OpenDSS Training Workshop

Introduction to OpenDSS

Roger C. Dugan
EPRI Knoxville, TN

August 30, 2021

Instructor

Roger C. Dugan, *Life Fellow, IEEE*

- **Roger is a Sr. Technical Executive with EPRI in Knoxville, Tennessee USA. He has 50 years of combined experience in distribution engineering with EPRI, Electrotek Concepts, and Cooper Power Systems. He holds the BSEE degree from Ohio University and the Master of Engineering in Electric Power Engineering degree from Rensselaer Polytechnic Institute, Troy, NY. Roger has worked on many diverse aspects of power engineering over his career because of his interests in applying computer methods to power system simulation. Beginning with a student internship with Columbus and Southern Ohio Electric Co, his work has been focused on Distribution Engineering. He was elected a Fellow of the IEEE for his contributions in harmonics and transients analysis. Recently, he has been very active in distributed generation, particularly as it applies to utility distribution systems and distribution system analysis. He was the 2005 recipient of the IEEE Excellence in Distribution Engineering Award. He is coauthor of Electrical Power Systems Quality published by McGraw-Hill, now in its 3rd edition. He serves on the IEEE PES Distribution System Analysis Subcommittee and is active in the Distribution Test Feeders WG.**

1. Introduction to the OpenDSS Program

What is the OpenDSS?

- The examples in this Workshop will be taught using the EPRI OpenDSS computer program
- OpenDSS is EPRI's main tool for research involving distribution system analysis
- Was made Open Source in 2008.
  - Over 120,000 downloads since.
- The program is freely available and is updated regularly
- You may find it on www.Sourceforge.net
- Also, bookmark the EPRI link site:
  - https://www.epri.com/#/pages/sa/opendss?lang=en-US
  - This is the "jump page" for OpenDSS internet locations

Overview of OpenDSS
OpenDSS – Open-Source Distribution System Simulator

Download OpenDSS Here

- Advanced distribution analysis platform that enables engineers to perform complex distribution analysis
- Flexible and customizable solution designed specifically to meet the challenges facing distribution engineers
- Enables engineers to easily model both traditional and advanced distribution technologies, resources, assets, and controls
- Leveraged throughout the industry for modeling and simulating advanced distribution applications
- Designed from the beginning to capture the time and spatial affects of distributed energy resources

- Brief history and current usage
  - Developed/designed in 1997 to capture the time and spatial affects of distributed energy resources
  - Open-sourced in 2008 to coordinate and advanced smart grid assessments
  - Primary modeling and simulation platform used to enable execution of cutting-edge research

Highlighting a Few Capabilities

Solution Capabilities

- Unbalanced multi-phase power flow
- Quasi-static time-series (QSTS)
- Fault analysis
- Harmonic analysis
- Flicker analysis
- Linear and non-linear analysis
- Stray voltage/current analysis

Grid Devices

- Full library of traditional assets (lines, conductors, transformers, cap banks, switches, etc.)
- Load models

Controls

- Line Reg/LTC, cap banks
- DER smart inverter
- Energy storage dispatch
- DMS/DERMS
- VVO
- Price modeling/dispatch

Automation

- Distribution automation
- Load transfers
- FLISR (Fault Location, Isolation, and Service Restoration)

DER Models

- Smart inverters
- Energy storage
- PV systems
- Wind systems
- Demand response
- Microgrids
- DER short-ckt

Solution Interfaces

- Distribution System Scripting language
- Full graphical user-interface
- Co-simulation capabilities
- Integrated SDK for customized development

High-Performance Solutions

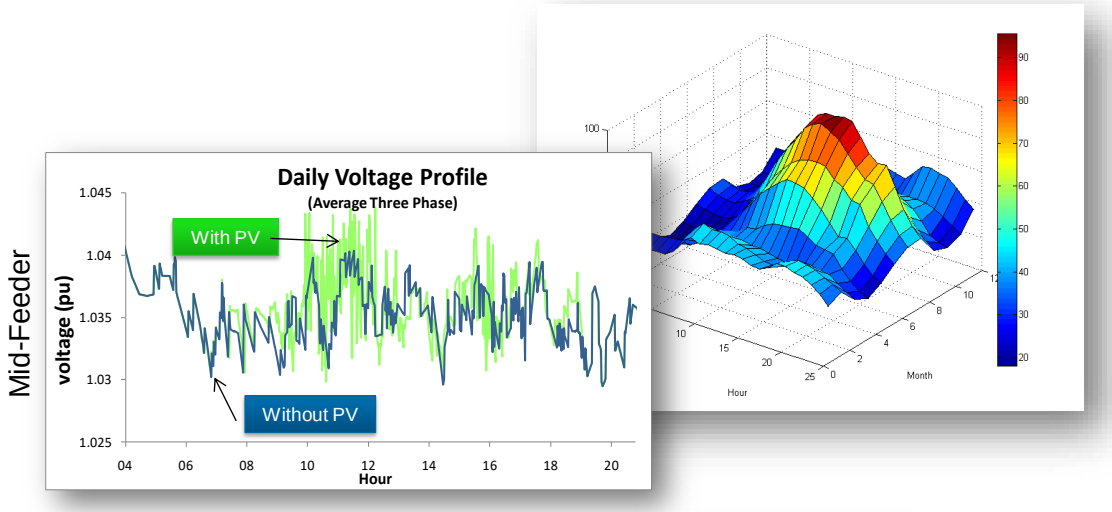
- Parallel processing using actors
- Multithreading circuit processing
- Multi-core management
- Fast power flow

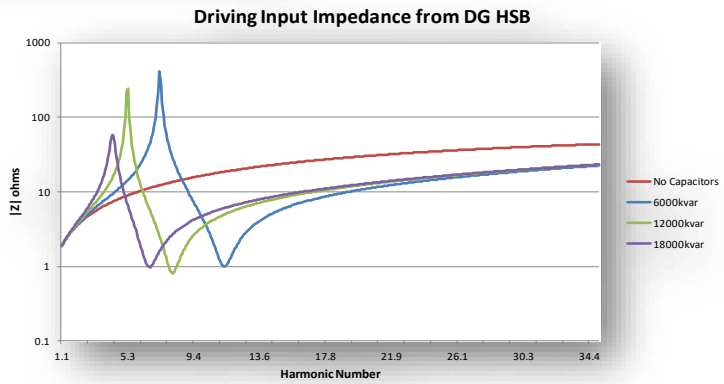
Misc

- Radial and networked systems
- Arbitrary sized systems (single feeder to planning area)
- Transmission and distribution modeling

Flexible Tool Enabling a Wide Range of Analysis Types

- DER Interconnection studies
- Locational value studies
- Hosting capacity studies
- DA/FLISR scheme evaluation
- Volt/var optimization
- Energy impact analysis
- DER protection impacts
- Power quality (harmonics/flicker)
- Long-range planning studies
- Smart inverter control optimization
- Planning for electrification





What is the OpenDSS?

- Script-driven, frequency-domain electrical circuit simulation tool
  - Script with text commands
  - Script with another program (Python, MATLAB, C++, Delphi, etc.)
    - Use Windows COM interface
    - Use Direct DLL interface
    - Drive with OpenDSS-G

- Specific models for:
  - Supporting **utility distribution system** analysis
  - Designed for the unbalanced, multi-phase North American power distribution systems
    - Can model European-style systems also
      - These typically have a simpler structure

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
  - More like EMT or Dynamics model than typical Distribution Power Flow

- Supports all **rms steady-state** (i.e., frequency domain) analyses commonly performed for utility distribution system planning
  - And many other types of analyses
  - Original purpose: **DG interconnection analysis**

What is the OpenDSS? (cont'd)

- What it Isn't
  - An *Electromagnetic* transients (EMT)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

Time- and Location-Dependent Benefits

- The OpenDSS was designed 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
  - "Quasi-Static Time-Series" (QSTS) simulations
  - Over distribution planning area

What are the Key Features?

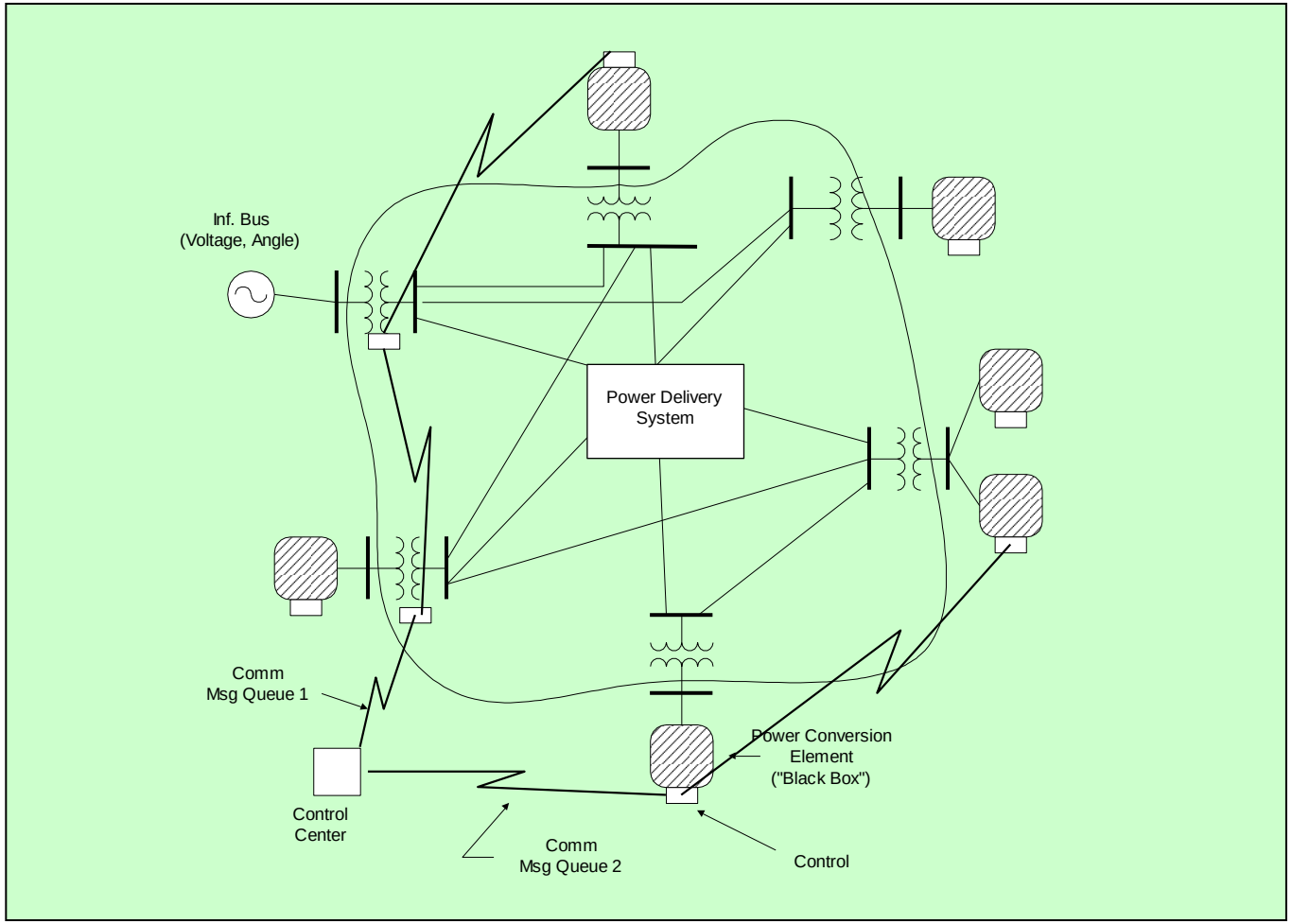
- Can model virtually any distribution circuit configuration
- Very flexible multi-phase **transformer model**
- Fast and efficient **QSTS simulations**
- Designed to allow expansion indefinitely
  - Impossible to anticipate everything users will want to do
  - Scripting language and COM interface allows easier customization
  - OOP design of software makes it easy to add new circuit element models

- See Documentation for more details
  - https://sourceforge.net/p/electricdss/code/HEAD/tree/trunk/Version8/Doc/
  - **Help** menu on OpenDSS.exe
  - **Doc folder** on your OpenDSS installation
  - **Examples folder** on your OpenDSS installation

Controls

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

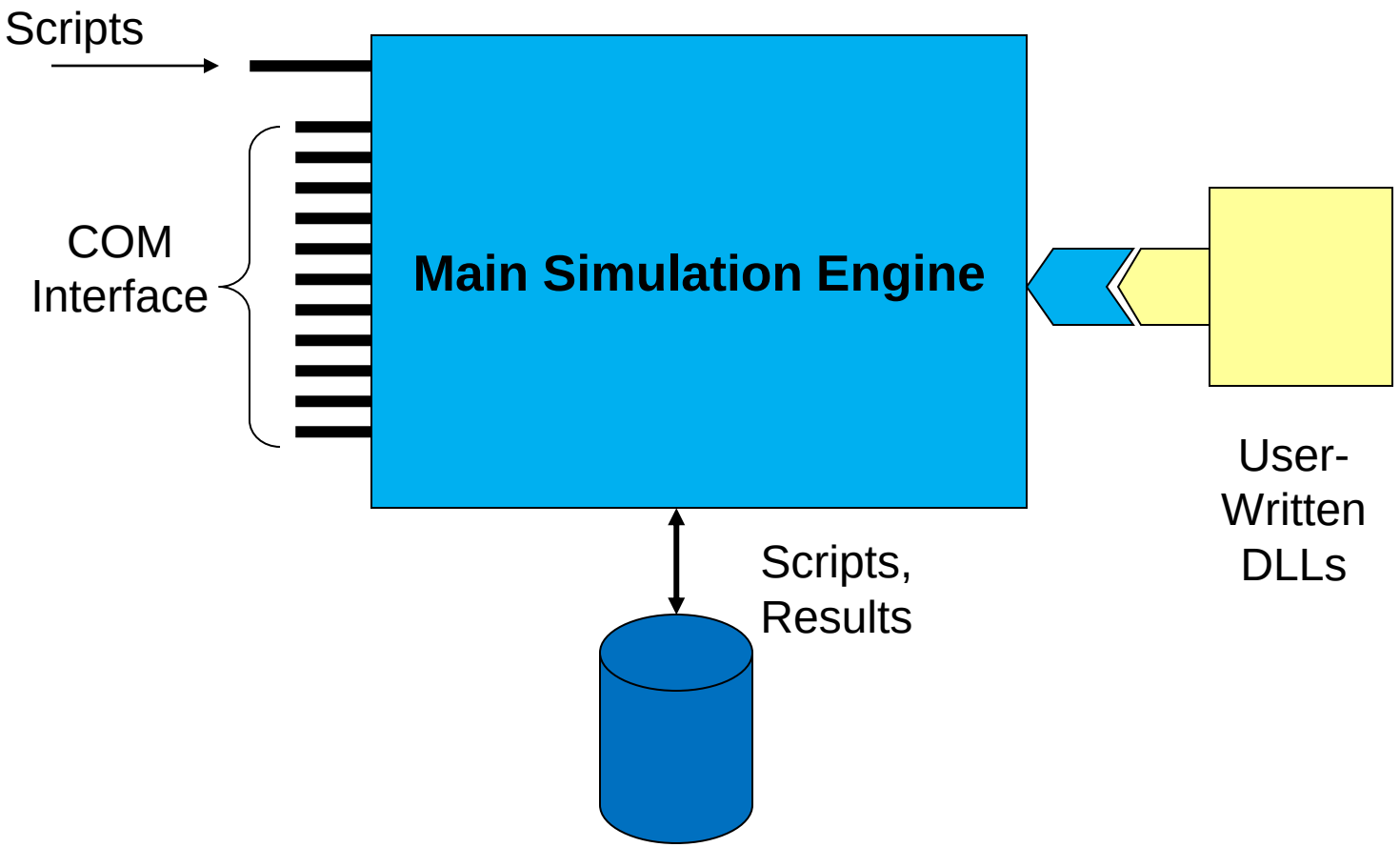
Overall Model Concept



User Interfaces Currently Implemented

- A **stand-alone executable** program (**OpenDSS.exe**) 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.
  - **OpenDSSEngine.DLL**
- A **direct DLL** interface (**OpenDSSDirect.dll)** that mimics the COM interface with a standard call library
  - For non-Windows platforms, such as HPCs
  - For programming languages that do not support COM or are not efficient at supporting COM
- Some users have published Python interfaces

DSS Structure



OpenDSS Research Directions

- Exploiting **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.
  - OpenDSS Version 8 Released in 2019 (now Version 9) performs parallel processing for faster QSTS simulations
  - Can solve different time periods in parallel
  - Employs Diakoptics to solve very large networks
- Improved Distribution Management Systems (DMS) functions
  - Several Existing EPRI research projects
- Improved distribution state estimation functions
- Improved **microgrid simulations** (Dynamics mode)
  - Hot topic currently

Built-in Solution Modes

- Snapshot (static) Power Flow
- Direct (non-iterative solution of I=YV)
- 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
- Harmonics
- Custom user-defined solutions

Input Data Requirements

- OpenDSS is a Script-driven circuit solution engine
- The OpenDSS was designed in 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.

Input Data Requirements

- OpenDSS Circuit elements have many properties
  - The program can accept more detailed data for lines, transformers, etc. than the typical data for distribution system analysis when they are available.
- Nearly all have reasonable **default values**
- Users only need to specify the property values that are
  - Different than the default values, and
  - Used in the circuit simulation

Variable States

- Throughout OpenDSS, property values remain at the value most recently defined until they are subsequently changed
  - They generally do NOT reset to original values unless you explicitly change them or Clear the circuit definition
    - This may be different than distribution power flow programs you are familiar with
    - OpenDSS is designed as an interactive *simulator*

Advanced Types of Data in OpenDSS

- 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,
- TCC Curves and other data for Fuse, Relay, and Recloser objects
- Machine data.

Equations

For the curious Electrical Engineer …

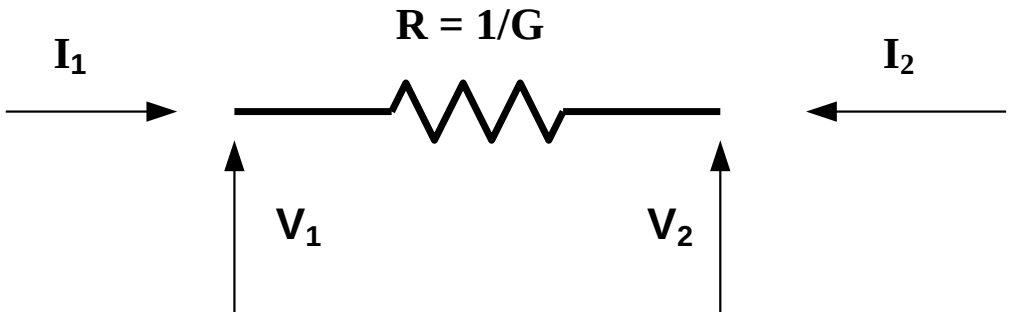
The Math …

- Nearly everything results in a **matrix** or **array**
  - **Nodal Admittance** formulation
  - Circuit elements modeled by *primitive admittance* matrices
    - Y_{prim}
  - **Primitive Y** matrices are used to build the **System Y** matrix

- OpenDSS Works In
  - Phase domain
  - Actual volts and amps
  - Symmetrical components and per units not used *inside* the program !! -- Input and output only!

Primitive Y Matrix

- Simple Resistor

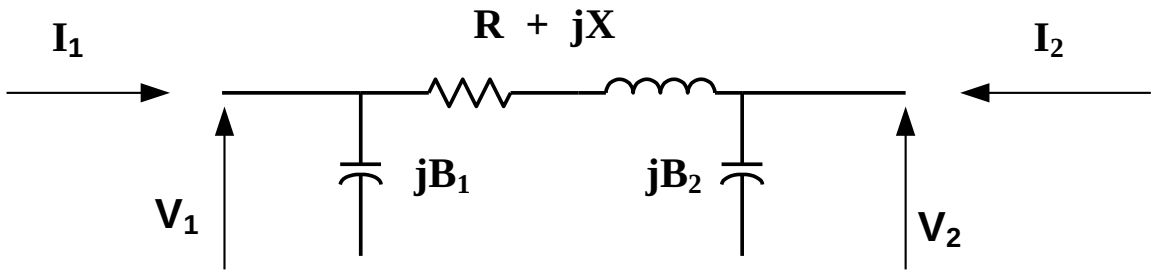


$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} G & -G \\ -G & G \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$$

$$Y_{prim}$$

Primitive Y Matrix, cont'd

- LINE model



$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} (R+jX)^{-1} + jB_1 & -(R+jX)^{-1} \\ -(R+jX)^{-1} & (R+jX)^{-1} + jB_2 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$$

$$Y_{prim}$$

What about 3-phase elements?

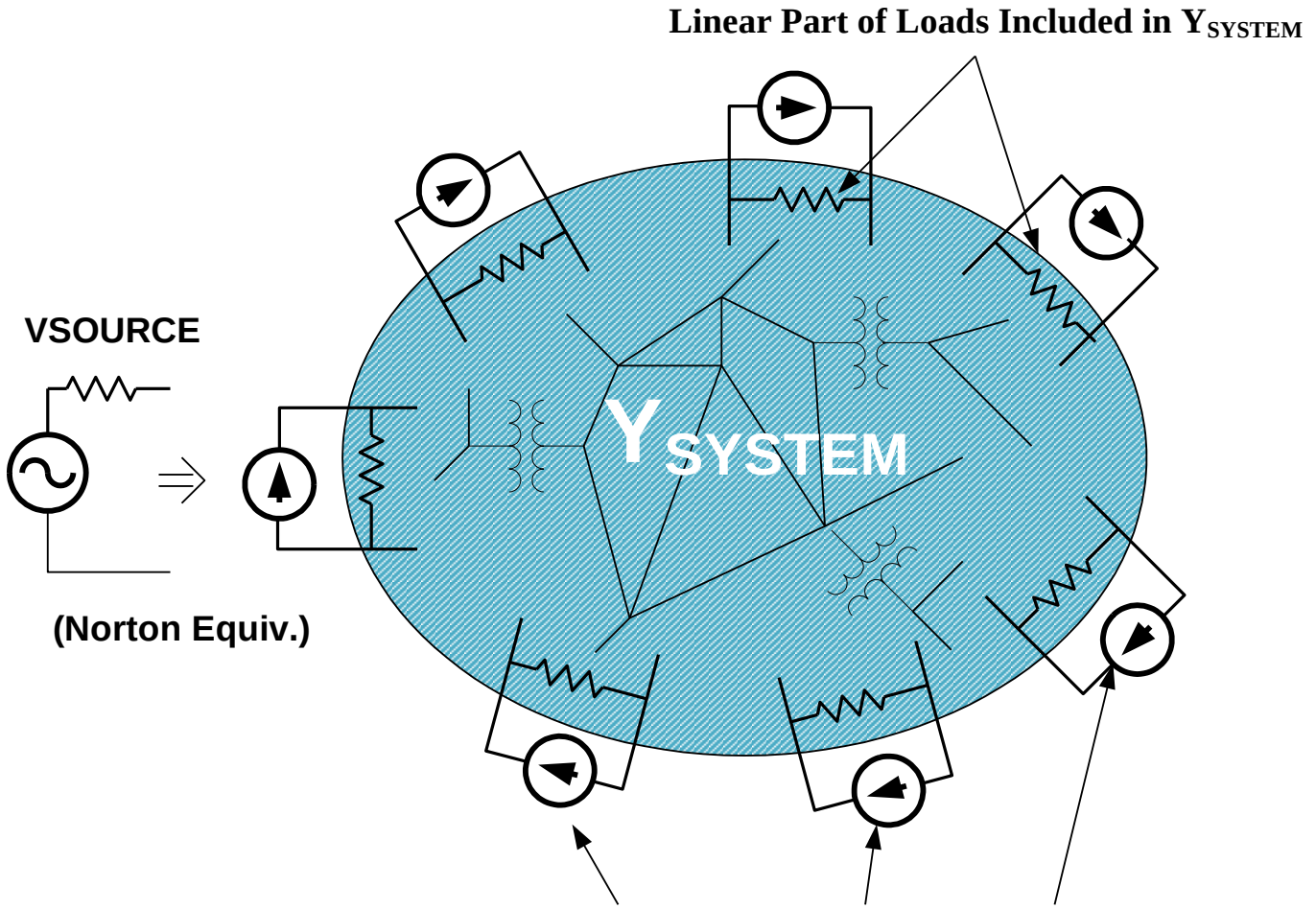
- Simply let **R, X, B, G, C**, etc. represent **3x3** matrix
  - Notation stays the same

- And it works!

- I1, I2, V1, V2 etc become 3x1 vectors

- This is basically how all the Circuit Element (CktElement class) models in OpenDSS work.

The Network Model



Nodal Admittance Equations

$$\begin{bmatrix} I_1 \\ I_2 \\ \cdots \\ I_S \\ \cdots \\ I_{L1} \\ \cdots \\ I_{L2} \\ \cdots \\ I_N \end{bmatrix} = \begin{bmatrix} Y_{SYSTEM} \\ \\ N \times N \\ (Sparse) \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ \cdots \\ V_S \\ \cdots \\ V_{L1} \\ \cdots \\ V_{L2} \\ \cdots \\ V_N \end{bmatrix}$$

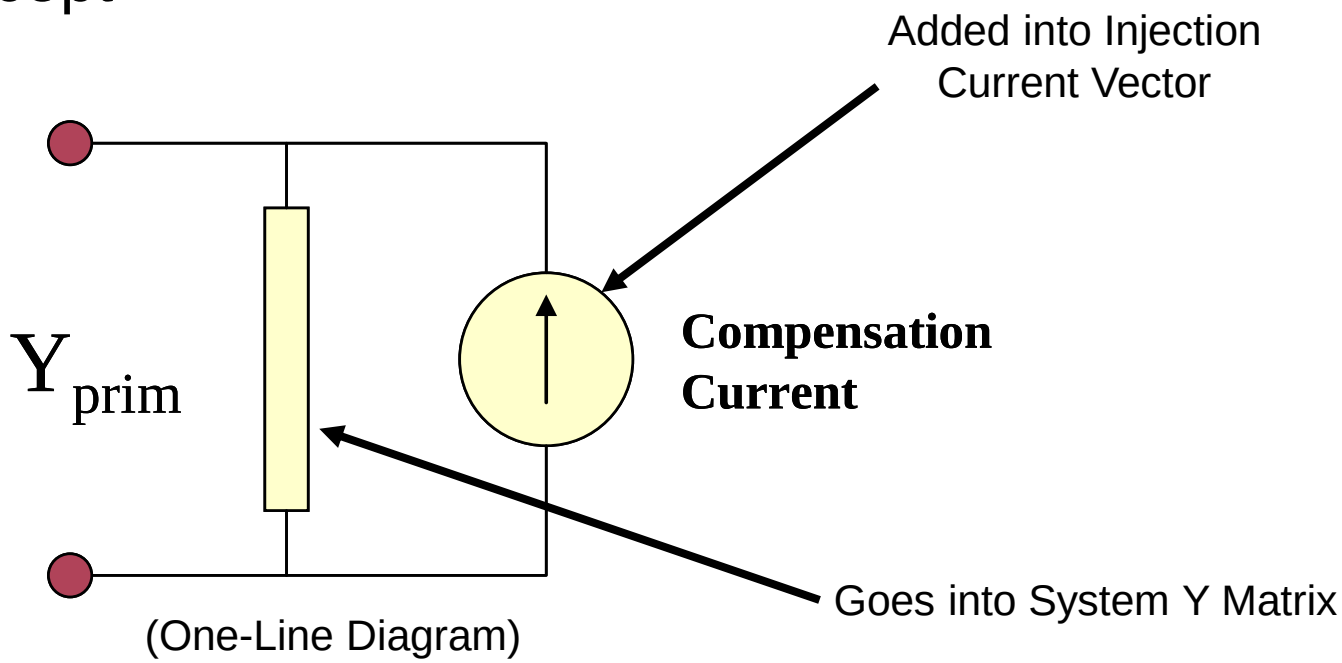
N = Number of NODES (not BUSES)

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
- … And is Fast!

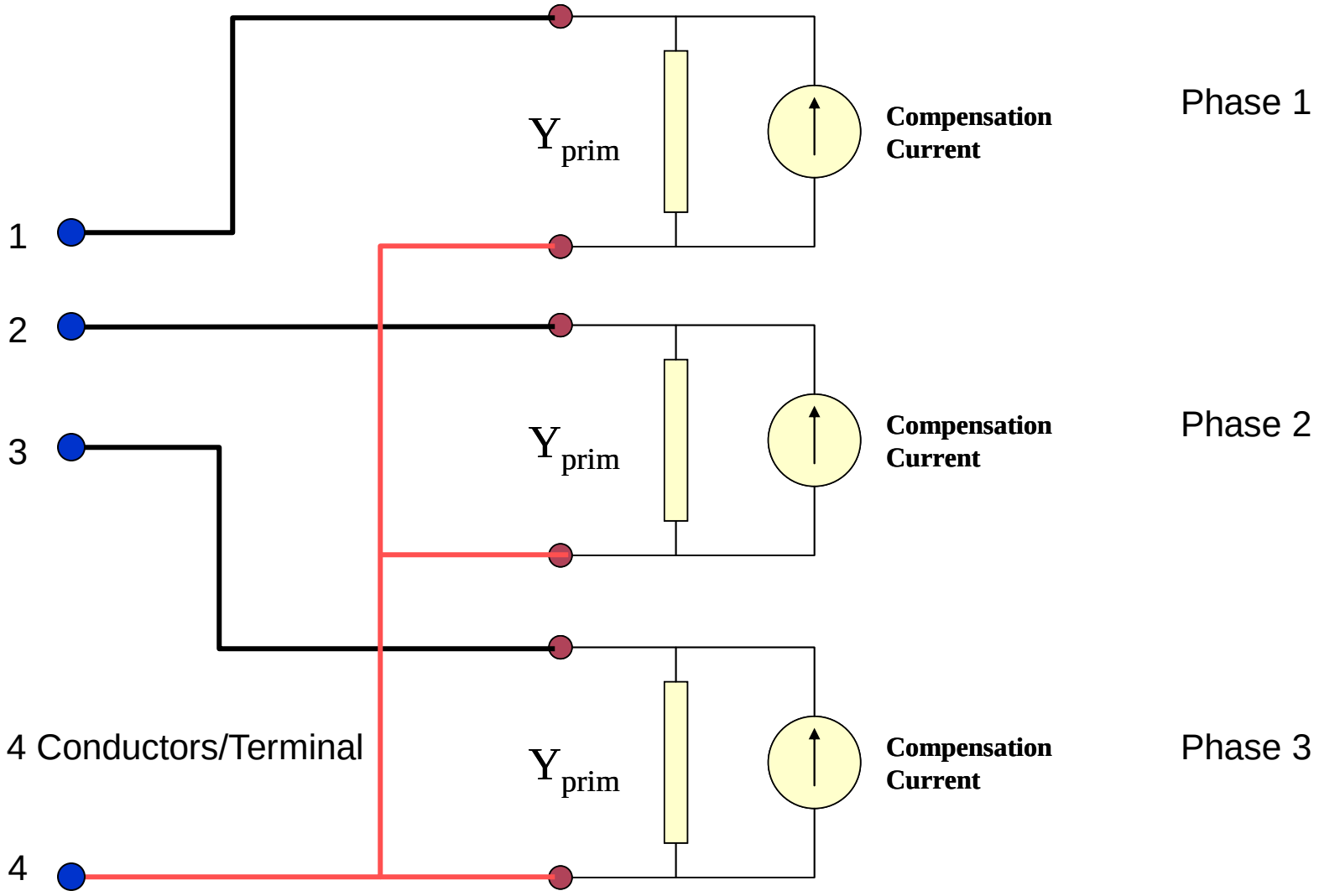
Load (a PC Element)

General Concept

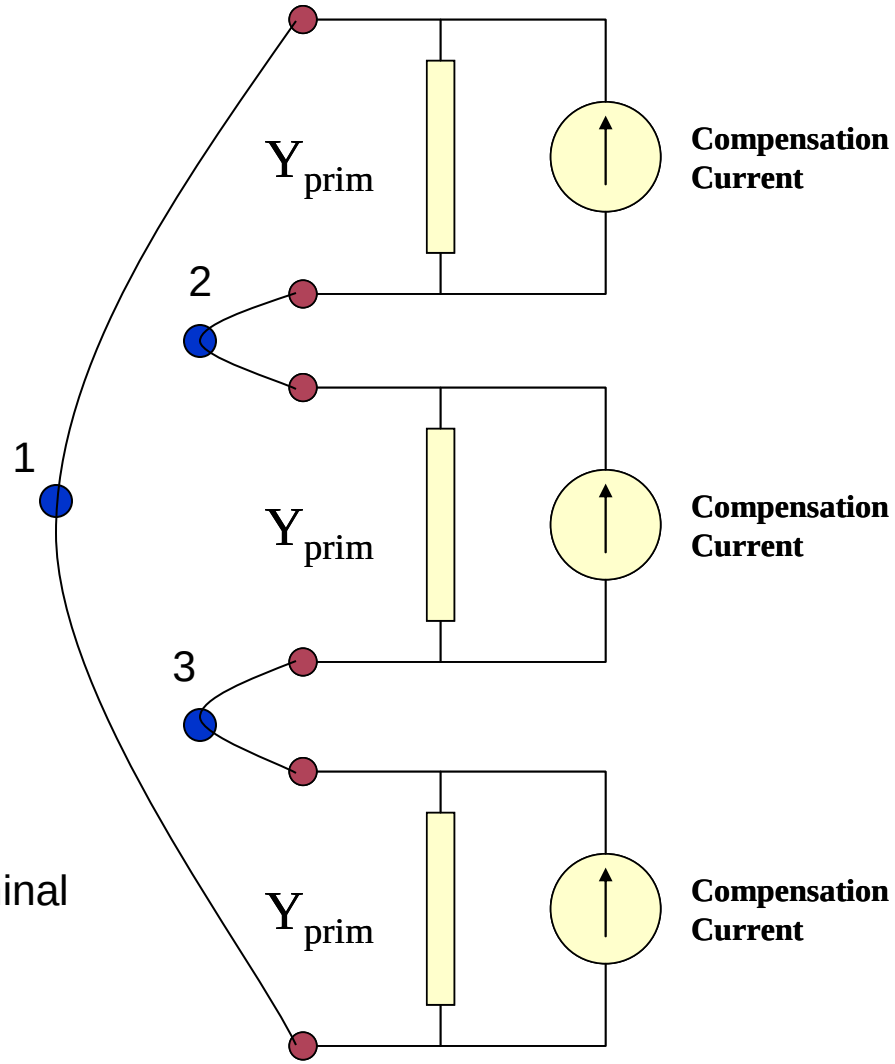


Most Power Conversion (PC) Elements are Modeled Like This

Load - 3-phase Y connected



Load - 3-phase Delta connected



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

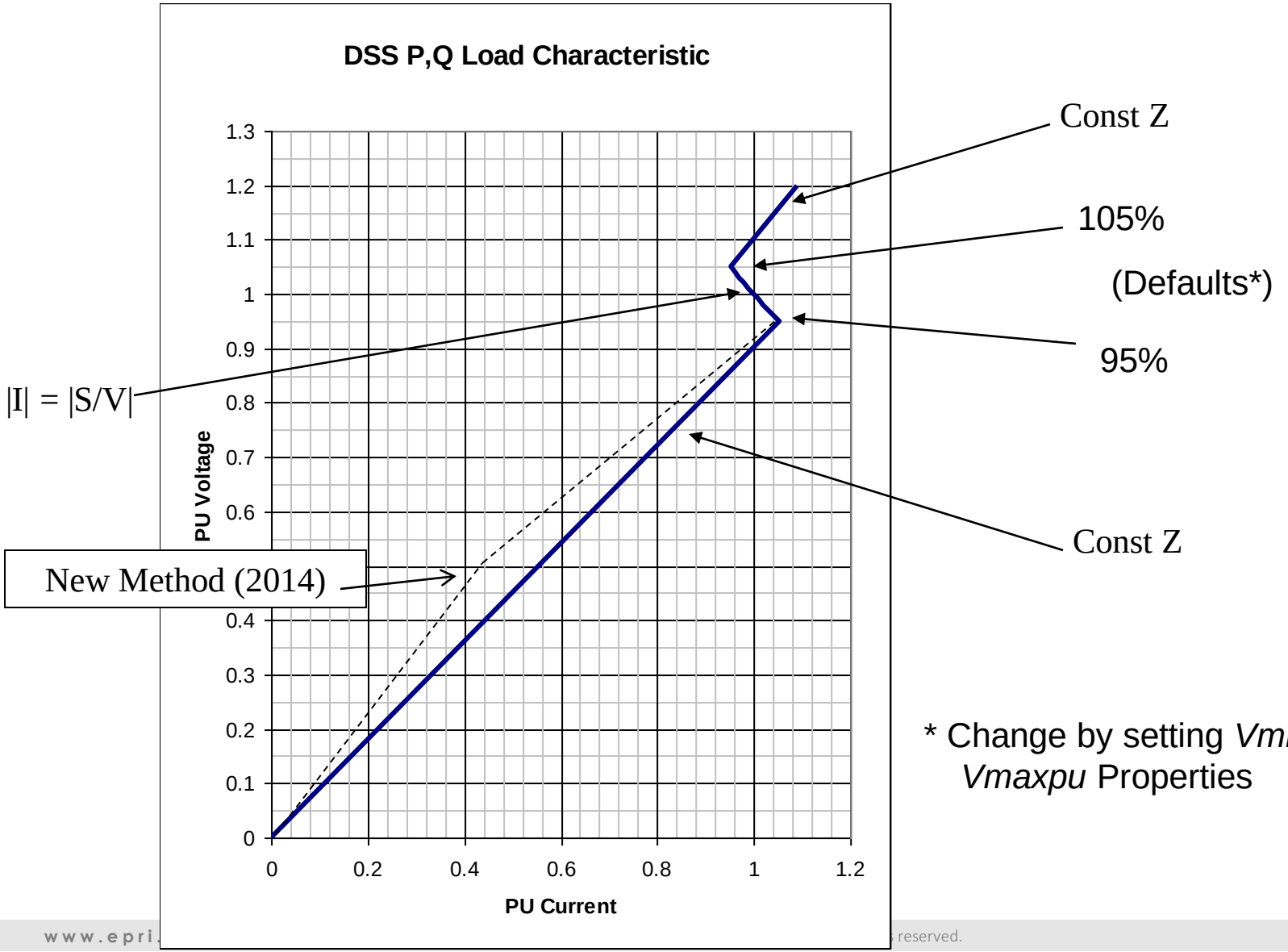
7:Const P, Fixed Impedance Q

8: Special ZIP load model

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
  - This script changes 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)



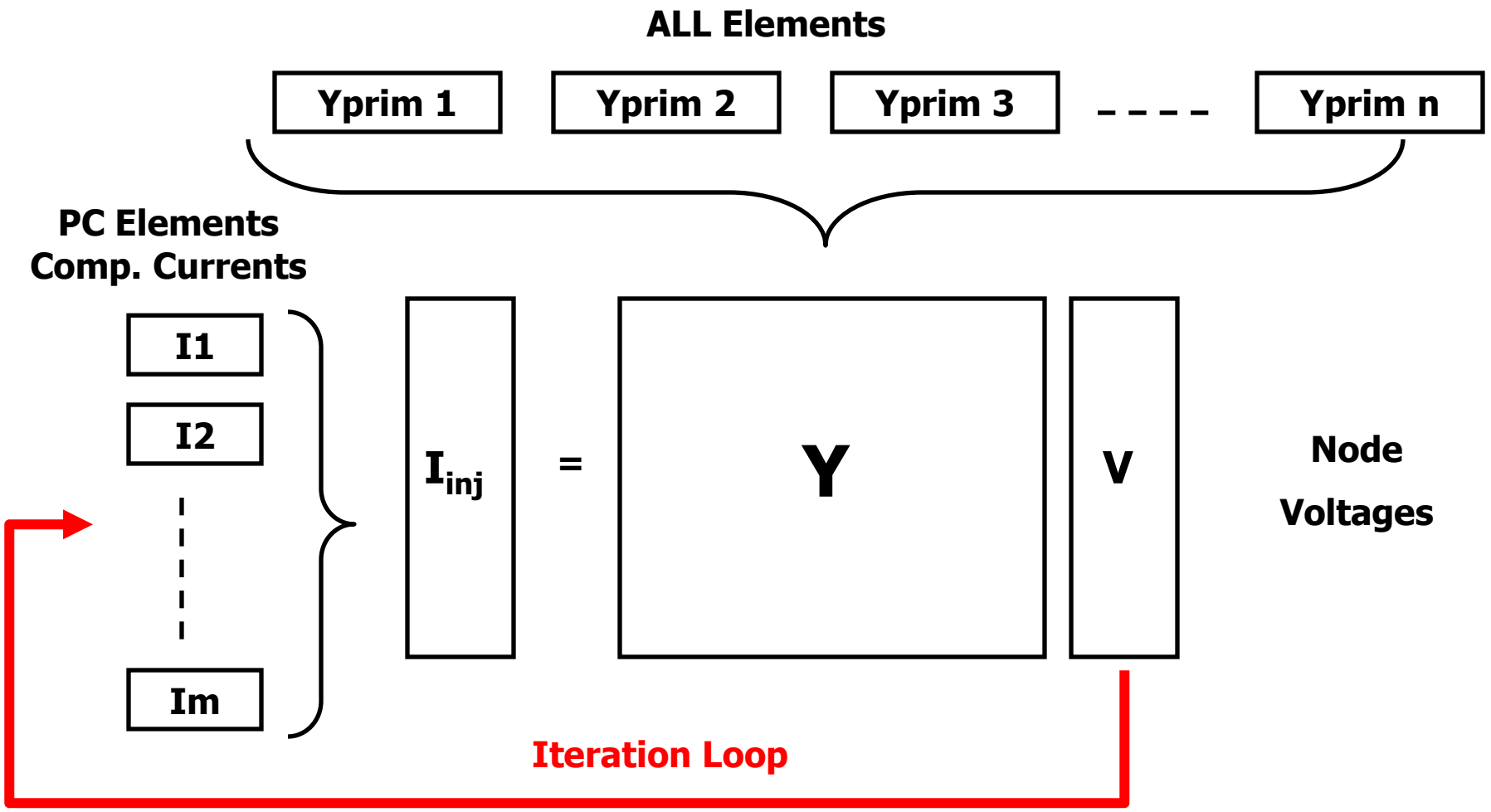
* Change by setting *Vminpu* and *Vmaxpu* Properties

Power Flow Solution Algorithm

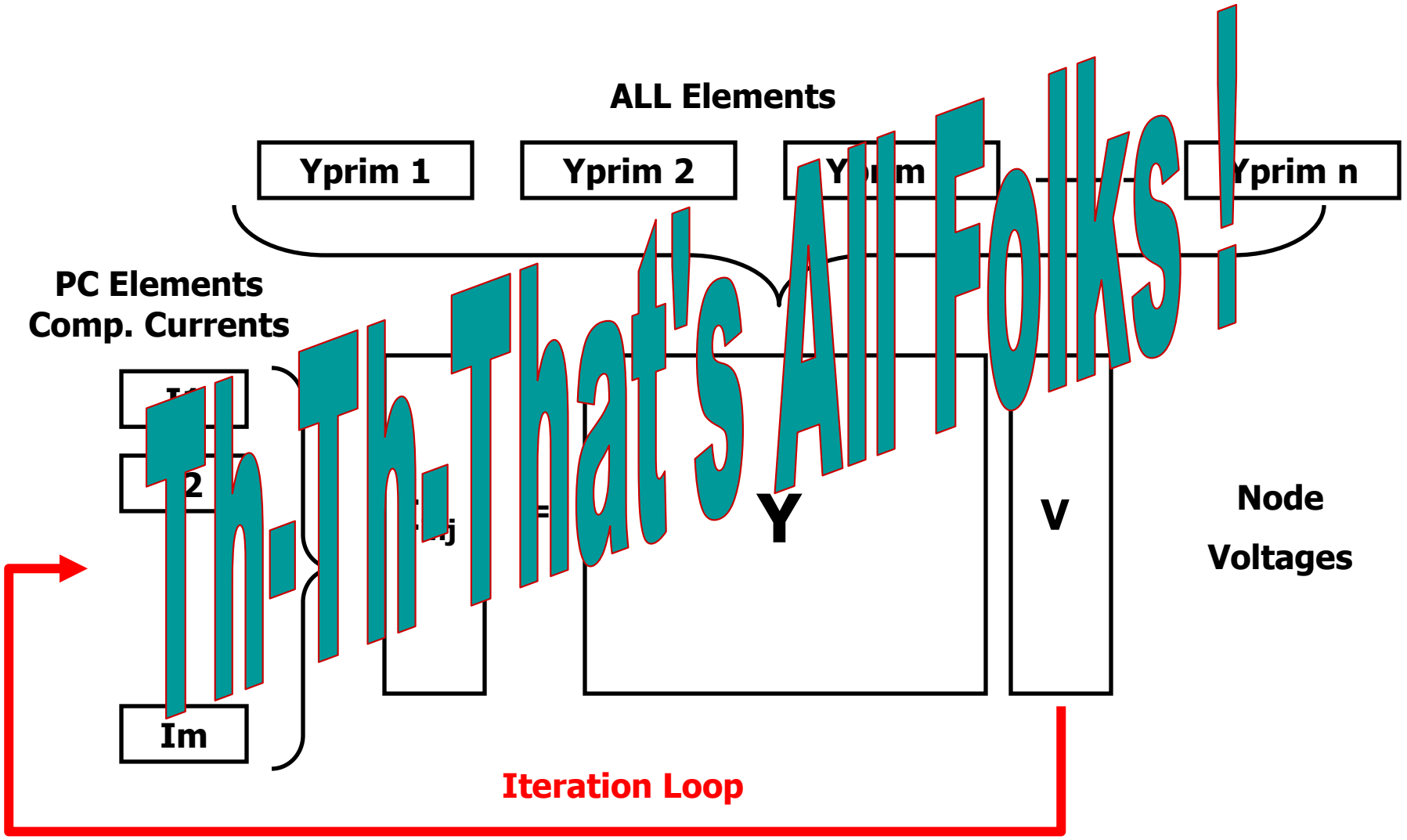
1. Initial Guess at Node Voltages, $\boldsymbol{V}$
2. Compute all Injection (Compensation) Currents, $\boldsymbol{I}$
   a. For PC Elements
3. Solve for new guess at $\boldsymbol{V}$
4. Repeat 2 and 3 until Converged

- Convergence is based on change in per unit voltage magnitude
  - Default tolerance = 0.0001
  - Good enough for distribution systems

Putting it All Together



Putting it All Together



A More Concise Form …

- Fixed-point solution form for normal solution

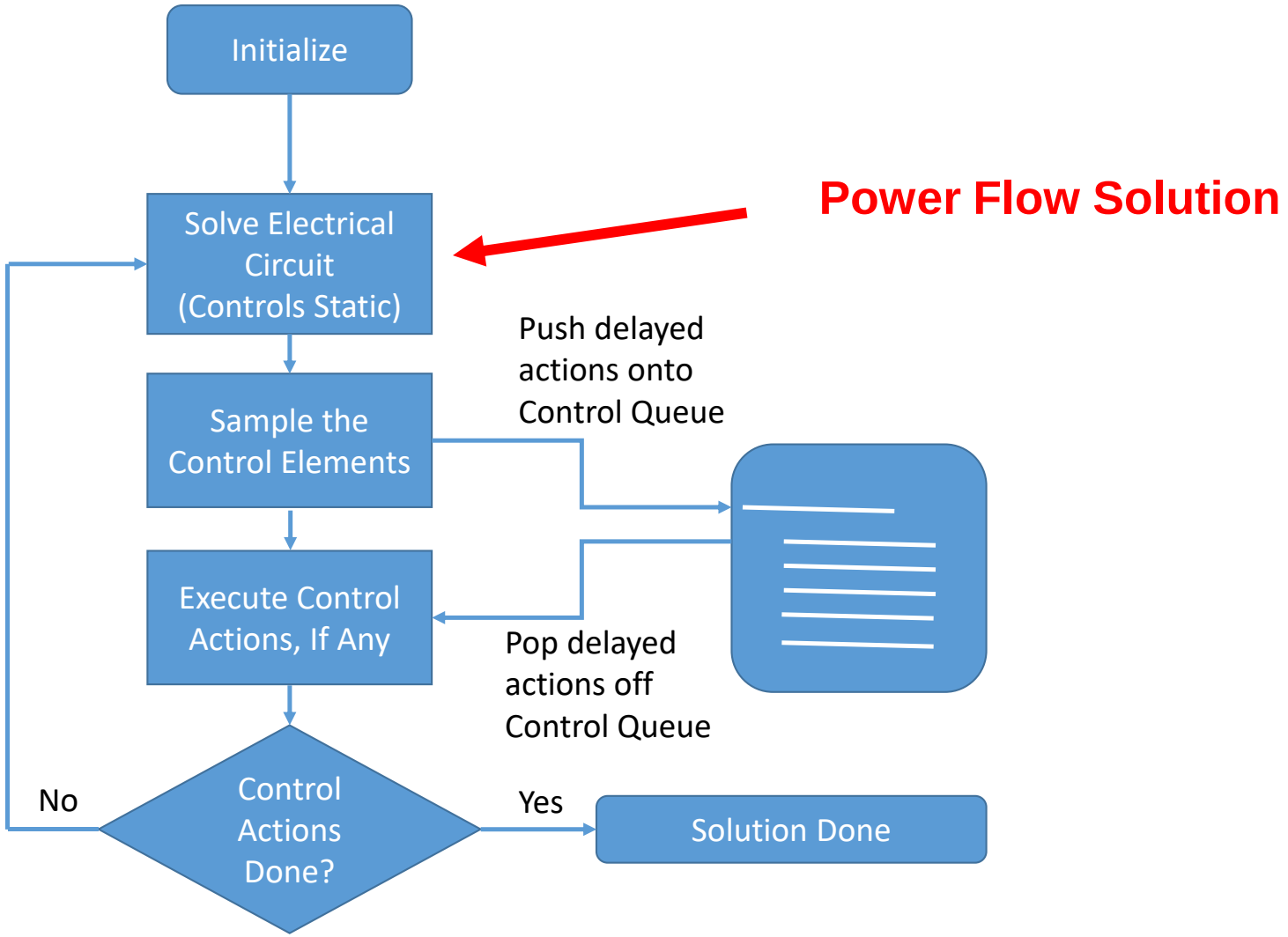
$$V_{n+1} = [Y_{system}]^{-1} I_{PC}(V_n) \quad n = 0, 1, 2, \ldots$$

…until converged

$I_{PC}(V)$ = *compensation* currents from Power Conversion (PC) elements in the circuit as a function of voltage

OpenDSS Solution Loop with Controls

Controls are sampled and executed after a converged power flow solution

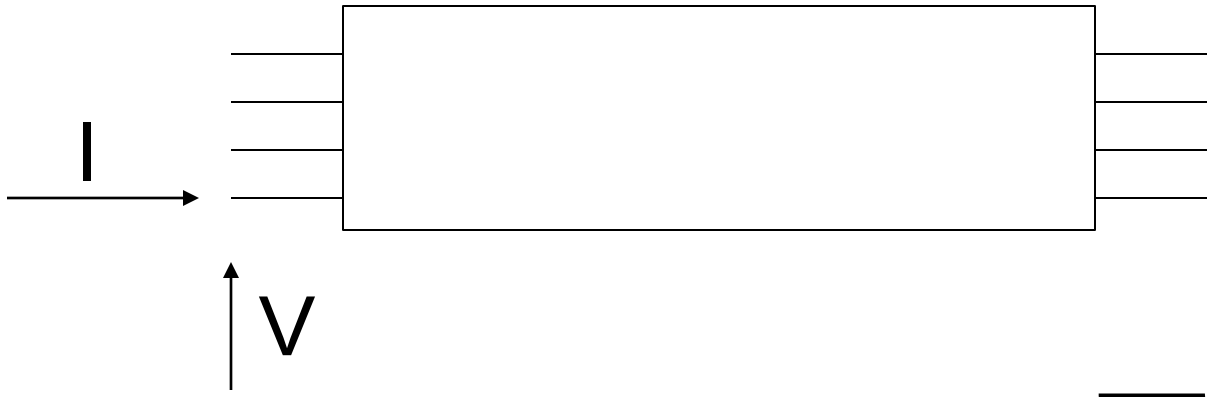


Solving the Power Flow …

- 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 the same as the “Newton-Raphson Power Flow”

Losses are computed quite simply for any device model

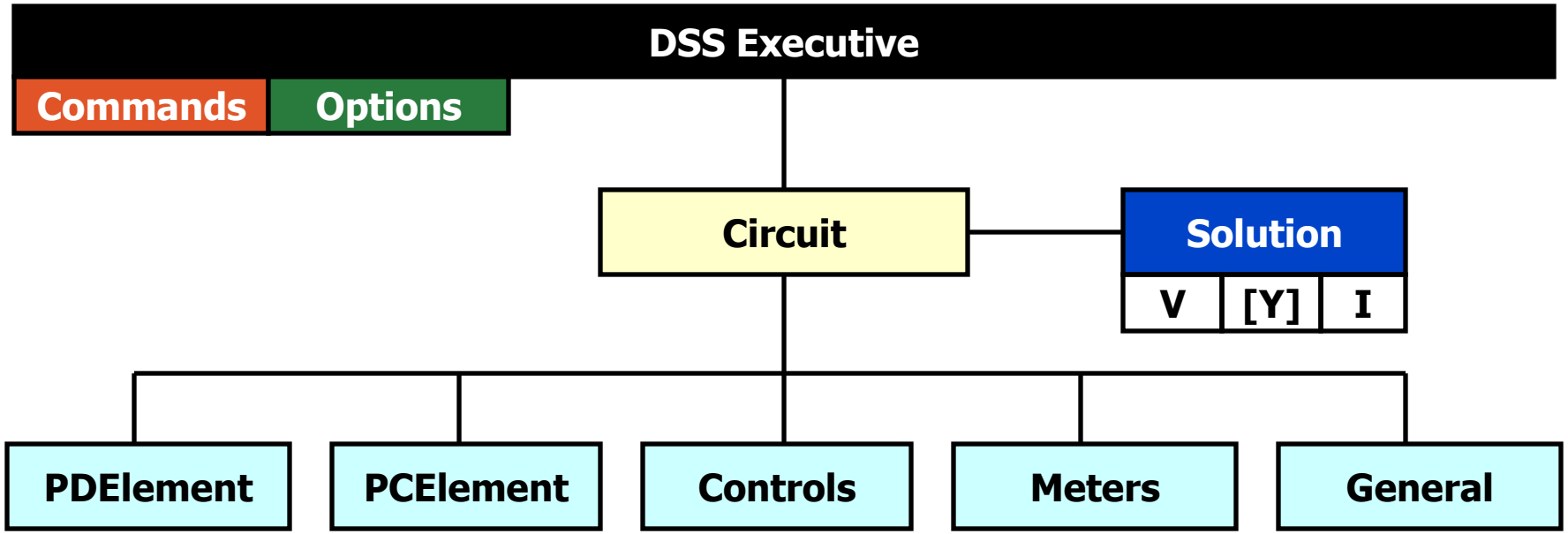
- Sum the powers into each conductor and losses are the power left over (not summing to zero)



$S_i = V_i I_i^* = P_i + jQ_i$

$$S_{loss} = \sum S_i$$

DSS Object Structure



DSS Class Structure