IOWA STATE UNIVERSITY
ECpE Department

OpenDSS - Introduction

Dr. Zhaoyu Wang
1113 Coover Hall, Ames, IA
wzy@iastate.edu

Contents

- General introduction of OpenDSS
- Installation & User interface
- Workflow
- Overall concept
- Overall structure of solving power flow

Introduction

What is OpenDSS?

The Open Distribution System Simulator (OpenDSS, or simply, DSS) is a comprehensive electrical system simulation tool for electric utility **distribution systems**.

- Open → Open Source
- DSS → Distribution System Simulator

- The Development of OpenDSS began in April 1997. **Roger Dugan** is the principal author of the software.
- In 2004, the DSS had been acquired by EPRI Solutions.
- In 2008, EPRI released the software under an open-source license to cooperate with other grid modernization efforts.

Objectives

1. Develop an **object-oriented** circuit description language that minimize the conversion effort (files in different format).
2. Build a **user-friendly** simulation software.
3. Build a tool that can model several substations and the distribution circuits between them simultaneously.
4. Capture both the **time- and location-dependent** value of DG.
5. Simulate some **untypical** situations.
6. Build a tool that could seamlessly incorporate harmonics analysis into the powerflow analysis without requiring the user to laboriously enter nonlinear device models.

…

EPRI | ELECTRIC POWER RESEARCH INSTITUTE

DSS

Features

1. The OpenDSS supports nearly all **rms steady-state** (i.e., frequency domain) analyses commonly performed for utility distribution systems planning and analysis.
2. In addition, it supports many **new types** of analyses that are designed to meet future needs (e.g. DGs), many of which are being dictated by the deregulation of utilities worldwide and the advent of the "smart grid".
3. **Other features**: support energy efficiency analysis of power delivery, smart grid applications, and harmonics analysis.
4. The DSS is designed to be indefinitely **expandable** so that it can be easily modified to meet future needs.

EPRI | ELECTRIC POWER RESEARCH INSTITUTE

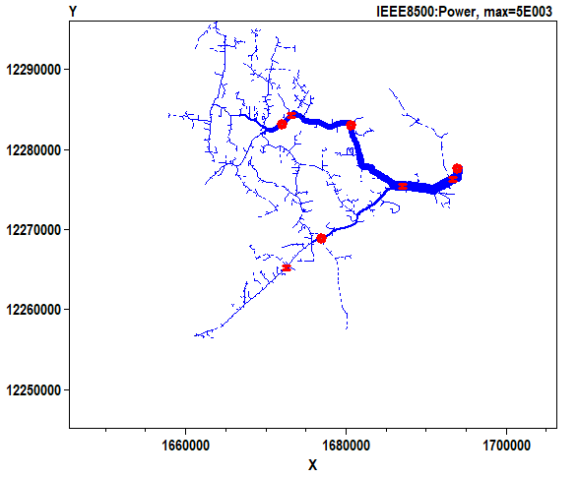
Introduction

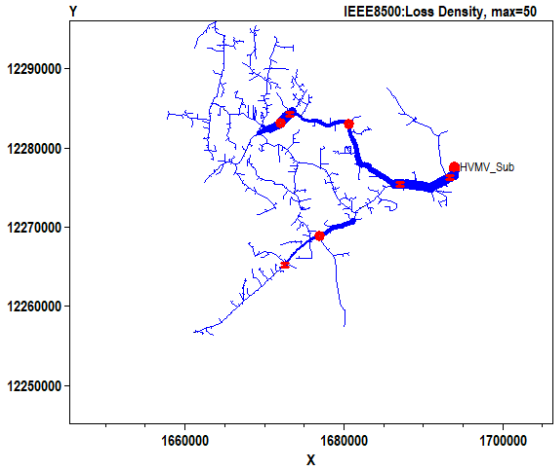
What can OpenDSS do?

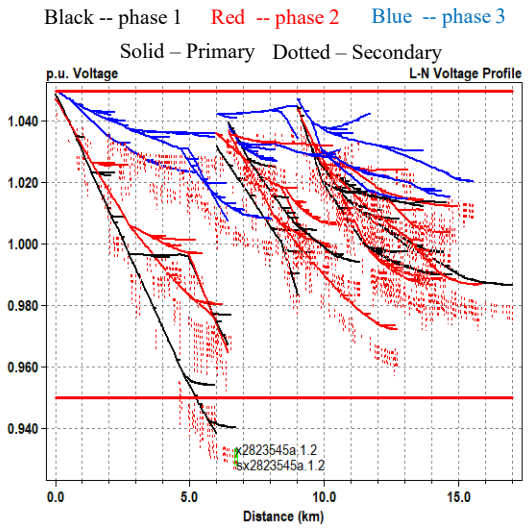
- **Power flow analysis**

  When a power flow is completed, the losses, nodal voltages, currents and power flows are available for the total system, each component, and certain defined areas.

  - OpenDSS can handle both ***radial*** distribution (MV) circuits and ***network*** (meshed) systems.
  - It can also be used to solve ***small***- to ***medium-sized*** networks with a transmission-style power flow.
  - The power flow executes in numerous solution ***modes*** including the standard single Snapshot mode, and Daily mode, etc.







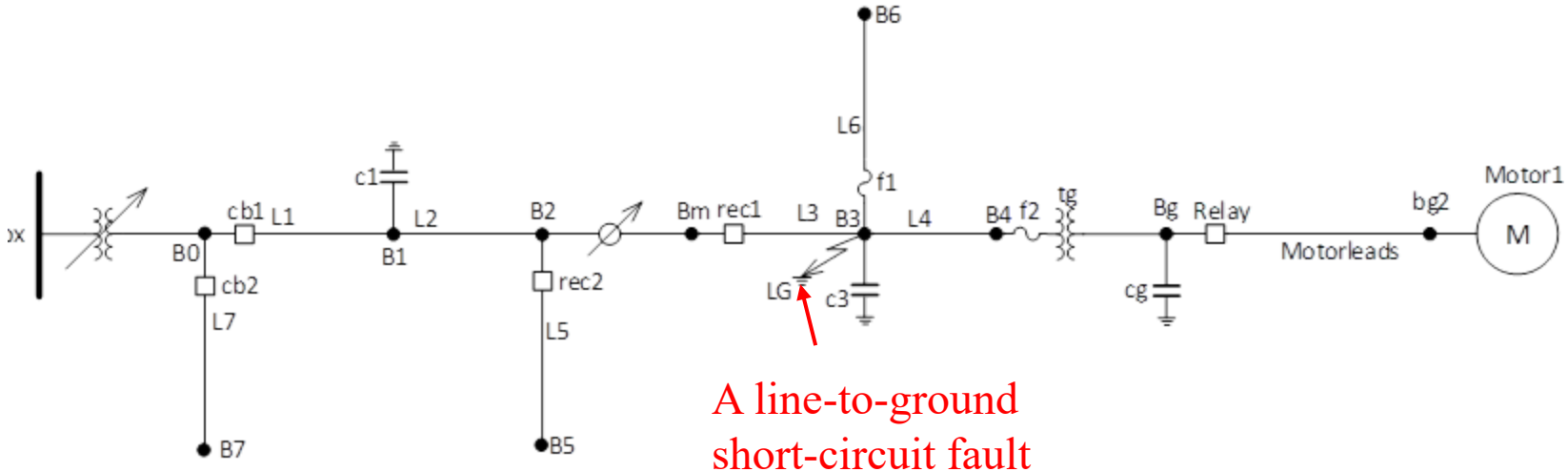
Introduction

What can OpenDSS do?

- **Fault study**

  Fault study can give us the ***short-circuit currents*** and ***voltages***, which can further be used for selecting circuit breakers, setting relays or reclosers, and analyzing the stability of system operation.

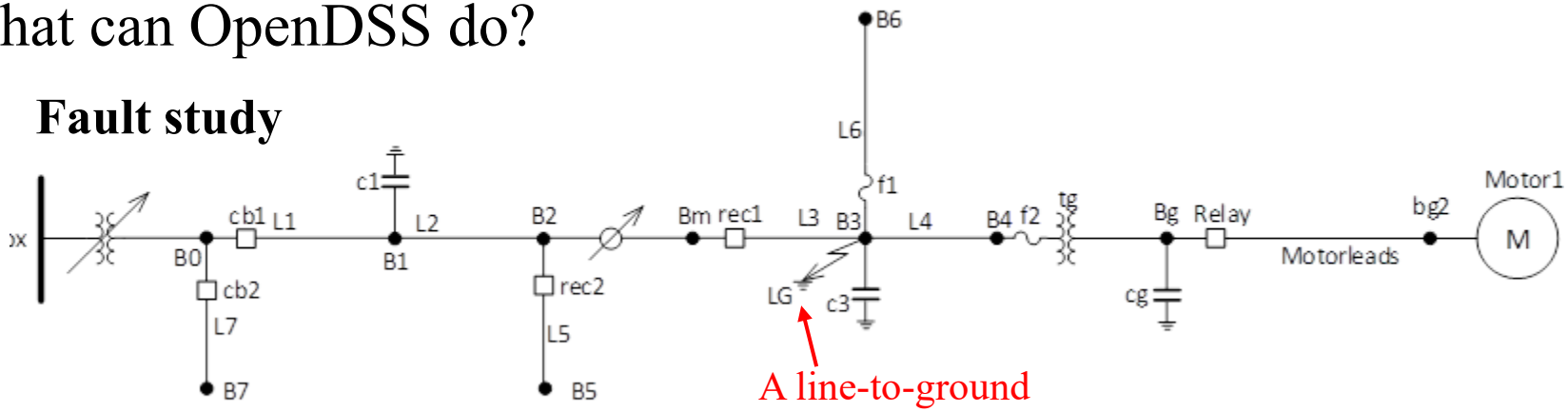
  - OpenDSS can perform fault study for all buses, reporting currents and voltages on all phases for all types of faults, including 3-phase faults, SLG faults on each phase, LL and L-L-G faults.

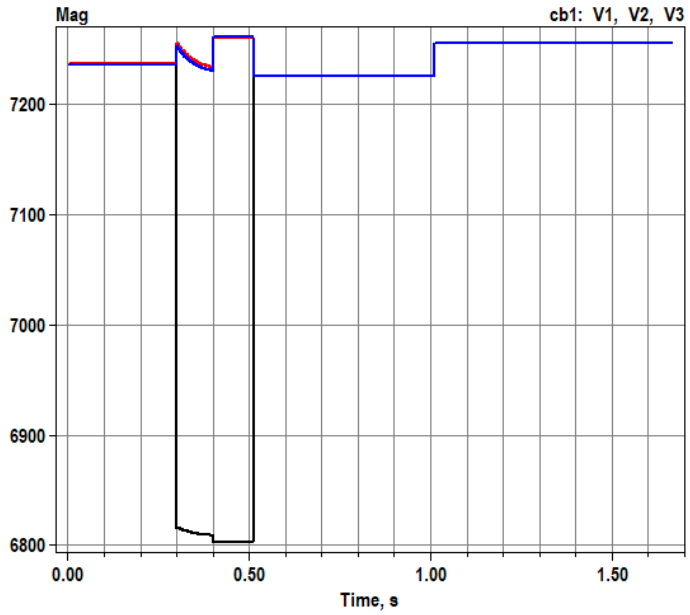


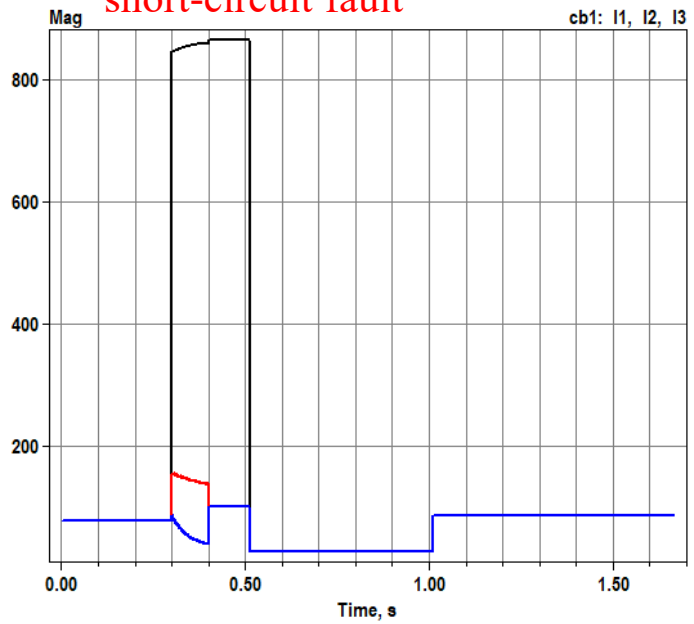
Introduction

What can OpenDSS do?

- **Fault study**







Black -- phase 1

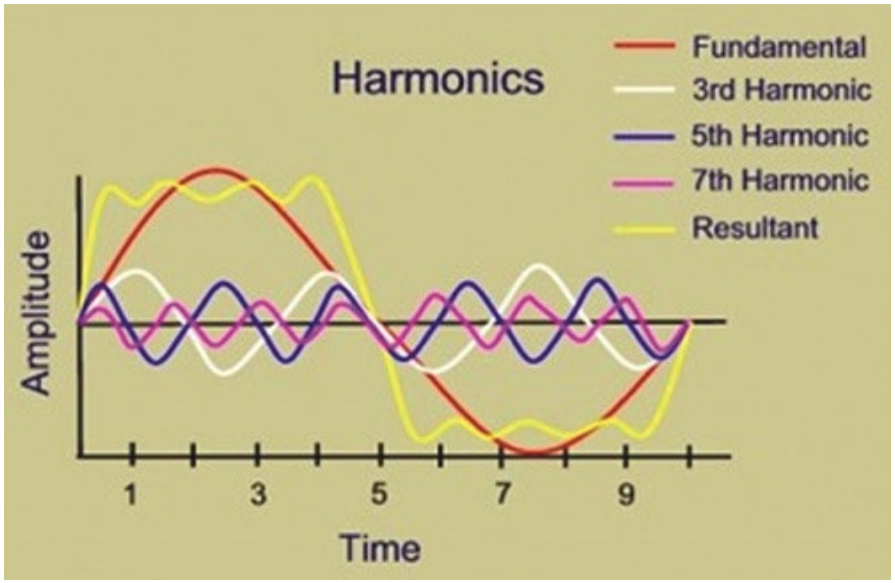
Red -- phase 2

Blue -- phase 3

Introduction

What can OpenDSS do?

- **Harmonic flow analysis**
  It can give us power flow results corresponding to each frequency. The user defines various harmonic spectra to represent harmonic sources of interest. The spectra are connected to Load, Generator, voltage source, current source objects and a few other power conversion elements as desired.



https://www.electricalindia.in/harmonics-in-power-system/
https://electronics.stackexchange.com/questions/264234/why-are-there-no-even-harmonics-in-ac-machines

Introduction

What can OpenDSS do?

- **Dynamics**

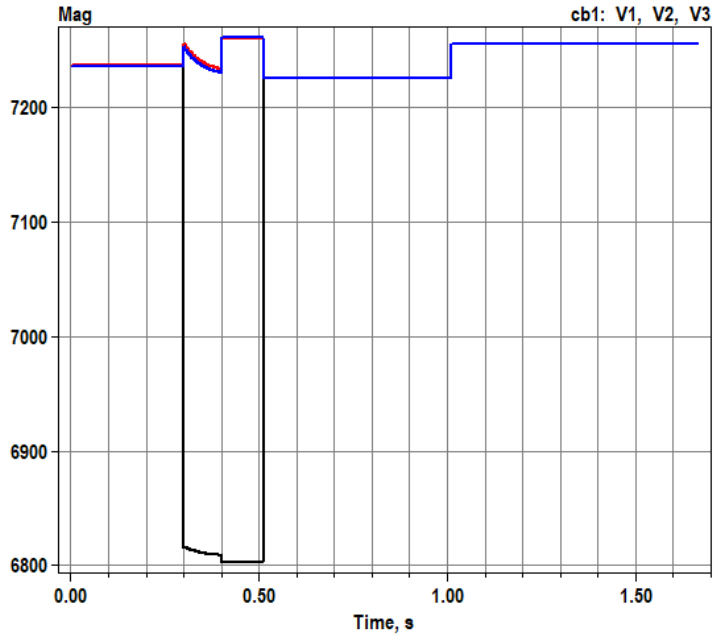
The OpenDSS can perform basic electromechanical transients, or dynamics, simulations. The capability of OpenDSS has been expanding steadily due to needs in inverter modeling and other applications where machine dynamics are important.

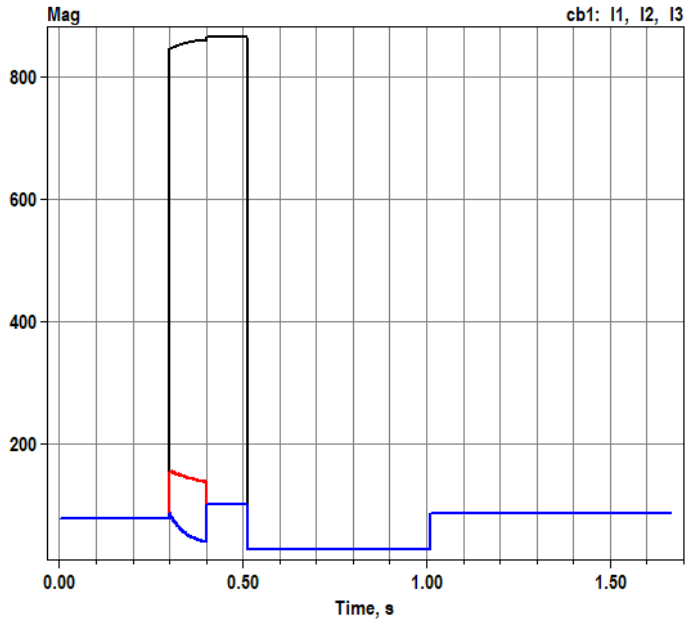
The built-in Generator model has a simple single-mass model that is adequate for many DG studies for common distribution system fault conditions. In addition, users may implement more sophisticated models by writing a DLL for the Generator model or by controlling the Generator model from an external program containing a more detailed governor and/or exciter model.

Introduction

What can OpenDSS do?

- **Dynamics**
  The OpenDSS can perform basic electromechanical transient, or dynamic simulations.





Black -- phase 1

Red -- phase 2

Blue -- phase 3

Introduction

What can OpenDSS do?

- **Load parametric variation**

The capabilities for doing parametric evaluation are provided for a variety of variables. Certain variables will be allowed to vary according to a function (e.g., load growth) or vary randomly for Monte Carlo and statistical studies.

EPRI | ELECTRIC POWER RESEARCH INSTITUTE

DSS

Introduction

What can OpenDSS do?

- **Geomagnetically induced current (GIC) analysis**

OpenDSS is capable of performing geomagnetically induced current (GIC) analysis of power systems. Currently, the analysis capability is limited to three-phase systems, and cannot be integrated into other types of simulations, e.g. load flow. The GIC analysis takes advantage of the N-phase modeling capability of OpenDSS to perform the analysis on a three-phase basis as opposed to a single-phase basis.

EPRI | ELECTRIC POWER RESEARCH INSTITUTE

DSS

Specific Applications

- Distribution Planning and Analysis
- General Multi-phase AC Circuit Analysis
- Analysis of Distributed Generation Interconnections
- Annual Load and Generation Simulations
- Risk-based Distribution Planning Studies
- Probabilistic Planning Studies
- Solar PV System Simulation
- Wind Plant Simulations
- Nuclear Plant Station Auxiliary Transformer Modeling
- Distribution Automation Control Assessment
- Protection System Simulation
- Storage Modeling
- Distribution Feeder Simulation with AMI Data
- Distribution State Estimation
- Ground Voltage Rise on Transmission Systems
- Geomagnetically-Induced Currents (GIC)
- EV Impacts Simulations
- Co-simulation of Power and Communications Networks
- Analysis of Unusual Transformer Configurations
- Harmonic and Interharmonic Distortion Analysis
- Neutral-to-earth Voltage Simulations
- Development of IEEE Test feeder cases
- Phase Shifter Simulation
- Arc Furnace Simulation
- Impulse Loads (car crushers, etc.)
- And more ….

EPRI | ELECTRIC POWER RESEARCH INSTITUTE

DSS

Resources

- OpenDSS Download: https://sourceforge.net/projects/electricdss/
- Forum/Discussion: https://sourceforge.net/p/electricdss/discussion/
- Documents/Files: https://sourceforge.net/projects/electricdss/files/

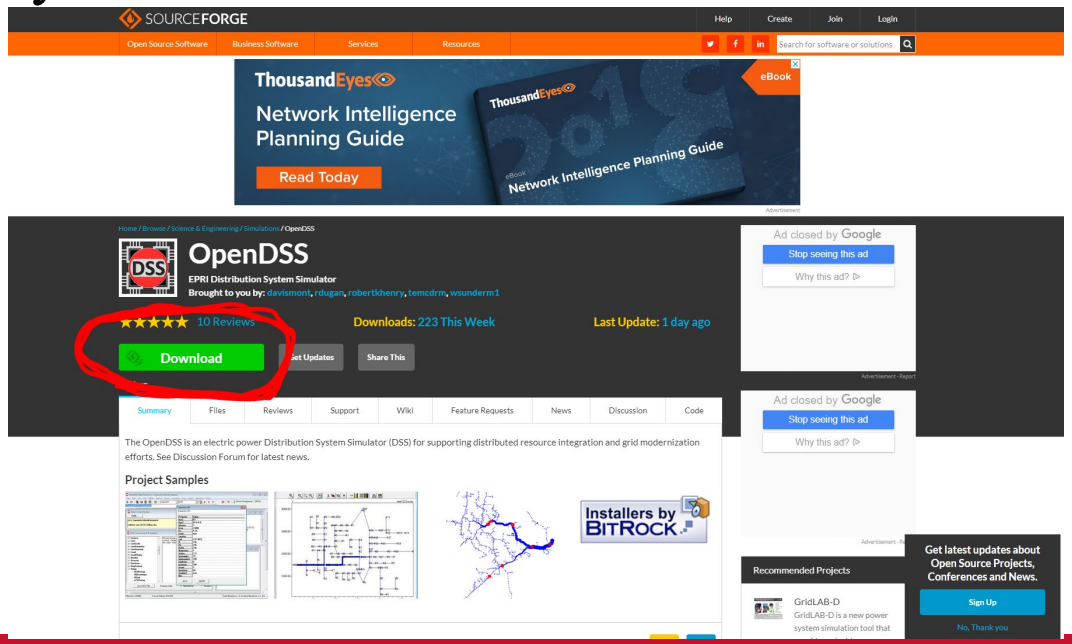
EPRI | ELECTRIC POWER RESEARCH INSTITUTE

Contents

- General introduction of OpenDSS
- Installation & User interface
- Workflow
- Overall concept
- Overall structure of solving power flow

Download

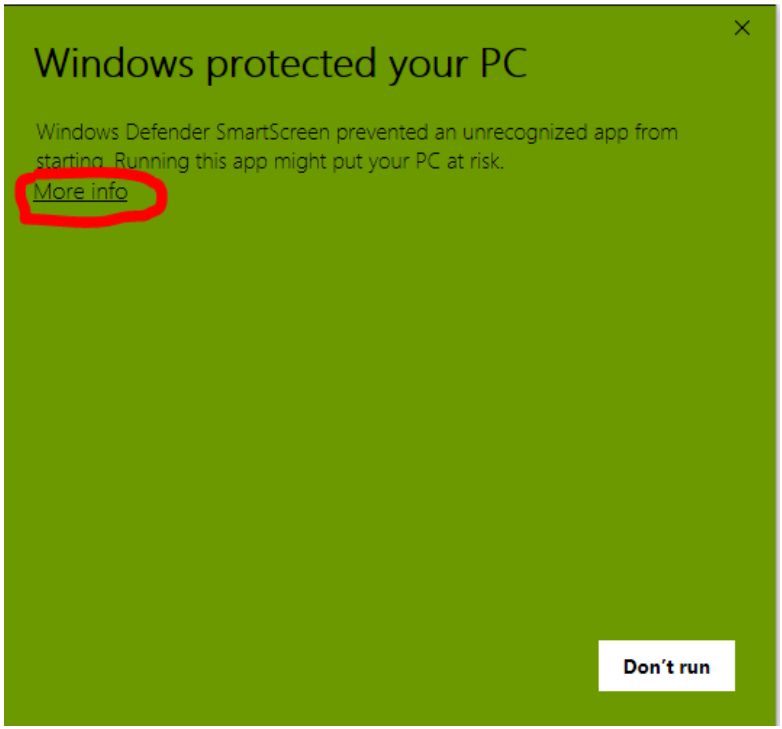
- Click https://sourceforge.net/projects/electricdss/
- Click “Download” button, the software will be downloaded automatically.

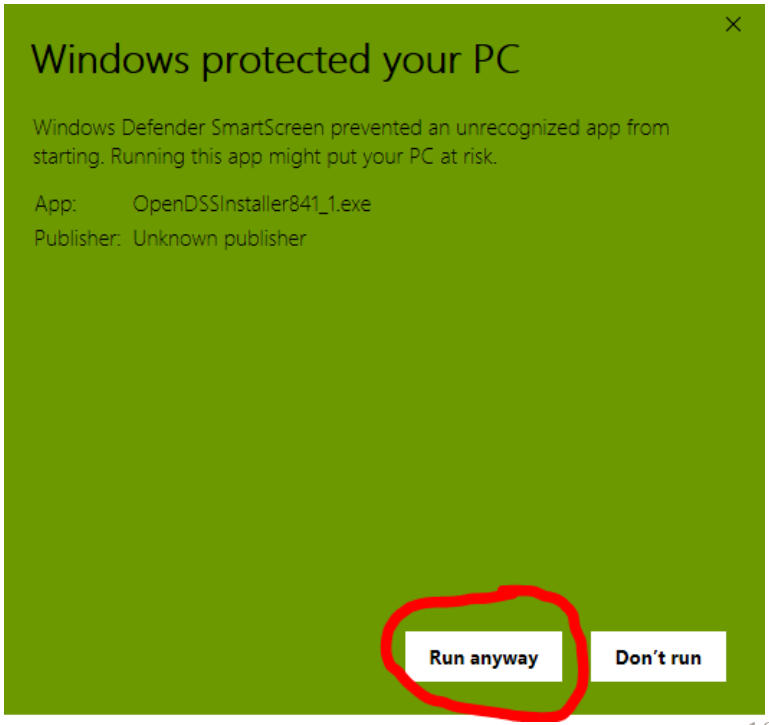


Installation

- Open the downloaded OpenDSSInstaller.

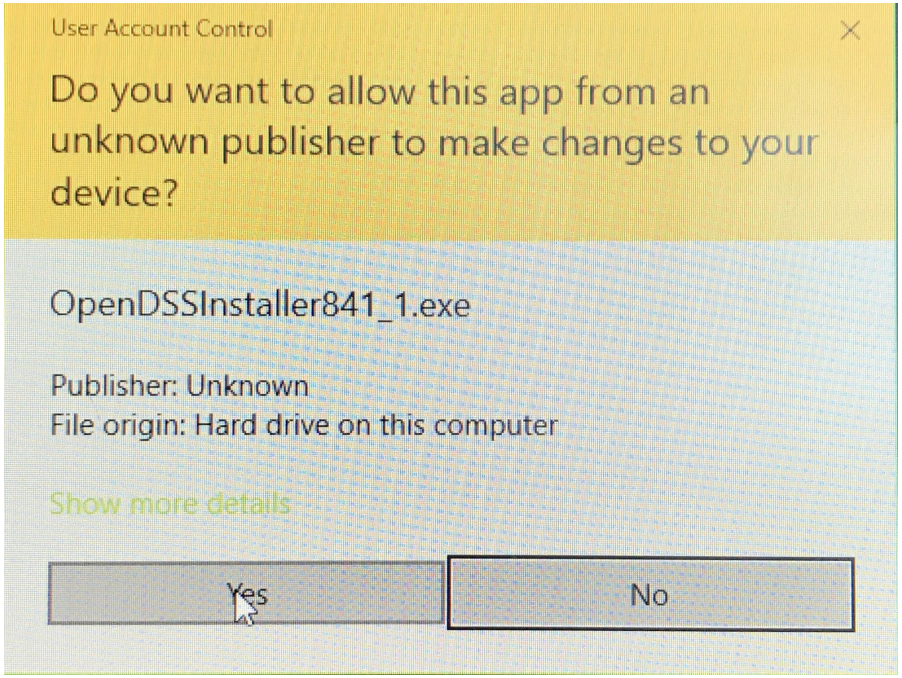
  (If your PC displays the reminder as below, click “more info”, then click “Run anyway”.)





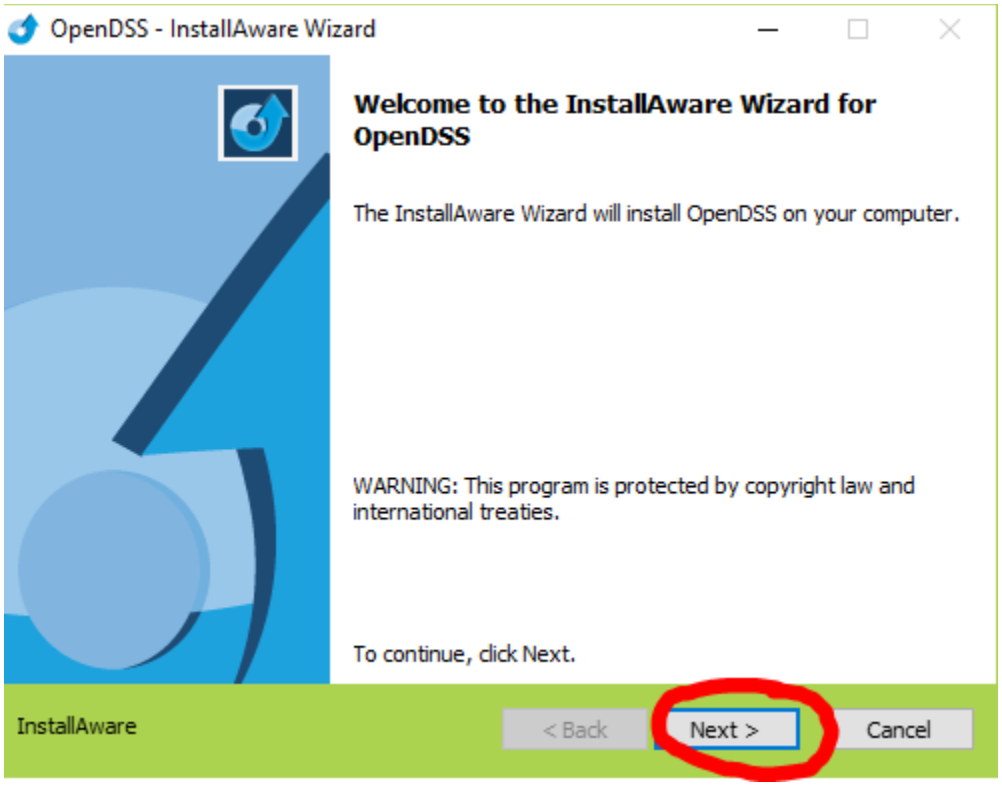
Installation

- Click “Yes” in the popup dialog box.



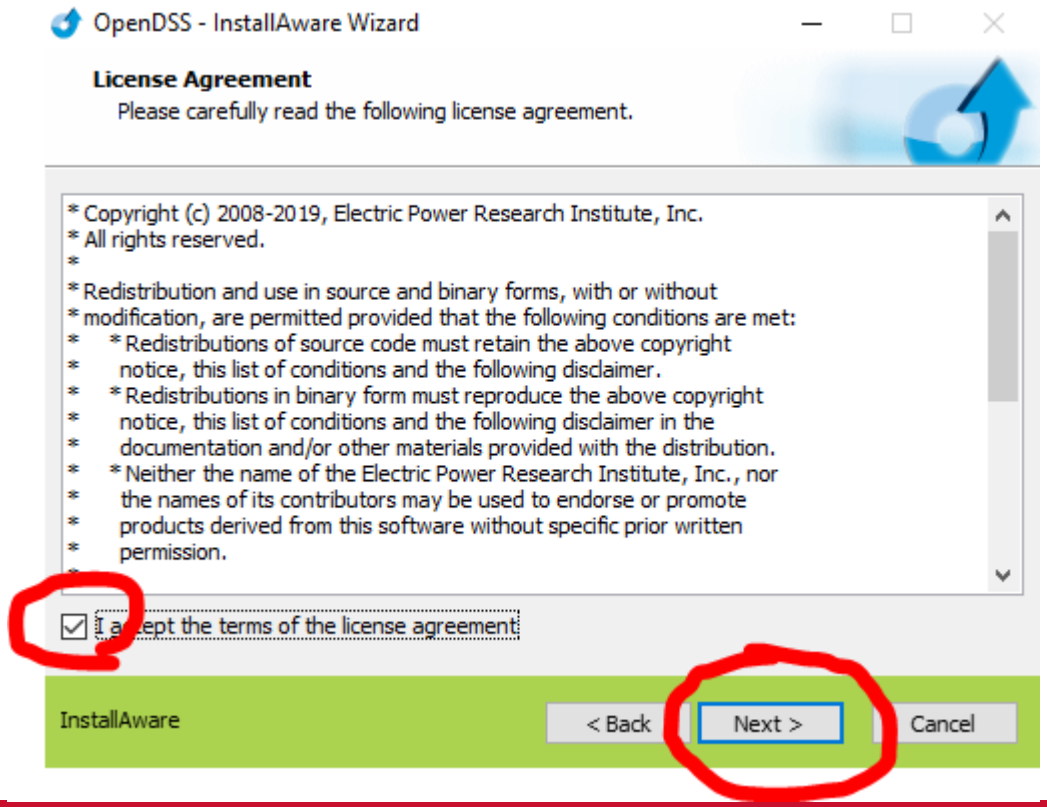
Installation

- Click “Next” button.



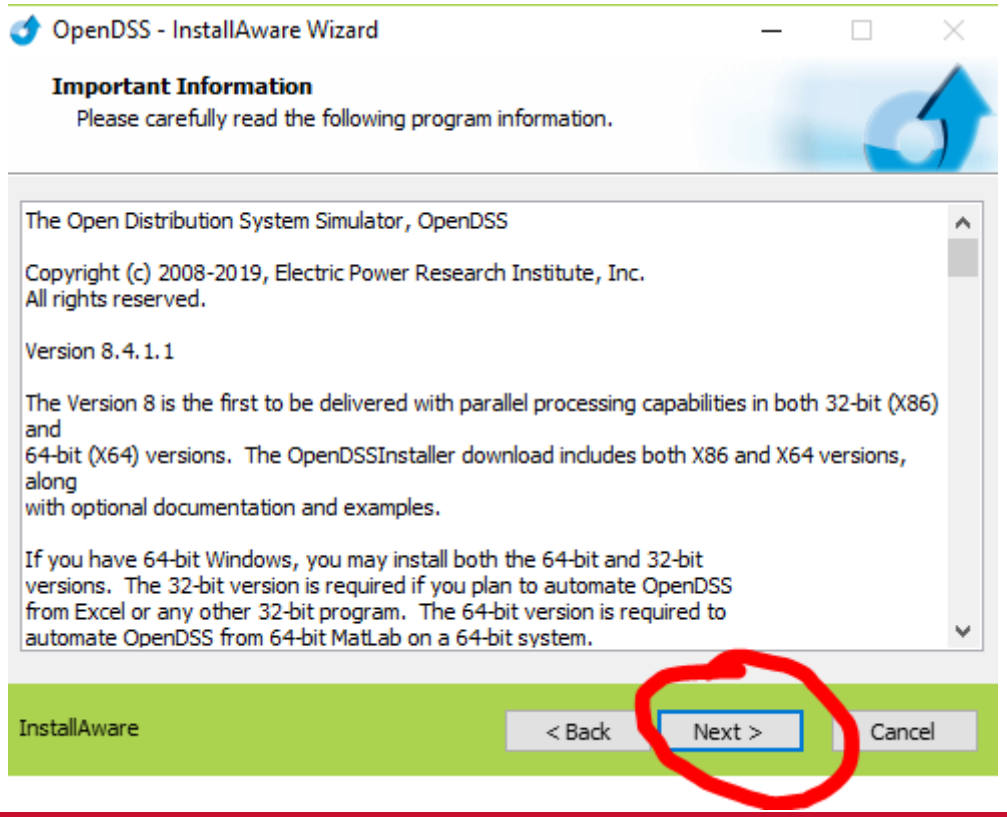
Installation

- Select “I accept the terms of the license agreement” and click “Next”.



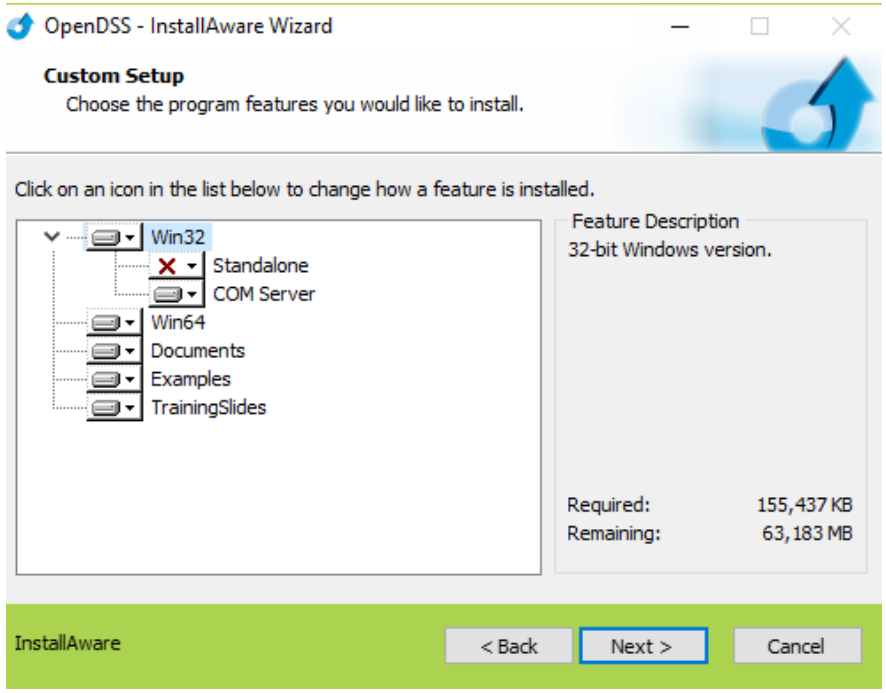
Installation

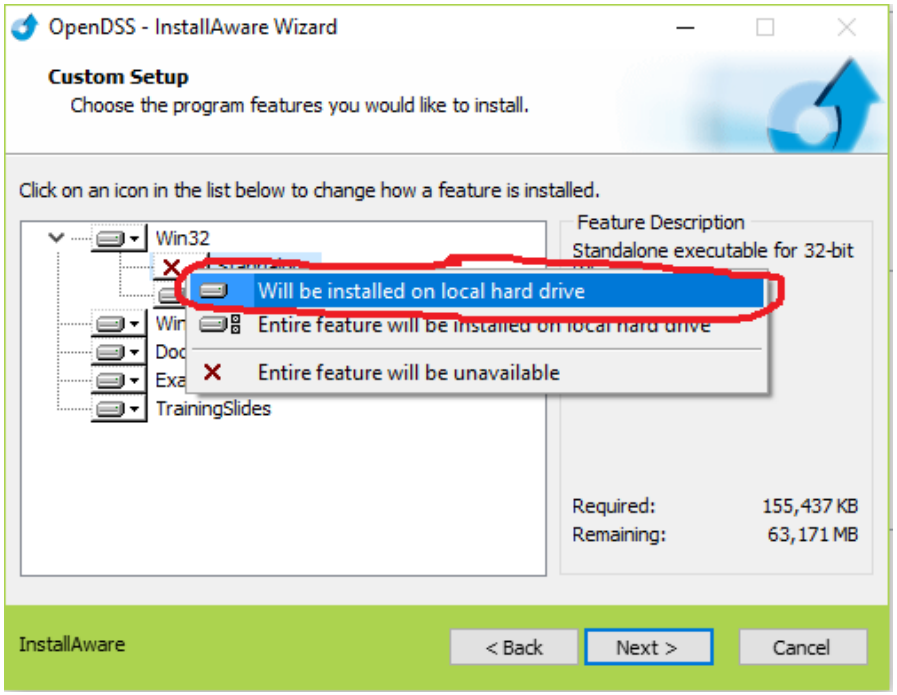
- Click “Next”.



Installation

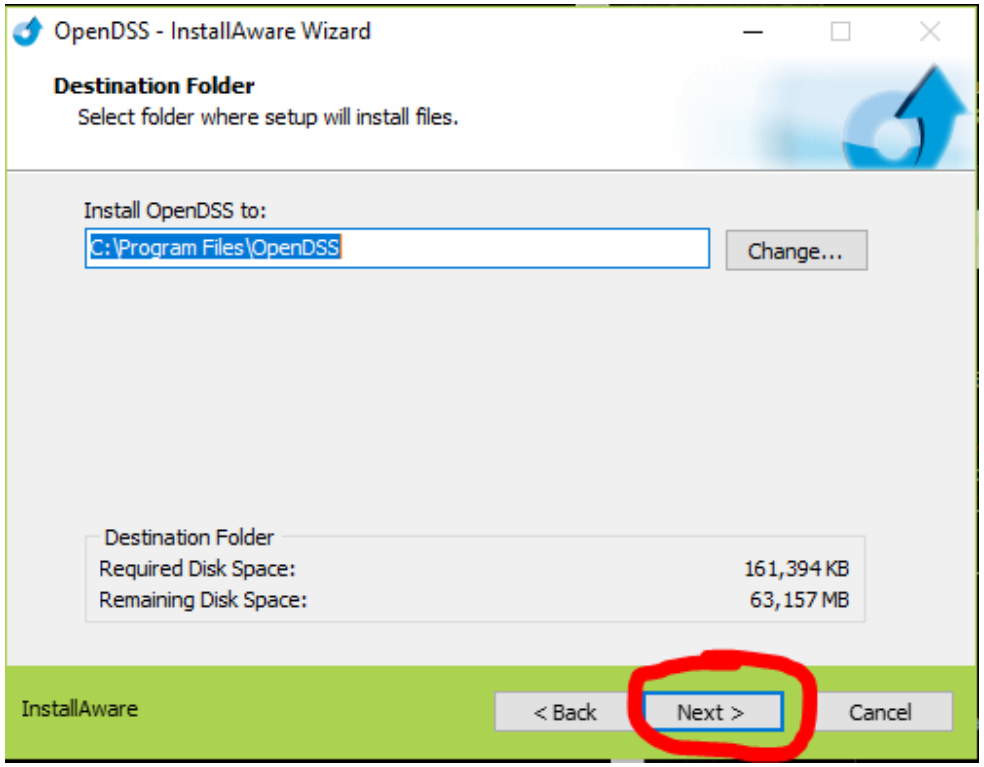
- Then, a dialog box like the picture on the left will appear. Click the red cross and select “Will be installed on local hard drive”.





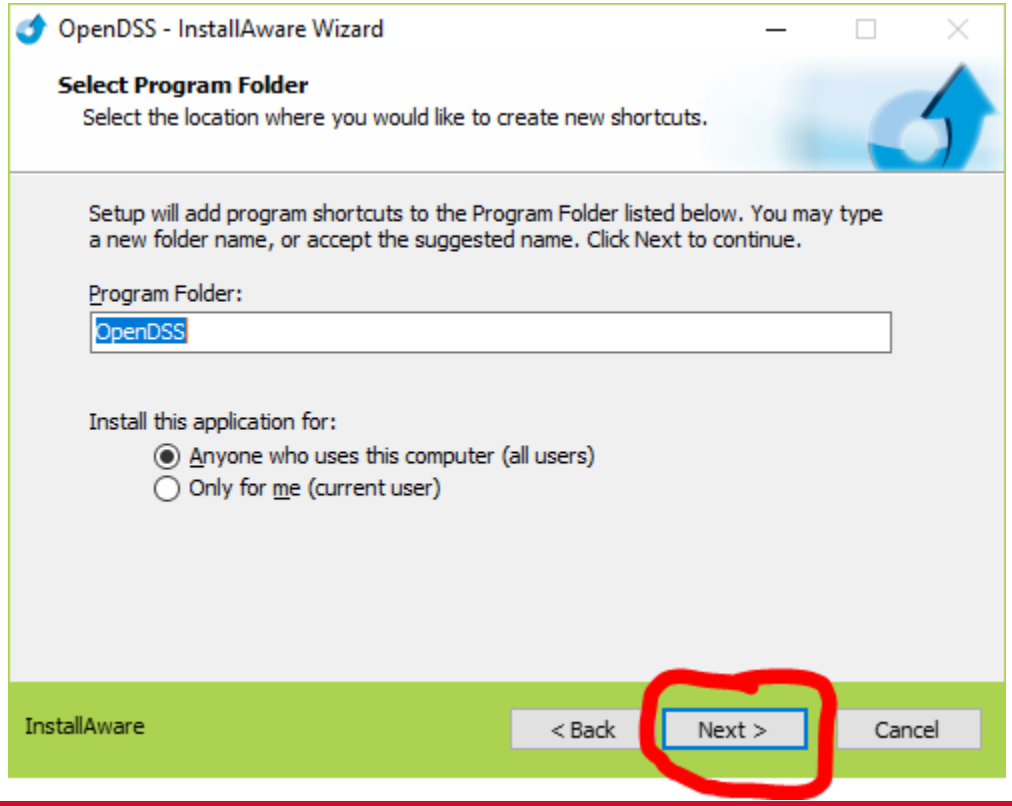
Installation

- Choose the directory where you want to install and click "Next".



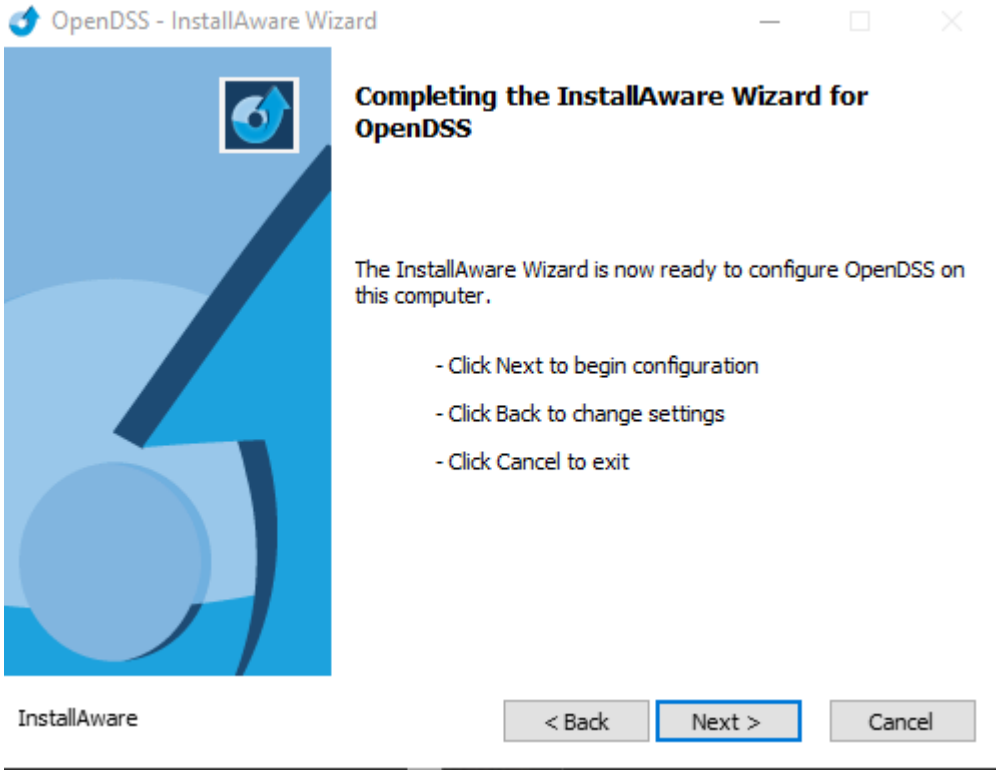
Installation

- Click “Next”.



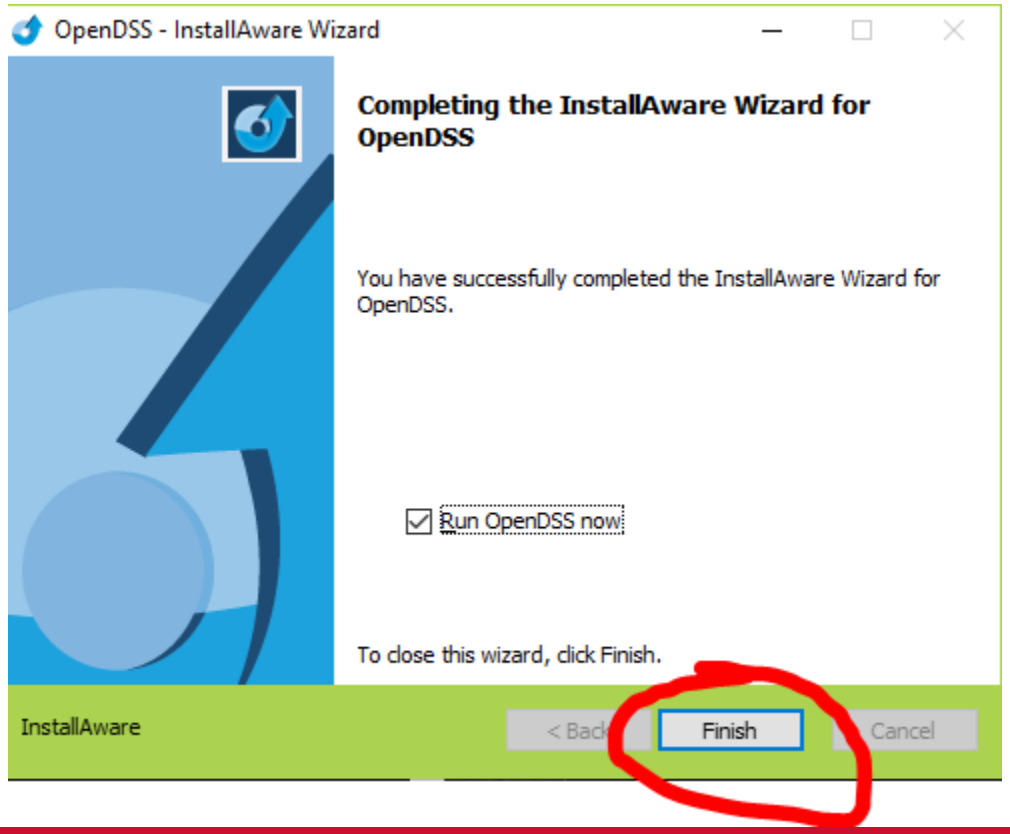
Installation

- Click “Next”.



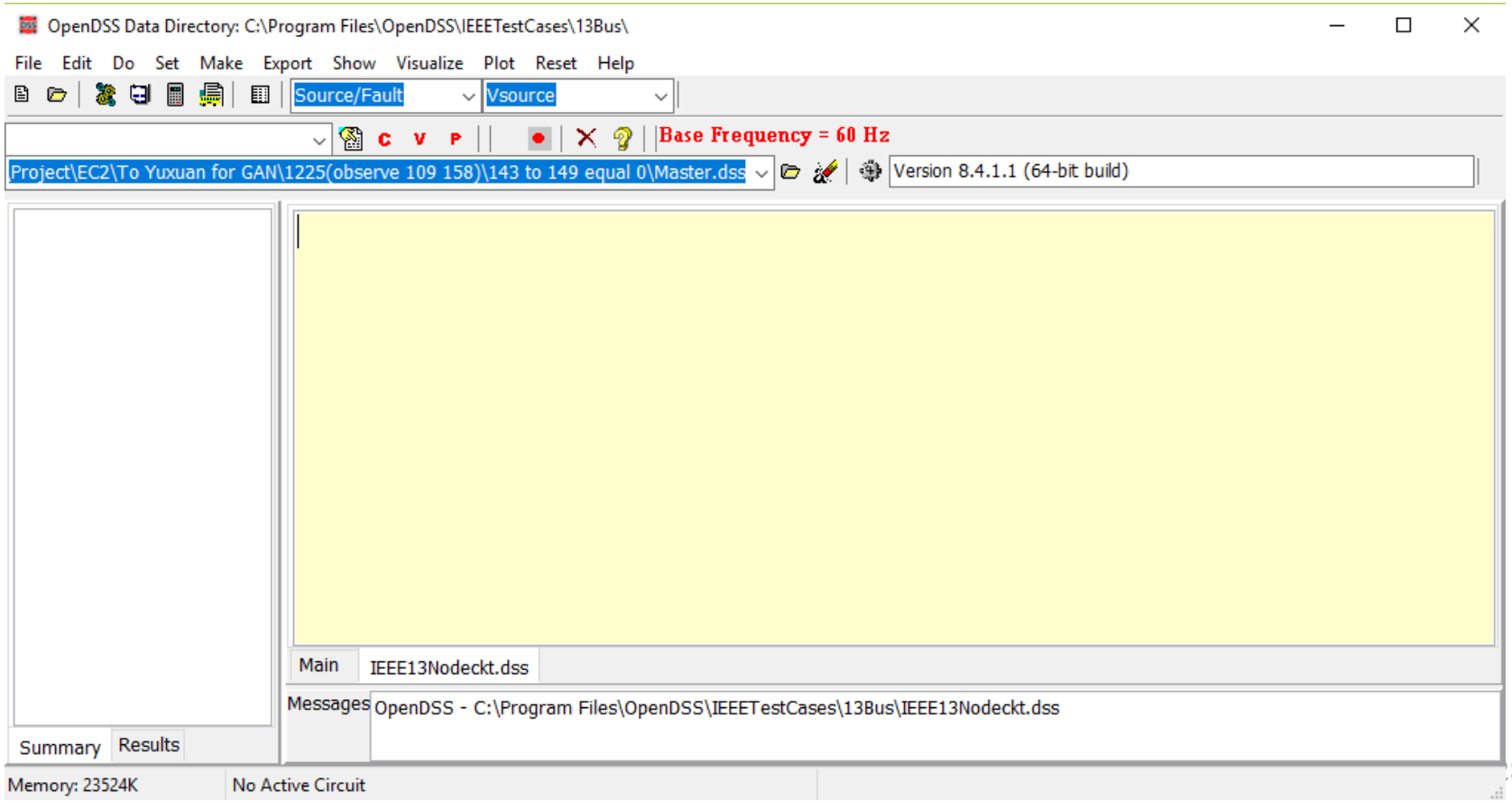
Installation

- Click “Finish” and check if the software has been successfully installed.



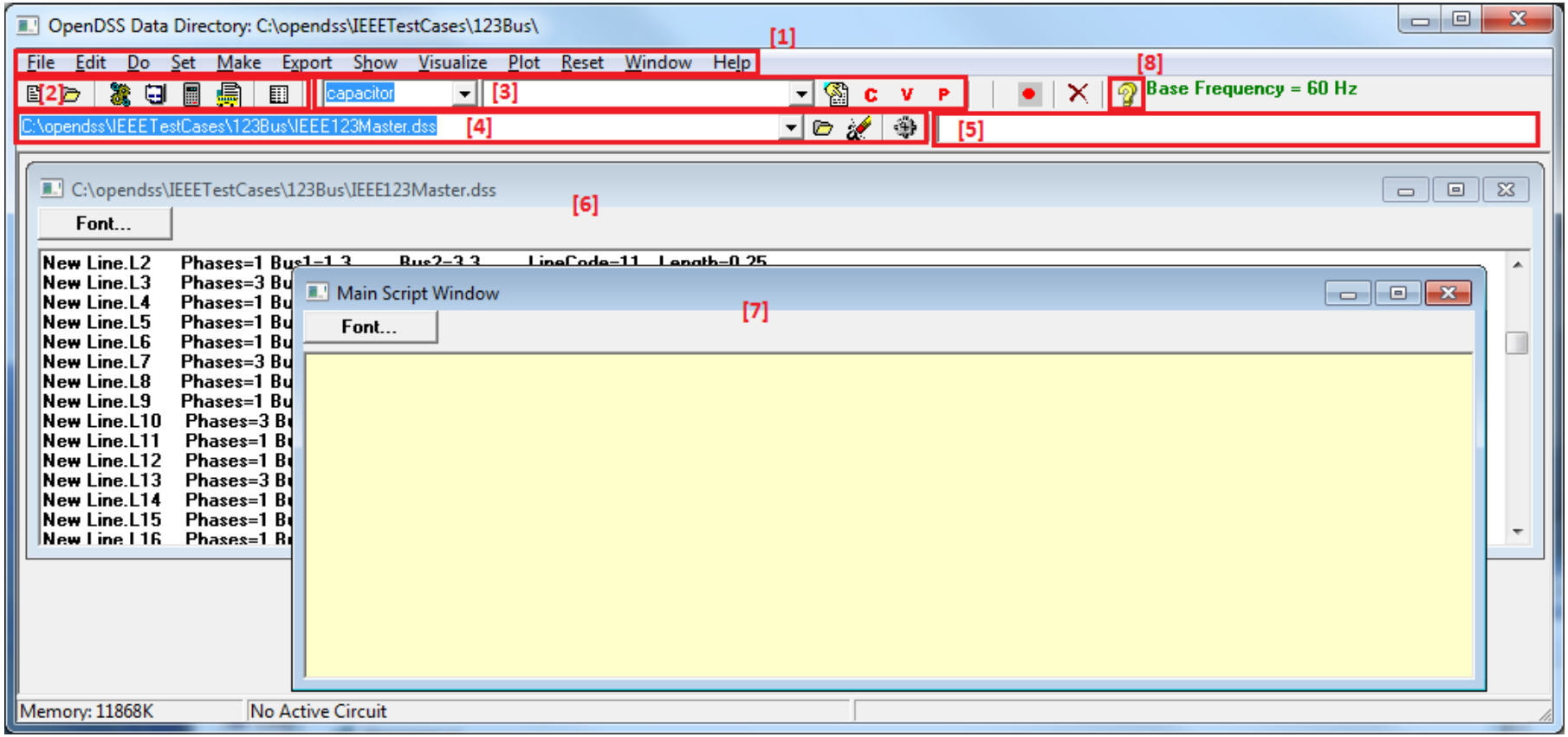
Installation

- If successful, an interface like below will appear.



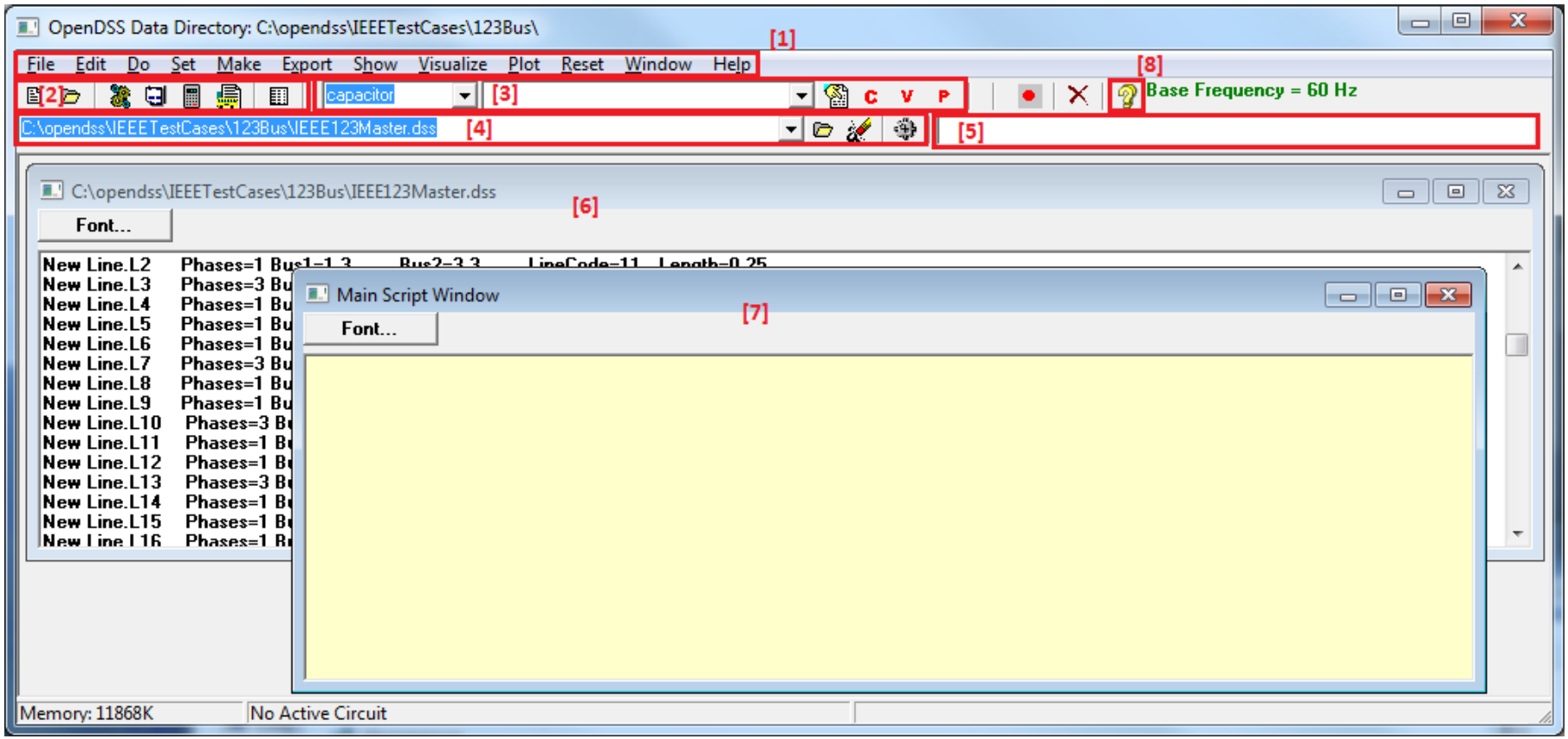
User Interface

- When opening OpenDSS, the user is greeted with the following window:



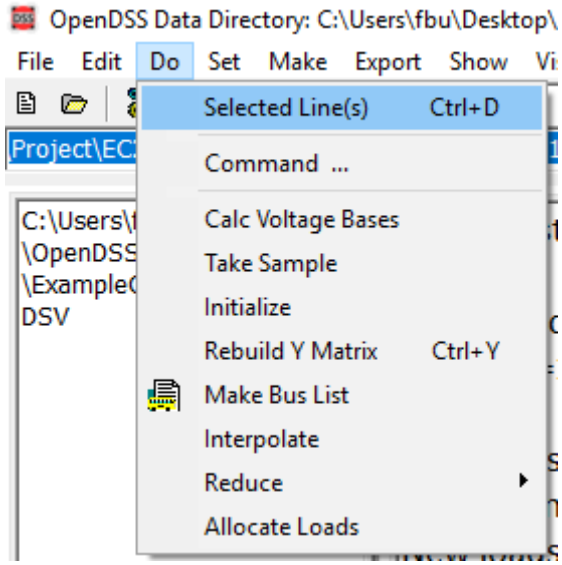
User Interface

- [1] Menu Structure: which drives most of the workflow in OpenDSS.



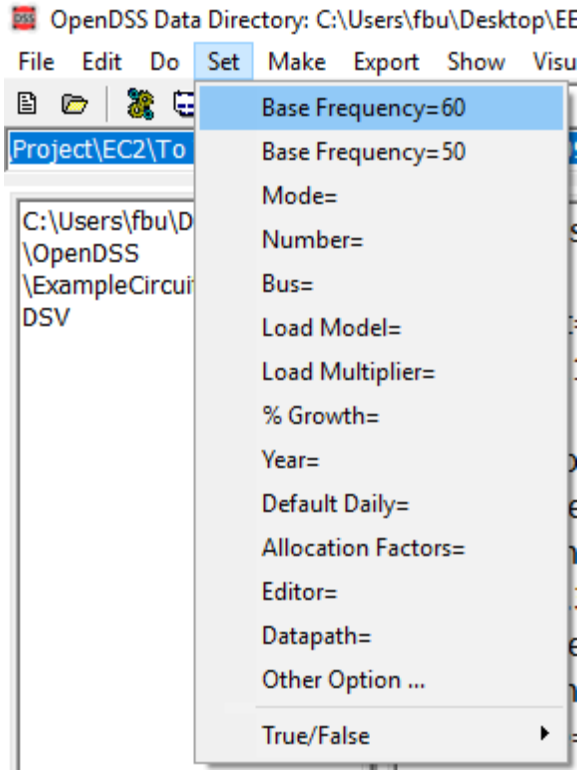
User Interface

- [1] Menu Structure:
  -- Do: the “Selected Line(s)” is commonly used, it is like “Run” .



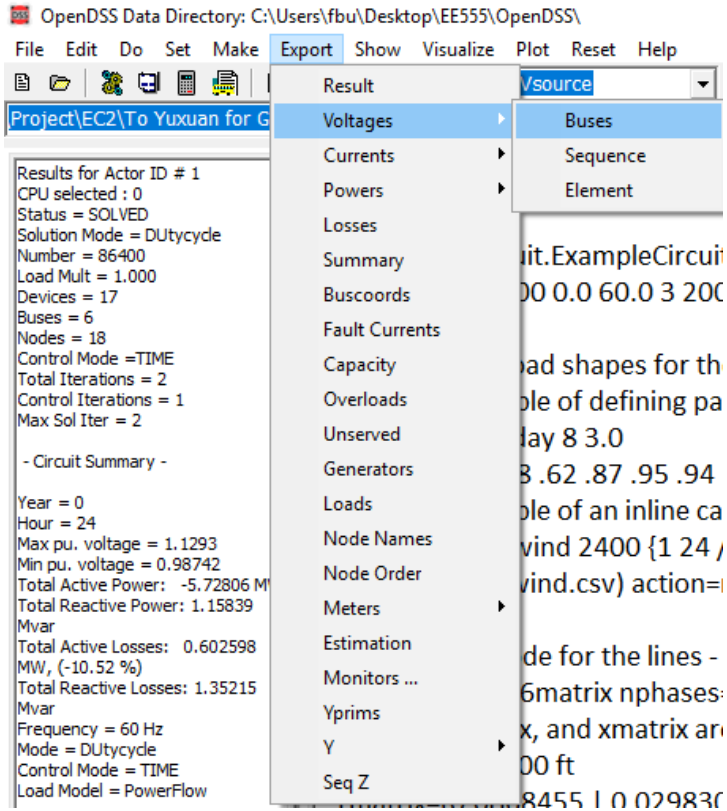
User Interface

- [1] Menu Structure:
  -- "Set": It allows us to set any solution parameter that can be set via the options scripting command.



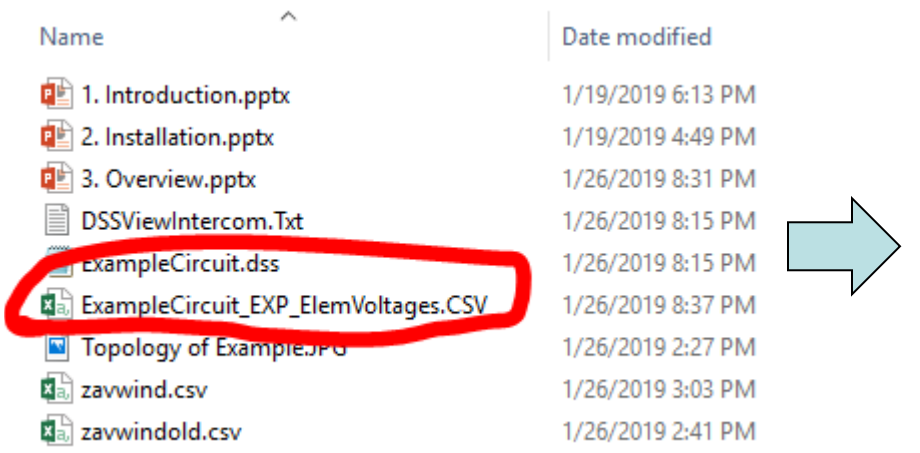
User Interface

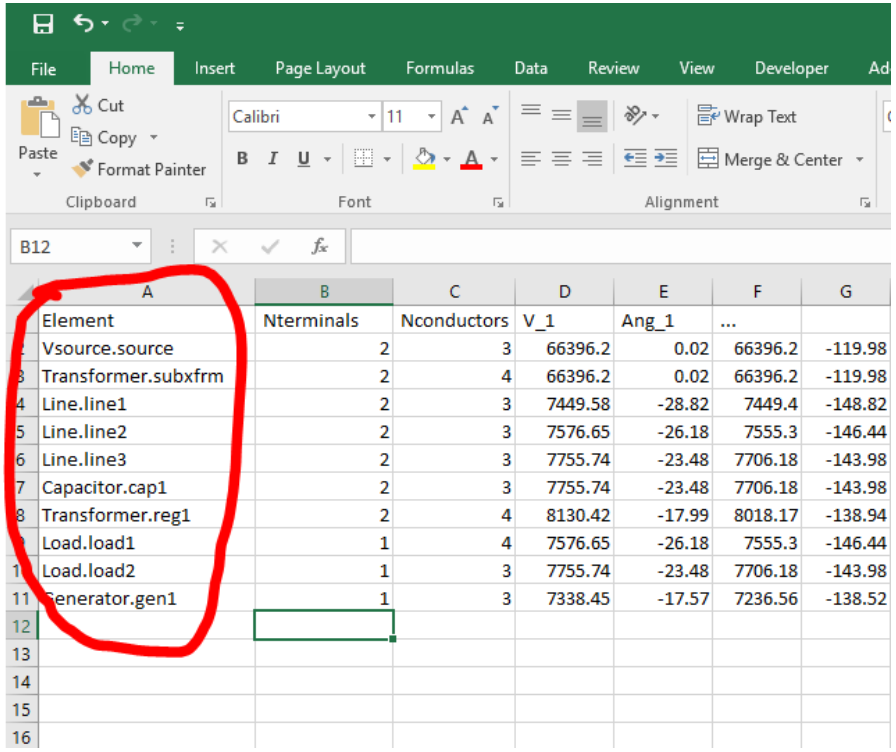
- [1] Menu Structure:
  -- “Export”: It allows us to save various reports to csv files



User Interface

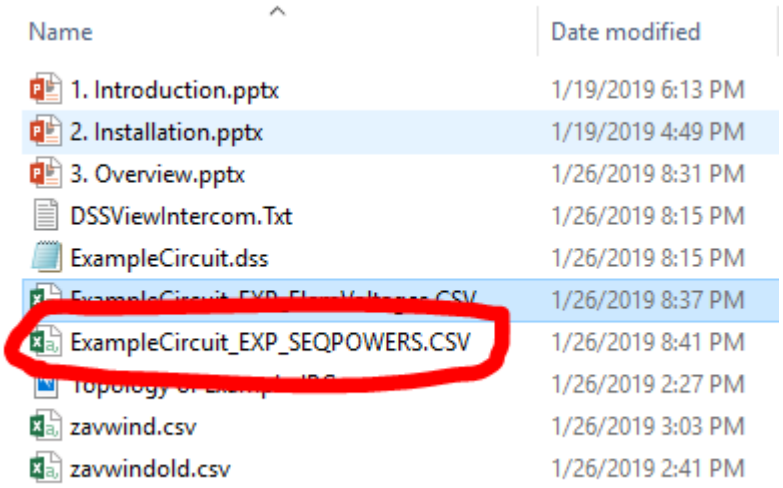
- [1] Menu Structure:
  -- “Export”: e.g., Export → Voltage → Element

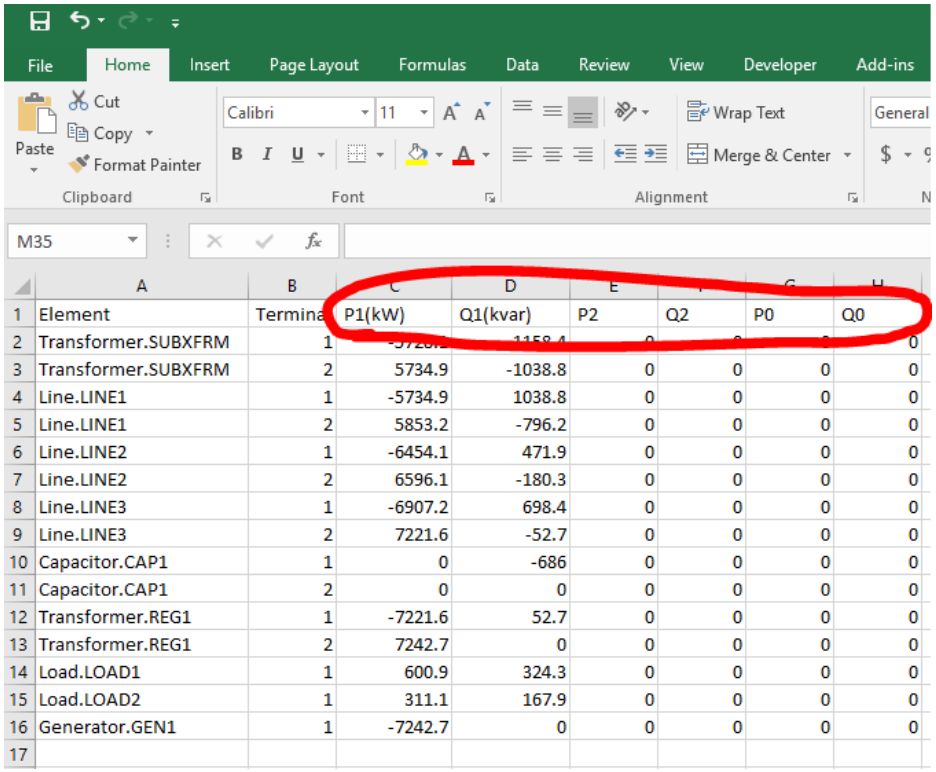




User Interface

- [1] Menu Structure:
  -- “Export”: e.g., Export → Powers → Sequence

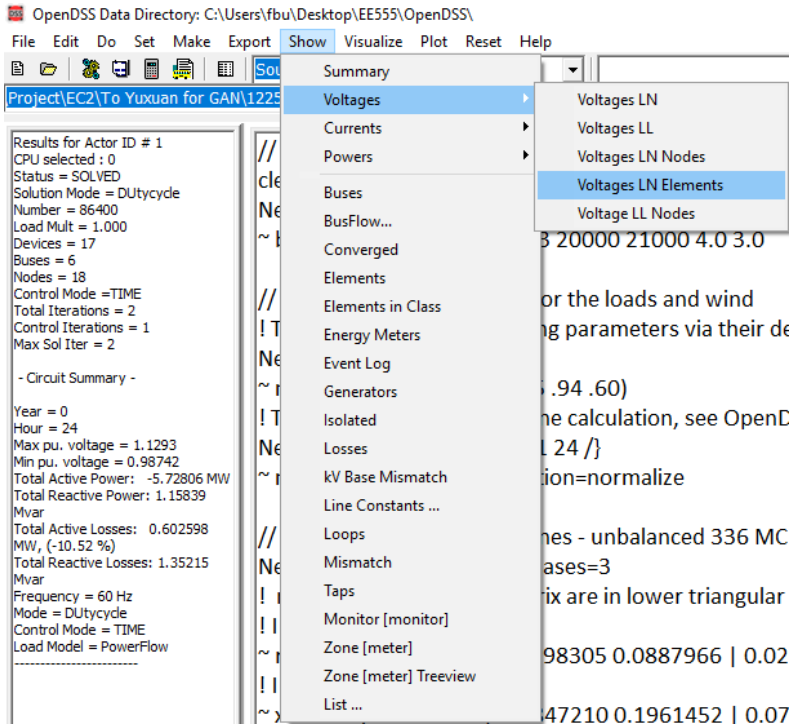




	A	B	C	D	E	F	G	H
1	Element	Termina	P1(kW)	Q1(kvar)	P2	Q2	P0	Q0
2	Transformer.SUBXFRM	1						0
3	Transformer.SUBXFRM	2	5734.9	-1038.8	0	0	0	0
4	Line.LINE1	1	-5734.9	1038.8	0	0	0	0
5	Line.LINE1	2	5853.2	-796.2	0	0	0	0
6	Line.LINE2	1	-6454.1	471.9	0	0	0	0
7	Line.LINE2	2	6596.1	-180.3	0	0	0	0
8	Line.LINE3	1	-6907.2	698.4	0	0	0	0
9	Line.LINE3	2	7221.6	-52.7	0	0	0	0
10	Capacitor.CAP1	1	0	-686	0	0	0	0
11	Capacitor.CAP1	2	0	0	0	0	0	0
12	Transformer.REG1	1	-7221.6	52.7	0	0	0	0
13	Transformer.REG1	2	7242.7	0	0	0	0	0
14	Load.LOAD1	1	600.9	324.3	0	0	0	0
15	Load.LOAD2	1	311.1	167.9	0	0	0	0
16	Generator.GEN1	1	-7242.7	0	0	0	0	0
17								

User Interface

- [1] Menu Structure:
  -- "Show": It contains much of the same information as the reports as the Export menu, but displays them directly in the GUI.



User Interface

- [1] Menu Structure:
  -- "Show": e.g., Currents → Currents Elm

ExampleCircuit_Curr_Elem.Txt - Notepad

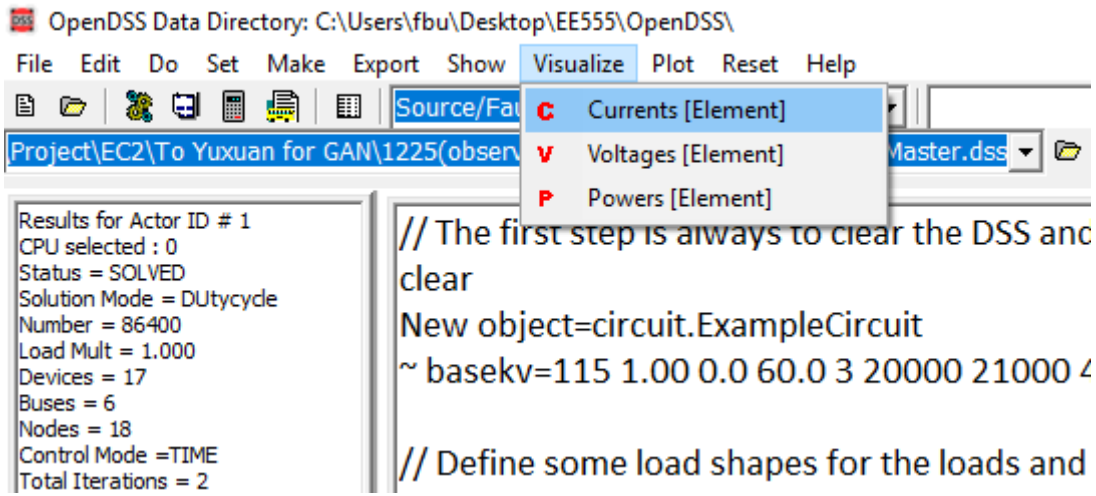
File Edit Format View Help

```
CIRCUIT ELEMENT CURRENTS

(Currents into element from indicated bus)

Power Delivery Elements

  Bus     Phase    Magnitude, A     Angle      (Real)   +j  (Imag)

ELEMENT = "Vsource.SOURCE"
SOURCEBUS     1           29.328 /_   11.4 =     28.745 +j    5.8189
SOURCEBUS     2           29.342 /_ -108.5 =    -9.3247 +j   -27.821
SOURCEBUS     3           29.347 /_  131.4 =    -19.421 +j    22.002
------------
SOURCEBUS     0           29.328 /_ -168.6 =    -28.745 +j   -5.8189
SOURCEBUS     0           29.342 /_   71.5 =     9.3247 +j    27.821
SOURCEBUS     0           29.347 /_  -48.6 =     19.421 +j   -22.002

ELEMENT = "Transformer.SUBXFRM"
SOURCEBUS     1           29.328 /_ -168.6 =    -28.745 +j   -5.8189
SOURCEBUS     2           29.342 /_   71.5 =     9.3247 +j    27.821
SOURCEBUS     3           29.347 /_  -48.6 =     19.421 +j   -22.002
SOURCEBUS     0                0 /_    0.0 =          0 +j         0
------------
SUBBUS        1           260.72 /_  -18.6 =     247.11 +j   -83.144
SUBBUS        2           260.84 /_ -138.5 =    -195.45 +j   -172.73
SUBBUS        3           260.81 /_  101.5 =    -51.889 +j    255.59
SUBBUS        0          0.36439 /_   50.1 =    0.23382 +j   0.27948

ELEMENT = "Line.LINE1"
SUBBUS        1           260.72 /_  161.4 =    -247.11 +j    83.144
SUBBUS        2           260.84 /_   41.5 =     195.45 +j    172.73
SUBBUS        3           260.81 /_  -78.5 =     51.889 +j   -255.59
------------
LOADBUS1      1           260.73 /_  -18.6 =     247.14 +j   -83.058
LOADBUS1      2           260.86 /_ -138.5 =     -195.4 +j   -172.82
LOADBUS1      3           260.83 /_  101.5 =    -51.982 +j     255.6

ELEMENT = "Line.LINE2"
LOADBUS1      1           285.66 /_  157.9 =    -264.62 +j     107.6
LOADBUS1      2           285.66 /_   37.9 =     225.35 +j    175.55
LOADBUS1      3           285.83 /_  -82.1 =     39.413 +j    -283.1
------------
LOADBUS2      1           285.66 /_  -22.1 =     264.65 +j   -107.51
LOADBUS2      2           285.66 /_ -142.1 =    -225.29 +j   -175.63
LOADBUS2      3           285.84 /_   97.9 =    -39.507 +j     283.1
```

User Interface

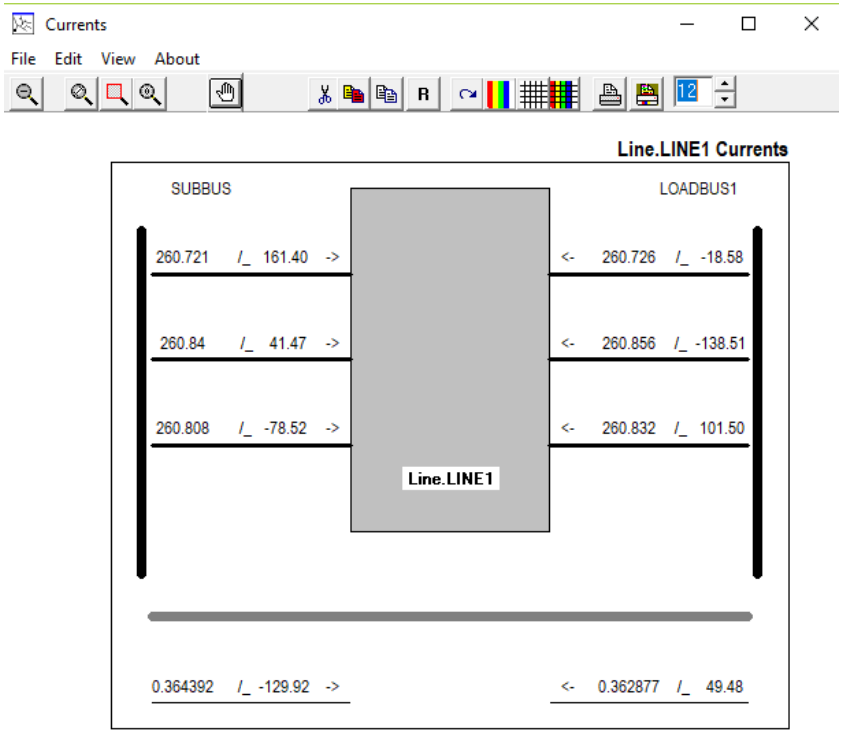
- [1] Menu Structure:

  -- “Visualize”: It provides a graphical output of the device selected via the element selector.

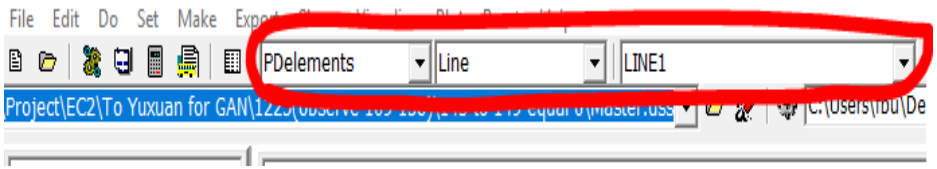


User Interface

- [1] Menu Structure:
  -- “Visualize”: e.g., Visualize → Currents[Element].



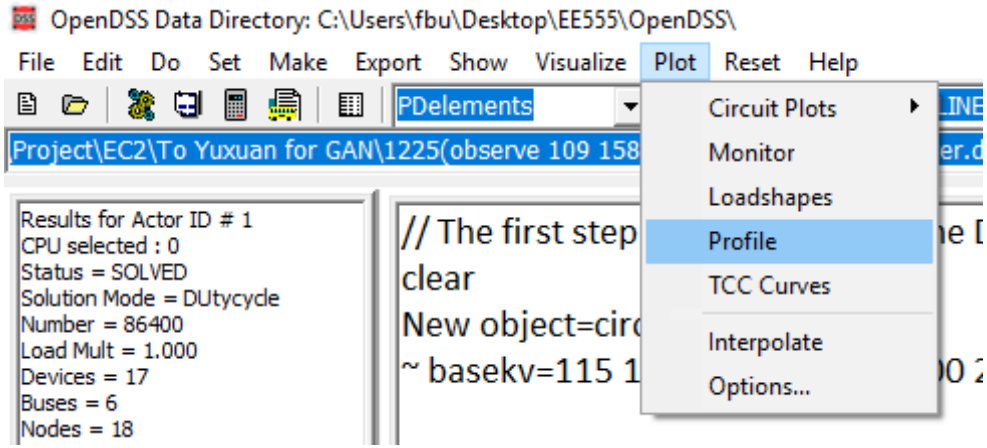
Note: Specific element should be firstly selected, as follows:



User Interface

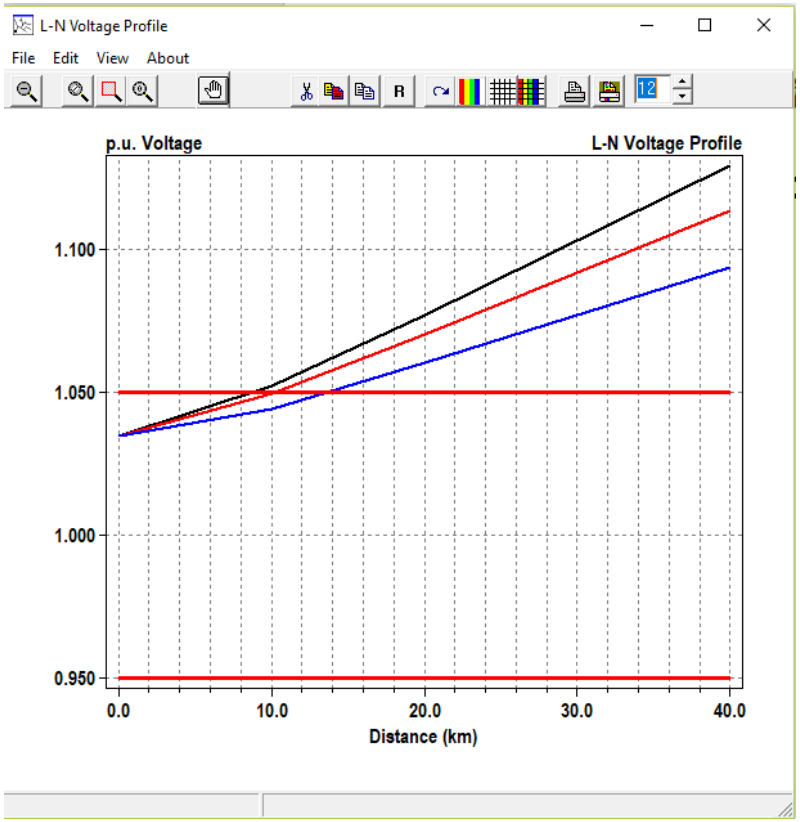
- [1] Menu Structure:

  -- “Plot”: It provides graphical output relevant to the whole system.



User Interface

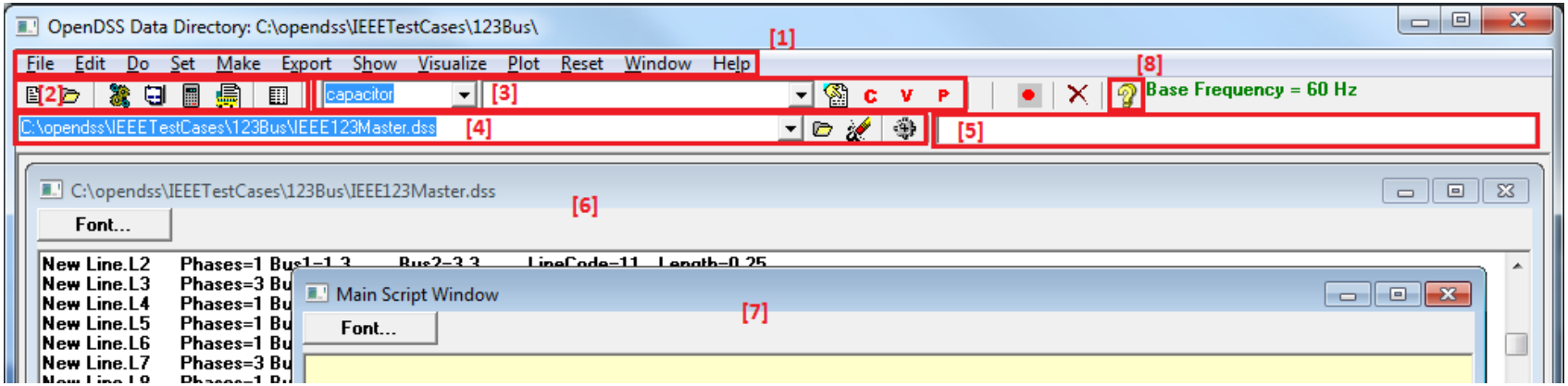
- [1] Menu Structure:
  -- “Plot”: e.g., Plot → Profile.

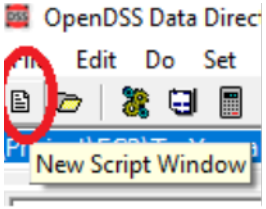


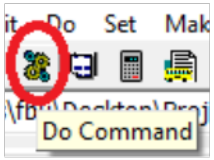
Note, Energymeter need to be defined in the script:
e.g., New Energymeter.em1 line.line1

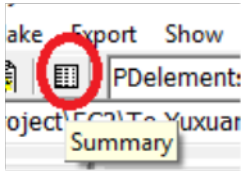
User Interface

- [2] Toolbar: provides direct access to many commonly used OpenDSS commands such as “Solve,” “Summary,” and “Do Command.”



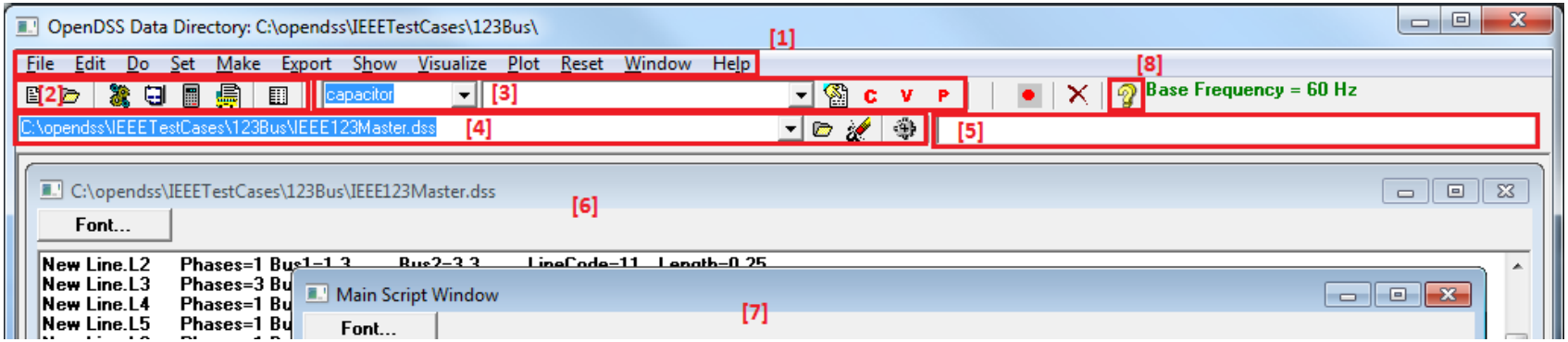






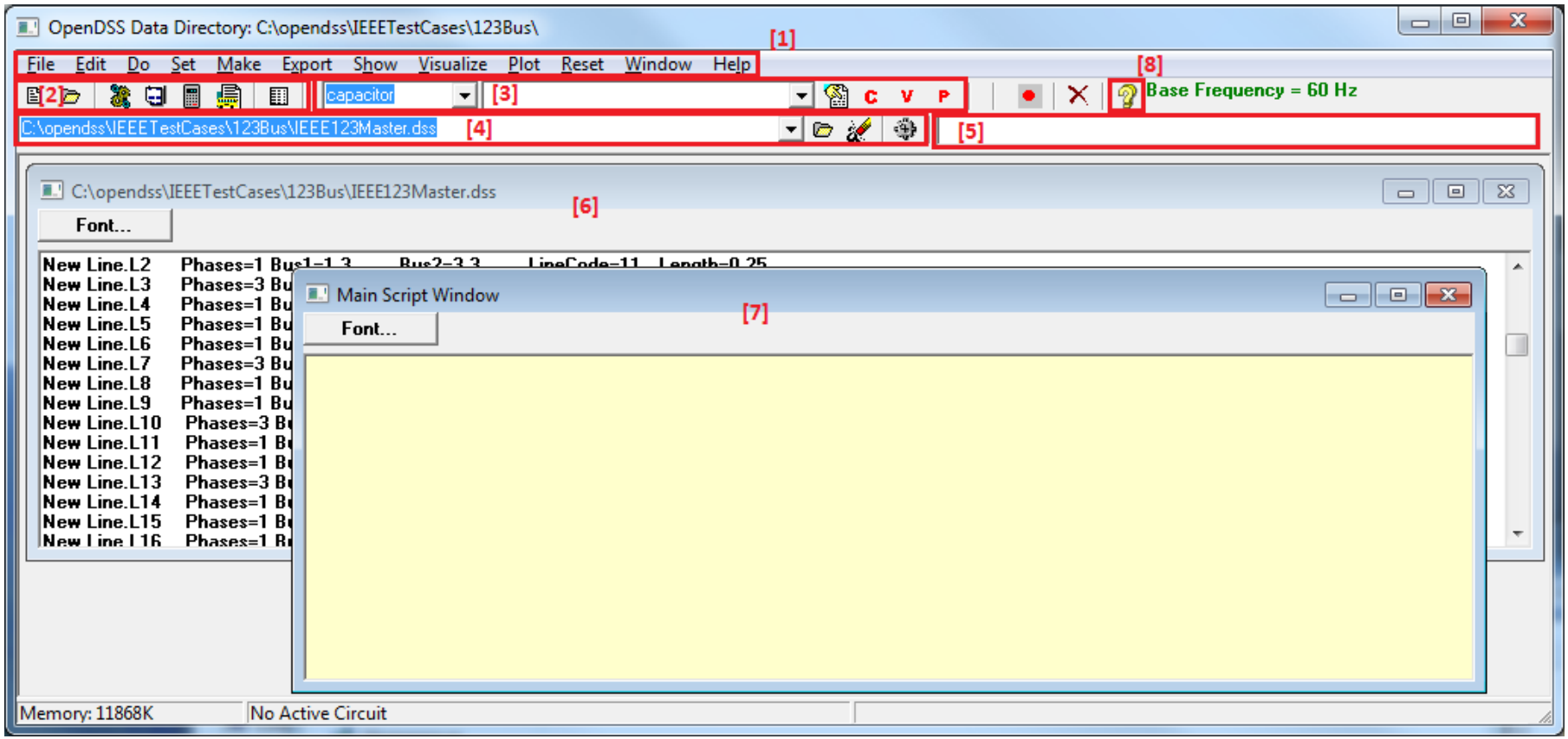
User Interface

- [3] Element tools: allows the user to select what circuit element (by type) to edit or display visualizations for.



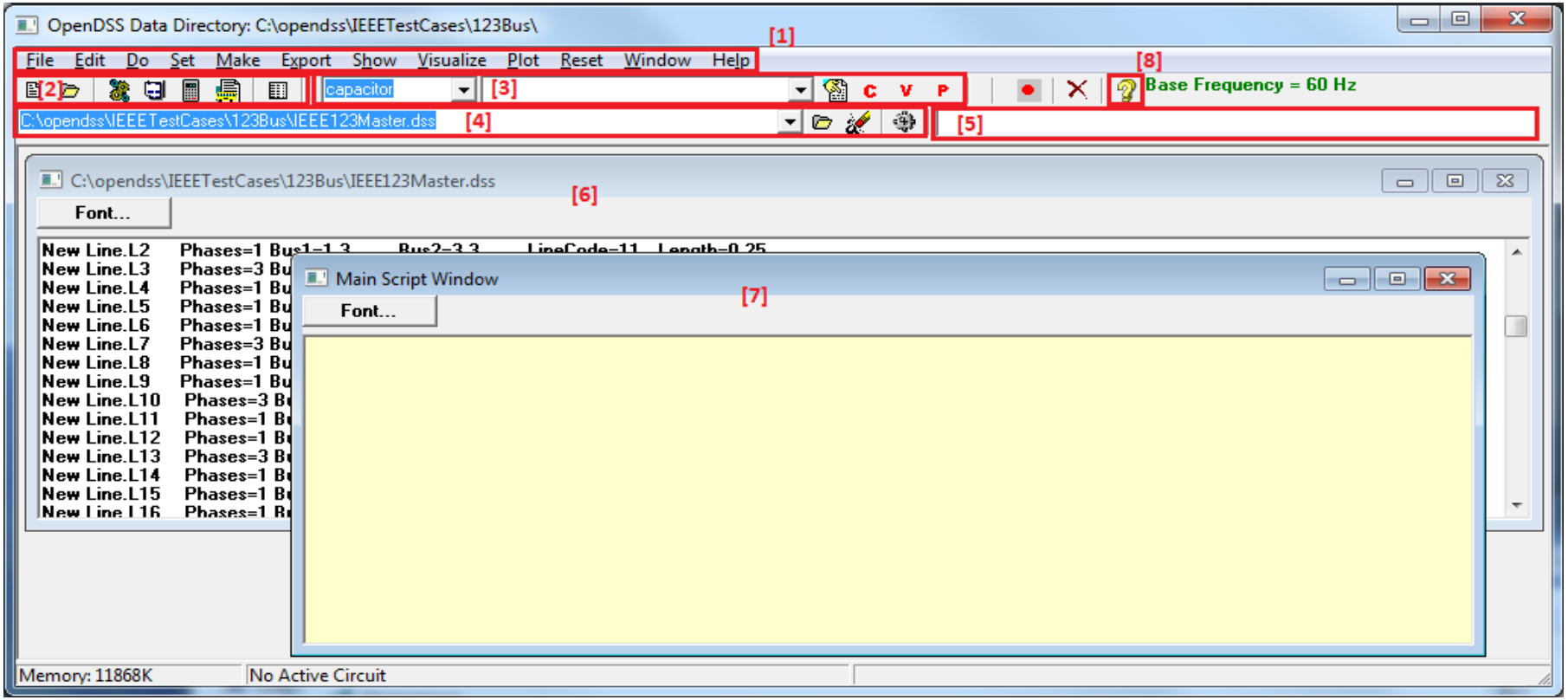
User Interface

- [4] Script tools: which allows one to select which of the current opened scripts to run.



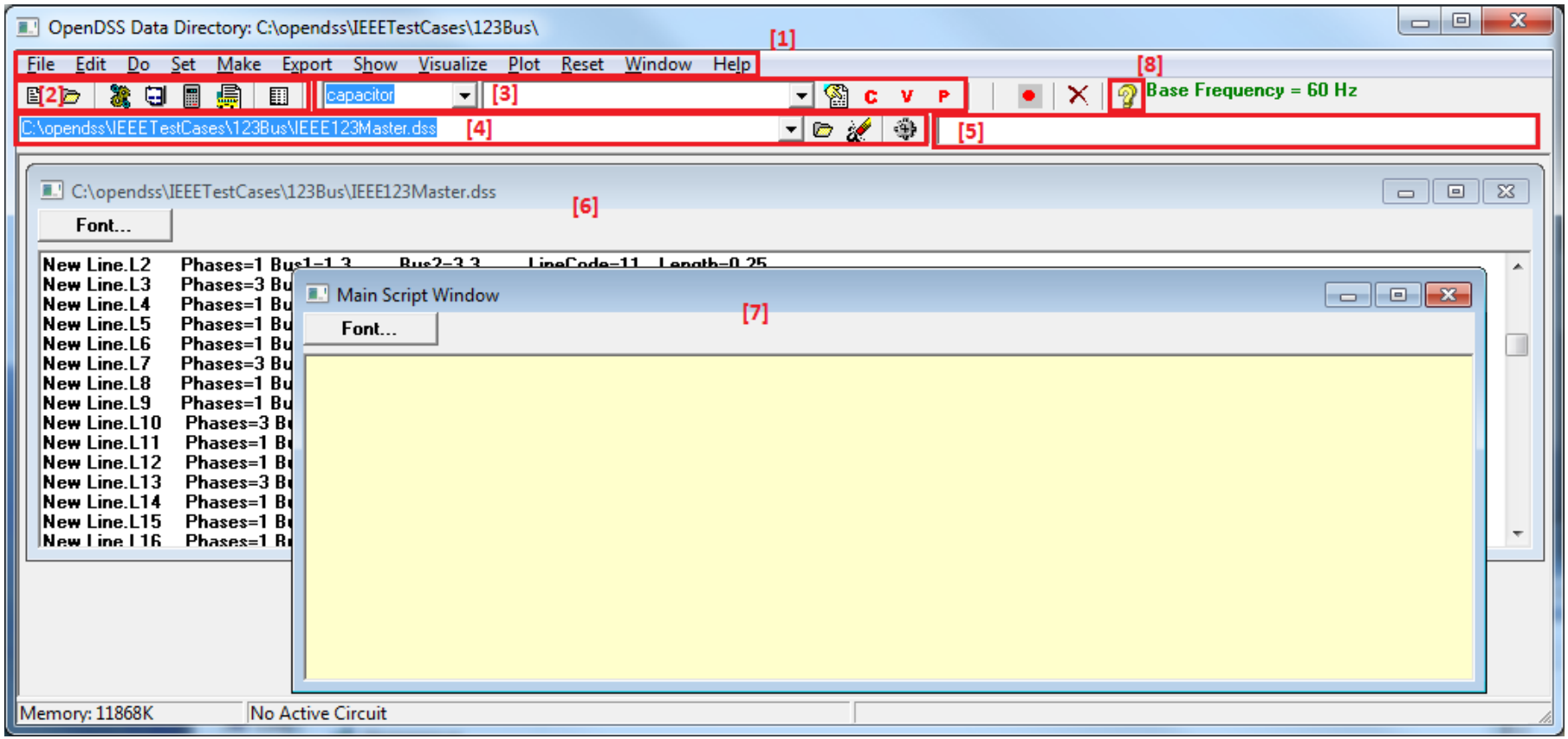
User Interface

- [5] Results bar: which provides a condensed version of the Results window which can be accessed through Show → Result Form.



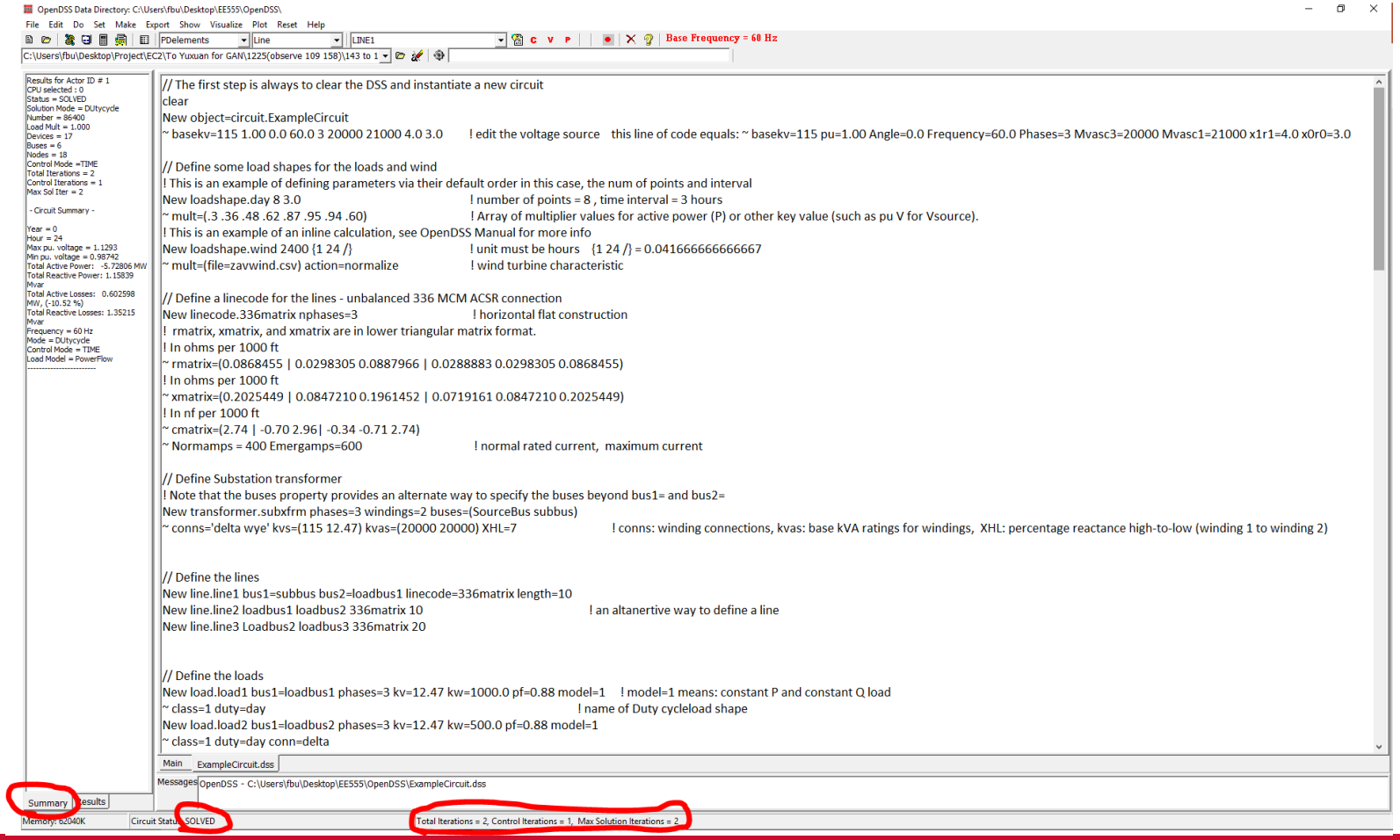
User Interface

- [6] Script Windows: to directly edit various *.dss files.



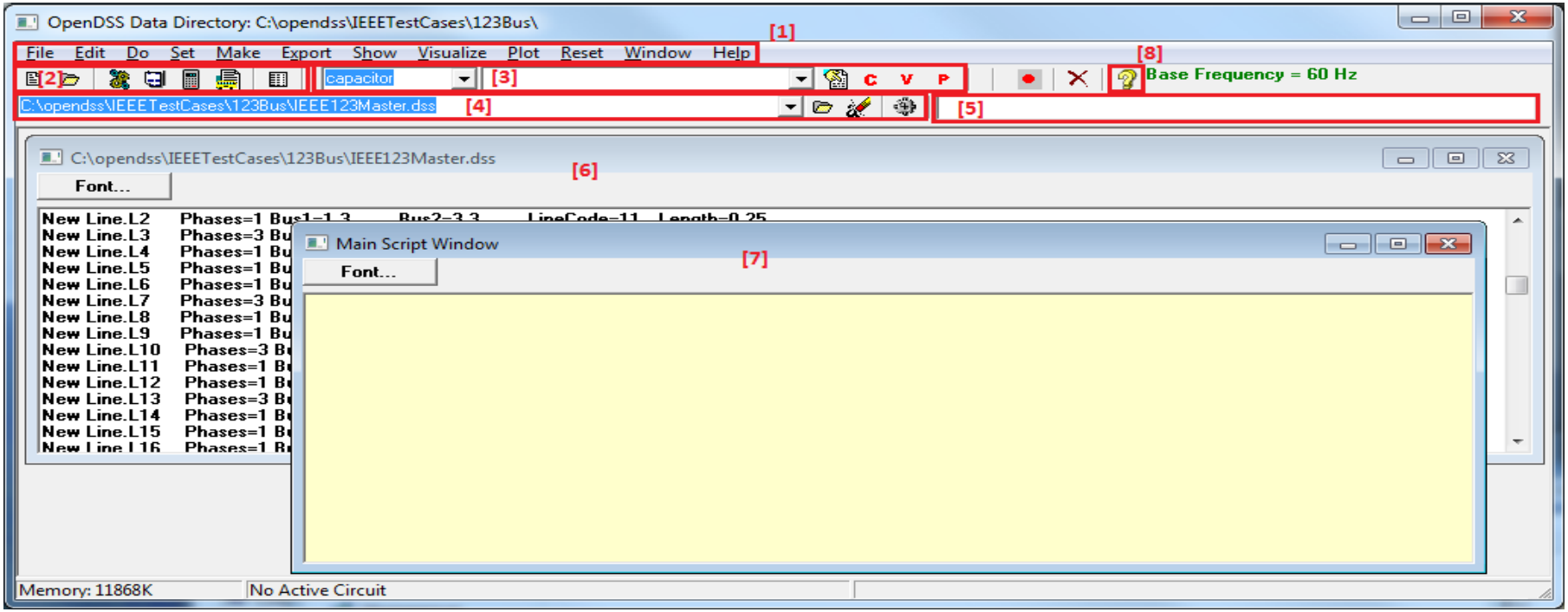
User Interface

- [6] Script Windows: to directly edit various *.dss files.



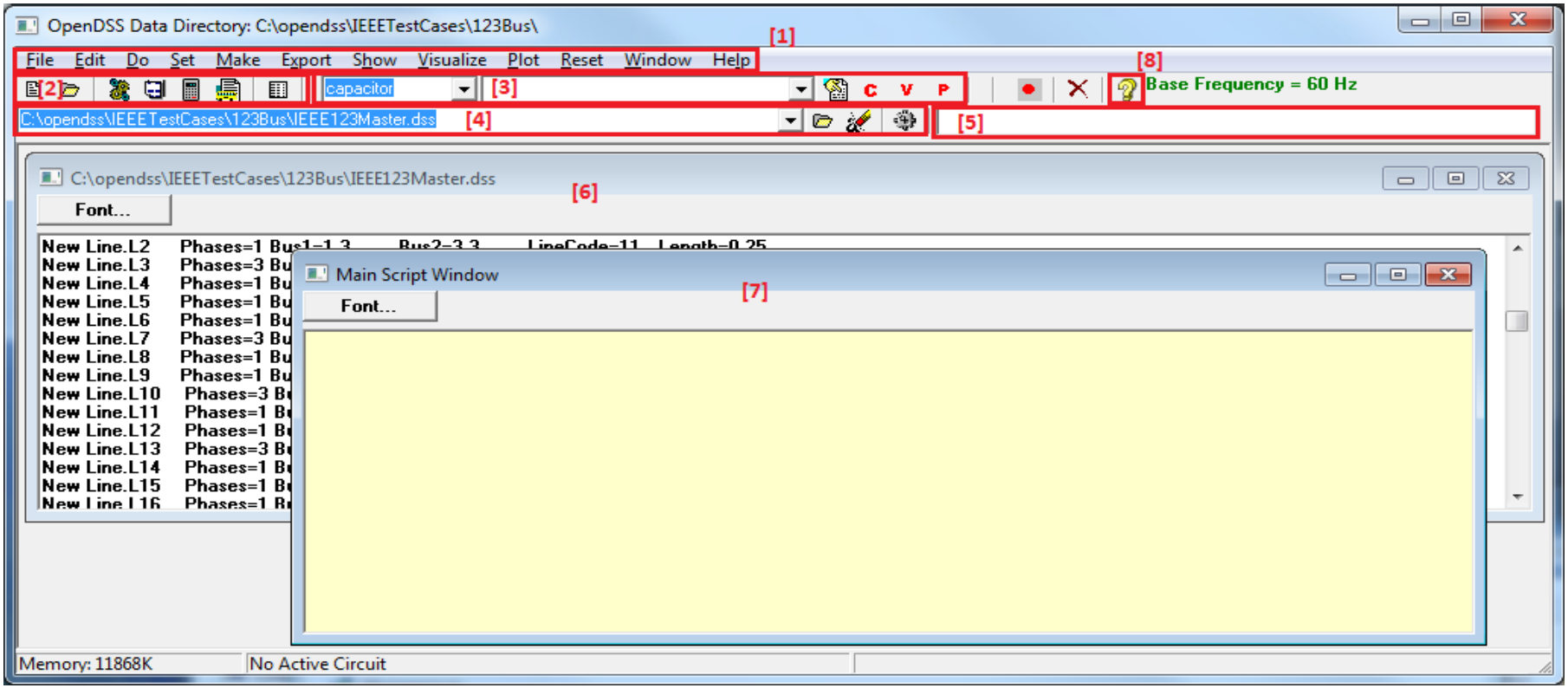
User Interface

- [7] Main Script Window: Main Script Window is a sort of "notepad" or "interactive window" for OpenDSS. The user can type small commands and run them via the "Do Command" feature (Ctrl-D).



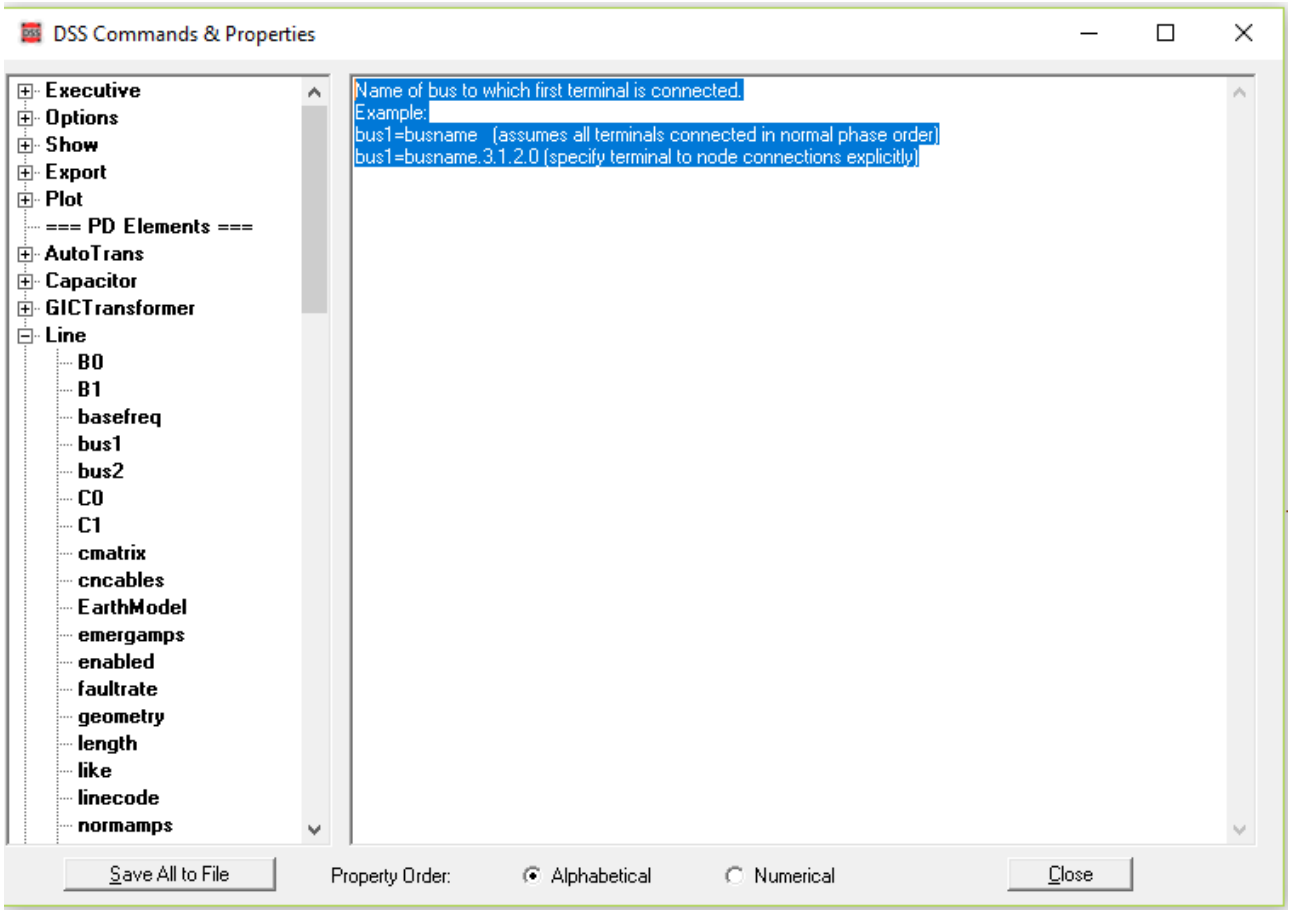
User Interface

- [8] Help button: brings up the OpenDSS Command and Element Properties Reference which gives a tree-view guide to the various script commands in OpenDSS.



User Interface

- [8] Help button: Very useful.



Contents

- General introduction of OpenDSS
- Installation & User interface
- Workflow
- Overall concept
- Overall structure of solving power flow

Workflow

In general, the user has 4 steps to develop a project:

- Define the circuit;
- Set up the circuit options;
- Solve;
- Perform analysis.

1. Define circuit

Create new lines, transformers, loads, generators, etc…

- the best way to do this is by creating a dss script;
- Once a script has been written, use “Compile file listed” button to run the selected script.

2. Set up circuit options

Such as the solution mode (snapshot, daily, harmonic, etc…)

- This is accomplished through the commands in the Set menu. The most basic form of analysis is the “snapshot” which is analogous to a traditional power flow.

3. Solve

Solve the power flow problem.

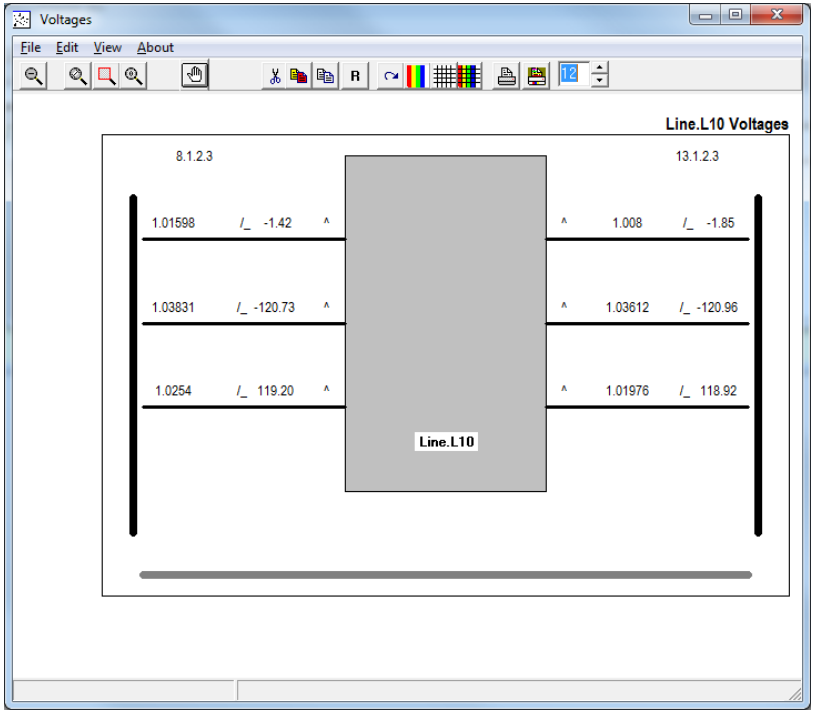
- Ensure that the bus list is created and the base voltages found by running Do → Calc Voltage Bases;
- Then use the "Solve" button in the toolbar to solve the circuit.

4. Perform analysis

The specifics of how to accomplish this vary from analysis to analysis, but some general tasks include:

- *Looking at a branch, transformer, load, or other element in the system* – To do this, first select the element type and then the element from the element toolbar. Then select the C, V, or P buttons to see a current, voltage, or power visualization. Select the "Form edit on active element" button to edit the element.

  …



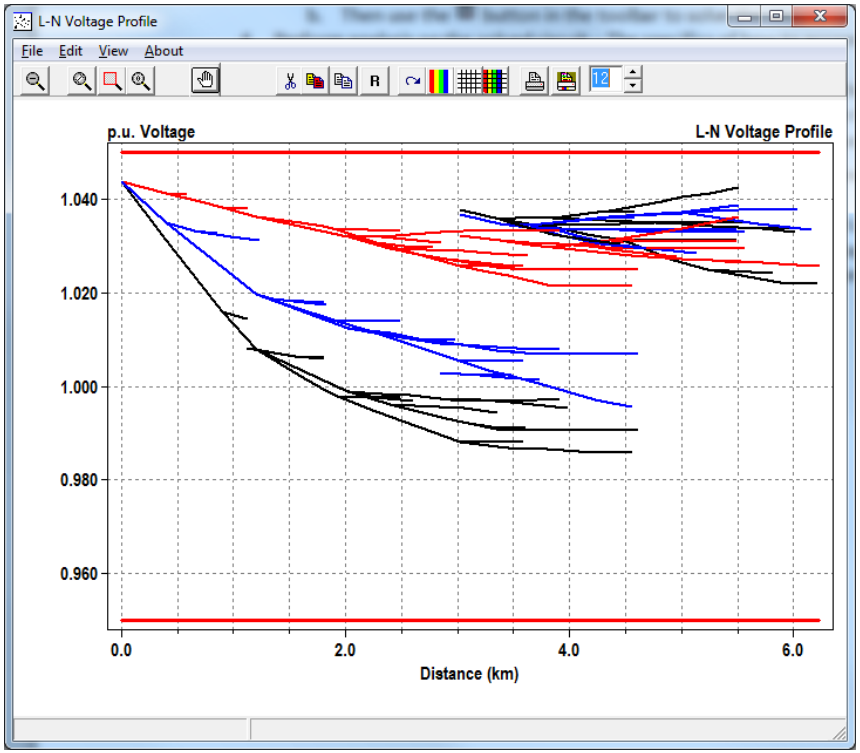
4. Perform analysis

The specifics of how to accomplish this vary from analysis to analysis, but some general tasks include:

…

- *Visualize the voltage profile of the system* – Type "plot profile" into the main script window, and press Ctrl-D to Do that command. A voltage profile will come up showing how the voltage progresses as one progresses down the feeder.

…



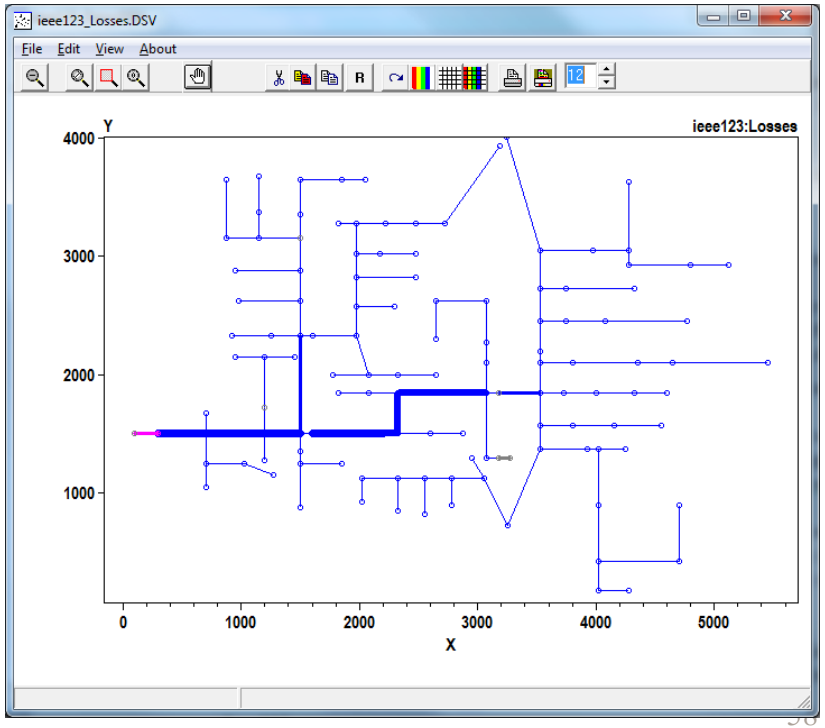
4. Perform analysis

The specifics of how to accomplish this vary from analysis to analysis, but some general tasks include:

…

- *Visualize data onto a one-line of the feeder* – If you provide bus location data via the "buscoords" command, you can superimpose power flow data onto a map/one-line of the system. To do this, go Plot > Circuit Plots > Circuit Plot.

…



4. Perform analysis

The specifics of how to accomplish this vary from analysis to analysis, but some general tasks include:

…

- *Export data for analysis in 3rd party program* – All results obtained through OpenDSS can be exported through the various commands in the "Export" menu. Results are .csv files.

Contents

- General introduction of OpenDSS
- Installation & User interface
- Workflow
- Overall concept
- Overall structure of solving power flow

Overall concept

- Overall circuit model;
- Bus and terminal models;
- Power delivery elements;
- Power conversion elements;
- Putting it all together.

Overall circuit model

The OpenDSS consists of a model of the electrical power distribution system in the rms steady state, overlaid with a communications network that interconnects controls on power delivery elements and on power conversion elements.

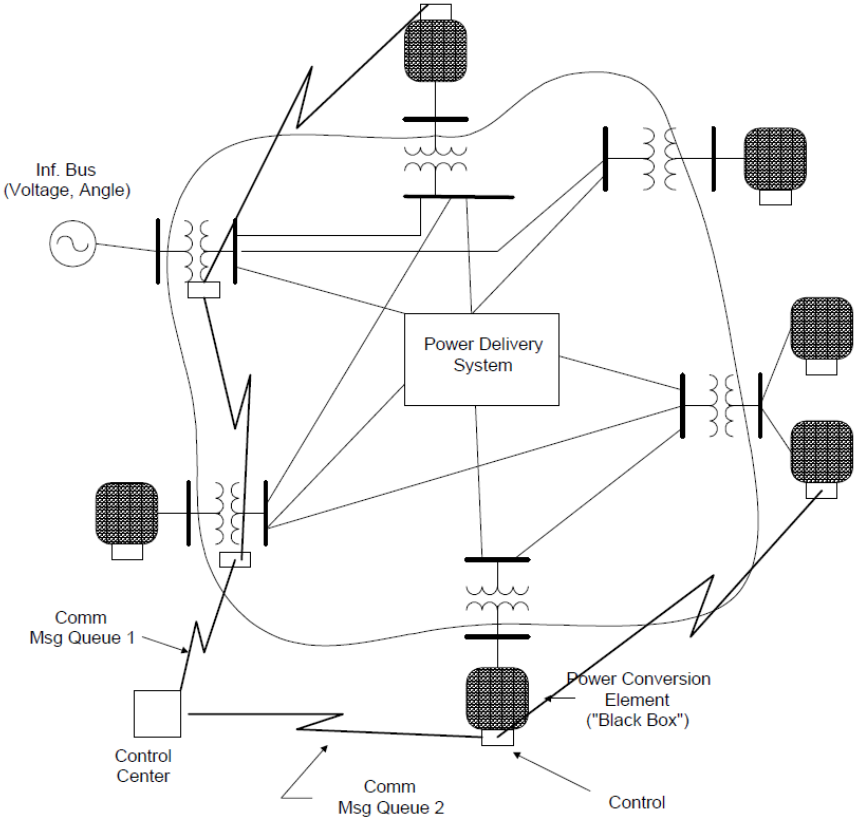


Figure 11: Electrical Circuit with Communications Network

Bus and terminal models
-- Bus definition

- **A bus is a circuit element having [1..N] nodes.** Buses are the connection point for all other circuit elements. In many other power system analysis programs, “bus” and “node” are nearly synonymous, but they are distinctively different in OpenDSS. Bus is the container of Node objects. That is to say, a Bus has Nodes.

Figure 12: Bus Definition

Bus and terminal models

-- Bus definition

- **The main electrical property of a Bus is voltage.** Each node has a voltage with respect to the zero voltage reference (remote ground). There is a nodal admittance equation written for every node (i.e., the current is summed at each node). That basically dictates the size of the problem that must be solved, although there is also computational overhead computing the injection, or compensation, currents.

Figure 12: Bus Definition

Bus and terminal models

-- Bus naming

- **Buses are named by an alphanumeric string of characters.** The names may be numbers, but are always treated as strings. Internally, buses will be numbered (actually, each node is numbered), but are referenced through the COM or command interface by name. The internal index number is subject to change if something in the circuit changes and is not a reliable way to refer to a bus or node.
- **It is better if names do not contain blanks, tabs, or other "white space" or control characters.** If the name contains a blank, for example, it must be enclosed in quotes (single or double, parentheses, or brackets). This can be a source of errors because of the different entry points for the names (user interface, files, COM interface). However, names from models in other programs such as PSS/E may contain names (may also be duplicates).

Bus and terminal models
-- Bus naming

- **Names may be any length and are passed to the OpenDSS through the COM interface as standard null-terminated strings (actually, widestrings).** This is the common way for representing strings in the Windows environment. Thus, it is a simple matter to drive the OpenDSS from most programming languages.

Bus and terminal models

-- Terminal definition

- **Each electrical element in the power system has one or more terminals. Each terminal has one or more conductors.** Each conductor conceptually contains a disconnect switch and a TCC (fuse) curve [The Fuse simulation in the terminal conductors has been disabled in this version and is being redesigned; a Relay object or a separate Fuse object can be used, if needed, to control the switches in the terminal conductors.]. The conductors are numbered [1, 2, 3,…].

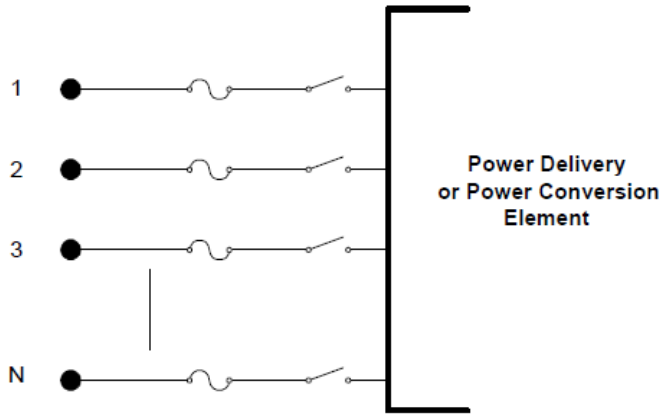


Figure 13: Terminal Definition

Bus and terminal models

-- Terminal definition

- **If the terminal is connected to an N-phase device, the first N conductors are assumed to correspond to the phases, in order.** The remaining conductors may be virtually any other conductor but are frequently neutrals or other non-power conductors.
- The OpenDSS bus is a connecting place with 1 or more nodes for connecting individual phases and other conductors, from the terminals of both power delivery elements and power conversion elements.

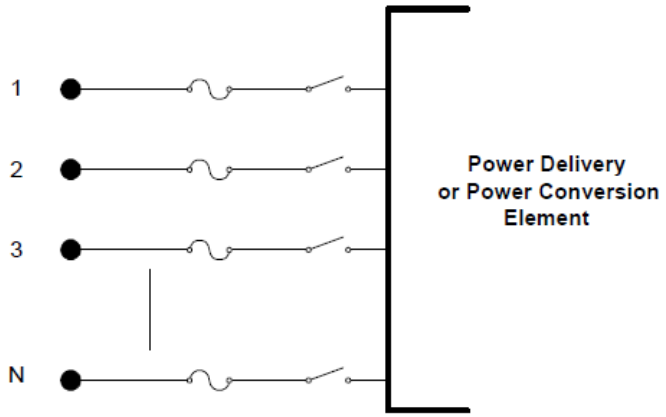


Figure 13: Terminal Definition

Bus and terminal models
-- Terminal definition

- **Buses are named with null-terminated strings of arbitrary length.** Therefore, you may use quite long names.
- **Node 0 of each bus is implicitly connected to the voltage reference** (i.e., the node's voltage is always zero and is never explicitly included in the Y matrix).

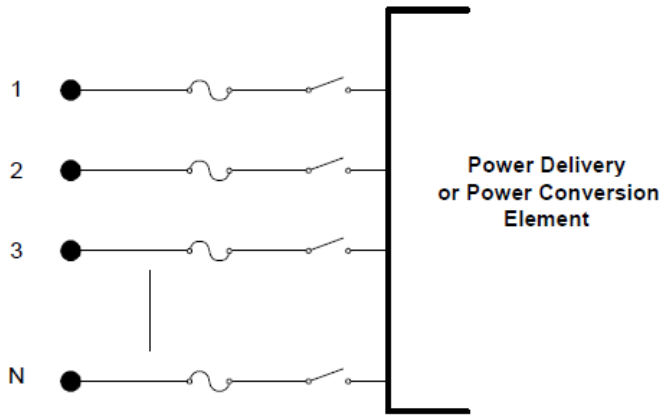


Figure 13: Terminal Definition

Bus and terminal models
-- Terminal references

- **Terminals are not named separately from the device.** Each device will have a name and a defined number of multiphase terminals. Terminals will be referenced explicitly by number [1, 2, 3 …] or by inference, in the sequence in which they appear. Internally, they will be sequenced by position in a list.

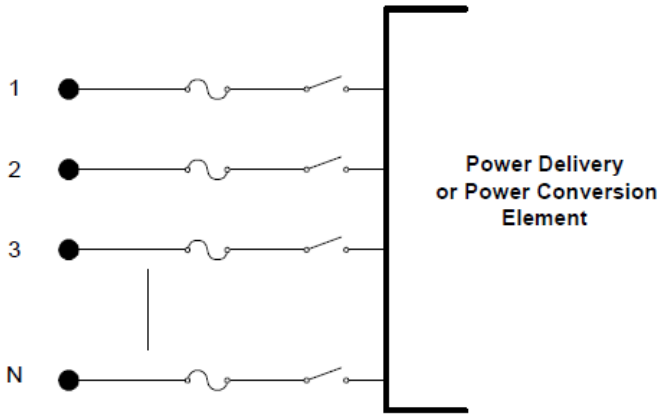


Figure 13: Terminal Definition

Bus and terminal models

-- Phases and other conductors

- **Each terminal has one or more phases, or normal power-carrying conductors and, optionally, a number of other conductors to represent neutral, or grounding, conductors or conductors for any other purpose.** By convention, if a device is declared as having N phases, the first N conductors of each terminal are assumed to be the phase conductors, in the same sequence on each terminal. Remaining conductors at each terminal refer to “neutral” or “ground” or "earth“ conductors. However, the conductors can represent other power carrying conductors. There is no hard

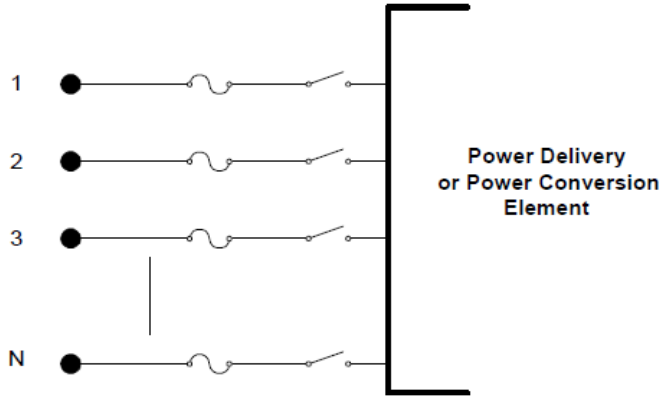
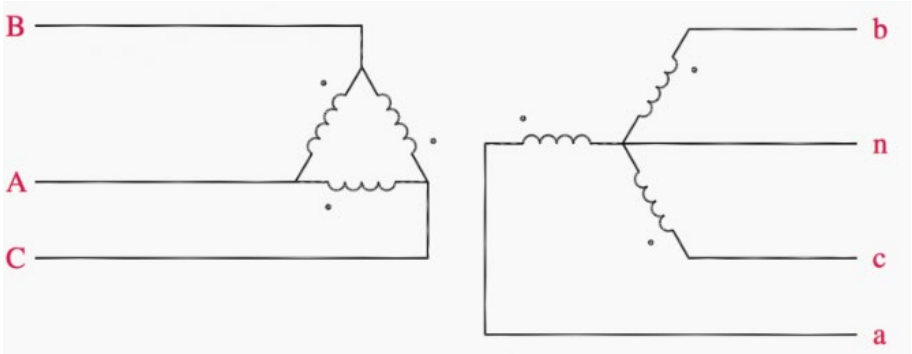


Figure 13: Terminal Definition

Bus and terminal models

-- Phases and other conductors

- **All terminals of a device are defined to have the same number of conductors.** For most devices, this causes no ambiguity, but for **transformers** with both delta and wye (star) winding connections, there will be an extra conductor at the delta-connected terminal. The neutral is explicit to allow connection of neutral impedances to the wye-connected winding. The extra conductor for the delta connection is simply connected to ground (voltage reference) and the admittances are all set to zero. Thus, the conductor effectively does not appear in the problem; it is ignored. However, you may see it appear in reports that explicitly list all the voltages and currents in a circuit.
- **The terminal conductors and bus nodes may be combined to form any practical connection.**

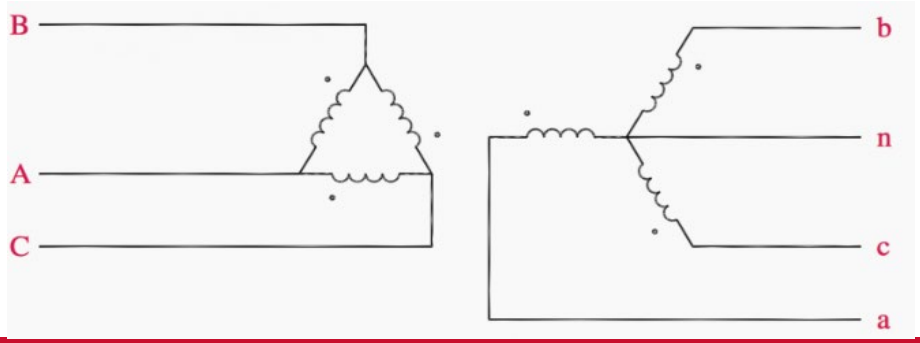


This picture is from https://electrical-engineering-portal.com/3-phase-transformer-connections

Bus and terminal models

-- Phases and other conductors

- **Buses have Nodes:** A bus may have any number of nodes (places to connect device terminal conductors). Nodes are integer numbers. The nodes may be arbitrarily numbered. However, the first N are by default reserved for the N phases of devices connected to them. Thus, if a bus has 3-phase devices connected to it, connections would be expected to nodes 1, 2, and 3. So the DSS would use these voltages to compute the sequence voltages, for example. Phase 1 would nominally represent the same phase throughout the circuit, although there is nothing to enforce that standard. It is up to the user to maintain a consistent definition. If only the default connections are used, the consistency is generally maintained automatically, although there can be exceptions.
- **Any other nodes would simply be points of connection with no special meaning.** Each Bus object keeps track of the allocation and designation of its nodes.



This picture is from https://electrical-engineering-portal.com/3-phase-transformer-connections

Bus and terminal models

-- Phases and other conductors

- **Node 0** of a bus is always the voltage reference (a.k.a, ground, or earth). That is, it always has a voltage of exactly zero volts.

Figure 12: Bus Definition

Bus and terminal models

-- Specifying connections

The user can define how terminals are to be connected to buses in **three** ways, not just one way:

- Generically connect a circuit element's terminal to a bus without specifying node-to-conductor connections. This is the default connection. Normal phase sequence is assumed. Phase 1 of the terminal is connected to Node 1 of the Bus, and so on. Neutrals default to ground (Node 0).

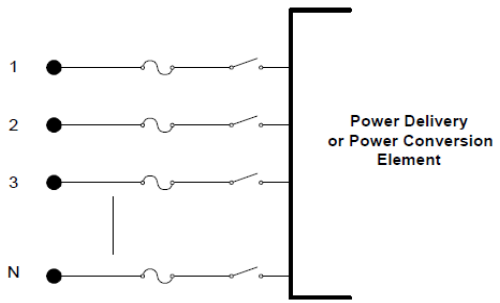


Figure 13: Terminal Definition

Figure 12: Bus Definition

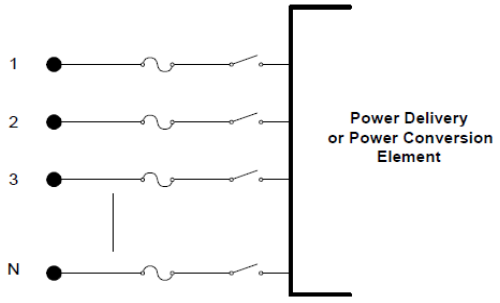


Figure 13: Terminal Definition

Bus and terminal models

-- Specifying connections

The user can define how terminals are to be connected to buses in **three** ways, not just one way:

- Generically connect a circuit element's terminal to a bus without specifying node-to-conductor connections. This is the default connection. Normal phase sequence is assumed. Phase 1 of the terminal is connected to Node 1 of the Bus, and so on. Neutrals default to ground (Node 0).

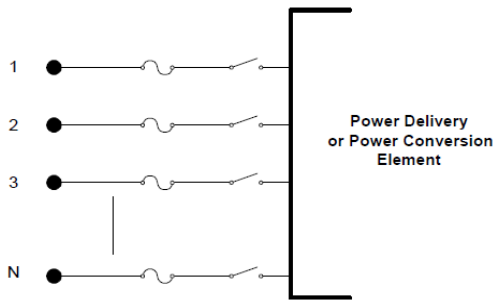


Figure 13: Terminal Definition

Figure 12: Bus Definition

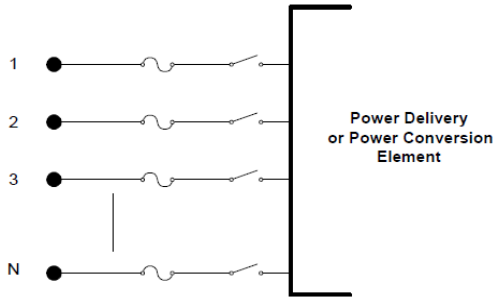


Figure 13: Terminal Definition

Power delivery elements

Power delivery elements usually consist of two or more multiphase terminals. Their basic function is to transport energy from one point to another. On the power system, the most common power delivery elements are lines and transformers. Thus, they generally have more than one terminal (capacitors and reactors can be an exception when shunt-connected rather than series-connected). Power delivery elements are standard electrical elements generally completely defined in the rms steady state by their *impedances*. Thus, they can be represented fully by the primitive **y** matrix (Yprim).

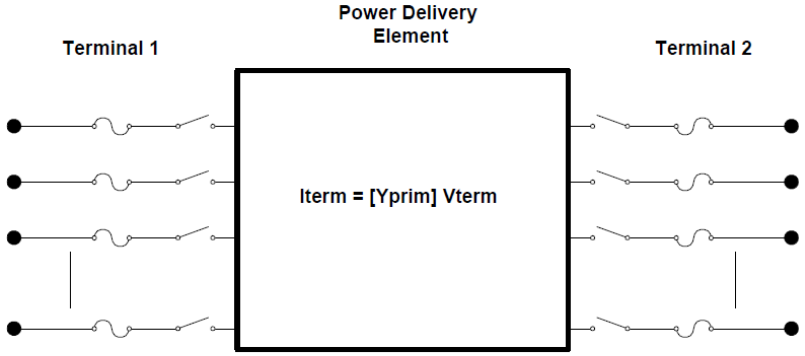


Figure 14: Power Delivery Element Definition

Power conversion elements

Power conversion elements convert power from electrical form to some other form, or vice-versa. Some may temporarily store energy and then give it back, as is the case for reactive elements. **Most will have only one connection to the power system and, therefore, only one multiphase terminal.**

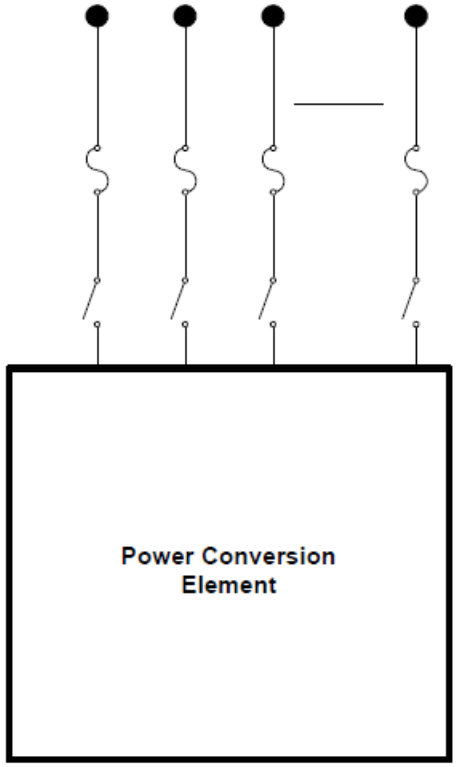


Figure 15: Power Conversion Element Definition

Power conversion elements

Within the OpenDSS, the typical implementation of a PC element is as shown below. Nonlinear elements, in particular Load and Generator elements, are treated as Norton equivalents with a constant Yprim and a "compensation" current or injection current that compensates for the nonlinear portion. This works well for most distribution loads and allows a wide range of models of load variation with voltage. It converges well for the vast majority of typical distribution system conditions. The Yprim matrix is generally kept constant for computational efficiency, although the OpenDSS does not require this. This limits the number of times the system Y matrix has to be rebuilt, which contributes greatly to computational efficiency for long runs, such as annual loading simulations.

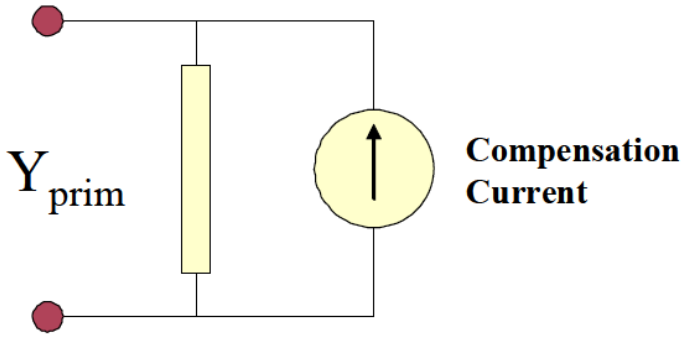


Figure 16. Compensation Current Model of PC Elements (one-line)

Power conversion elements

The Compensation current is the current that is added into the injection current vector in the main solver (see next section). This model accommodates a large variety of load models quite easily. The Load element models presently implemented are:

- **Constant P and constant Q load model**: generically called *constant power load model.* It is the model most commonly used in power flow studies. It can suffer convergence problems when the voltage deviates too far from the normal range;
- **Constant Z (or constant impedance) load model**: P and Q vary by the square of the voltage. This load model usually guarantees the convergence in any loading condition. The model is essentially linear;
- **Constant P and quadratic Q load model**: the reactive power, Q, varies quadratically with the voltage (as a constant reactance) while the active power, P, is independent from the voltage, somewhat like a motor;

…

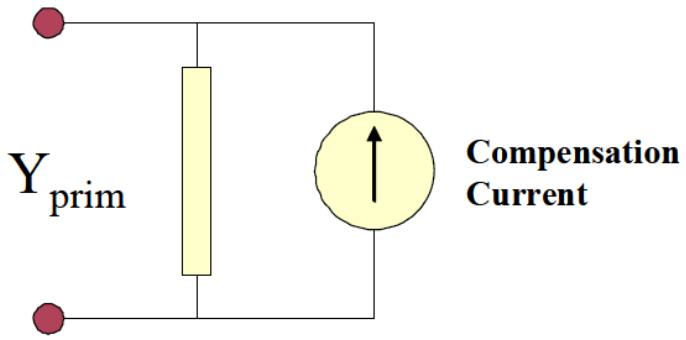


Figure 16. Compensation Current Model of PC Elements (one-line)

Power conversion elements

…

- **Exponential load model**: the voltage dependency of P and Q is defined by exponential parameters (see CVRwatts and CVRvars). This model is typically used for Conservation Voltage Reduction (CVR) studies. It is also used for general distribution feeder load mix models when the exact behavior of loads is not known;
- **Constant I** (or constant current magnitude) load model: P and Q vary linearly with the voltage magnitude while the load current magnitude remains constant. This is a common in distribution system analysis programs;
- **Constant P and fixed Q2** (Q is a fixed value independent of time and voltage);
- **Constant P and quadratic Q**: Q varies with square of the voltage
- **ZIP load model**: P and Q are described as a mixture of constant power, constant current and constant impedance load models whose contribution is defined by coefficients (see ZIPV).

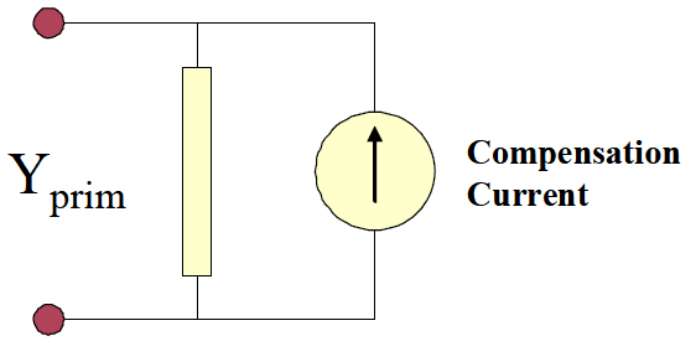


Figure 16. Compensation Current Model of PC Elements (one-line)

Power conversion elements

Loads can be exempt from loadshape mulitipliers. All load models revert to constant impedance constant Z load model outside the normal voltage range (that can defined by the user), in an attempt to guarantee convergence even when the voltage drops very low. This is important for performing annual simulations.

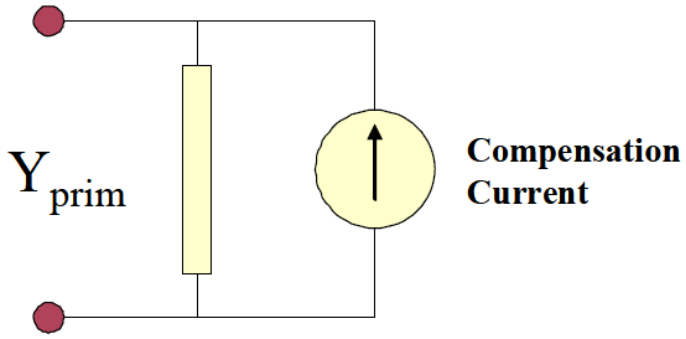


Figure 16. Compensation Current Model of PC Elements (one-line)

Putting it all together

The figure below illustrates how the DSS puts all the PD elements and PC elements together to perform a solution.

- Basically, the upper structure of the OpenDSS (the part that is written in Delphi) manages the creation and modification of the primitive y matrices (Yprim) for each element in the circuit as well as managing the bus lists, the collection of results through Meter elements, and the execution of Control elements. The Yprim matrices are fed to the sparse matrix solver, which constructs the system Y matrix.

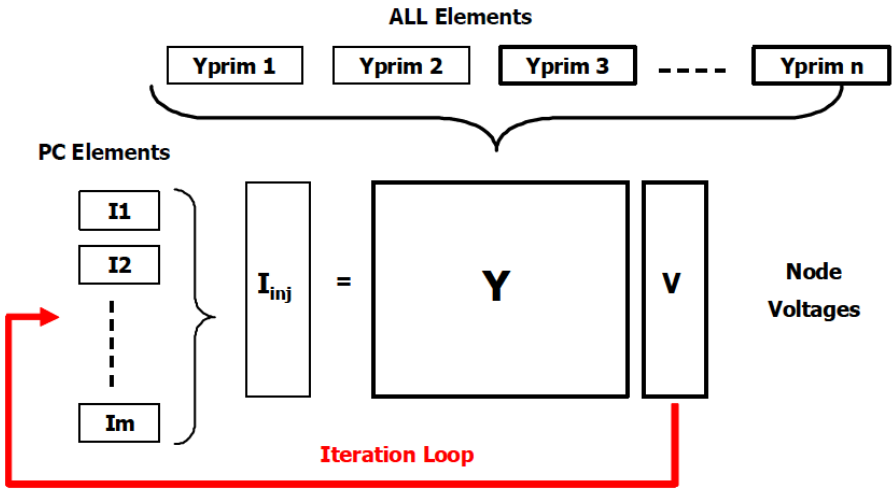


Figure 17. OpenDSS Solution Loop

Putting it all together

- An initial guess at the voltages is obtained by performing a zero load power flow. That is, all shunt elements are disconnected and only the series power delivery elements are considered. This gets all the phase angles and voltage magnitudes in the proper relationship. This is important because the OpenDSS is designed to solve arbitrary n-phase networks in which there can be all sorts of transformer ratios and connections.

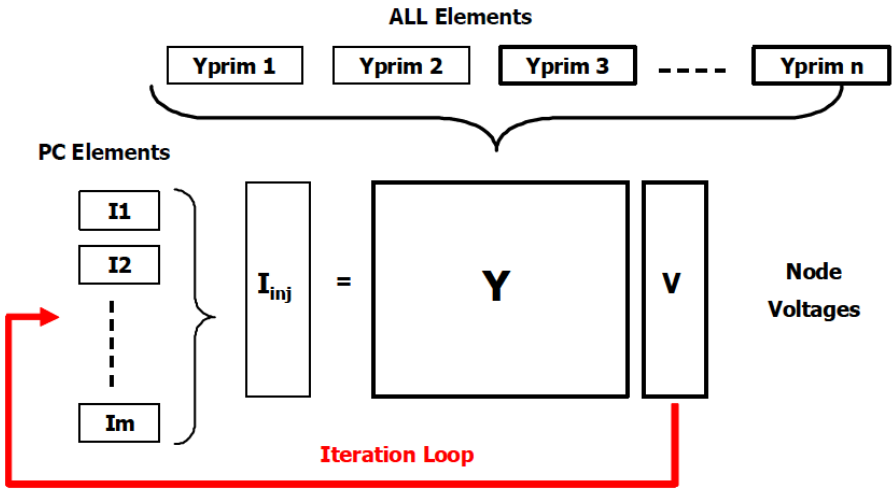


Figure 17. OpenDSS Solution Loop

Putting it all together

- The iteration cycle begins by obtaining the injection currents from all the power conversion (PC) elements in the system and adding them into the appropriate slot in the Iinj vector. The sparse set is then solved for the next guess at the voltages. The cycle repeats until the voltages converge to typically 0.0001 pu. The system Y matrix is typically not rebuilt during this process so the iterations go quickly.

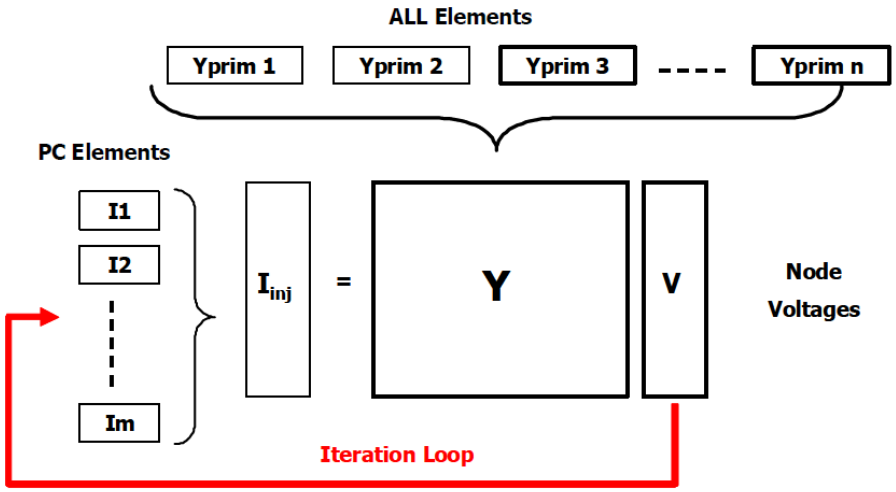


Figure 17. OpenDSS Solution Loop

Putting it all together

- This simple iterative solution has been found to converge quite well for most distribution systems that have adequate capacity to serve the load. The key is to have a dominant bulk power source, which is the case for most distribution systems. In the DSS, this is the "Normal" solution algorithm. There is also a "Newton" algorithm for more difficult systems (not to be confused with the typical Newton-Raphson power flow method).

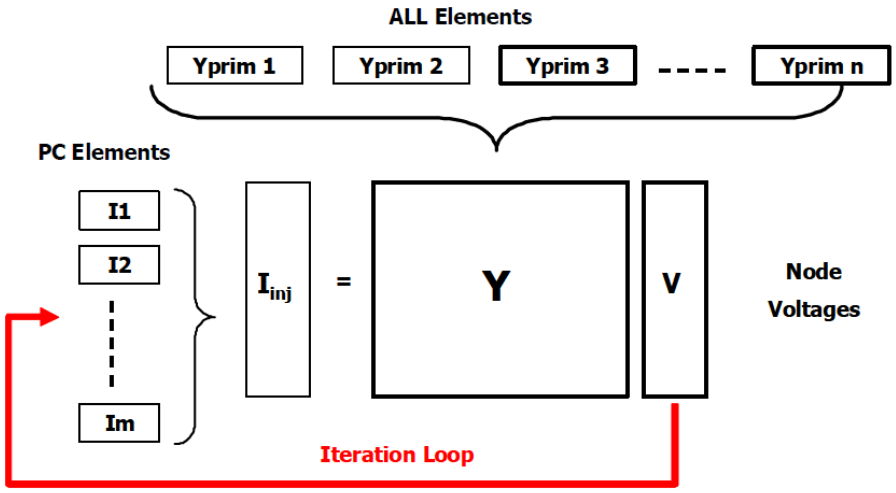


Figure 17. OpenDSS Solution Loop

Putting it all together

- When performing Daily or Yearly simulations, the solution at the present time step is used as the starting point for the solution at the next time step. Unless there is a large change in load, the solution will typically converge in 2 iterations – one to do the solution and one to check it to make sure it is converged. Thus, the DSS is able to perform such calculations quite efficiently.
- Control iterations are performed outside this loop. That is, a converged solution is achieved before checking to see if control actions are required.

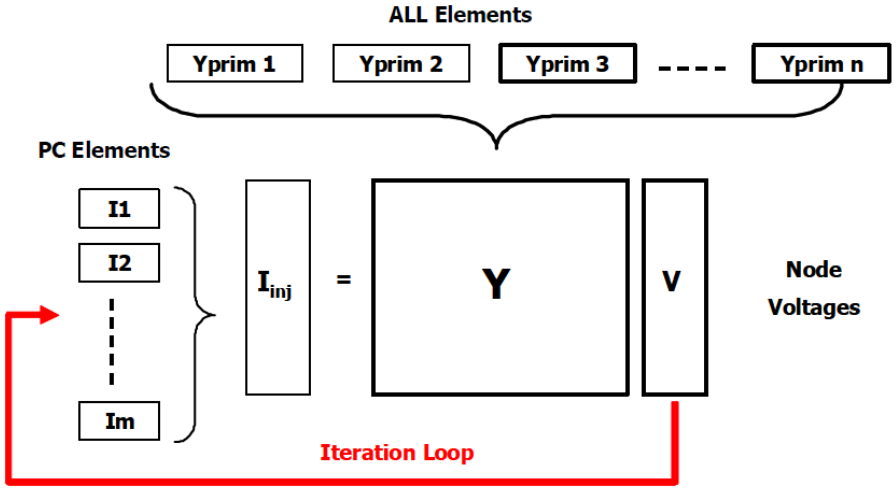


Figure 17. OpenDSS Solution Loop

Contents

- General introduction of OpenDSS
- Installation & User interface
- Workflow
- Overall concept
- Overall structure of solving power flow

Overall Structure of Solving Power Flow

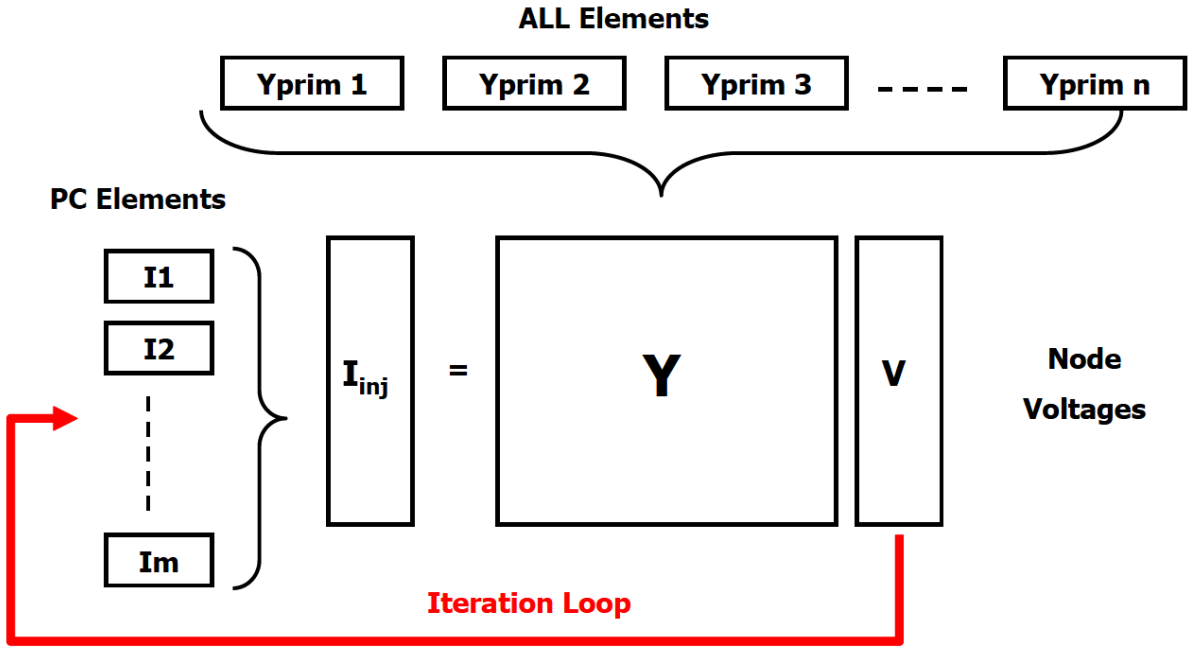


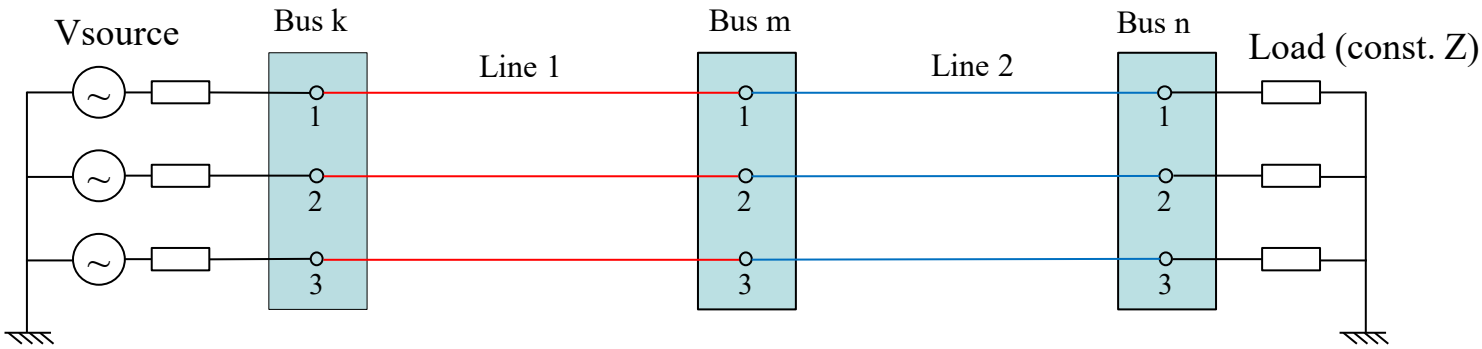
Figure 17. OpenDSS Solution Loop

This figure illustrates how the OpenDSS puts all the *power delivery* elements and *power conversion* elements together to perform a solution.

- Power delivery elements *transport* energy from one point to another.
  - Line, transformer, capacitor and reactor, etc.
- Power conversion elements *convert* power from electrical form to some other form, or vice versa.
  - Load and generator, etc.

Overall Structure of Solving Power Flow

Example 1 (Constant-Z Load):

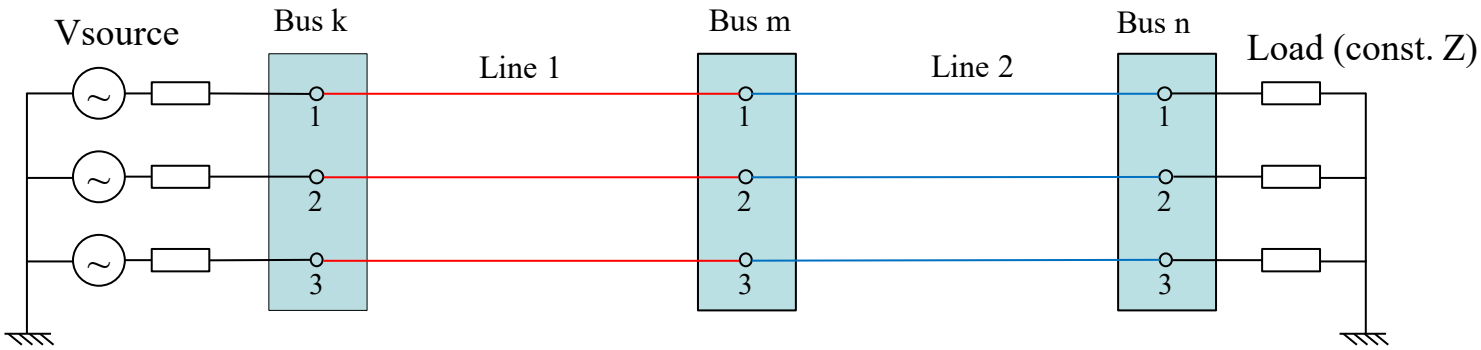


OpenDSS model:

```
Clear
//-------------- Define a new circuit --------------------//
new circuit.experiment

//---------------Define a Voltage source------------------//
Edit "Vsource.source" phases=3 basekv=13.8 Pu=1.02 bus1=K R1=5 X1=0 R0=5 X0=0

//----------------- Define line codes  --------------------//
New LineCode.Code1 nphases= 3 Units= mi
 ~ Rmatrix= (0.62 | 0.17 0.62 | 0.17 0.17 0.62 )
 ~ Xmatrix= (1.21 | 0.43 1.21 | 0.43 0.43 1.21 )

New LineCode.Code2 nphases= 3 Units= mi
 ~ Rmatrix= (1.19 | 0.23 1.19 | 0.23 0.23 1.19 )
 ~ Xmatrix= (1.21 | 0.49 1.21 | 0.49 0.49 1.21 )
… (continued)
```

Overall Structure of Solving Power Flow

Example 1:



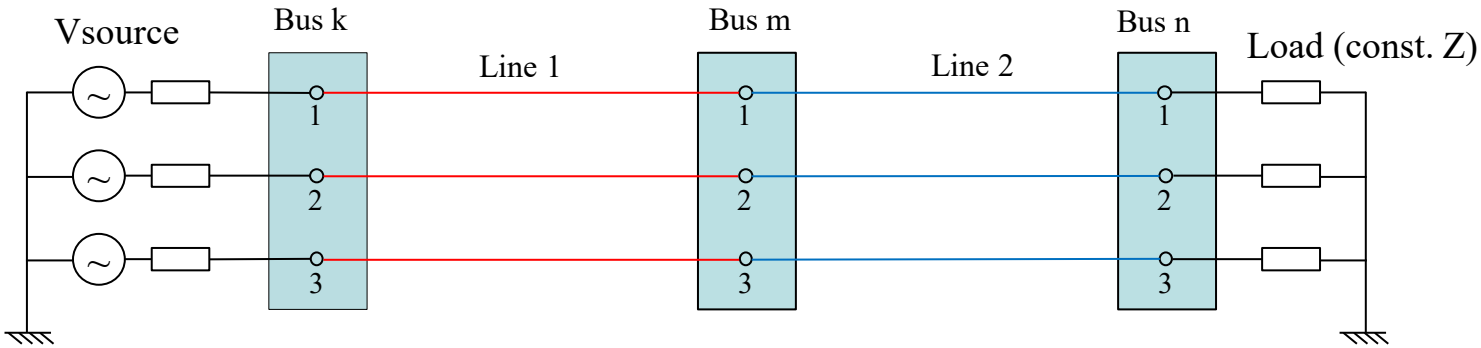
OpenDSS model:

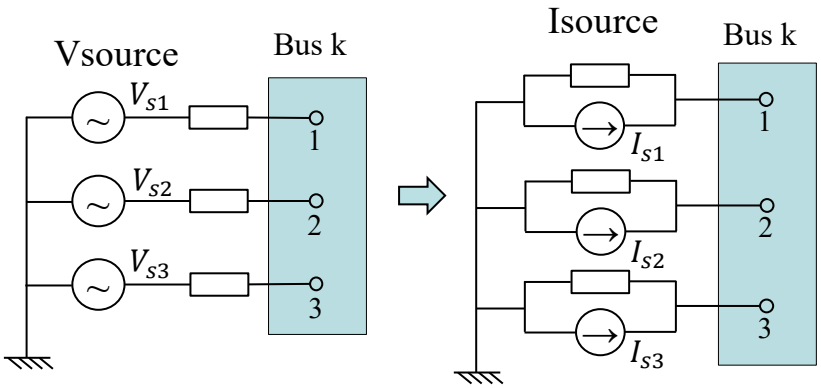
```
… (continued)
//---------------  Define line 1 and 2  ------------------//
New Line.1 phases=3  Bus1=k.1.2.3 Bus2=m.1.2.3  length=1 units=mi LineCode=Code1
New Line.2 phases=3  Bus1=m.1.2.3 Bus2=n.1.2.3  length=1 units=mi LineCode=Code2

//----------------  Define a load   ----------------------//
New Load.L Bus1=n.1.2.3.0       Phases=3 Conn=wye Model=2 kV=13.8   kW=500 kvar=0

Set VoltageBases = "13.8"
Solve
```

Constant-impedance load

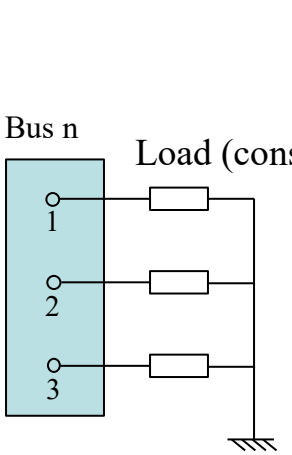
Overall Structure of Solving Power Flow

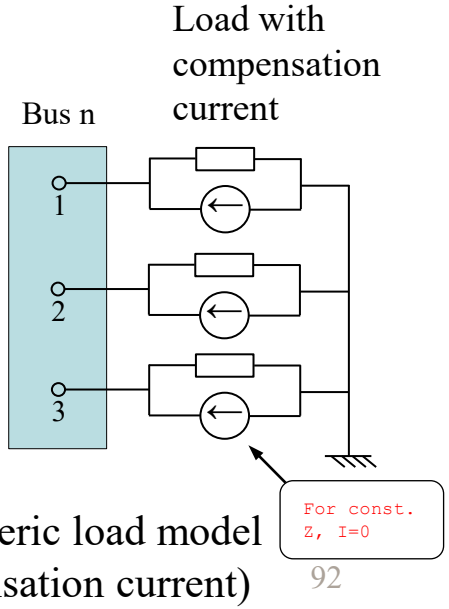
Example 1 (Injection Current):





Thevinen equivalent to Norton equivalent

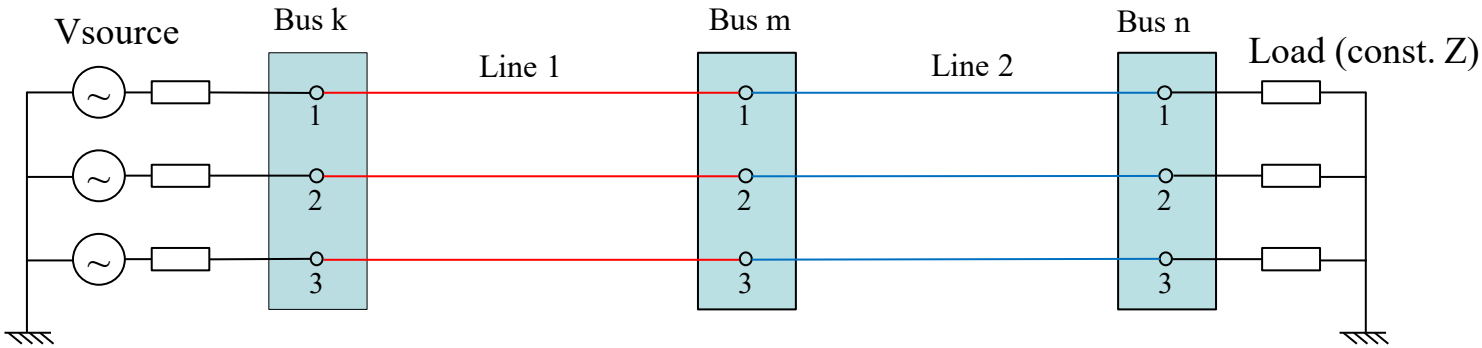


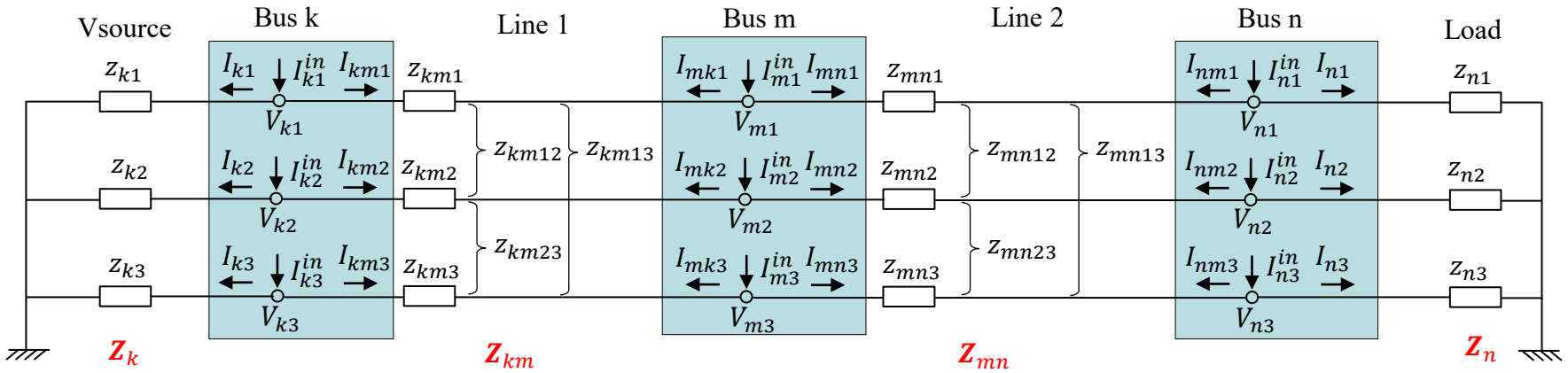


Various loads to a generic load model (Constant Z + compensation current)

Overall Structure of Solving Power Flow

Example 1 (Injection Current):





$$I_{k1}^{in} = I_{s1} = \frac{V_{s1}}{z_{k1}}, I_{k2}^{in} = I_{s2} = \frac{V_{s2}}{z_{k2}}, I_{k3}^{in} = I_{s3} = \frac{V_{s3}}{z_{k3}}$$

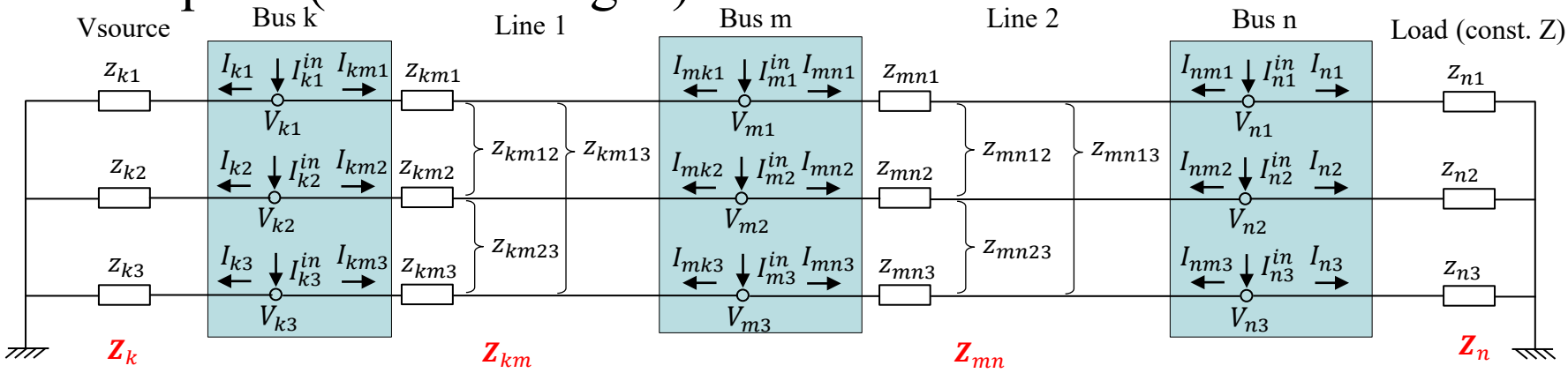
$$I_{m1}^{in} = I_{m2}^{in} = I_{m3}^{in} = 0$$

Compensation current:

$$I_{n1}^{in} = I_{n2}^{in} = I_{n3}^{in} = 0$$

Overall Structure of Solving Power Flow

Example 1 (Constructing **Y**):



Definitions:

$$\boldsymbol{Z}_k = \begin{bmatrix} z_{k1} & 0 & 0 \\ 0 & z_{k2} & 0 \\ 0 & 0 & z_{k3} \end{bmatrix}, \quad \boldsymbol{Y}_k = \boldsymbol{Z}_k^{-1} = \begin{bmatrix} y_{k1} & 0 & 0 \\ 0 & y_{k2} & 0 \\ 0 & 0 & y_{k3} \end{bmatrix}$$

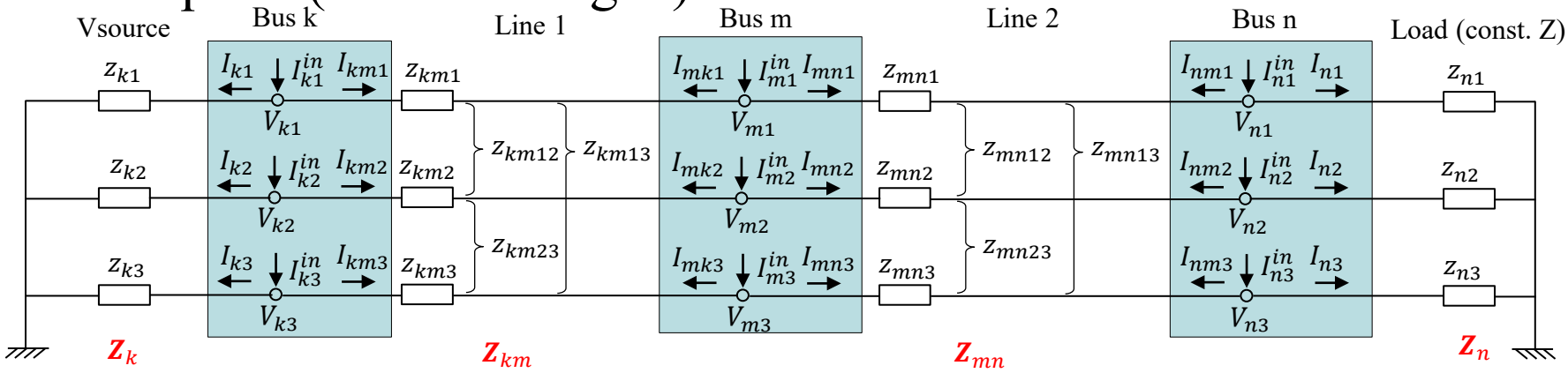
$$\boldsymbol{Z}_{km} = \begin{bmatrix} z_{km1} & z_{km12} & z_{km13} \\ z_{km12} & z_{km2} & z_{km23} \\ z_{km13} & z_{km23} & z_{km3} \end{bmatrix}, \quad \boldsymbol{Y}_{km} = \boldsymbol{Z}_{km}^{-1} = \begin{bmatrix} y_{km1} & y_{km12} & y_{km13} \\ y_{km12} & y_{km2} & y_{km23} \\ y_{km13} & y_{km23} & y_{km3} \end{bmatrix}$$

$$\boldsymbol{Z}_{mn} = \begin{bmatrix} z_{mn1} & z_{mn12} & z_{mn13} \\ z_{mn12} & z_{mn2} & z_{mn23} \\ z_{mn13} & z_{mn23} & z_{mn3} \end{bmatrix}, \quad \boldsymbol{Y}_{mn} = \boldsymbol{Z}_{mn}^{-1} = \begin{bmatrix} y_{mn1} & y_{mn12} & y_{mn13} \\ y_{mn12} & y_{mn2} & y_{mn23} \\ y_{mn13} & y_{mn23} & y_{mn3} \end{bmatrix}$$

$$\boldsymbol{Z}_n = \begin{bmatrix} z_{n1} & 0 & 0 \\ 0 & z_{n2} & 0 \\ 0 & 0 & z_{n3} \end{bmatrix}, \quad \boldsymbol{Y}_n = \boldsymbol{Z}_n^{-1} = \begin{bmatrix} y_{n1} & 0 & 0 \\ 0 & y_{n2} & 0 \\ 0 & 0 & y_{n3} \end{bmatrix}$$

Overall Structure of Solving Power Flow

Example 1 (Constructing **Y**):



(1) Bus k:

1) Branch currents of Vsource:

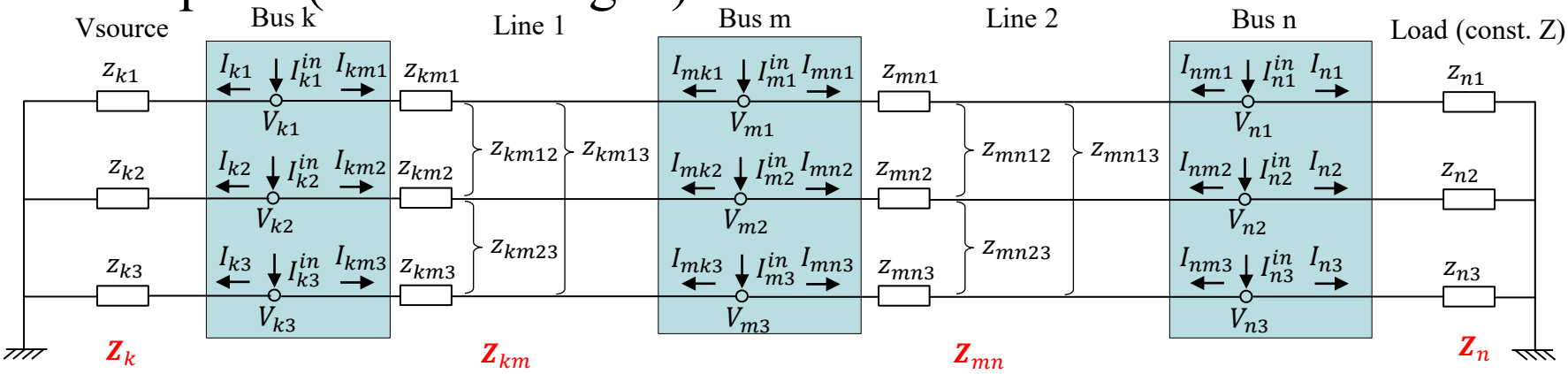
According to Kirchhoff's voltage law:

$$\begin{bmatrix} V_{k1} \\ V_{k2} \\ V_{k3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} z_{k1} & 0 & 0 \\ 0 & z_{k2} & 0 \\ 0 & 0 & z_{k3} \end{bmatrix} \begin{bmatrix} I_{k1} \\ I_{k2} \\ I_{k3} \end{bmatrix} \Rightarrow \begin{bmatrix} I_{k1} \\ I_{k2} \\ I_{k3} \end{bmatrix} = \begin{bmatrix} z_{k1} & 0 & 0 \\ 0 & z_{k2} & 0 \\ 0 & 0 & z_{k3} \end{bmatrix}^{-1} \begin{bmatrix} V_{k1} - 0 \\ V_{k2} - 0 \\ V_{k3} - 0 \end{bmatrix} = \begin{bmatrix} y_{k1} & 0 & 0 \\ 0 & y_{k2} & 0 \\ 0 & 0 & y_{k3} \end{bmatrix} \begin{bmatrix} V_{k1} \\ V_{k2} \\ V_{k3} \end{bmatrix}$$

$$\Rightarrow \boldsymbol{I}_k = \boldsymbol{Y}_k \boldsymbol{V}_k, \text{ where, } \boldsymbol{I}_k = \begin{bmatrix} I_{k1} \\ I_{k2} \\ I_{k3} \end{bmatrix}, \boldsymbol{V}_k = \begin{bmatrix} V_{k1} \\ V_{k2} \\ V_{k3} \end{bmatrix}.$$

Overall Structure of Solving Power Flow

Example 1 (Constructing **Y**):



(1) Bus k:

2) Branch currents of Line 1:

According to Kirchhoff's voltage law:

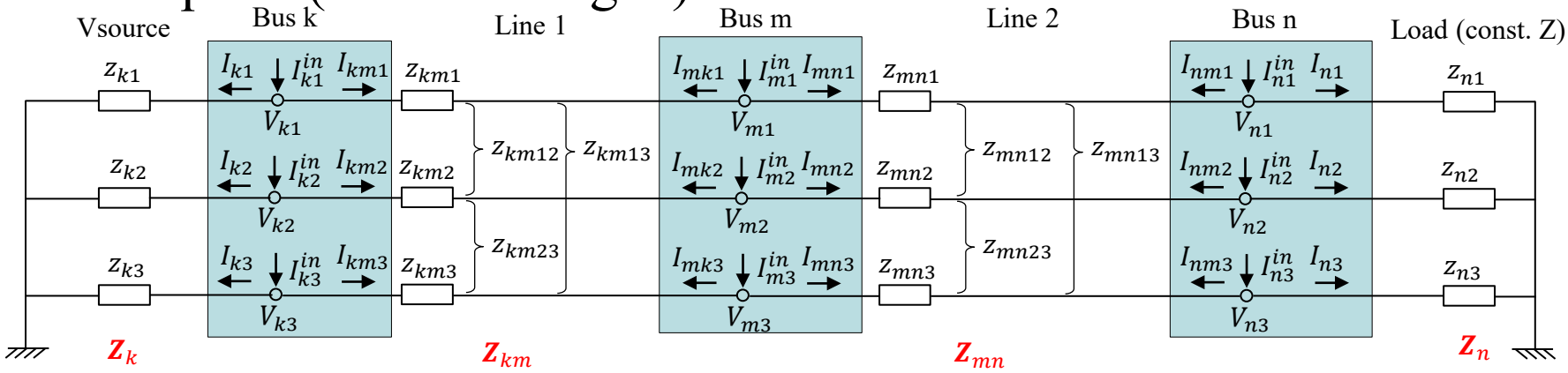
$$\begin{bmatrix} V_{k1} \\ V_{k2} \\ V_{k3} \end{bmatrix} = \begin{bmatrix} V_{m1} \\ V_{m2} \\ V_{m3} \end{bmatrix} + \begin{bmatrix} z_{km1} & z_{km12} & z_{km13} \\ z_{km12} & z_{km2} & z_{km23} \\ z_{km13} & z_{km23} & z_{km3} \end{bmatrix} \begin{bmatrix} I_{km1} \\ I_{km2} \\ I_{km3} \end{bmatrix} \Rightarrow \begin{bmatrix} I_{km1} \\ I_{km2} \\ I_{km3} \end{bmatrix} = \begin{bmatrix} z_{km1} & z_{km12} & z_{km13} \\ z_{km12} & z_{km2} & z_{km23} \\ z_{km13} & z_{km23} & z_{km3} \end{bmatrix}^{-1} \begin{bmatrix} V_{k1} - V_{m1} \\ V_{k2} - V_{m2} \\ V_{k3} - V_{m3} \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} I_{km1} \\ I_{km2} \\ I_{km3} \end{bmatrix} = \begin{bmatrix} y_{km1} & y_{km12} & y_{km13} \\ y_{km12} & y_{km2} & y_{km23} \\ y_{km13} & y_{km23} & y_{km3} \end{bmatrix} \begin{bmatrix} V_{k1} - V_{m1} \\ V_{k2} - V_{m2} \\ V_{k3} - V_{m3} \end{bmatrix} \Rightarrow \boldsymbol{I}_{km} = \boldsymbol{Y}_{km}(\boldsymbol{V}_k - \boldsymbol{V}_m)$$

where, $\boldsymbol{I}_{km} = \begin{bmatrix} I_{km1} \\ I_{km2} \\ I_{km3} \end{bmatrix}$, $\boldsymbol{V}_m = \begin{bmatrix} V_{m1} \\ V_{m2} \\ V_{m3} \end{bmatrix}$.

Overall Structure of Solving Power Flow

Example 1 (Constructing **Y**):



(1) Bus k:

Injection currents at Bus k:

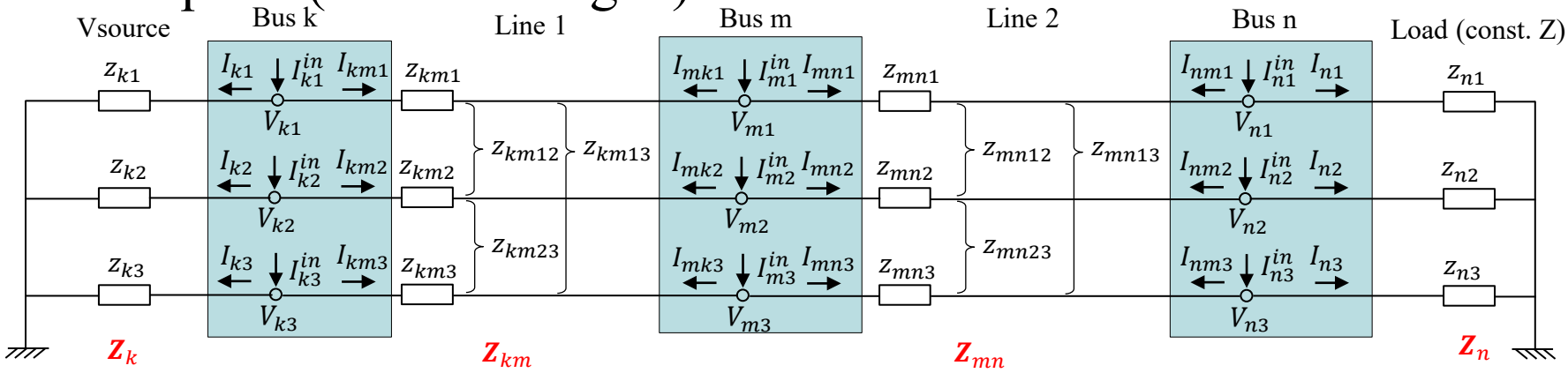
$$\begin{bmatrix} I_{k1}^{in} \\ I_{k2}^{in} \\ I_{k3}^{in} \end{bmatrix} = \begin{bmatrix} I_{k1} \\ I_{k2} \\ I_{k3} \end{bmatrix} + \begin{bmatrix} I_{km1} \\ I_{km2} \\ I_{km3} \end{bmatrix}$$

$$\Rightarrow \boldsymbol{I}_k^{in} = \boldsymbol{I}_k + \boldsymbol{I}_{km} = \boldsymbol{Y}_k \boldsymbol{V}_k + \boldsymbol{Y}_{km}(\boldsymbol{V}_k - \boldsymbol{V}_m)$$
$$= (\boldsymbol{Y}_k + \boldsymbol{Y}_{km})\boldsymbol{V}_k - \boldsymbol{Y}_{km}\boldsymbol{V}_m$$

$$\text{where, } \boldsymbol{I}_k^{in} = \begin{bmatrix} I_{k1}^{in} \\ I_{k2}^{in} \\ I_{k3}^{in} \end{bmatrix}.$$

Overall Structure of Solving Power Flow

Example 1 (Constructing **Y**):



(2) Bus m:

1) Branch currents of Line 1

According to Kirchhoff's voltage law:

Strictly, z_{km*} should be z_{mk*}. For a line, $z_{km*} = z_{mk*}$, so z_{km*} is used here.

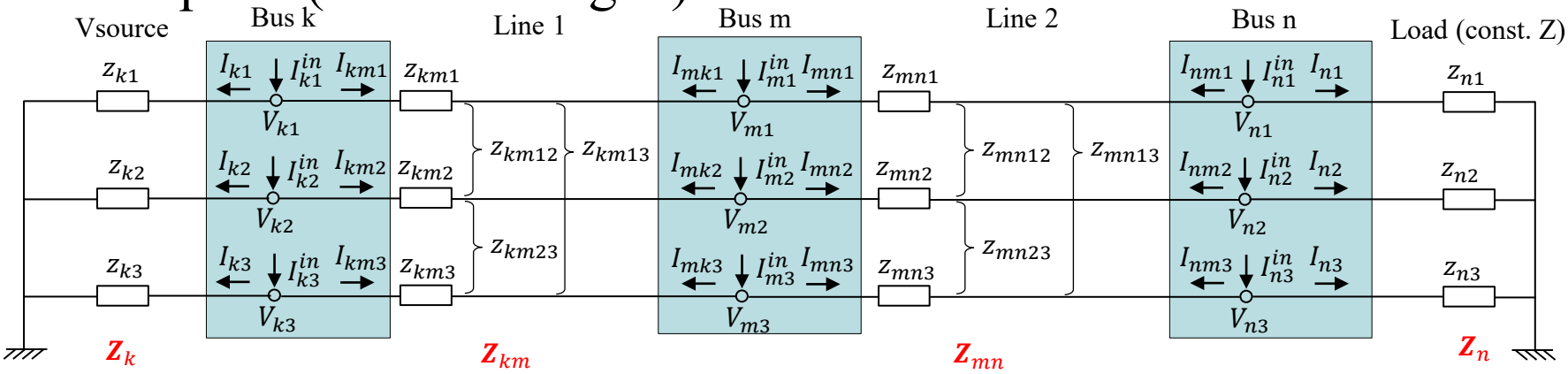
$$\begin{bmatrix} V_{m1} \\ V_{m2} \\ V_{m3} \end{bmatrix} = \begin{bmatrix} V_{k1} \\ V_{k2} \\ V_{k3} \end{bmatrix} + \begin{bmatrix} z_{km1} & z_{km12} & z_{km13} \\ z_{km12} & z_{km2} & z_{km23} \\ z_{km13} & z_{km23} & z_{km3} \end{bmatrix} \begin{bmatrix} I_{mk1} \\ I_{mk2} \\ I_{mk3} \end{bmatrix} \Rightarrow \begin{bmatrix} I_{mk1} \\ I_{mk2} \\ I_{mk3} \end{bmatrix} = \begin{bmatrix} z_{km1} & z_{km12} & z_{km13} \\ z_{km12} & z_{km2} & z_{km23} \\ z_{km13} & z_{km23} & z_{km3} \end{bmatrix}^{-1} \begin{bmatrix} V_{m1} - V_{k1} \\ V_{m2} - V_{k2} \\ V_{m3} - V_{k3} \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} I_{mk1} \\ I_{mk2} \\ I_{mk3} \end{bmatrix} = \begin{bmatrix} y_{km1} & y_{km12} & y_{km13} \\ y_{km12} & y_{km2} & y_{km23} \\ y_{km13} & y_{km23} & y_{km3} \end{bmatrix} \begin{bmatrix} V_{m1} - V_{k1} \\ V_{m2} - V_{k2} \\ V_{m3} - V_{k3} \end{bmatrix} \Rightarrow \boldsymbol{I}_{mk} = \boldsymbol{Y}_{km}(\boldsymbol{V}_m - \boldsymbol{V}_k)$$

where, $\boldsymbol{I}_{mk} = \begin{bmatrix} I_{mk1} \\ I_{mk2} \\ I_{mk3} \end{bmatrix}$.

Overall Structure of Solving Power Flow

Example 1 (Constructing **Y**):



(2) Bus m:

2) Branch currents of Line 2

According to Kirchhoff's voltage law:

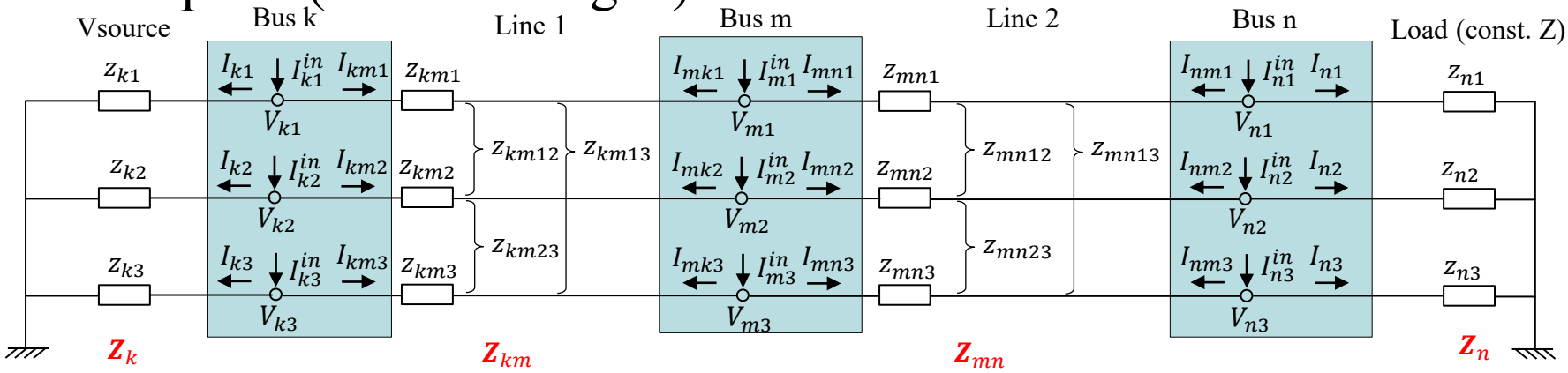
$$\begin{bmatrix} V_{m1} \\ V_{m2} \\ V_{m3} \end{bmatrix} = \begin{bmatrix} V_{n1} \\ V_{n2} \\ V_{n3} \end{bmatrix} + \begin{bmatrix} z_{mn1} & z_{mn12} & z_{mn13} \\ z_{mn12} & z_{mn2} & z_{mn23} \\ z_{mn13} & z_{mn23} & z_{mn3} \end{bmatrix} \begin{bmatrix} I_{mn1} \\ I_{mn2} \\ I_{mn3} \end{bmatrix} \Rightarrow \begin{bmatrix} I_{mn1} \\ I_{mn2} \\ I_{mn3} \end{bmatrix} = \begin{bmatrix} z_{mn1} & z_{mn12} & z_{mn13} \\ z_{mn12} & z_{mn2} & z_{mn23} \\ z_{mn13} & z_{mn23} & z_{mn3} \end{bmatrix}^{-1} \begin{bmatrix} V_{m1} - V_{n1} \\ V_{m2} - V_{n2} \\ V_{m3} - V_{n3} \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} I_{mn1} \\ I_{mn2} \\ I_{mn3} \end{bmatrix} = \begin{bmatrix} y_{mn1} & y_{mn12} & y_{mn13} \\ y_{mn12} & y_{mn2} & y_{mn23} \\ y_{mn13} & y_{mn23} & y_{mn3} \end{bmatrix} \begin{bmatrix} V_{m1} - V_{n1} \\ V_{m2} - V_{n2} \\ V_{m3} - V_{n3} \end{bmatrix} \Rightarrow \boldsymbol{I}_{mn} = \boldsymbol{Y}_{mn}(\boldsymbol{V}_m - \boldsymbol{V}_n)$$

where, $\boldsymbol{I}_{mn} = \begin{bmatrix} I_{mn1} \\ I_{mn2} \\ I_{mn3} \end{bmatrix}$, $\boldsymbol{V}_n = \begin{bmatrix} V_{n1} \\ V_{n2} \\ V_{n3} \end{bmatrix}$.

Overall Structure of Solving Power Flow

Example 1 (Constructing **Y**):



(2) Bus m:

Injection currents at Bus m:

$$\begin{bmatrix} I_{m1}^{in} \\ I_{m2}^{in} \\ I_{m3}^{in} \end{bmatrix} = \begin{bmatrix} I_{mk1} \\ I_{mk2} \\ I_{mk3} \end{bmatrix} + \begin{bmatrix} I_{mn1} \\ I_{mn2} \\ I_{mn3} \end{bmatrix}$$

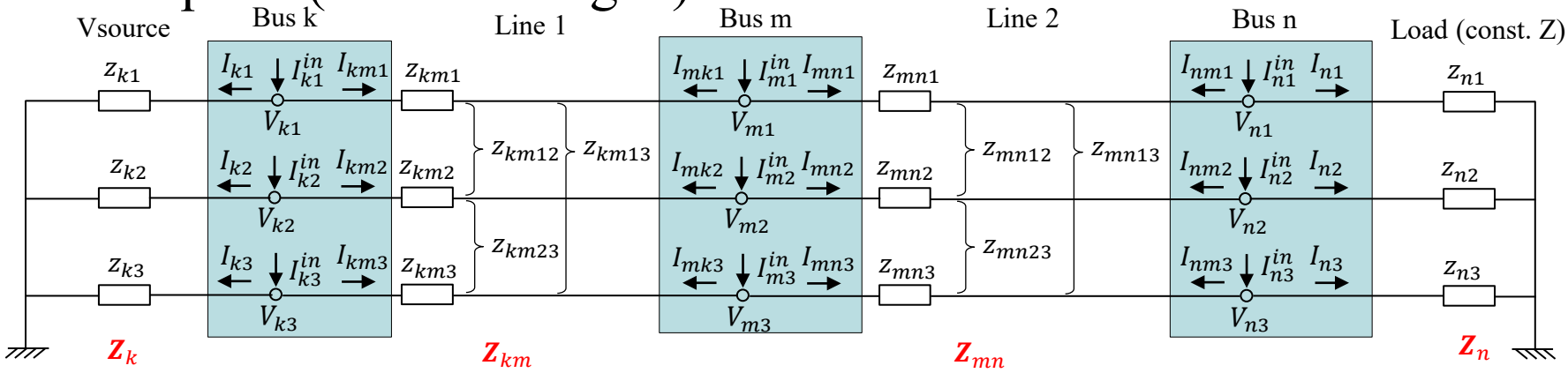
$$\Rightarrow \boldsymbol{I}_m^{in} = \boldsymbol{I}_{mk} + \boldsymbol{I}_{mn} = \boldsymbol{Y}_{km}(\boldsymbol{V}_m - \boldsymbol{V}_k) + \boldsymbol{Y}_{mn}(\boldsymbol{V}_m - \boldsymbol{V}_n)$$

$$= (\boldsymbol{Y}_{km} + \boldsymbol{Y}_{mn})\boldsymbol{V}_m - \boldsymbol{Y}_{km}\boldsymbol{V}_k - \boldsymbol{Y}_{mn}\boldsymbol{V}_n$$

$$\text{where, } \boldsymbol{I}_m^{in} = \begin{bmatrix} I_{m1}^{in} \\ I_{m2}^{in} \\ I_{m3}^{in} \end{bmatrix}.$$

Overall Structure of Solving Power Flow

Example 1 (Constructing **Y**):



(3) Bus n:

1) Branch currents of Line 2

According to Kirchhoff's voltage law:

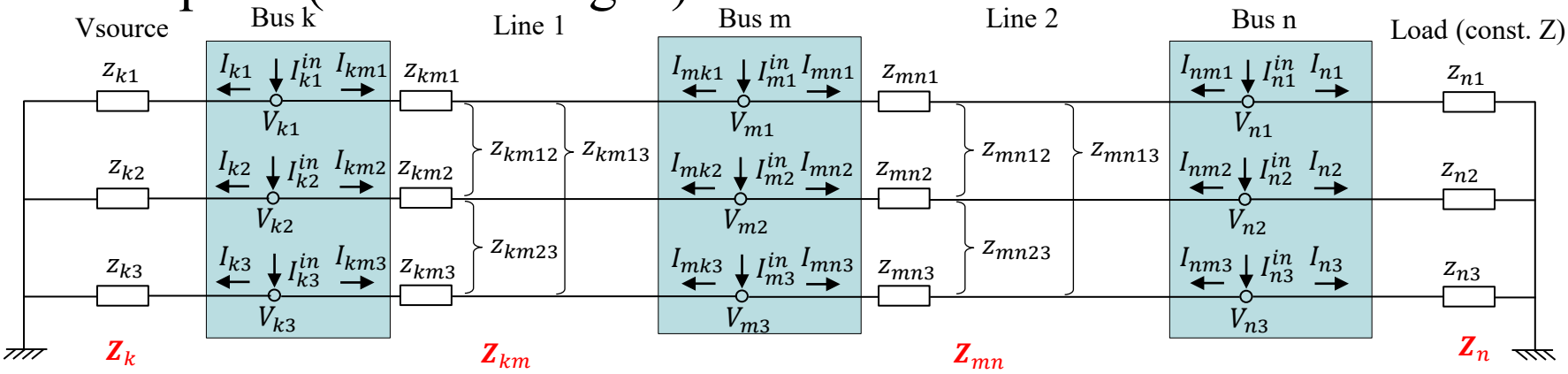
$$\begin{bmatrix} V_{n1} \\ V_{n2} \\ V_{n3} \end{bmatrix} = \begin{bmatrix} V_{m1} \\ V_{m2} \\ V_{m3} \end{bmatrix} + \begin{bmatrix} z_{mn1} & z_{mn12} & z_{mn13} \\ z_{mn12} & z_{mn2} & z_{mn23} \\ z_{mn13} & z_{mn23} & z_{mn3} \end{bmatrix} \begin{bmatrix} I_{nm1} \\ I_{nm2} \\ I_{nm3} \end{bmatrix} \Rightarrow \begin{bmatrix} I_{nm1} \\ I_{nm2} \\ I_{nm3} \end{bmatrix} = \begin{bmatrix} z_{mn1} & z_{mn12} & z_{mn13} \\ z_{mn12} & z_{mn2} & z_{mn23} \\ z_{mn13} & z_{mn23} & z_{mn3} \end{bmatrix}^{-1} \begin{bmatrix} V_{n1} - V_{m1} \\ V_{n2} - V_{m2} \\ V_{n3} - V_{m3} \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} I_{nm1} \\ I_{nm2} \\ I_{nm3} \end{bmatrix} = \begin{bmatrix} y_{mn1} & y_{mn12} & y_{mn13} \\ y_{mn12} & y_{mn2} & y_{mn23} \\ y_{mn13} & y_{mn23} & y_{mn3} \end{bmatrix} \begin{bmatrix} V_{n1} - V_{m1} \\ V_{n2} - V_{m2} \\ V_{n3} - V_{m3} \end{bmatrix} \Rightarrow \boldsymbol{I}_{nm} = \boldsymbol{Y}_{mn}(\boldsymbol{V}_n - \boldsymbol{V}_m)$$

where, $\boldsymbol{I}_{nm} = \begin{bmatrix} I_{nm1} \\ I_{nm2} \\ I_{nm3} \end{bmatrix}$.

Overall Structure of Solving Power Flow

Example 1 (Constructing **Y**):



(3) Bus n:

2) Branch currents of Load

According to Kirchhoff's voltage law:

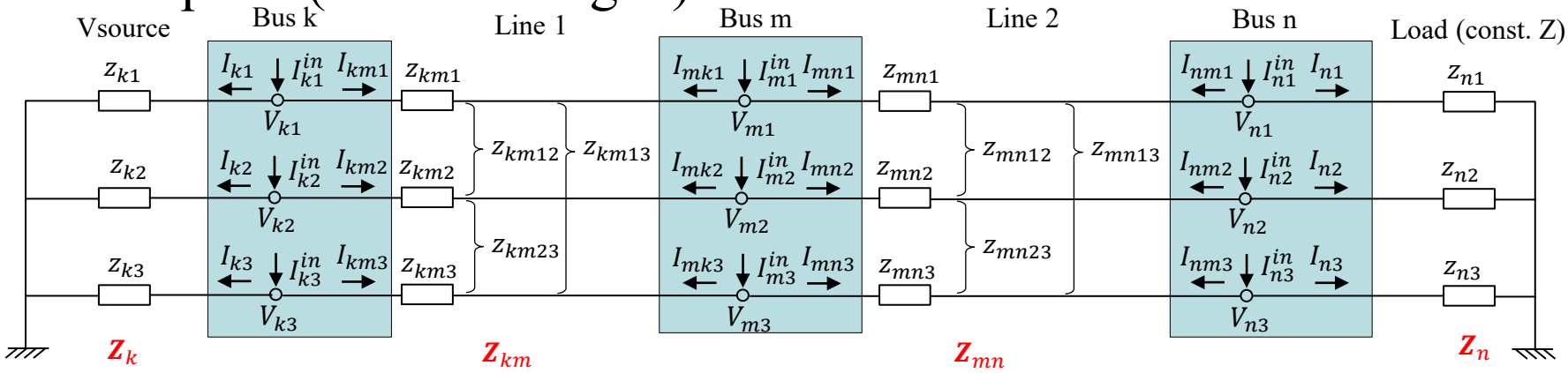
$$\begin{bmatrix} V_{n1} \\ V_{n2} \\ V_{n3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} z_{n1} & 0 & 0 \\ 0 & z_{n2} & 0 \\ 0 & 0 & z_{n3} \end{bmatrix} \begin{bmatrix} I_{n1} \\ I_{n2} \\ I_{n3} \end{bmatrix} \Rightarrow \begin{bmatrix} I_{n1} \\ I_{n2} \\ I_{m3} \end{bmatrix} = \begin{bmatrix} z_{n1} & 0 & 0 \\ 0 & z_{n2} & 0 \\ 0 & 0 & z_{n3} \end{bmatrix}^{-1} \begin{bmatrix} V_{n1} \\ V_{n2} \\ V_{n3} \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} I_{n1} \\ I_{n2} \\ I_{m3} \end{bmatrix} = \begin{bmatrix} y_{n1} & 0 & 0 \\ 0 & y_{n2} & 0 \\ 0 & 0 & y_{n3} \end{bmatrix} \begin{bmatrix} V_{n1} \\ V_{n2} \\ V_{n3} \end{bmatrix} \Rightarrow \boldsymbol{I}_n = \boldsymbol{Y}_n \boldsymbol{V}_n$$

where, $\boldsymbol{I}_n = \begin{bmatrix} I_{n1} \\ I_{n2} \\ I_{m3} \end{bmatrix}$.

Overall Structure of Solving Power Flow

Example 1 (Constructing **Y**):



(3) Bus n:

Injection currents at Bus n:

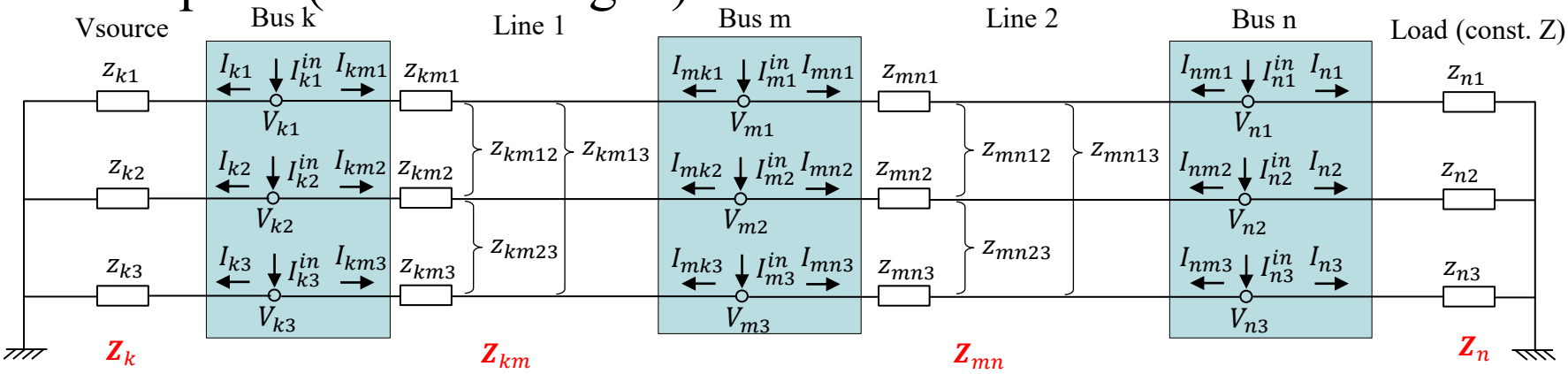
$$\begin{bmatrix} I_{n1}^{in} \\ I_{n2}^{in} \\ I_{n3}^{in} \end{bmatrix} = \begin{bmatrix} I_{nm1} \\ I_{nm2} \\ I_{nm3} \end{bmatrix} + \begin{bmatrix} I_{n1} \\ I_{n2} \\ I_{n3} \end{bmatrix}$$

$$\Rightarrow \boldsymbol{I}_n^{in} = \boldsymbol{I}_{nm} + \boldsymbol{I}_n = \boldsymbol{Y}_{mn}(\boldsymbol{V}_n - \boldsymbol{V}_m) + \boldsymbol{Y}_n\boldsymbol{V}_n$$
$$= (\boldsymbol{Y}_{mn} + \boldsymbol{Y}_n)\boldsymbol{V}_n - \boldsymbol{Y}_{mn}\boldsymbol{V}_m$$

$$\text{where, } \boldsymbol{I}_n^{in} = \begin{bmatrix} I_{n1}^{in} \\ I_{n2}^{in} \\ I_{n3}^{in} \end{bmatrix}.$$

Overall Structure of Solving Power Flow

Example 1 (Constructing **Y**):



Put all the injection currents together:

$$\begin{cases} \boldsymbol{I}_k^{in} = (\boldsymbol{Y}_k + \boldsymbol{Y}_{km})\boldsymbol{V}_k - \boldsymbol{Y}_{km}\boldsymbol{V}_m \\ \boldsymbol{I}_m^{in} = (\boldsymbol{Y}_{km} + \boldsymbol{Y}_{mn})\boldsymbol{V}_m - \boldsymbol{Y}_{km}\boldsymbol{V}_k - \boldsymbol{Y}_{mn}\boldsymbol{V}_n \\ \boldsymbol{I}_n^{in} = (\boldsymbol{Y}_{mn} + \boldsymbol{Y}_n)\boldsymbol{V}_n - \boldsymbol{Y}_{mn}\boldsymbol{V}_m \end{cases}$$

$$\Rightarrow \begin{bmatrix} \boldsymbol{I}_k^{in} \\ \boldsymbol{I}_m^{in} \\ \boldsymbol{I}_n^{in} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Y}_k + \boldsymbol{Y}_{km} & -\boldsymbol{Y}_{km} & \boldsymbol{0} \\ -\boldsymbol{Y}_{km} & \boldsymbol{Y}_{km} + \boldsymbol{Y}_{mn} & -\boldsymbol{Y}_{mn} \\ \boldsymbol{0} & -\boldsymbol{Y}_{mn} & \boldsymbol{Y}_{mn} + \boldsymbol{Y}_n \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_k \\ \boldsymbol{V}_m \\ \boldsymbol{V}_n \end{bmatrix}$$

$$\Rightarrow \mathbf{I}_{\text{inj}} = \mathbf{YV}$$

where, $\mathbf{I}_{\text{inj}} = \begin{bmatrix} \boldsymbol{I}_k^{in} \\ \boldsymbol{I}_m^{in} \\ \boldsymbol{I}_n^{in} \end{bmatrix}$, $\mathbf{Y} = \begin{bmatrix} \boldsymbol{Y}_k + \boldsymbol{Y}_{km} & -\boldsymbol{Y}_{km} & \boldsymbol{0} \\ -\boldsymbol{Y}_{km} & \boldsymbol{Y}_{km} + \boldsymbol{Y}_{mn} & -\boldsymbol{Y}_{mn} \\ \boldsymbol{0} & -\boldsymbol{Y}_{mn} & \boldsymbol{Y}_{mn} + \boldsymbol{Y}_n \end{bmatrix}$, $\mathbf{V} = \begin{bmatrix} \boldsymbol{V}_k \\ \boldsymbol{V}_m \\ \boldsymbol{V}_n \end{bmatrix}$.

Conclusions for constructing $\mathbf{Y}$: (1) $\mathbf{Y}_{ij} = \mathbf{Y}_{ji}$; (2) $\mathbf{Y}_{ij} = -\boldsymbol{Y}_{ij}$; (3) $\mathbf{Y}_{ii} = \boldsymbol{Y}_i + \sum_{k=1,k\neq i}^{N} \boldsymbol{Y}_{ik}$

Overall Structure of Solving Power Flow

Exported primitive admittance matrix of each element from OpenDSS:

	G						B					
	Vsource											
Y_k	0.2	0	0	0	0	0	-0.2	0	0	0	0	0
	0	0	0.2	0	0	0	0	0	-0.2	0	0	0
	0	0	0	0	0.2	0	0	0	0	0	-0.2	0
$-Y_k$	-0.2	0	0	0	0	0	0.2	0	0	0	0	0
	0	0	-0.2	0	0	0	0	0	0.2	0	0	0
	0	0	0	0	-0.2	0	0	0	0	0	0.2	0
	Line 1											
Y_{km}	0.4314	-0.7738	-0.1235	0.1881	-0.1235	0.1881	-0.4314	0.7738	0.1235	-0.1881	0.1235	-0.1881
	-0.1235	0.1881	0.4314	-0.7738	-0.1235	0.1881	0.1235	-0.1881	-0.4314	0.7738	0.1235	-0.1881
	-0.1235	0.1881	-0.1235	0.1881	0.4314	-0.7738	0.1235	-0.1881	0.1235	-0.1881	-0.4314	0.7738
$-Y_{km}$	-0.4314	0.7738	0.1235	-0.1881	0.1235	-0.1881	0.4314	-0.7738	-0.1235	0.1881	-0.1235	0.1881
	0.1235	-0.1881	-0.4314	0.7738	0.1235	-0.1881	-0.1235	0.1881	0.4314	-0.7738	-0.1235	0.1881
	0.1235	-0.1881	0.1235	-0.1881	-0.4314	0.7738	-0.1235	0.1881	-0.1235	0.1881	0.4314	-0.7738
	Line 2											
Y_{mn}	0.5176	-0.4304	-0.1491	0.0696	-0.1491	0.0696	-0.5176	0.4304	0.1491	-0.0696	0.1491	-0.0696
	-0.1491	0.0696	0.5176	-0.4304	-0.1491	0.0696	0.1491	-0.0696	-0.5176	0.4304	0.1491	-0.0696
	-0.1491	0.0696	-0.1491	0.0696	0.5176	-0.4304	0.1491	-0.0696	0.1491	-0.0696	-0.5176	0.4304
$-Y_{mn}$	-0.5176	0.4304	0.1491	-0.0696	0.1491	-0.0696	0.5176	-0.4304	-0.1491	0.0696	-0.1491	0.0696
	0.1491	-0.0696	-0.5176	0.4304	0.1491	-0.0696	-0.1491	0.0696	0.5176	-0.4304	-0.1491	0.0696
	0.1491	-0.0696	0.1491	-0.0696	-0.5176	0.4304	-0.1491	0.0696	-0.1491	0.0696	0.5176	-0.4304
	Load											
Y_n	0.002625	0	0	0	0	0	-0.00263	0				
	0	0	0.002625	0	0	0	-0.00263	0				
	0	0	0	0	0.002625	0	-0.00263	0				
	-0.00263	0	-0.00263	0	-0.00263	0	0.007877	0				

$-Y_k$ Y_k $-Y_{km}$ Y_{km} $-Y_{mn}$ Y_{mn}

Why the Vsource has 6 rows, while the Load has 4 rows? In OpenDSS, a 3-phase Vsource is a two-terminal object, and is defined as a voltage source behind an impedance, so it has 6 nodes. In contrast, a 3-phase wye-connected load only has one terminal, and only has 4 nodes, A, B, C and neutral.

Overall Structure of Solving Power Flow

Put all elements' Yprim together:

	Vsource												
Y_k	0.2	0	0	0	0	0	-0.2	0	0	0	0	0	$-Y_k$
	0	0	0.2	0	0	0	0	0	-0.2	0	0	0	
	0	0	0	0	0.2	0	0	0	0	0	-0.2	0	
$-Y_k$	-0.2	0	0	0	0	0	0.2	0	0	0	0	0	Y_k
	0	0	-0.2	0	0	0	0	0	0.2	0	0	0	
	0	0	0	0	-0.2	0	0	0	0	0	0.2	0	
	Line 1												
Y_{km}	0.4314	-0.7738	-0.1235	0.1881	-0.1235	0.1881	-0.4314	0.7738	0.1235	-0.1881	0.1235	-0.1881	$-Y_{km}$
	-0.1235	0.1881	0.4314	-0.7738	-0.1235	0.1881	0.1235	-0.1881	-0.4314	0.7738	0.1235	-0.1881	
	-0.1235	0.1881	-0.1235	0.1881	0.4314	-0.7738	0.1235	-0.1881	0.1235	-0.1881	-0.4314	0.7738	
$-Y_{km}$	-0.4314	0.7738	0.1235	-0.1881	0.1235	-0.1881	0.4314	-0.7738	-0.1235	0.1881	-0.1235	0.1881	Y_{km}
	0.1235	-0.1881	-0.4314	0.7738	0.1235	-0.1881	-0.1235	0.1881	0.4314	-0.7738	-0.1235	0.1881	
	0.1235	-0.1881	0.1235	-0.1881	-0.4314	0.7738	-0.1235	0.1881	-0.1235	0.1881	0.4314	-0.7738	
	Line 2												
Y_{mn}	0.5176	-0.4304	-0.1491	0.0696	-0.1491	0.0696	-0.5176	0.4304	0.1491	-0.0696	0.1491	-0.0696	$-Y_{mn}$
	-0.1491	0.0696	0.5176	-0.4304	-0.1491	0.0696	0.1491	-0.0696	-0.5176	0.4304	0.1491	-0.0696	
	-0.1491	0.0696	-0.1491	0.0696	0.5176	-0.4304	0.1491	-0.0696	0.1491	-0.0696	-0.5176	0.4304	
$-Y_{mn}$	-0.5176	0.4304	0.1491	-0.0696	0.1491	-0.0696	0.5176	-0.4304	-0.1491	0.0696	-0.1491	0.0696	Y_{mn}
	0.1491	-0.0696	-0.5176	0.4304	0.1491	-0.0696	-0.1491	0.0696	0.5176	-0.4304	-0.1491	0.0696	
	0.1491	-0.0696	0.1491	-0.0696	-0.5176	0.4304	-0.1491	0.0696	-0.1491	0.0696	0.5176	-0.4304	
	Load												
Y_n	0.002625	0	0	0	0	0	-0.00263	0					
	0	0	0.002625	0	0	0	-0.00263	0					
	0	0	0	0	0.002625	0	-0.00263	0					
	-0.00263	0	-0.00263	0	-0.00263	0	0.007877	0					

$Y_k + Y_{km}$ ⇩ Constructing **Y** $-Y_{km}$ $Y_{km} + Y_{mn}$ $-Y_{mn}$

	K.1		K.2		K.3		M.1		M.2		M.3		N.1		N.2		N.3	
K.1	0.6314	-0.7738	-0.1235	0.1881	-0.1235	0.1881	-0.4314	0.7738	0.1235	-0.1881	0.1235	-0.1881	0	0	0	0	0	0
K.2	-0.1235	0.1881	0.6314	-0.7738	-0.1235	0.1881	0.1235	-0.1881	-0.4314	0.7738	0.1235	-0.1881	0	0	0	0	0	0
K.3	-0.1235	0.1881	-0.1235	0.1881	0.6314	-0.7738	0.1235	-0.1881	0.1235	-0.1881	-0.4314	0.7738	0	0	0	0	0	0
M.1	-0.4314	0.7738	0.1235	-0.1881	0.1235	-0.1881	0.9490	-1.2042	-0.2726	0.2577	-0.2726	0.2577	-0.5176	0.4304	0.1491	-0.0696	0.1491	-0.0696
M.2	0.1235	-0.1881	-0.4314	0.7738	0.1235	-0.1881	-0.2726	0.2577	0.9490	-1.2042	-0.2726	0.2577	0.1491	-0.0696	-0.5176	0.4304	0.1491	-0.0696
M.3	0.1235	-0.1881	0.1235	-0.1881	-0.4314	0.7738	-0.2726	0.2577	-0.2726	0.2577	0.9490	-1.2042	0.1491	-0.0696	0.1491	-0.0696	-0.5176	0.4304
N.1	0	0	0	0	0	0	-0.5176	0.4304	0.1491	-0.0696	0.1491	-0.0696	0.5202	-0.4304	-0.1491	0.0696	-0.1491	0.0696
N.2	0	0	0	0	0	0	0.1491	-0.0696	-0.5176	0.4304	0.1491	-0.0696	-0.1491	0.0696	0.5202	-0.4304	-0.1491	0.0696
N.3	0	0	0	0	0	0	0.1491	-0.0696	0.1491	-0.0696	-0.5176	0.4304	-0.1491	0.0696	-0.1491	0.0696	0.5202	-0.4304

$-Y_{mn}$ $Y_n + Y_{mn}$

Overall Structure of Solving Power Flow

How to solve power flow?

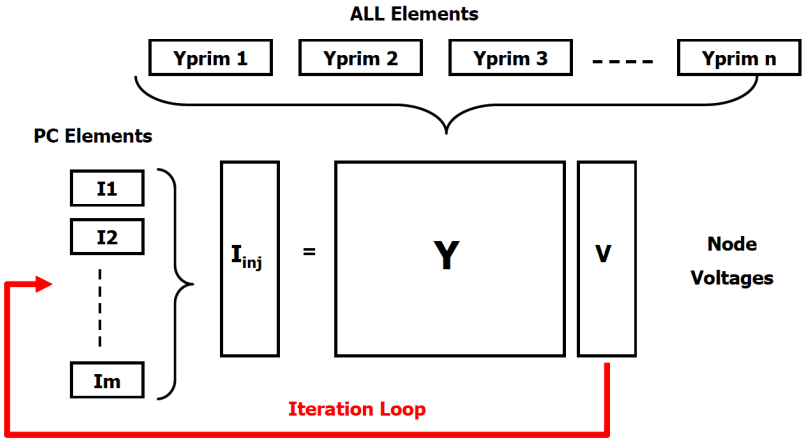


Figure 17. OpenDSS Solution Loop

A general outline of solution of the power flow:

- Step 1: Initialization.
  - Calculate the injection currents of the Vsource: $\boldsymbol{I}_{source} = \boldsymbol{Y}_{source} * \boldsymbol{V}_{source}$
  - Calculate the initial nodal voltages using $\boldsymbol{I}_{source}$ with all other injections currents set to 0: $\mathbf{V}_0 = \mathbf{Y}^{-1} * \mathbf{I}_{\text{inj},0}$
- Step 2:
  Calculate the injection (compensation) currents for all the power conversion (PC) elements and adding them into the appropriate slot in the current vector to obtain $\mathbf{I}_{inj,n}$, where, n denotes the n'th iteration.
- Step 3:
  Update voltage vector: $\mathbf{V}_{n+1} = \mathbf{Y}^{-1} * \mathbf{I}_{\text{inj},n}$
- Step 4:

  Check whether the voltage error exceeds the tolerance. If not, go to Step 2.

Note, regarding the injection currents:
1. $\boldsymbol{I}_{source}$ does not change in each iteration;
2. The injection currents for a non-constant-Z load will change in each iteration.

Overall Structure of Solving Power Flow

$\mathbf{Y}^{-1}=$

																R ↓	X ↓
4.9355	0.0004	0.0001	0.0002	0.0001	0.0002	4.9274	-0.0152	-0.0022	-0.0053	-0.0022	-0.0053	4.9119	-0.0307	-0.0052	-0.0116	-0.0052	-0.0116
0.0001	0.0002	4.9355	0.0004	0.0001	0.0002	-0.0022	-0.0053	4.9274	-0.0152	-0.0022	-0.0053	-0.0052	-0.0116	4.9119	-0.0307	-0.0052	-0.0116
0.0001	0.0002	0.0001	0.0002	4.9355	0.0004	-0.0022	-0.0053	-0.0022	-0.0053	4.9274	-0.0152	-0.0052	-0.0116	-0.0052	-0.0116	4.9119	-0.0307
4.9274	-0.0152	-0.0022	-0.0053	-0.0022	-0.0053	5.5428	1.1746	0.168	0.4164	0.168	0.4164	5.5301	1.1525	0.1674	0.4063	0.1674	0.4063
-0.0022	-0.0053	4.9274	-0.0152	-0.0022	-0.0053	0.168	0.4164	5.5428	1.1746	0.168	0.4164	0.1674	0.4063	5.5301	1.1525	0.1674	0.4063
-0.0022	-0.0053	-0.0022	-0.0053	4.9274	-0.0152	0.168	0.4164	0.168	0.4164	5.5428	1.1746	0.1674	0.4063	0.1674	0.4063	5.5301	1.1525
4.9119	-0.0307	-0.0052	-0.0116	-0.0052	-0.0116	5.5301	1.1525	0.1674	0.4063	0.1674	0.4063	6.7083	2.3318	0.3988	0.8812	0.3988	0.8812
-0.0052	-0.0116	4.9119	-0.0307	-0.0052	-0.0116	0.1674	0.4063	5.5301	1.1525	0.1674	0.4063	0.3988	0.8812	6.7083	2.3318	0.3988	0.8812
-0.0052	-0.0116	-0.0052	-0.0116	4.9119	-0.0307	0.1674	0.4063	0.1674	0.4063	5.5301	1.1525	0.3988	0.8812	0.3988	0.8812	6.7083	2.3318

$$\boldsymbol{I}_{source} = \begin{bmatrix} I_{s1} \\ I_{s2} \\ I_{s3} \end{bmatrix} = \boldsymbol{Y}_{source} * \boldsymbol{V}_{source} = \begin{bmatrix} 1625.36 + j0 \\ -812.68 - j1407.6 \\ -812.68 + j1407.6 \end{bmatrix} A$$

$$\mathbf{I}_{\text{inj},0} = \begin{bmatrix} I_{k1,0}^{in} \\ I_{k2,0}^{in} \\ I_{k3,0}^{in} \\ I_{m1,0}^{in} \\ I_{m2,0}^{in} \\ I_{m3,0}^{in} \\ I_{n1,0}^{in} \\ I_{n2,0}^{in} \\ I_{n3,0}^{in} \end{bmatrix} = \begin{bmatrix} I_{s1} \\ I_{s2} \\ I_{s3} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1625.36 + j0 \\ -812.68 - j1407.6 \\ -812.68 + j1407.6 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Overall Structure of Solving Power Flow

Matlab code for solving power flow using ***OpenDSS's*** method:

```
clear;

j = sqrt(-1);

%% Compute the Y matrix of each element
% Vsource
Z_k = [5+j*0, 0, 0; 0, 5+j*0, 0; 0, 0, 5+j*0];
Y_k = inv(Z_k);

% line 1
Rmatrix_1 = [0.62, 0.17, 0.17; 0.17, 0.62, 0.17; 0.17, 0.17, 0.62];
Xmatrix_1 = [1.21, 0.43, 0.43; 0.43, 1.21, 0.43; 0.43, 0.43, 1.21];

Z_km = Rmatrix_1 * 1 + j * Xmatrix_1 * 1;
Y_km = inv(Z_km);

% line 2
Rmatrix_2 = [1.19, 0.23, 0.23; 0.23, 1.19, 0.23; 0.23, 0.23, 1.19];
Xmatrix_2 = [1.21, 0.49, 0.49; 0.49, 1.21, 0.49; 0.49, 0.49, 1.21];
Z_mn = Rmatrix_2 * 1 + j * Xmatrix_2 * 1;
Y_mn = inv(Z_mn);

% load
S_load = 500 + j * 0;
S_n = [S_load/3; S_load/3; S_load/3];
V_load_mag = 13.8;
Z_LN = 1000*V_load_mag^2/conj(S_load);
Z_n = [Z_LN, 0, 0; 0, Z_LN, 0; 0, 0, Z_LN];
Y_n = inv(Z_n);

% system Y matrix
Y = [Y_k + Y_km, -Y_km, zeros(3,3); ...
     -Y_km, Y_km + Y_mn, -Y_mn; ...
     zeros(3,3), -Y_mn, Y_mn + Y_n];

%% initialization
I_source = [(1.02*13800/3^0.5)/5; (1.02*13800/3^0.5)/5 * (-1/2-j*3^0.5/2); (1.02*13800/3^0.5)/5 * (-1/2+j*3^0.5/2)];
I_0 = zeros(9, 1);
I_0(1:3, :) = I_source;

V_0 = Y\I_0;  % For a constant impedance load, we do not have to perform iterations
```

A constant-Z load does not need a compensation current. This is the final voltage vector.

Overall Structure of Solving Power Flow

Matlab code for solving power flow using ***Forward*** method:

```
clear;

j = sqrt(-1);
%% Compute the Z matrix of each element
% Vsource
Z_k = [5+j*0, 0, 0; 0, 5+j*0, 0; 0, 0, 5+j*0];

% line 1
Rmatrix_1 = [0.62, 0.17, 0.17; 0.17, 0.62, 0.17; 0.17, 0.17, 0.62];
Xmatrix_1 = [1.21, 0.43, 0.43; 0.43, 1.21, 0.43; 0.43, 0.43, 1.21];
Z_km = Rmatrix_1 * 1 + j * Xmatrix_1 * 1;

% line 2
Rmatrix_2 = [1.19, 0.23, 0.23; 0.23, 1.19, 0.23; 0.23, 0.23, 1.19];
Xmatrix_2 = [1.21, 0.49, 0.49; 0.49, 1.21, 0.49; 0.49, 0.49, 1.21];
Z_mn = Rmatrix_2 * 1 + j * Xmatrix_2 * 1;

% load
S_load = 500 + j * 0;
S_n = [S_load/3; S_load/3; S_load/3];
V_load_mag = 13.8;
Z_LN = 1000*V_load_mag^2/conj(S_load);
Z_n = [Z_LN, 0, 0; 0, Z_LN, 0; 0, 0, Z_LN];

%% Vsource & Z_total
V0 = 1.02*13800/(3^0.5);
Vnom = [V0;V0*exp(j*(-2*pi/3));V0*exp(j*(2*pi/3))];
Z_total = [Z_k+Z_km+Z_mn+Z_n];

%% Load Current
I = inv(Z_total)*Vnom;

%% Bus Voltage
Vk = Vnom - Z_k*I;   % Forward
Vm = Vk - Z_km*I;    % Forward
Vn = Vm - Z_mn*I;    % Forward
```

Overall Structure of Solving Power Flow

Result Comparison:

Node	Real	Imaginary
Vk1	8021.87	0.41
Vk2	-4010.58	-6947.34
Vk3	-4011.28	6946.94
Vm1	8012.36	-15.92
Vm2	-4019.97	-6930.94
Vm3	-3992.39	6946.87
Vn1	7992.16	-30.95
Vn2	-4022.89	-6905.93
Vn3	-3969.27	6936.89

Computed Voltages (volts) by Matlab Code Using OpenDSS's Method

Node	Real	Imaginary
Vk1	8021.87	0.41
Vk2	-4010.58	-6947.34
Vk3	-4011.28	6946.94
Vm1	8012.36	-15.92
Vm2	-4019.97	-6930.94
Vm3	-3992.39	6946.87
Vn1	8012.36	-15.92
Vn2	-4019.97	-6930.94
Vn3	-3992.39	6946.87

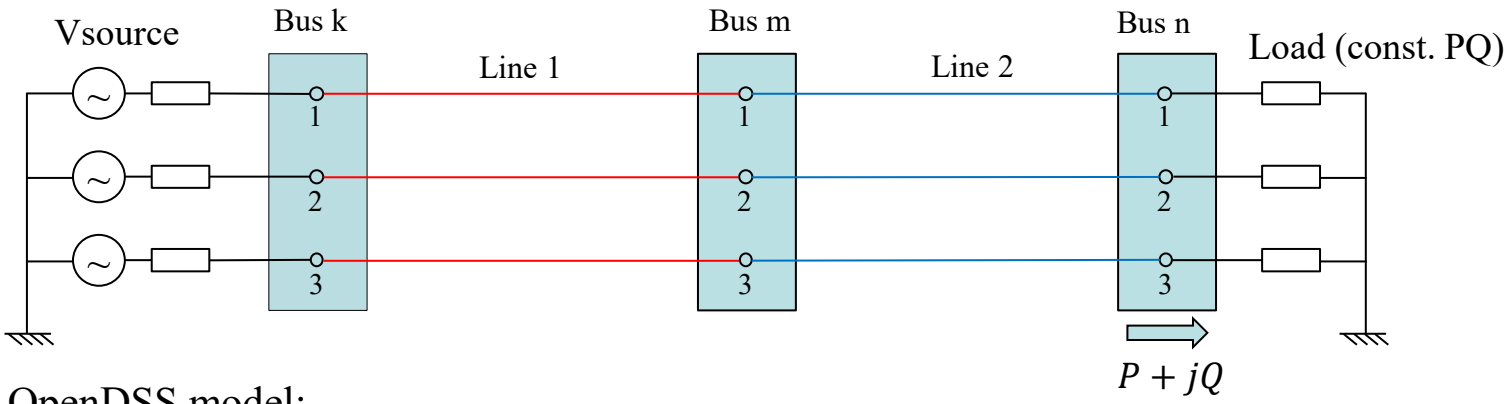
Computed Voltages (volts) by Matlab Code Using Forward Method

Node	Real	Imaginary
Vk1	8021.87	0.00
Vk2	-4010.94	-6947.14
Vk3	-4010.94	6947.14
Vm1	8012.38	-15.38
Vm2	-4019.51	-6931.23
Vm3	-3992.87	6946.61
Vn1	7992.17	-30.69
Vn2	-4022.66	-6906.08
Vn3	-3969.51	6936.77

Voltages (volts) Exported from OpenDSS

Overall Structure of Solving Power Flow

Example 2 (Constant-PQ Load):

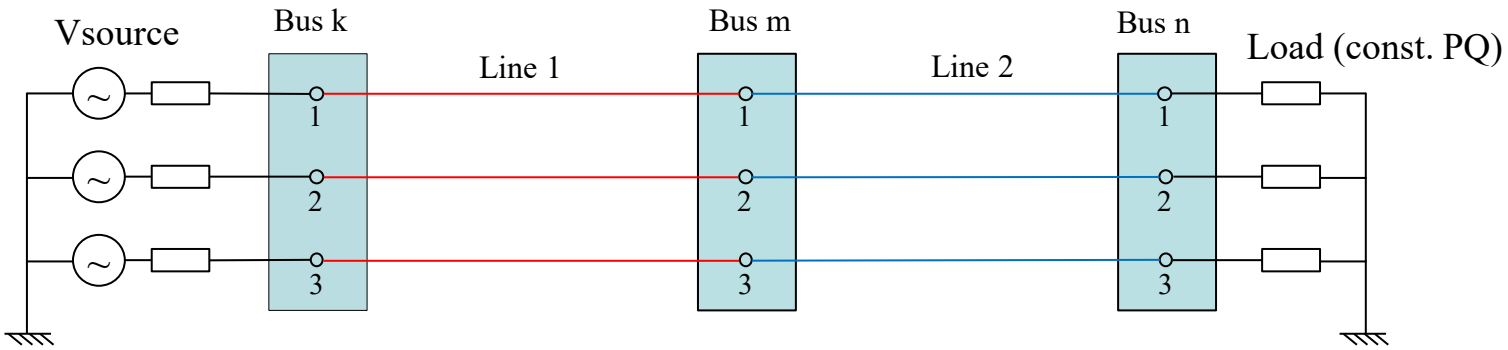


OpenDSS model:

```
Clear
//-------------- Define a new circuit --------------------//
new circuit.experiment

//---------------Define a Voltage source------------------//
Edit "Vsource.source" phases=3 basekv=13.8 Pu=1.02 bus1=K R1=5 X1=0 R0=5 X0=0

//----------------- Define line codes  --------------------//
New LineCode.Code1 nphases= 3 Units= mi
 ~ Rmatrix= (0.62 | 0.17 0.62 | 0.17 0.17 0.62 )
 ~ Xmatrix= (1.21 | 0.43 1.21 | 0.43 0.43 1.21 )

New LineCode.Code2 nphases= 3 Units= mi
 ~ Rmatrix= (1.19 | 0.23 1.19 | 0.23 0.23 1.19 )
 ~ Xmatrix= (1.21 | 0.49 1.21 | 0.49 0.49 1.21 )
… (continued)
```

Overall Structure of Solving Power Flow

Example 2:



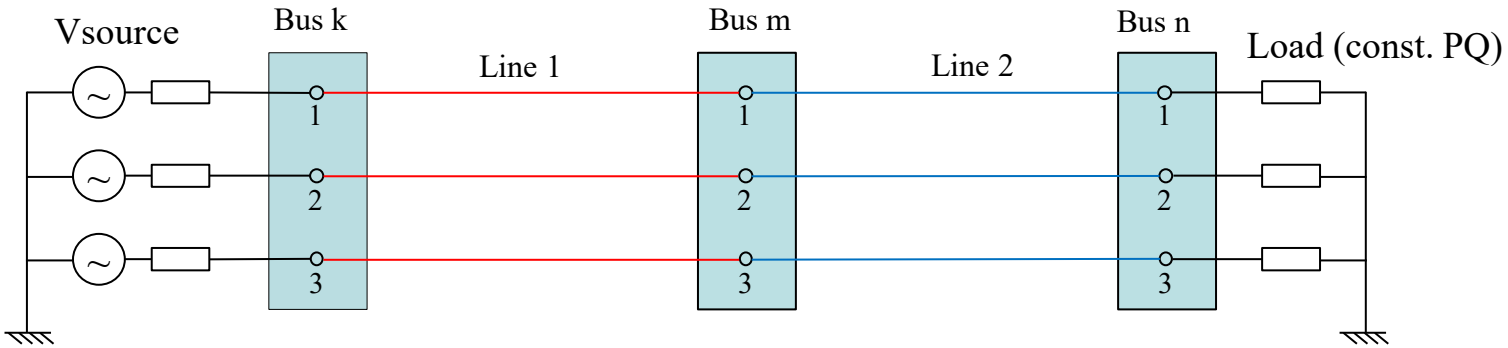
OpenDSS model:

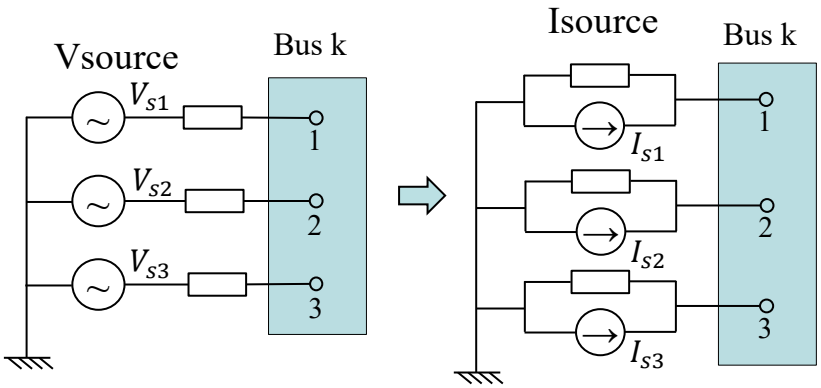
```
… (continued)
//---------------  Define line 1 and 2  ------------------//
New Line.1 phases=3  Bus1=k.1.2.3 Bus2=m.1.2.3  length=1 units=mi LineCode=Code1
New Line.2 phases=3  Bus1=m.1.2.3 Bus2=n.1.2.3  length=1 units=mi LineCode=Code2

//----------------  Define a load   ----------------------//
New Load.L Bus1=n.1.2.3.0       Phases=3 Conn=wye Model=1 kV=13.8   kW=500 kvar=500

Set VoltageBases = "13.8"
Solve
```

Constant-PQ load

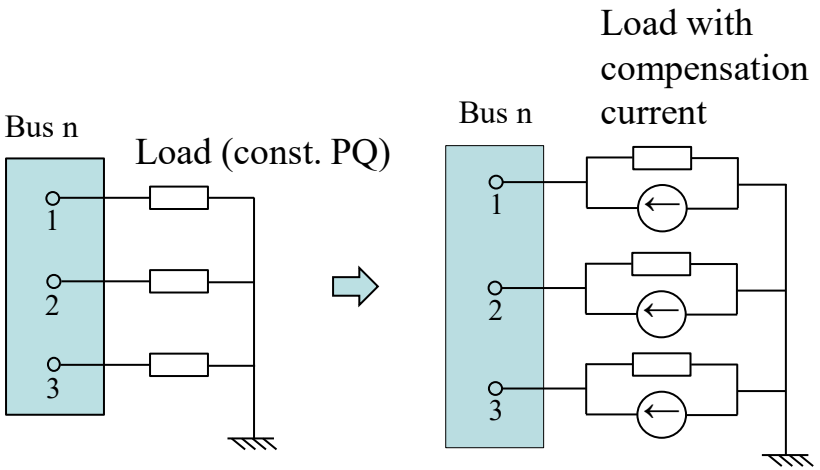
Overall Structure of Solving Power Flow

Example 2 (Injection Current):





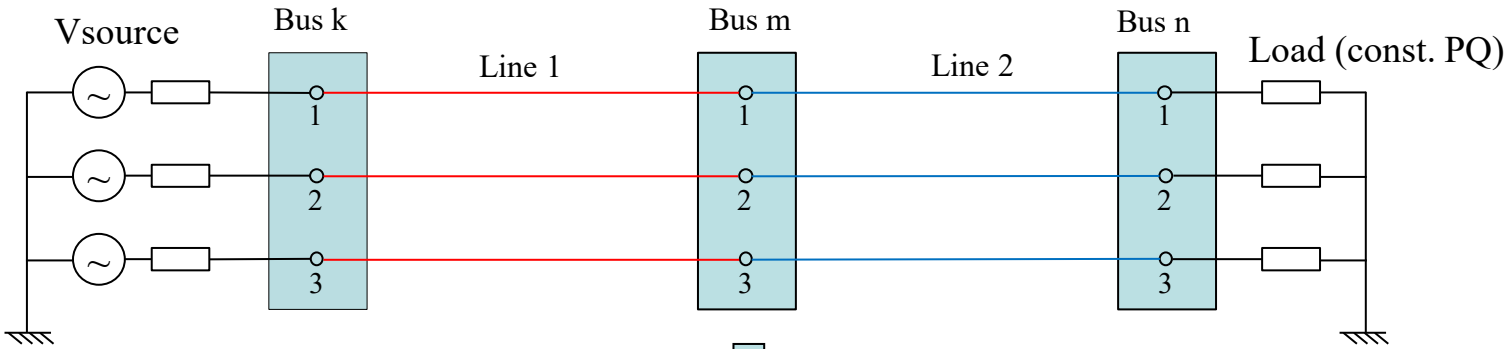
Thevinen equivalent to Norton equivalent

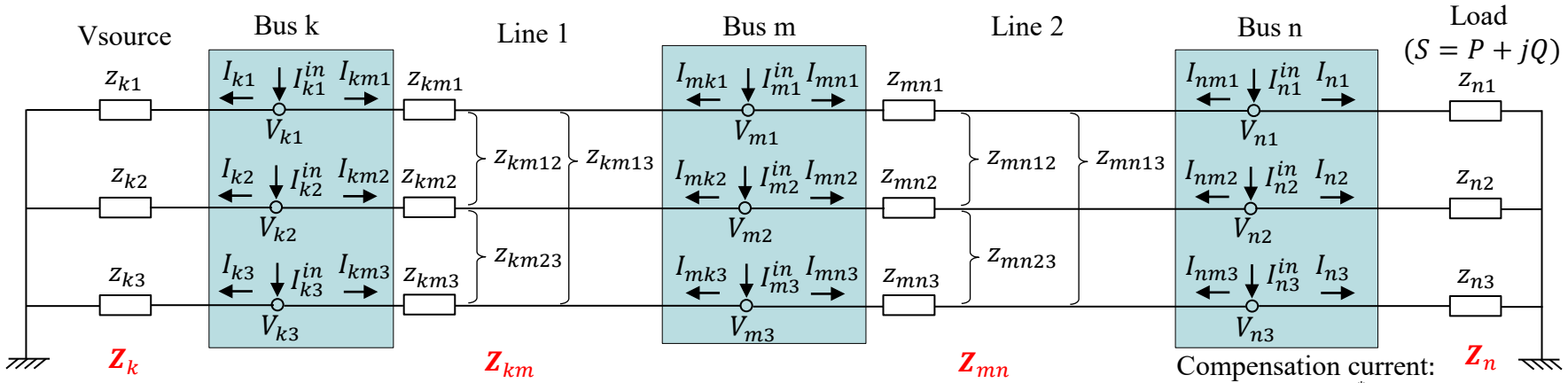


Various loads to a generic load model (Constant Z + compensation current)

Overall Structure of Solving Power Flow

Example 2 (Injection Current):





$$I_{k1}^{in} = I_{s1} = \frac{V_{s1}}{z_{k1}}, I_{k2}^{in} = I_{s2} = \frac{V_{s2}}{z_{k2}}, I_{k3}^{in} = I_{s3} = \frac{V_{s3}}{z_{k3}}$$

$$I_{m1}^{in} = I_{m2}^{in} = I_{m3}^{in} = 0$$

Compensation current:

$$I_{n1}^{in} = -\left[\left(\frac{S}{3V_{n1}}\right)^* - \frac{V_{n1}}{z_{n1}}\right],$$
$$I_{n2}^{in} = -\left[\left(\frac{S}{3V_{n2}}\right)^* - \frac{V_{n2}}{z_{n2}}\right],$$
$$I_{n3}^{in} = -\left[\left(\frac{S}{3V_{n3}}\right)^* - \frac{V_{n3}}{z_{n3}}\right]$$

Updated in each iteration

Overall Structure of Solving Power Flow

$\mathbf{Y}^{-1}=$

																R ↓	X ↓
4.9352	0.0634	0.0003	-0.0001	0.0003	-0.0001	4.9119	0.0556	-0.0073	-0.0035	-0.0073	-0.0035	4.8812	0.055	-0.0163	-0.0068	-0.0163	-0.0068
0.0003	-0.0001	4.9352	0.0634	0.0003	-0.0001	-0.0073	-0.0035	4.9119	0.0556	-0.0073	-0.0035	-0.0163	-0.0068	4.8812	0.055	-0.0163	-0.0068
0.0003	-0.0001	0.0003	-0.0001	4.9352	0.0634	-0.0073	-0.0035	-0.0073	-0.0035	4.9119	0.0556	-0.0163	-0.0068	-0.0163	-0.0068	4.8812	0.055
4.9119	0.0556	-0.0073	-0.0035	-0.0073	-0.0035	5.5076	1.2497	0.155	0.4179	0.155	0.4179	5.4732	1.2398	0.1446	0.4084	0.1446	0.4084
-0.0073	-0.0035	4.9119	0.0556	-0.0073	-0.0035	0.155	0.4179	5.5076	1.2497	0.155	0.4179	0.1446	0.4084	5.4732	1.2398	0.1446	0.4084
-0.0073	-0.0035	-0.0073	-0.0035	4.9119	0.0556	0.155	0.4179	0.155	0.4179	5.5076	1.2497	0.1446	0.4084	0.1446	0.4084	5.4732	1.2398
4.8812	0.055	-0.0163	-0.0068	-0.0163	-0.0068	5.4732	1.2398	0.1446	0.4084	0.1446	0.4084	6.6214	2.4303	0.3614	0.8818	0.3614	0.8818
-0.0163	-0.0068	4.8812	0.055	-0.0163	-0.0068	0.1446	0.4084	5.4732	1.2398	0.1446	0.4084	0.3614	0.8818	6.6214	2.4303	0.3614	0.8818
-0.0163	-0.0068	-0.0163	-0.0068	4.8812	0.055	0.1446	0.4084	0.1446	0.4084	5.4732	1.2398	0.3614	0.8818	0.3614	0.8818	6.6214	2.4303

This is different from the $\mathbf{Y}^{-1}$ in Example 1, due to a different load.

$$\boldsymbol{I}_{source} = \begin{bmatrix} I_{s1} \\ I_{s2} \\ I_{s3} \end{bmatrix} = \boldsymbol{Y}_{source} * \boldsymbol{V}_{source}$$

$$= \begin{bmatrix} 1625.36 + j0 \\ -812.68 - j1407.6 \\ -812.68 + j1407.6 \end{bmatrix} A$$

$$\mathbf{I}_{\text{inj},0} = \begin{bmatrix} I_{k1,0}^{in} \\ I_{k2,0}^{in} \\ I_{k3,0}^{in} \\ I_{m1,0}^{in} \\ I_{m2,0}^{in} \\ I_{m3,0}^{in} \\ I_{n1,0}^{in} \\ I_{n2,0}^{in} \\ I_{n3,0}^{in} \end{bmatrix} = \begin{bmatrix} I_{s1} \\ I_{s2} \\ I_{s3} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1625.36 + j0 \\ -812.68 - j1407.6 \\ -812.68 + j1407.6 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Overall Structure of Solving Power Flow

Matlab code for solving power flow using ***OpenDSS***'s method:

```
clear;

j = sqrt(-1);

%% compute the Y matrix of each element
% Vsource
Z_k = [5+j*0, 0, 0; 0, 5+j*0, 0; 0, 0, 5+j*0];
Y_k = inv(Z_k);

% line 1
Rmatrix_1 = [0.62, 0.17, 0.17; 0.17, 0.62, 0.17; 0.17, 0.17, 0.62];
Xmatrix_1 = [1.21, 0.43, 0.43; 0.43, 1.21, 0.43; 0.43, 0.43, 1.21];

Z_km = Rmatrix_1 * 1 + j * Xmatrix_1 * 1;
Y_km = inv(Z_km);

% line 2
Rmatrix_2 = [1.19, 0.23, 0.23; 0.23, 1.19, 0.23; 0.23, 0.23, 1.19];
Xmatrix_2 = [1.21, 0.49, 0.49; 0.49, 1.21, 0.49; 0.49, 0.49, 1.21];
Z_mn = Rmatrix_2 * 1 + j * Xmatrix_2 * 1;
Y_mn = inv(Z_mn);

% load
S_load = 500 + j * 500;
S_n = [S_load/3; S_load/3; S_load/3];
V_load_mag = 13.8;
Z_LN = 1000*V_load_mag^2/conj(S_load);
Z_n = [Z_LN, 0, 0; 0, Z_LN, 0; 0, 0, Z_LN];
Y_n = inv(Z_n);

… (continued)
```

Overall Structure of Solving Power Flow

Matlab code for solving power flow using ***OpenDSS***'s method:

```
…(continued)
% system Y matrix
Y = [Y_k + Y_km, -Y_km, zeros(3,3); ...
     -Y_km, Y_km + Y_mn, -Y_mn; ...
     zeros(3,3), -Y_mn, Y_mn + Y_n];

%% initialization
I_source = [(1.02*13800/3^0.5)/5; (1.02*13800/3^0.5)/5 * (-1/2-j*3^0.5/2); (1.02*13800/3^0.5)/5 * (-
1/2+j*3^0.5/2)];
I_0 = zeros(9, 1);
I_0(1:3, :) = I_source;

V_0 = Y\I_0;
V_i = V_0;
I_i = I_0;
V_last = V_0;

%% iteration
V_error = [];
for i = 1:10000
    I_load_inj =  - conj(1000*S_n ./ V_i(end-2:end, :)) + Y_n * V_i(end-2:end, :);
    I_i(end-2:end, :) = I_load_inj;
    V_i = Y\I_i;

    V_diff_iter = V_i - V_last;
    V_diff_iter_pu = max(abs(V_diff_iter))/(13800/3^0.5);
    V_last = V_i;
    V_error = [V_error; V_diff_iter_pu];

    if V_diff_iter_pu <= 10^(-15)
       break;
    end
end
```

Overall Structure of Solving Power Flow

Matlab code for solving power flow using ***Backward-Forward*** algorithm:

```
clear;
j = sqrt(-1);

%% Compute the Z matrix of each element
% Vsource
Z_k = [5+j*0, 0, 0; 0, 5+j*0, 0; 0, 0, 5+j*0];
% line 1
Rmatrix_1 = [0.62, 0.17, 0.17; 0.17, 0.62, 0.17; 0.17, 0.17, 0.62];
Xmatrix_1 = [1.21, 0.43, 0.43; 0.43, 1.21, 0.43; 0.43, 0.43, 1.21];
Z_km = Rmatrix_1 * 1 + j * Xmatrix_1 * 1;
% line 2
Rmatrix_2 = [1.19, 0.23, 0.23; 0.23, 1.19, 0.23; 0.23, 0.23, 1.19];
Xmatrix_2 = [1.21, 0.49, 0.49; 0.49, 1.21, 0.49; 0.49, 0.49, 1.21];
Z_mn = Rmatrix_2 * 1 + j * Xmatrix_2 * 1;
% load
S_load = 500 + j * 500;
S_n = [S_load/3; S_load/3; S_load/3];
V_load_mag = 13.8;
Z_LN = 1000*V_load_mag^2/conj(S_load);
Z_n = [Z_LN, 0, 0; 0, Z_LN, 0; 0, 0, Z_LN];


… (continued)
```

Overall Structure of Solving Power Flow

Matlab code for solving power flow using ***Backward-Forward*** algorithm:

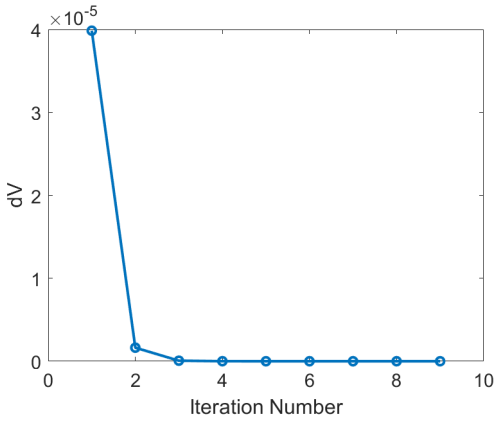
… (continued)

```
%% Initilization
I = zeros(3,1);
Vprevious = zeros(3,1);
V0 = 1.02*13800/(3^0.5);
Vnom = [V0; V0*exp(j*(-2*pi/3)); V0*exp(j*(2*pi/3))];
Vk = Vnom - Z_k*I;
Vm = Vk - Z_km*I;
Vn = Vm - Z_mn*I;
Verror = abs(abs(Vn) - abs(Vprevious))/(13800/3^0.5);
m = 0;



%% Iteration
ave_Verror = [];
while (max(Verror)>=10^(-15))
    Vk = Vnom - Z_k*I;   % Forward
    Vm = Vk - Z_km*I;    % Forward
    Vn = Vm - Z_mn*I;    % Forward
    I = conj(1000*S_n./Vn);     % Backward
    Verror = abs(abs(Vn)-abs(Vprevious))/(13800/3^0.5);
    Vprevious = Vn;
    m = m+1;
    ave_Verror = [ave_Verror, max(Verror)];
end
```

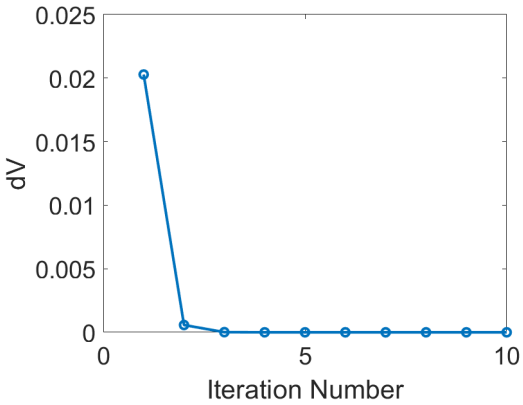
Overall Structure of Solving Power Flow

Results:

$$dV = \frac{||V_i| - |V_{i-1}||}{V_{nominal}}$$



Matlab Code Using OpenDSS's Method



Matlab Code Using Backward-Forward Method

Node	Real	Imaginary
Vk1	8020.78	103.35
Vk2	-3920.89	-6997.88
Vk3	-4099.90	6894.53
Vm1	7995.12	96.12
Vm2	-3914.32	-6972.04
Vm3	-4080.80	6875.92
Vn1	7959.89	100.70
Vn2	-3892.74	-6943.81
Vn3	-4067.15	6843.12

Computed Voltages (volts) by Matlab Code Using OpenDSS's Method

Node	Real	Imaginary
Vk1	8020.78	103.35
Vk2	-3920.89	-6997.88
Vk3	-4099.90	6894.53
Vm1	7995.12	96.12
Vm2	-3914.32	-6972.04
Vm3	-4080.80	6875.92
Vn1	7995.12	96.12
Vn2	-3914.32	-6972.04
Vn3	-4080.80	6875.92

Computed Voltages (volts) by Matlab Code Using Backward-Forward Method

Node	Real	Imaginary
Vk1	8020.78	103.60
Vk2	-3920.67	-6998.00
Vk3	-4100.11	6894.40
Vm1	7995.13	96.29
Vm2	-3914.18	-6972.13
Vm3	-4080.95	6875.84
Vn1	7959.91	100.03
Vn2	-3893.33	-6943.50
Vn3	-4066.59	6843.47

Voltages (volts) Exported from OpenDSS

Thank you!