Introduction to The Next Generation of Distribution Analysis Tools

Summer course D1

Davis Montenegro Martinez Ph.D.
Engineer/Scientist III

Universidad de los Andes
Bogotá, Colombia
June 4 – 7, 2019

Electric Power Research Institute - EPRI

Together...Shaping the Future of Electricity

Advancing safe, reliable, affordable, and environmentally responsible electricity for society through global collaboration, thought leadership, and science and technology innovation

Areas of focus

3420 Hillview Avenue, Palo Alto, California 94304-1338 • PO Box 10412, Palo Alto, California 94303-0813 USA
800.313.3774 • 650.855.2121 • askepri@epri.com • www.epri.com

The instructor

Davis Montenegro, *Member, IEEE*

Davis Montenegro-Martinez serves as Engineer/Scientist III at the Electric Power Research Institute (EPRI) in the areas of power system modeling, analysis and high performance computing. He received his B.Sc. degree in electronics engineering from Universidad Santo Tomás, Bogotá, Colombia (2004); he is M.Sc. in electrical engineering from Universidad de los Andes, Bogotá , Colombia (2012). He received his Ph.D. in electrical engineering from Universidad de los Andes (2015), and a Ph.D. in electrical engineering from the University Grenoble-Alpes, France (2015).

Before joining EPRI, Davis served for 10 years as a lecturer for Universidad Santo Tomas in Colombia, during this time he was also technology consultant in the areas of industrial automation, software and electronic hardware design focused in the electric power industry, specifically in monitoring and control for meter calibration laboratories. His expertise in parallel computing techniques is being used at EPRI for incorporating multi-core processing to power system analysis methods such as QSTS, reducing the computational time required to perform these analysis using standard computing architectures.

Dr. Montenegro is also a member of the International Council on Large Electric Systems CIGRE, he was awarded with the IEEE 2016 I&CPS Ralph H. Lee Department Prize Paper Award at the 2017 I&CPS Technical Conference Awards luncheon in Niagara Falls, ON, Canada, for the paper titled "Energy Storage Modeling for Distribution Planning." He was also awarded an IEEE recognition in 2017 for notable services and contributions towards the advancement of IEEE and the engineering professions chairing of IM09 (Instrumentation and Measurements) Society Chapter, Colombian Section 2015–2017.

Agenda

Course content

- The current scenario
- Traditional planning
- The tools for the new challenges
  - Quasi-Static-Time-Series (QSTS) Simulation
    - Yearly based simulations
    - Daily based simulations
- Getting deeper into the issue
  - Parallel processing
    - Understanding the processor
    - Temporal parallelization

Course content

- Spatial parallelization
- Actor based Diakoptics (A-Diakoptics)

- Introduction to OpenDSS -> planning command suite
  - The energy Meter
  - The Demand Interval Files
  - Mitigation strategies
  - The parallel processing suite
  - Smart inverters
  - PV Systems and smart inverters
  - Storage Controller

Course content

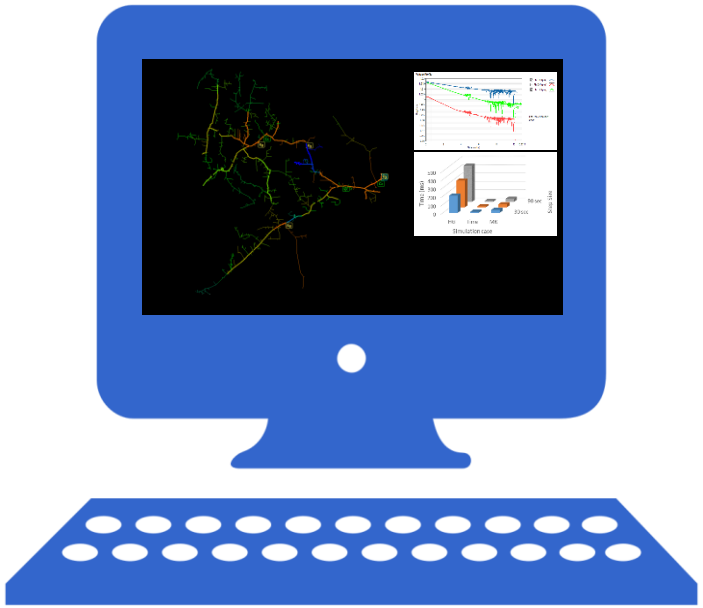
- Introduction to OpenDSS-G
  - Integrated planning tools
  - Using storage devices
  - Using smart inverters
  - Re-conductor
  - Co-Simulation
  - Coordinating Distributed resources
  - Creating wide area controllers
- Conclusions

Course distribution

Day	Summary
4- Jun	Introduction to the course program, the needs and goals that motivate the development of new tools for analyzing distribution power systems. The tools to be used during the course will also be introduced during this day, proposing exercises and study cases to facilitate their use during the next three days.
5- Jun	This day we will be introduced to quasi-static-time-series (QSTS) simulations, their effects on distribution system analysis and the output provided. The aim is to validate the model features in terms of control devices across the power system, their setup and coordination for time-based analysis. Also, we will be introduced to parallel processing, the architecture of multicore computers and how to handle the complexity introduced by them using adequate frameworks and simulation methods.
6- Jun	This day we will introduce Distributed Energy Resources (DER) as an alternative for mitigating undesired events in distribution power systems. The aim during this session is to visualize how DER can be used to relieve the power system during peaks of demand or stress situations using QSTS simulations and their data output.
7- Jun	This day we will be focused on co-simulation techniques, the aim is to introduce the audience to real-time and fast simulation for covering different simulation needs. During this session the audience will be able to put into practice all the knowledge acquired during the previous sessions through multiple study cases and exercises proposed.

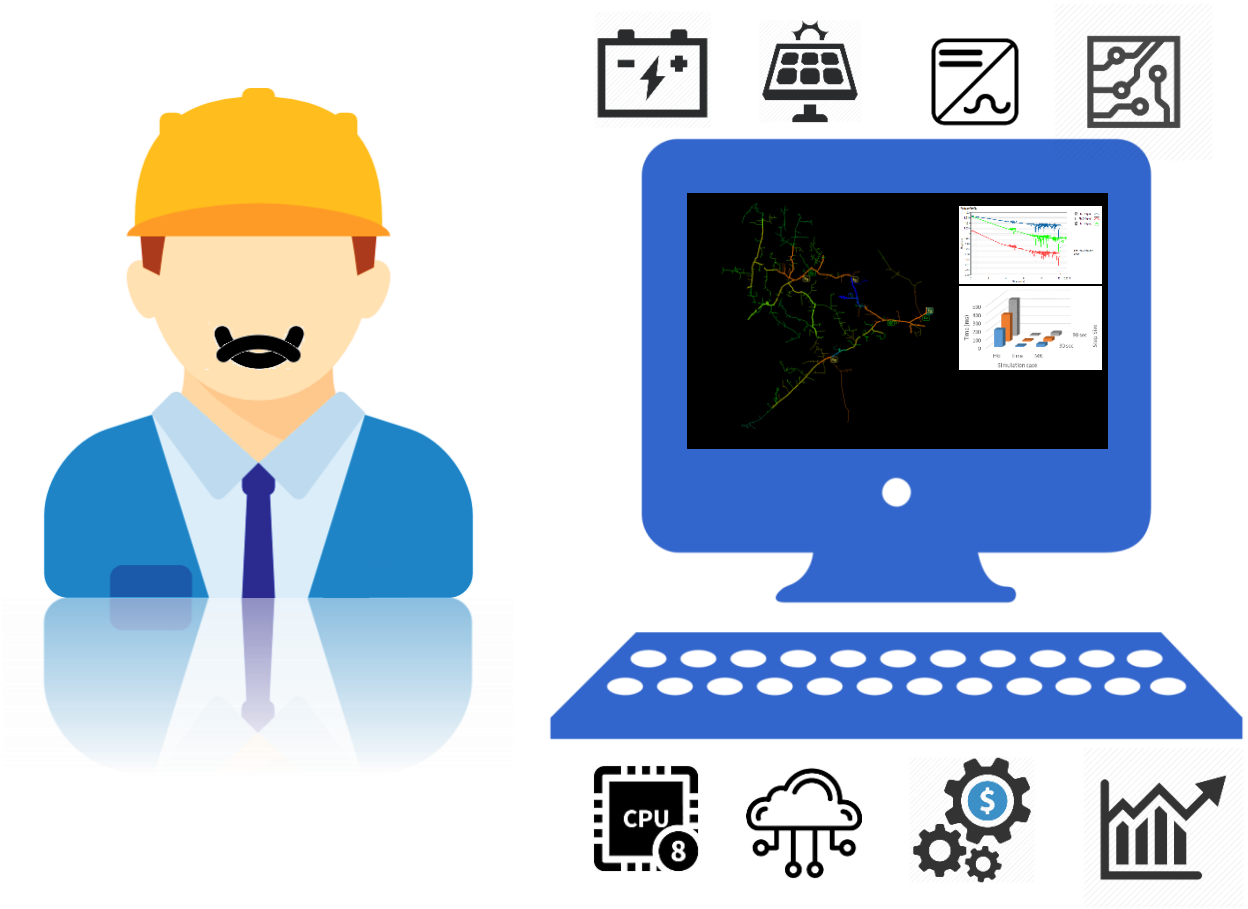
The current scenario

The current scenario



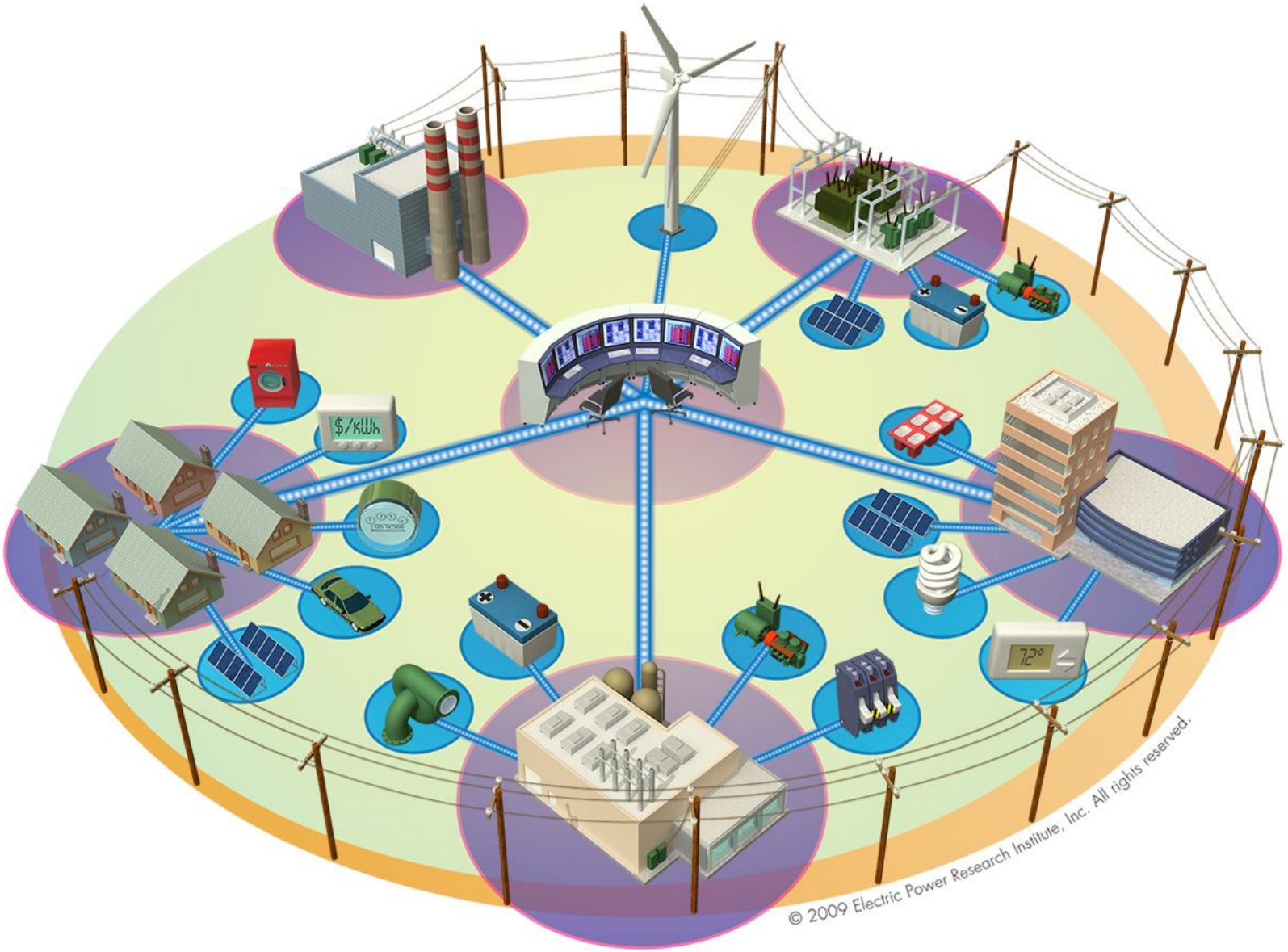
- Snap shot simulation
- Extreme values
- Simple and efficient
- Seemed to have everything

The current scenario



- Accelerate advanced analysis
- Varying operational conditions
- Big data processing
- The evolution is happening

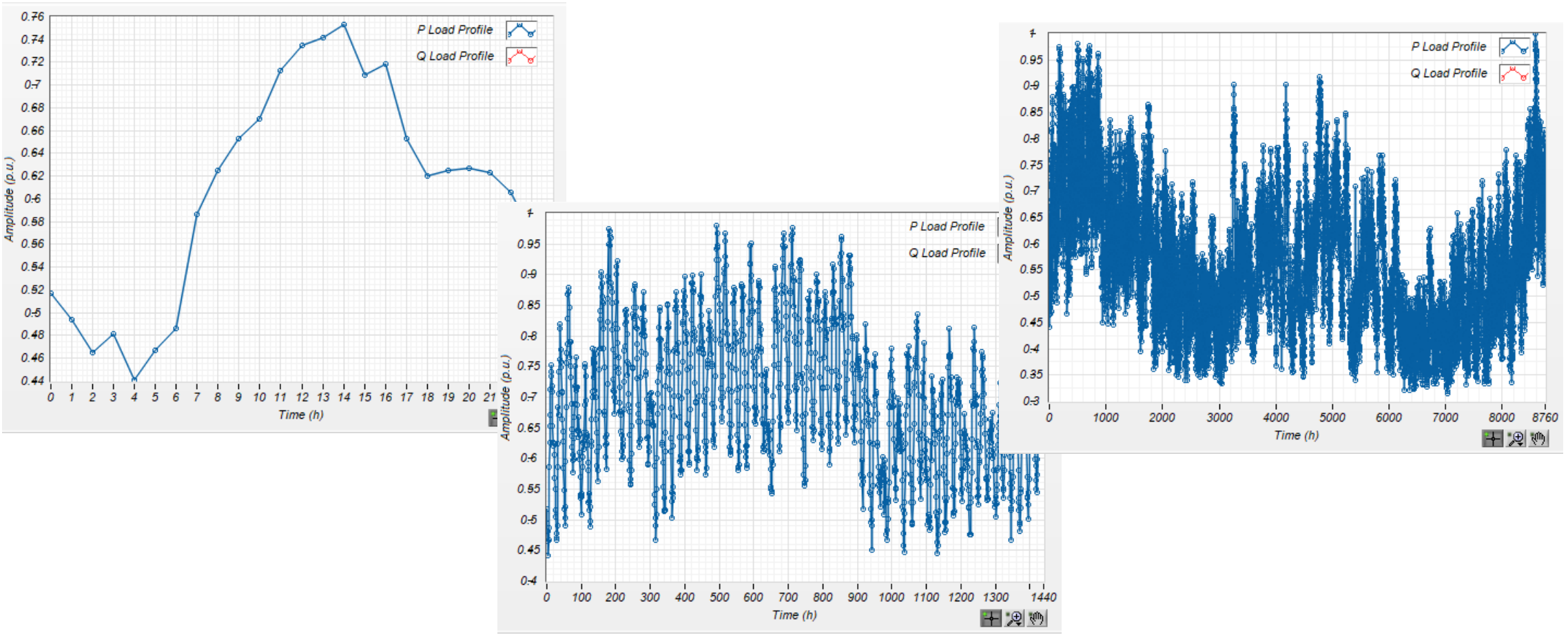
The current scenario



The tools for attending the new challenges

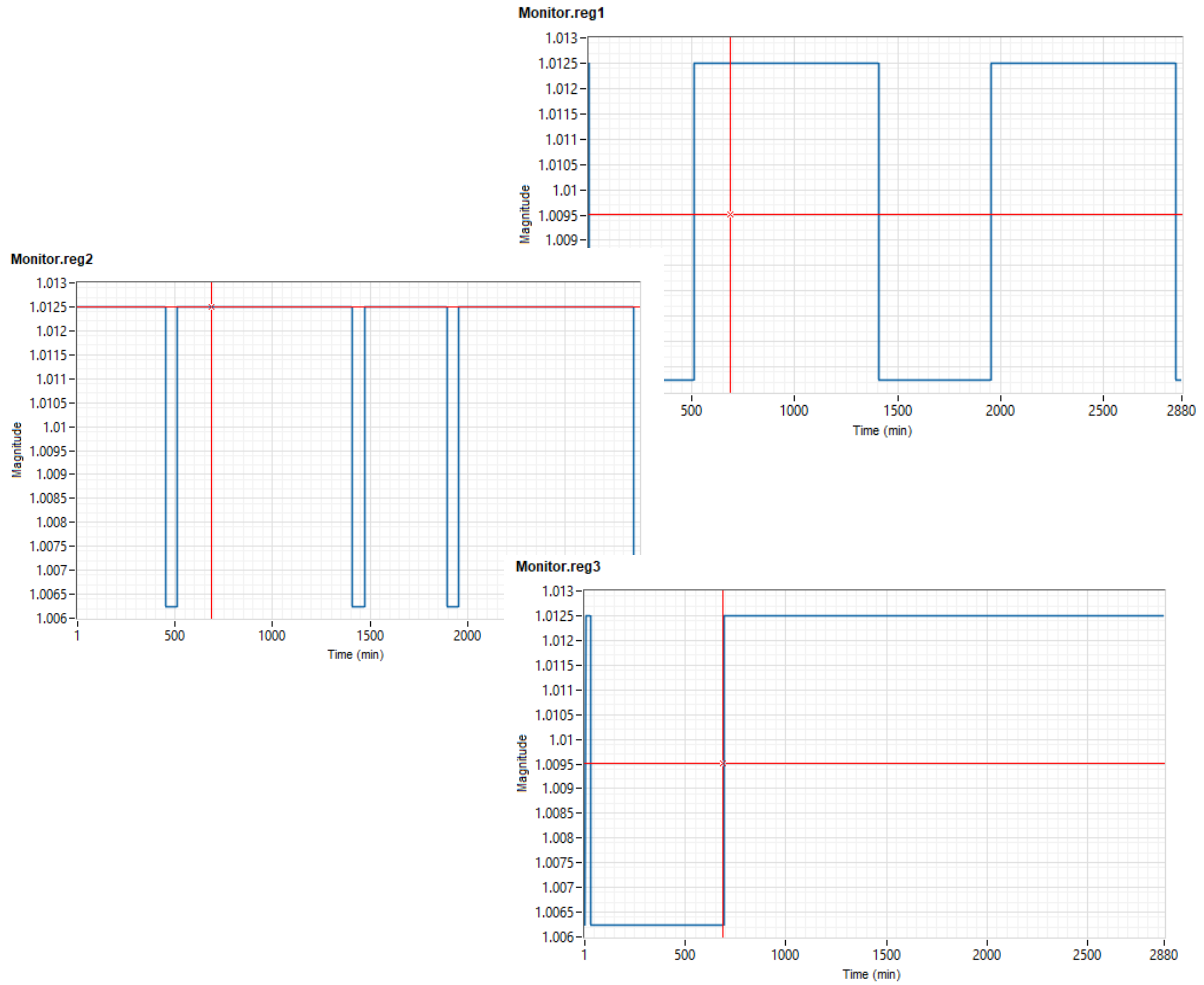
The tools for attending the new challenges

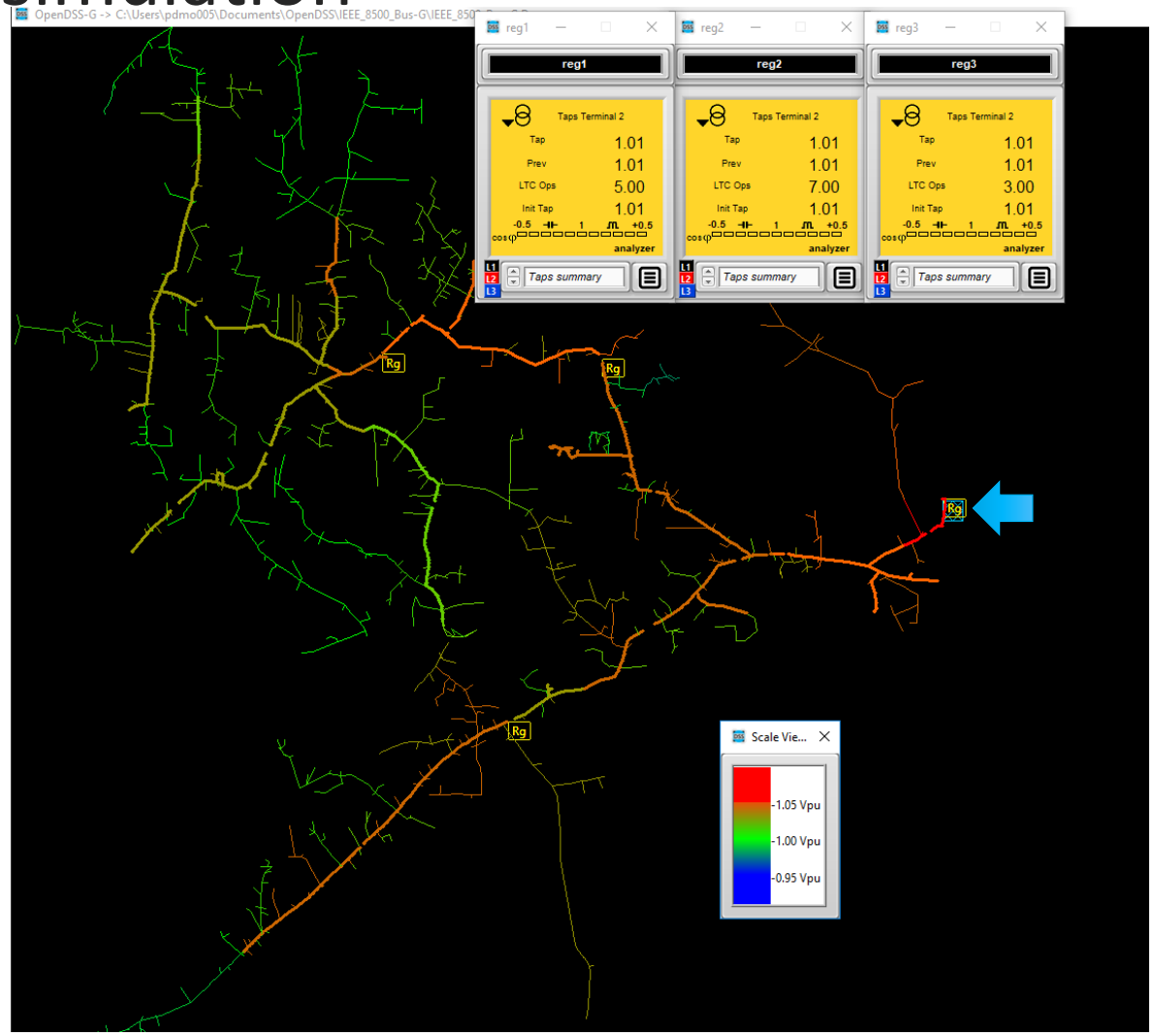
- Quasi-Static-Time-Series (QSTS) simulation



The tools for attending the new challenges

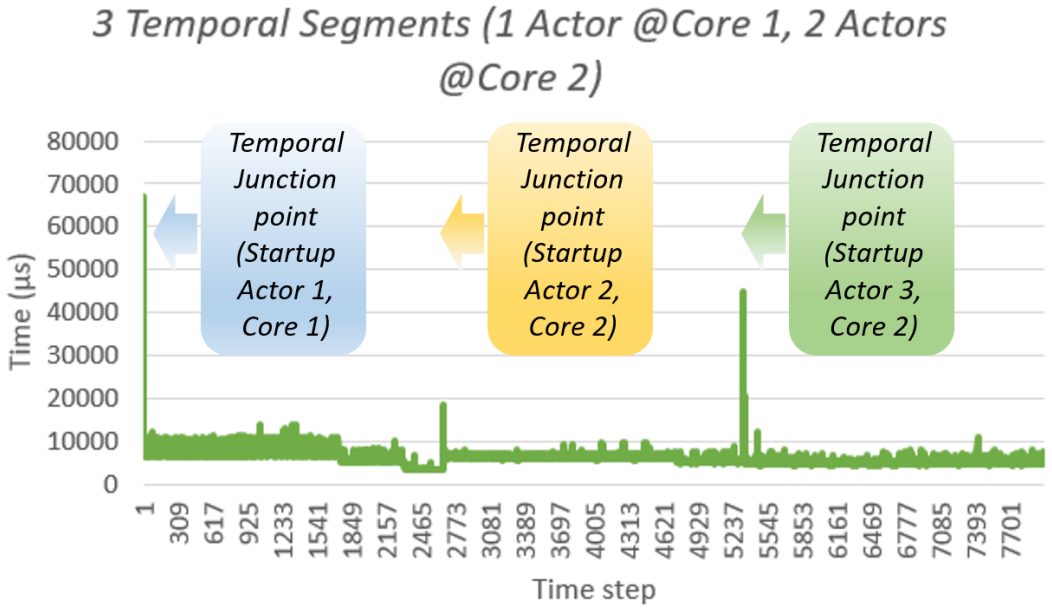
- Quasi-Static-Time-Series (QSTS) simulation

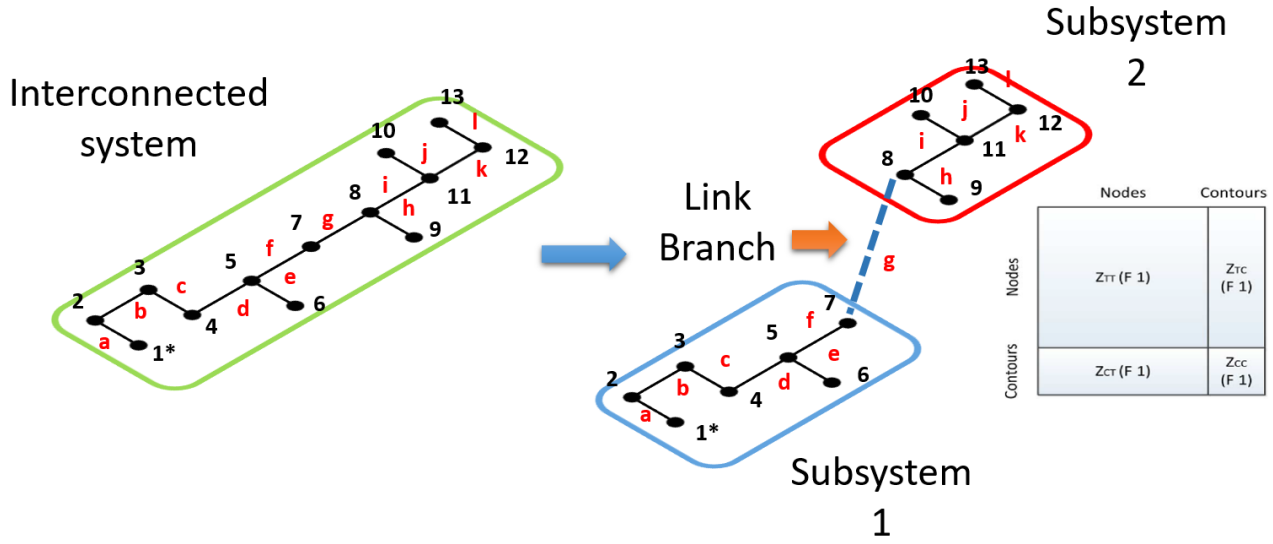




The tools for attending the new challenges

- Catching up with technology

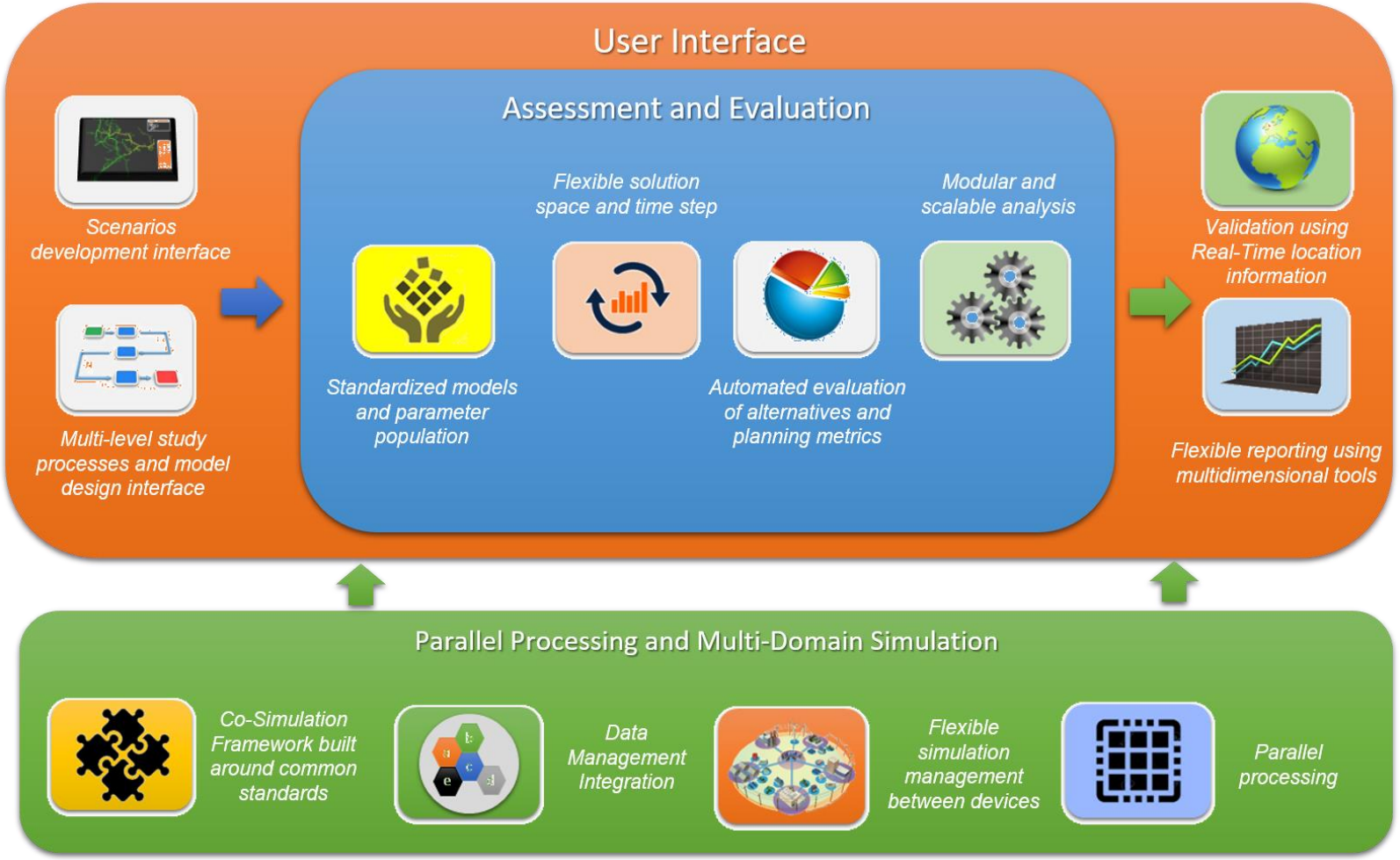




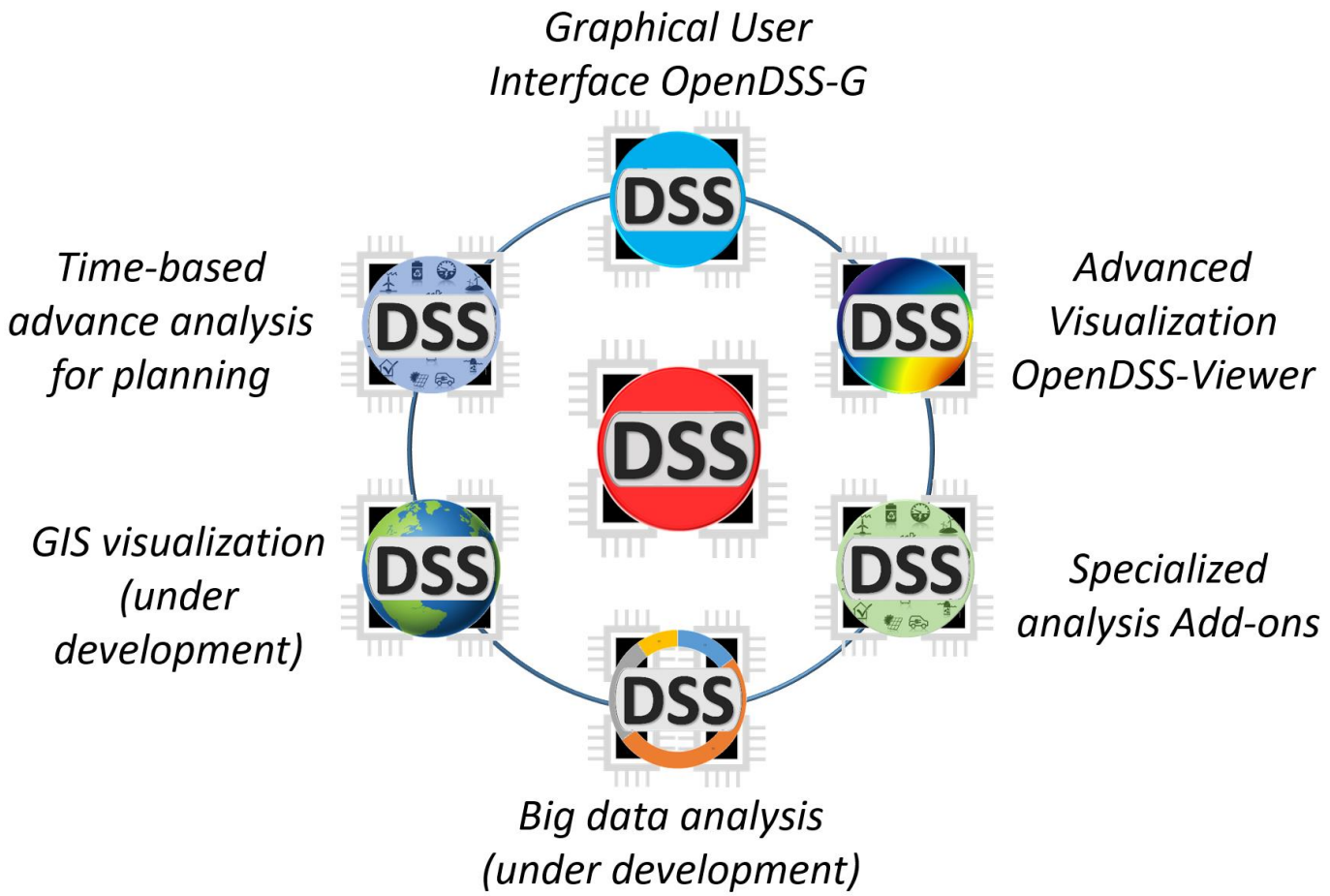
And other techniques that can be used to take advantage of multicore computers when using simulators

The tools for attending the new challenges

- Catching up with technology



The tools for attending the new challenges



https://www.epri.com/#/pages/sa/opendss?lang=en

Introduction to OpenDSS

Introduction to OpenDSS

- OpenDSS – Open-Source Distribution System Simulator
  - 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

Introduction to OpenDSS

- 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

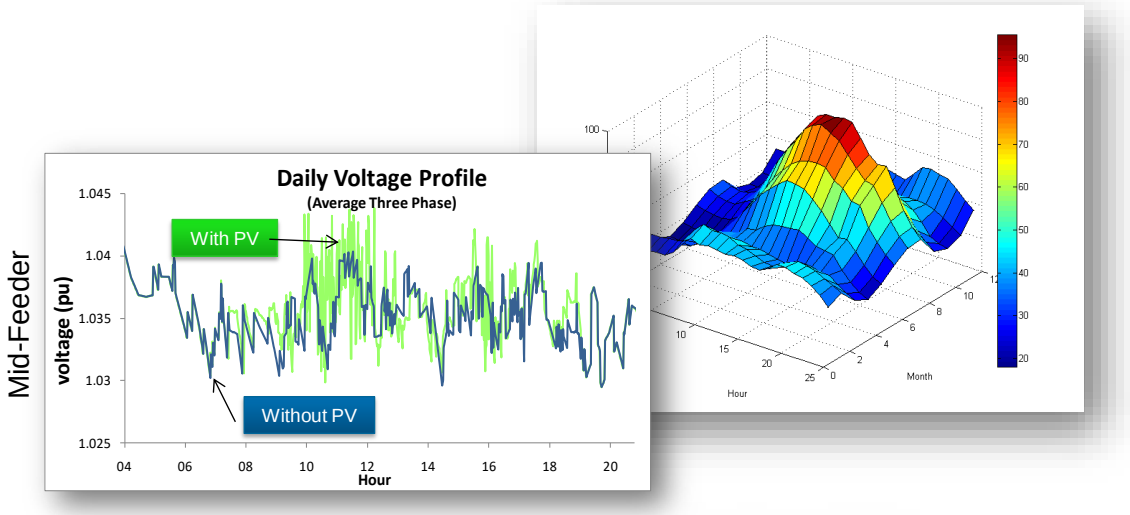
- Parallel processing using actors
- Multithreading circuit processing
- Multi-core management
- Fast power flow

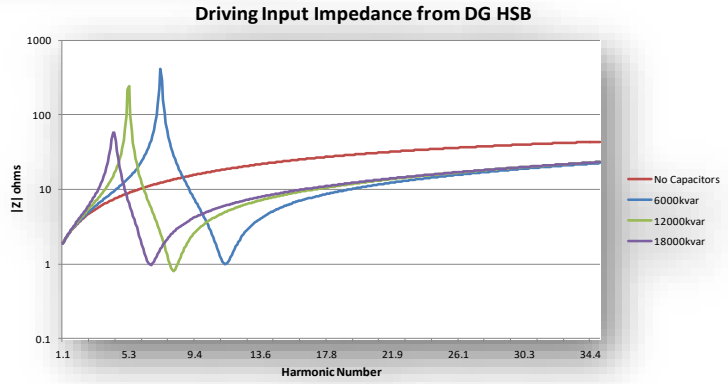
Misc

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

Introduction to OpenDSS

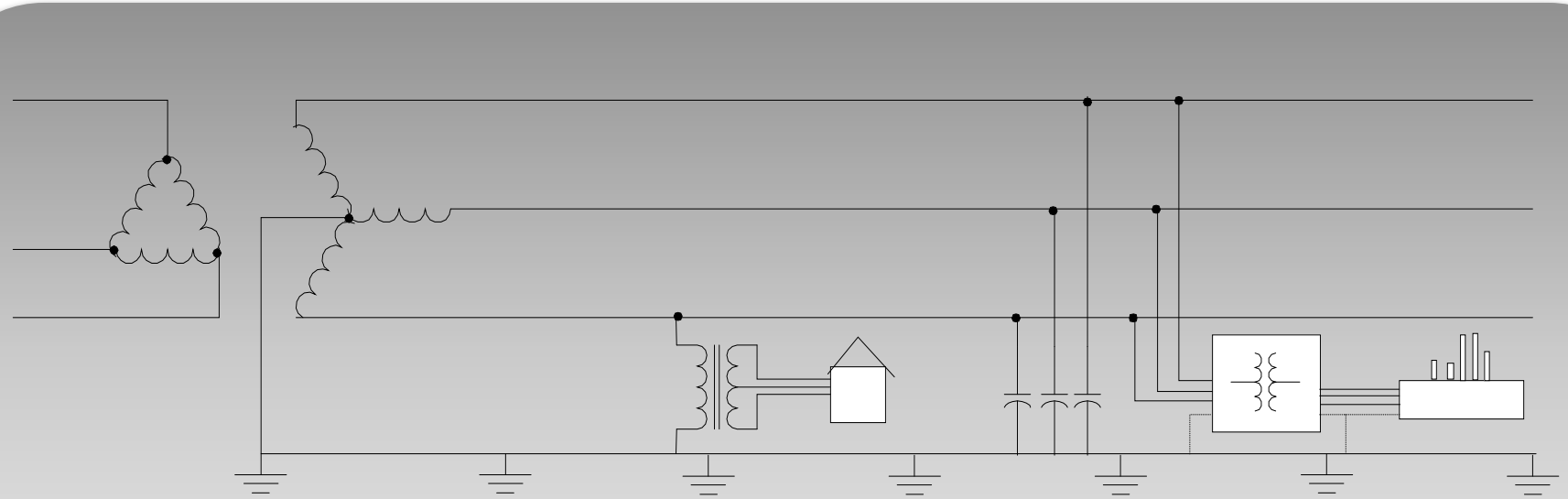
- Flexible Tool Enabling a Wide Range of Analysis Types
  - DER Interconnection studies
  - Locational value studies
  - Hosting capacity studies
  - DA/FLISR scheme evaluation
  - Volt/var optimization
  - Energy impact analysis
  - DER protection impacts
  - Power quality (harmonics/flicker)
  - Long-range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





Introduction to OpenDSS

- Typical North American Distribution System

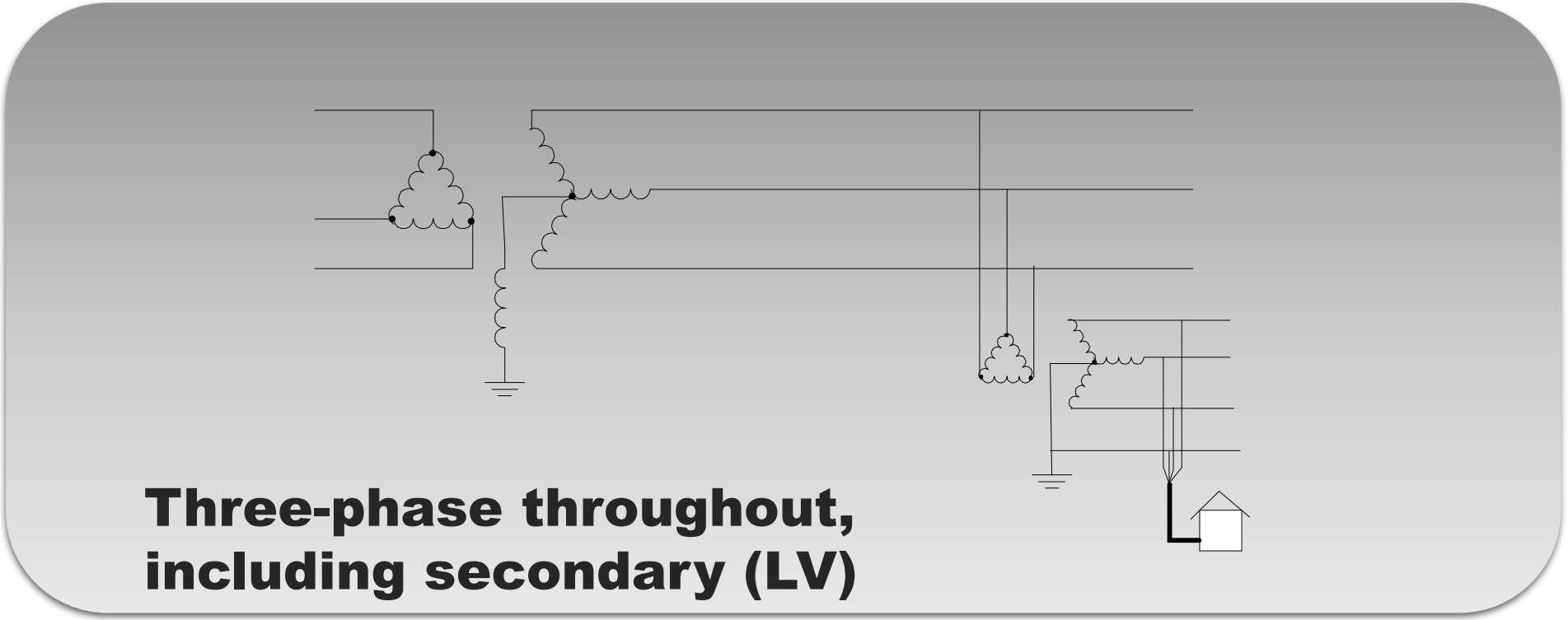


Typical 4-wire multi-grounded neutral system

Unigrounded/Delta 3-wire also common on West Coast

Introduction to OpenDSS

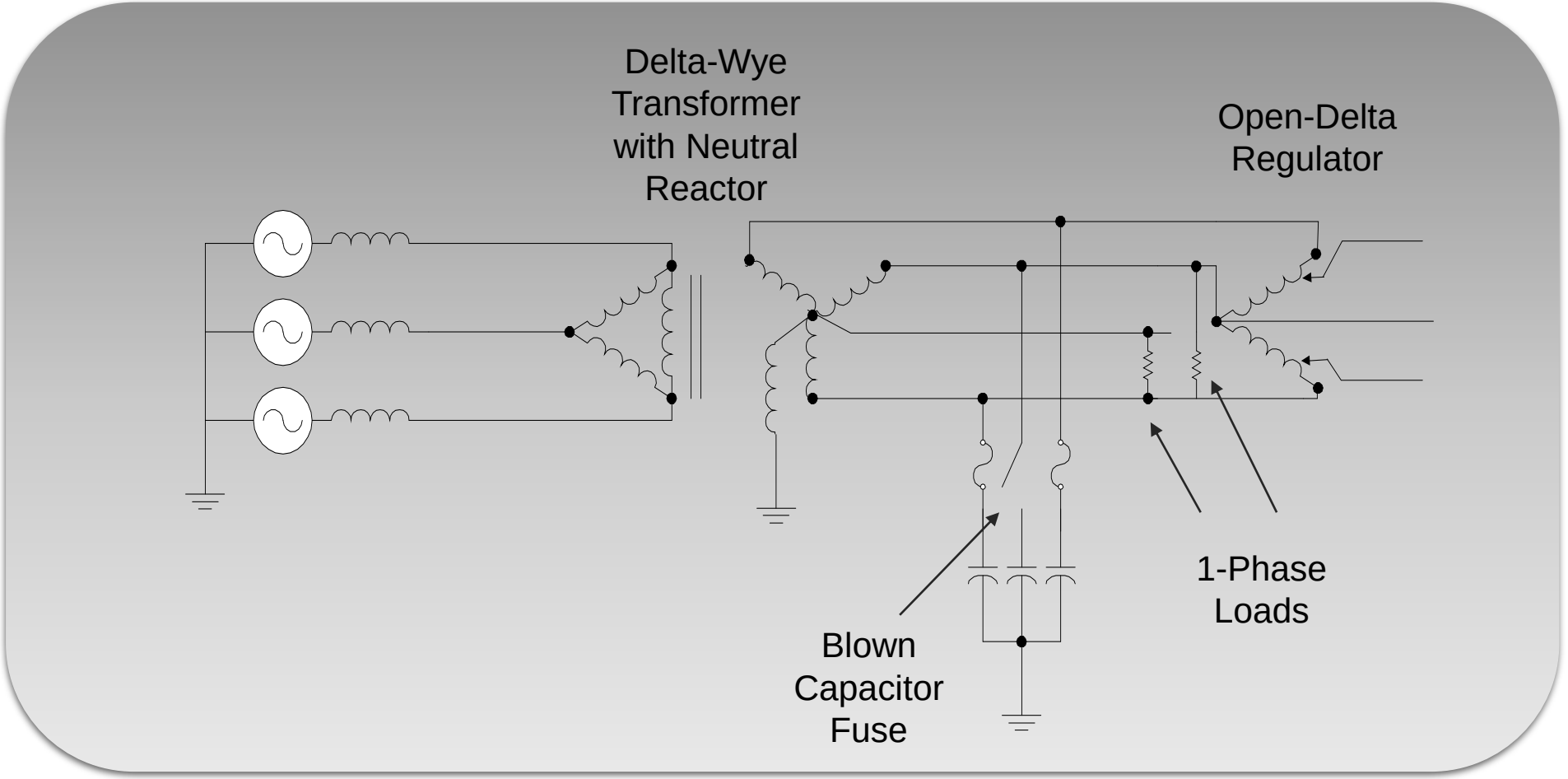
- Typical European Style System



  - **3-wire unigrounded primary**

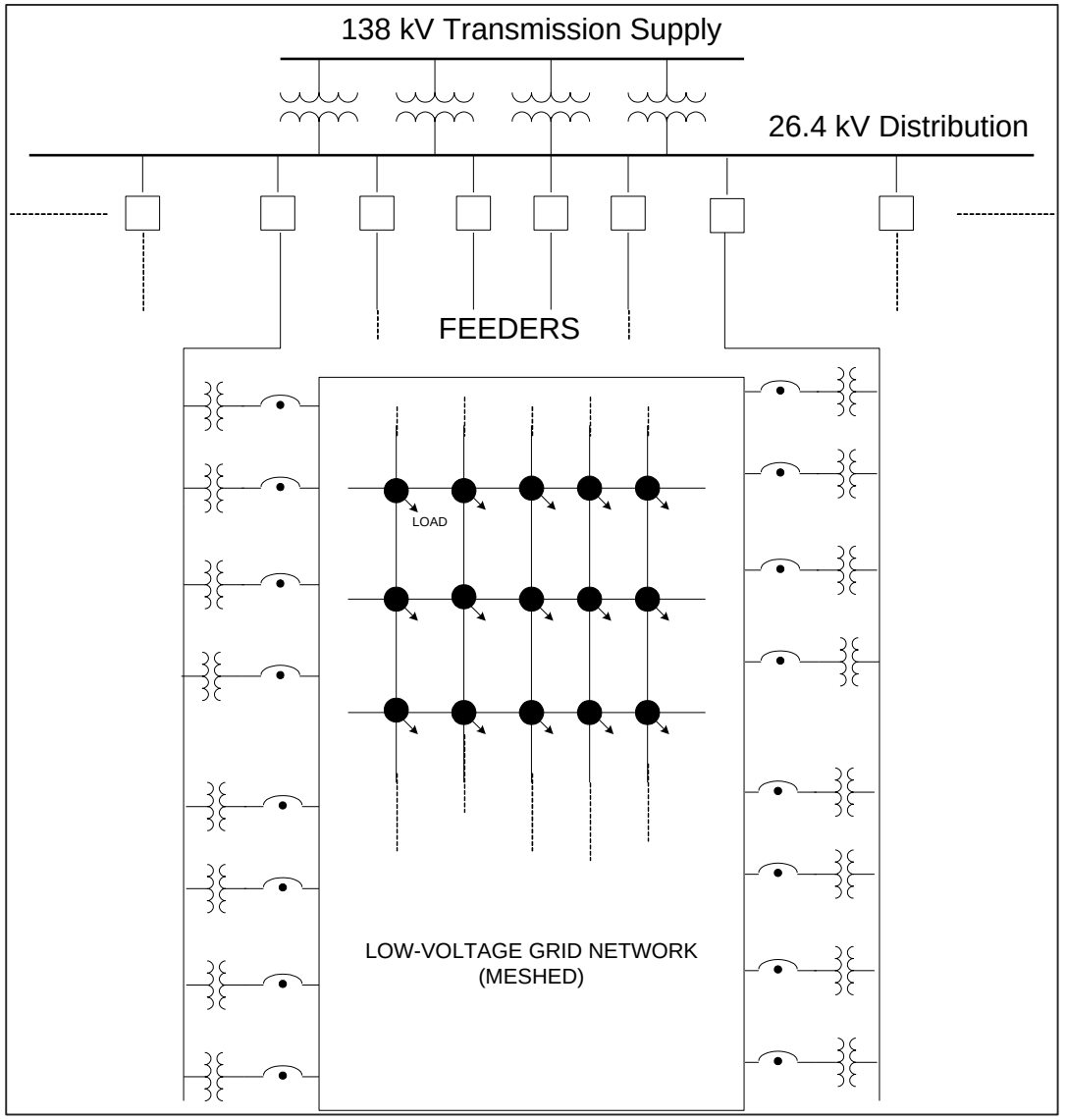
Introduction to OpenDSS

- Why a Positive Sequence Model is Often Inadequate for Distribution System Analysis of North American System



Introduction to OpenDSS

Urban Low-Voltage Network Systems



What is the OpenDSS?

- Script-driven, frequency-domain electrical circuit simulation tool
  - Nodal Admittance formulation
- Specific models for:
  - Supporting **utility distribution system** analysis
  - Designed for the unbalanced, multi-phase North American power distribution systems
    - As well as European-style systems
      - These typically have a simpler structure

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 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 is NOT:
  - 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

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

User Interfaces

- A **stand-alone executable** program that provides a text-based interface (multiple windows)
- An **in-process COM server** (for Windows) that supports driving the simulator from user-written programs.
- A **direct DLL** interface that mimics the COM interface
  - For non-Windows platforms, such as HPCs
  - For programming languages that do not support COM or are not efficient at supporting COM

Repository on SourceForge.Net

Tree [r2195] / trunk / Download Snapshot History

SSH | HTTPS | RO Read/Write SSH access: svn checkout --username=rdugan svn+ssh://rdugan@svn.code.sf.net/p/electricdss/code/trunk electricdss-code

File	Date	Author	Commit
Design	2011-10-22	temcdrm	[r606] add console interface
Distrib	2018-04-09	temcdrm	[r2181] fixing case sensitivity
Doc	4 days ago	temcdrm	[r2193] lazarus build instructions
Parallel_Version	11 hours ago	davismont	[r2195] Bug found in the overload report when performin...
Source	2018-04-11	davismont	[r2184] Update before changing my PC
Test	2018-02-11	temcdrm	[r2132] smaller xneut so it doesn't perturb the voltage...
Training	2017-06-23	rdugan	[r1977] Updates to PPTx slides
License.txt	2015-11-06	rdugan	[r1244] Misc updates
OpenDSSGroup.bdsgroup	2008-09-11	temcdrm	[r11] initial KLUSolve build with pointer handles
Sourceforgelinks.html	2011-06-27	rdugan	[r554]

Accessing the SourceForge.Net Source Code Repository with TortoiseSVN

- Install a TortoiseSVN client from Tortoisesvn.net/downloads.
- Recommendation:

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

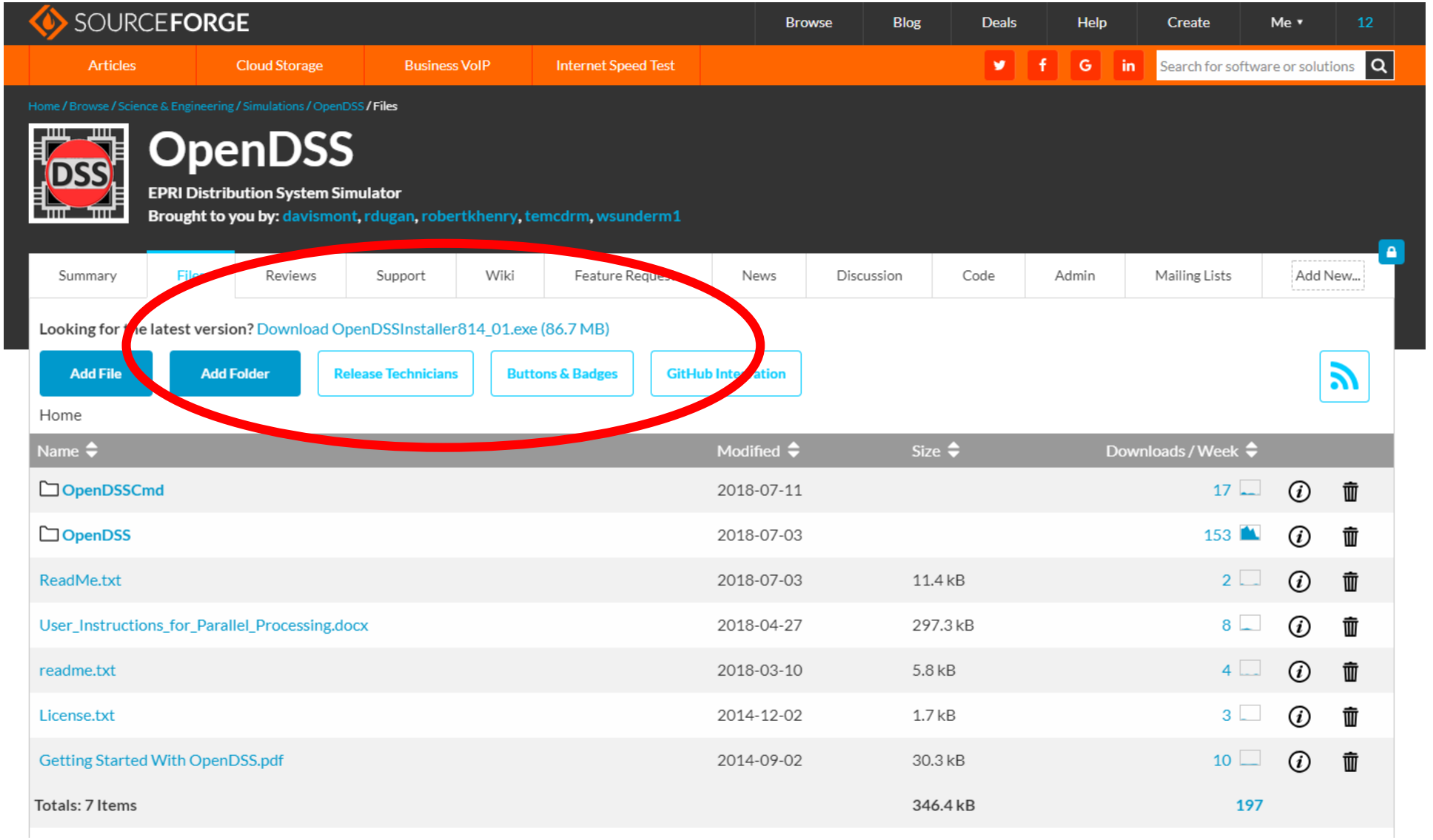
3 - the repository URL is

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

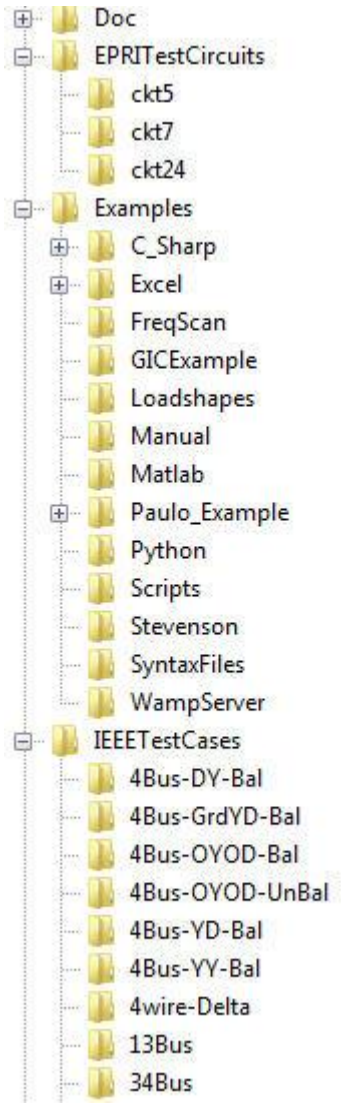
(Change the checkout directory if it points somewhere other than what you want.)

Thereafter, to update a folder or file, right-click on the folder or file and select **SVN Update**

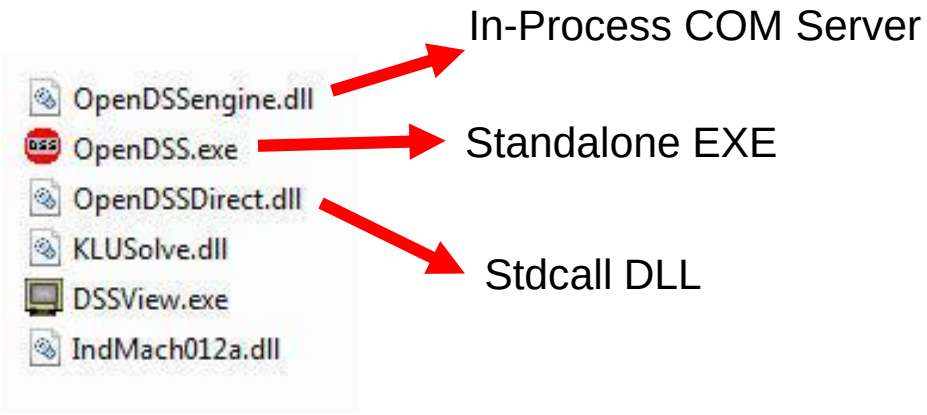
Download the Installer Files



OpenDSS Files Installed

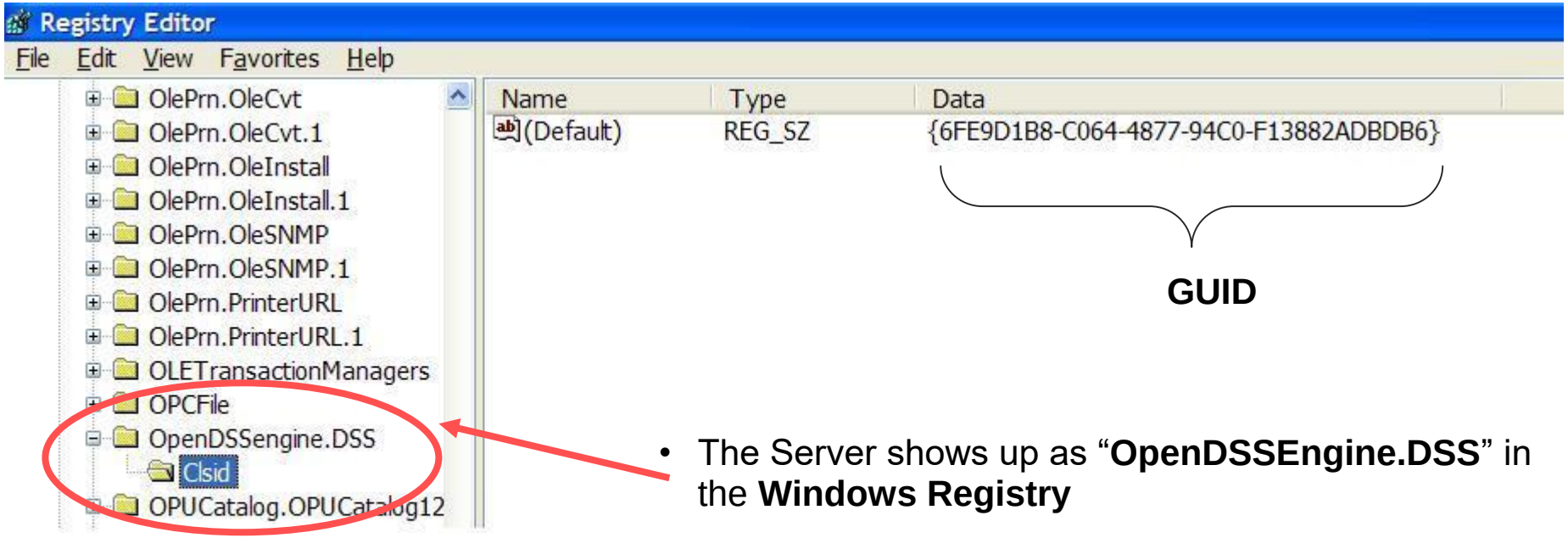


Main Program Files



Registering the COM server

Windows Registry Entry

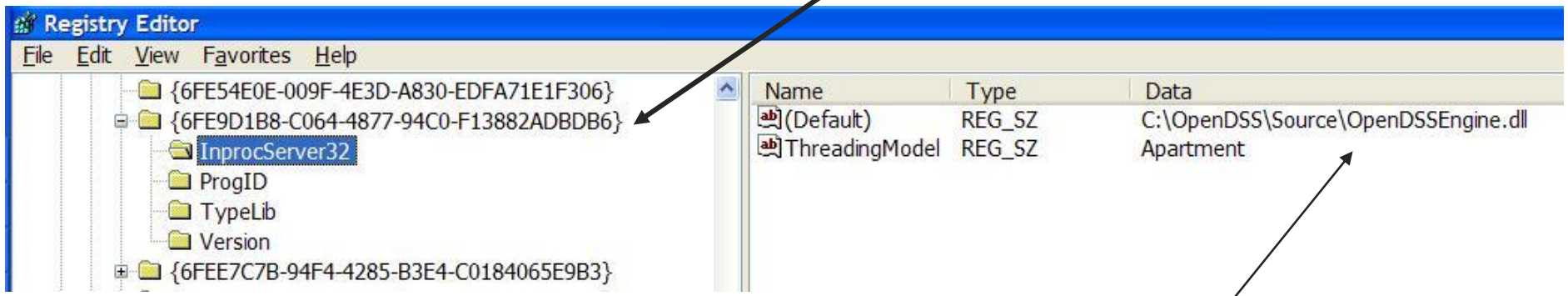


- The Server shows up as "**OpenDSSEngine.DSS**" in the **Windows Registry**

The OpenDSS is now available to any program on the computer

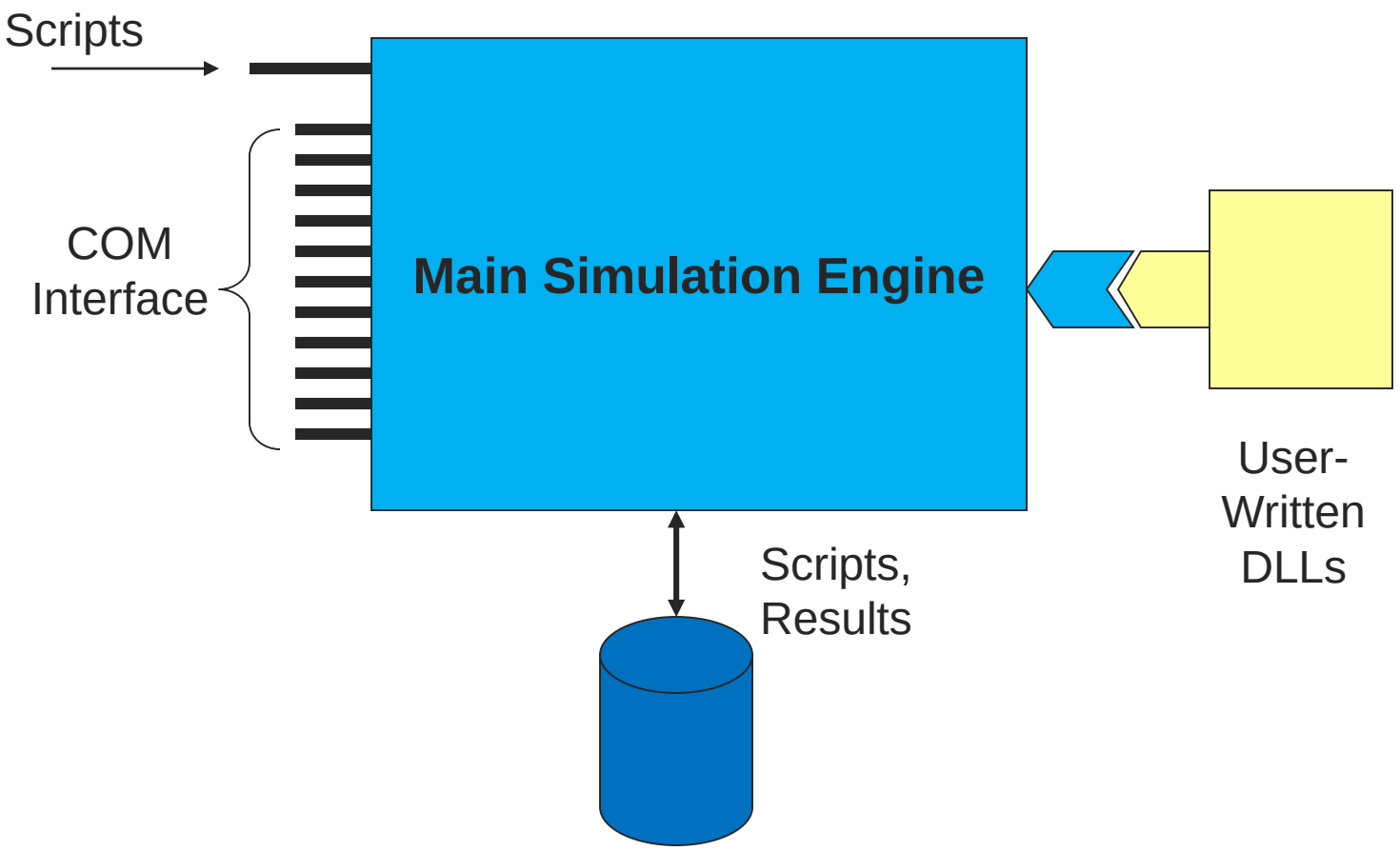
The GUID References the DLL File

If you look up the GUID in RegEdit

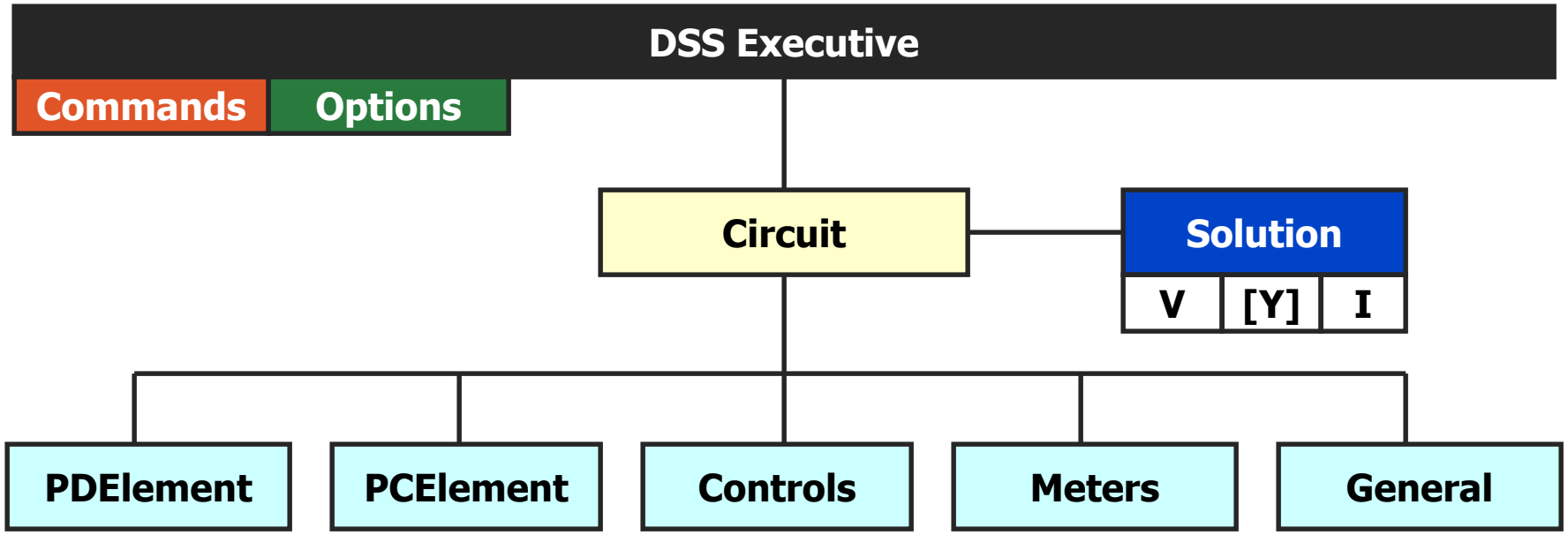


Points to OpenDSSEngine.DLL
(In-process server, Apartment Threading model)

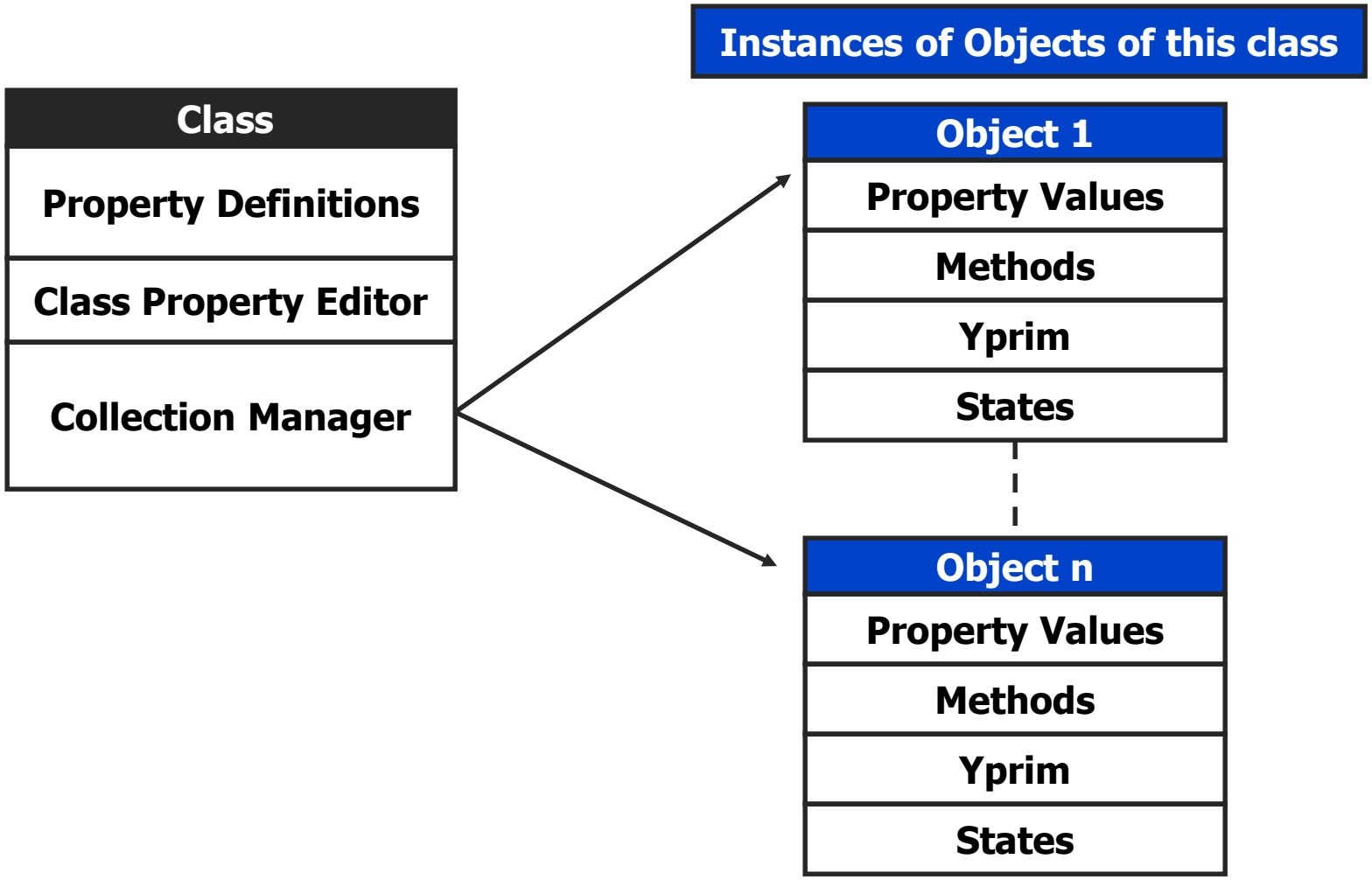
DSS Structure



DSS Object Structure



DSS Class Structure



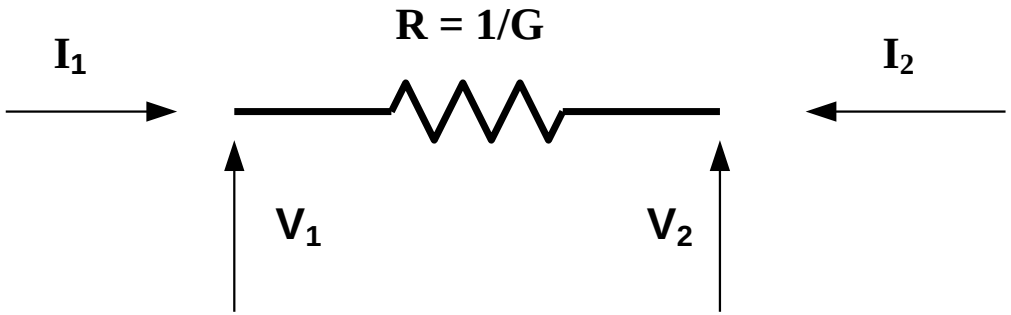
The Math …

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

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

Primitive Y Matrix

- Simple Resistor

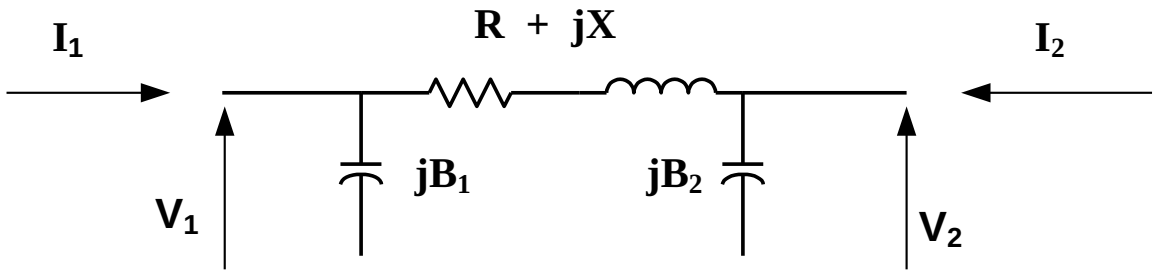


$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} G & -G \\ -G & G \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$$

$$\mathbf{Y}_{\mathbf{prim}}$$

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?

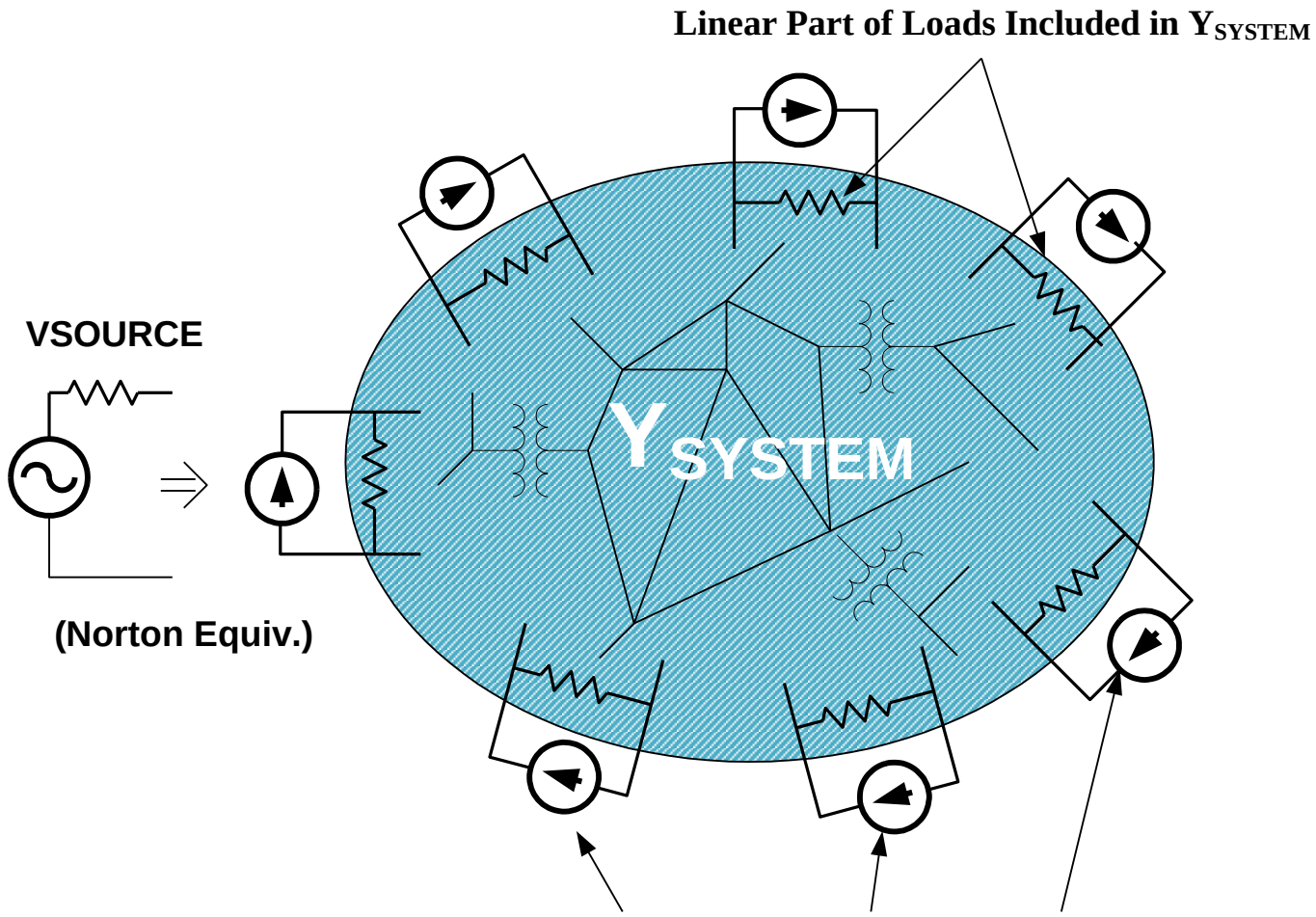
- Simply let **R, X, B, G, C**, etc. represent **3x3** matrix
  - Notation stays the same

- And it works!

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

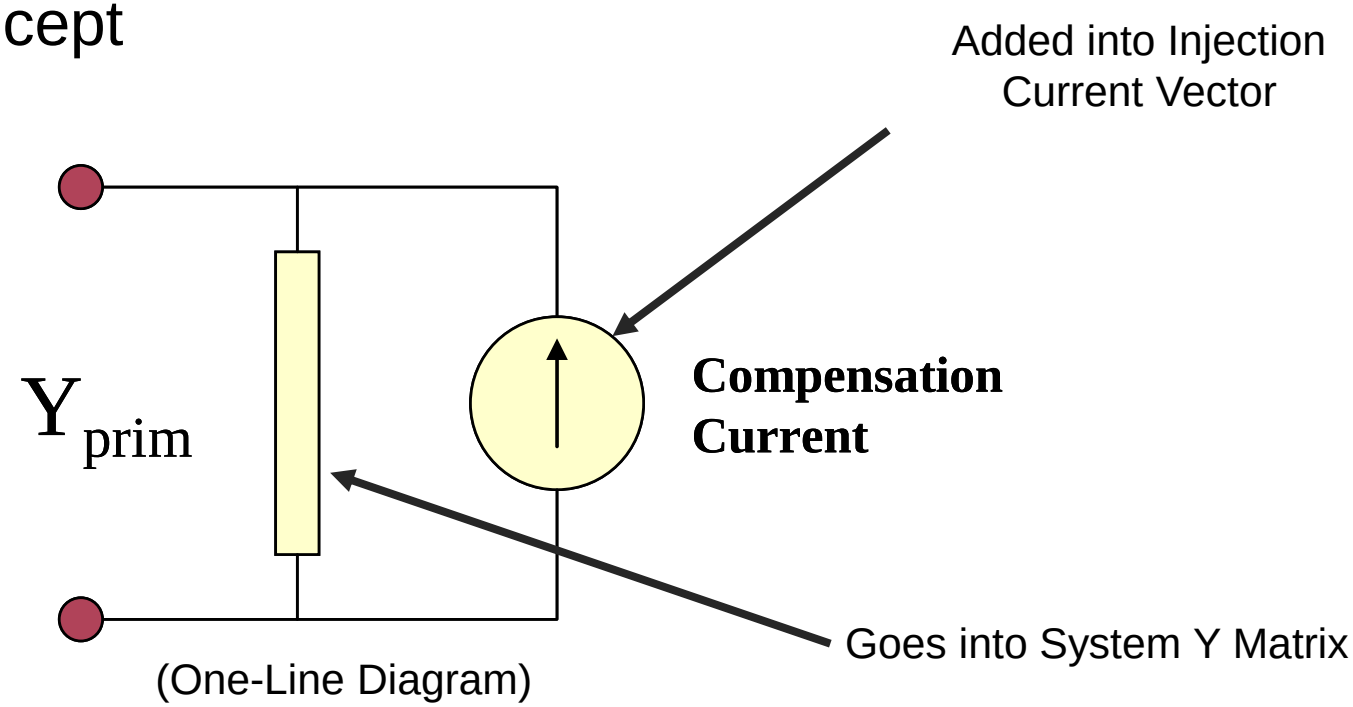
- This is basically how all the Circuit Element (CktElement) models in OpenDSS work.

The Network Model



Load (a PC Element)

General Concept



Most Power Conversion (PC) Elements are Modeled Like This

Nodal Admittance Equations

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

N = Number of NODES (not BUSES)

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.

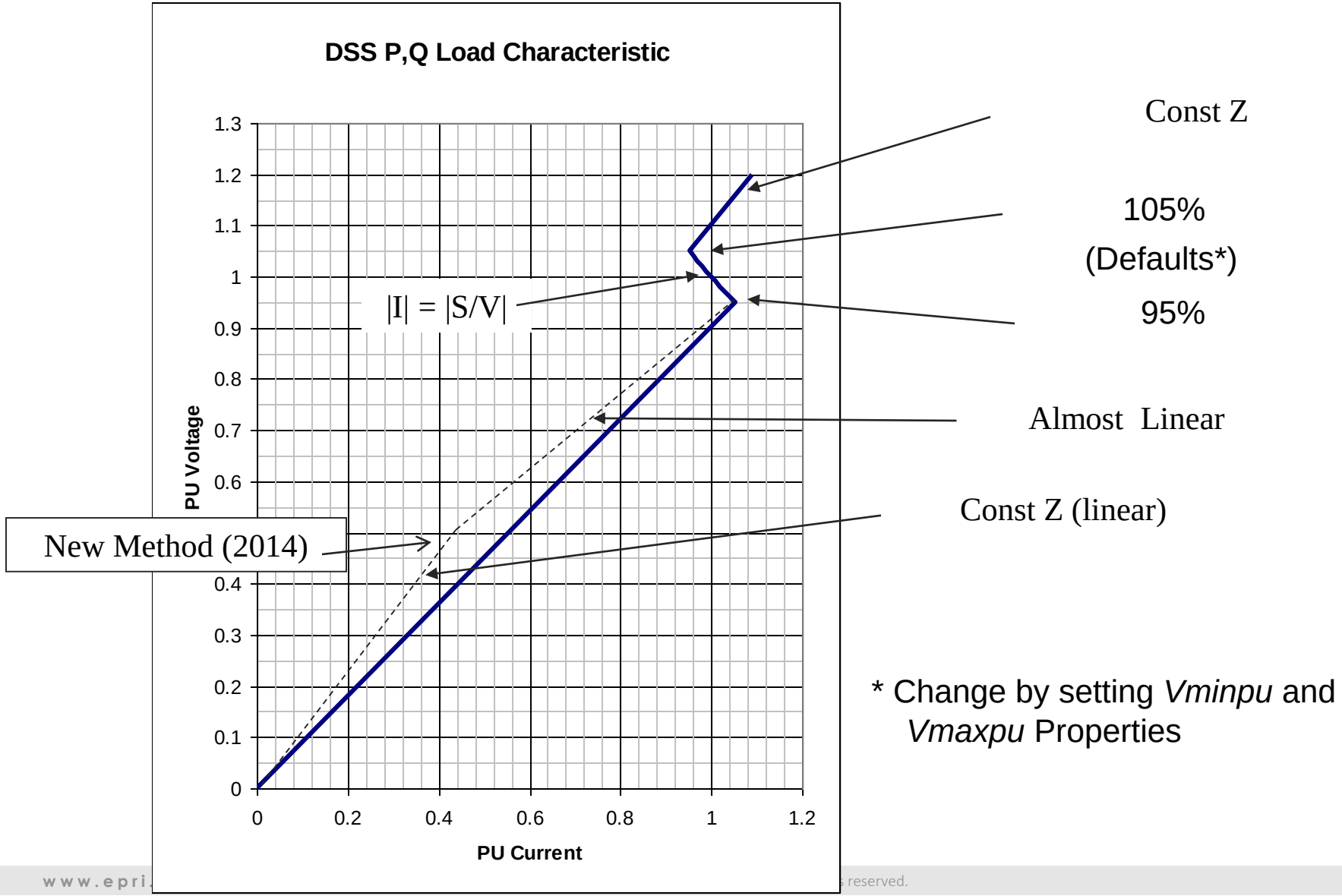
5:Constant Current Magnitude

6:Const P, Fixed Q

7:Const P, Fixed Impedance Q

8: Special ZIP load model

Standard P + jQ Load Model (1)



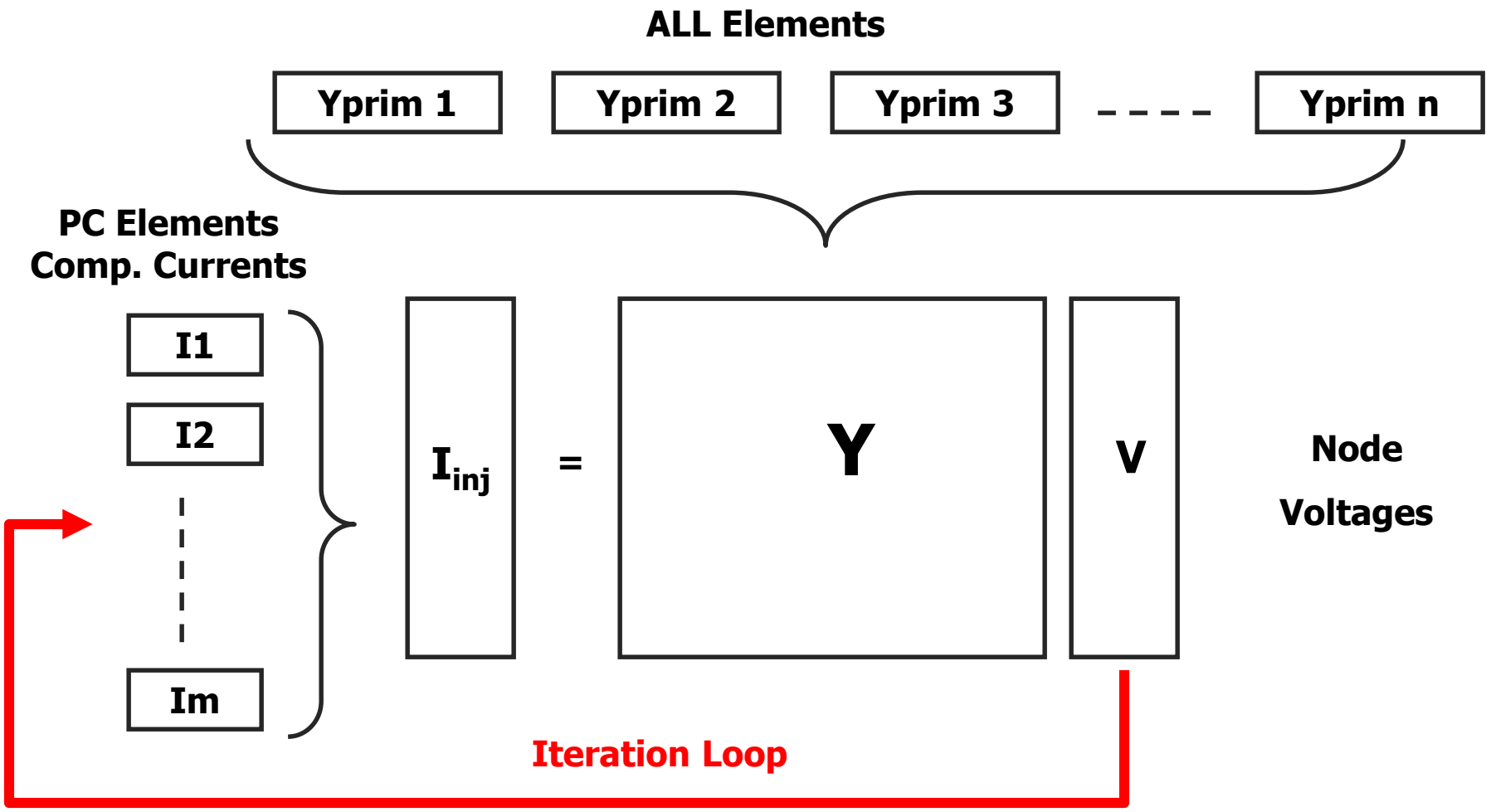
* Change by setting *Vminpu* and *Vmaxpu* Properties

Power Flow Solution Algorithm

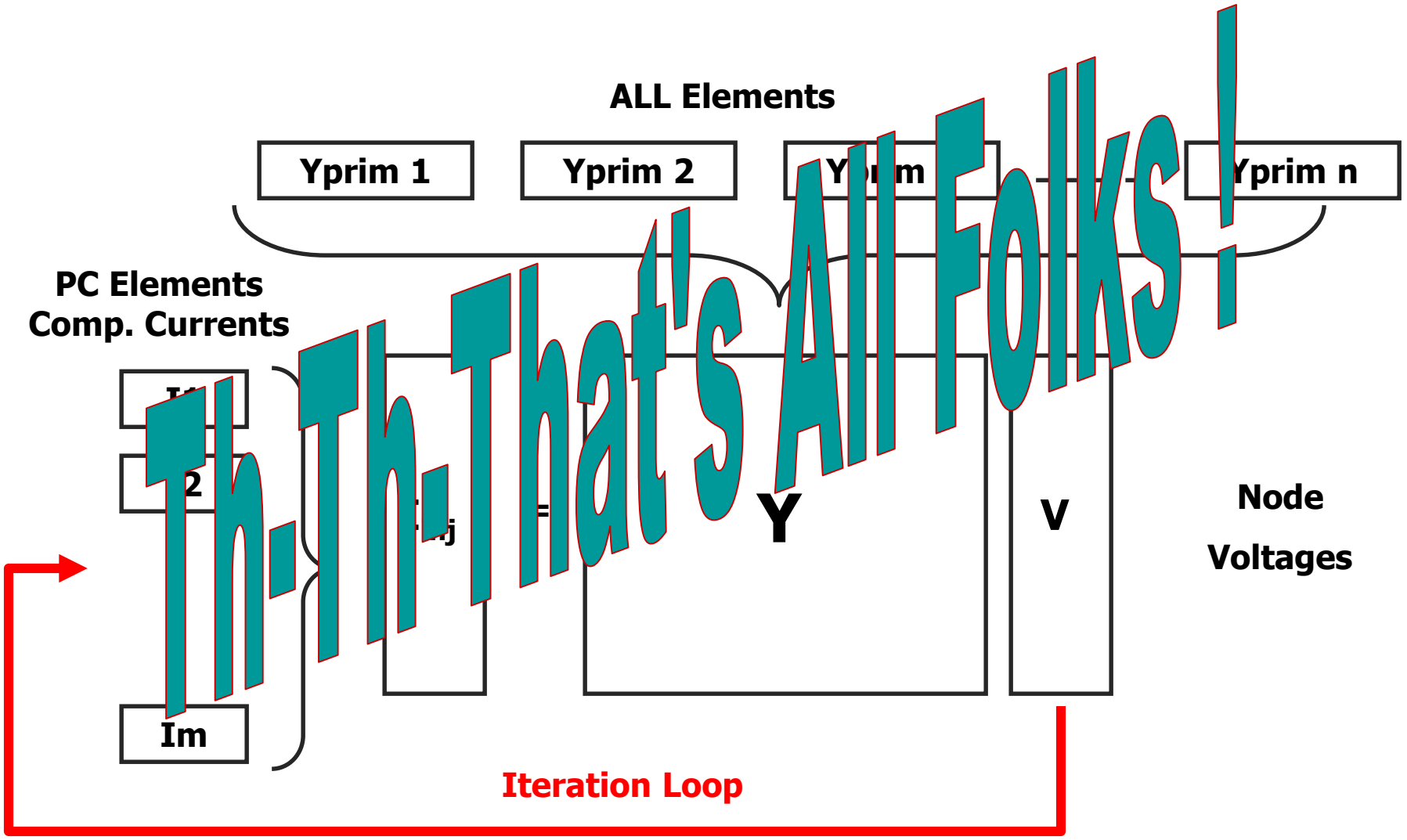
1. Initial Guess at Node Voltages, $\mathbf{V}$
2. Compute all Injection (Compensation) Currents, $\mathbf{I}$
   a. For PC Elements
3. Solve for new guess at $\mathbf{V}$
4. Repeat 2 and 3 until Converged

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

Putting it All Together



Putting it All Together



A More Concise Form …

- Fixed-point solution form for normal solution

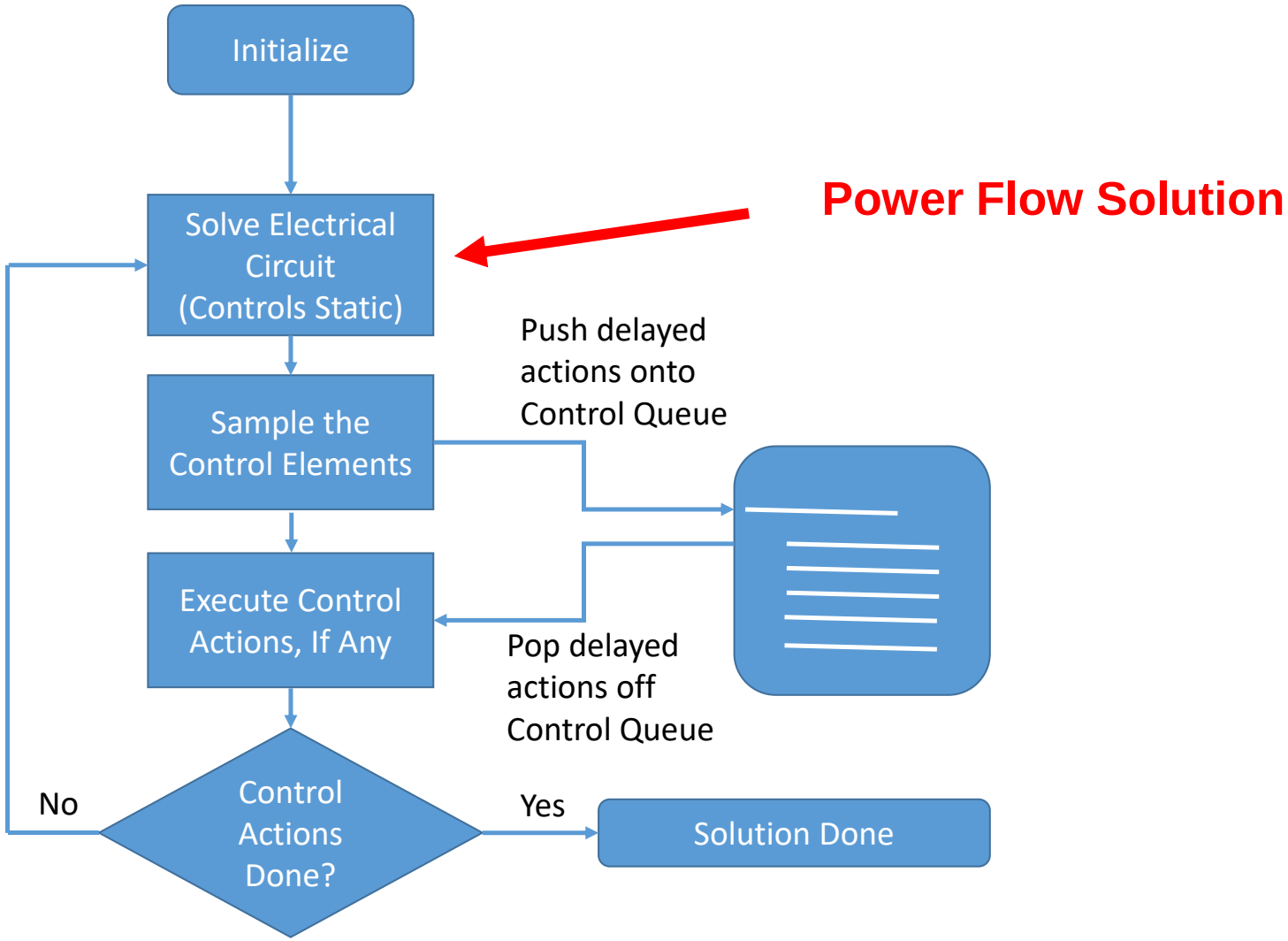
$$V_{n+1} = [Y_{system}]^{-1} I_{PC}(V_n) \quad n = 0, 1, 2, \ldots$$

… until converged

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

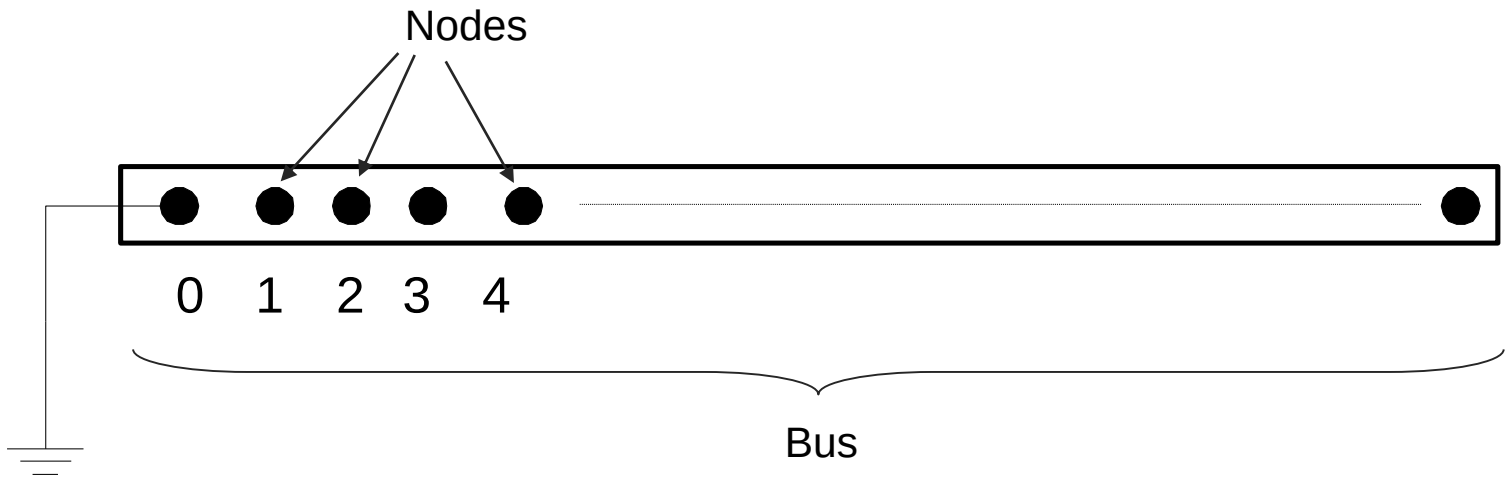
OpenDSS Solution Loop with Controls

Controls are sampled and executed after a converged power flow solution



Circuit Modeling Basics

DSS Bus Model (Bus ≠ Node)



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

Bus1=*BusName.1.2.3.0*

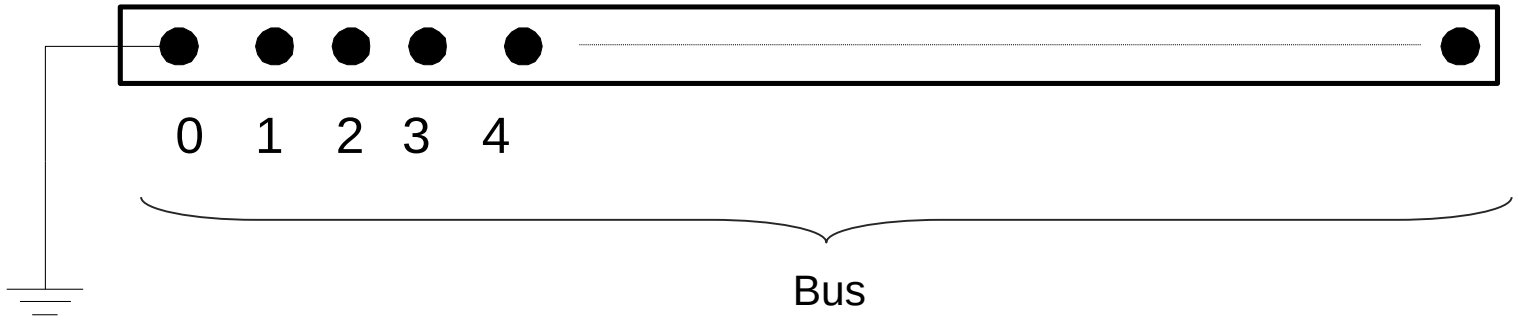
(This is the default for a 3-phase circuit element)

Shorthand notation for taking the default:

Bus1=*BusName*

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

Node Numbers



The voltage at Node 0 = 0 (always)

The other Node numbers are arbitrary

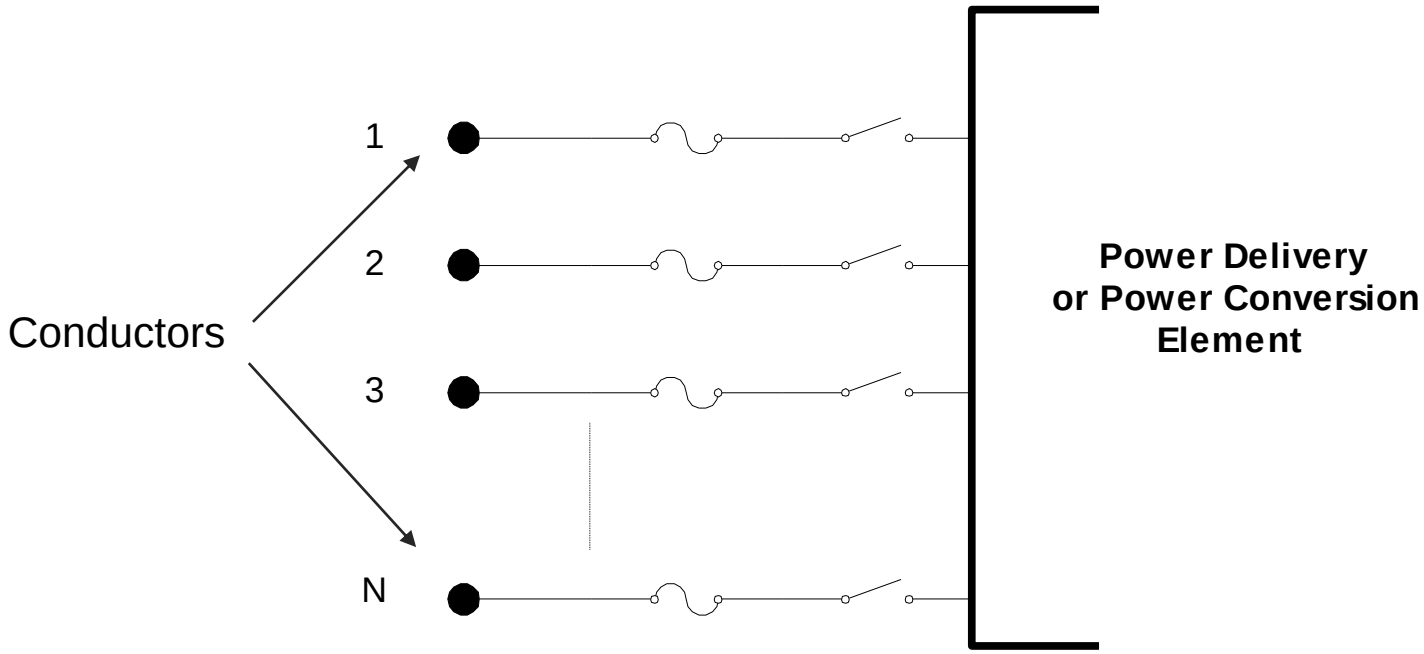
By convention, Nodes 1, 2, 3 correspond to phase ABC

But they don't have to

You can have a very large number of nodes at a Bus

They do not have to be pre-declared

DSS Terminal Definition

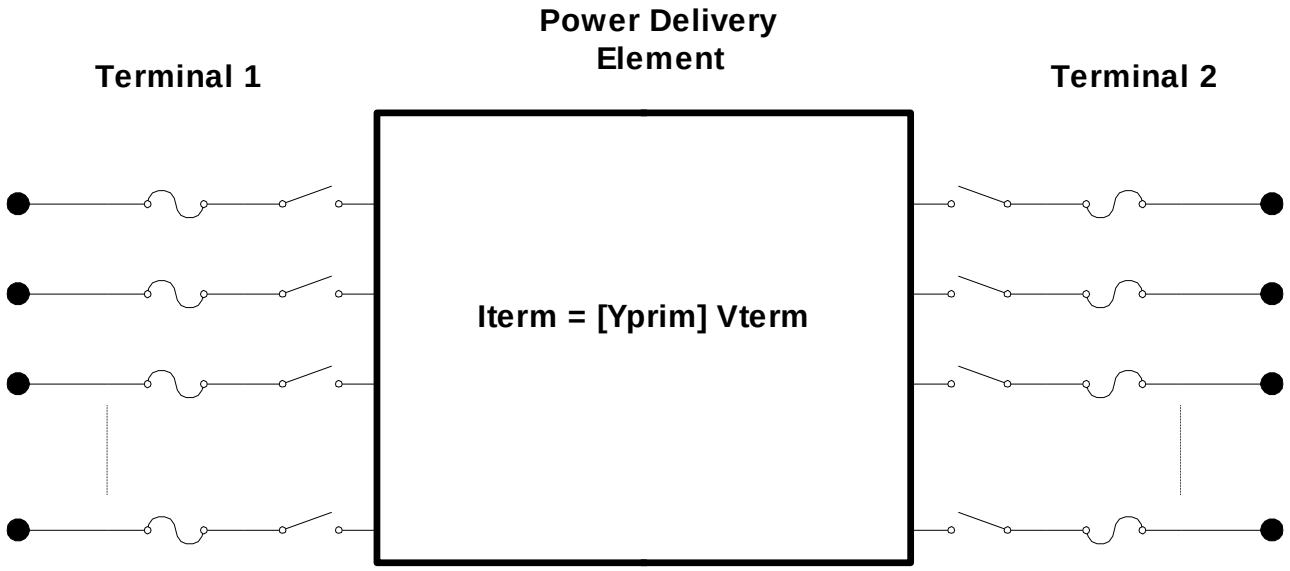


Circuit Elements have one or more *Terminals* with 1..N conductors.

Conductors connect to *Nodes* at a *Bus*

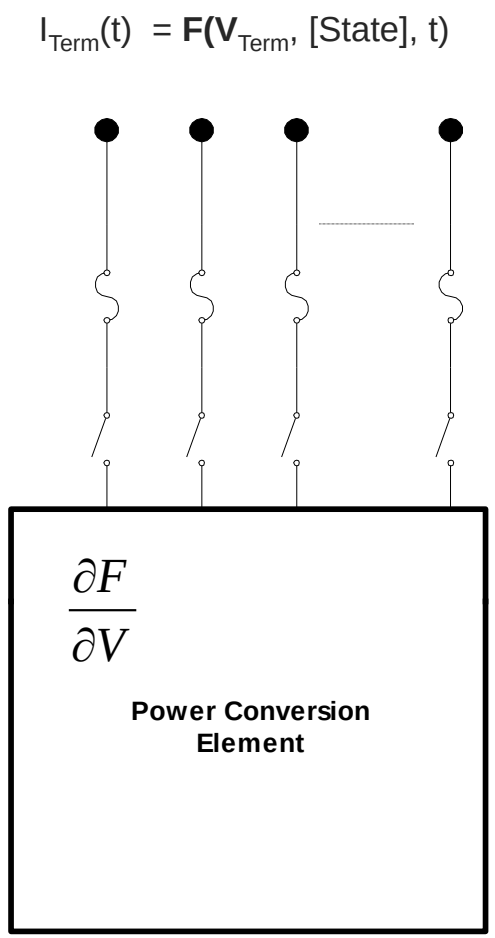
Each *Terminal* connects to one and only one *Bus*

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.

Scripting Basics

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

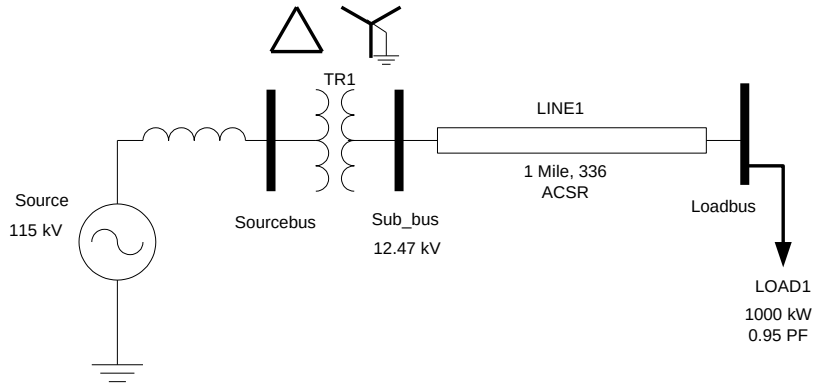
- Array or string delimiter pairs: **[ ] , { },( ),“ “,‘ ‘**
- Matrix row delimiter: **|**
- Value delimiters: **,** (comma)
  any white space (tab or space)
- Class, Object, Bus, or Node delimiter: **.** (period)
- Keyword / value separator: **=**
- Continuation of previous line: **~** (More)
- Comment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

A Basic Script

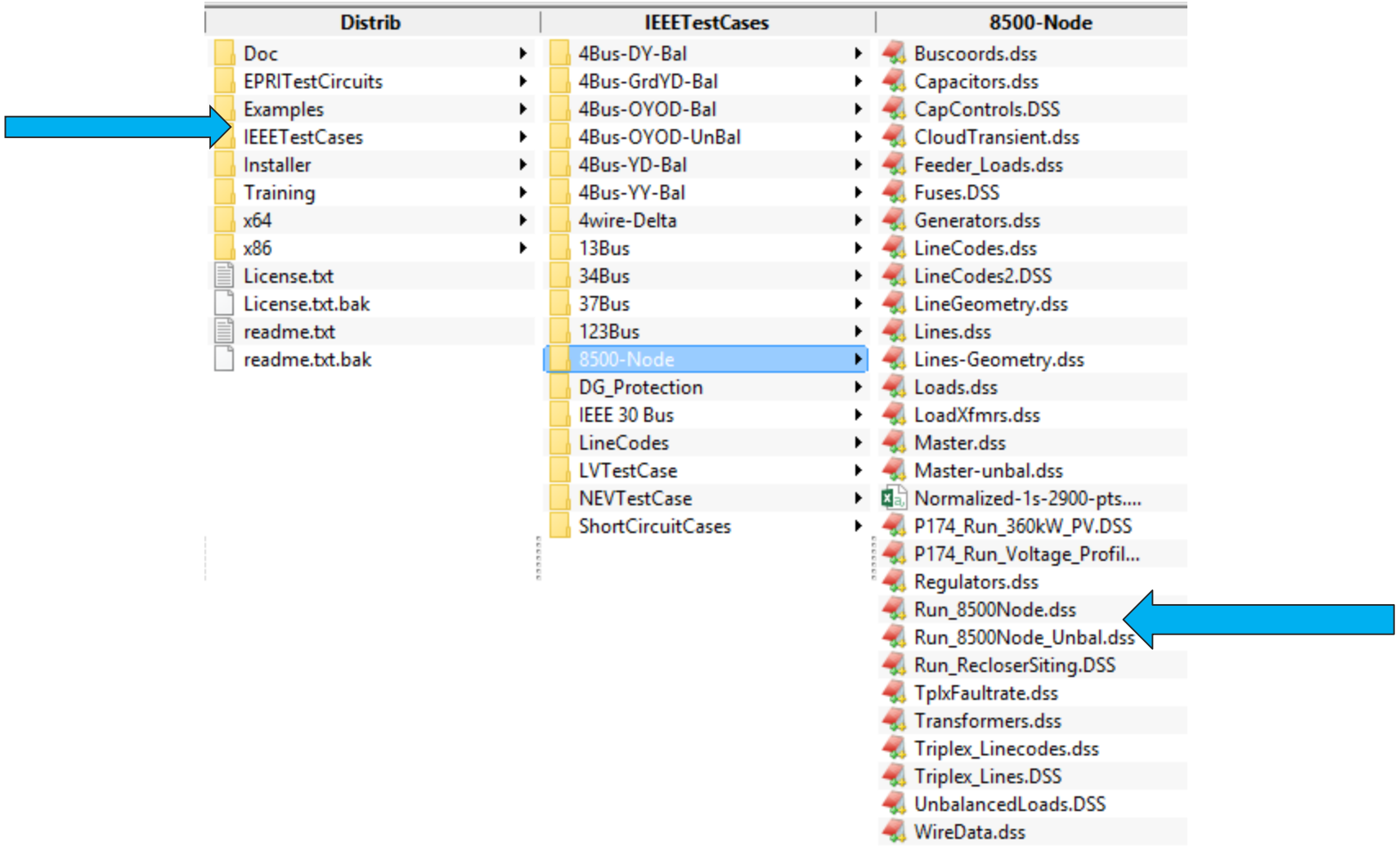



```
Clear
New Circuit.Simple     ! Creates voltage source  (Vsource.Source)
Edit Vsource.Source BasekV=115 pu=1.05  ISC3=3000  ISC1=2500  !Define source V and Z
New Transformer.TR1 Buses=[SourceBus, Sub_Bus] Conns=[Delta Wye] kVs= [115 12.47]
~ kVAs=[20000 20000] XHL=10
New Linecode.336ACSR R1=0.058 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 LN Nodes
Show Currents Element
Show Powers kVA Elements
```

(You can Copy and Paste this into OpenDSS.EXE

Example:
IEEE 8500-Node Test Feeder

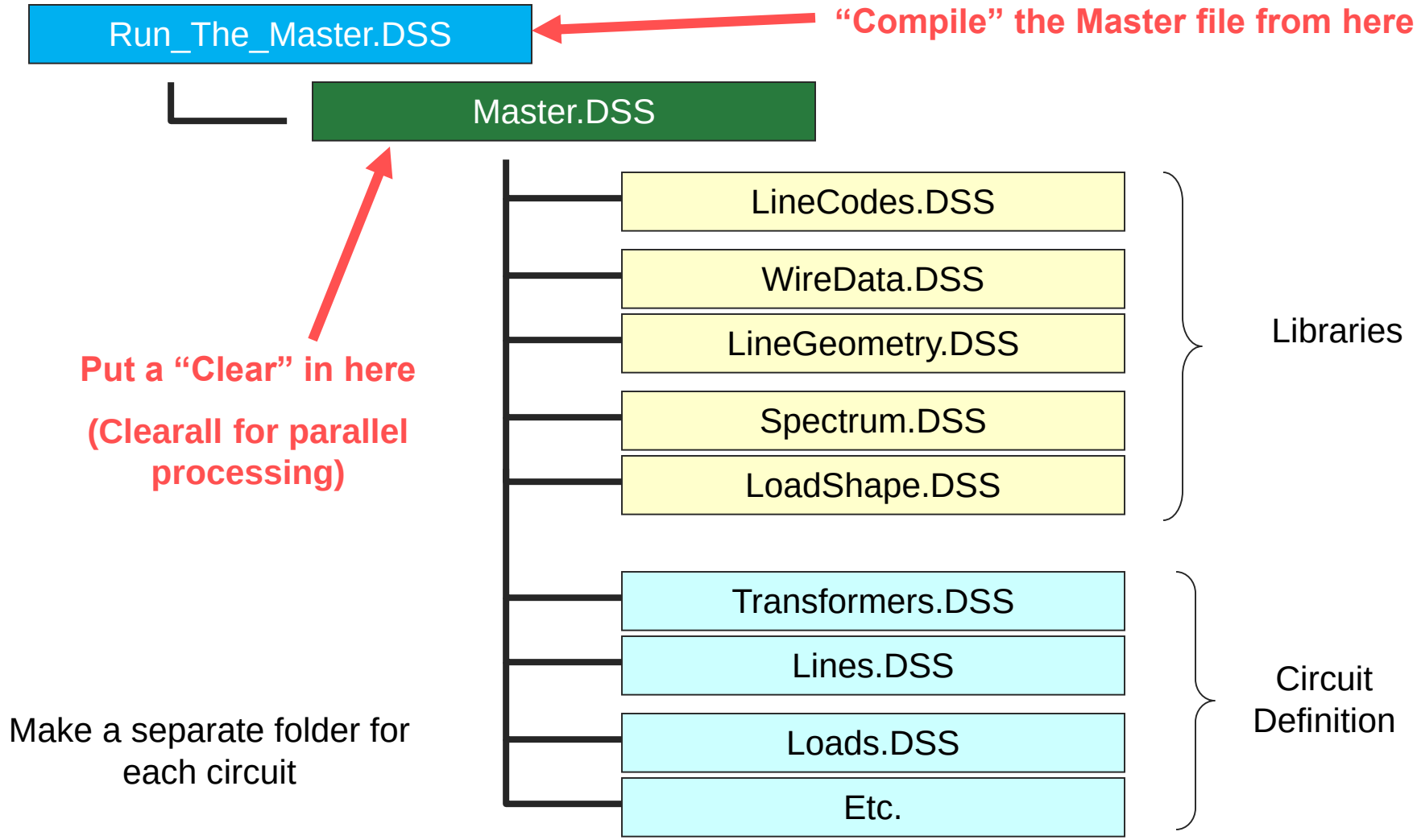
Location of the IEEE 8500-Node Test Feeder Files



Scripting Large Circuits

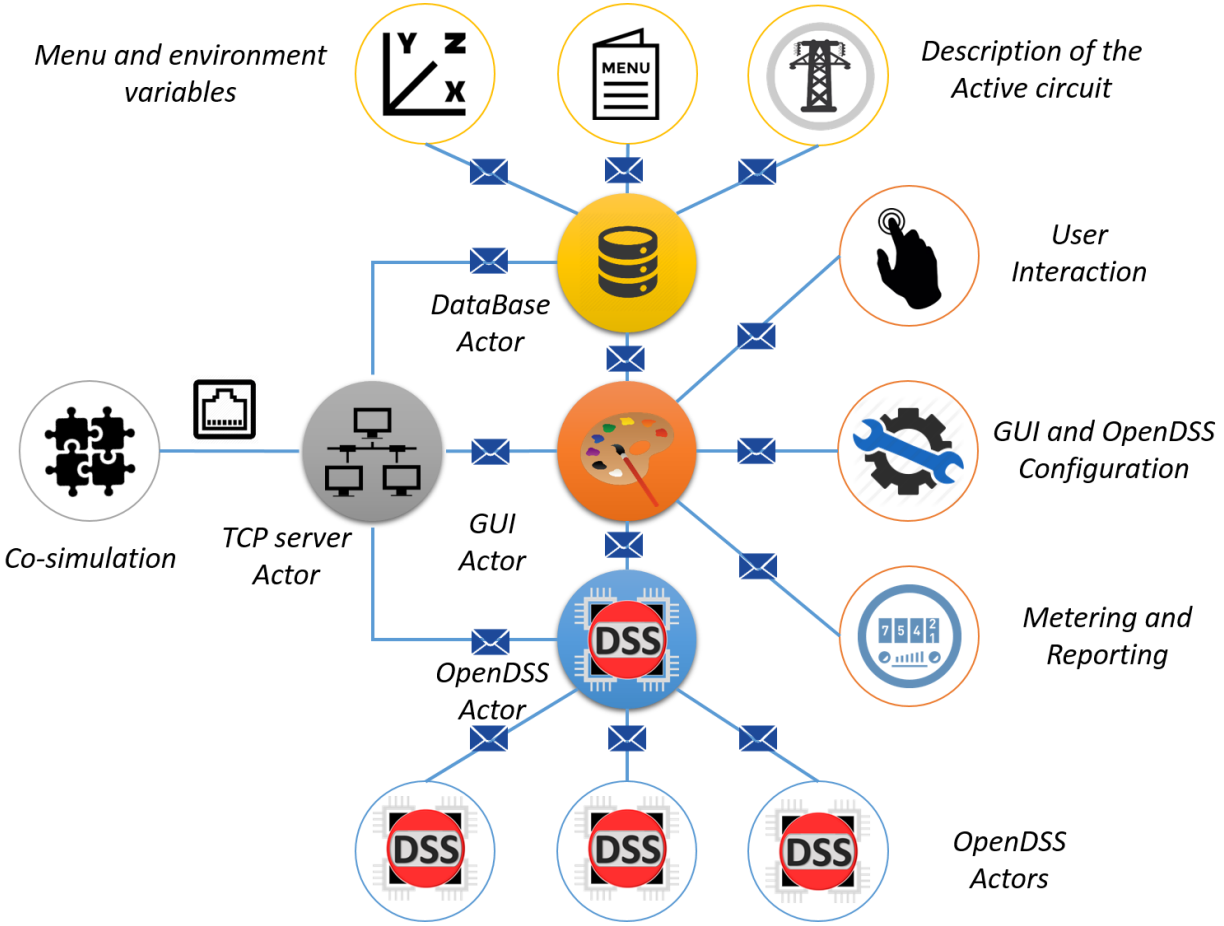
- For small circuits, it is often sufficient to put all the scripts in a single file
  - Many of the IEEE test feeder examples are mostly in a single file
- When you have large amounts of data, a more disciplined approach is recommended
- **Redirect** Command
  - Redirects the input to another file
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

A Common Sense Structuring of Script Files



Introduction to OpenDSS-G

Introduction to OpenDSS-G



OpenDSS-G (formerly DSSim-PC) is a new step in the evolution of simulation tools for planning and operations based in OpenDSS.

This interface has adopted the functionalities of OpenDSS for making easier to the user to use the advance features of the platform.

Introduction to OpenDSS-G

- Installing OpenDSS-G

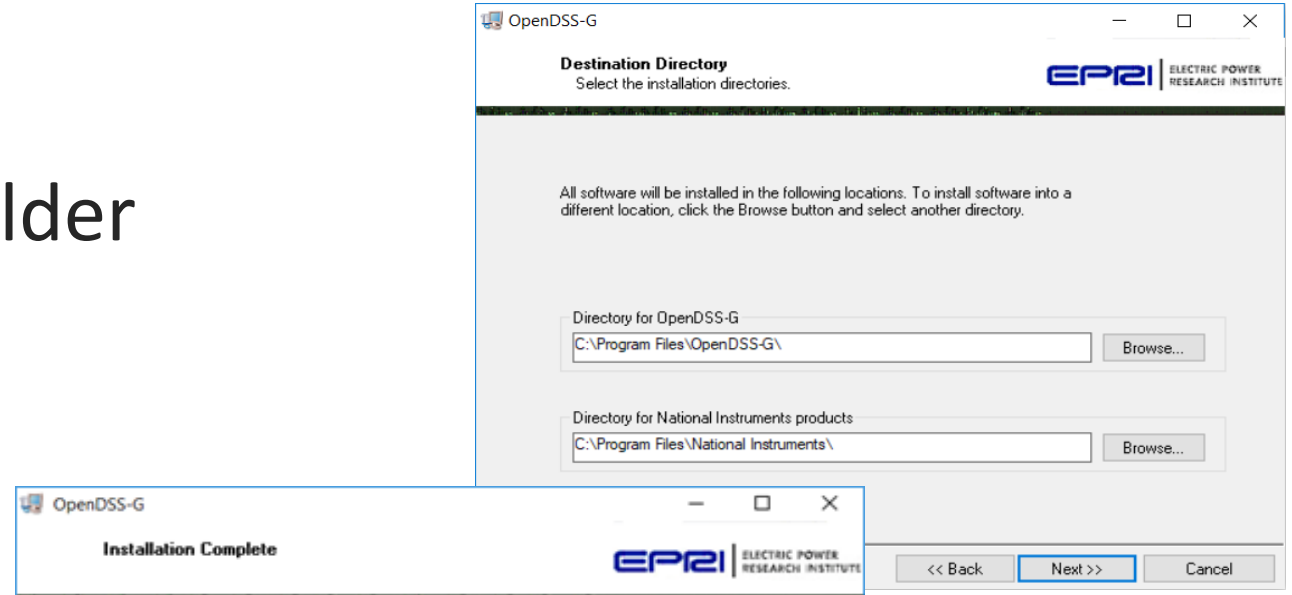
1. Download (https://www.epri.com/#/pages/sa/opendss?lang=es)

2. Follow the instructions

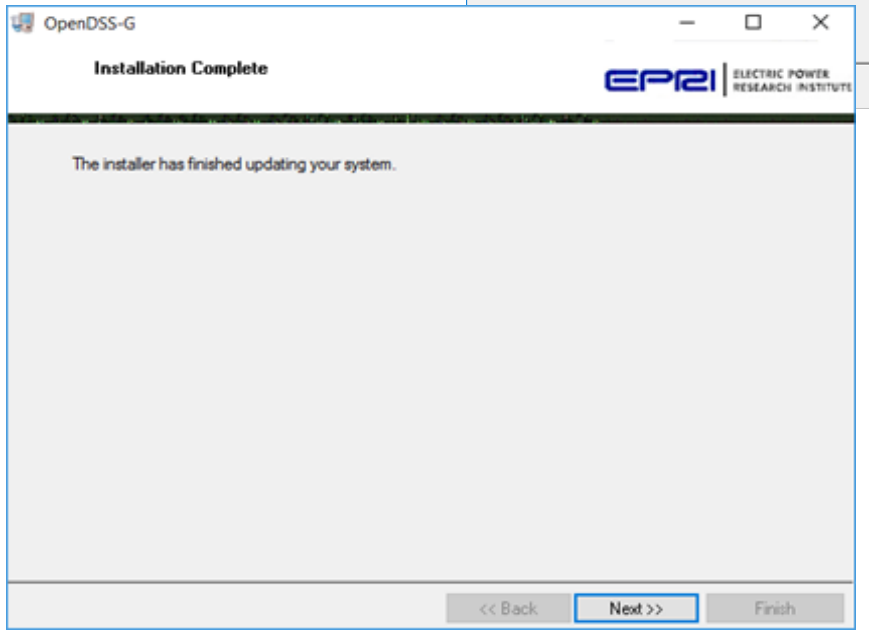
Introduction to OpenDSS-G

- Installing OpenDSS-G

3. Specify the destination folder



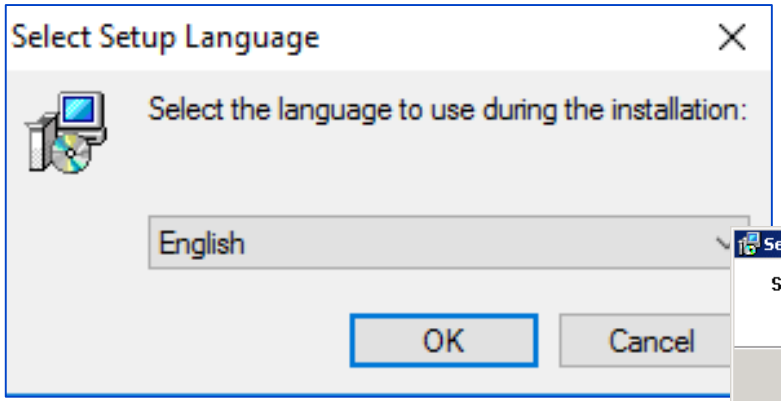
4. Finish

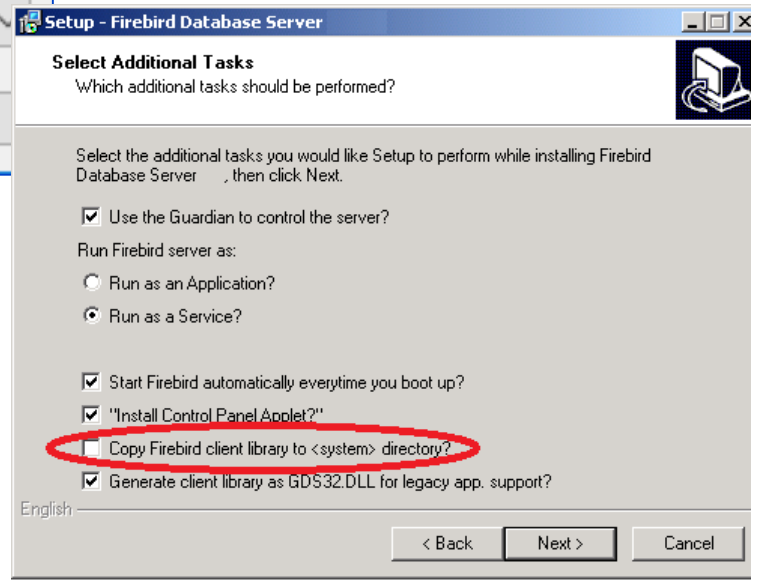


Introduction to OpenDSS-G

- Installing OpenDSS-G

5. Install the database





6. This is not mandatory but just in case

Introduction to OpenDSS-G

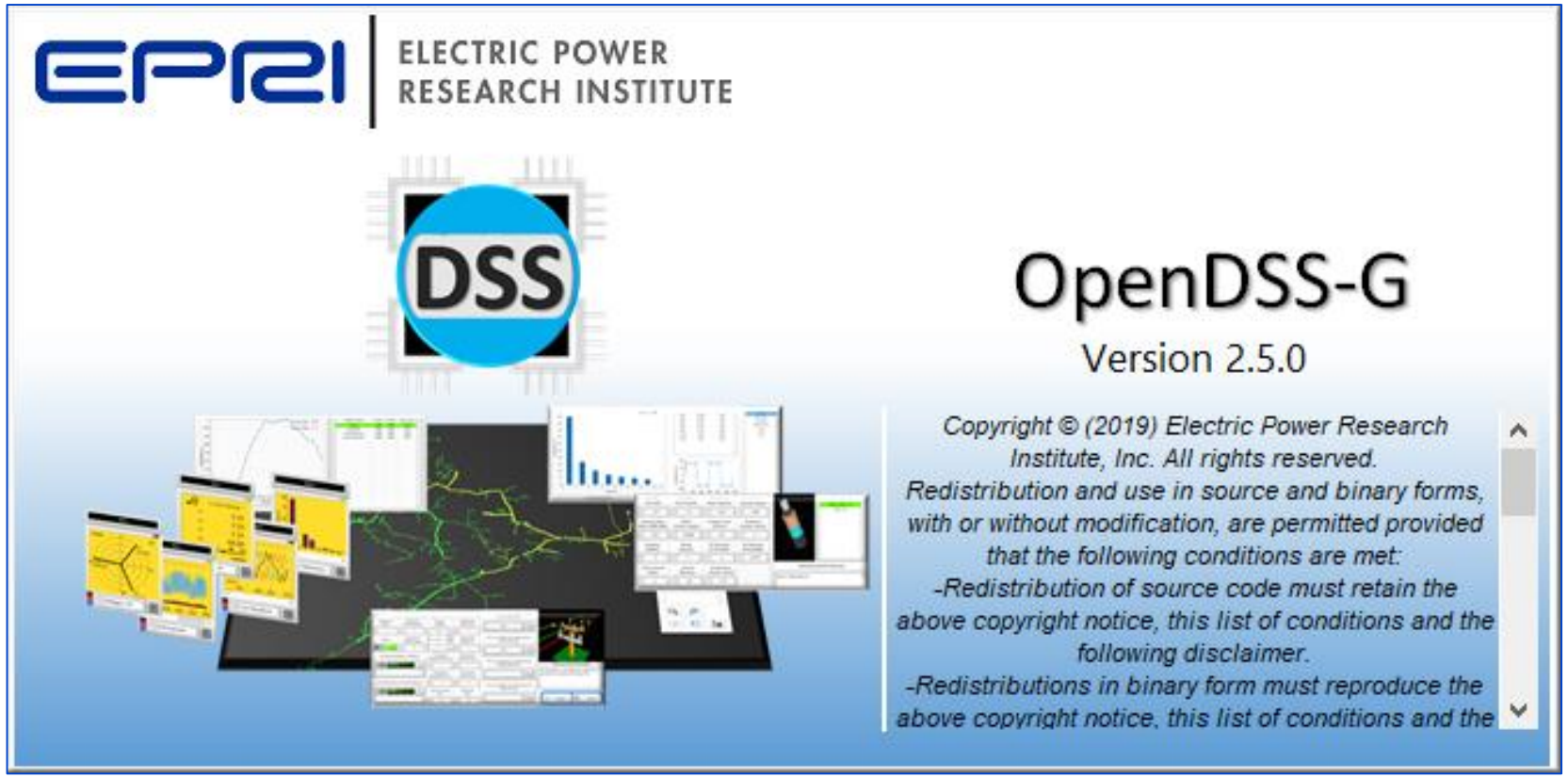
- Installing OpenDSS-G

7. Finally, the ODBC

At this point your system should be all set!

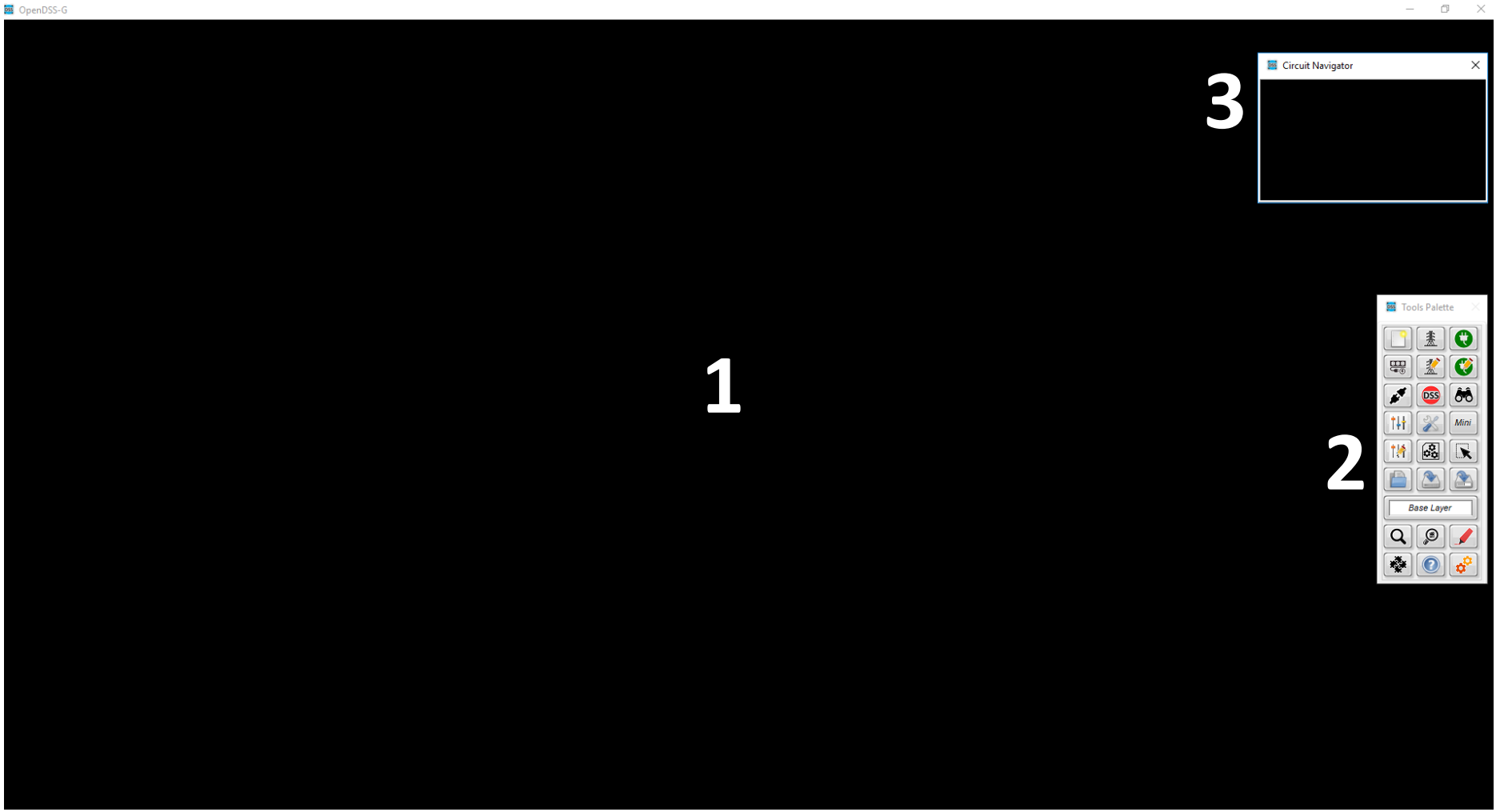
Introduction to OpenDSS-G

- OpenDSS-G



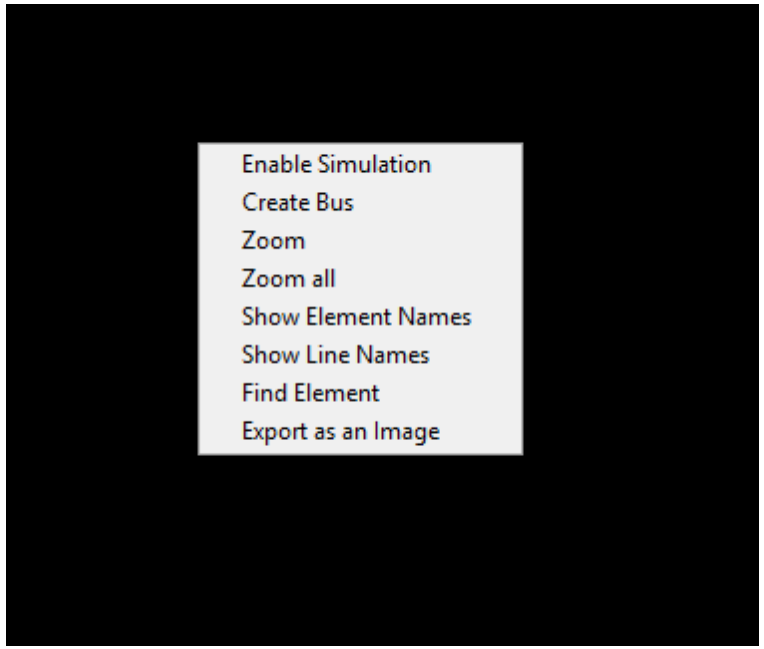
Introduction to OpenDSS-G

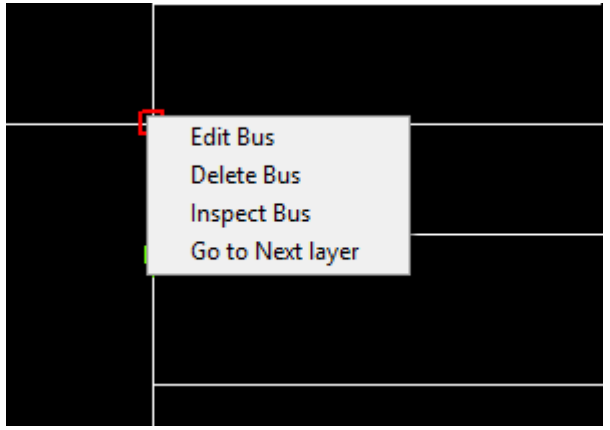
- The workspace



Introduction to OpenDSS-G

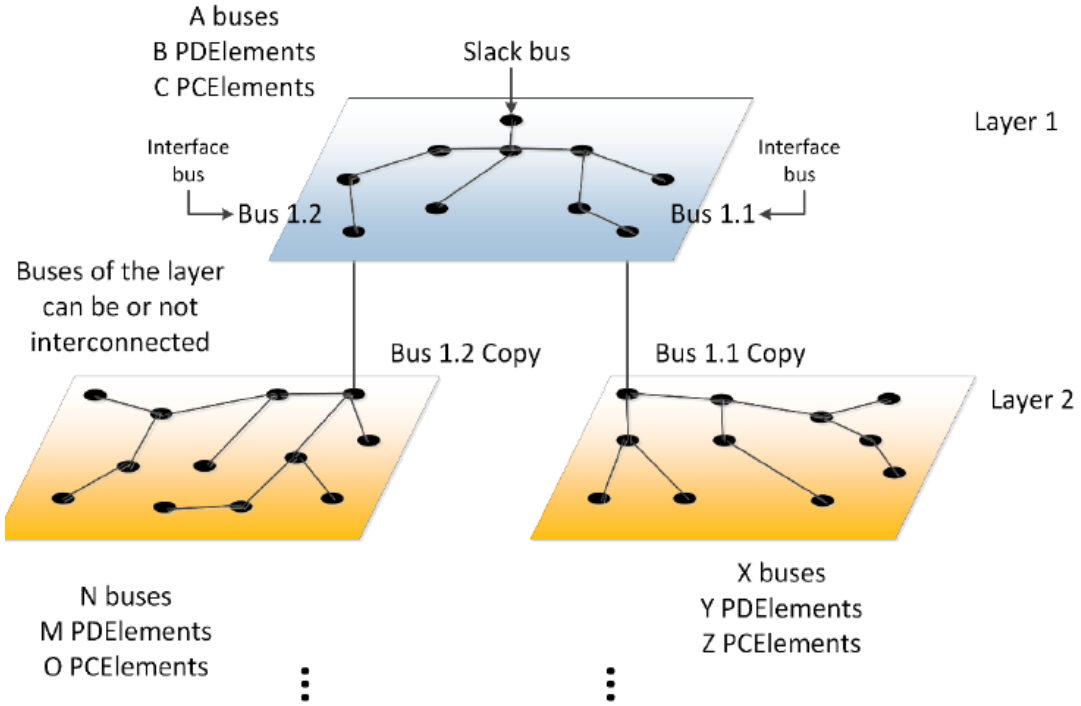
- The workspace

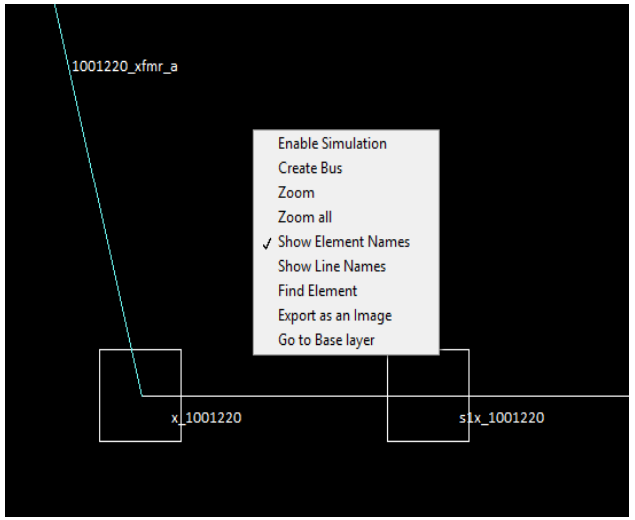


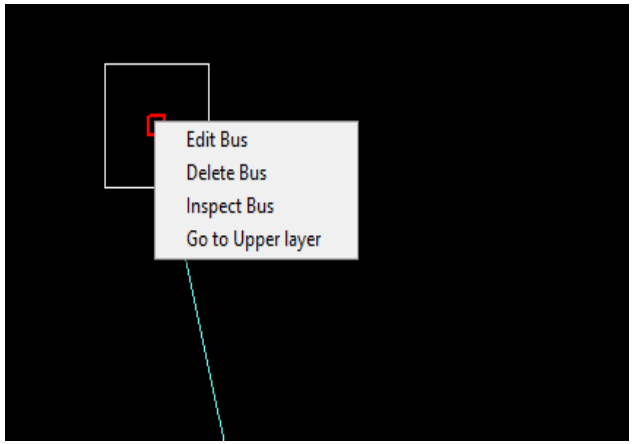


Introduction to OpenDSS-G

- The workspace



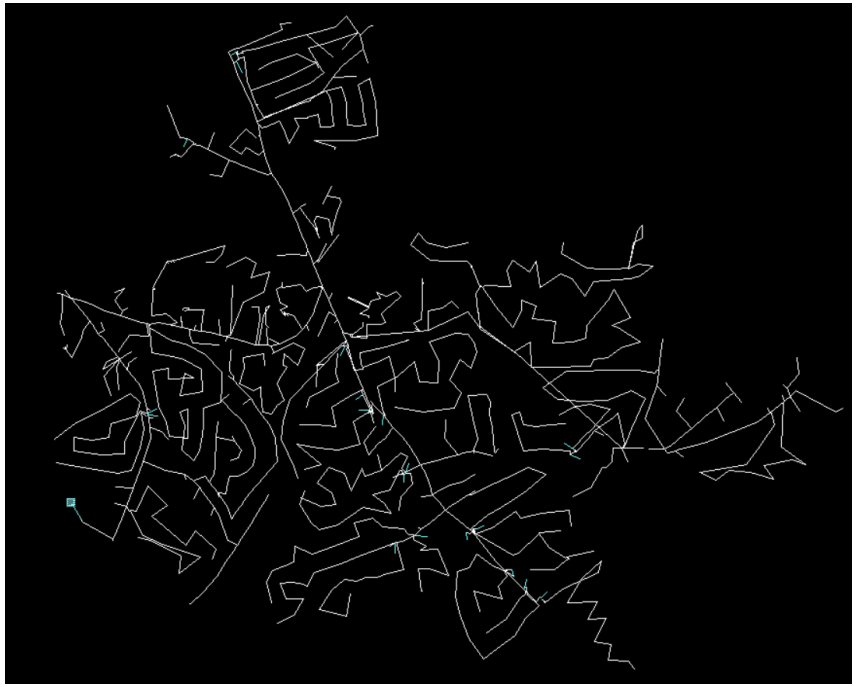




Introduction to OpenDSS-G

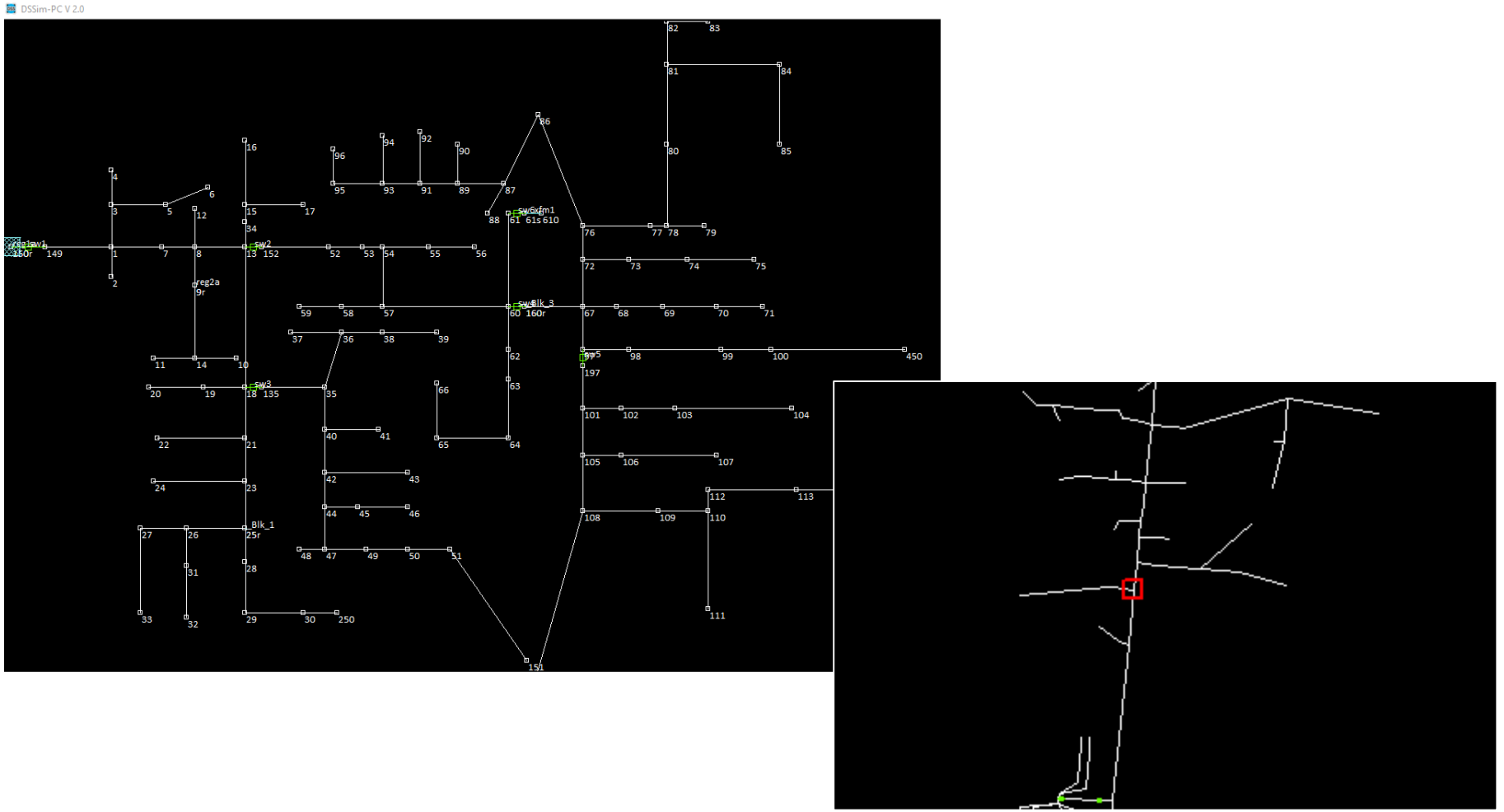
- The workspace

Type	Description	Color
Line	N-phases line	White
Transformer	N-phases, N-windings	Light Blue
Capacitor	N-phases, series capacitor	Pink
Reactor	N-phases, series reactor	Yellow
Opening branch	Switch, recloser, fuse, relay	Light green



Introduction to OpenDSS-G

- The workspace

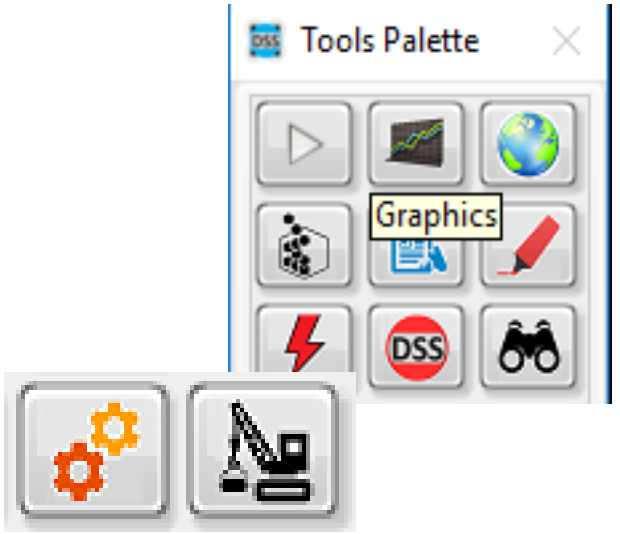


Introduction to OpenDSS-G

- The Tools palette

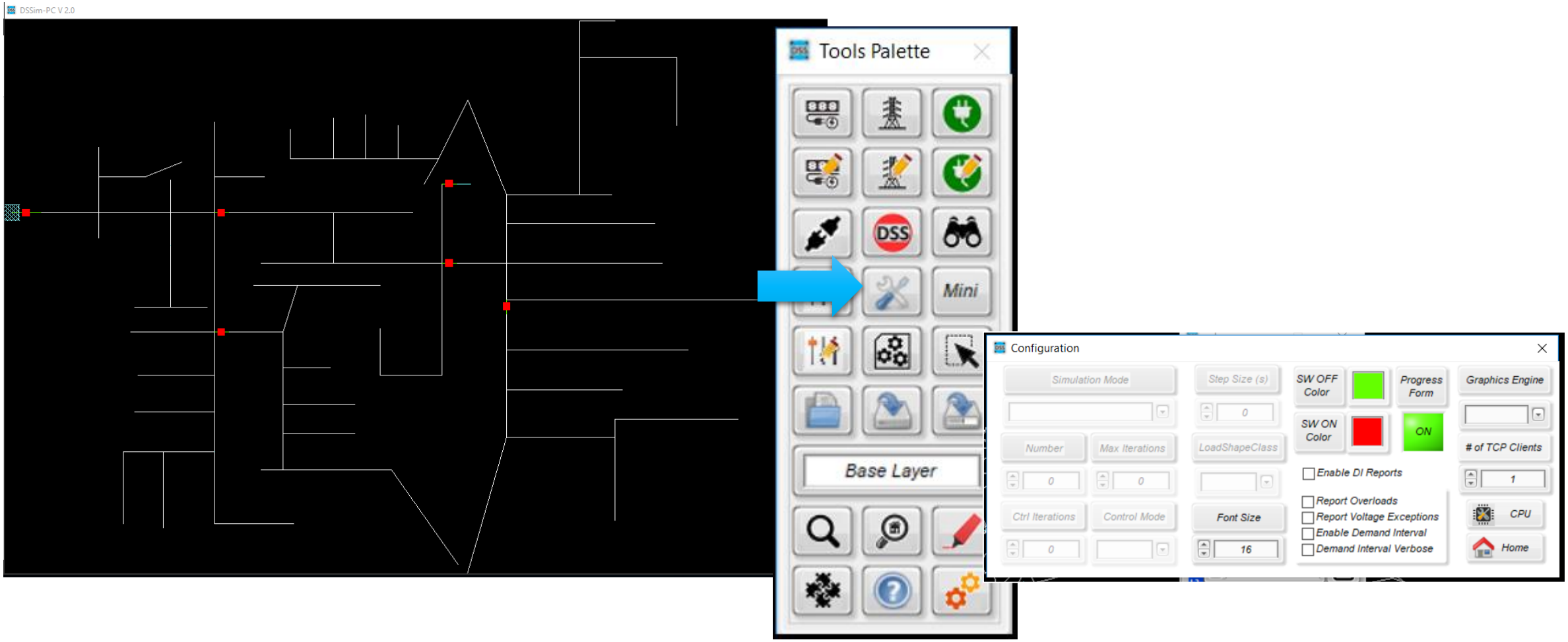
Introduction to OpenDSS-G

- The Tools palette



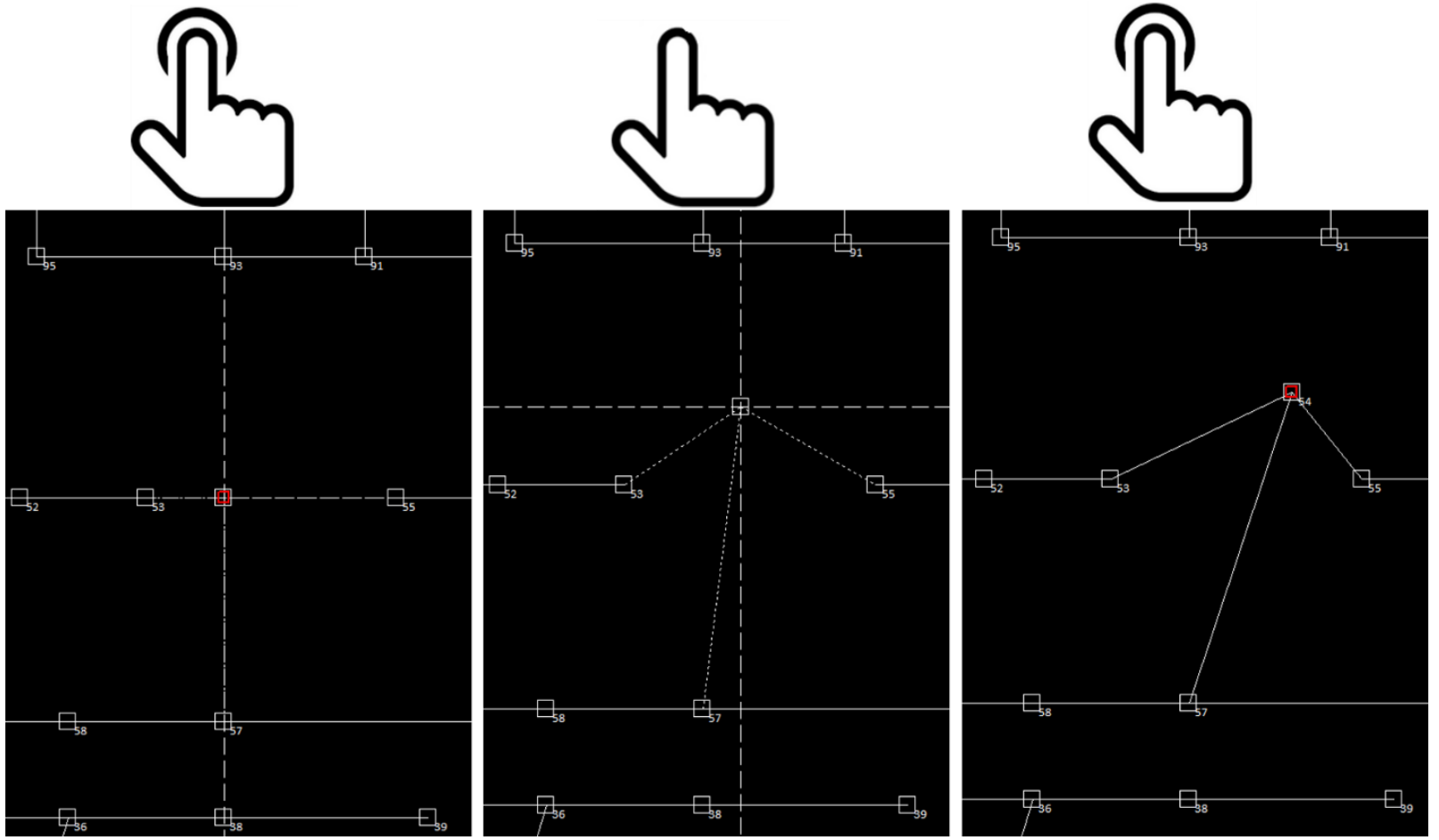
Introduction to OpenDSS-G

- Configuring the environment



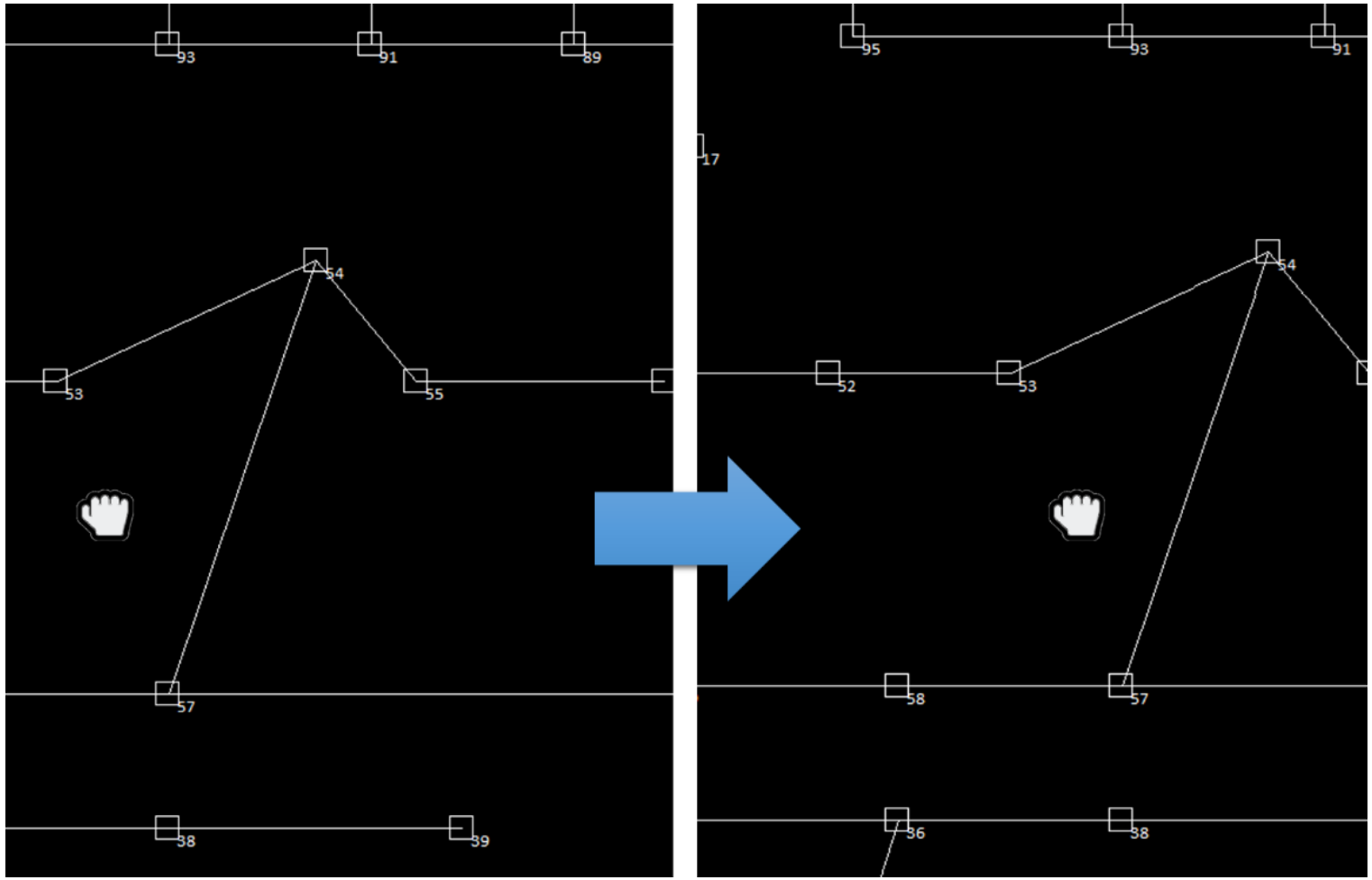
Introduction to OpenDSS-G

- Editing tools



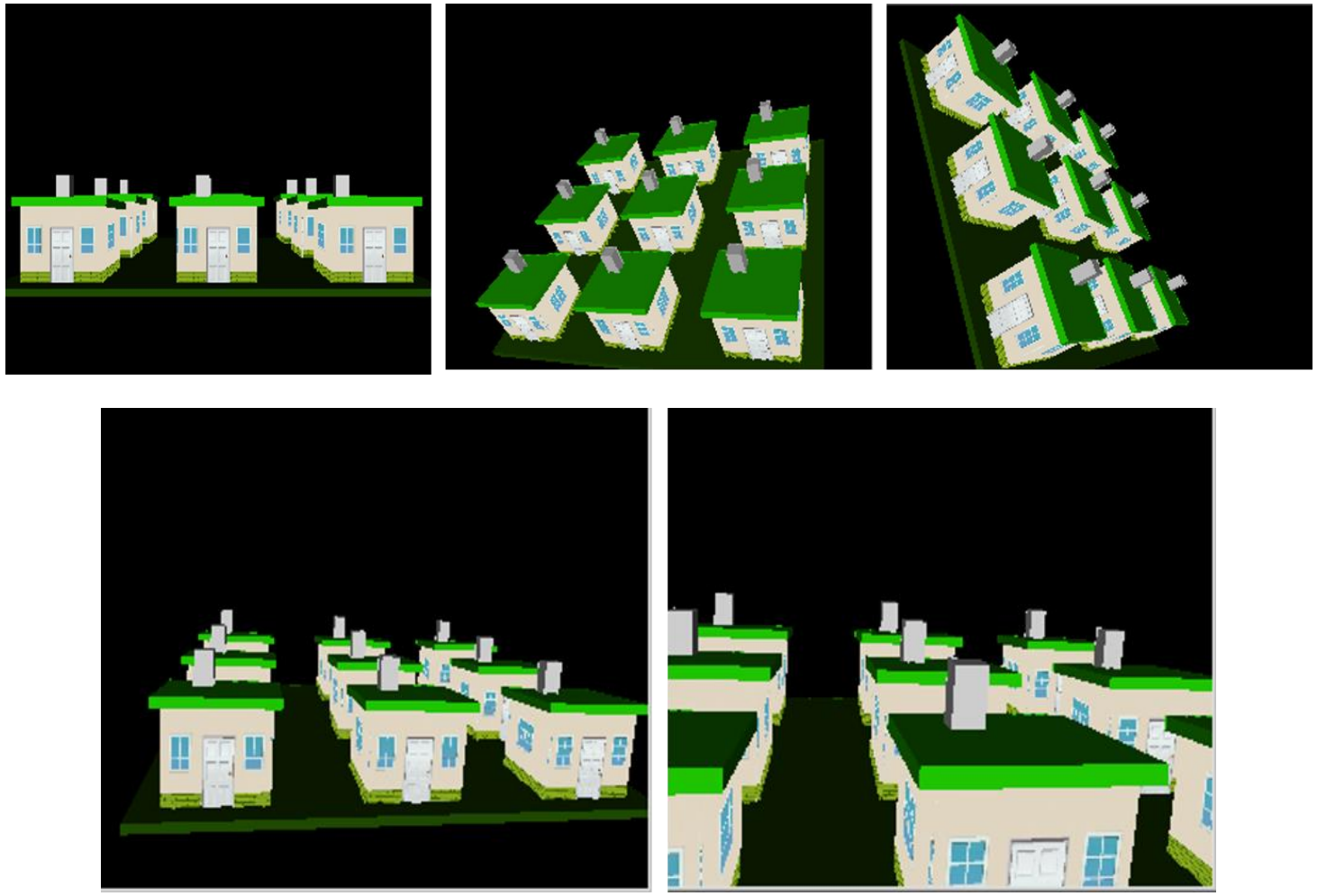
Introduction to OpenDSS-G

- Editing tools



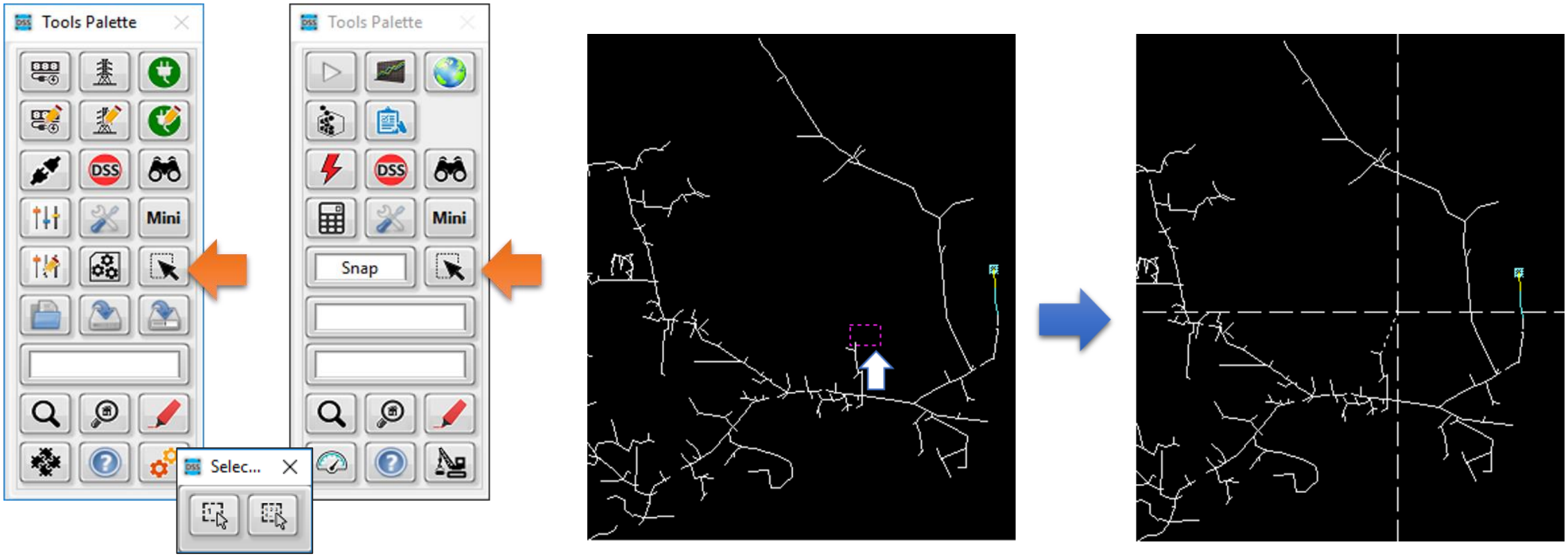
Introduction to OpenDSS-G

- Editing tools



Introduction to OpenDSS-G

- Editing tools

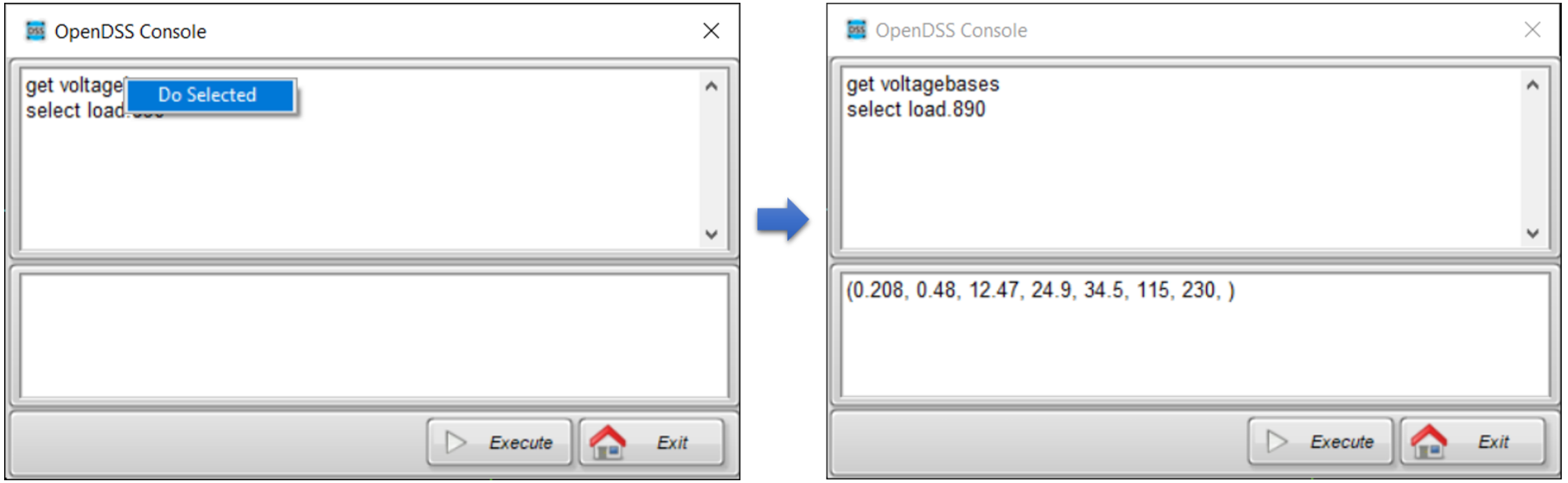


Introduction to OpenDSS-G

- Editing tools

Introduction to OpenDSS-G

- Editing tools



Creating a model

Model creation

- There are a couple of options

1. Importing an existing model

2. Describing your circuit from the scratch

Model creation

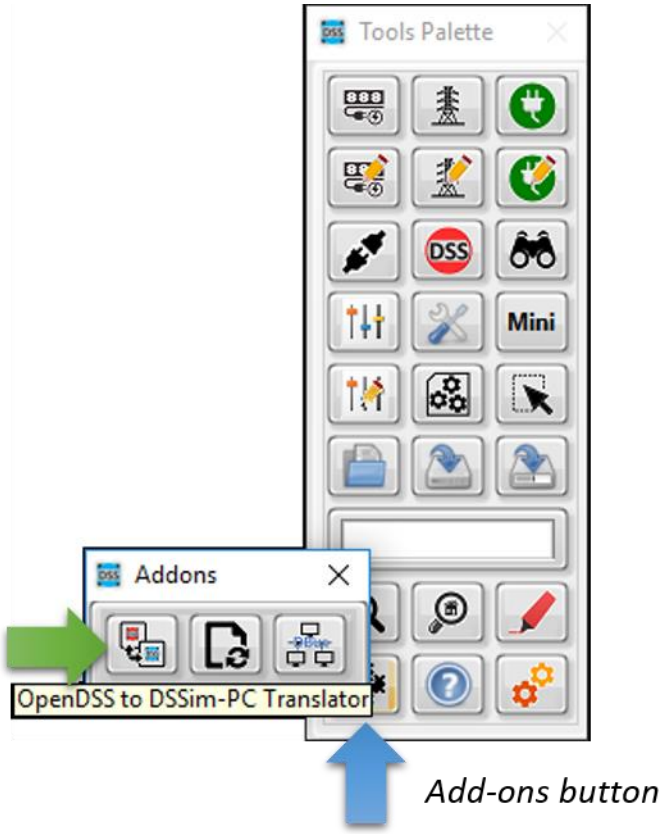
- Before anything

Download the examples to be used in this session:

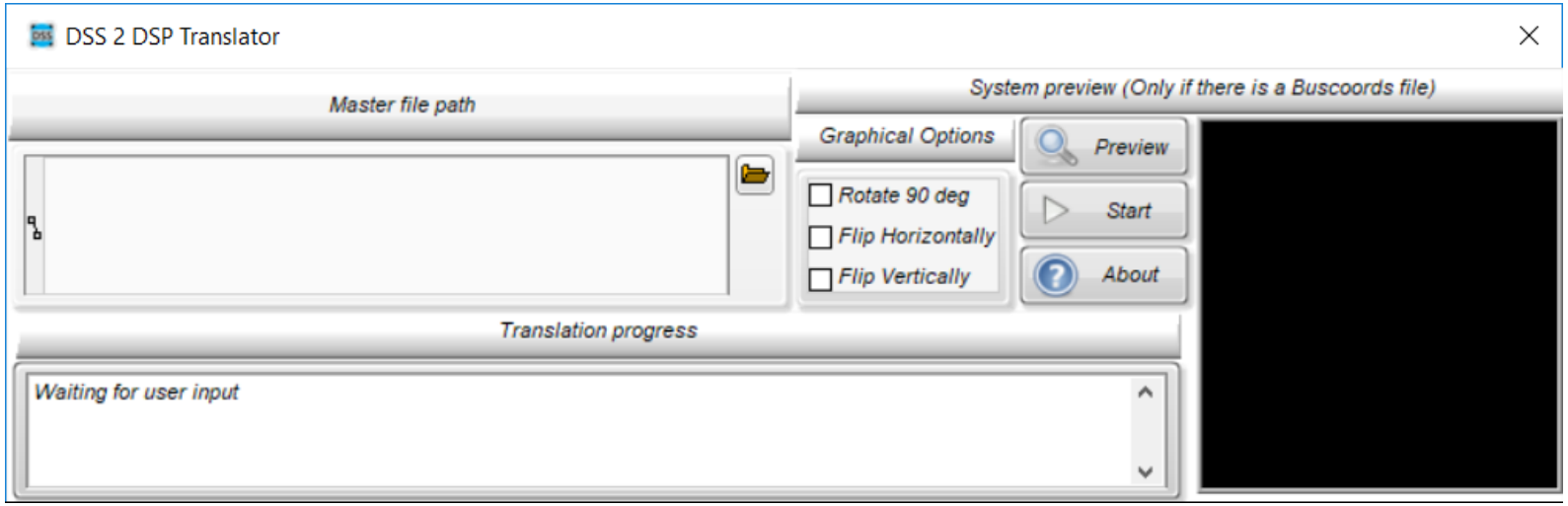
https://sourceforge.net/p/dssimpc/code/HEAD/tree/trunk/Distribution/Examples/

Model creation

- Importing a model



Add-ons button



Model creation

- Importing a model

Let’s import an OpenDSS circuit

Model creation

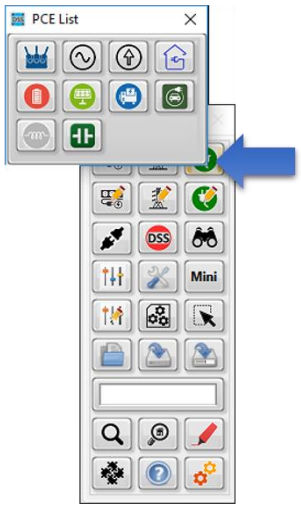
- Building the model from the scratch

a)

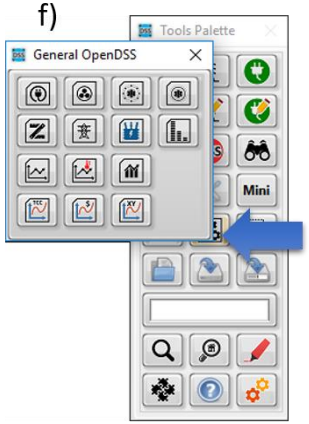
b)

c)

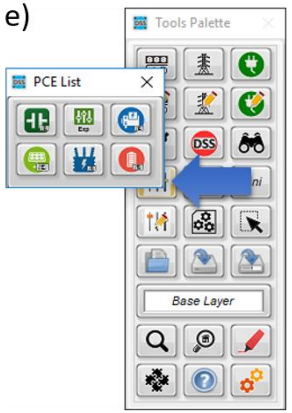
d)

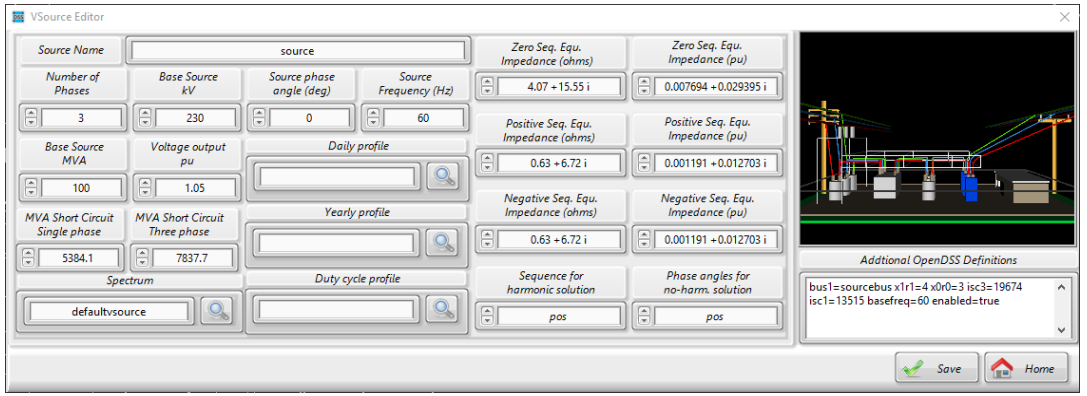


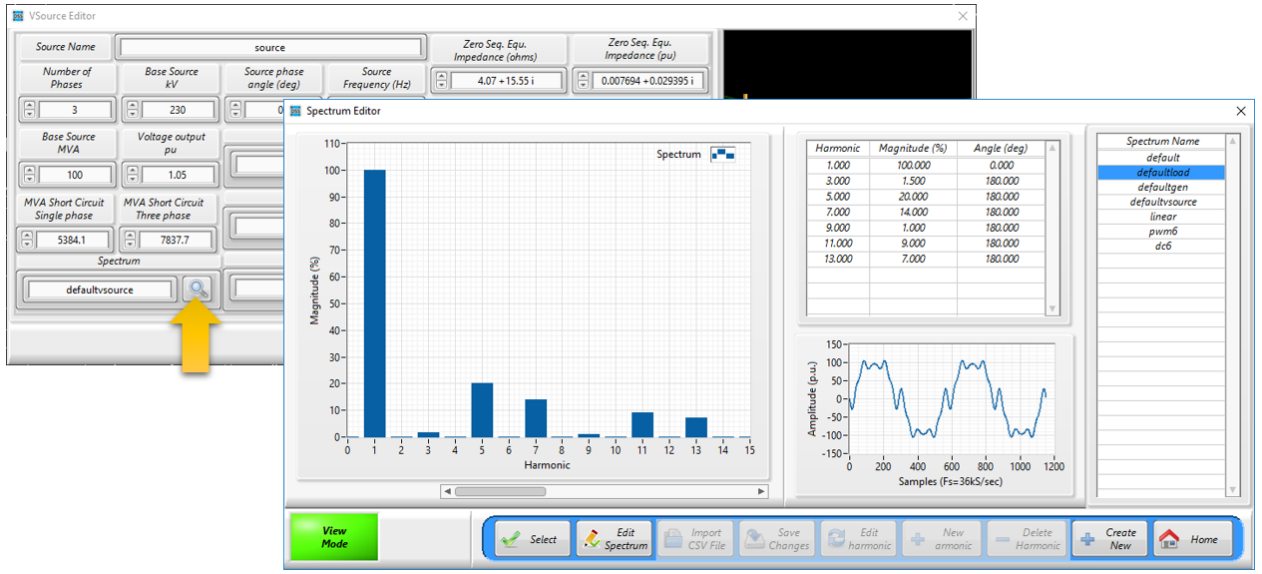
f)



e)

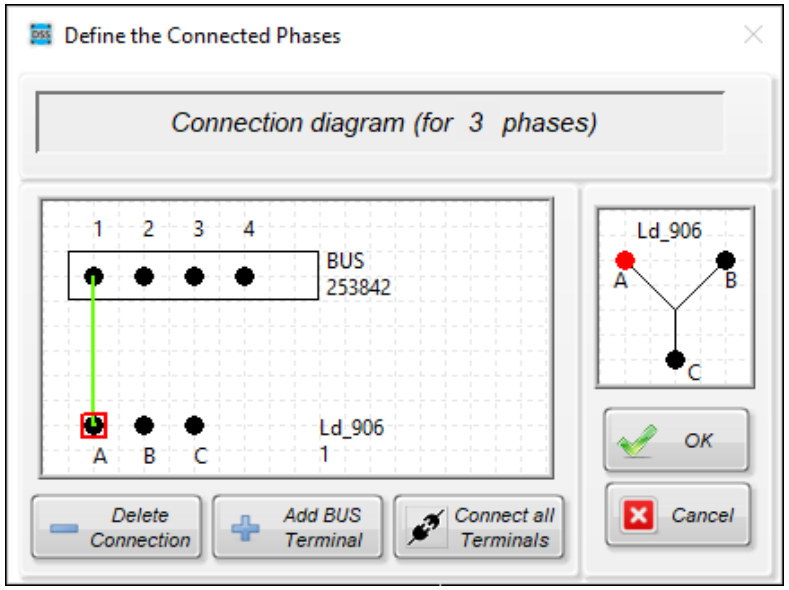






Model creation

- Building the model from the scratch



Rules:

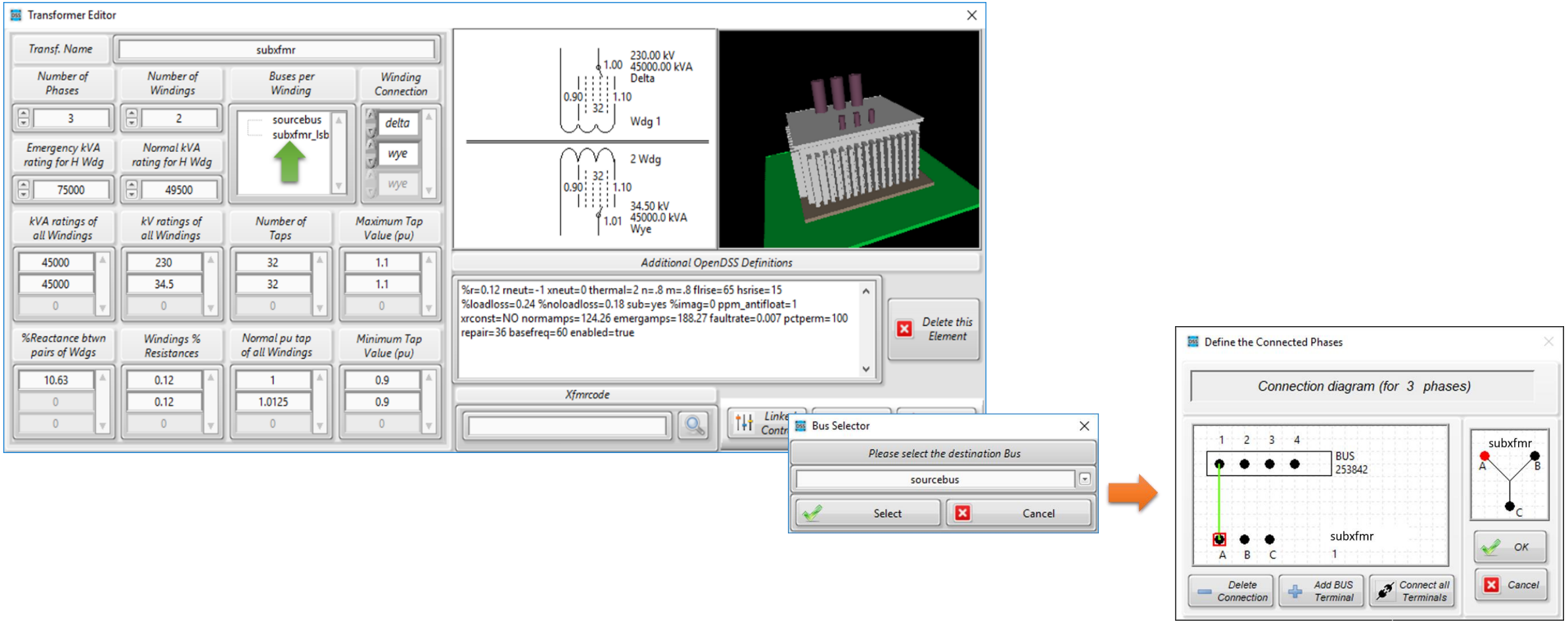
1. Connect your PD/PCElement to an existing bus

2. Follow the instructions

Model creation

- Building the model from the scratch

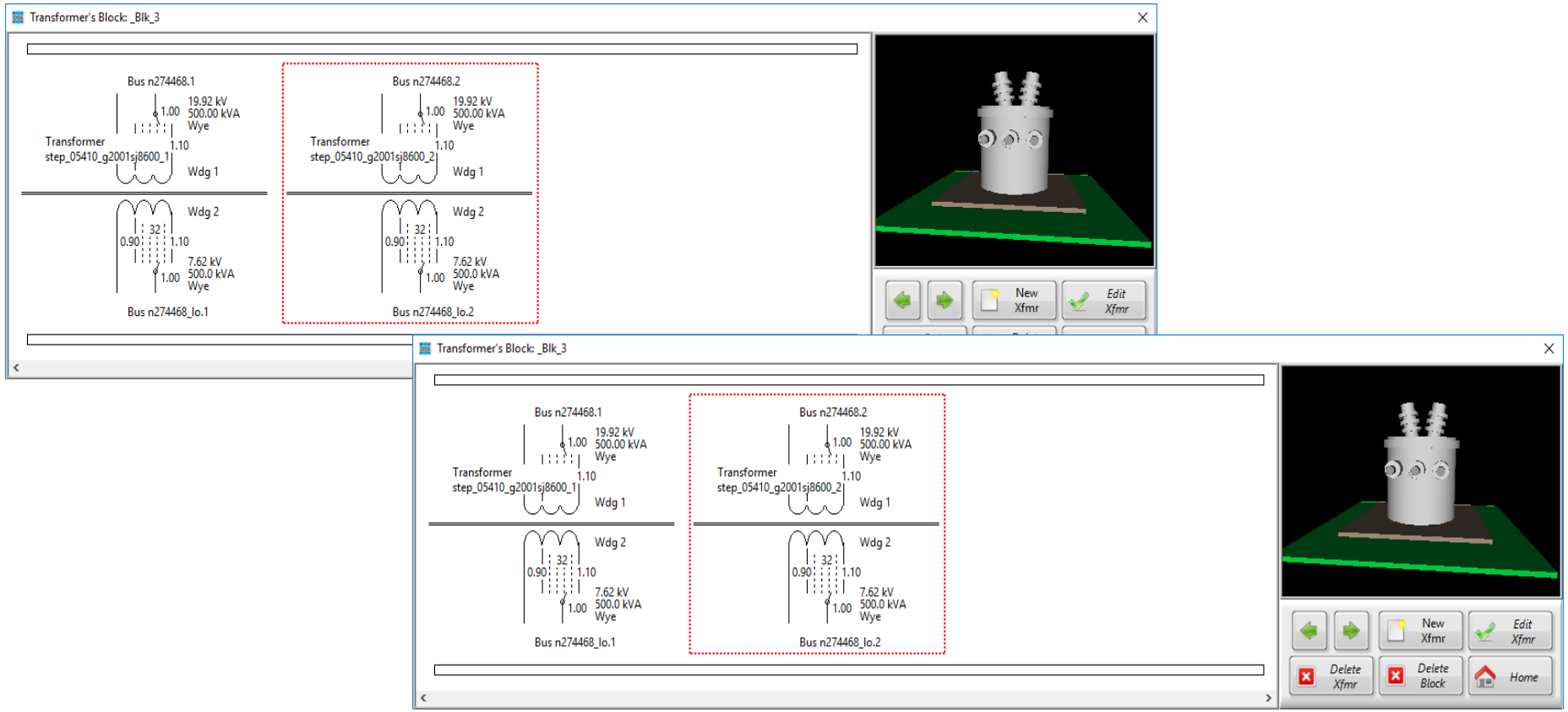
(Special elements)



Model creation

- Building the model from the scratch

(Special elements)



Model creation

- Building the model from the scratch

Let’s build a simple circuit

Today's challenge

Today's challenge

Given the system at
https://sourceforge.net/p/dssimpc/code/HEAD/tree/trunk/Distribution/Examples/Ckt_7_Storage/

Check if the system reports any overload during a yearly simulation, if it does, try to mitigate the issues using storage, PV or reconductoring the line

Together...Shaping the Future of Electricity