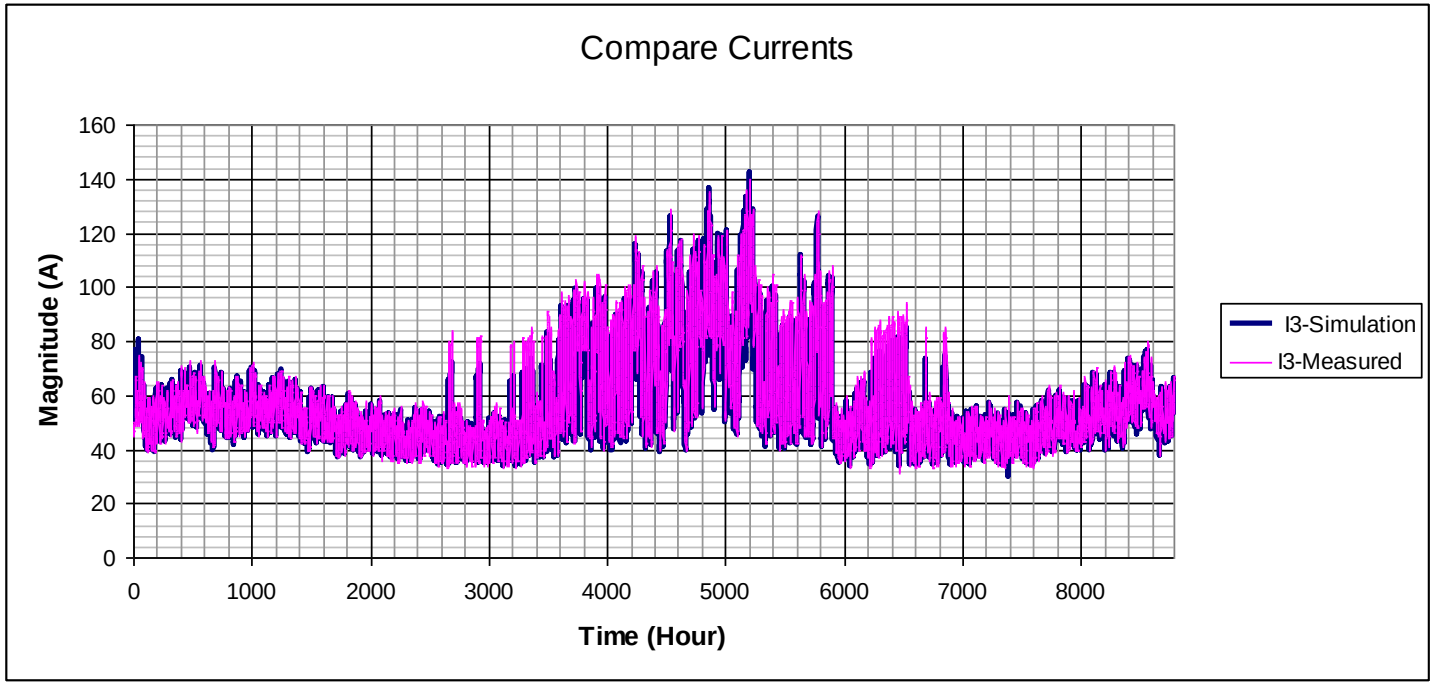


# Annual Loadshape

- Each Loadshape is saved to csv file and defined in each load

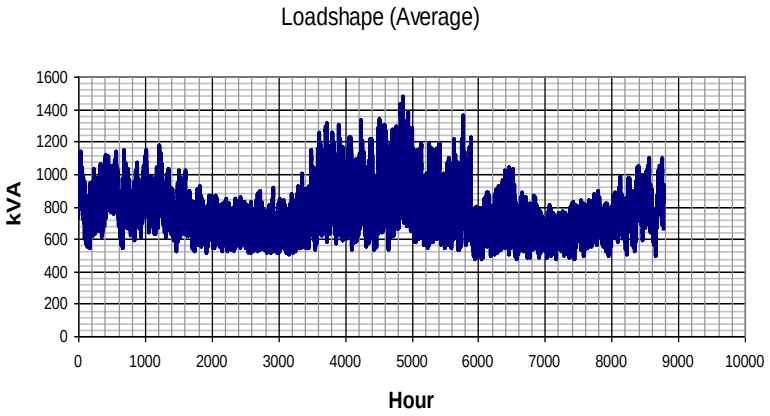
**!DSS Script:**

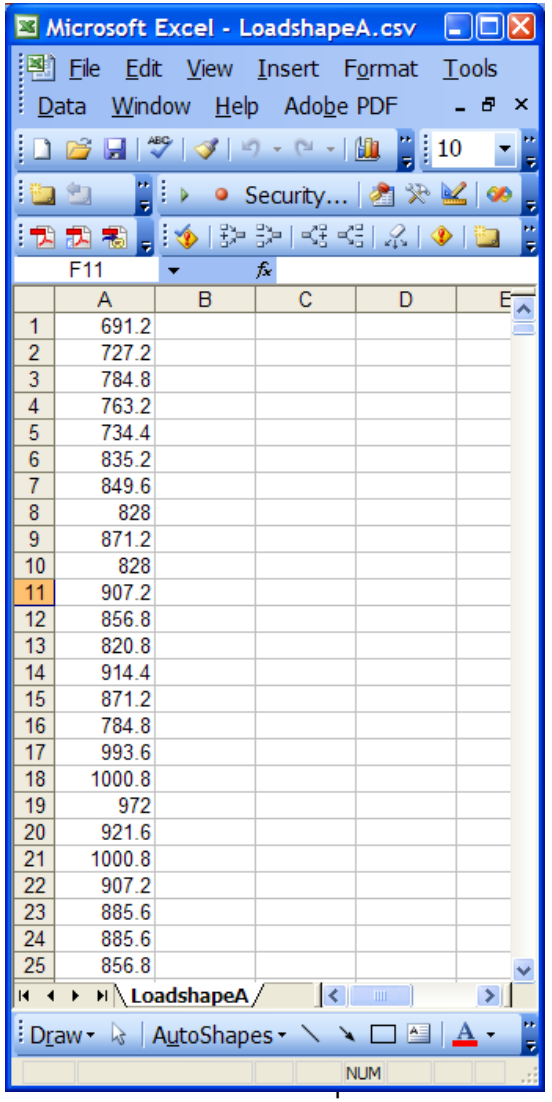
**New load.X_1 phases=1 bus1=1_sec_x.3 kV=0.24 xfkVA=50 pf=0.91 status=variable ~ model=4 CVRwatts=0.8 CVRvars=3 conn=wye yearly=LoadshapeC**



# Annual Loadshape

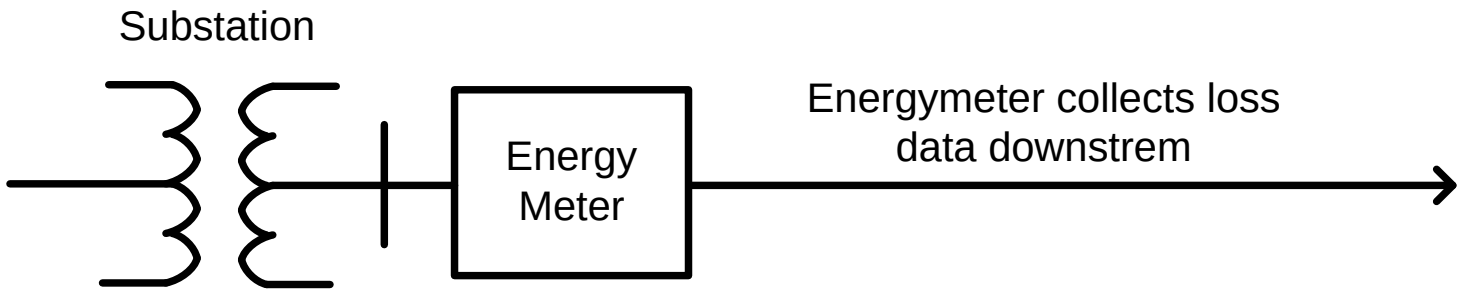
```
!!!Define Energy Meter
New EnergyMeter._EM  element=Line.XFMR_CONN terminal=2 option=R
~action=C peakcurrent=(201,207,208)

!!!!Define Load Shape
New Loadshape.LoadshapeA npts=8784 interval=1 mult=(file=LoadshapeA.csv)
~action=normalize
New Loadshape.LoadshapeB npts=8784 interval=1 mult=(file=LoadshapeB.csv)
~action=normalize
New Loadshape.LoadshapeC npts=8784 interval=1 mult=(file=LoadshapeC.csv)
~action=normalize
New Loadshape.LoadshapeAVG npts=8784 interval=1
~ mult=(file=LoadshapeAVG.csv) action=normalize

!!!!Allocate Loads
Compile (C:\Projects\Green Circuits\Master.dss)
AllocateLoads
Dump Alloc
Solve
```



# Loss Breakdown

- Energymeter used to capture loss results:
  - Compute Line losses and segregate by 3-phase and other (1- and 2-phase) line losses.
  - Compute Line losses
  - Compute Zone losses.
  - Compute Sequence losses in lines and segregate by line mode losses and zero mode losses.
  - Compute losses and segregate by voltage base.
  - Compute Transformer losses.
  - Report Overvoltages
  - Report Undervoltages
  - Report Overloads

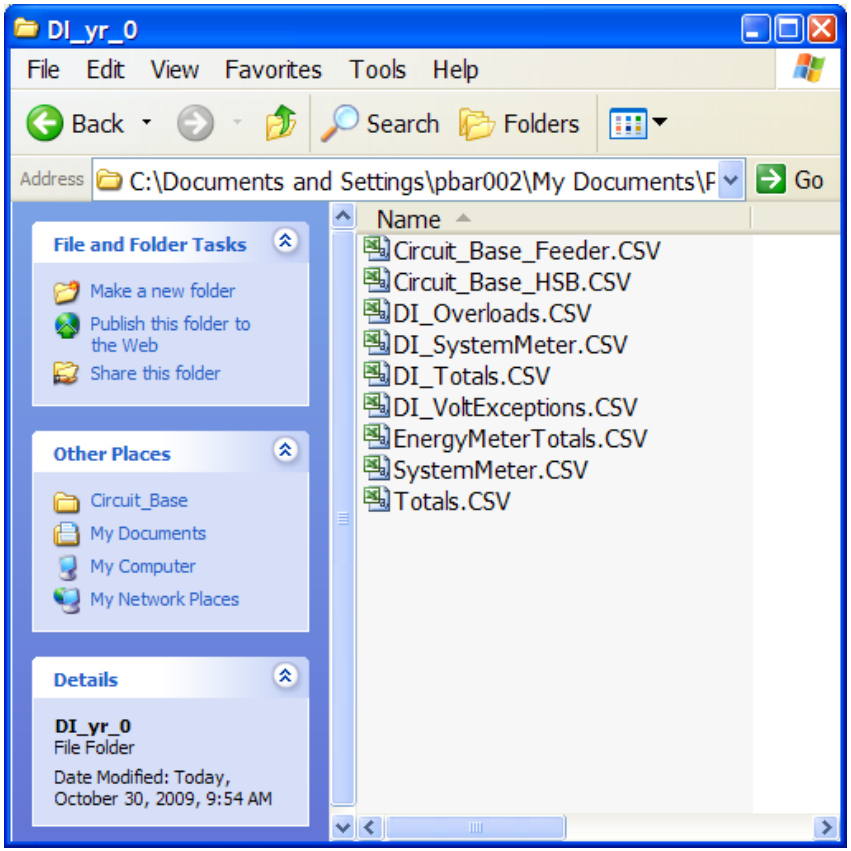


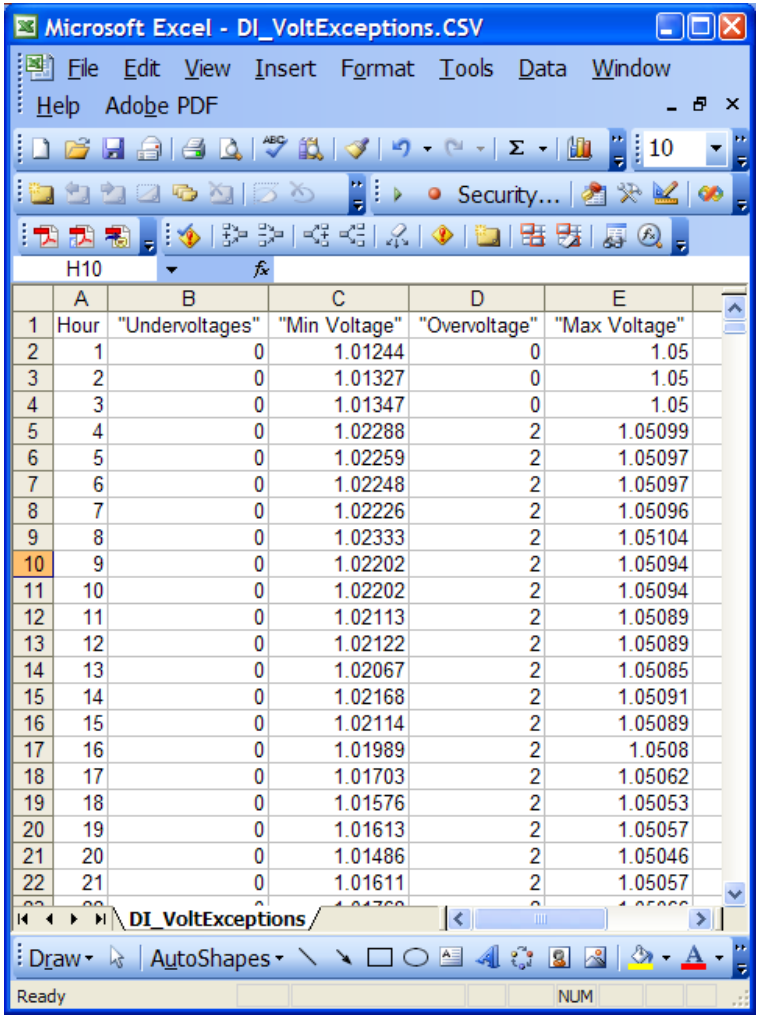
# Energymeter Registers

- Registers:
- Reg 1 = kWh
- Reg 2 = kvarh
- Reg 3 = Max kW
- Reg 4 = Max kVA
- Reg 5 = Zone kWh
- Reg 6 = Zone kvarh
- Reg 7 = Zone Max kW
- Reg 8 = Zone Max kVA
- Reg 9 = Overload kWh Normal
- Reg 10 = Overload kWh Emerg
- Reg 11 = Load EEN
- Reg 12 = Load UE
- Reg 13 = Zone Losses kWh
- Reg 14 = Zone Losses kvarh
- Reg 15 = Zone Max kW Losses
- Reg 16 = Zone Max kvar Losses
- Reg 17 = Load Losses kWh
- Reg 18 = Load Losses kvarh
- Reg 19 = No Load Losses kWh
- Reg 20 = No Load Losses kvarh
- Reg 21 = Max kW Load Losses
- Reg 22 = Max kW No Load Losses
- Reg 23 = Line Losses
- Reg 24 = Transformer Losses
- Reg 25 = Line Mode Line Losses
- Reg 26 = Zero Mode Line Losses
- Reg 27 = 3-phase Line Losses
- Reg 28 = 1- and 2-phase Line Losses
- Reg 29 = Gen kWh
- Reg 30 = Gen kvarh
- Reg 31 = Gen Max kW
- Reg 32 = Gen Max kVA
- Reg 33 = 12.5 kV Losses
- Reg 34 = 0.415 kV Losses
- Reg 35 = Aux3
- Reg 36 = Aux4
- Reg 37 = Aux5
- Reg 38 = Aux6
- Reg 39 = Aux7
- Etc.

# Energymeter Deposits

C:\Circuit_Base\Circuit_Base_Yearly\DI_yr_0

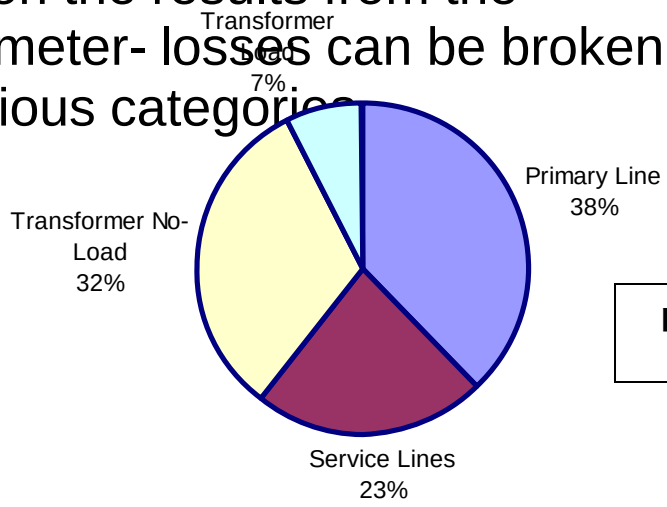


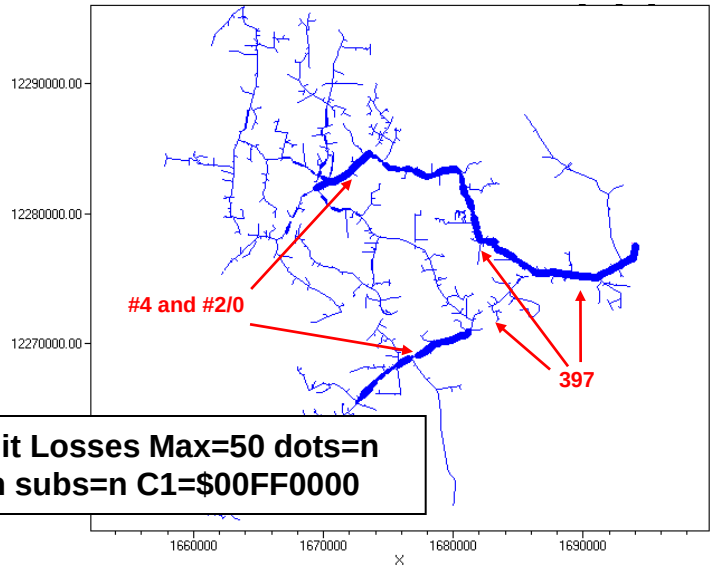

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Hour | "Undervoltages" | "Min Voltage" | "Overvoltage" | "Max Voltage" |
| 2 | 1 | 0 | 1.01244 | 0 | 1.05 |
| 3 | 2 | 0 | 1.01327 | 0 | 1.05 |
| 4 | 3 | 0 | 1.01347 | 0 | 1.05 |
| 5 | 4 | 0 | 1.02288 | 2 | 1.05099 |
| 6 | 5 | 0 | 1.02259 | 2 | 1.05097 |
| 7 | 6 | 0 | 1.02248 | 2 | 1.05097 |
| 8 | 7 | 0 | 1.02226 | 2 | 1.05096 |
| 9 | 8 | 0 | 1.02333 | 2 | 1.05104 |
| 10 | 9 | 0 | 1.02202 | 2 | 1.05094 |
| 11 | 10 | 0 | 1.02202 | 2 | 1.05094 |
| 12 | 11 | 0 | 1.02113 | 2 | 1.05089 |
| 13 | 12 | 0 | 1.02122 | 2 | 1.05089 |
| 14 | 13 | 0 | 1.02067 | 2 | 1.05085 |
| 15 | 14 | 0 | 1.02168 | 2 | 1.05091 |
| 16 | 15 | 0 | 1.02114 | 2 | 1.05089 |
| 17 | 16 | 0 | 1.01989 | 2 | 1.0508 |
| 18 | 17 | 0 | 1.01703 | 2 | 1.05062 |
| 19 | 18 | 0 | 1.01576 | 2 | 1.05053 |
| 20 | 19 | 0 | 1.01613 | 2 | 1.05057 |
| 21 | 20 | 0 | 1.01486 | 2 | 1.05046 |
| 22 | 21 | 0 | 1.01611 | 2 | 1.05057 |

# Reviewing the Data

- Losses can be plotted to view high loss areas
- Based on the results from the Energymeter- losses can be broken down into various categories



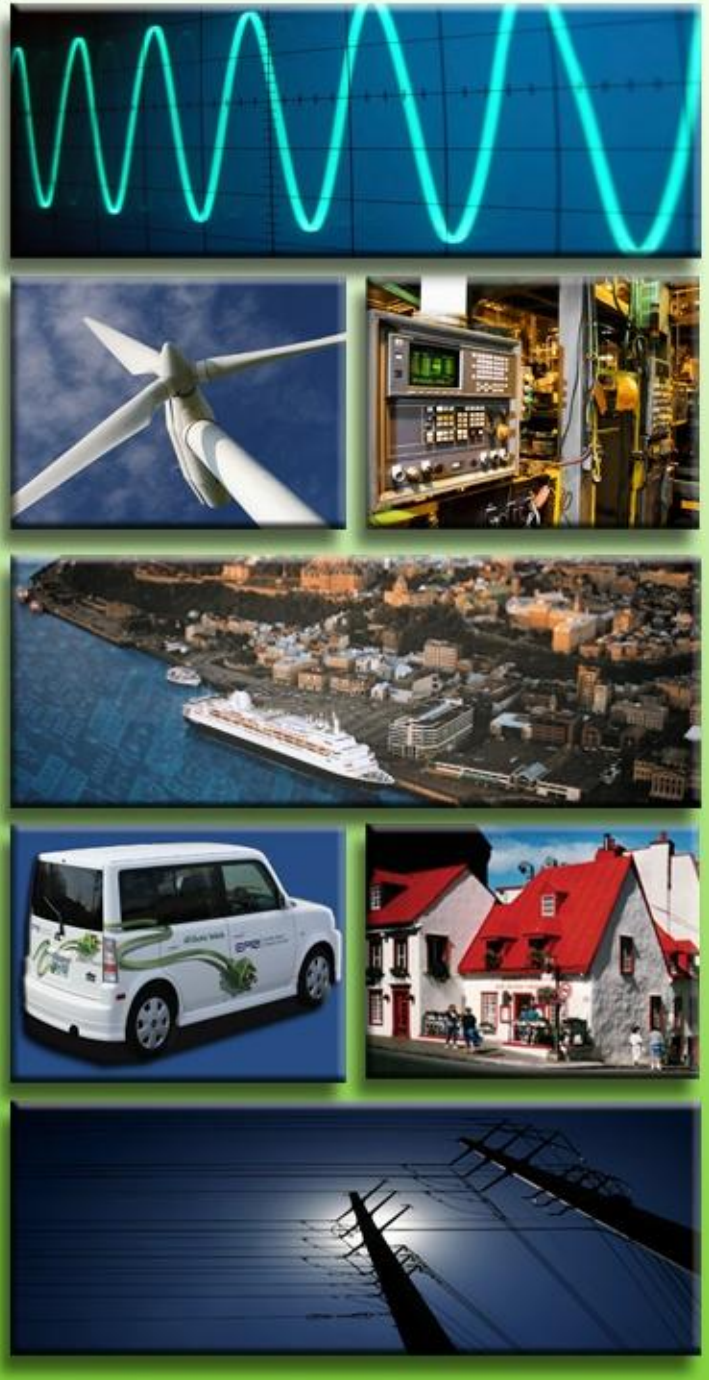

Highest Loss Density (In Bold)




**plot circuit Losses Max=50 dots=n labels=n subs=n C1=$00FF0000**

One-line of the IEEE 8500-Node Test Feeder Circuit

| | Peak Demand | | Annual Energy | |
|---|---|---|---|---|
| | kW | % Peak | kWh | % Consumpt. |
| Consumption/Demand | **4315** | | **20321760** | |
| Total Losses | 163 | 3.77% | 559103 | 2.75% |
| Line Losses | 127 | 2.94% | 339313 | 1.67% |
| Xfmr Losses | 36 | 0.83% | 219790 | 1.08% |
| Load Losses | 143 | 3.32% | 380592 | 1.87% |
| No-Load Losses | 20 | 0.45% | 178511 | 0.88% |
| Primary Losses | 113 | 2.61% | 431413 | 2.12% |
| Secondary Losses | 50 | 1.16% | 127691 | 0.63% |



EPRI ELECTRIC POWER RESEARCH INSTITUTE

# Intrinsic Simulation Modes

Discussion of the various solution modes built into the OpenDSS

# Built-in Solution Modes

- Snapshot (static) Power Flow
- Direct (non-iterative)
- Daily mode (default: 24 1-hr increments)
- Yearly mode (default 8760 1-hr increments)
- Duty cycle (1 to 5s increments)
- Dynamics (electromechanical transients)
- Fault study
- Monte carlo fault study
- Harmonic
- Custom user-defined solutions

# Snapshot Mode

- This is the default solution mode after creating a new circuit model
- The basic power flow solution mode for the OpenDSS
- Performs a single solution each time Solve is issued
- Control mode defaults to “Static”
- Defaults to iterative power flow solution
  - Load model is “Powerflow”

# Direct Mode

- Performs a non-iterative solution of the present system admittance equation
- Loads and Generators represented by constant impedance
  - Value in their Norton equivalents

- This is a useful mode when convergence is difficult, but needed to perform Harmonic, Dynamic, or Faultstudy mode solution.
  - Will nearly always give a reasonable starting point for these solution modes

# Daily Mode

- Defaults
  - Stepsize = 1.0 h
  - Controlmode = static
  - Number = 24

- Designed for simulations of a load profile for one day for distribution planning studies, losses, other energy studies
- Time step of 0.25 .. 1.0 h typically
- One or more days (change number)
- Set the Daily=  property of Load and Generator to follow a loadshape
- Similar functionality to Yearly mode

# Yearly Mode

- Defaults
  - Stepsize = 1.0 h
  - Controlmode = static
  - Number = 8760

- Designed for annual simulations for distribution planning studies, losses, other energy studies
- Time step of 0.25 .. 1.0 h typically
- All or part of year
- Set the Yearly=  property of Load and Generator to follow a loadshape

# DutyCycle Mode

- Defaults
  - Stepsize = 1.0 s
  - Controlmode = time
  - Number = 100

- Designed for simulations of wind turbine output, solar PV output, stone crushers, and other impulse loads
- Time step of 1 – 10 s
- Duration of a few minutes up to 1 day
- Set the Duty=  property of Load and Generator to follow a loadshape

# Dynamics Mode

- Defaults
  - Stepsize = 0.001 s
  - Controlmode = time
  - Number = 100

- Electromagnetic transients simulations
- Generator state variables integrated
  - Predictor-corrector
- Intended for simulations of a few seconds duration

# FaultStudy Mode

- Defaults to dynamic model so machine contributions to faults are included
- Requires a power flow solution to start
- Solve causes short circuit equivalents to be computed for each bus
- Follow immediately by “Show Faultstudy” report
  - Computes fault currents
- Typical sequence:
  - Solve
  - Solve mode=Faultstudy
  - Show Faultstudy

# Monte Carlo Fault Study Mode

- A special kind of fault study useful for some reliability or power quality studies
- The circuit model is first populated by a selected number of Fault objects.
  - Example: SLG faults at all buses
  - User can distribute faults in any manner
- This solution mode activates the faults one at a time and computes a Snapshot solution with loads modeled as admittances
- Be sure to place Monitors around the circuit where you want to see voltage sag magnitudes, fault currents, etc.
  - Answers will come fast!

# Harmonic Mode

- Defaults
  - Controlmode = OFF
  - Loadmodel = Admittance

- Performs a direct solution at each frequency presently defined in the problem by Spectrum objects.
  - See Harmonics option
- Requires a power flow solution
- Typical sequence
  - Solve
  - Solve mode=harmonics
- Be sure you have Monitor objects defined!

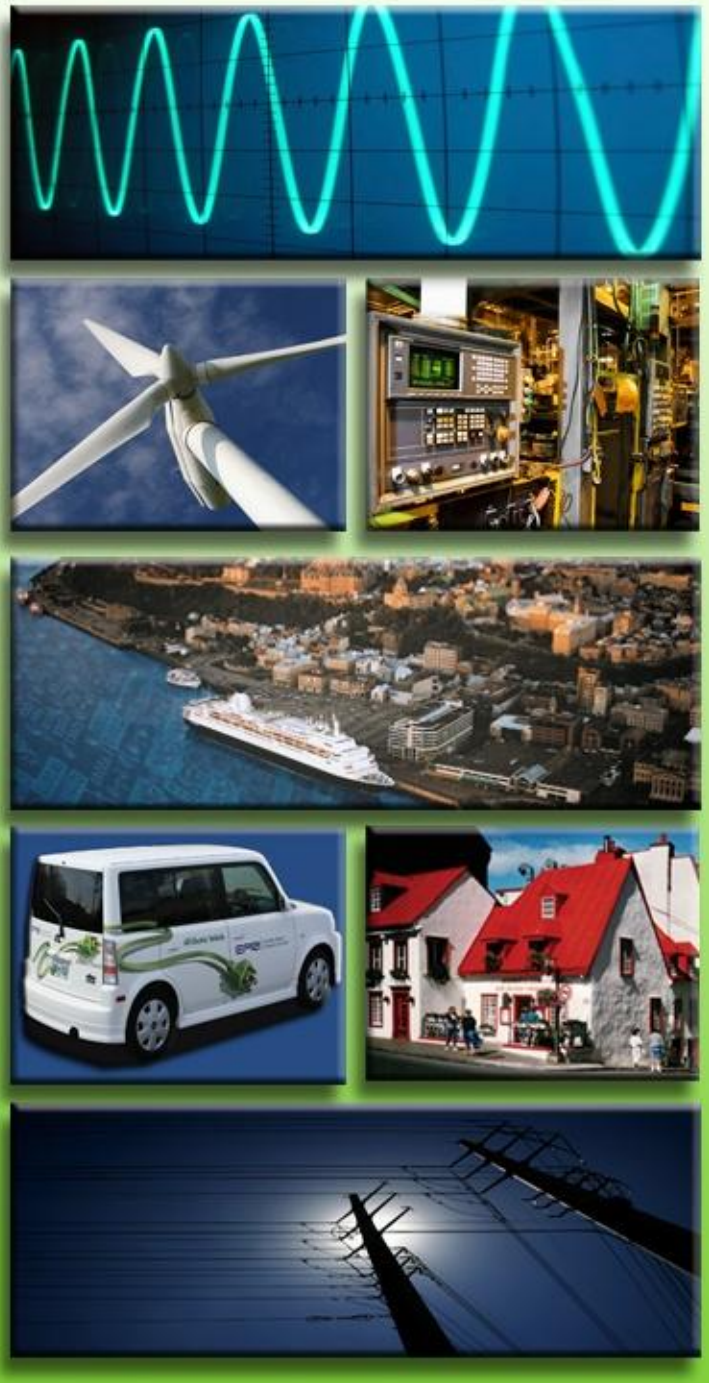
# Control Modes

- Control models are a key feature of OpenDSS and it is important to understand how they work in different solution modes.
- Key Modes
  - OFF
    - Controls are disabled; controlled parameters stay the same as last value
  - TIME
    - Controls operate when solution reaches a specified time. For time steps smaller than control delay
  - STATIC
    - For time steps larger than control delay. Controls operate in sequence of time delay values.
      - Shortest delay operates first
      - Default for Snapshot, Daily, Yearly modes

# Questions?

# Agenda, Day 2

| | |
|---|---|
| 8:30 AM - 9:30 AM | **Custom Simulations**<br>An introduction into how to implement your own solution modes through scripts and an introduction to the Open Distribution System Simulator COM interface |
| 9:45 AM - 11:00 PM | **Using the Open Distribution System Simulator COM Interface**<br>Illustrating the use of the COM interface with an Excel example. This is a class participation example and a laptop with Excel is recommended. |
| 11:00 AM - 12:00 PM | **Other Examples and Wrap-up** |
| 12:00 PM | **Meeting Adjourn** |



EPRI | ELECTRIC POWER RESEARCH INSTITUTE

# Custom Simulations

Discussion of the various solution modes built into the OpenDSS

# Custom Simulation Scripting in Snapshot Mode

- This is the default solution mode
- Attempts one solution for each “solve”
- Solves the circuit “as is”
- If you want something done, you have to specifically tell it
  - Set Load and Generator kW, etc.
    - Load.MyLoad.kW=125
    - Loadshapes are not used in this mode!
  - Sample Monitors and meters
    - Solve
    - Sample

# Custom Simulation Scripting in "Time" Mode

- Similar to Snapshot mode EXCEPT:
  - Loads, Generators can follow a selected Loadshape
    - Duty, Daily, or Yearly
  - Monitors are automatically sampled
    - But not Energymeters; do that explicitly if desired
  - Time is automatically incremented AFTER solve

# Snapshot Mode Scripting Example

```
! Start the ramp down at 1 sec
Set sec=1
Generator.PV1.kW=(2500 250 -)
Solve
Sample
Set sec=2
Generator.PV1.kW=(2500 500 -)
Solve
Sample

Set sec = 2.020834372 ! Unit 1
storage.jo0235001304 state=discharging %discharge=11.9
Solve
Sample
Set sec = 2.022028115 ! Unit 2
storage.jo0235000257.state=discharging %discharge=11.9
Solve
Sample
Set sec = 2.023158858 ! Unit 3
storage.jo0235000265.state=discharging %discharge=11.9
Solve
Sample
Set sec = 2.024604602 ! Unit 4
storage.jo0235000268_1.state=discharging %discharge=11.9
Solve
Sample
Set sec = 2.025738325 ! Unit 5
storage.jo0235000268_2.dispmode=discharging %discharge=11.9
Solve
Sample

Etc.
```

Set Gen kW explicitly each time step

Solve and Sample explicitly at each step

Set each Storage unit discharge rate explicitly each time step

(time is used for recording purposes only)

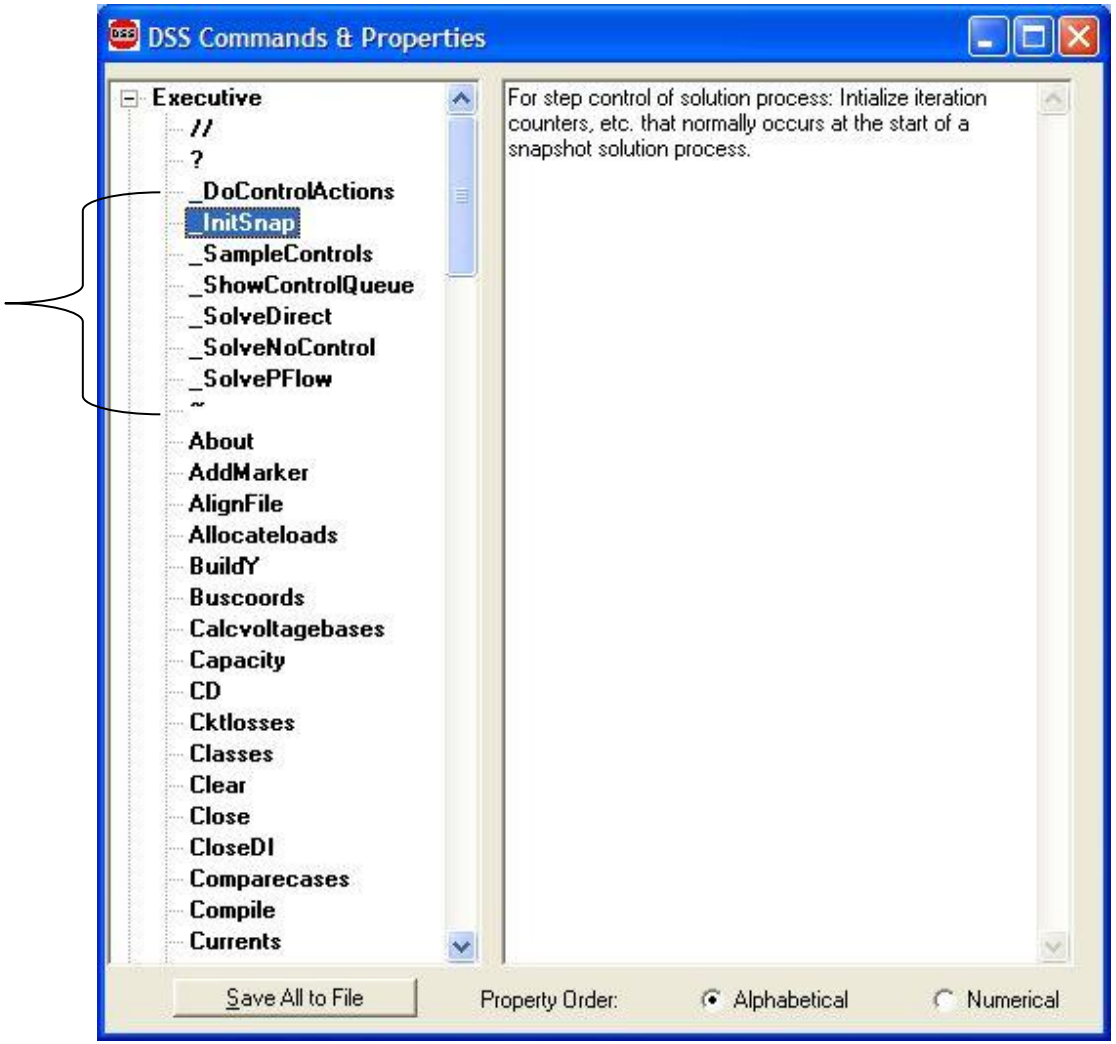
# Time Mode Scripting Example

```
! Start the ramp down at 1 sec
Set mode=time loadshapeclass=duty
set stepsize=1s


Solve   ! Base case t=0
Solve    ! t=t+1 = 2
Solve    ! t=2 second solution; t=t+1 = 3

Set sec = 2.020834372 ! Unit 1 (reset t)
storage.jo0235001304.state=discharging %discharge=11.9
Solve
Set sec = 2.022028115 ! Unit 2
storage.jo0235000257.state=discharging %discharge=11.9
Solve
Set sec = 2.023158858 ! Unit 3
storage.jo0235000265.state=discharging %discharge=11.9
Solve
Set sec = 2.024604602 ! Unit 4
storage.jo0235000268_1.state=discharging %discharge=11.9
Solve
Set sec = 2.025738325 ! Unit 5
storage.jo0235000268_2.dispmode=discharging %discharge=11.9
Solve

Etc.

Set sec=3
Solve   ! t=3 solution; t=t+1 = 4

….
```

All Loads, Generators will follow assigned Duty cycle loadshape

# Custom Simulation Scripting: Rolling Your Own Solution Algorithm

**These commands allow step-by-step control of the solution process**



# Custom Simulation Scripting: Rolling Your Own Solution Algorithm

- The basic Snapshot solution process:
  - Initialize Snapshot (**_InitSnap**)
  - Repeat until converged:
    - Solve Circuit (**_SolveNoControl**)
    - Sample control devices (**_SampleControls**)
    - Do control actions, if any (**_DoControlActions**)

- You may wish, for example, to interject custom control actions after the _SolveNoControl step

# Custom Simulation Scripting, cont'd

- Via COM interface
  - Whatever you want (if you can write code)

- See Examples Folder on Sourceforge site
  - Excel: SampleDSSDriver.xls
  - Matlab:
    - VoltageProfileExample.m
    - DSSMonteCarlo.m

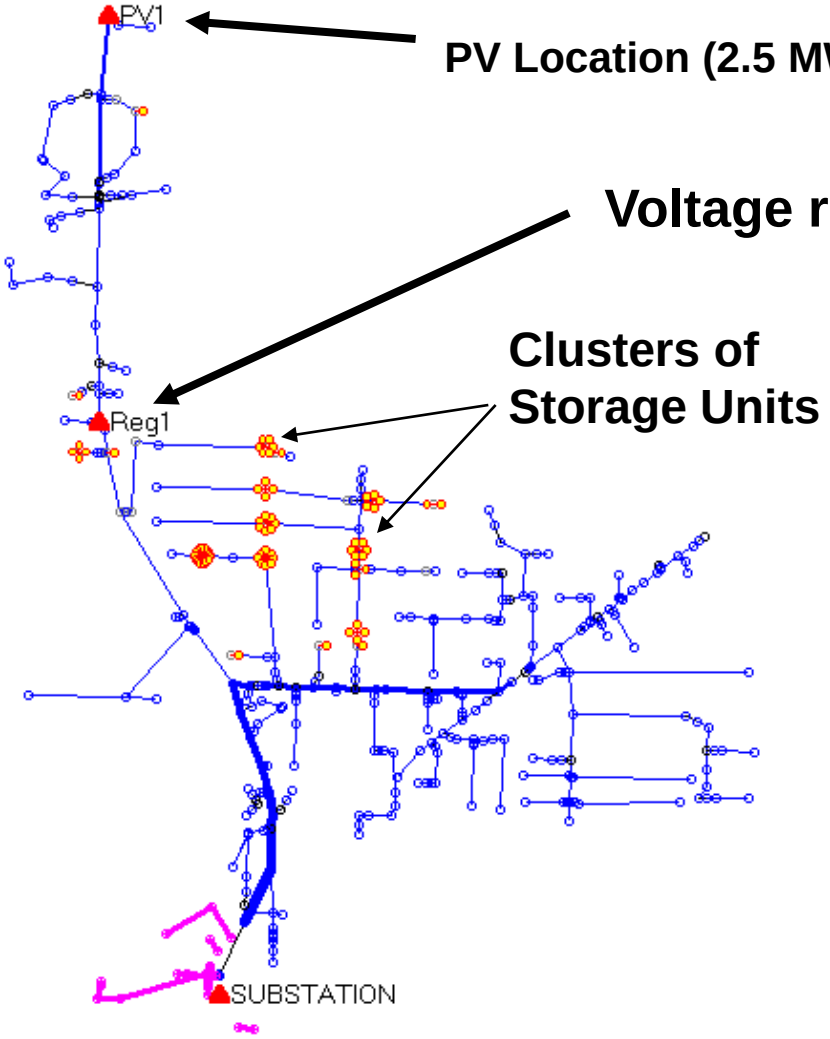
# For More Information …

- See OpenDSS Custom Scripting.Doc
  - (Sourceforge site, “Doc” Folder)

EPRI ELECTRIC POWER RESEARCH INSTITUTE

# Example:
# Energy Storage Dispatch

# The Problem (A Hypothetical Case)




**PV Location (2.5 MW)**

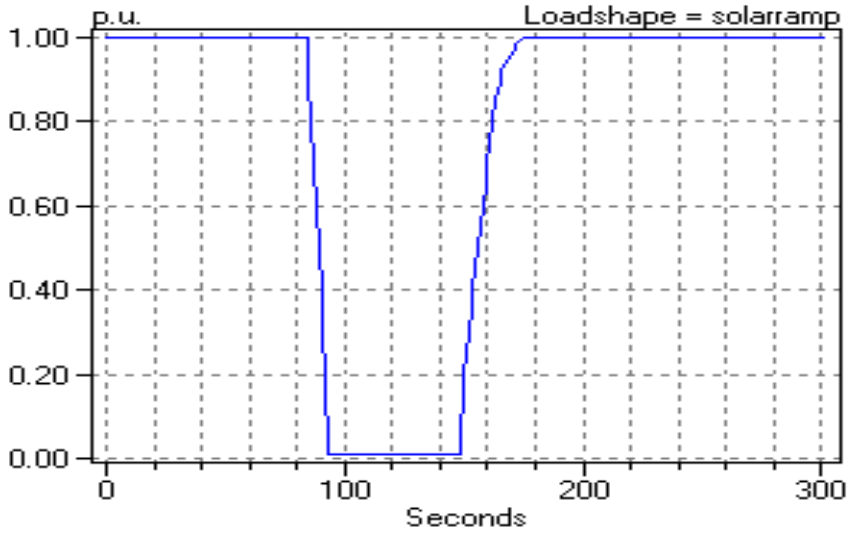
**Voltage regulator**

**Clusters of Storage Units**

Ref: EPRI/AEP Smart Grid Demo

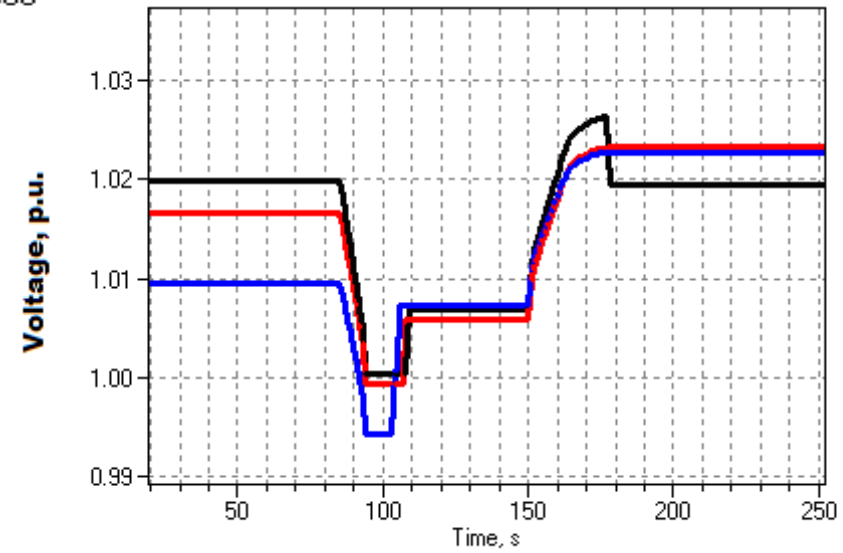
Community Energy Storage Concept

# Solar Ramp Rate Issue




Assumed Solar Ramping Function

Result



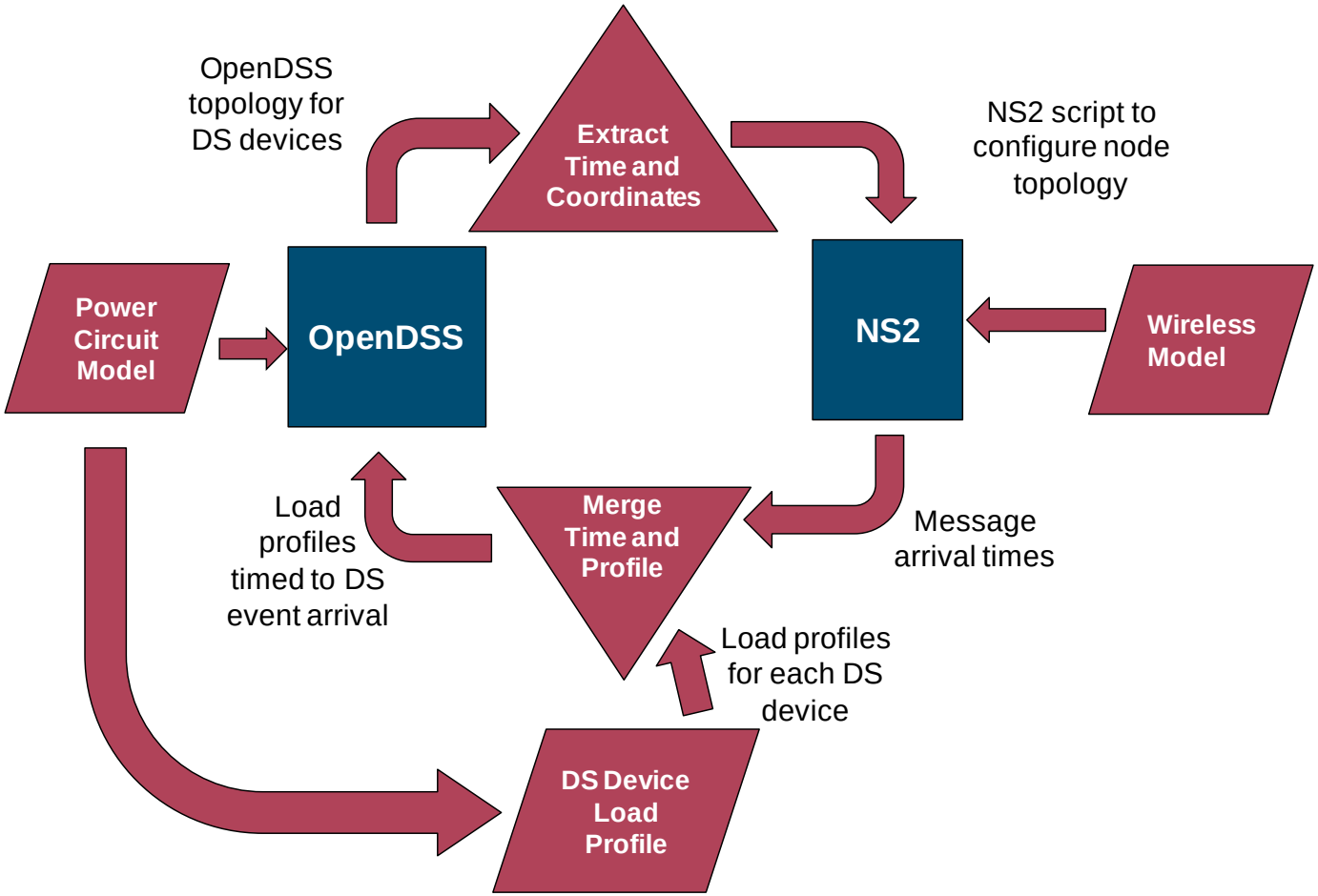
# The Question

- Can you dispatch the 84 CES units fast enough to compensate for the sudden loss of PV generation on a “Cloud Transient” ?

- Why it might not work:
  - Communications latency
  - CES not in right location or insufficient capacity

- Calls for a “Hybrid” simulation
  - Communications network (NS2)
  - Distribution network (OpenDSS)

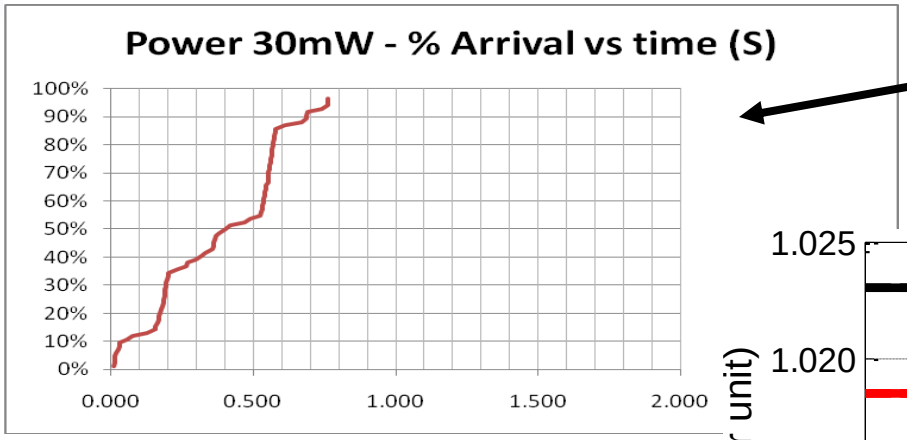
# How We Did It



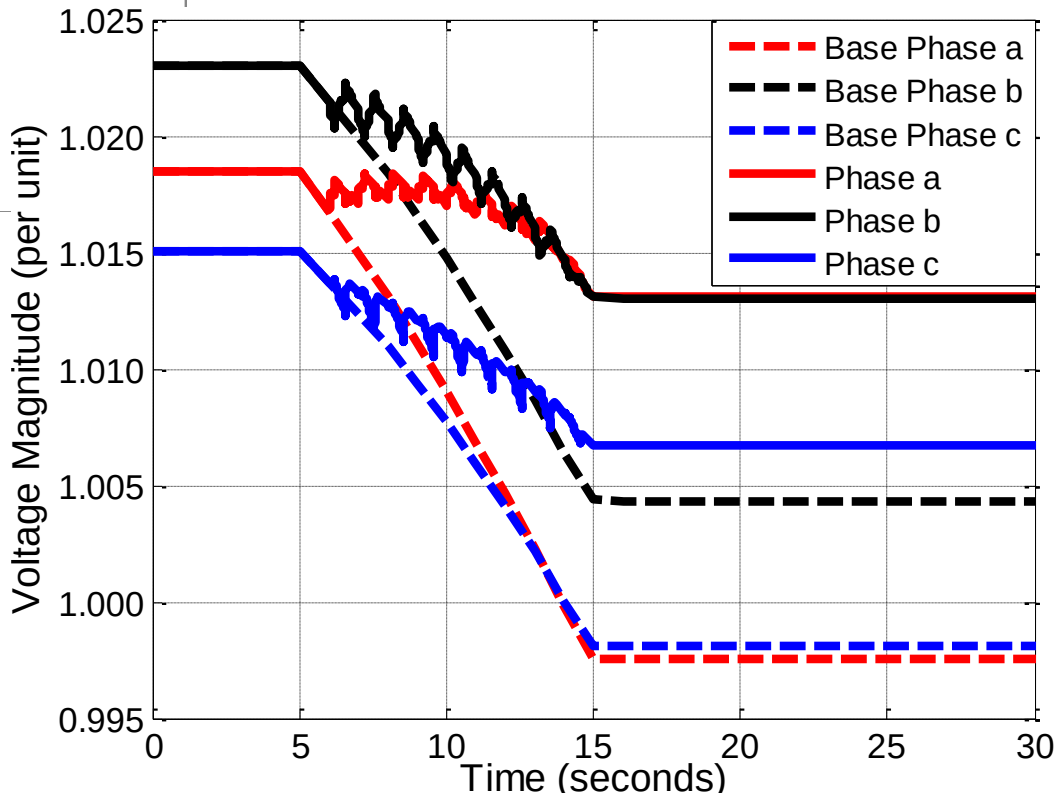
# OpenDSS Script (Snippet)

```
Set sec=20
Solve      ! Init steady state at t=20
Sample


! Start the ramp down at 1 sec
Set sec=21
Generator.PV1.kW=(2500 250 -)  ! Decrement 10%
Solve
Sample
Set sec=22
Generator.PV1.kW=(2500 500 -) ! Decrement another 10%
Solve
Sample

Set sec = 22.020834372 ! Unit 1  message arrives
storage.jo0235001304.state=discharging %discharge=11.9
Solve
Sample
Set sec = 22.022028115 ! Unit 2  message arrives
storage.jo0235000257.state=discharging %discharge=11.9
Solve
Sample

Etc.
```

# Results (for down ramp only)

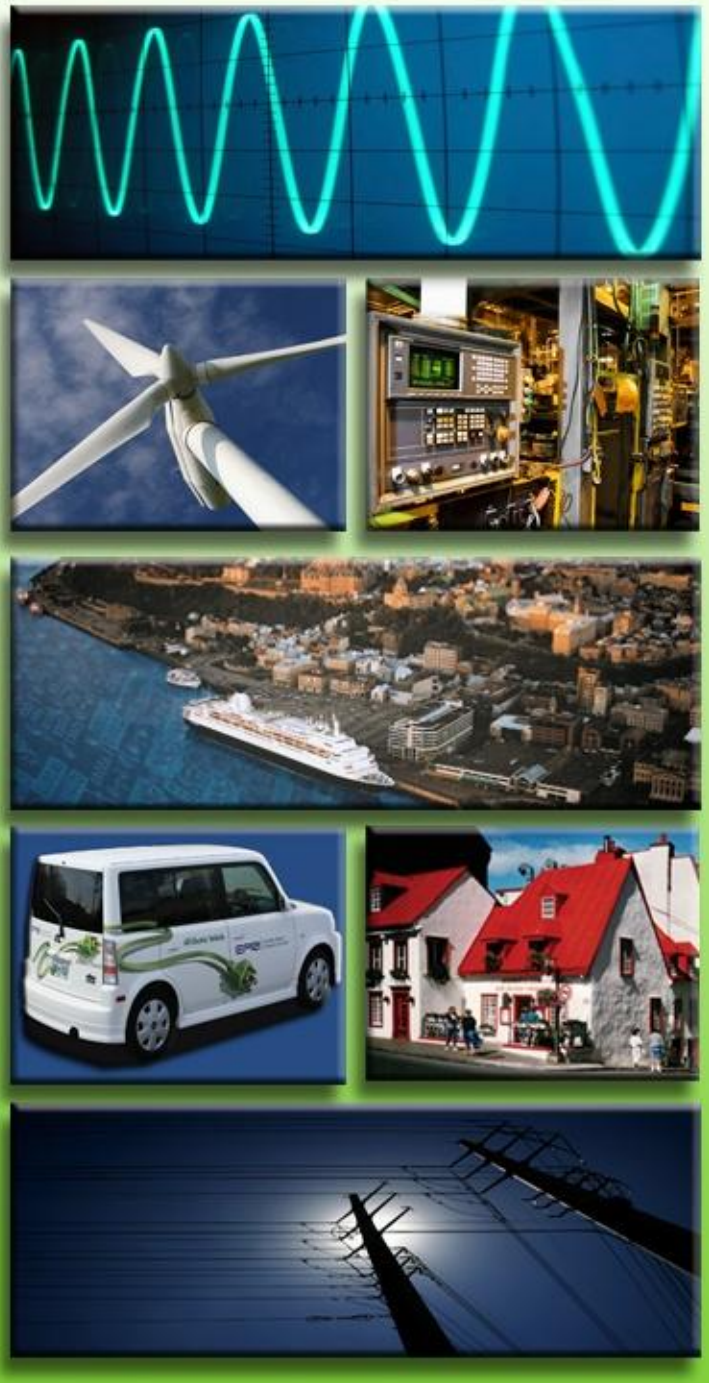



Comm Simulation



# Comm and Power Co-simulation

- New and active reseach area
- Working to more tightly link NS2 and OpenDSS
- Communications latency is an important issue with Smart Grid
  - Power engineers tend to assume communications will happen
  - But there are limits

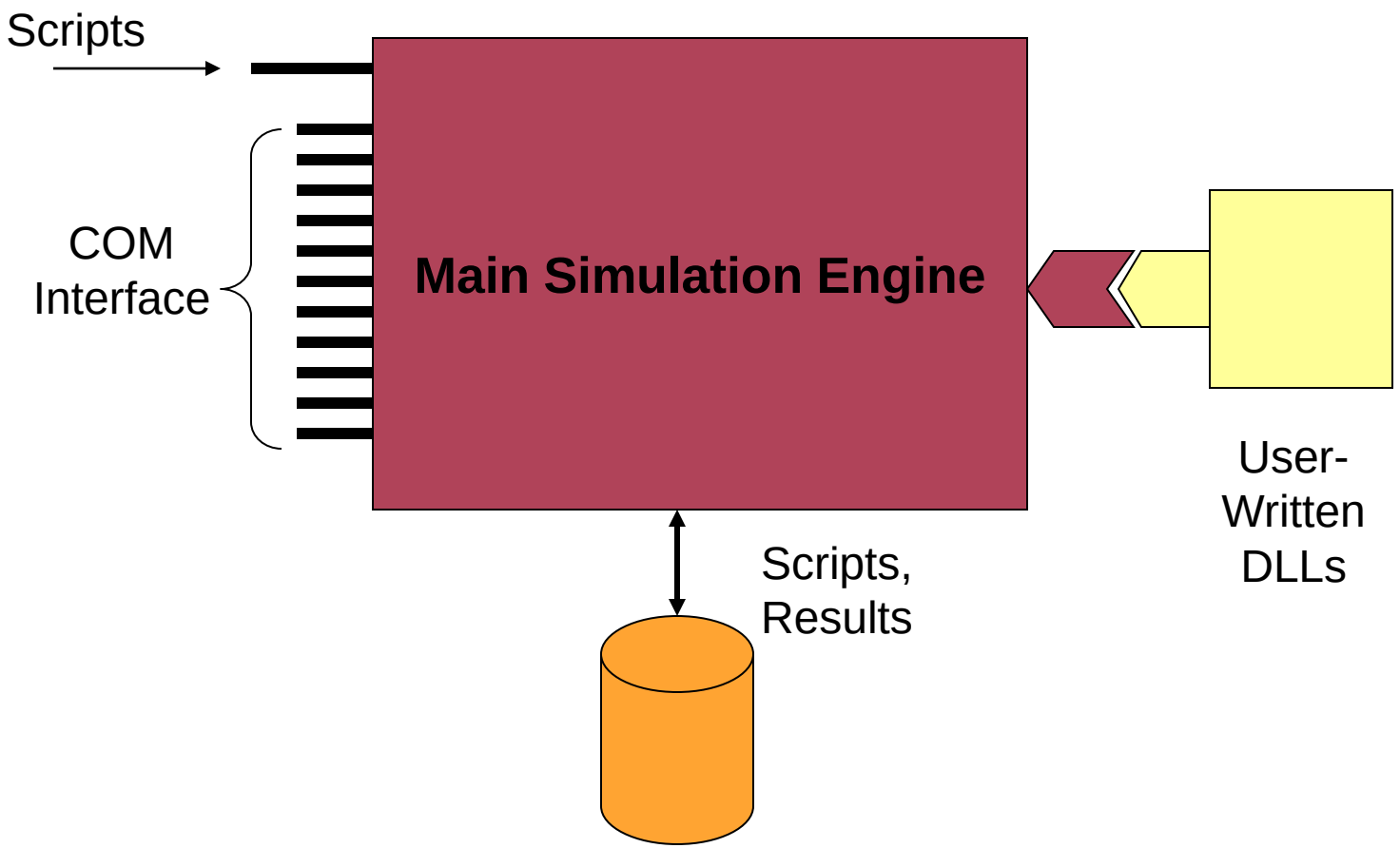


EPRI ELECTRIC POWER RESEARCH INSTITUTE

# Using the COM Interface

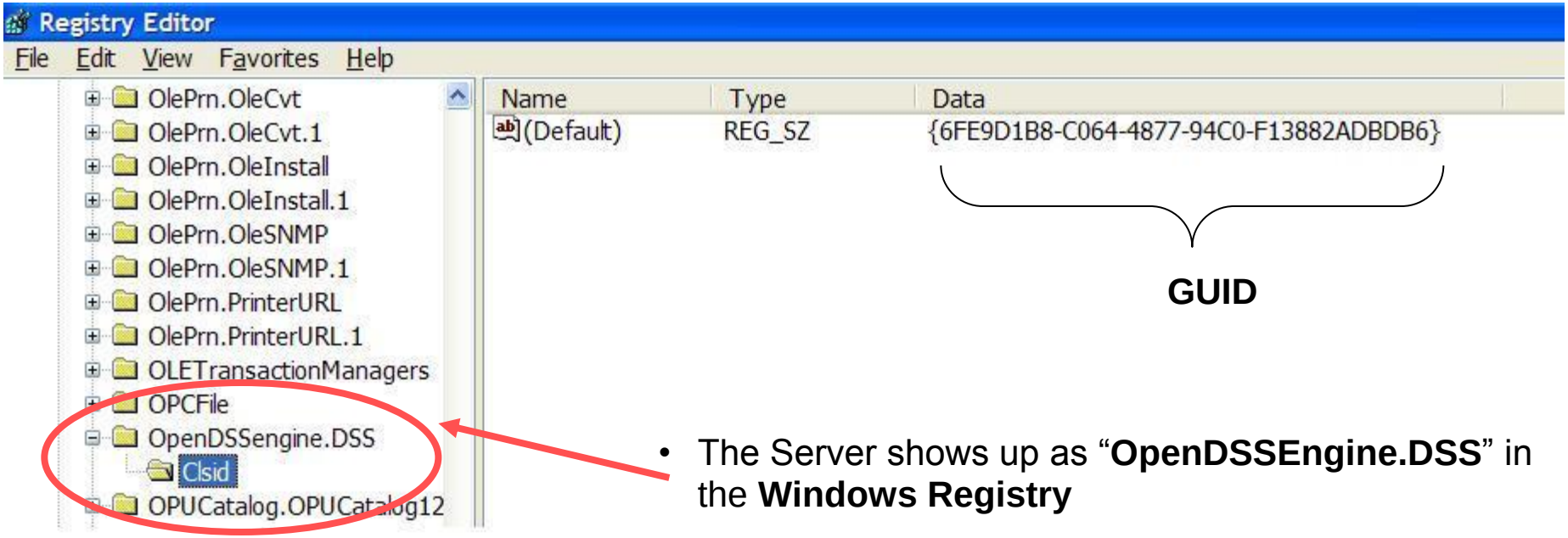
Discussion of the various solution modes built into the OpenDSS

# DSS Structure



# OpenDSSEngine.DSS is Registered

## Windows Registry Entry




- The Server shows up as "**OpenDSSEngine.DSS**" in the **Windows Registry**

**The OpenDSS is now available to any program on the computer**

# Accessing the COM Server

Examples of accessing the COM server in various languages

- In MATLAB:
  - `DSSobj = actxserver('OpenDSSEngine.DSS');`
- In VBA:
  - ```
    Public DSSobj As OpenDSSEngine.DSS
    Set DSSobj = New OpenDSSEngine.DSS
    ```
- In Dephi
  - `{Import Type Library}`
  - `DSSObj := coDSS.Create;`
- In PYTHON:
  - ```
    self.engine =
    win32com.client.Dispatch("OpenDSSEngine.DSS")
    ```

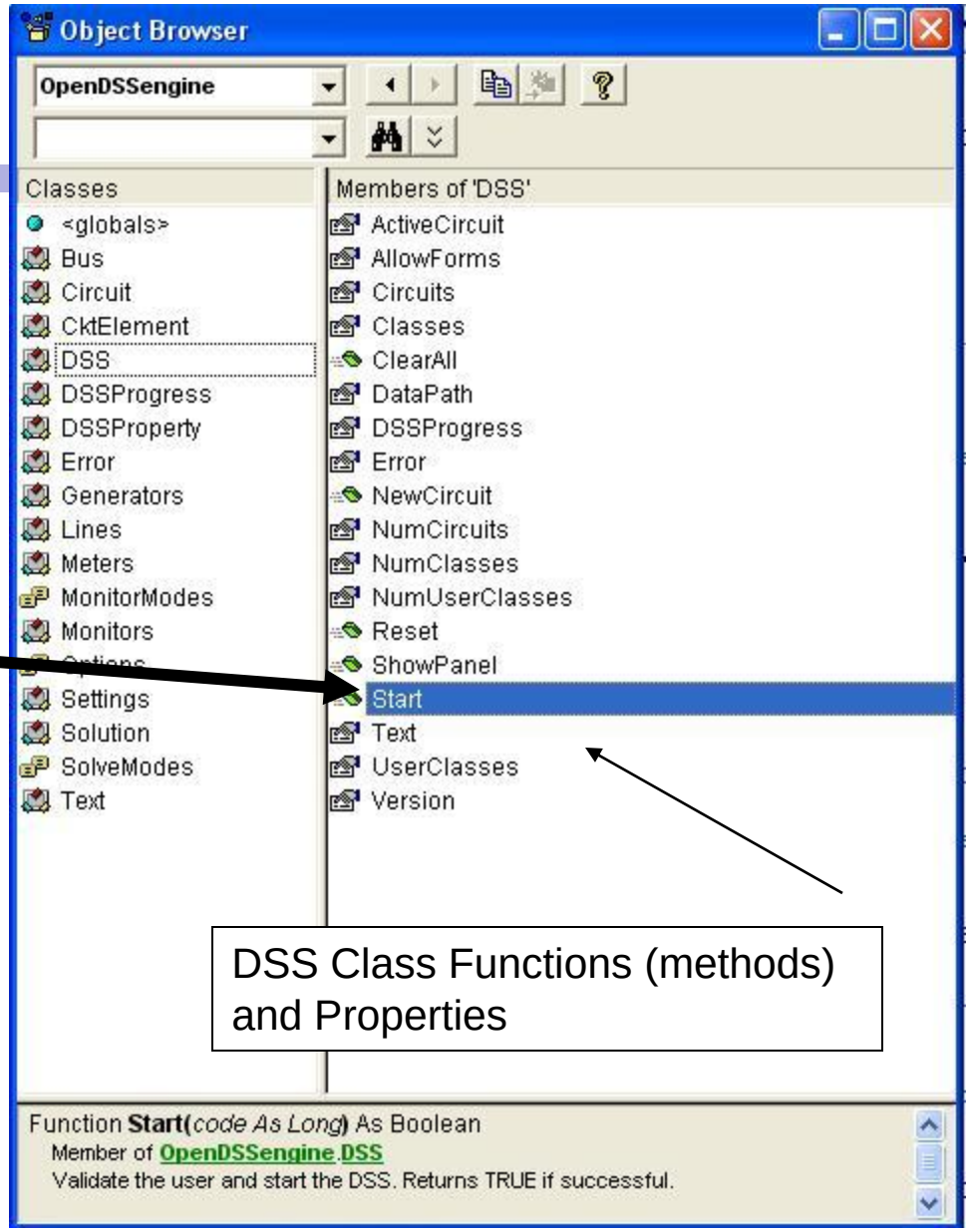
# Active objects concept

- There is one registered *In-Process COM* interface:
  - ***OpenDSSEngine.DSS***
    - That is, the DSS interface is the one you instantiate
    - The DSS interface creates all the others.
- The interfaces generally employ the idea of an **ACTIVE object**
  - Active circuit,
  - Active circuit element,
  - Active bus, etc.
  - The interfaces generally point to the active object
    - To work with another object, change the active object.

# DSS Interface

This interface is instantiated upon loading OpenDSSEngine.DSS and then instantiates all other interfaces

Call the Start(0) method to initialize the DSS



# A Simple VBA Macro

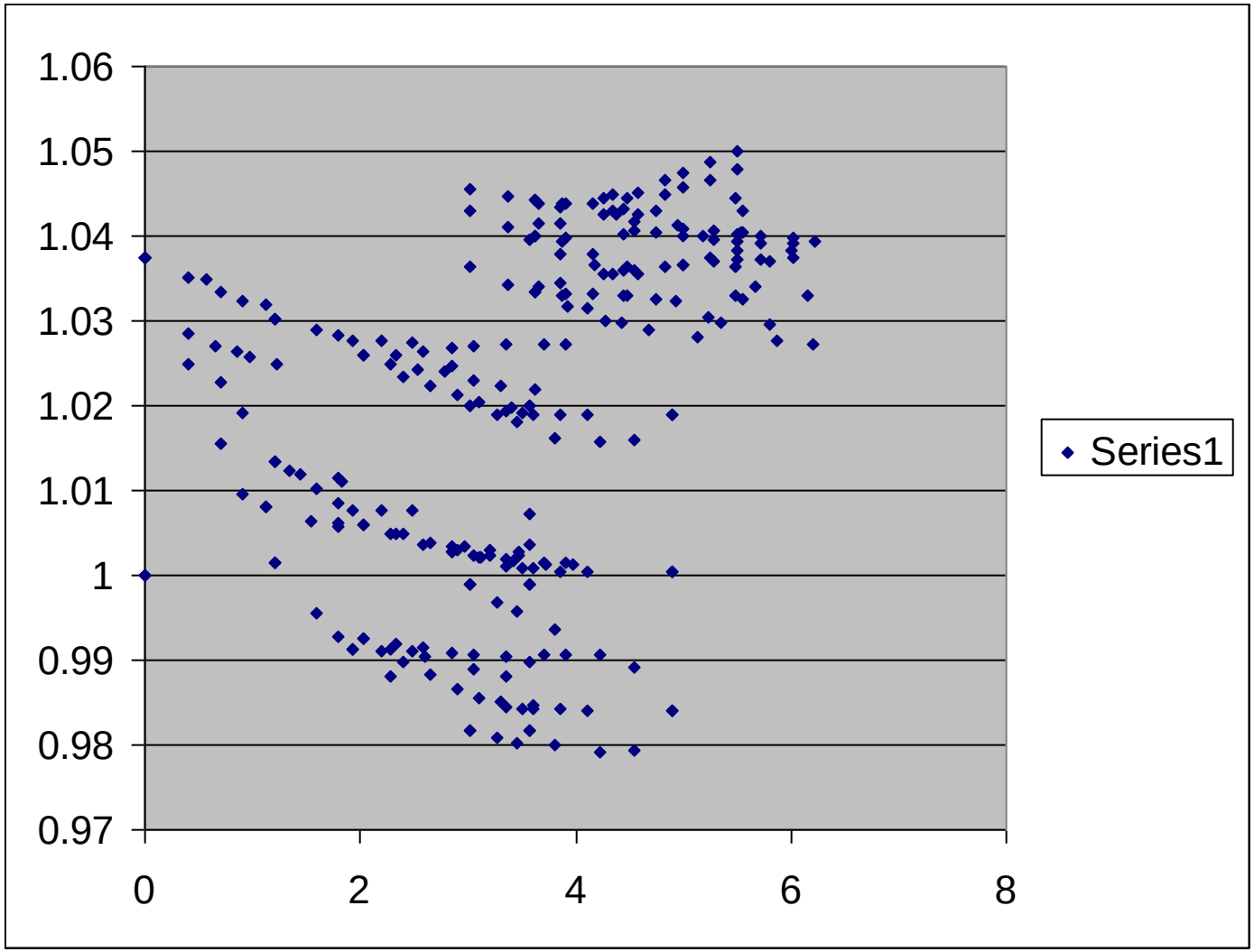
To run the IEEE 123-bus Test Feeder and plot the voltage profile

```
Option Explicit
Public MyOpenDSS As OpenDSSengine.DSS
Public MyText As OpenDSSengine.Text
Public MyCircuit As OpenDSSengine.Circuit

Public Sub MyMacro()
   Set MyOpenDSS = New OpenDSSengine.DSS
   MyOpenDSS.Start (0)

   Set MyText = MyOpenDSS.Text
   Set MyCircuit = MyOpenDSS.ActiveCircuit

   MyText.Command = "Compile (C:\OpenDSS\IEEETestCases\123Bus\IEEE123Master.dss)"
   MyText.Command = "New Energymeter.M1 element=Line.L115 terminal=1"
   MyText.Command = "Solve"

   Dim MyVoltages As Variant
   Dim MyNames As Variant
   Dim Mydistances As Variant
   MyVoltages = MyCircuit.AllBusVmagPu
   MyNames = MyCircuit.AllNodeNames
   Mydistances = MyCircuit.AllNodeDistances

   Dim i As Integer, irow As Integer
   irow = 1
   For i = LBound(MyVoltages) To UBound(MyVoltages)
      ActiveSheet.Cells(irow, 1).Value = MyNames(i)
      ActiveSheet.Cells(irow, 2).Value = Mydistances(i)
      ActiveSheet.Cells(irow, 3).Value = MyVoltages(i)
      irow = irow + 1
   Next I
   Set MyOpenDSS = Nothing
End Sub
```

# Steps to Do This

- Register OpenDSSEngine.DLL  (once)
- Start Excel
- Type alt-F11 to open VBA editor
  - Or *Tools>Macro* …
- Select OpenDSS Engine under *Tools>References*
- *Insert>Module*
- Enter code into blank module
  - Use correct path name for your computer
- Execute the macro “MyMacro”

# Resulting Chart in Excel



EPRI ELECTRIC POWER RESEARCH INSTITUTE

# More on Accessing COM Interface from VBA

MS Excel is one of the better tools for exposing the COM interface

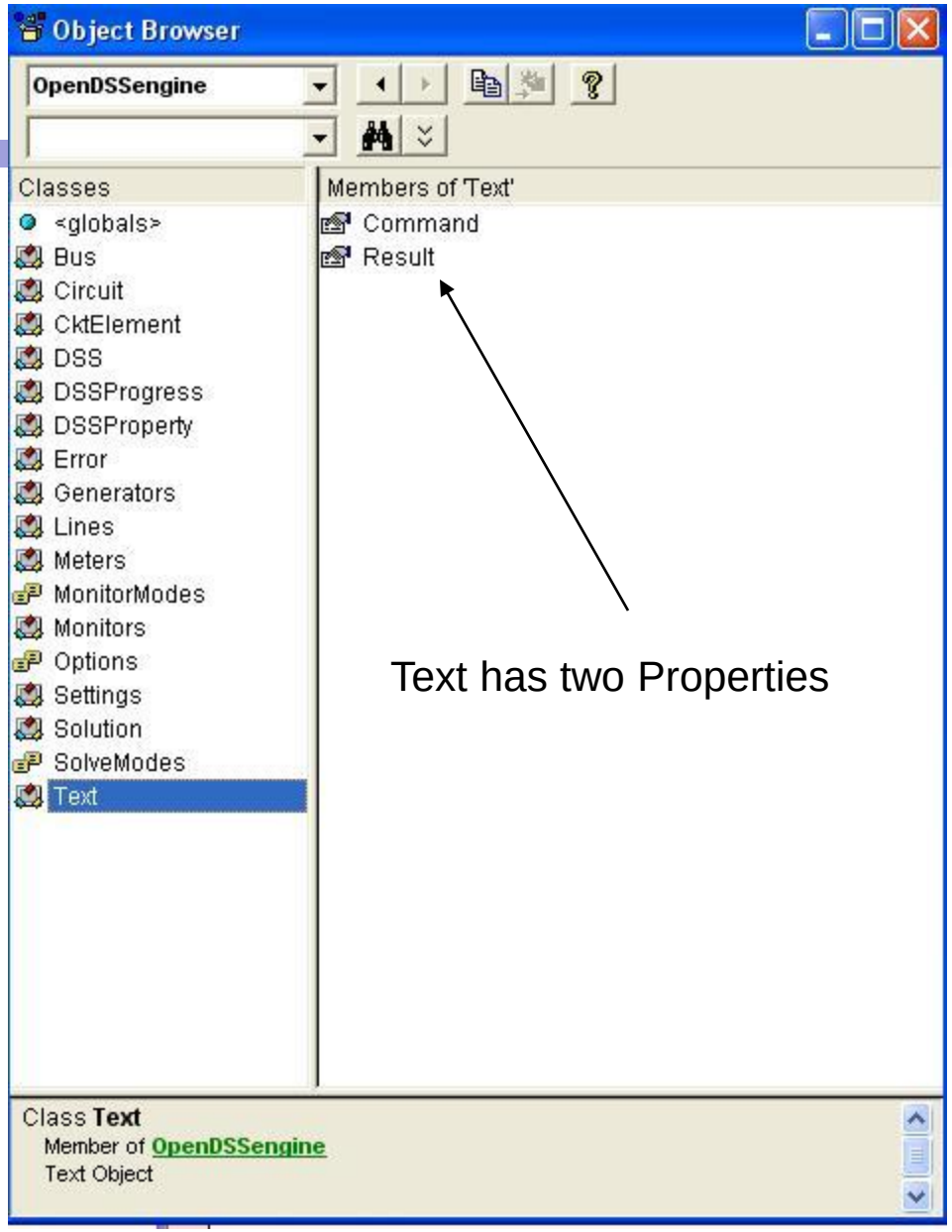
# Instantiate the DSS Interface and Attempt Start

```
Public Sub StartDSS()

' Create a new instance of the DSS
    Set DSSobj = New OpenDSSengine.DSS
' Start the DSS
    If Not DSSobj.Start(0) Then
        MsgBox "DSS Failed to Start"
    Else
        MsgBox "DSS Started successfully"
      ' Assign a variable to the Text interface for easier access
        Set DSSText = DSSobj.Text
    End If


End Sub
```

# COM Interface

Interfaces as Exposed by VBA Object Browser in MS Excel

Text interface is simplest




Text has two Properties

# Assign a Variable to the Text Interface

```
Public Sub StartDSS()

' Create a new instance of the DSS
    Set DSSobj = New OpenDSSengine.DSS
' Start the DSS
    If Not DSSobj.Start(0) Then
        MsgBox "DSS Failed to Start"
    Else
        MsgBox "DSS Started successfully"
      ' Assign a variable to the Text interface for easier access
        Set DSSText = DSSobj.Text
    End If


End Sub
```

# Now Use the Text Interface …

- You can issue any of the DSS script commands from the Text interface

```
' Always a good idea to clear the DSS when loading a new circuit
    DSSText.Command = "clear"
' Compile the script in the file listed under "fname" cell on the main form
    DSSText.Command = "compile " + fname
' Set regulator tap change limits for IEEE 123 bus test case
With DSSText
    .Command = "RegControl.creg1a.maxtapchange=1  Delay=15  !Allow only one tap change per solution.
   This one moves first"
    .Command = "RegControl.creg2a.maxtapchange=1  Delay=30  !Allow only one tap change per solution"
    .Command = "RegControl.creg3a.maxtapchange=1  Delay=30  !Allow only one tap change per solution"
    .Command = "RegControl.creg4a.maxtapchange=1  Delay=30  !Allow only one tap change per solution"
    .Command = "RegControl.creg3c.maxtapchange=1  Delay=30  !Allow only one tap change per solution"
    .Command = "RegControl.creg4b.maxtapchange=1  Delay=30  !Allow only one tap change per solution"
    .Command = "RegControl.creg4c.maxtapchange=1  Delay=30  !Allow only one tap change per solution"
    .Command = "Set MaxControlIter=30"
  End With
```

# Result Property

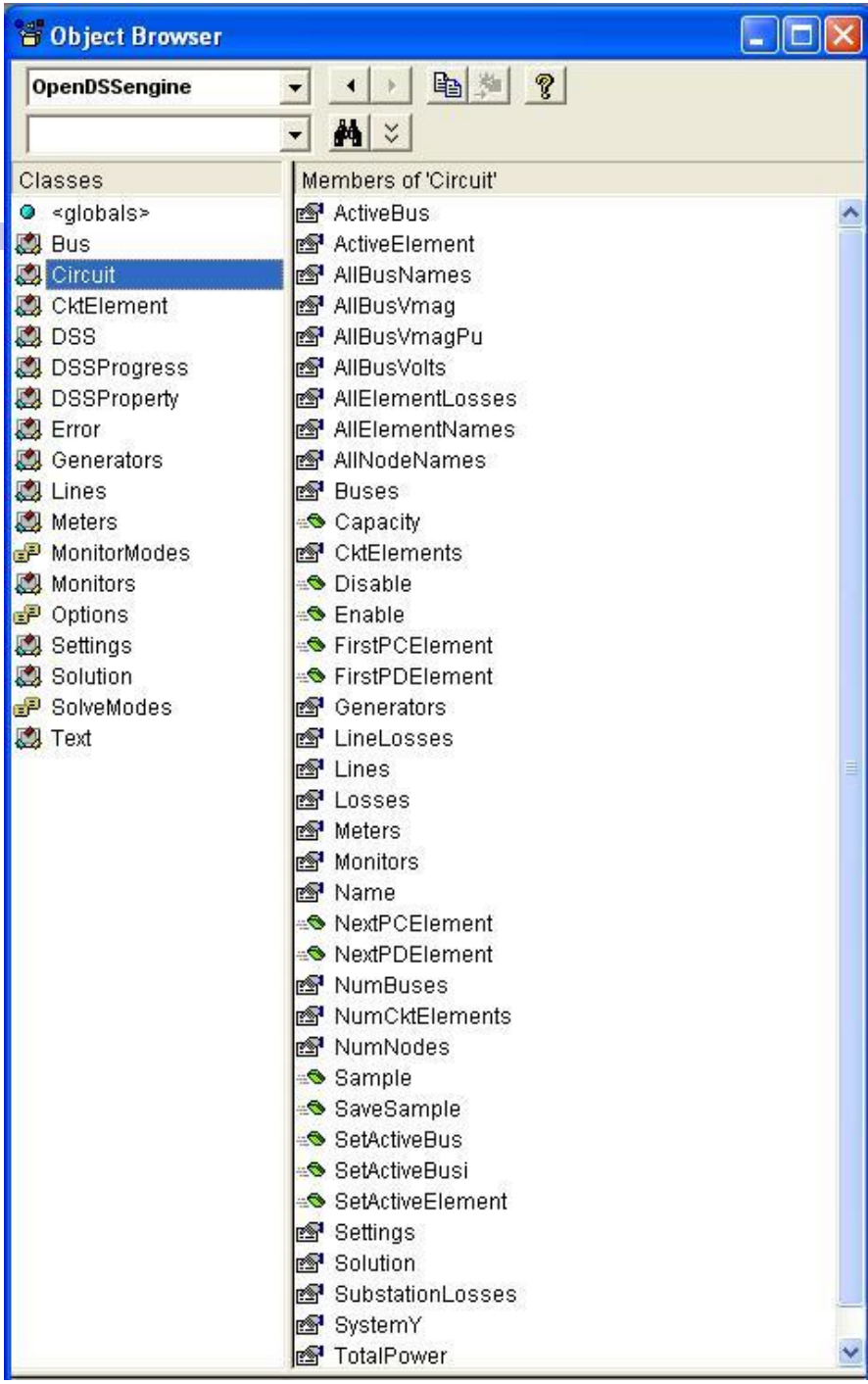
- The Result property is a Read Only property that contains any result messages the most recent command may have issued.
  - Error messages
  - Requested values

```
' Example: Query line length

DSSText.Command = "? Line.L1.Length"

S = DSSText.Result       ' Get the answer

MsgBox S            ' Display the answer
```

# Circuit Interface

This interface is used to

1) Get many of the results for the most recent solution of the circuit
2) Select individual circuit elements in a variety of ways
3) Select the active bus
4) Enable/Disable circuit elements



# Circuit Interface

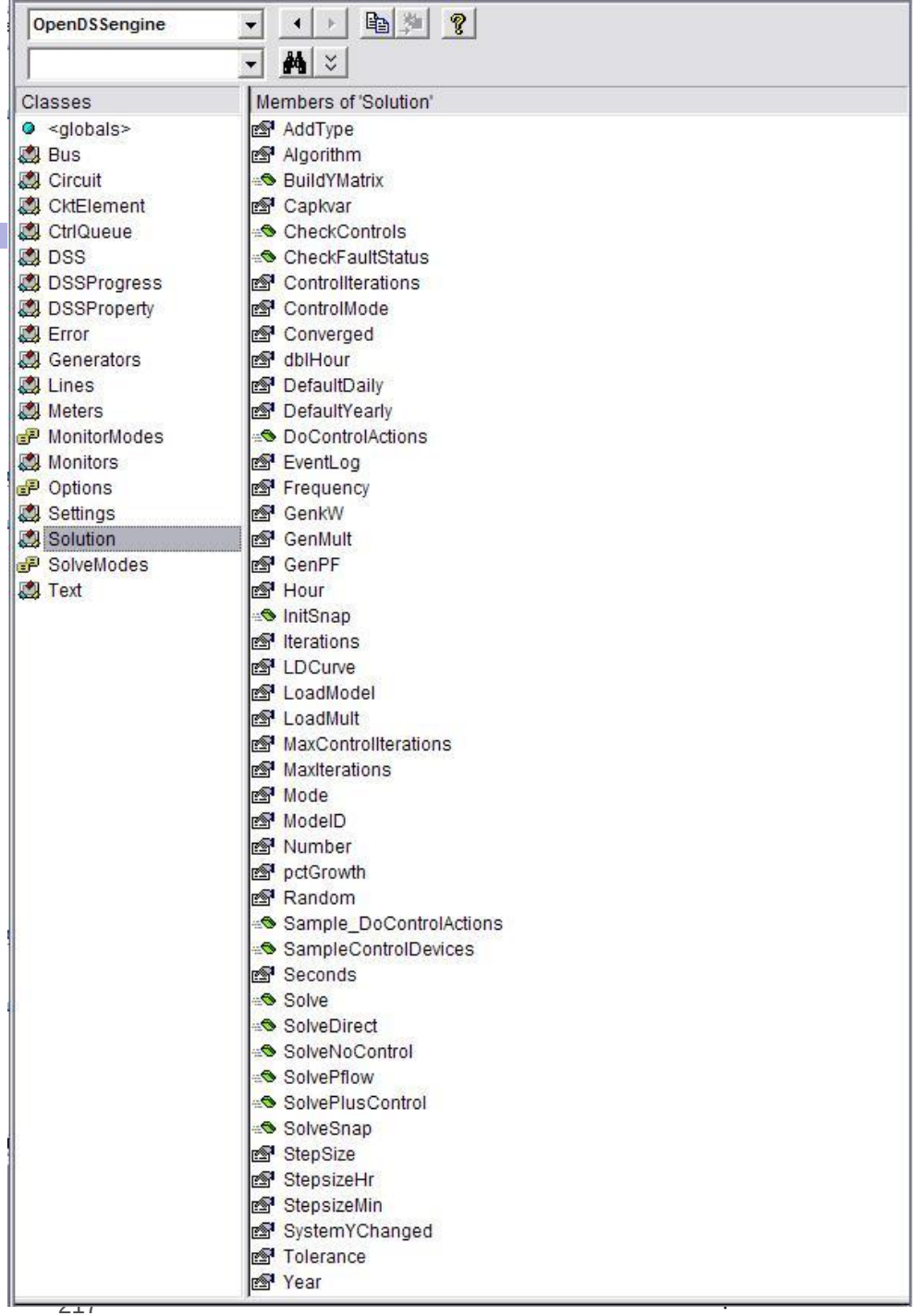
Since the Circuit interface is used often, it is recommended that a special variable be assigned to it:

```
Public DSSCircuit As OpenDSSengine.Circuit
…
DSSText.Command = "Compile xxxx.dss"
Set DSSCircuit = DSSobj.ActiveCircuit
DSSCircuit.Solution.Solve
…   ' Retrieving array quantities into variants
V = DSSCircuit.AllBusVmagPu
VL =DSSCircuit.AllElementLosses
```

# Solution Interface

The Solution Interface is used to

1) Execute a solution
2) Set the solution mode
3) Set solution parameters (iterations, control iterations, etc.)
4) Set the time and time step size



# Solution Interface

Assuming the existence of a DSSCircuit variable referencing the Circuit interface

```
Set DSSSolution = DSSCircuit.Solution
With DSSSolution
…
     .LoadModel=dssAdmittance
     .dblHour = 750.75
     .solve

End With
```

Use the With statement in VBA to simplify coding

# CktElement Interface

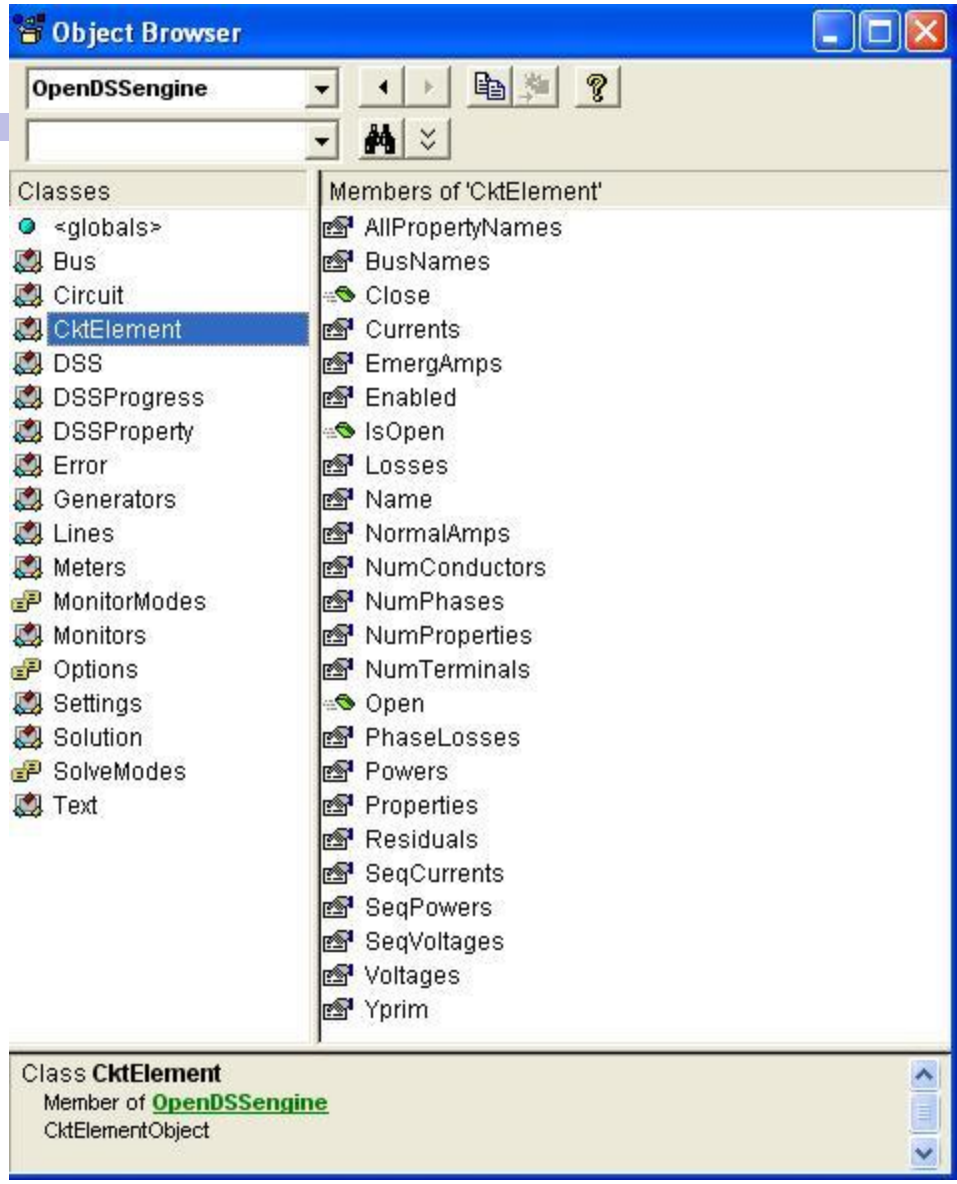
This interface provides specific values of the Active Circuit Element

Some values are returned as variant arrays

```
V = DSSCircuit.ActiveElement.Powers

V = DSSCircuit.ActiveElement.seqCurrents

V = DSSCircuit.ActiveElement.Yprim
```

Other values are scalars

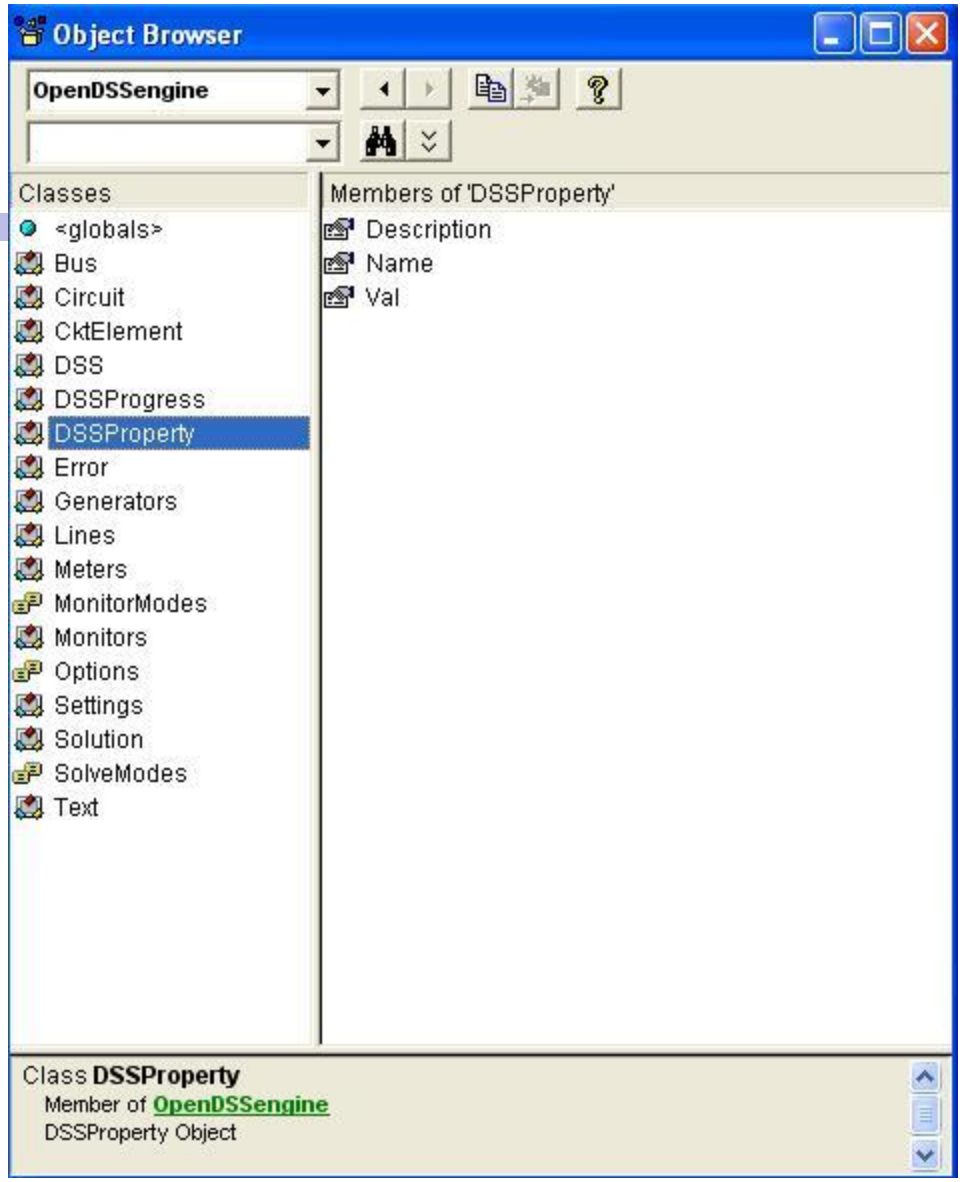
```
Name = DSSCircuit.ActiveElement.Name

Nph = DSSCircuit.ActiveElement.NumPhases
```



# Properties Interface

**This interface gives access to a String value of each public property of the active element**

**“Val” is a read/write property**



# Properties Interface

```
With DSSCircuit.ActiveElement
     ' Get all the property names
     VS = .AllPropertyNames
     ' Get a property value by numeric index
     V = .Properties(2).Val
     ' Get same property value by name (VS is 0 based)
     V = .Properties(VS(1)).Val
     ' Set Property Value by Name
     DSSCircuit.SetActiveElement("Line.L1")
     .Properties('R1').Val = ".068"
End With
```

The last two statements are equivalent to:

DSSText.Command = "Line.L1.R1=.068"

# Lines Interface

This interface is provided to iterate through all the lines in the circuit and change the most common properties of Lines.

# Example: Setting all LineCodes to a Value

```
Set DSSLines = DSSCircuit.Lines
.   .   .
iL = DSSLines.First   'sets active
Do While iL>0
 DSSLines.LineCode = MyNewLineCode
 iL = DSSLines.Next   ' get next line
Loop
```

# VBA Example

```
Option Explicit

Public DSSobj As OpenDSSengine.DSS
Public DSSText As OpenDSSengine.Text
Public DSSCircuit As OpenDSSengine.Circuit

Public Sub StartDSS()

' Create a new instance of the DSS
   Set DSSobj = New OpenDSSengine.DSS

' Assign a variable to the Text interface for easier
  access
   Set DSSText = DSSobj.Text

' Start the DSS
   If Not DSSobj.Start(0) Then MsgBox "DSS
  Failed to Start"

End Sub
```

Define some public variables that are used throughout the project

This routine instantiates the DSS and starts it. It is also a good idea at this time to assign the text interface variable.

# VBA Example

```
Public Sub LoadCircuit(fname As String)

' Always a good idea to clear the DSS when loading a new
   circuit
    DSSText.Command = "clear"

' Compile the script in the file listed under "fname" cell on the
   main form
    DSSText.Command = "compile " + fname

' The Compile command sets the current directory the that of
   the file
' Thats where all the result files will end up.

' Assign a variable to the Circuit interface for easier access
    Set DSSCircuit = DSSobj.ActiveCircuit

End Sub
```

This subroutine loads the circuit from the base script files using the Compile command through the Text interface. “fname” is a string contains the name of the master file.

There is an active circuit now, so assign the DSSCircuit variable.

# VBA Example

```
Public Sub LoadSeqVoltages()

' This Sub loads the sequence voltages onto Sheet1 starting in Row 2

   Dim DSSBus As OpenDSSengine.Bus
   Dim iRow As Long, iCol As Long, i As Long, j As Long
   Dim V As Variant
   Dim WorkingSheet As Worksheet

   Set WorkingSheet = Sheet1   'set to Sheet1 (target sheet)

   iRow = 2
   For i = 1 To DSSCircuit.NumBuses  ' Cycle through all buses

      Set DSSBus = DSSCircuit.Buses(i)  ' Set ith bus active

   ' Bus name goes into Column 1
      WorkingSheet.Cells(iRow, 1).Value = DSSCircuit.ActiveBus.Name

   ' Load sequence voltage magnitudes of active bus into variant array
      V = DSSBus.SeqVoltages

   ' Put the variant array values into Cells
   ' Use Lbound and UBound because you don't know the actual range
      iCol = 2
      For j = LBound(V) To UBound(V)
         WorkingSheet.Cells(iRow, iCol).Value = V(j)
         iCol = iCol + 1
      Next j
     iRow = iRow + 1
   Next i

End Sub
```

This Sub puts the sequence voltage onto a spreadsheet

Define a variable for the Bus interface

Define a variant to pick up the arrays

Cycle through all the buses

Get the bus name

Get the voltages into the variant array

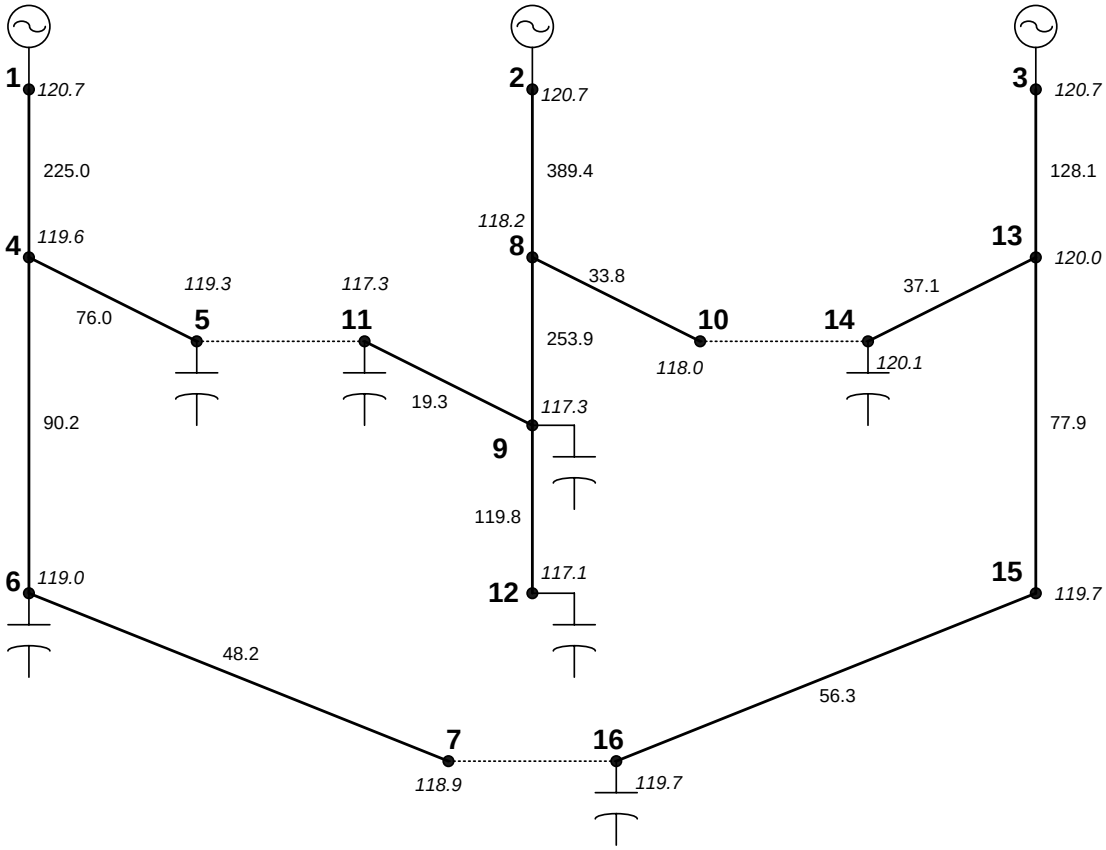
Put them on the spreadsheet

# Branch Exchange Example

See
OpenDSS Wiki on SourceForge.Net

TechNote_Excel_VBA_Example

# Loss Minimization – Branch Exchange




- Branch Exchange
- Loop Breaking
- Branch & Bound
- Heuristics
- Optimization
  - Genetic Alg.
  - Simulated Annealing
  - Ant Colony
  - Et cetera

```
While Not done
    r = iter + 1
    ws.Cells(r, 10) = iter
    ckt.Solution.Solve                  ‘ solve the current system
    ThisLoss = ckt.Losses(0)
    ws.Cells(r, 11) = ThisLoss   ‘ write current losses, # loops, # isolated loads to sheet
    ws.Cells(r, 12) = CStr(ckt.Topology.NumLoops) & " _ " & _
                      CStr(ckt.Topology.NumIsolatedLoads)

    Vmax = 0# ‘ track the maximum voltage difference across any open switch
    ToClose = ""
    ToOpen = ""
    LowBus = ""
    c = 14                          ‘ column number for output
    i = swt.First                   ‘ check all SwtControls
    While i > 0 ' find the open switch with biggest delta-V

      … (Inner loop – next slide)

    Wend
    ws.Cells(r, 13) = CStr(Vmax)

    done = True          ' unless we found a switch pair to exchange
    If Len(ToOpen) > 0 And Len(ToClose) > 0 Then                    ‘ found a switch pair to exchange
      swt.name = ToClose                ‘ do the switch close-open via SwtControl interface
      swt.Action = dssActionClose
      swt.name = ToOpen
      swt.Action = dssActionOpen
      done = False ' try again   ‘ i.e., run solution again and look for the next exchange
    End If

    iter = r
    ' stop if too many iterations, system is non-radial, or losses go up
    If iter > 10 Or ckt.Topology.NumIsolatedLoads > 0 Or ThisLoss > LastLoss Then
      done = True               ‘ met one of the three stopping criteria
    End If
    LastLoss = ThisLoss         ‘ best loss total found so far
  Wend
```

# Inner Loop

```
While i > 0 ' find the open switch with biggest delta-V
    If swt.Action = dssActionOpen Then          ' check only open switches
      ws.Cells(r, c) = swt.name
      Set elem = ckt.CktElements(swt.SwitchedObj)
      Vdiff = Abs(elem.SeqVoltages(1) - elem.SeqVoltages(4))       ' V1 across switch
      If Vdiff > Vmax Then       ' if highest V1 difference so far…
        LowBus = FindLowBus              ' which side of open switch has lowest V?
        topo.BusName = LowBus            ' start from that bus in the topology
        Set elem = ckt.ActiveCktElement
        k = 1
        While (Not elem.HasSwitchControl) And (k > 0)        ' trace back from low bus to src
          k = topo.BackwardBranch                              ' until we find a closed switch
        Wend
        If elem.HasSwitchControl Then                  ' if we found a switch to close…
          Vmax = Vdiff                    ' keep this as the highest voltage difference found
          ToClose = swt.name              ' we will close this currently-open switch
          ToOpen = Mid(elem.Controller, 12)          ' and open the switch from back-trace
        End If
      End If
      c = c + 1
    End If
    i = swt.Next
  Wend
```

EPRI ELECTRIC POWER RESEARCH INSTITUTE

# Running from MATLAB

# Running OpenDSS From Matlab

- Starting the DSS

```
%Start up the DSS
[DSSStartOK, DSSObj, DSSText] = DSSStartup;
```

```
function [Start,Obj,Text] = DSSStartup
% Function for starting up the DSS
%
%instantiate the DSS Object
Obj = actxserver('OpenDSSEngine.DSS');
%
%Start the DSS.   Only needs to be executed the first time w/in a
%Matlab session
Start = Obj.Start(0);

% Define the text interface to return
Text = Obj.Text;
```

# Using the DSS through the DSSText Interface from Matlab (harmonics example)

```
%Compile the DSS circuit script
DSSText.Command = 'compile master.dss';

% get an interface to the active circuit called "DSSCircuit"
DSSCircuit = DSSObj.ActiveCircuit;

%Determine which connection type for the source and call
%appropriate DSS file
switch XFMRType
case 1
 DSSText.Command = 'redirect directconnectsource.DSS';
case 2
 DSSText.Command = 'redirect deltadelta.DSS';
case 3
 DSSText.Command = 'redirect deltawye.DSS';
otherwise
 disp('Unknown source Connection Type')
end

%Set the system frequency and vsource frequency for harmonic requested
DSSText.Command = ['set frequency=(' num2str(Freq) ' 60 *)'];
DSSText.Command = ['vsource.source.frequency=(' num2str(Freq) ' 60 *)'];
```

# Using the DSS through the DSSText Interface from Matlab (harmonics example) (cont'd)

```
% Vary the parameters according to a random distribution
% If more parameters need to be varied, just add them to the below
% list.  Set ParamNum to total number of parameters varied
 ParamNum = 6;    %ParamNum used for sorting/plotting
 for Case_Count = 1:Max_Cases
%Create index in the OutputData matrix to keep the cases in order
 OutputData(Case_Count,1) = Case_Count;
 % Generate random new coordinates for each conductor
 [x1 y1 x2 y2 x3 y3 geomean] = RandomGeometry(8,0.75,30);
            (... etc. etc. )
%define a new line geometry with random spacing
DSSText.Command = ['New LineGeometry.OHMOD nconds=3 nphases=3  cond=1  wire=acsr336    x='
num2str(x1) '   ' num2str(y1) '   units=ft  cond=2  wire=acsr336    x=' num2str(x2) '   '
num2str(y2) '   units=ft cond=3  wire=acsr336    x=' num2str(x3) '   ' num2str(y3) '
units=ft'];
%Solve the circuit
 DSSText.Command = 'solve';
              (etc. etc.)
```

# Together…Shaping the Future of Electricity