OpenDSS Training Workshop

Session II – New Features/Updates and introduction to DSS-G

Davis Montenegro
Celso Rocha

Web – OpenDSS training
08/23/2022

Instructors

Davis Montenegro, *Senior Member, IEEE*

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

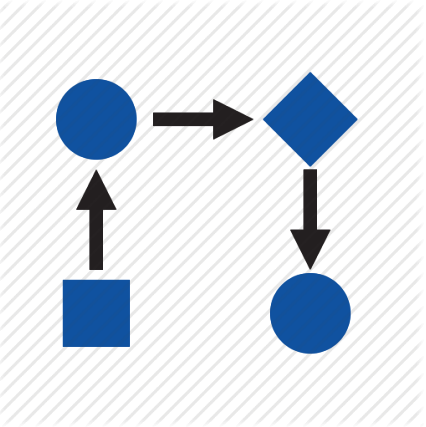
Before joining EPRI, Davis served for 10 years as a lecturer for Universidad Santo Tomas in Colombia, during this time he was also technology consultant in the areas of industrial automation, software and electronic hardware design focused on the electric power industry, specifically in monitoring and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

Instructors

Celso Rocha, Member, IEEE

Celso Rocha serves as Engineer Scientist II at the Electric Power Research Institute (EPRI) in Knoxville, Tennessee, USA. He holds the BSEE (2017) degree and the Master (2021) degree in Electrical Engineering with emphasis in energy and automation from University of Sao Paulo, Brazil. His work has been focused on Distribution Engineering, with a broad range of topics including DER integration, impacts and mitigation strategies assessments, active network management through optimization, DER modeling for QSTS, new planning methodologies for resiliency, and grid models generation, verification and validation from utility data repositories. He has 6 years of experience with OpenDSS, having taught several OpenDSS courses at conferences, universities and industry.

The evolution of OpenDSS into a parallel computing machine

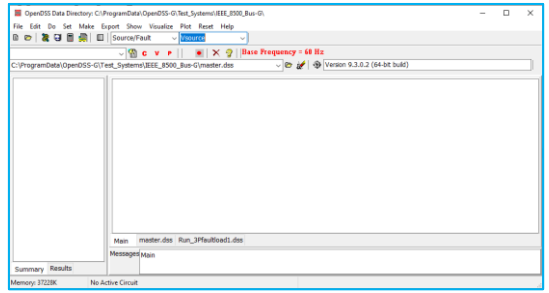


After being released in 2008 as open-source software OpenDSS has become widely used around the world. One of the features that makes OpenDSS popular is that the package offers interfaces for co-simulation.

Interfacing with OpenDSS

User Interfaces

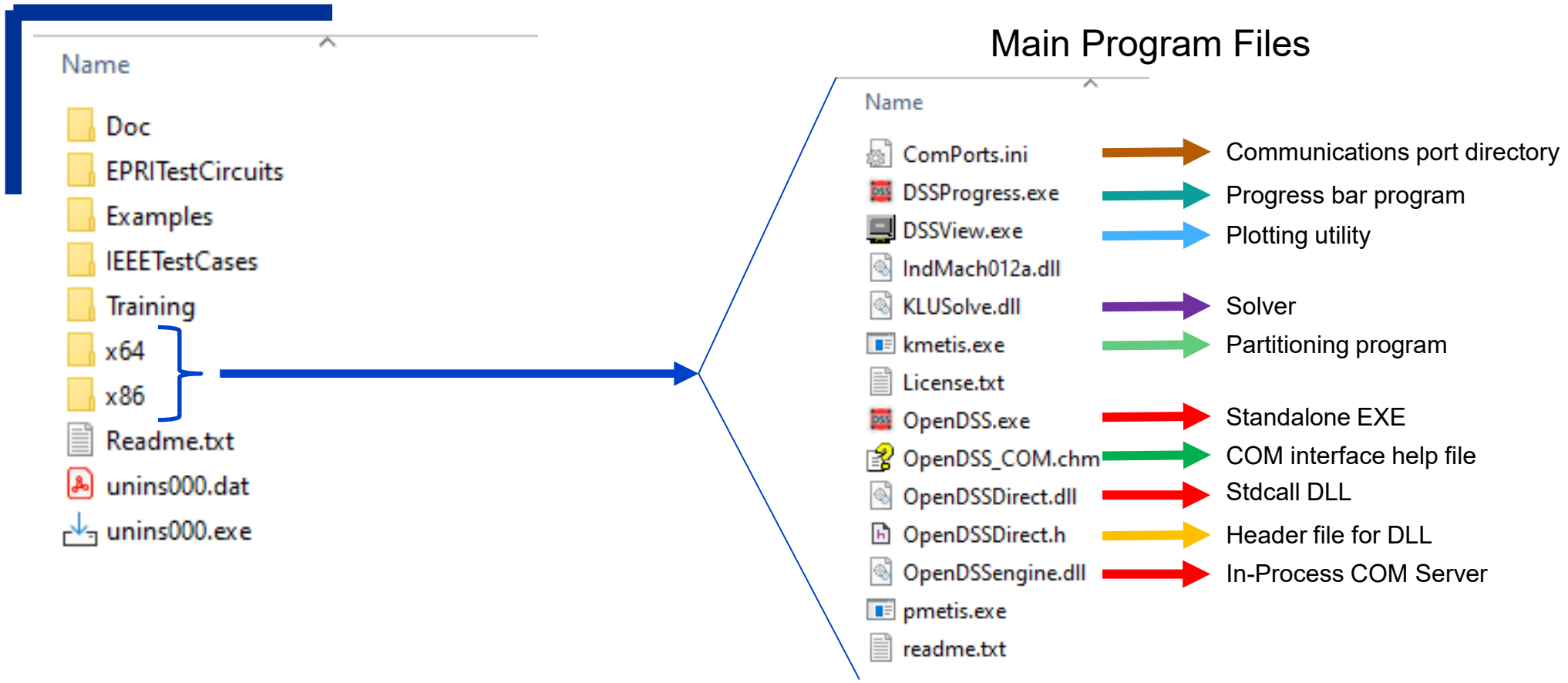
- A **stand-alone executable** program that provides a text-based interface (multiple windows)



- An **in-process COM server** (for Windows) that supports driving the simulator from user-written programs.

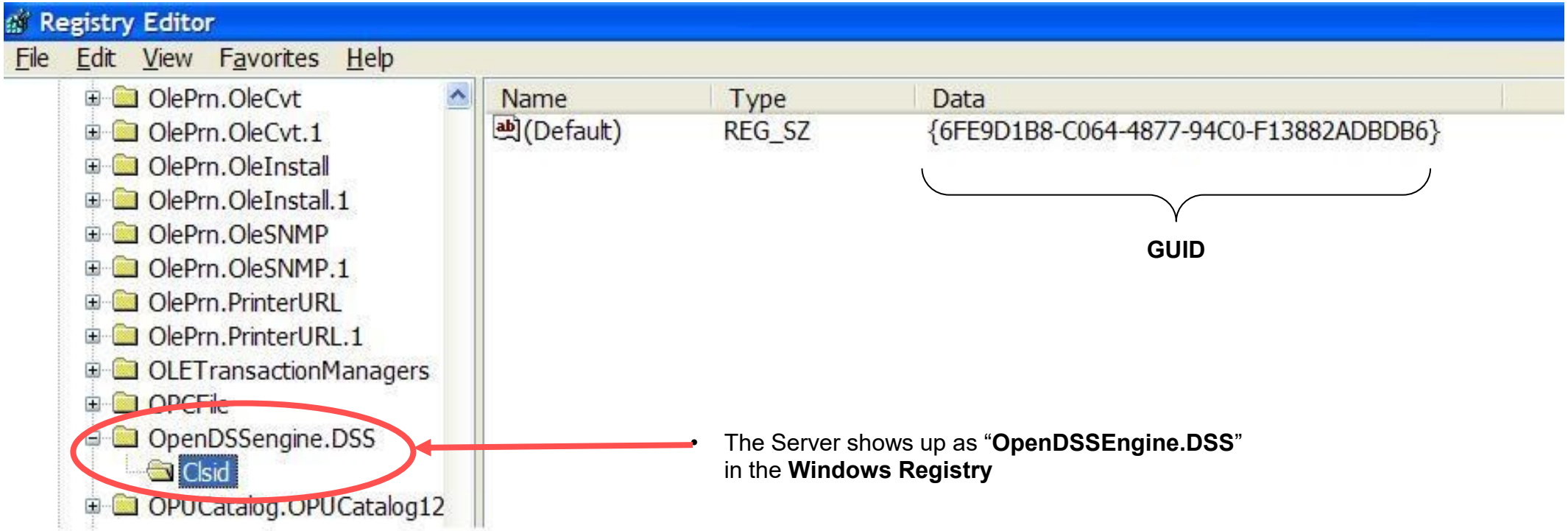
- A **direct DLL** interface that mimics the COM interface
  - For non-Windows platforms, such as HPCs
  - For programming languages that do not support COM or are not efficient at supporting COM

OpenDSS Files Installed



Registering the COM server

Windows Registry Entry

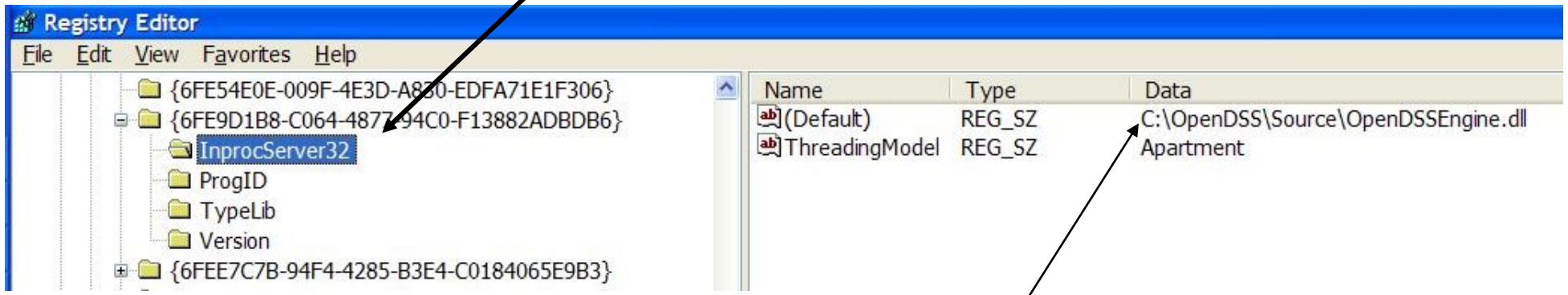


- The Server shows up as "**OpenDSSEngine.DSS**" in the **Windows Registry**

The OpenDSS is now available to any program on the computer

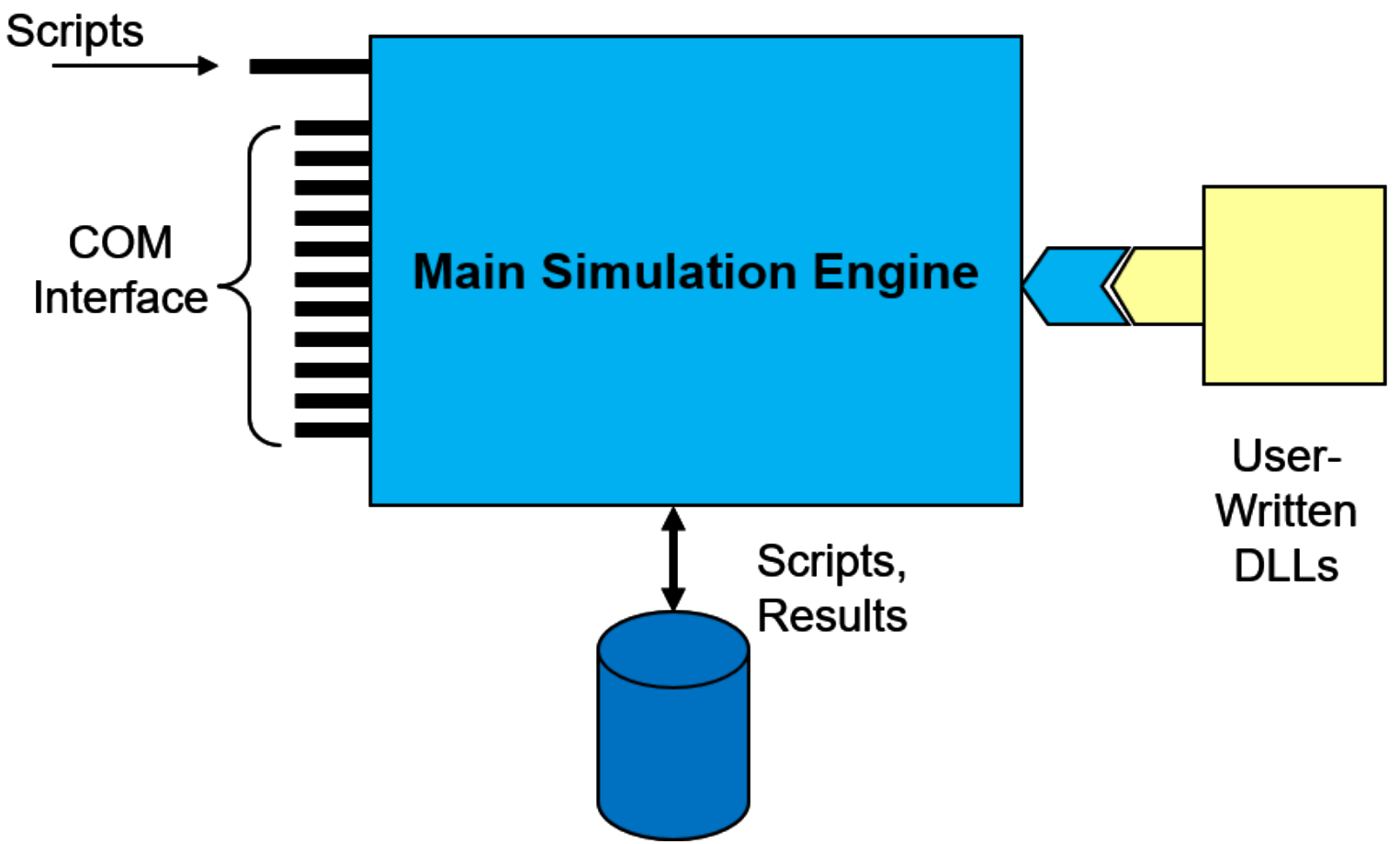
The GUID References the DLL File

If you look up the GUID in RegEdit

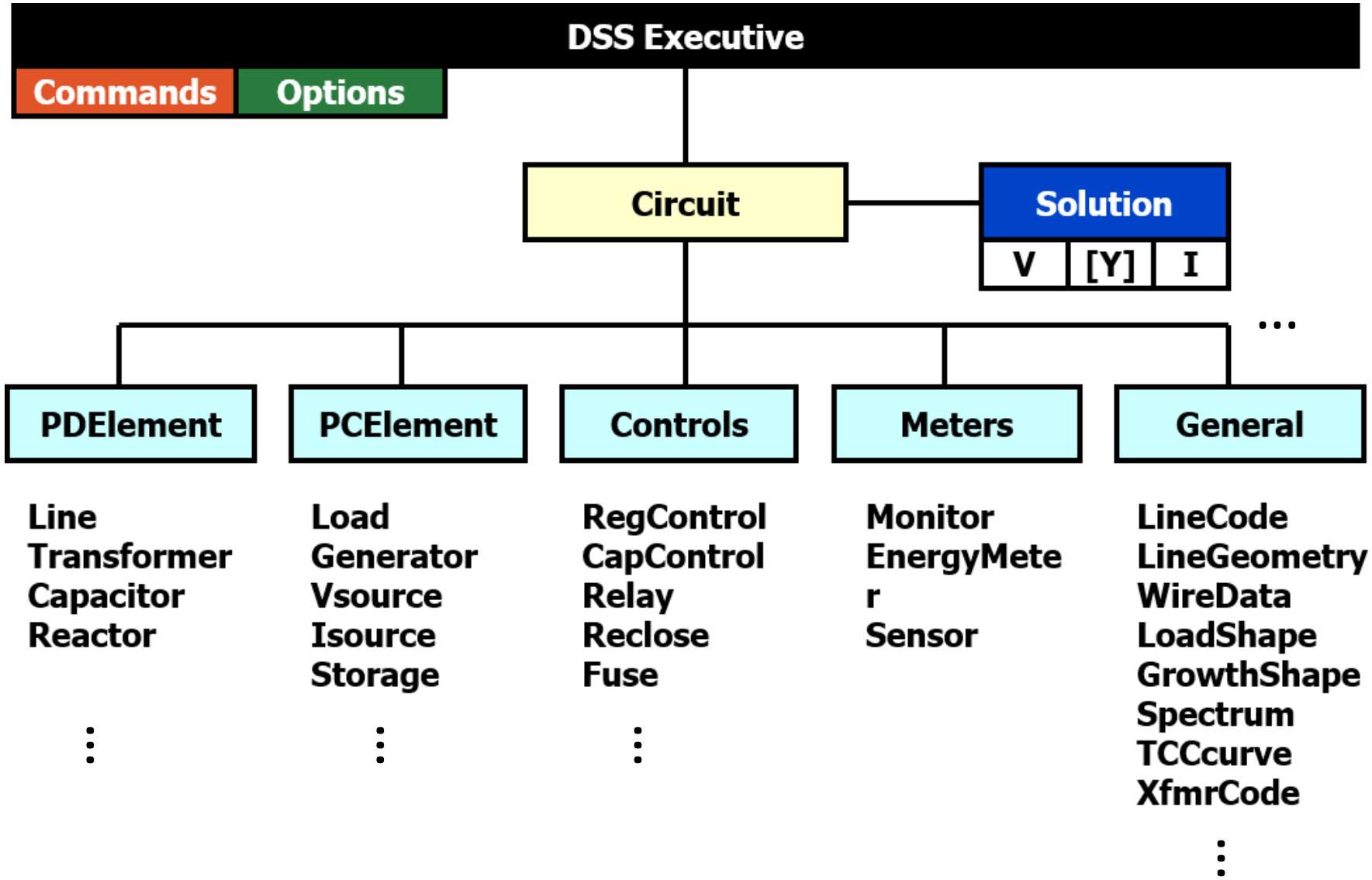


Points to OpenDSSEngine.DLL
(In-process server, Apartment Threading model)

DSS Structure

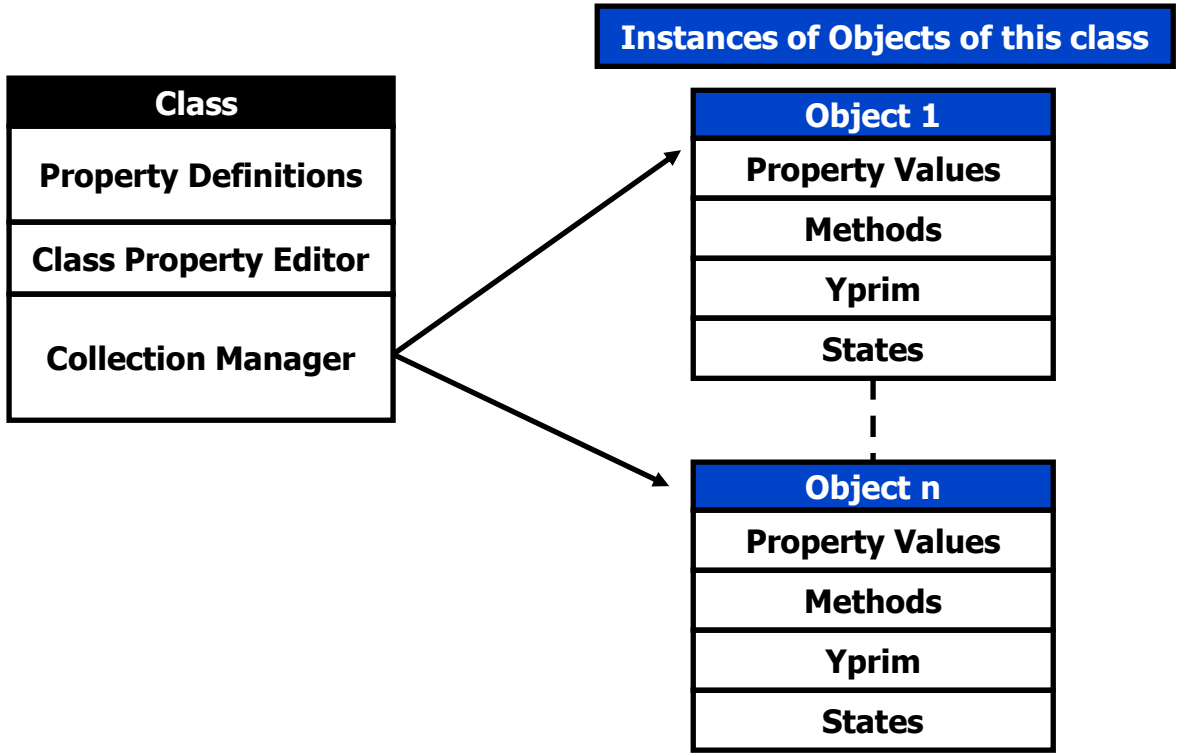


DSS Object Structure



Constantly growing

DSS Class Structure



- Each object has properties and methods
- Each property returns data according to the call type

How can I query about the available interfaces, properties and methods

User Interfaces

- Using a program language that depicts the interface for you (e.g. MS excel, VBA, etc.).

- Reading the documentation: https://sourceforge.net/p/electricdss/code/HEAD/tree/trunk/Version8/Distrib/Doc/

- Using the query tools available in your programming language:

  MATLAB : get, properties…

  Python: getattr, getAllAttributeNames…

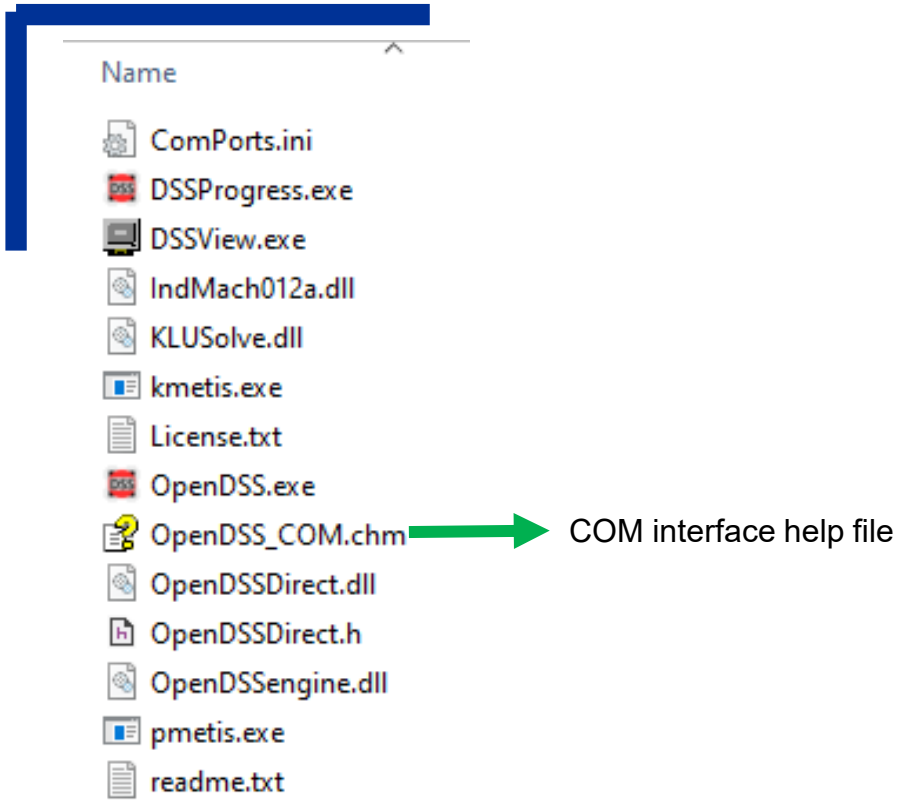
- If working with DirectDLL, then, you'll have to read the documentation and probably use the header file provided.

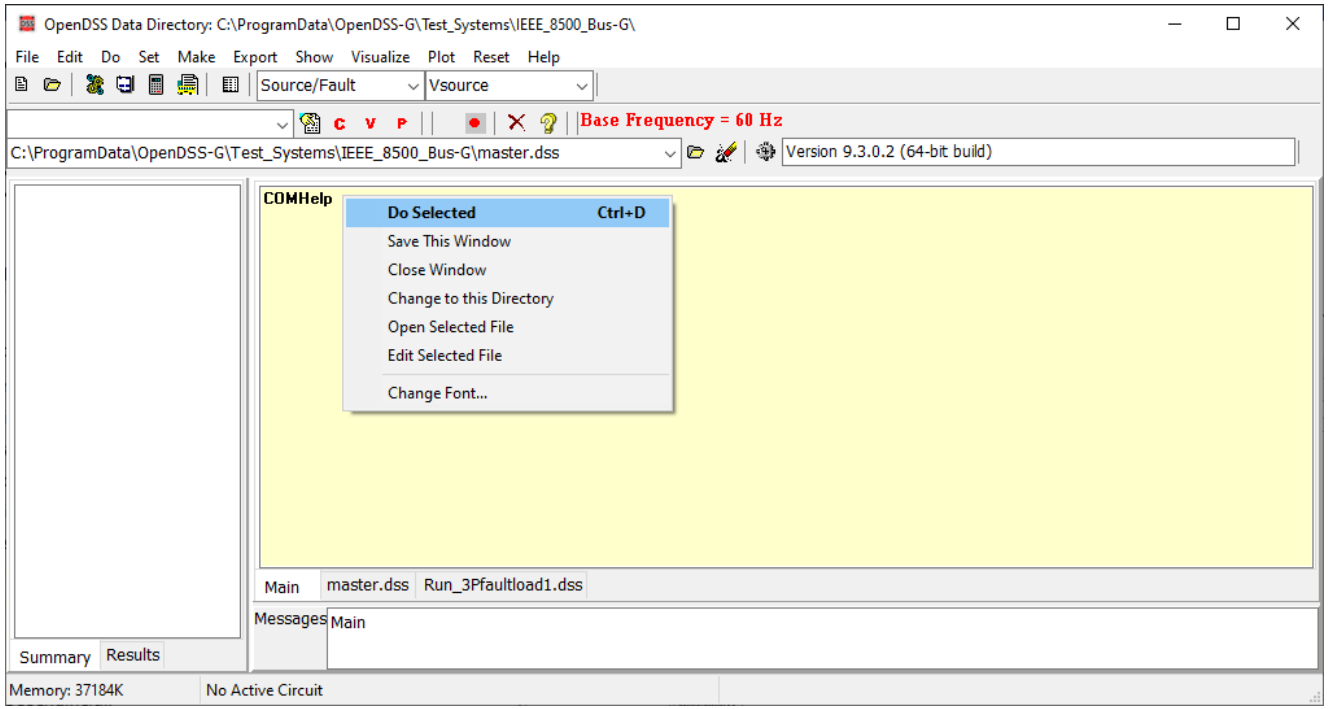
User Interfaces

Or…

User Interfaces

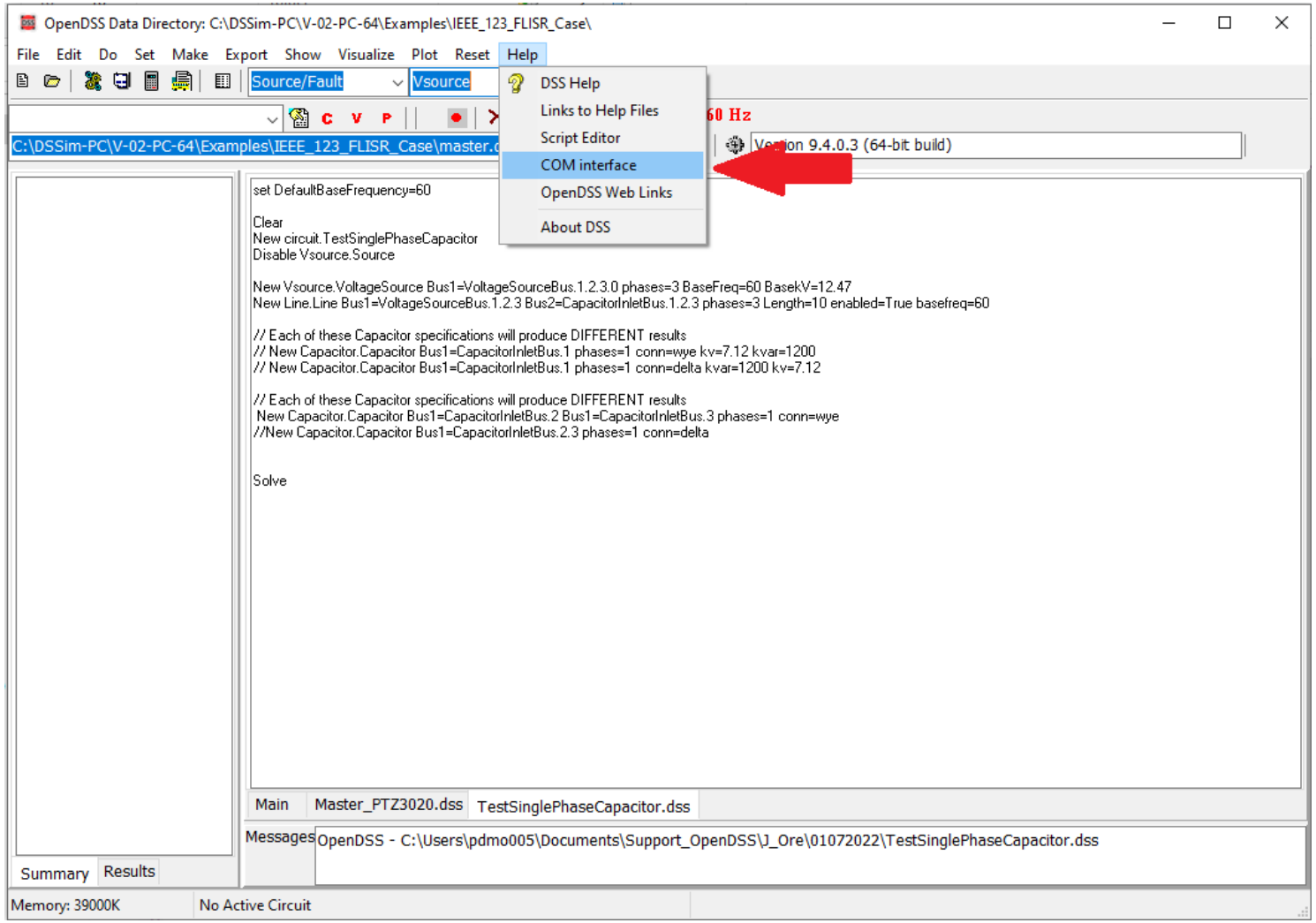
- Use the COM help file





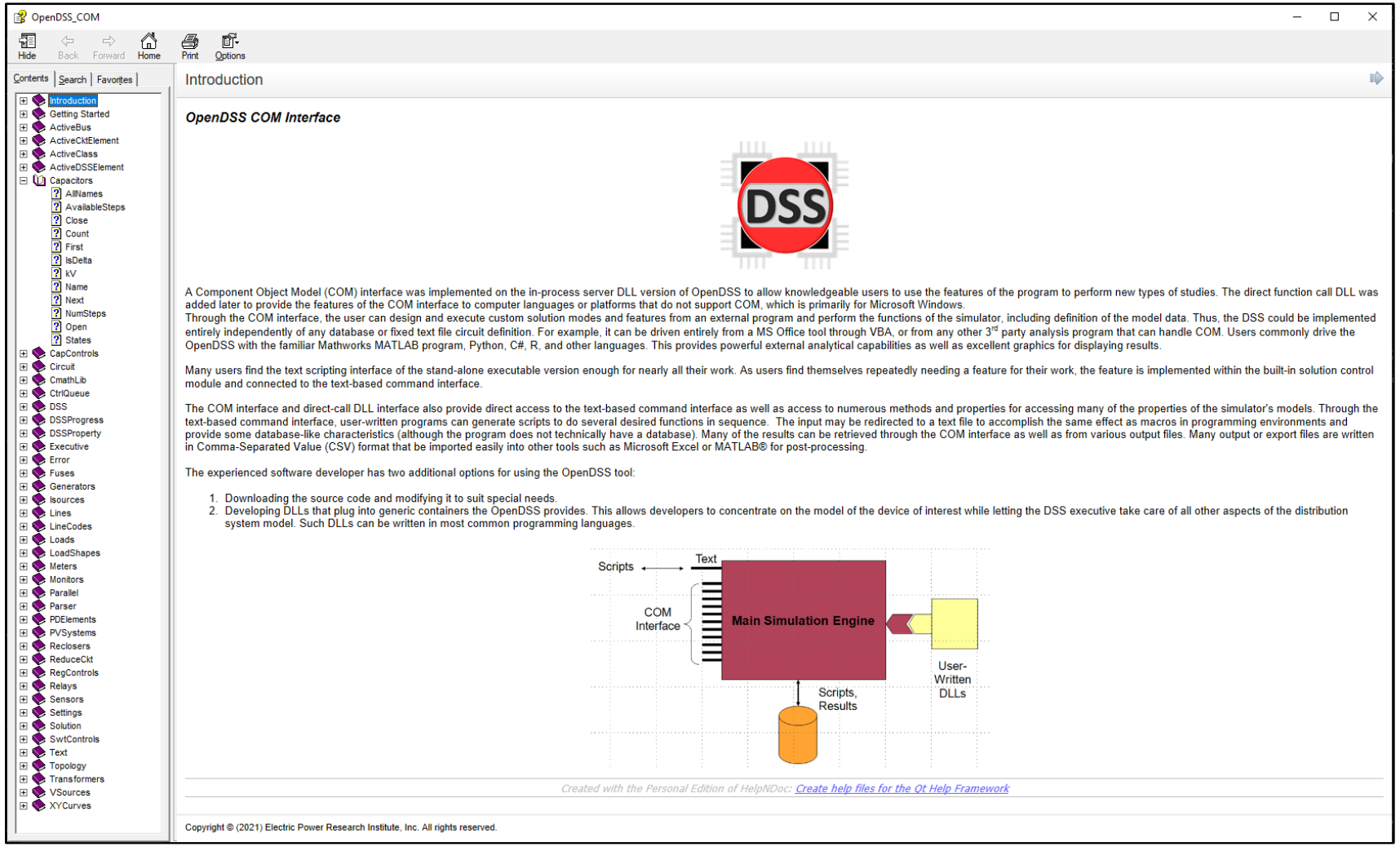
User Interfaces

- Use the COM help file



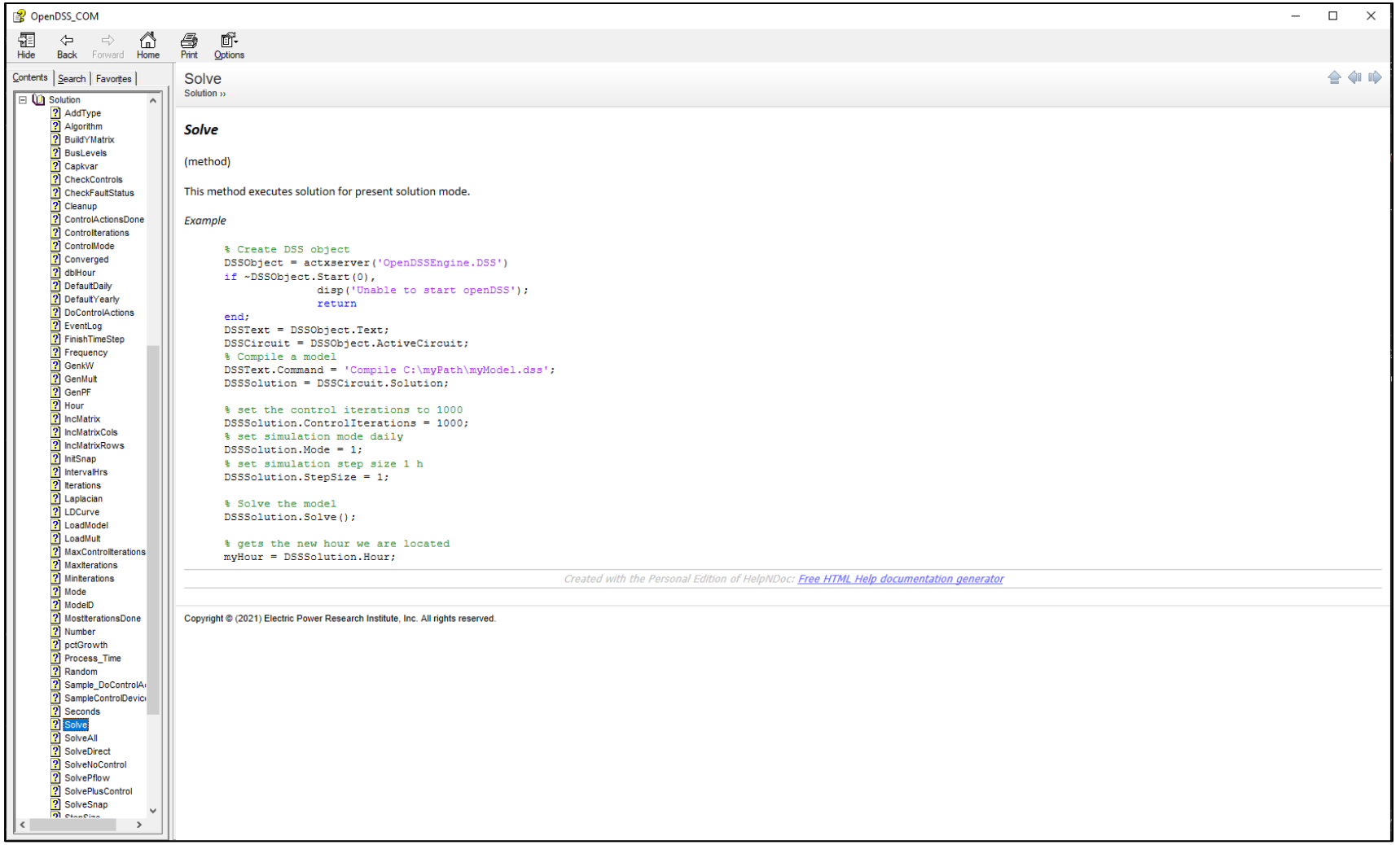
User Interfaces

- Use the COM help file



User Interfaces

- Use the COM help file



Examples on using COM

Direct DLL (Shared library)

EPRI | ELECTRIC POWER RESEARCH INSTITUTE

Direct connection Shared Library (DLL) for OpenDSS

Davis Montenegro, Celso Rocha, Paulo Radatz

Last update 08-26-2021

The direct connection shared library is a DLL that implements the same classes, properties, and methods of the OpenDSS-PM COM interface. This alternative was generated to accelerate the In-process co-simulation between OpenDSS and external software when the client software does not support early bindings connection to COM servers/controls.

Normally, high level programming languages do not support early bindings, which make them use late bindings for data exchanging with COM servers. Late bindings procedures add an important overhead to the co-simulation process specially when executing loops.

So, if your programming language does not support early bindings connection with COM servers, this is the library you should use to accelerate your simulations. This library is called OpenDSSDDirect.dll and can be accessed directly without needing to register it into the OS registry.

However, if your programming language supports early bindings, keep using the COM interface, the simulation speed will be accelerated naturally when using this connection procedure instead of late bindings.

The properties implemented in this library are the same implemented in the COM interface, so, this manual can be used as a reference manual for the classes, properties and methods included in the COM interface.

ActiveClass Interface

This interface implements the ActiveClass (IActiveClass) interface of OpenDSS by declaring 4 procedures for accessing the different properties included in this interface.

ActiveClassI (Int) Interface

This interface can be used to read/modify the properties of the ActiveClass Class where the values are integers. The structure of the interface is as follows:

```
int32_t ActiveClassI(int32_t Parameter, int32_t argument) ;
```

This interface returns an integer (signed 32 bits), the variable "parameter" is used to specify the property of the class to be used and the variable "argument" can be used to modify the value of the property when necessary. Reading and writing properties are separated and require a different parameter number to be executed.

The properties (parameter) are integer numbers and are described as follows:

EPRI | ELECTRIC POWER RESEARCH INSTITUTE

Alternative Interfaces

These interfaces were made to cover particular requests and can be available in future versions of this DLL. The structure of these interfaces can be different from the rest of the interfaces proposed above.

InitAndGetYparams Interface

This interface initializes the YMatrix of the system in case of being necessary. It must be executed before executing the interface *GetCompressedYMatrix* to reduce the possibility of errors. The structure of this interface is as follows:

```
Uint32 InitAndGetYparams(uint32 hY, uint32 nBus, uint32 nNZ);
```

GetCompressedYMatrix Interface

This interface gets the YMatrix of the system in a reduced format by delivering the pointers of row, column and matrix values. The structure of this interface is as follows:

```
Void GetCompressedYMatrix(uint32 hY, uint32 nBus, uint32 nNz, uint32 ColPtr, uint32 RowIdx,
                          uint32 cVals);
```

ZeroInjCurr Interface

This interface commands to OpenDSS to initialize currents vector as an array of zeros, which is basically the first step in the solution algorithm (DoNormalSolution routine). The structure of this interface is as follows:

```
Void ZeroInjCurr();
```

GetSourceInjCurrents Interface

This interface commands to OpenDSS to include the currents injected by voltage/current sources into the currents vector, following the procedure proposed in the solution algorithm (DoNormalSolution routine). The structure of this interface is as follows:

```
Void GetSourceInjCurrents();
```

GetPCInjCurr Interface

Direct connection Shared Library (DLL) for OpenDSS

Examples on using Direct DLL

Myths and legends about interfaces

Myths and legends about user Interfaces

- COM interface is slow…

How to speed up your co-simulation using OpenDSS COM interface

Authors: Davis Montenegro, Roger Dugan
October 27, 2015

Many times users have mentioned performance issues when co-simulating using OpenDSS. The interface proposed in OpenDSS to perform co-simulations is the COM interface, which can be accessed from almost every programming language; however, not necessarily in the faster way.

Because of this, many users blame the COM interface for their performance issues. The aim of this document is to clarify why users could experience a low performance in terms of co-simulation speed and to give a guide about how to improve it.

Early Binding and Late Binding

Early and Late bindings are two computer programming mechanisms for accessing COM servers/ActiveX controls considering the features of the interface objects and classes. Late binding is focused on giving flexibility in case the objects/classes contained within the interface are polymorphic. For doing this, the late binding mechanism uses a Virtual table (vtable) that includes the memory address of each allocated class and its features. Every time the external program accesses an object/class using the COM interface it must go to the vtable first (if something changes it will be updated into the vtable), then look for the object to obtain the memory address and its features, which adds significant overhead on each iteration.

In contrast, early binding considers that every object is static in time and will be the same during the connection with the external software. For doing this, the vtable is eliminated and the index to each object/class is made by specifying the DispID memory address directly during the compilation phase. This eliminates the overhead generated by accessing vtables and the access to the COM interface objects is drastically faster [1].

To get connected to the COM interface using early binding, first check to see if your programming language supports early binding connection. Normally, all programming languages support late bindings but not necessarily early binding. However, in the next sections, this article presents an alternative for programming languages that do not support early binding.

If your programming software language supports early binding the activation of this connection mechanism consists of adding a couple of code lines when establishing the connection with the COM server. To evaluate the performance of the early binding vs. late

And remember:

It all depends!

R. C. Dugan

..false

Myths and legends about user Interfaces

- Direct DLL is faster than COM interface…

The Delphi Test Routine

The actual Delphi code for the loop in Figure 1 is:

```
procedure TForm1.Button2Click(Sender: TObject);
Var
     i:Integer;
     iLine : Integer;
     DSSLoads : ILoads;

begin
      SW := TStopWatch.Create();
      DSSLoads := DSSCircuit.Loads ;
      SW.Start;
      With DSSLoads  Do
        for i := 1 to 1000 do  Begin
          iLine := First;
          while iline > 0 do  Begin
               kW := 50.0;
               kvar := 20.0;
               iLine := Next;
          End;
      End;

      SW.Stop ;
      Edit1.Text := Format('%d ms for %d Loads 1000 times',
                           [SW.ElapsedMilliseconds, DSSLoads.count]);
end;
```

The early-binding connection to the OpenDSS COM interface is made using the following code to make a connection to the IDSS interface. Then some local variables are used to achieve a little better performance by maintaining a static connection to other interfaces that are created by the call to "coDSS.Create".

Definition of Variables

```
Var
    DSSObject: IDSS;  // DSS Interface
    DSSText:   IText;
    DSSCircuit: ICircuit;
    DSSSolution:ISolution;
    DSSProgressfrm:IDSSProgress;
```

Connection to the Interface

```
        TRY

            DSSObject := coDSS.Create; // Creates early binding
            DSSObject.Start(0);
        EXCEPT
            On E:Exception Do Begin
              MessageDlg('Did not Start.',  mtConfirmation, [mbYes, mbNo],
0, mbYes);
              Exit;
            End;

        END;
```

VBA Early and Late Binding

VBA (Visual Basic) is a programming language that supports both early binding in a simple way as well as the traditional initialization of the COM interface using late binding.

Late Binding

```
Public DSSObj as Object
....
Set DSSObj = CreateObject("OpenDSSEngine.DSS")
```

Early Binding

```
Public DSSObj as OpenDSSEngine.DSS
...
Set DSSObj = New OpenDSSEngine.DSS
```

The measured computing times using each technique are shown in Figure 2.

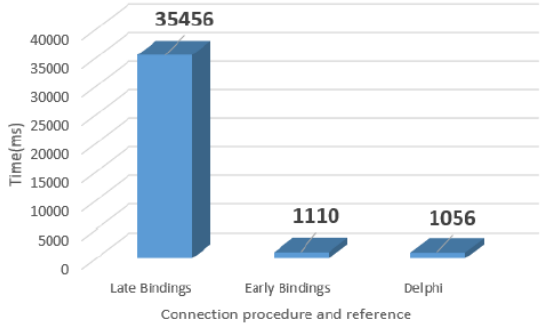


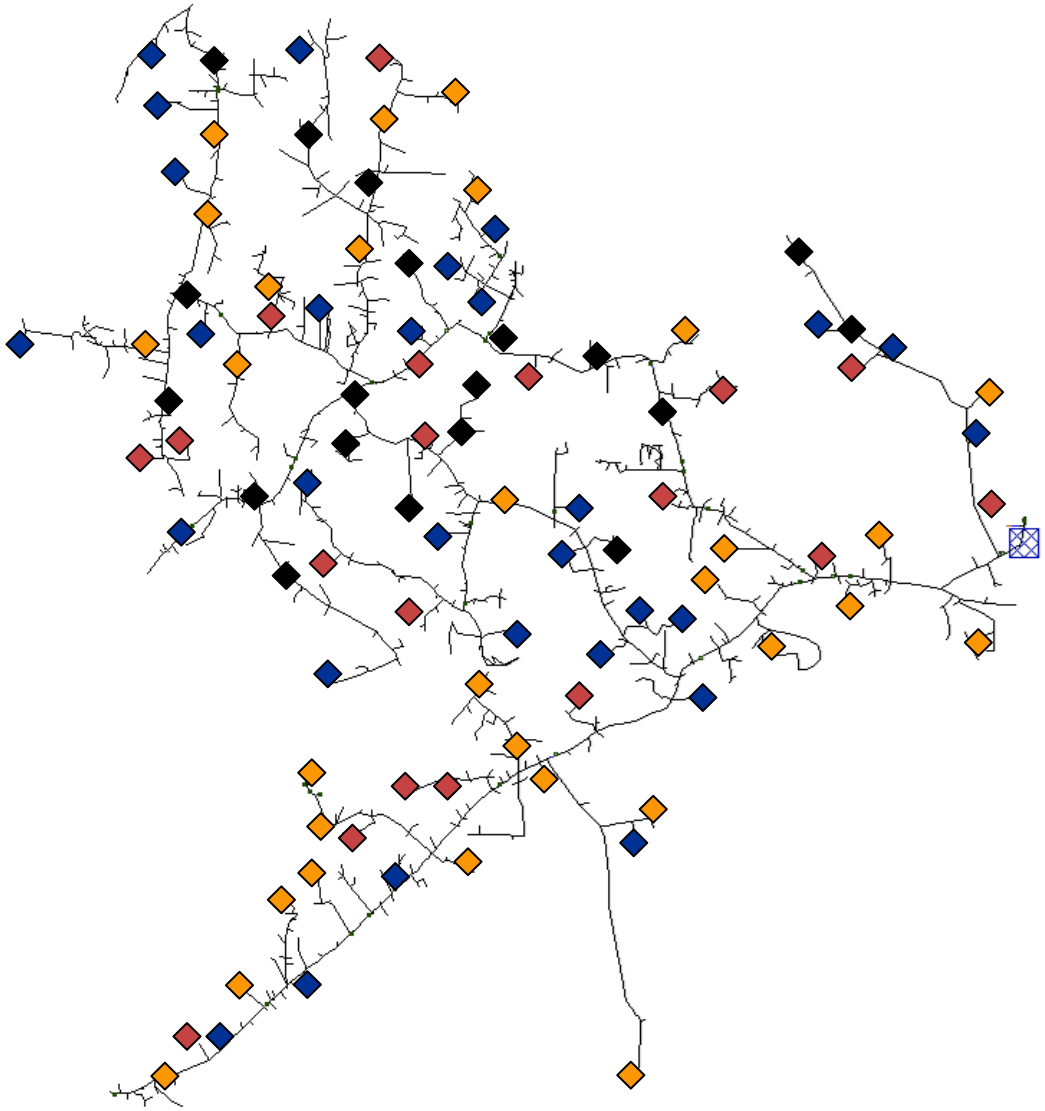
Figure 2. Computing times in *ms* for the algorithm in Figure 1 using VBA late binding and early binding in comparison with the reference (Delphi)

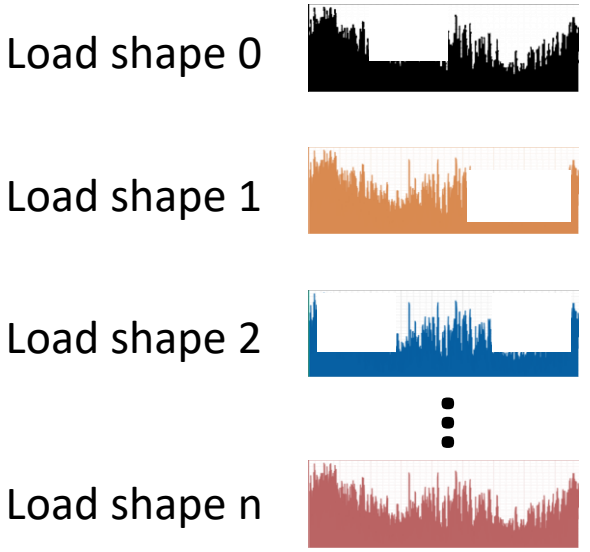
..false

New functionalities in DSS

Load shape aggregation

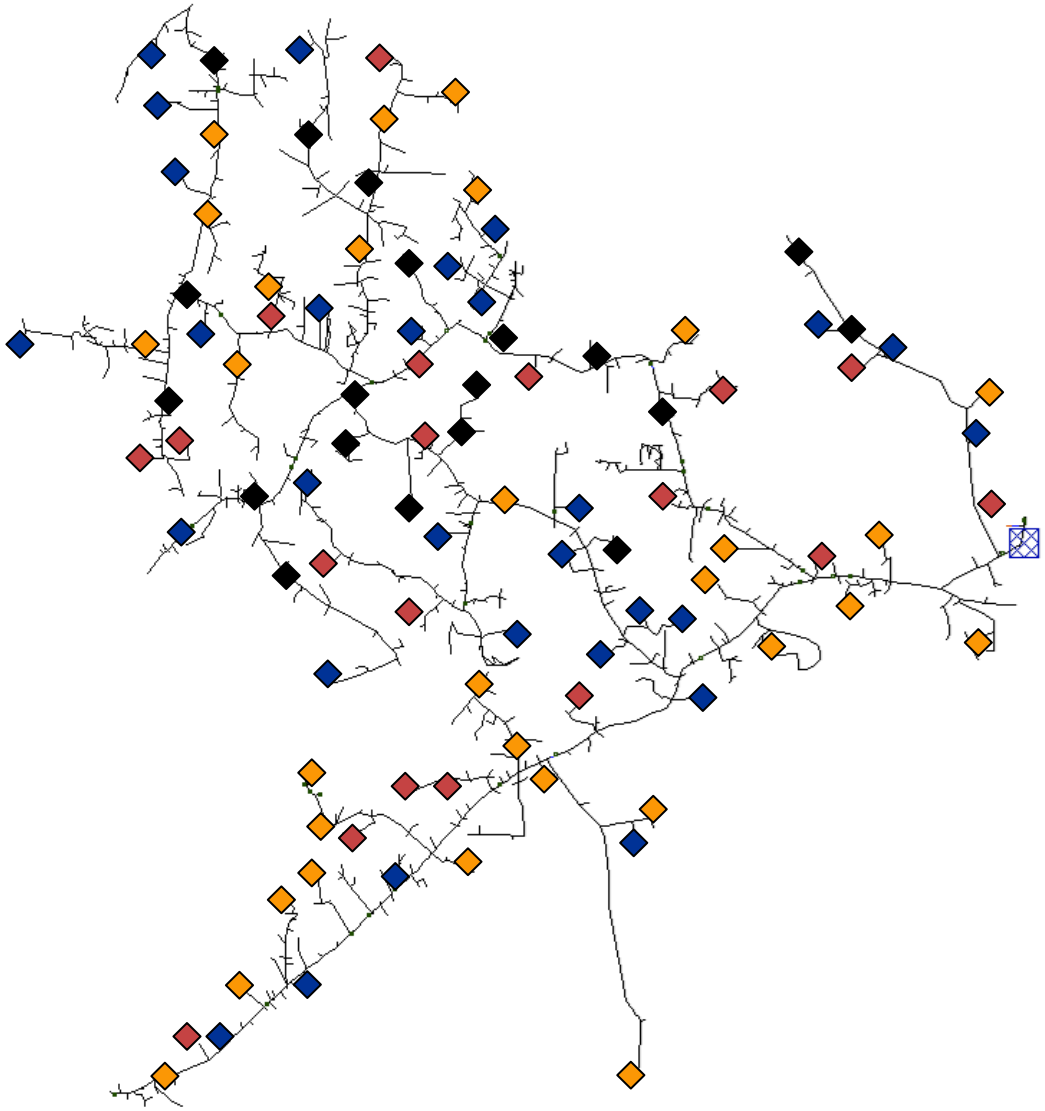
Data diversity within a distribution model

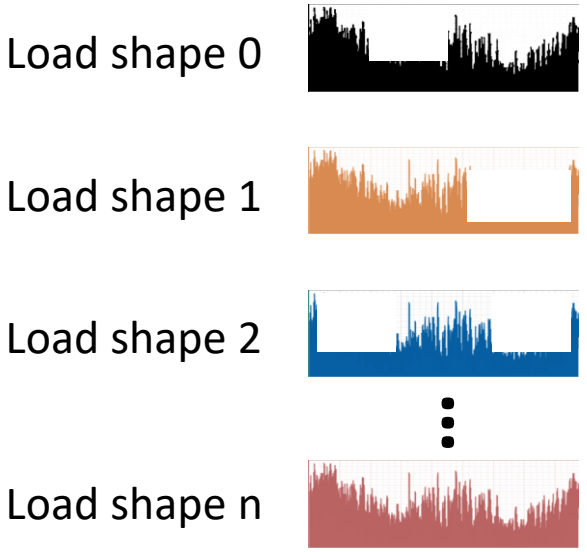




- Abundant data coming from AMI on the field help to create a very accurate model, improving the fidelity of the model for power system studies.
- But nothing is for free…

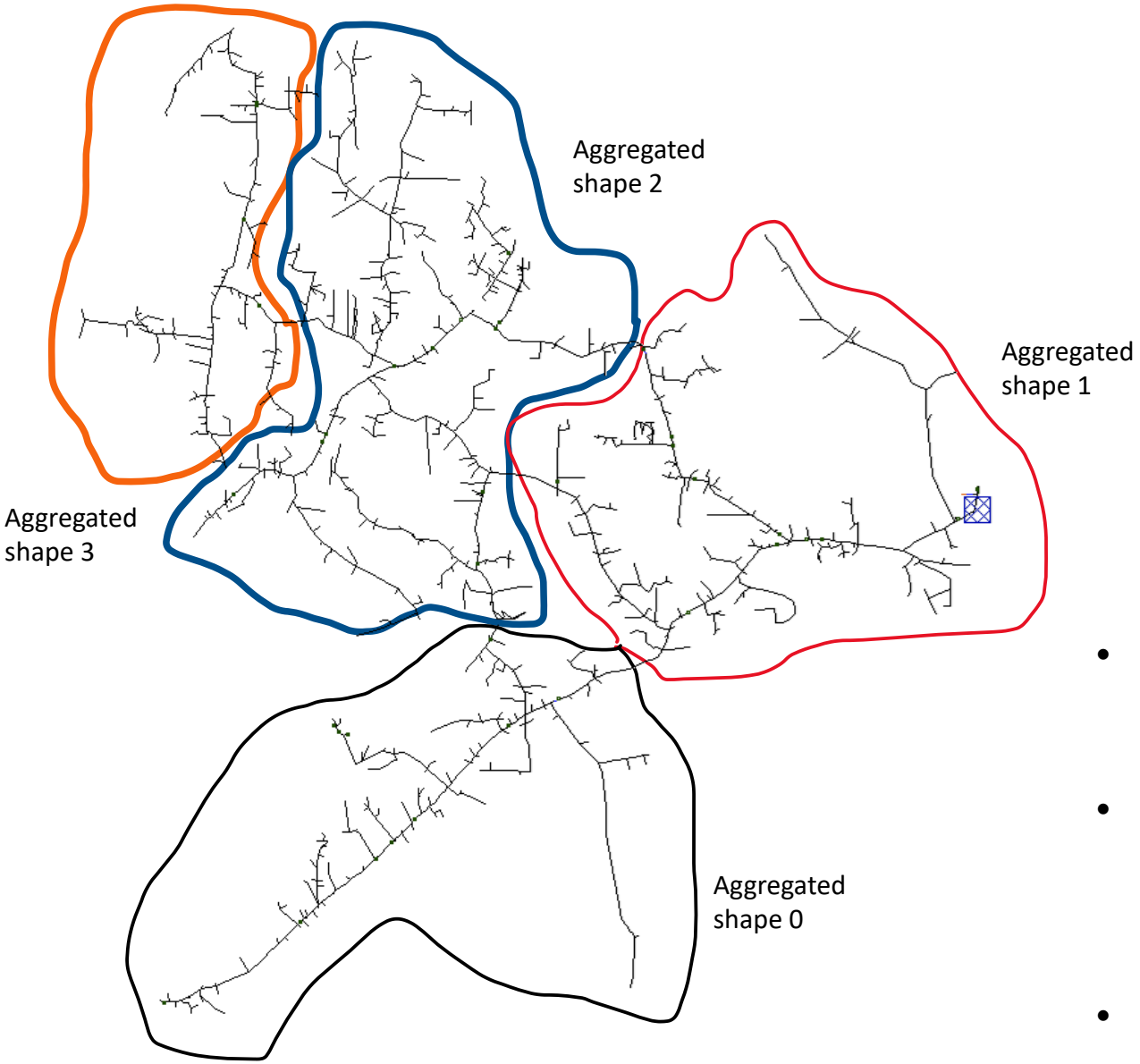
Data diversity within a distribution model

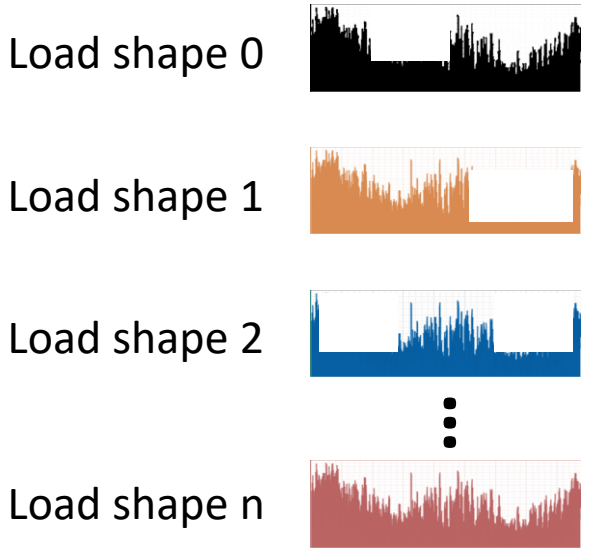




- Large amount of data means large amount of memory.
- Parallelization becomes an issue in this case, limiting the simulation capabilities when working with standard computer architectures.
- We need to transform data into information.

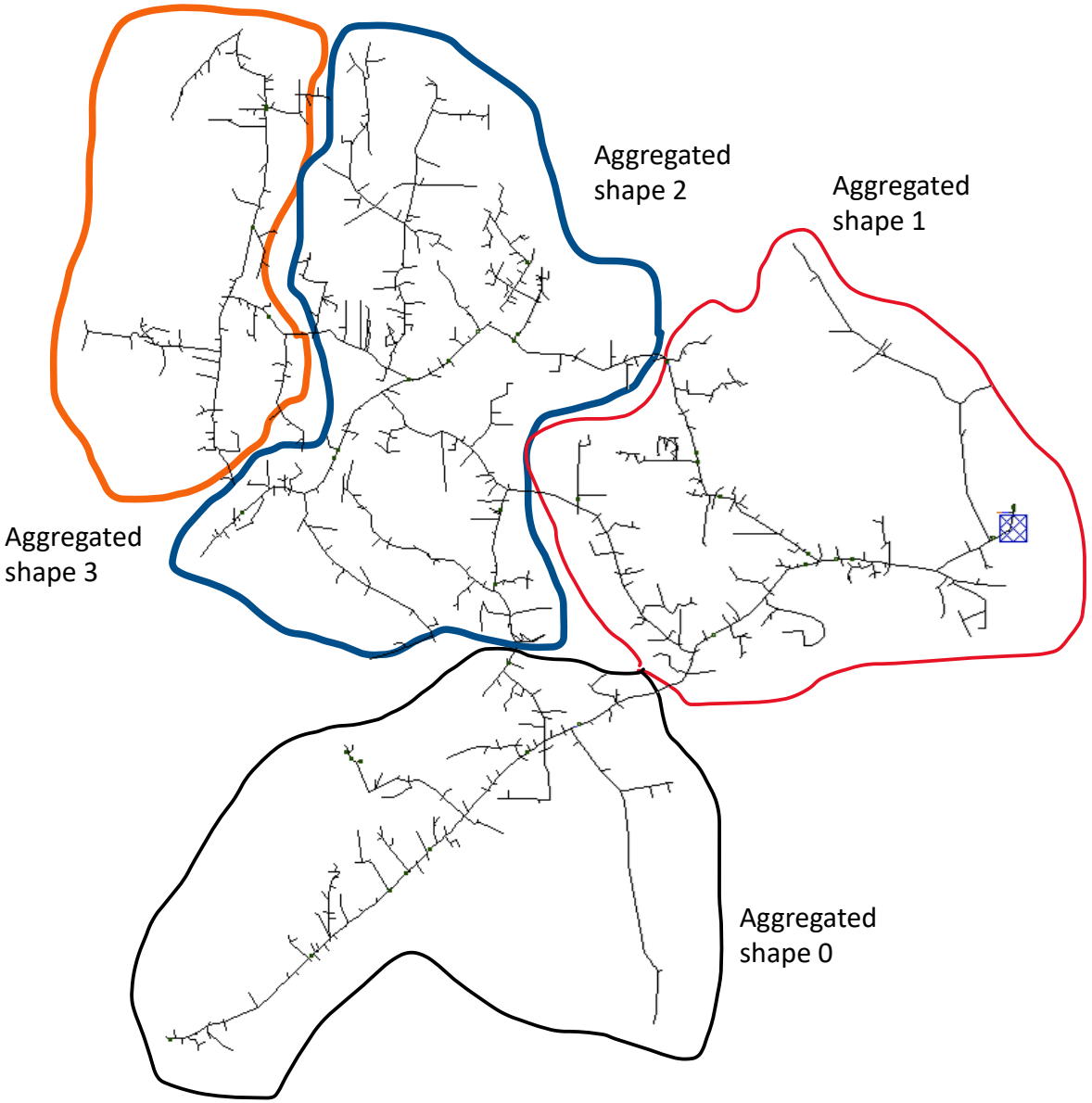
Making the model lighter





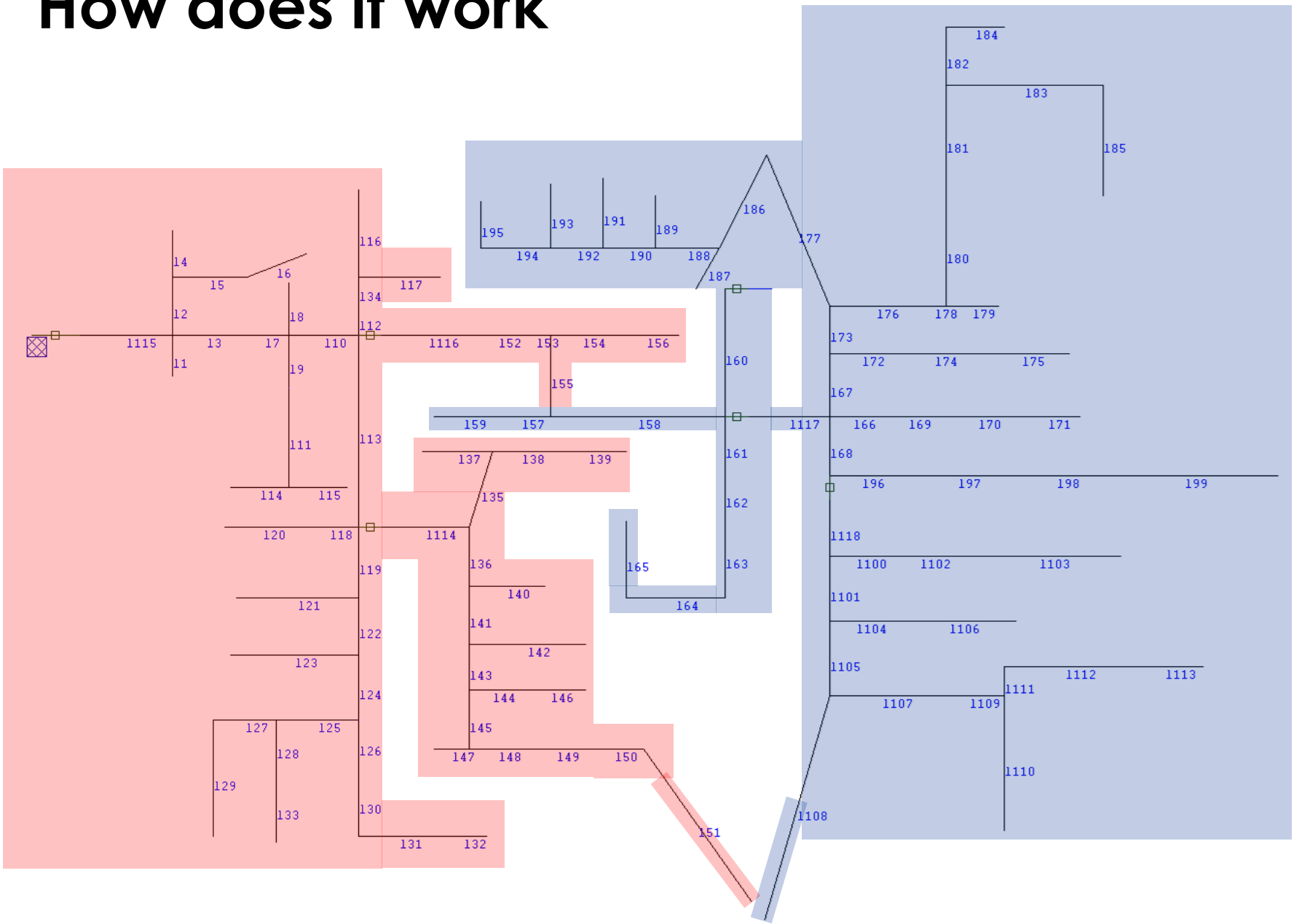
- Splitting the model into zones for aggregating load profiles sounds like a promising alternative.
- By measuring the power behavior at the head of the zone, we can estimate an approximate shape that will work for everybody within the zone.
- It must be flexible enough to reflect accuracy.

How does it work



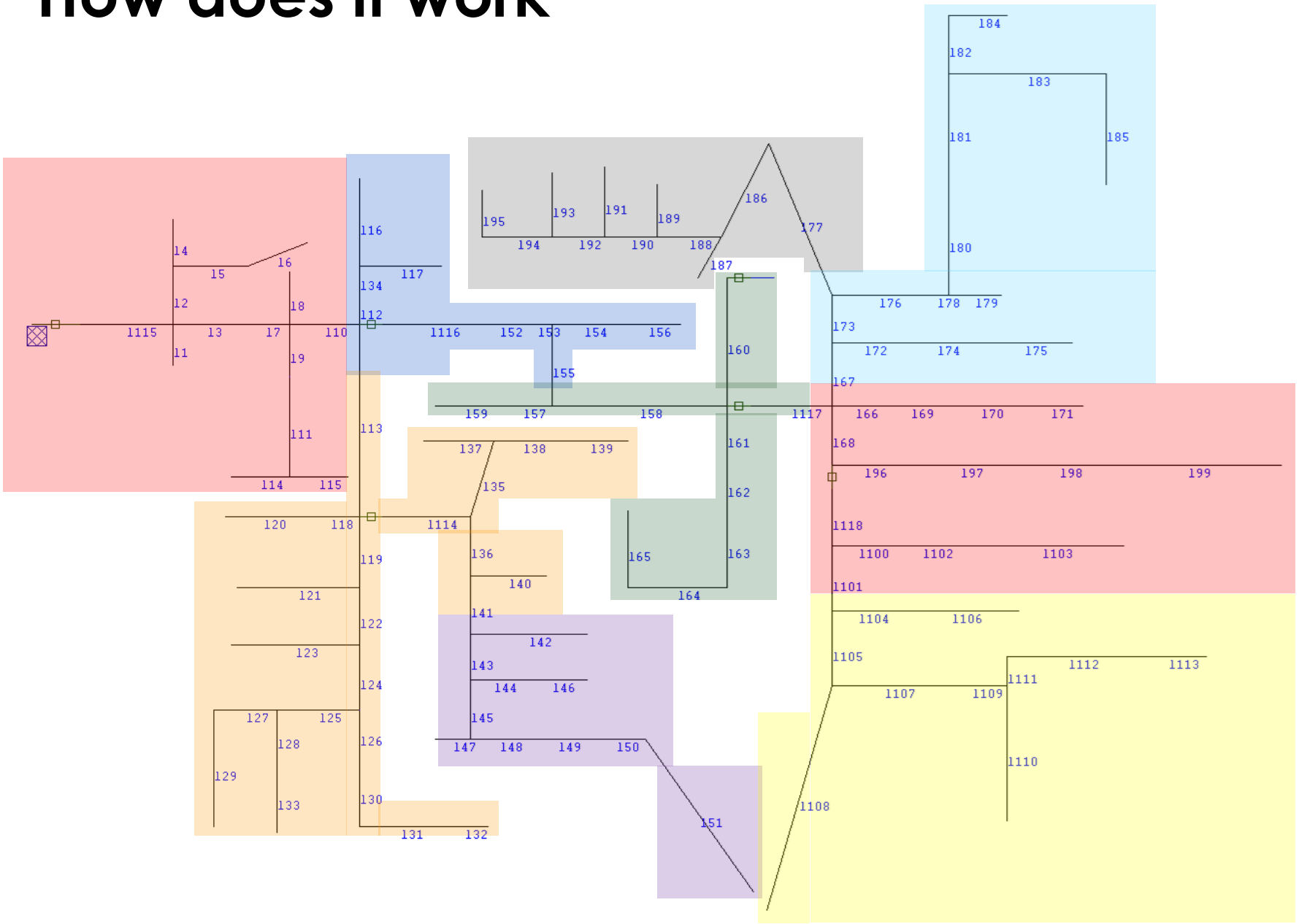
- OpenDSS uses MeTIS for tearing the model in order to obtain a symmetrical distribution based on the number of buses.
- MeTIS was integrated into OpenDSS when developing the A-Diakoptics suite.
- After creating the zones, OpenDSS will add monitors to the head of each zone and run a yearly simulation to check on how the power profile looks at each point.
- With the new measurements creates an aggregate load profile that is then linked to the loads within each zone.
- The accuracy of this approach depends on the number of zones created; this number of zones is customizable for the user. The user decides if the aggregation is good enough or if it requires more zones to be considered.

How does it work



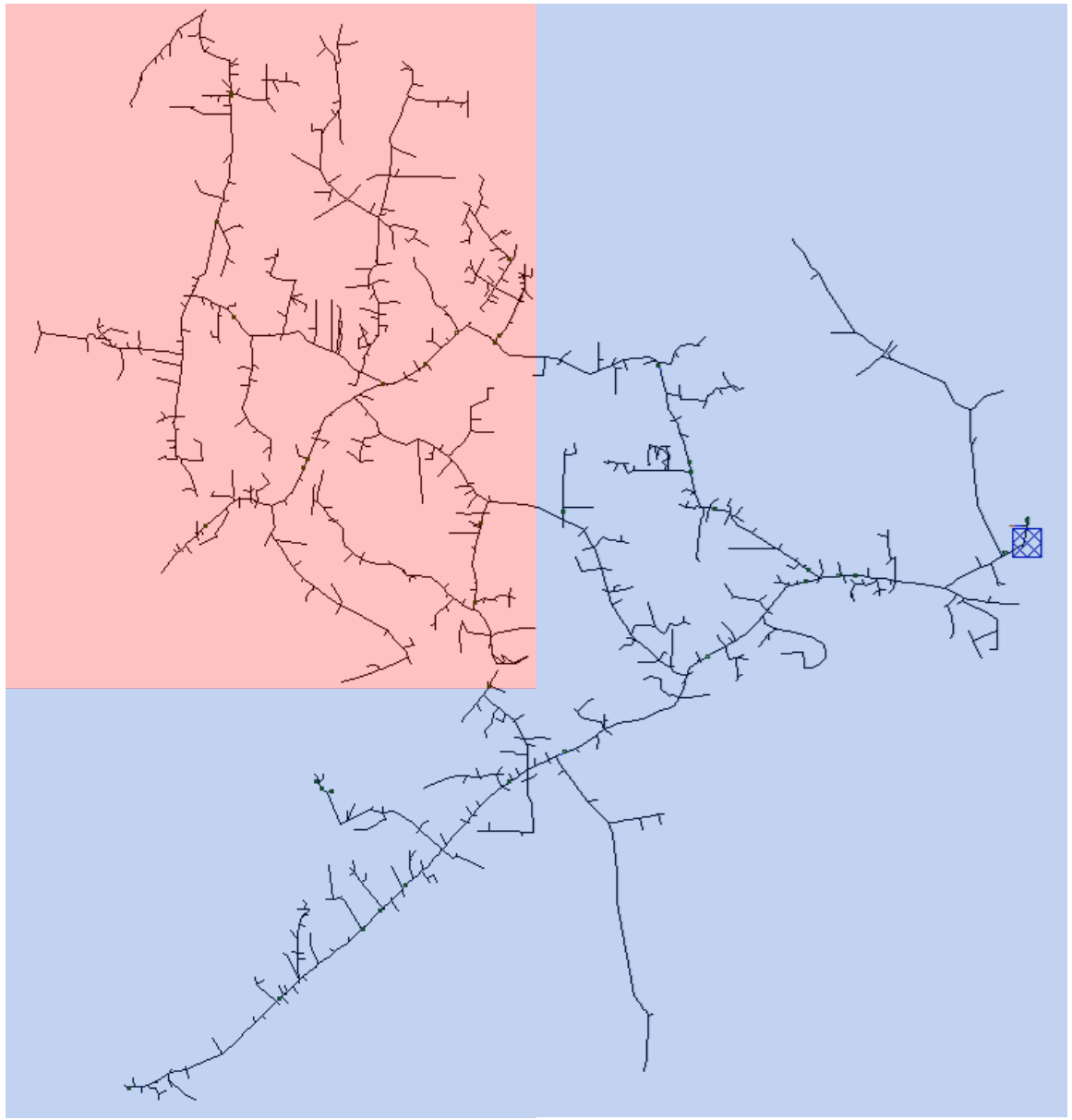
Example tearing the IEEE123 test model, 2 zones

How does it work



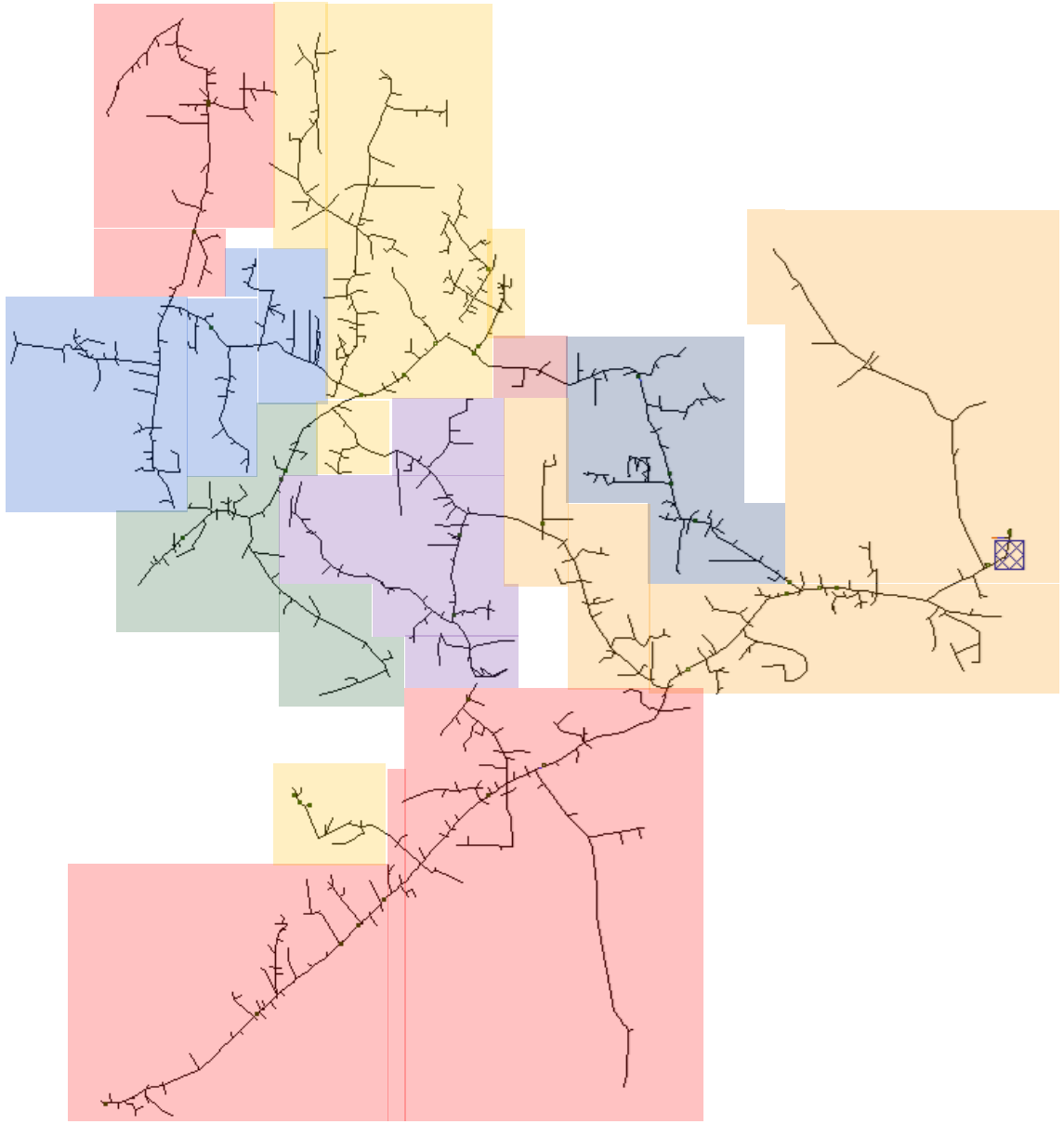
Example tearing the IEEE123 test model, 9 zones

How does it work



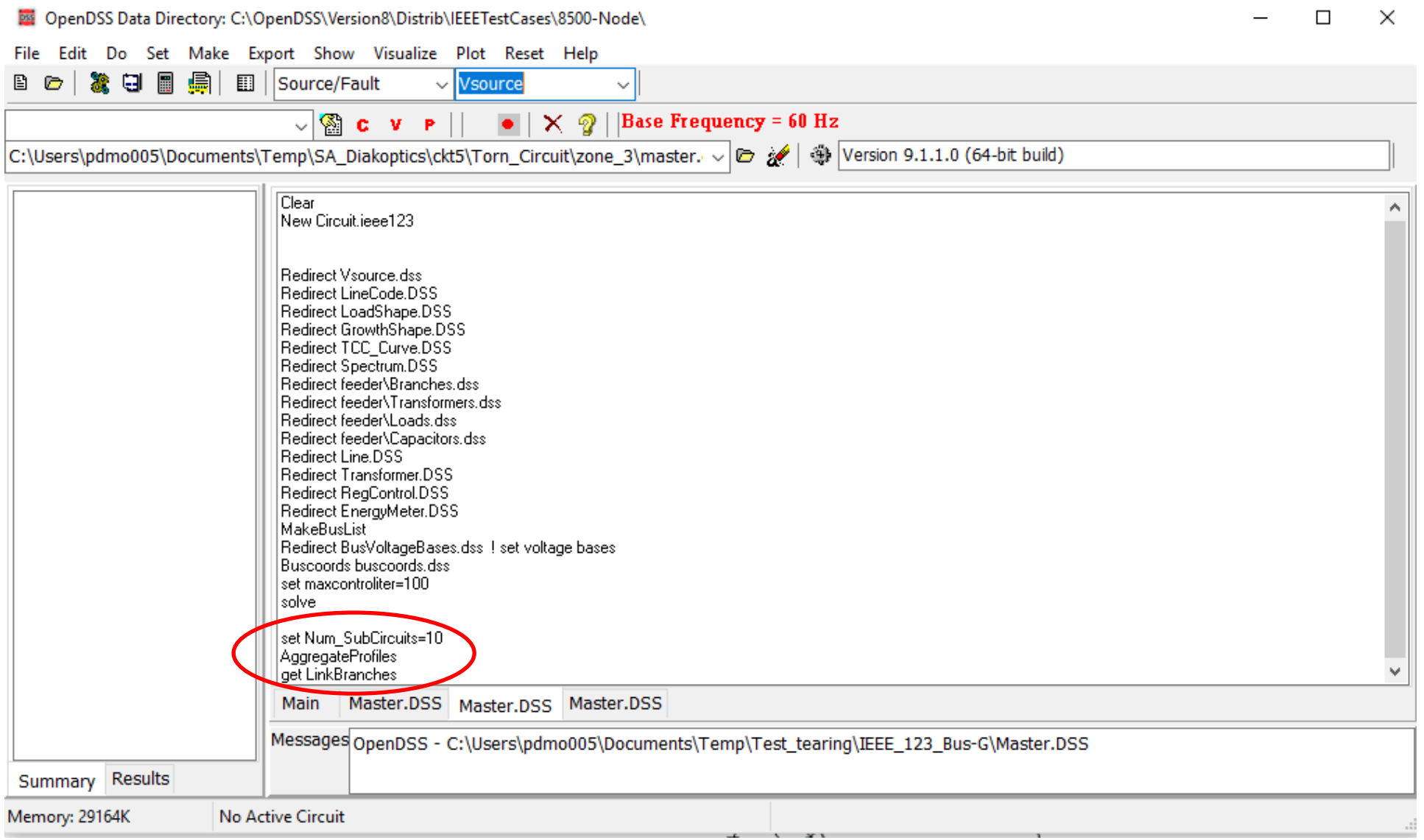
Example tearing the IEEE 8500 test model, 2 zones

How does it work



Example tearing the IEEE 8500 test model, 10 zones

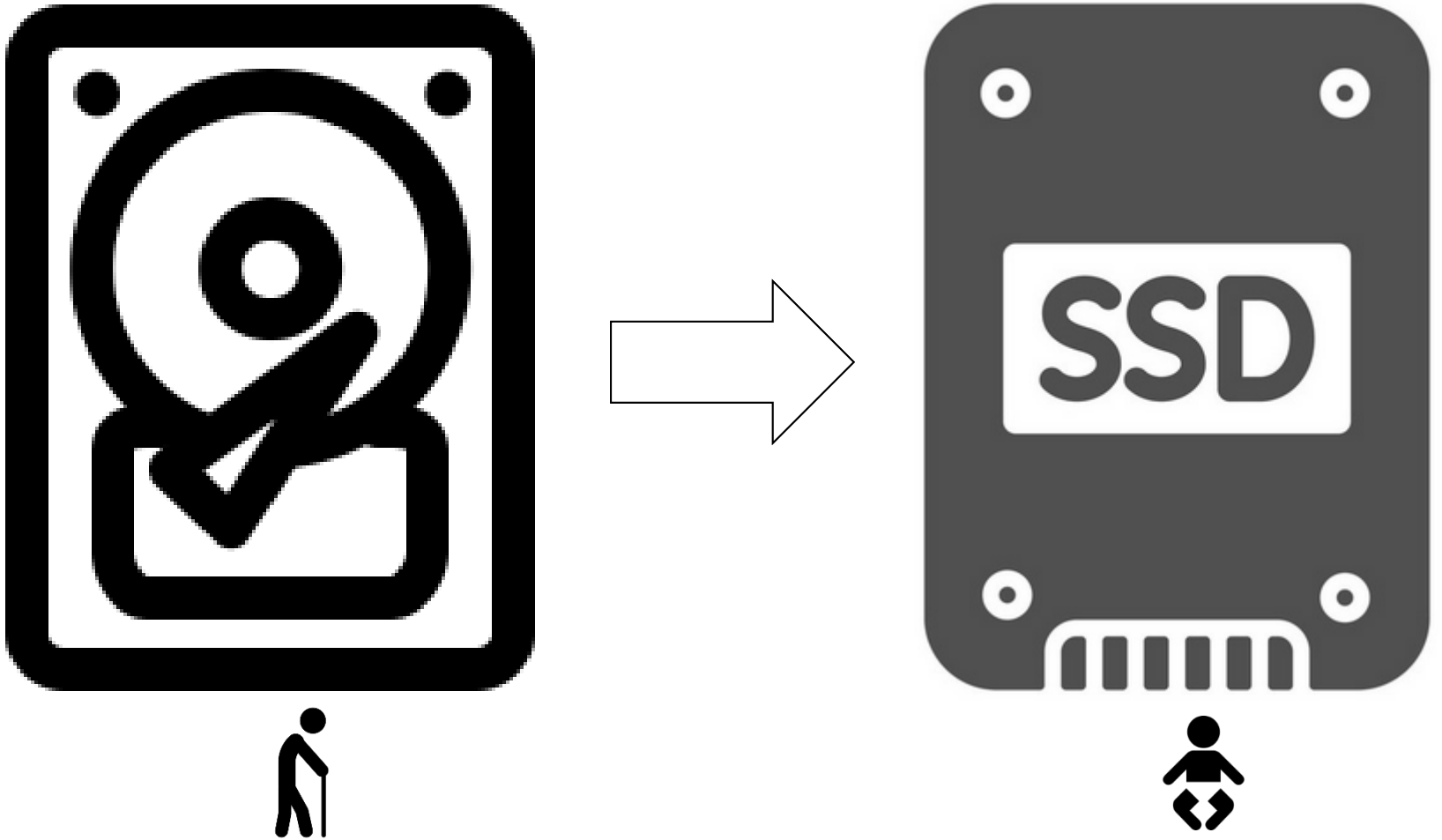
The command in OpenDSS



Memory mapping

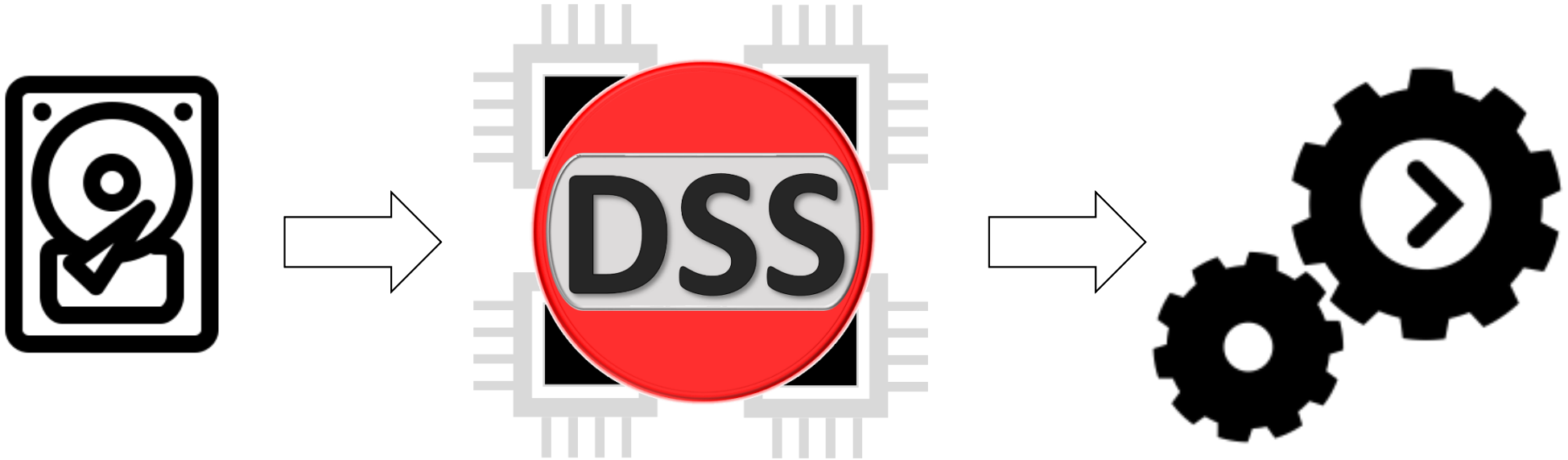
Introducing memory mapped files into OpenDSS

Catching up with technology



Introducing memory mapped files into OpenDSS

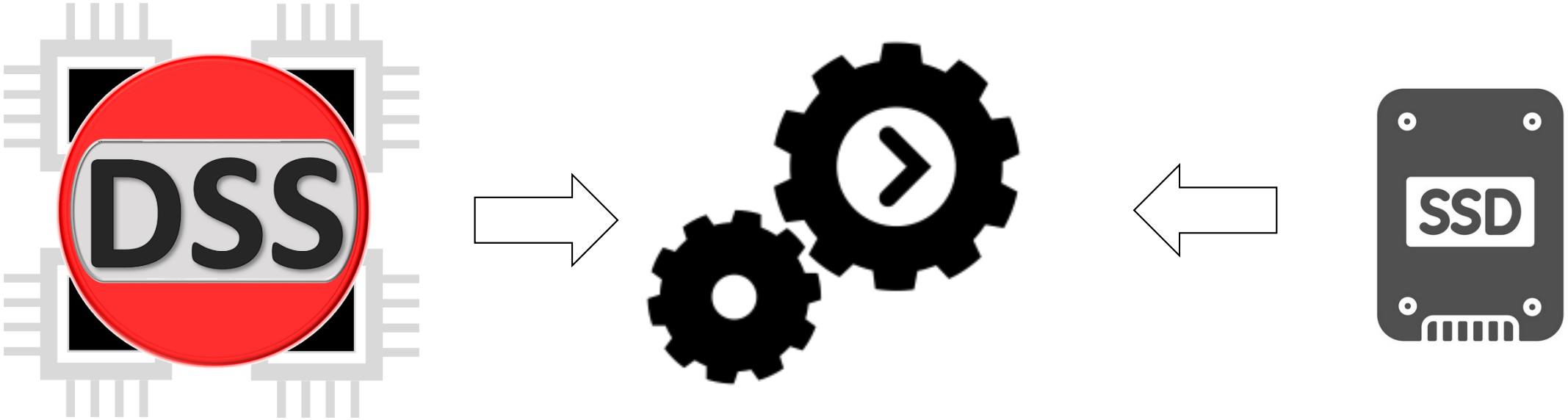
Catching up with technology



39

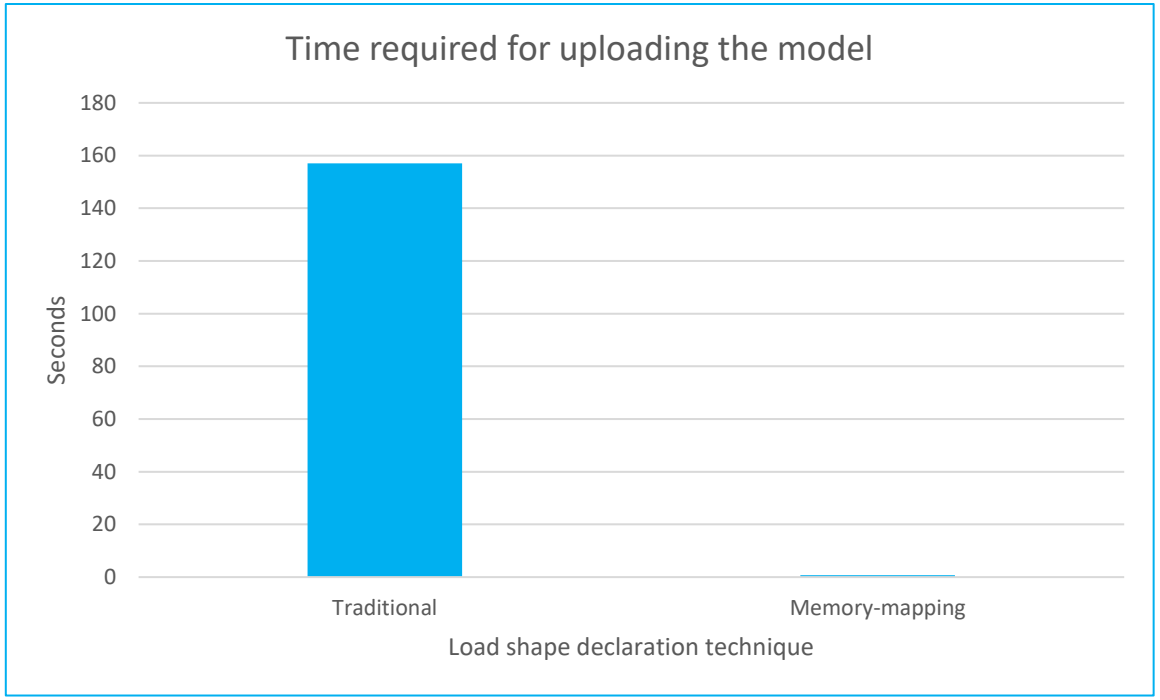
Introducing memory mapped files into OpenDSS

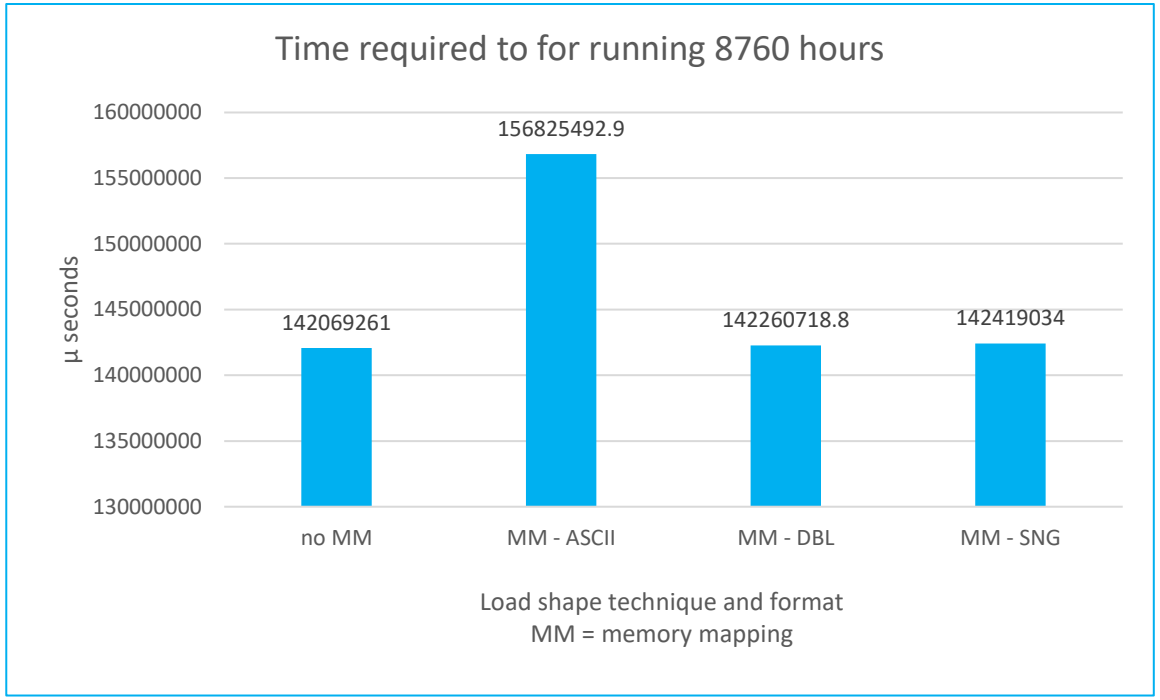
Catching up with technology



Introducing memory mapped files into OpenDSS

Catching up with technology





New Loadshape.LS_PhaseA npts=8760 interval=1 MemoryMapping=Yes mult=(file=myFile.txt)

Dynamic expressions

What are dynamic expressions

There is a current interest in the industry for integrating dynamic studies into large scale models for different applications.

OpenDSS has such functionality, however, the dynamic behavior of PCE such as generators is scripted within the model code, suggesting that for representing a different dynamic (swinging equation), the user needs to modify the source code for the element or create a new one.

Dynamic expressions is an object type in OpenDSS for entering a customizable swinging equation describing a dynamic behavior that can be adopted by other elements in the model

Dynamic expressions

$$\dot{V} = -\frac{D}{M}V + \frac{Pshaft - Pterm}{M}$$
$$\dot{\theta} = V$$

```
ClearAll

// Set the base frequency to 60Hz
Set DefaultBaseFrequency=60

Var @Zbase=53.615

// quasi-ideal source for inf. bus at SourceBus with initial conditions Vpu=0.90081 and Vangle = 0
New Circuit.SimpleDemo
~ BasekV=345 pu=0.90081 phases=3
~ Angle = 0.0 Model=ideal puZideal=[ 1.0e-7, 0.00001]  BaseMVA=2220

// New parallel lines from SourceBus to high side of transformer
New Line.Source_HT_1 Bus1=SourceBus Bus2=HT  R1=0 X1=(0.5 @Zbase *) R0=0 X0=(0.5 @Zbase *) C1=0
C0=0 length=1 Units=mi
New Line.Source_HT_2 Bus1=SourceBus Bus2=HT  R1=0 X1=(0.93 @Zbase *) R0=0 X0=(0.93 @Zbase *)
C1=0 C0=0 length=1 Units=mi

// New Transformer with reactance j0.15
New Transformer.Step_Up Phases=3 Windings=2 XHL=15 ppm=0
~ buses=(HT LT) conns='wye wye' kvs="345 24" kvas="2220000 2220000" %Loadloss=0

// Constant kW at specified power factor
New Generator.G1 Bus1=LT kV=24 kW=(2220000 0.9 *) kvar=(2220000 0.436 *)   Model=1 vminpu= 0.80
Vmaxpu=1.4  MVA=2220  XRdp=1e12  Xdp=0.3  Xdpp=0.25 H=3.5 D=0

// Set all voltage bases in model
```

Dynamic expressions

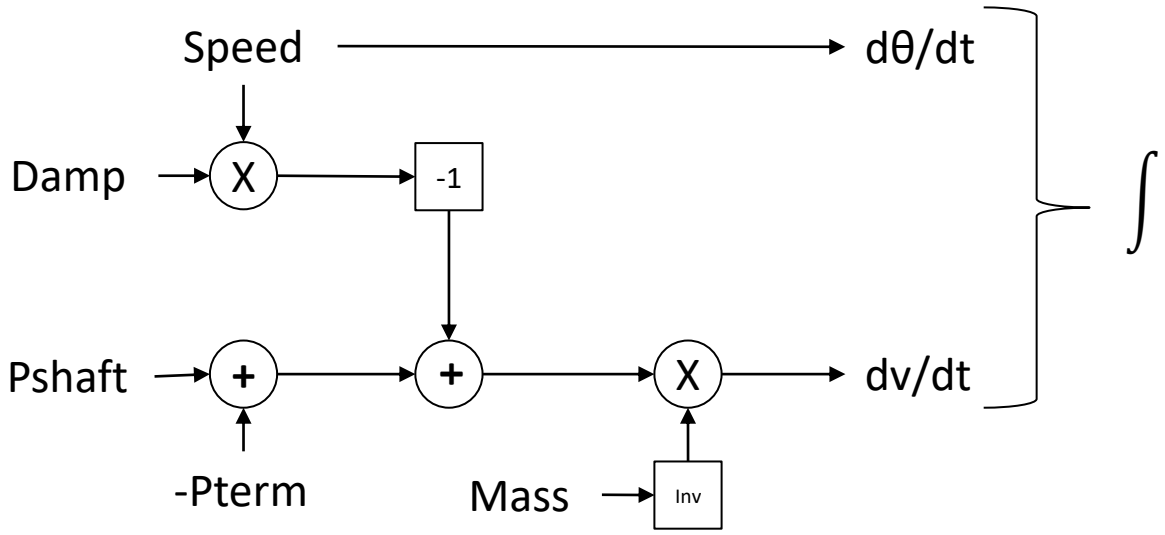
$$\dot{V} = -\frac{D}{M}V + \frac{Pshaft - Pterm}{M}$$
$$\dot{\theta} = V$$

```
ClearAll

// Set the base frequency to 60Hz
Set DefaultBaseFrequency=60
Var @Zbase=53.615
// quasi-ideal source for inf. bus at SourceBus with initial conditions Vpu=0.90081 and Vangle = 0
New Circuit.SimpleDemo
~ BasekV=345 pu=0.90081 phases=3
~ Angle = 0.0 Model=ideal puZideal=[ 1.0e-7, 0.00001]  BaseMVA=2220

// New parallel lines from SourceBus to high side of transformer
New Line.Source_HT_1 Bus1=SourceBus Bus2=HT  R1=0 X1=(0.5 @Zbase *) R0=0 X0=(0.5 @Zbase *) C1=0
C0=0 length=1 Units=mi
New Line.Source_HT_2 Bus1=SourceBus Bus2=HT  R1=0 X1=(0.93 @Zbase *) R0=0 X0=(0.93 @Zbase *)
C1=0 C0=0 length=1 Units=mi

// New Transformer with reactance j0.15
New Transformer.Step_Up Phases=3 Windings=2 XHL=15 ppm=0
~ buses=(HT LT) conns='wye wye' kvs="345 24" kvas="2220000 2220000" %Loadloss=0

// Creates the diffrential equation for the generator's dynamics
New DynamicExp.myDiffEq nvariables=6 varnames=[Speed Mass PShaft Pterm Damp theta]
~ expression=[Speed dt = 1 Mass / (Pshaft Pterm - Damp Speed * -) *; theta dt = Speed]

New Generator.G1 Bus1=LT kV=24 kW=(2220000 0.9 *) kvar=(2220000 0.436 *)   Model=1 vminpu= 0.80
Vmaxpu=1.4 DynamicEq=myDiffEq MVA=2220  XRdp=1e12  Xdp=0.3  Xdpp=0.25
// Initializes the dynamic equation's state variables
~ Damp = 0 PShaft = P0 Pterm = P Speed = 0 theta = Edp
~ Mass = (3.5 2 * 2220000000 376.99112 / *)
~ DynOut = [Speed theta]
// Set all voltage bases in model
…
```

Dynamic expressions

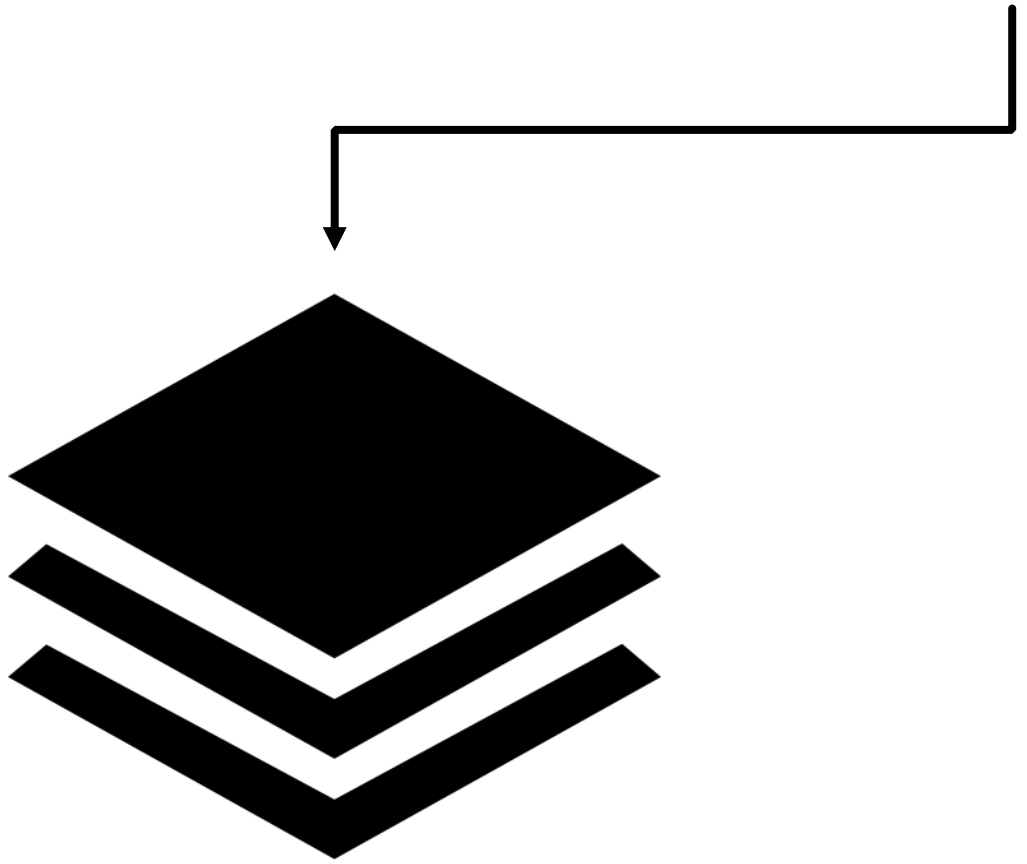
$$\dot{V} = -\frac{D}{M}V + \frac{Pshaft - Pterm}{M}$$
$$\dot{\theta} = V$$

[Speed dt = 1 Mass / (Pshaft Pterm - Damp Speed * -) *;
theta dt = Speed]

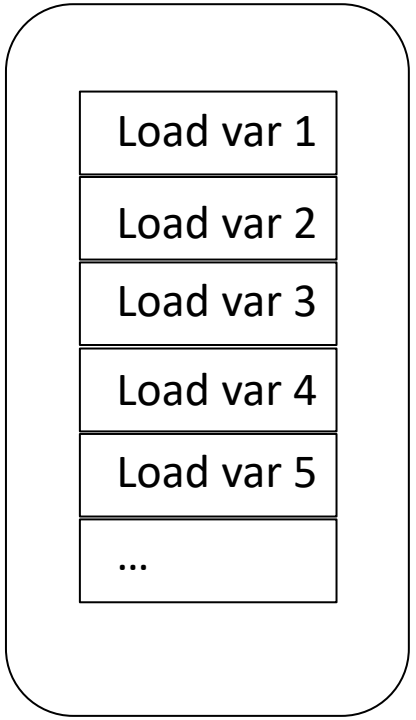


Dynamic expressions

"[v dt = 1 M / (Pshaft Pterm - D v * -) *; theta dt = v]"

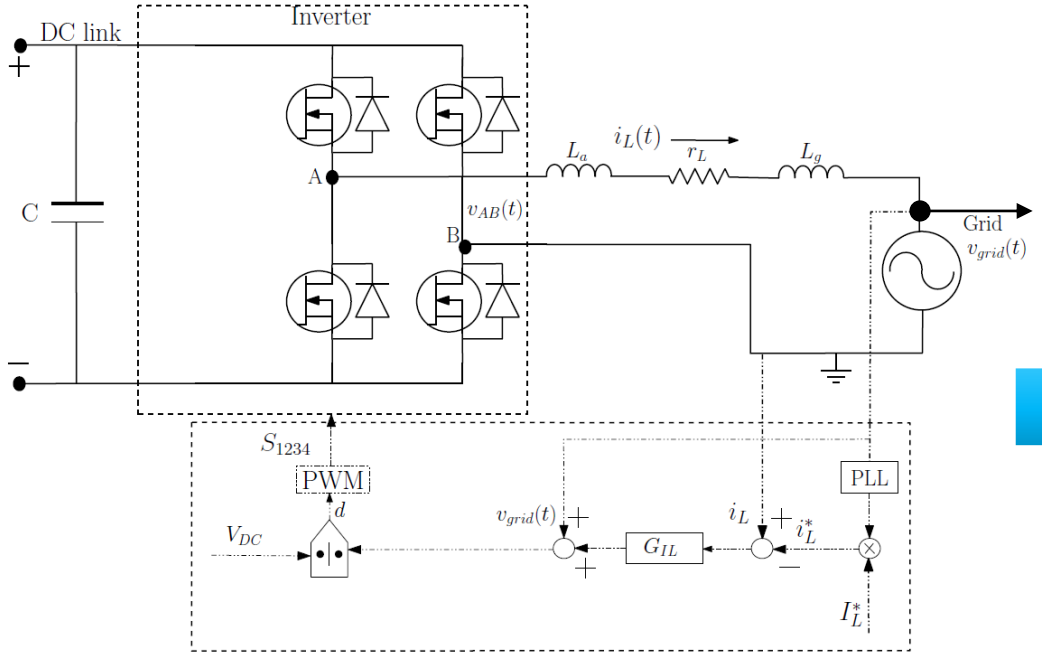


Load var 1
Load var 2
Operate
Load var 3
Operate
Load var 4
Load var 5
Operate
…

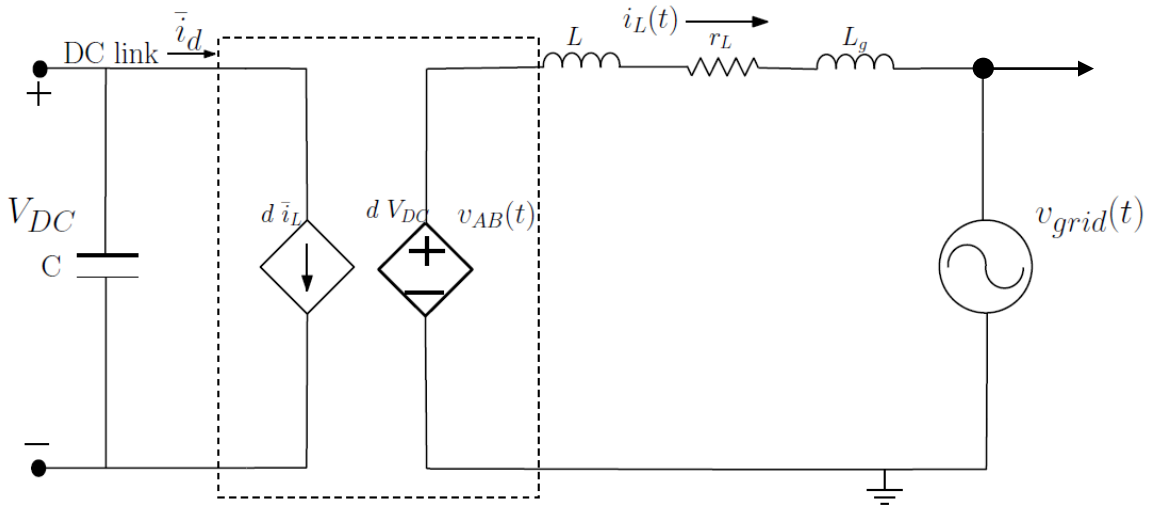


Adopting obj
“Generator”

Dynamic expressions



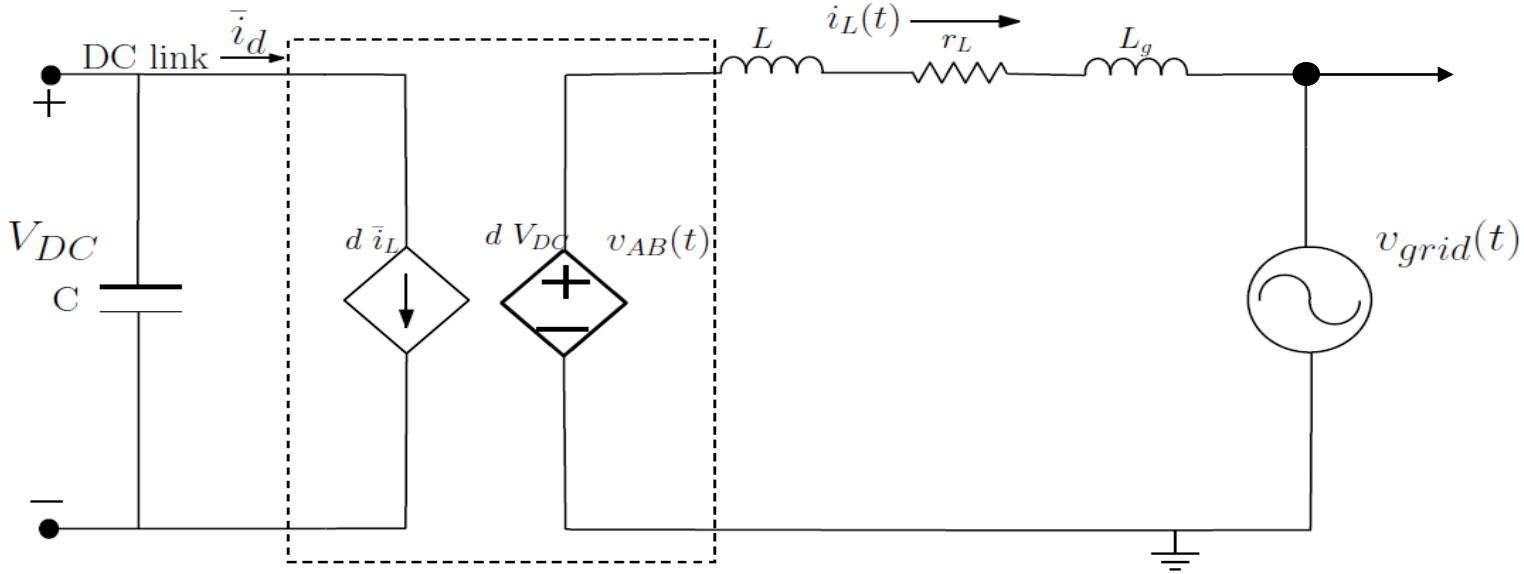
Detailed model



Approximated average model in dynamic phasor domain

Dynamic expressions

Approximated average model in dynamic phasor domain



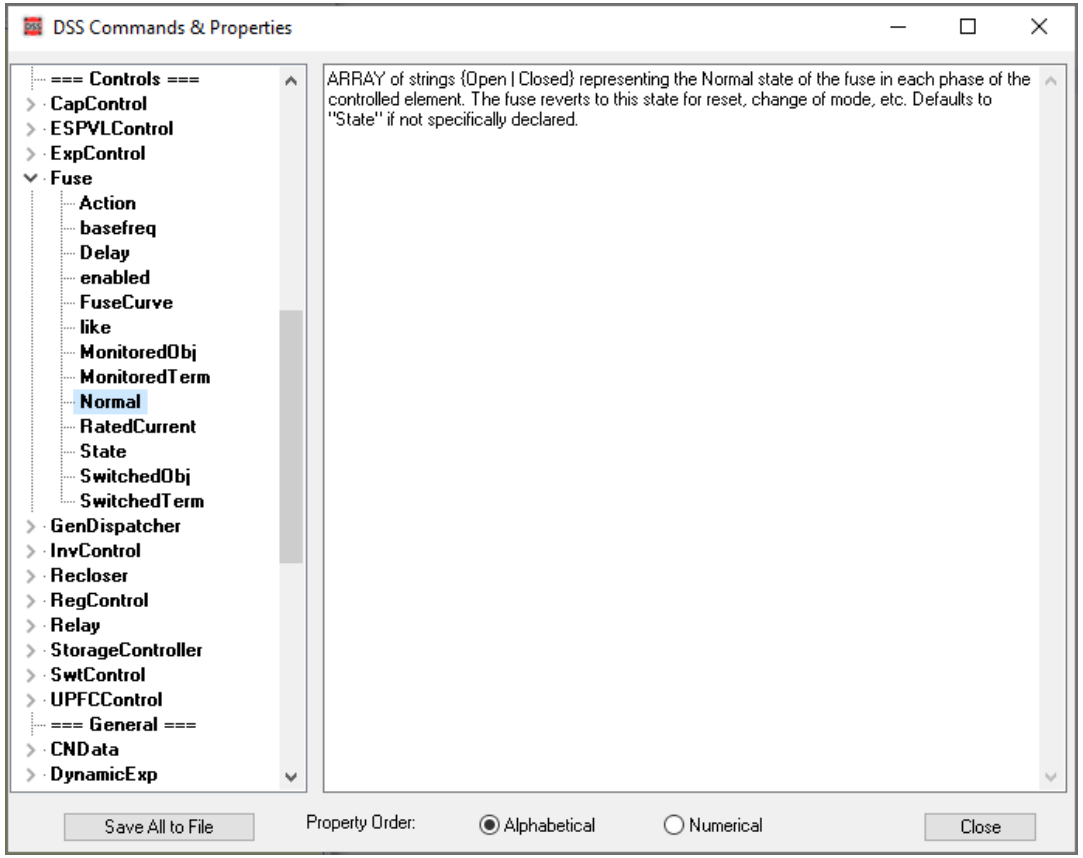
$$\frac{d\langle i_L\rangle_1^R}{dt} = \omega\ \langle i_L\rangle_1^I + \frac{1}{L}\ \left(-r_L\ \langle i_L\rangle_1^R\ -\ \langle x_1\rangle_1^R\ V_{DC}\right)$$

$$\frac{d\langle i_L\rangle_1^I}{dt} = -\ \omega\ \langle i_L\rangle_1^R + \frac{1}{L}\ \left(-r_L\ \langle i_L\rangle_1^I\ -\ \frac{\langle V_{gm}\rangle_1^I}{2} - \langle x_1\rangle_1^I\ V_{DC}\right)$$

Update: Protection Devices Control Elements

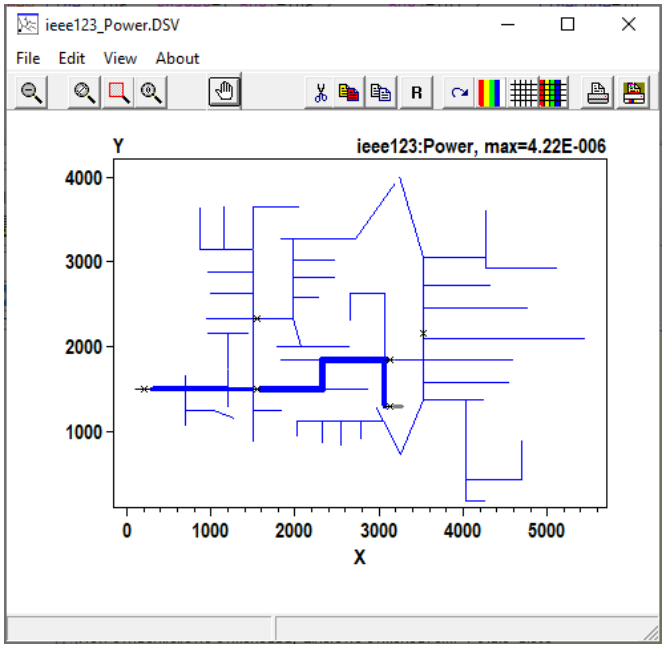
Properties *State* and *Normal* added to Fuses, Reclosers and Relays

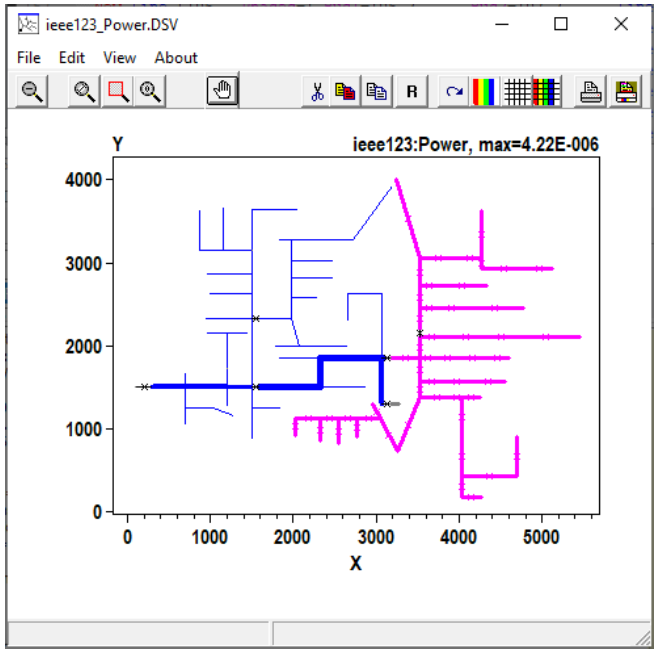
- *State*: actual state of the device
- *Normal*: normal state of the device
  - If not specified, defaults to *State*
- Simulates manual control of the devices.
- State change is **immediately** applied (no need for solving a power-flow)
- *Action* property has been deprecated and currently redirects to *State.*



Action property for SwtControls is still functional. To be used for a time delayed action (requires a power flow solution). For an instantaneous action, use *State.*

Properties *State* and *Normal* added to Fuses, Reclosers and Relays

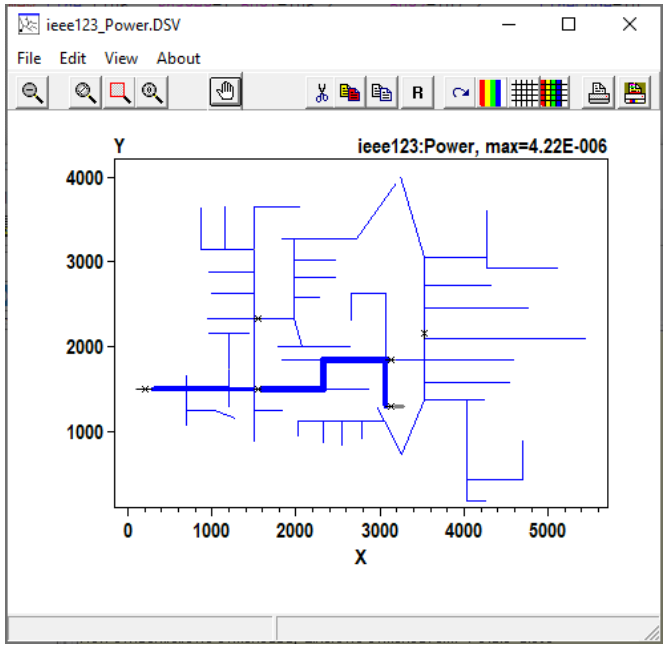





```
Compile "C:\Program
Files\OpenDSS\IEEETestCases\123Bus\IEEE123Master.dss"
Buscoords Buscoords.dat


! Explicitly defining lines modeling switches as switches
Edit Line.Sw1 switch=True
Edit Line.Sw2 switch=True
Edit Line.Sw3 switch=True
Edit Line.Sw4 switch=True
Edit Line.Sw5 switch=True
Edit Line.Sw6 switch=True
Edit Line.Sw7 switch=True
Edit Line.Sw8 switch=True


! Add SwtControl to Switches
New SwtControl.Sw1 SwitchedObj=Line.Sw1 SwitchedTerm=1 State=Close
New SwtControl.Sw2 SwitchedObj=Line.Sw2 SwitchedTerm=1 State=Close
New SwtControl.Sw3 SwitchedObj=Line.Sw3 SwitchedTerm=1 State=Close
New SwtControl.Sw4 SwitchedObj=Line.Sw4 SwitchedTerm=1 State=Close
New SwtControl.Sw5 SwitchedObj=Line.Sw5 SwitchedTerm=1 State=Close
New SwtControl.Sw6 SwitchedObj=Line.Sw6 SwitchedTerm=1 State=Close
New SwtControl.Sw7 SwitchedObj=Line.Sw7 SwitchedTerm=1 State=Open
New SwtControl.Sw8 SwitchedObj=Line.Sw8 SwitchedTerm=1 State=Open

! Place EM on Feeder Head.
New EnergyMeter.m1 element=transformer.reg1a
Set MarkSwitches=Yes

! Plot circuit - original configuration
Plot circuit

! Isolate system downstream Sw4
Edit SwtControl.Sw4 State=Open
ReprocessBuses ! ReprocessBuses to show isolated area in circuit plot.
Plot circuit
```

Properties *State* and *Normal* added to Fuses, Reclosers and Relays

- Whenever controls are reset, the devices state return to *Normal*

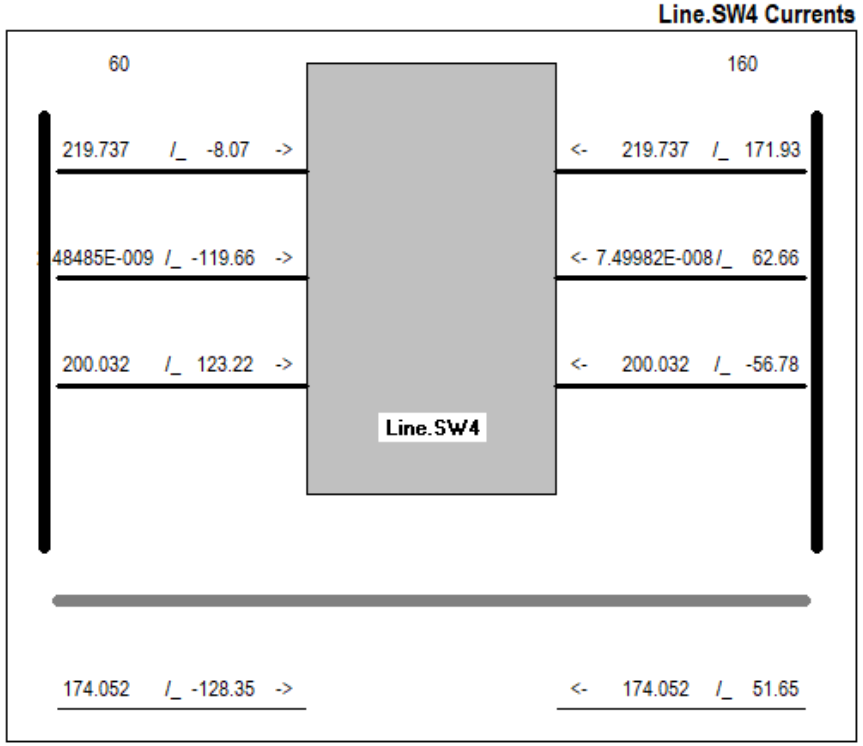



```
! Option 1: Resetting specific element to Normal Set
Edit SwtControl.Sw4 Reset=True
ReprocessBuses
Plot circuit


! Option 2: "Change" solution mode - automatically
reset all
Set mode=snapshot
ReprocessBuses
Plot circuit
```

State and *Normal* properties for Fuses

- Fuse Control Element (CE) operates independently in each phase
- Number of phases depends on the *switchedobj*
- *State* and *Normal* properties reflect this modeling

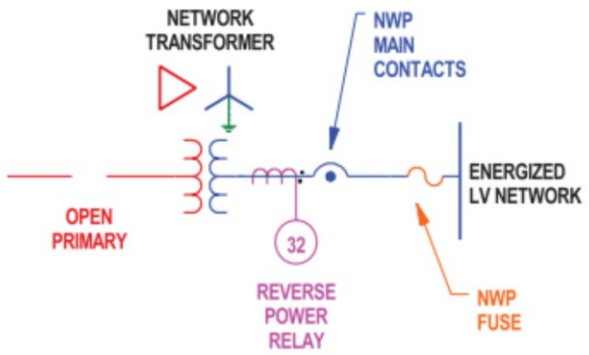
```
Compile "C:\Program
Files\OpenDSS\IEEETestCases\123Bus\IEEE123Master.dss"

! Explicitly defining lines modeling switches as switches
Edit Line.Sw4 switch=True

! Add Sample Fuse
New Fuse.Sw4 monitoredobj=Line.Sw4 state=[closed closed closed]
~ fusecurve=Tlink RatedCurrent=1000

Edit Fuse.Sw4 state=[closed, open, closed]
Solve
Visualize currents "Line.SW4"
```

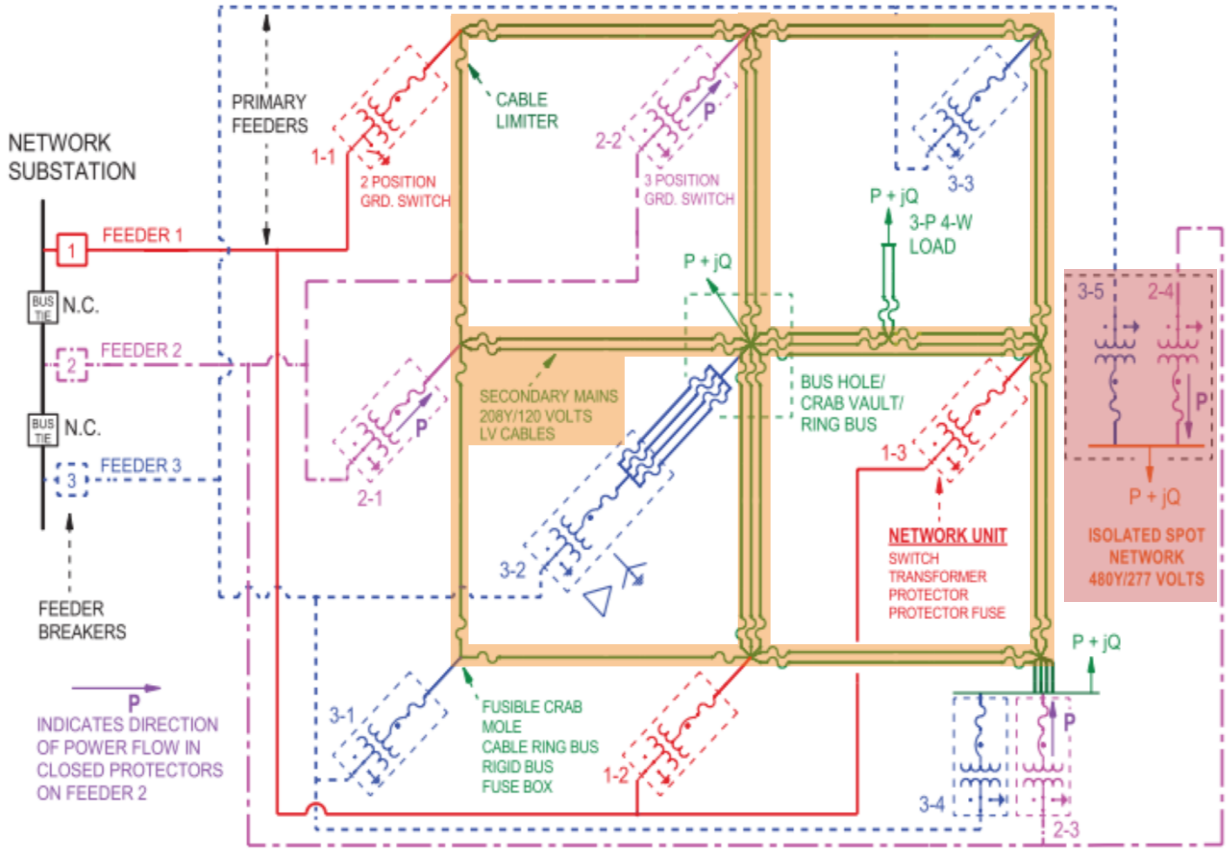


New Feature: Directional Overcurrent (DOC) Relay

The Directional Overcurrent (DOC) Element

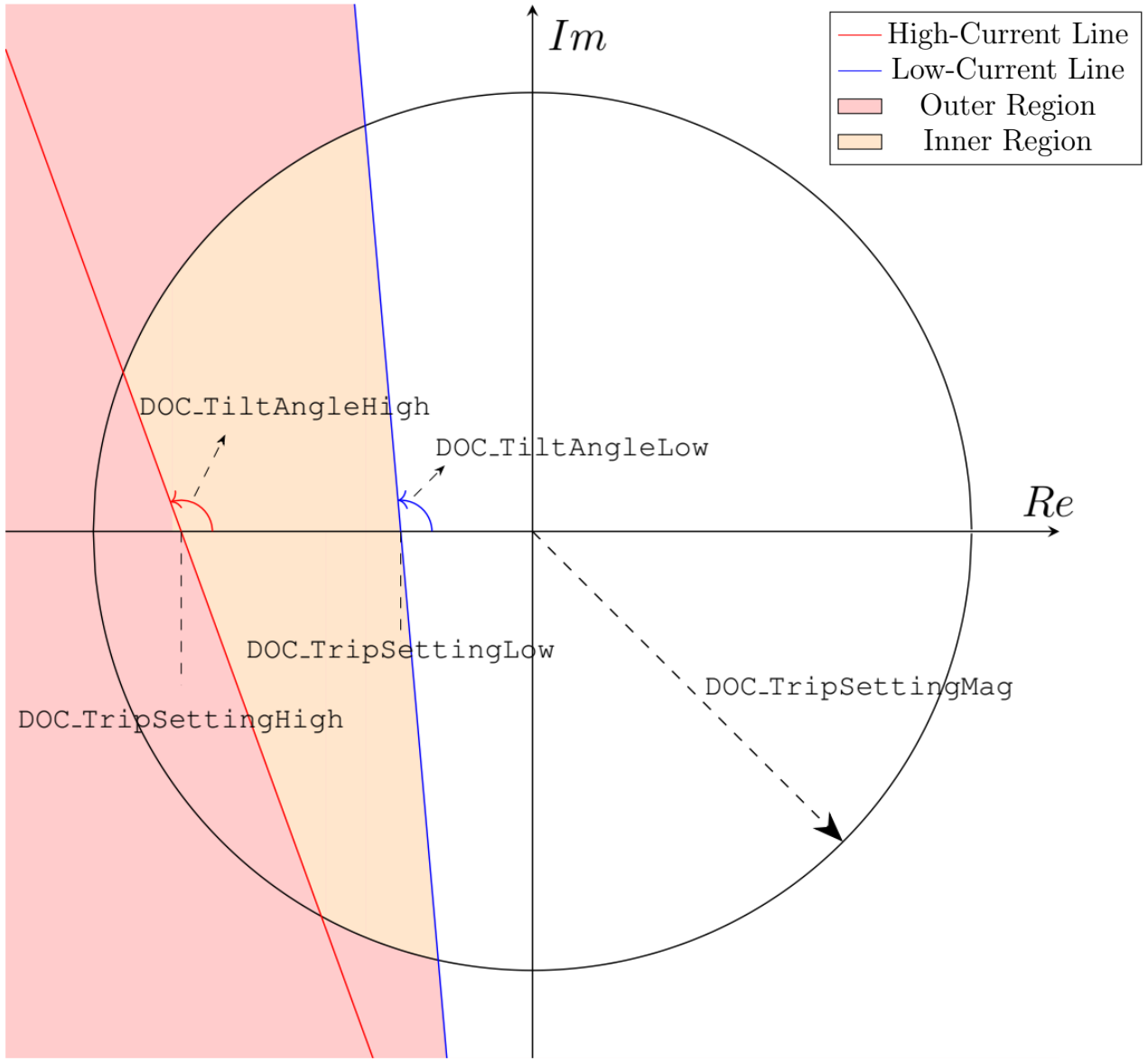
- Mainly intended for modeling Network Protectors (NWP), but not limited.
- Breaker + Relay + Fuse (Backup)
- Main Functionalities:
  1. Protects LV network from faults on the primary
  2. Automatically open and protects the service transformer when a feeder is out of service
  3. Automatically reclose the circuit breaker if P and Q flows are expected to be in forward direction
  4. Close on “dead” network





Source: [EPRI Underground Distribution Systems Reference Book: 2020 Revision. EPRI, Palo Alto, CA: 2020. 3002018091.]

Generic Trip Characteristic

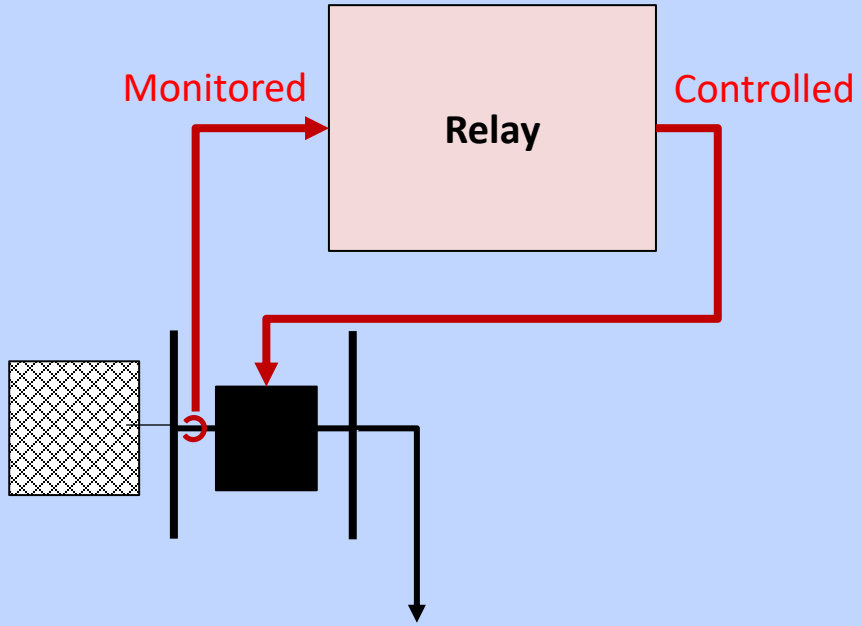


OpenDSS – Relay Object		
	Property Name	**Default Value**
Trip Characteristic	DOC_TiltAngleLow	90
	DOC_TripSettingLow	0
	DOC_TiltAngleHigh	90
	DOC_TripSettingHigh	-1 (deactivated)
	DOC_TripSettingMag	-1 (deactivated)
	DOC_P1Blocking	True
Trip Time Outer Region	Delay*	0
	PhaseCurve*	None (deactivated)
	PhaseTrip*	1
	TDPhase*	1
Trip Time Inner Region	DOC_DelayInner	-1 (deactivated)
	DOC_PhaseCurveInner	None (deactivated)
	DOC_PhaseTripInner	1
	DOC_TDPhaseInner	1

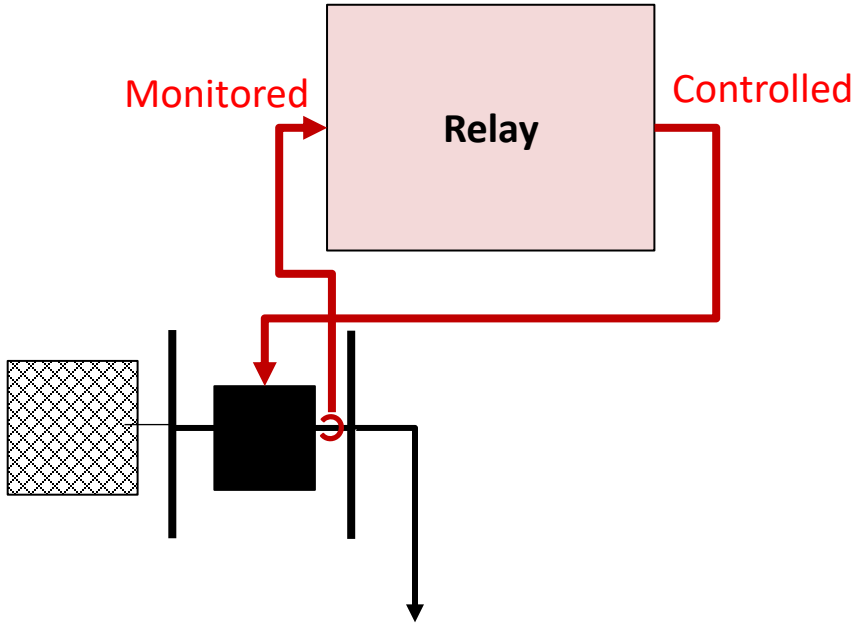
*: previously existing property

Modeling

- For Tripping on Reverse Power-Flow (E.g., NWP)
  - Set monitored *terminal* to that where forward active power flow is expected (power flowing **into** the terminal)



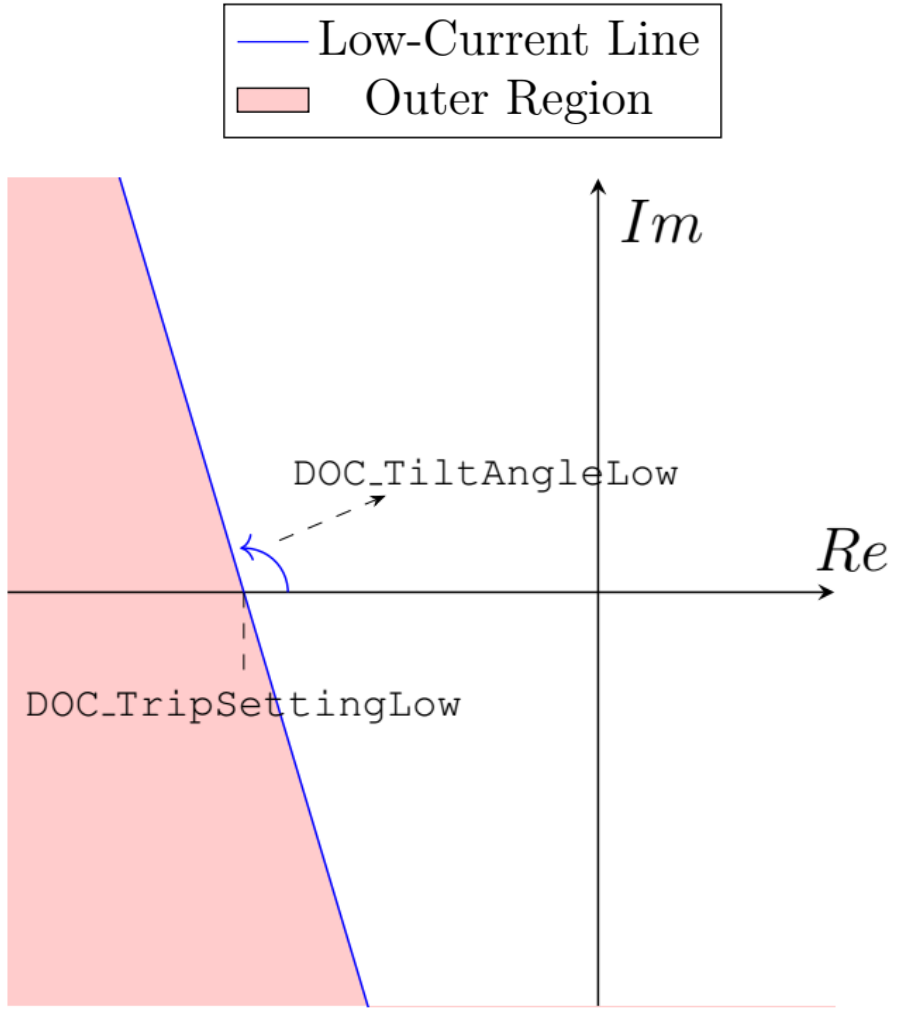
- For Tripping on Forward Power-Flow
  - Set monitored *terminal* to that where reverse active power flow is expected (power flowing **out** the terminal)



Implementing Standard Functions IEEE C57.12.44-2014

- Sensitive Trip
  - Trip Setting of 5 Amps
  - Instantaneous Trip

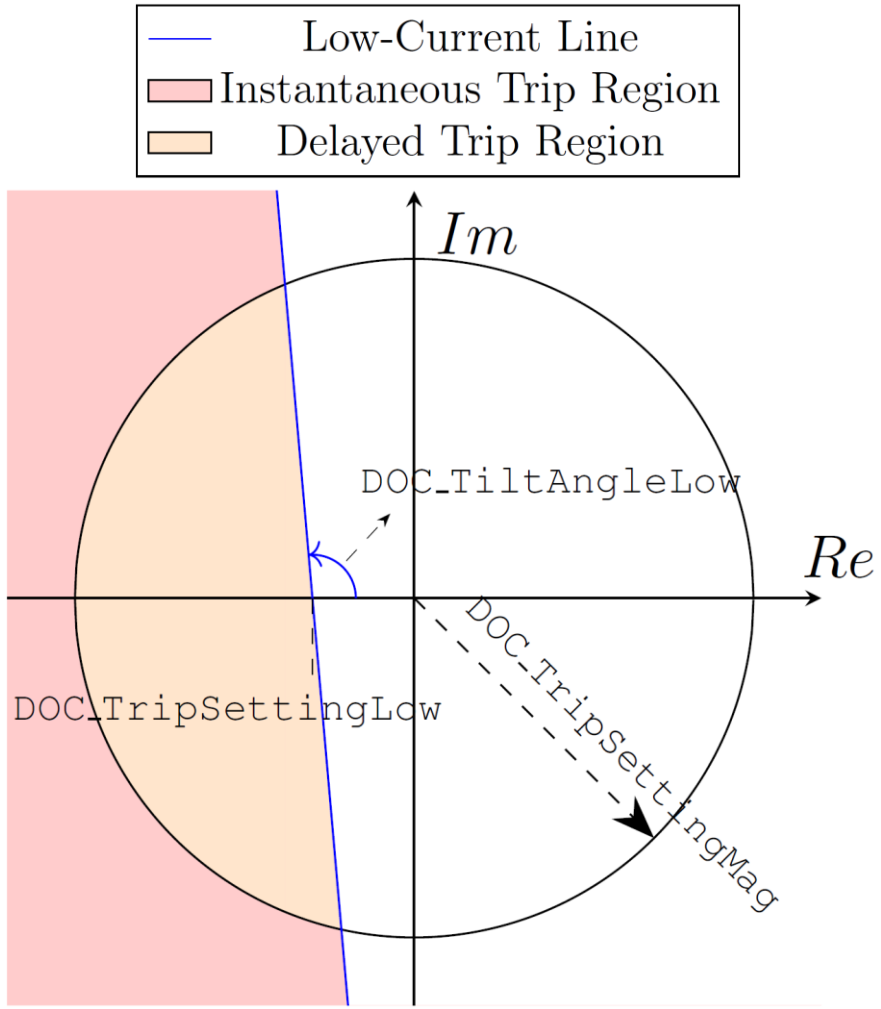
```
! DOC Relay with Standard Sensitive Trip Function

New Relay.MyDOCRelay monitoredobj=Line.MyMonitoredLine
~ type=DOC DOC_TiltAngleLow=90 DOC_TripSettingLow=5
delay=0
```



Implementing Standard Functions IEEE C57.12.44-2014

- Time Delay Trip
  - Resistive Delayed Trip Setting: 5A
  - Time Delay: 2.5 minutes
  - Instantaneous Trip Setting: 150A

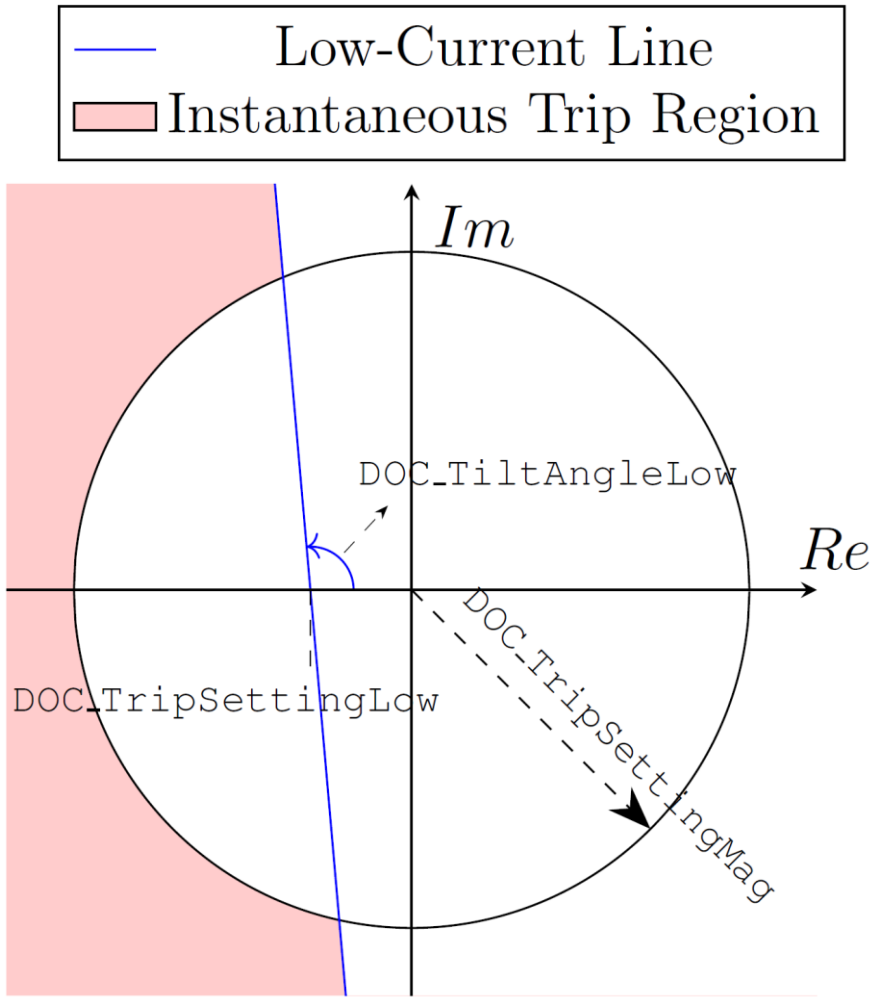
```
! DOC Relay with Standard Time Delay Trip Function

New Relay.MyDOCRelay
~monitoredobj=Line.MyMonitoredLine type=DOC
~DOC_TiltAngleLow=90 DOC_TripSettingLow=5
~DOC_TripSettingMag=40
~delay=0 DOC_DelayInner=150 ! Delay Settings
```



Implementing Standard Functions IEEE C57.12.44-2014

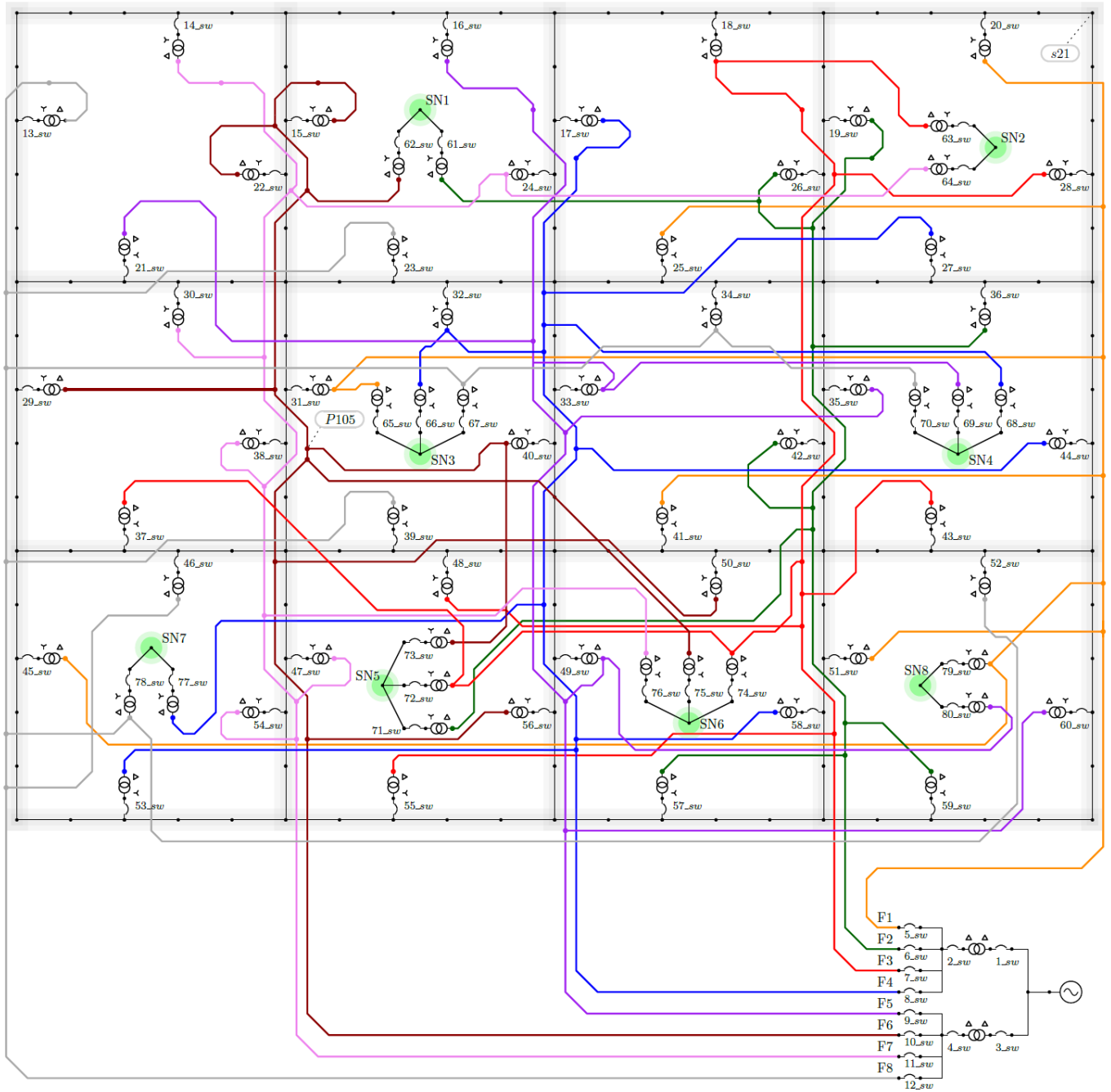
- Insensitive Trip
  - Resistive Delayed Trip Setting: 5A
  - Instantaneous Trip Setting: 40A

```
! DOC Relay with Standard Insensitive Trip Function

New Relay.MyDOCRelay
monitoredobj=Line.MyMonitoredLine type=DOC
~ DOC_TiltAngleLow=90 DOC_TripSettingLow=5
~ DOC_TripSettingMag=40
~ delay=0 DOC_DelayInner=-1 ! Delay Settings. Trip
in inner region is off
```



Example – IEEE 390 Buses Test Case

- 1 LV grid operated at 216 V , fed by 48 network transformers.
- 8 LV spot networks operated at 480 V , fed by 20 network transformers.
- 8 MV feeders are fed by 2 substation transformers

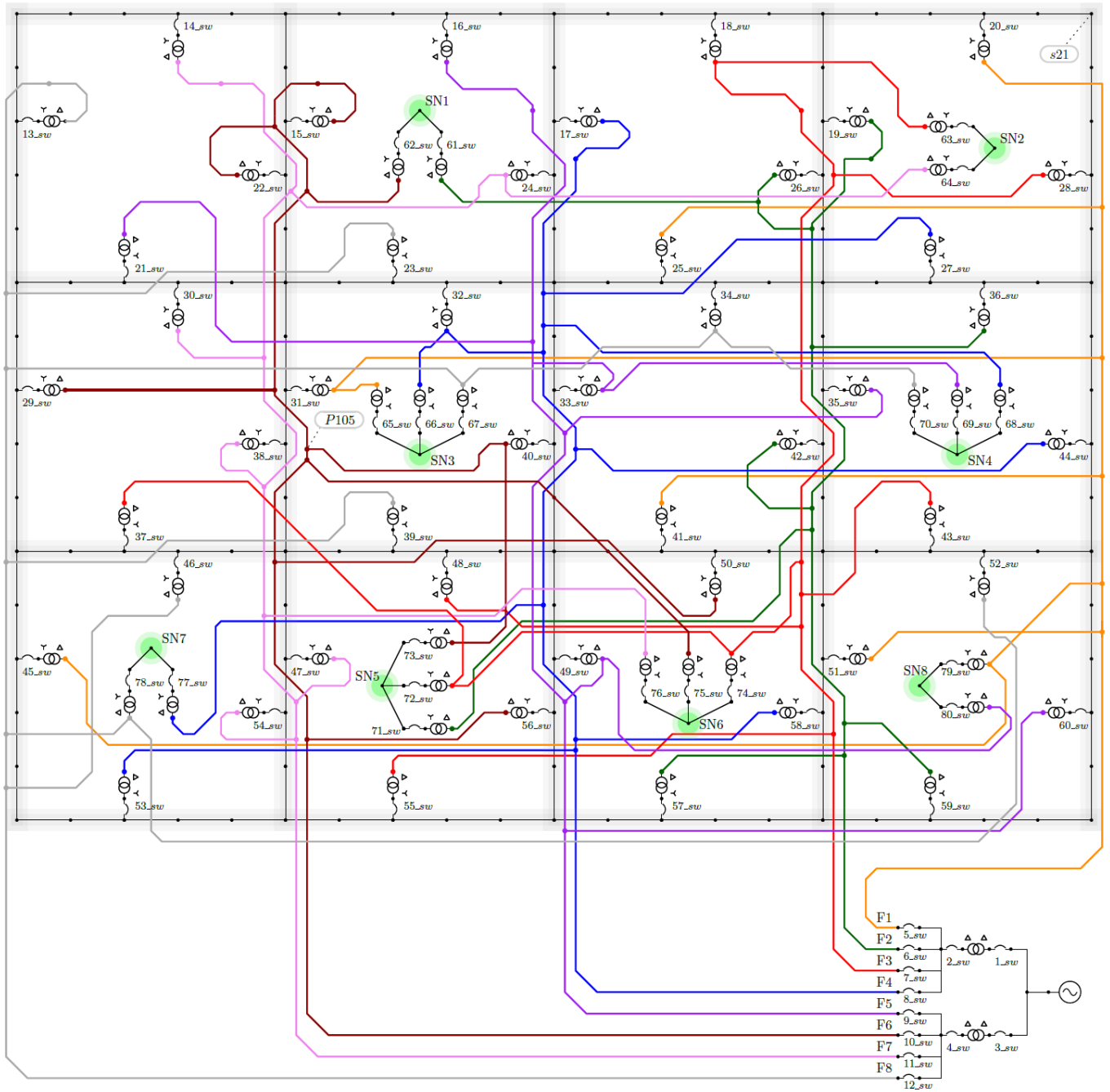


https://cmte.ieee.org/pes-testfeeders/resources/

Example – IEEE 390 Buses Test Case

- Added NWPs:
  - All 68 NWPs in sensitive trip mode with instantaneous trip setting set to 0.15% of their primary CT rating

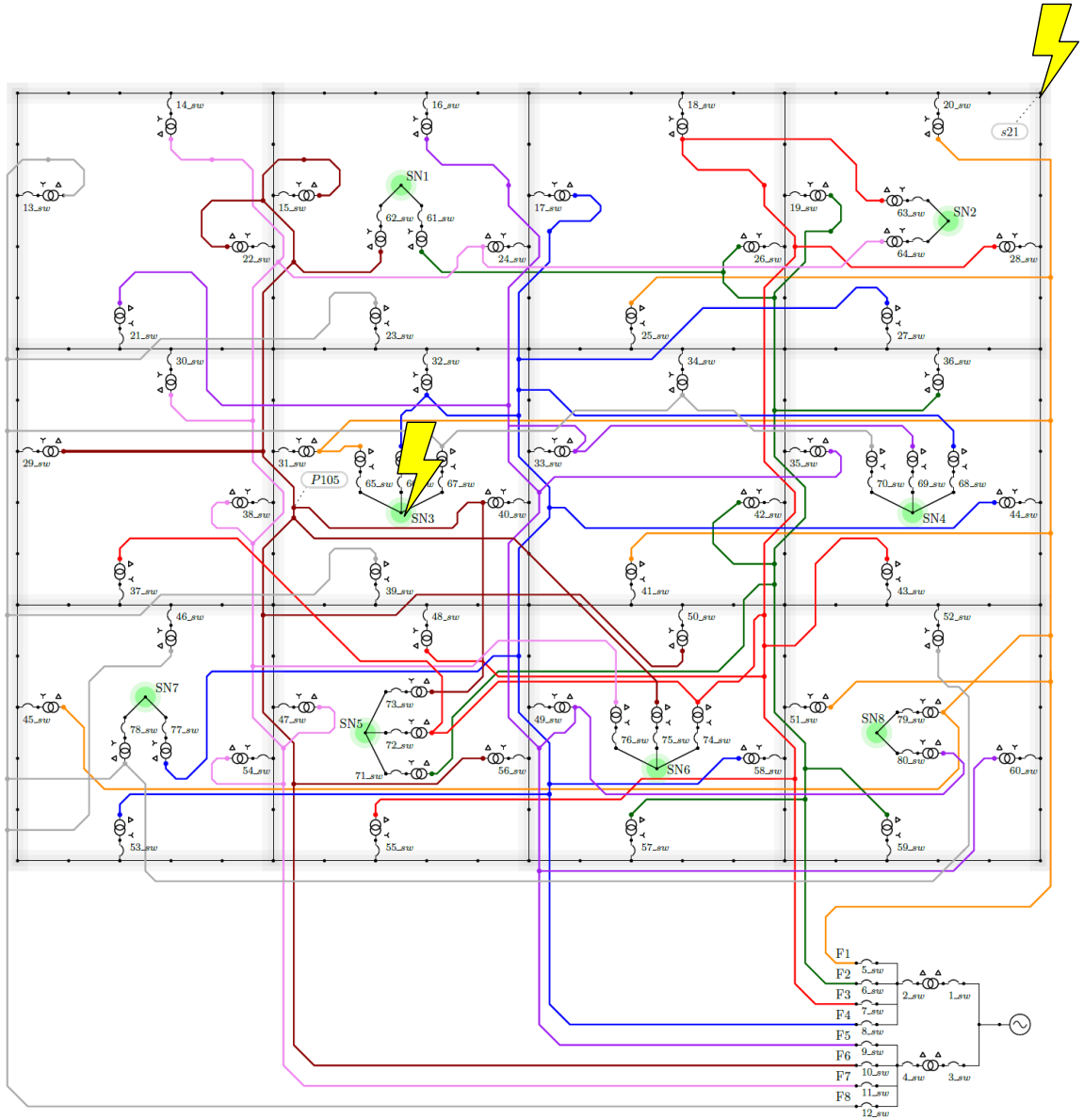
```
var @nwp_inst_ct_pu = 0.0015

new relay.37_sw monitoredobj=line.37_sw
~monitoredterm=1 type=doc
~doc_tiltanglelow=95
~doc_tripsettinglow=(3500 @nwp_inst_ct_pu *)
~delay=@nwp_time_delay ! Sensitive Trip
```



Example - Faults on LV Network

```
! SLG Fault on Bus S21 - Top Right Corner of
LV network
New Fault.fault bus1=S21.1 bus2=S21.0
~ phases=1 r=0.0001

Solve
Show eventlog
```

```
! SLG Fault on Bus S203 (Spot Network 3)

New Fault.fault bus1=S203.1 bus2=S203.0
~ phases=1 r=0.0001

Solve
Show eventlog
```

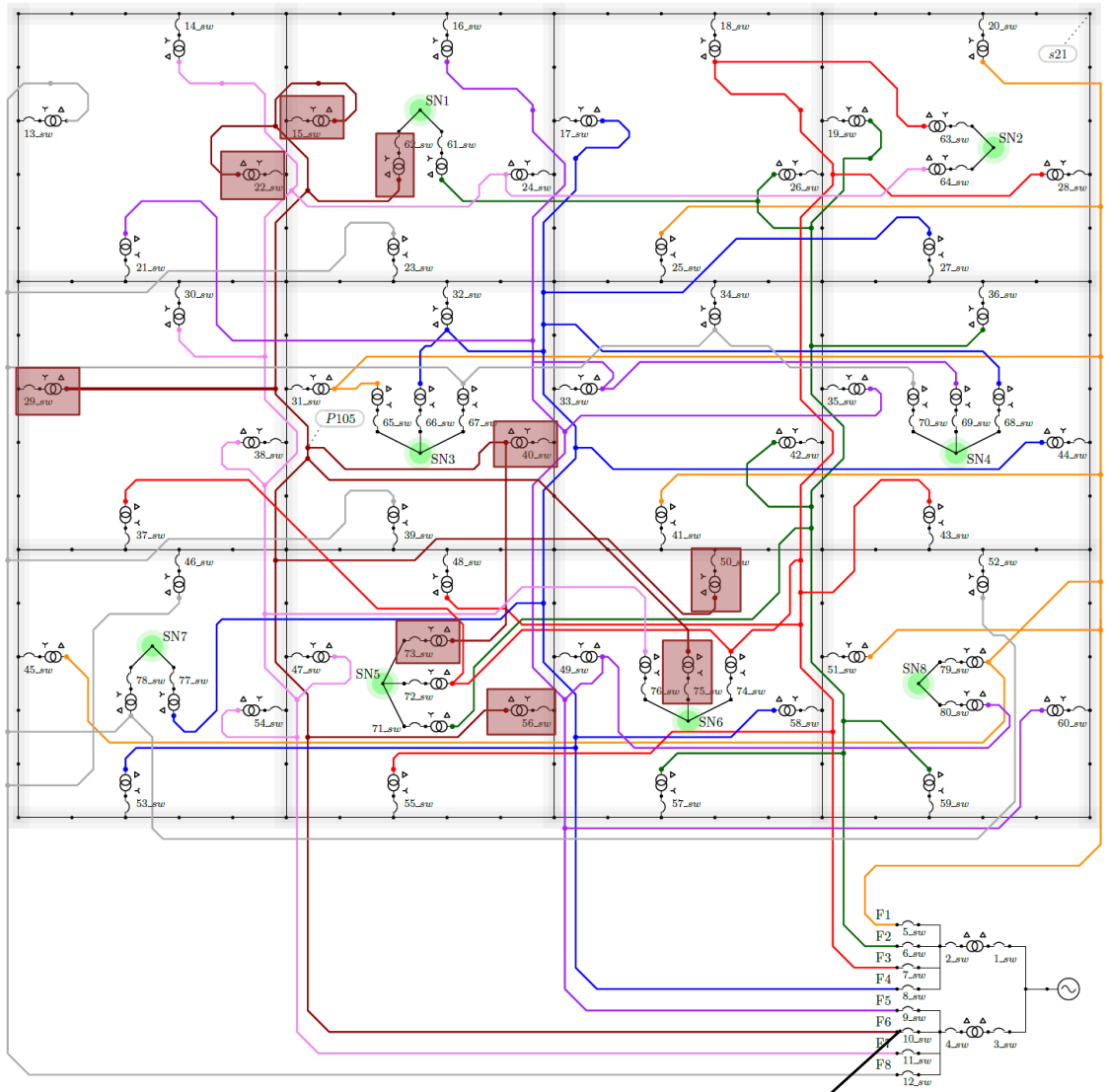


Example – Feeder 6 Out of Service

```
! Feeder 6 Out of Service
Open Line.10_sw ! Open feeder breaker

Solve
Show eventlog
```

```
Hour=0, Sec=0, ControlIter=1, Element=Relay.75_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.56_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.50_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.40_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.29_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=2, Element=Relay.22_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=2, Element=Relay.15_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=3, Element=Relay.73_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=4, Element=Relay.62_sw, Action=OPENED ON DOC & LOCKED OUT
```

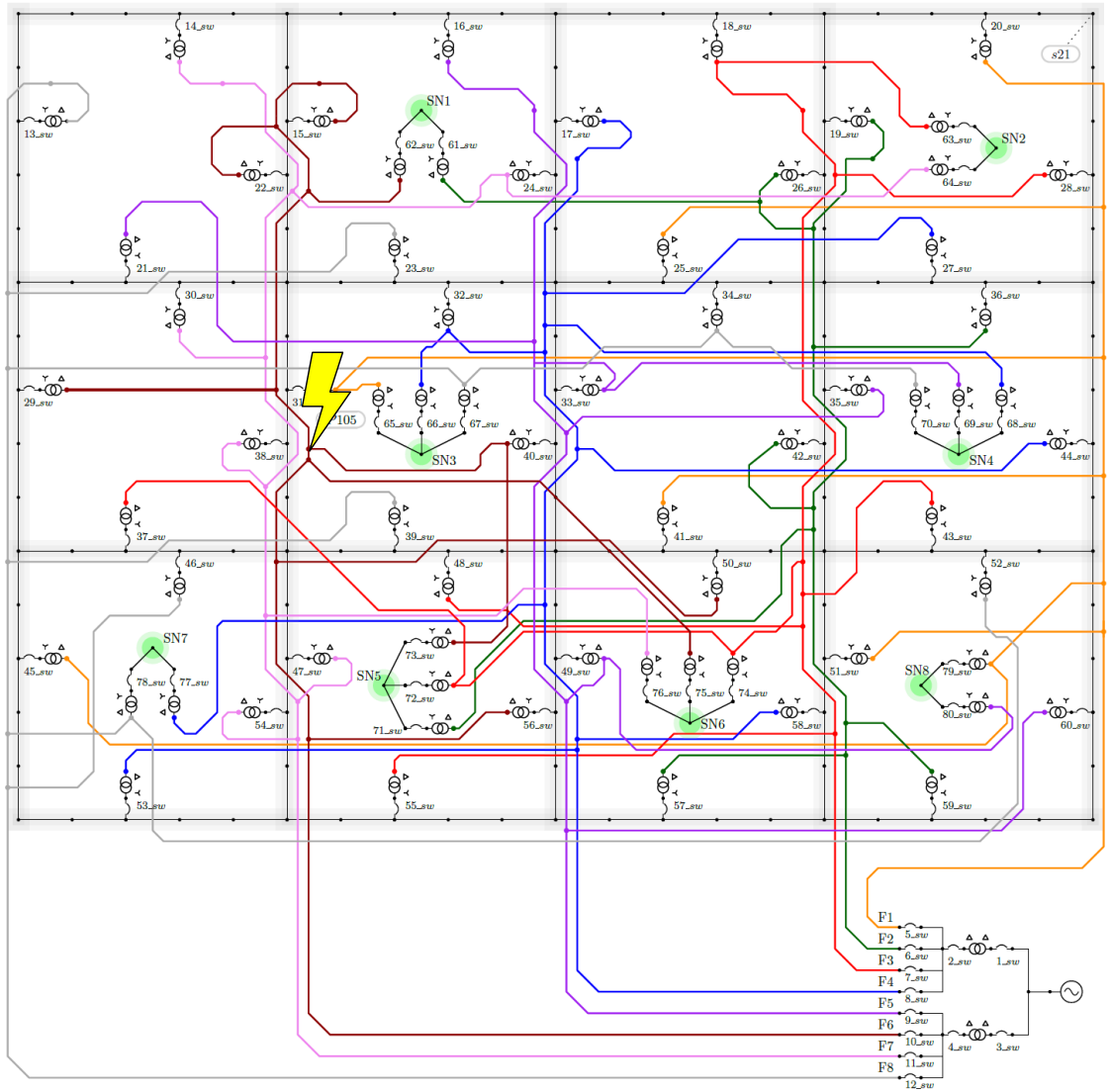
Cascading effect



Feeder 6 Breaker Opened

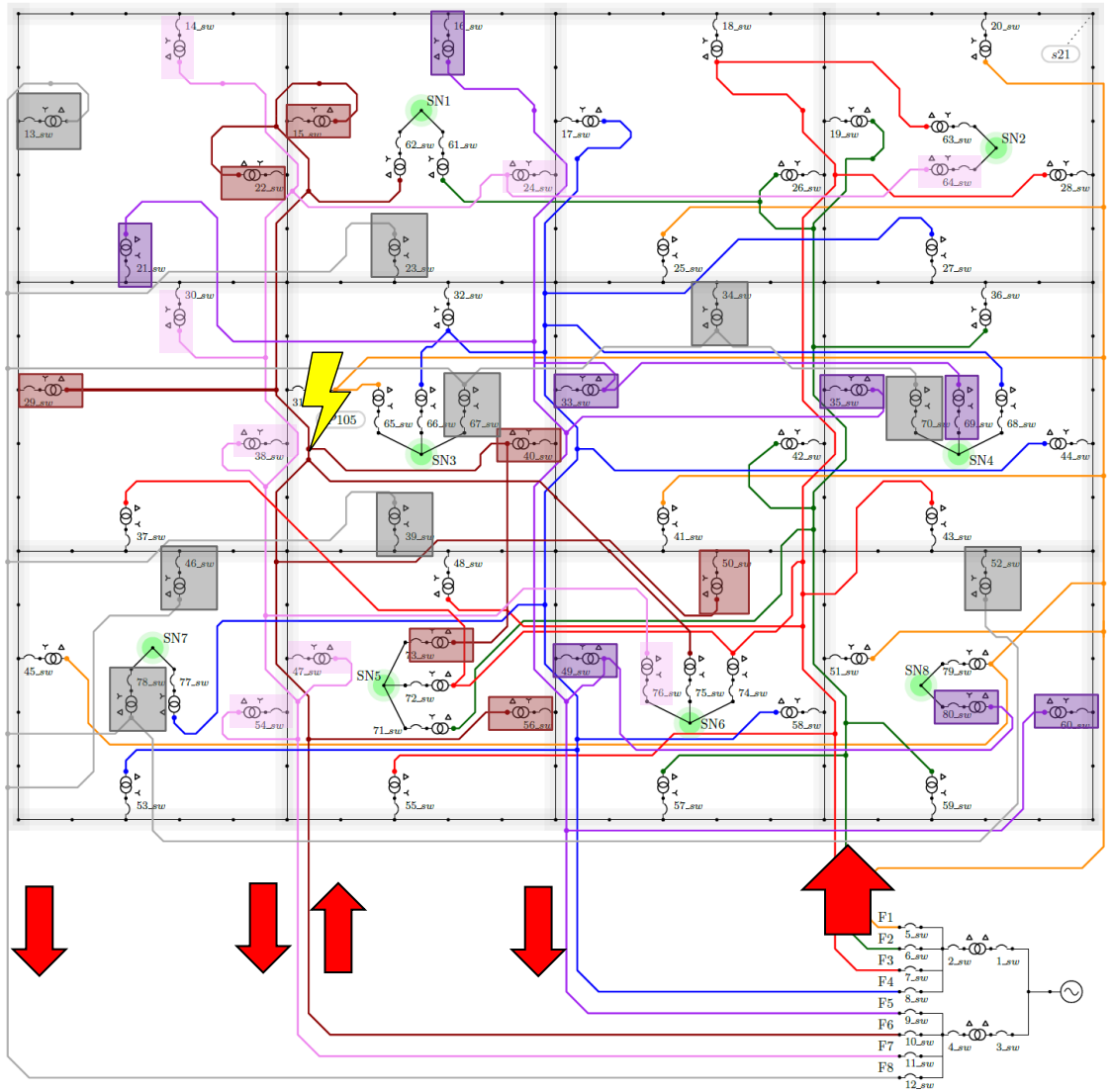
Example – LL Fault on Feeder 6

```
! Adding OC relays to feeder breakers
New Relay.F6 monitoredobj=Line.10_sw
~ type=current delay=0.1
~ PhaseTrip=20000

! LL Fault on Bus P105 (Feeder 6)
New Fault.fault bus1=p105.1 bus2=p105.2 phases=1
~ r=0.0001

Set controlmode=event
Solve
Show eventlog
```



Example – LL Fault on Feeder 6

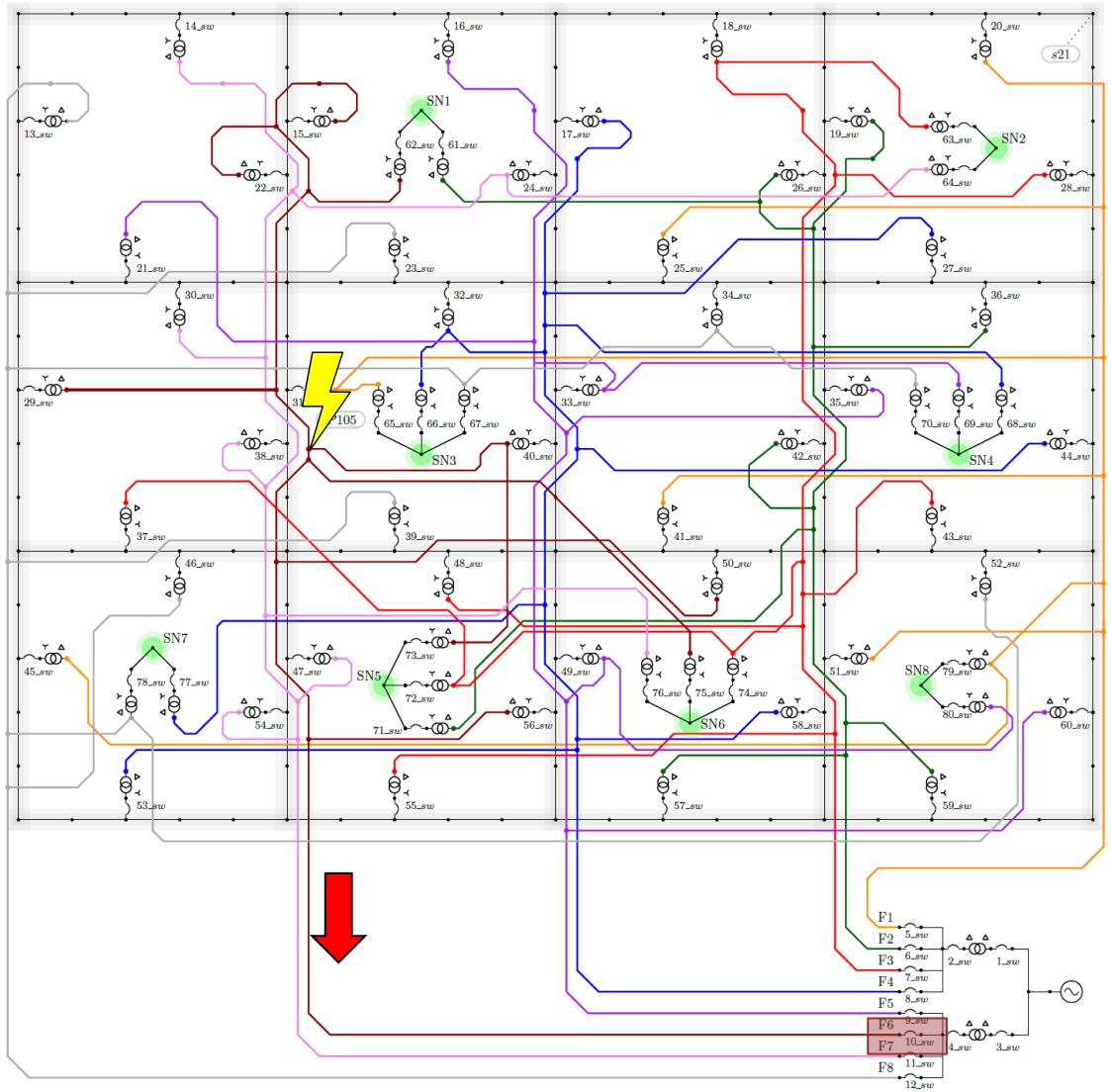
Hour=0, Sec=0, ControlIter=1, Element=Relay.80_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.78_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.76_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.73_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.70_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.69_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.67_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.64_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.60_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.56_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.54_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.52_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.50_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.49_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.47_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.46_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.39_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.38_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.35_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.34_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.33_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.30_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.24_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.23_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.21_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.16_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.14_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.13_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=2, Element=Relay.40_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=2, Element=Relay.29_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=2, Element=Relay.22_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=2, Element=Relay.15_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0.1, ControlIter=3, Element=Relay.feeder_6, Action=OPENED ON PH & LOCKED OUT
Hour=0, Sec=0.1, ControlIter=3, Element= , Action=PHASE TARGET
Hour=0, Sec=0.1, ControlIter=4, Element=Relay.75_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0.1, ControlIter=4, Element=Relay.62_sw, Action=OPENED ON DOC & LOCKED OUT



Feeders 4, 5 and 6 back-feed the fault

Example – LL Fault on Feeder 6

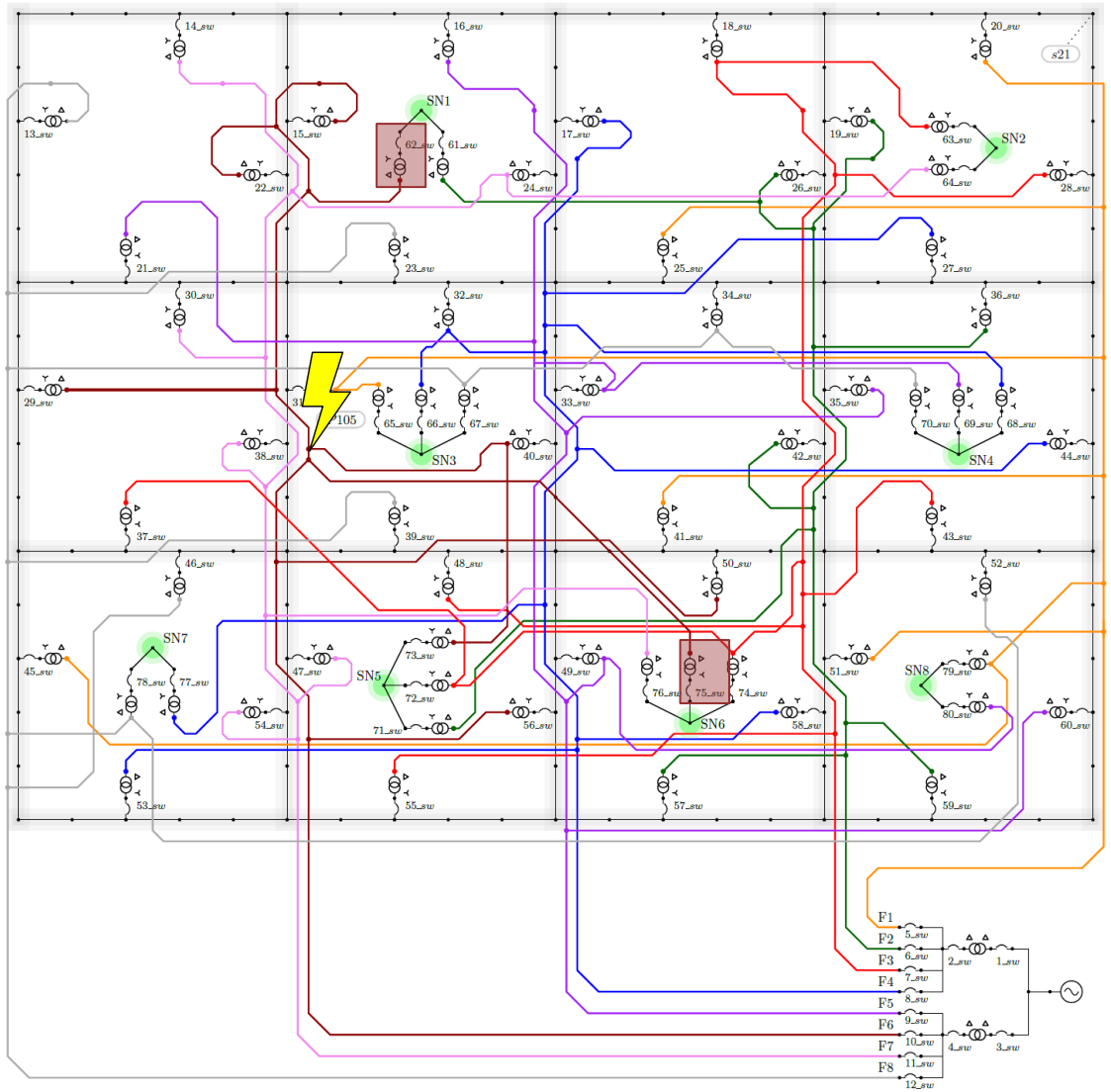
Hour=0, Sec=0, ControlIter=1, Element=Relay.80_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.78_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.76_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.73_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.70_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.69_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.67_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.64_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.60_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.56_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.54_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.52_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.50_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.49_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.47_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.46_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.39_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.38_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.35_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.34_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.33_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.30_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.24_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.23_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.21_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.16_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.14_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.13_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=2, Element=Relay.40_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=2, Element=Relay.29_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=2, Element=Relay.22_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=2, Element=Relay.15_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0.1, ControlIter=3, Element=Relay.feeder_6, Action=OPENED ON PH & LOCKED OUT
Hour=0, Sec=0.1, ControlIter=3, Element= , Action=PHASE TARGET
Hour=0, Sec=0.1, ControlIter=4, Element=Relay.75_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0.1, ControlIter=4, Element=Relay.62_sw, Action=OPENED ON DOC & LOCKED OUT



Feeder 6 Breaker Trips with 0.1s delay

Example – LL Fault on Feeder 6

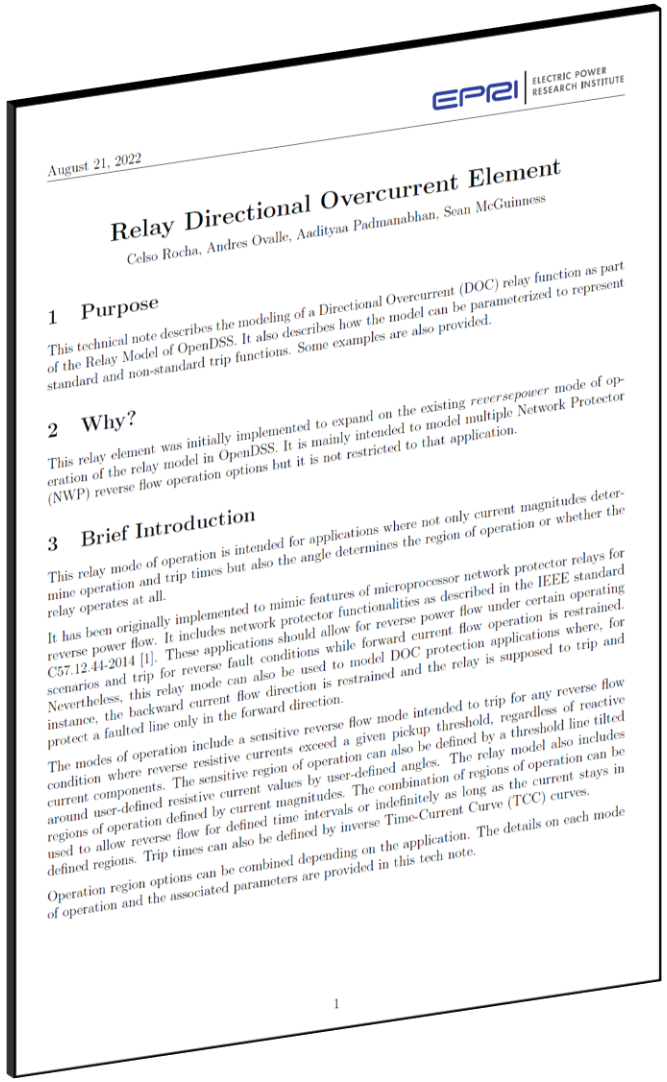
Hour=0, Sec=0, ControlIter=1, Element=Relay.80_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.78_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.76_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.73_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.70_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.69_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.67_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.64_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.60_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.56_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.54_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.52_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.50_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.49_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.47_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.46_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.39_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.38_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.35_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.34_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.33_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.30_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.24_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.23_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.21_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.16_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.14_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=1, Element=Relay.13_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=2, Element=Relay.40_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=2, Element=Relay.29_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=2, Element=Relay.22_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0, ControlIter=2, Element=Relay.15_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0.1, ControlIter=3, Element=Relay.feeder_6, Action=OPENED ON PH & LOCKED OUT
Hour=0, Sec=0.1, ControlIter=3, Element= , Action=PHASE TARGET
Hour=0, Sec=0.1, ControlIter=4, Element=Relay.75_sw, Action=OPENED ON DOC & LOCKED OUT
Hour=0, Sec=0.1, ControlIter=4, Element=Relay.62_sw, Action=OPENED ON DOC & LOCKED OUT



Remaining NWPs on F6 Trip

Where do I go from here?

- Technical note available at your local OpenDSS installation Doc folder (*C:\Program Files\OpenDSS\Doc*)
  - DOC Relay
  - Other relevant technical notes/docs:
    - Python to OpenDSS Interface for Modeling Control Systems
    - OpenDSS COM CtrlQueue Interface
    - OpenDSS Manual
    - Change to OpenDSS for the DMS Controls Sandbox End of Time Step Cleanup
    - OpenDSS Custom Scripting
- OpenDSS Virtual Training 2021 – Controls in OpenDSS
- Official Forum at SourceForge
  - https://sourceforge.net/p/electricdss/discussion/



EPRI | ELECTRIC POWER RESEARCH INSTITUTE

August 21, 2022

Relay Directional Overcurrent Element

Celso Rocha, Andres Ovalle, Aadityaa Padmanabhan, Sean McGuinness

1 Purpose

This technical note describes the modeling of a Directional Overcurrent (DOC) relay function as part of the Relay Model of OpenDSS. It also describes how the model can be parameterized to represent standard and non-standard trip functions. Some examples are also provided.

2 Why?

This relay element was initially implemented to expand on the existing *reversepower* mode of operation of the relay model in OpenDSS. It is mainly intended to model multiple Network Protector (NWP) reverse flow operation options but it is not restricted to that application.

3 Brief Introduction

This relay mode of operation is intended for applications where not only current magnitudes determine operation and trip times but also the angle determines the region of operation or whether the relay operates at all.

It has been originally implemented to mimic features of microprocessor network protector relays for reverse power flow. It includes network protector functionalities as described in the IEEE standard C57.12.44-2014 [1]. These applications should allow for reverse power flow under certain operating scenarios and trip for reverse fault conditions while forward current flow operation is restrained. Nevertheless, this relay mode can also be used to model DOC protection applications where, for instance, the backward current flow direction is restrained and the relay is supposed to trip and protect a faulted line only in the forward direction.

The modes of operation include a sensitive reverse flow mode intended to trip for any reverse flow condition where reverse resistive currents exceed a given pickup threshold, regardless of reactive current components. The sensitive region of operation can also be defined by a threshold line tilted around user-defined resistive current values by user-defined angles. The relay model also includes regions of operation defined by current magnitudes. The combination of regions of operation can be used to allow reverse flow for defined time intervals or indefinitely as long as the current stays in defined regions. Trip times can also be defined by inverse Time-Current Curve (TCC) curves.

Operation region options can be combined depending on the application. The details on each mode of operation and the associated parameters are provided in this tech note.

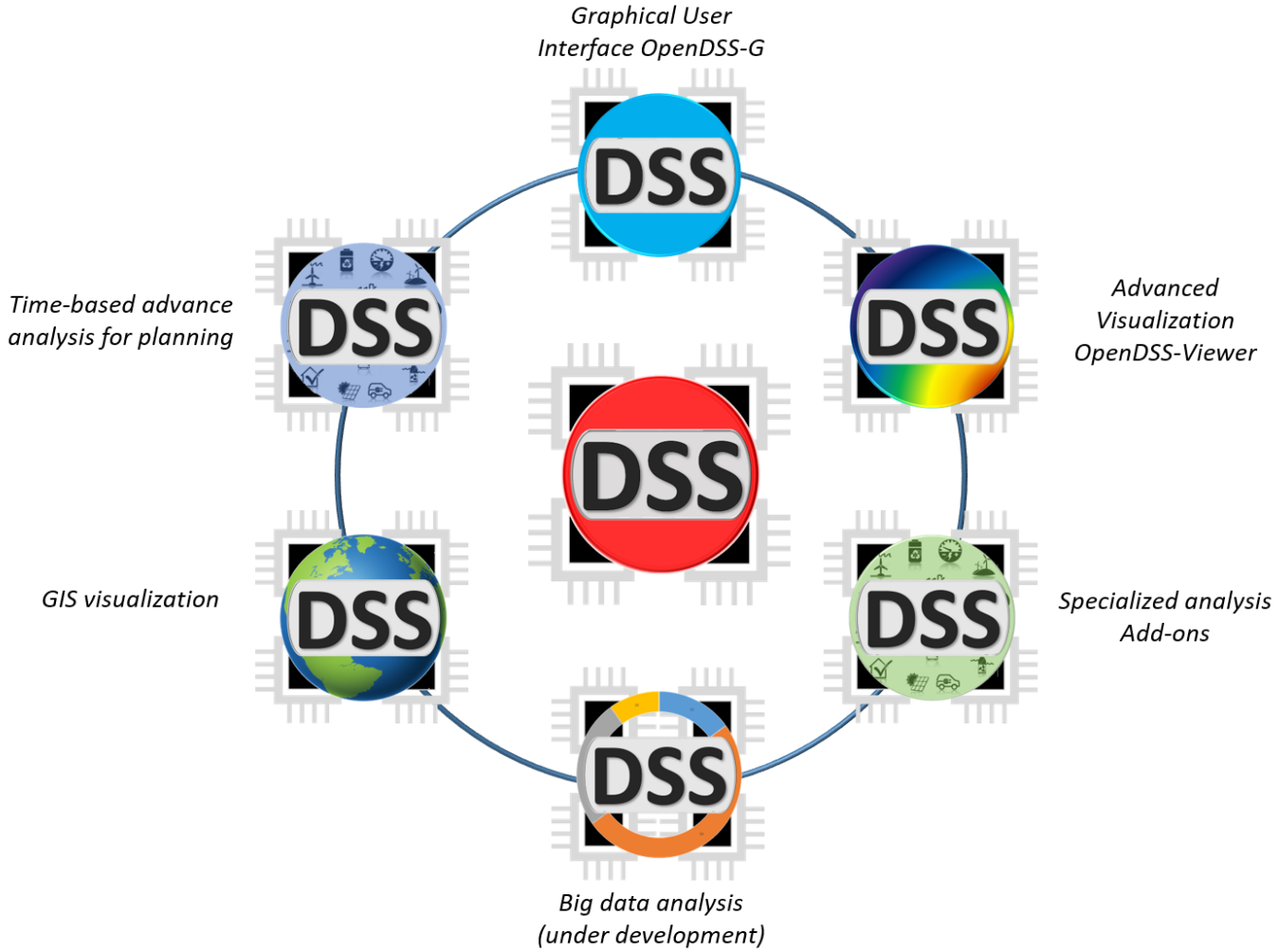
1

OpenDSS-G

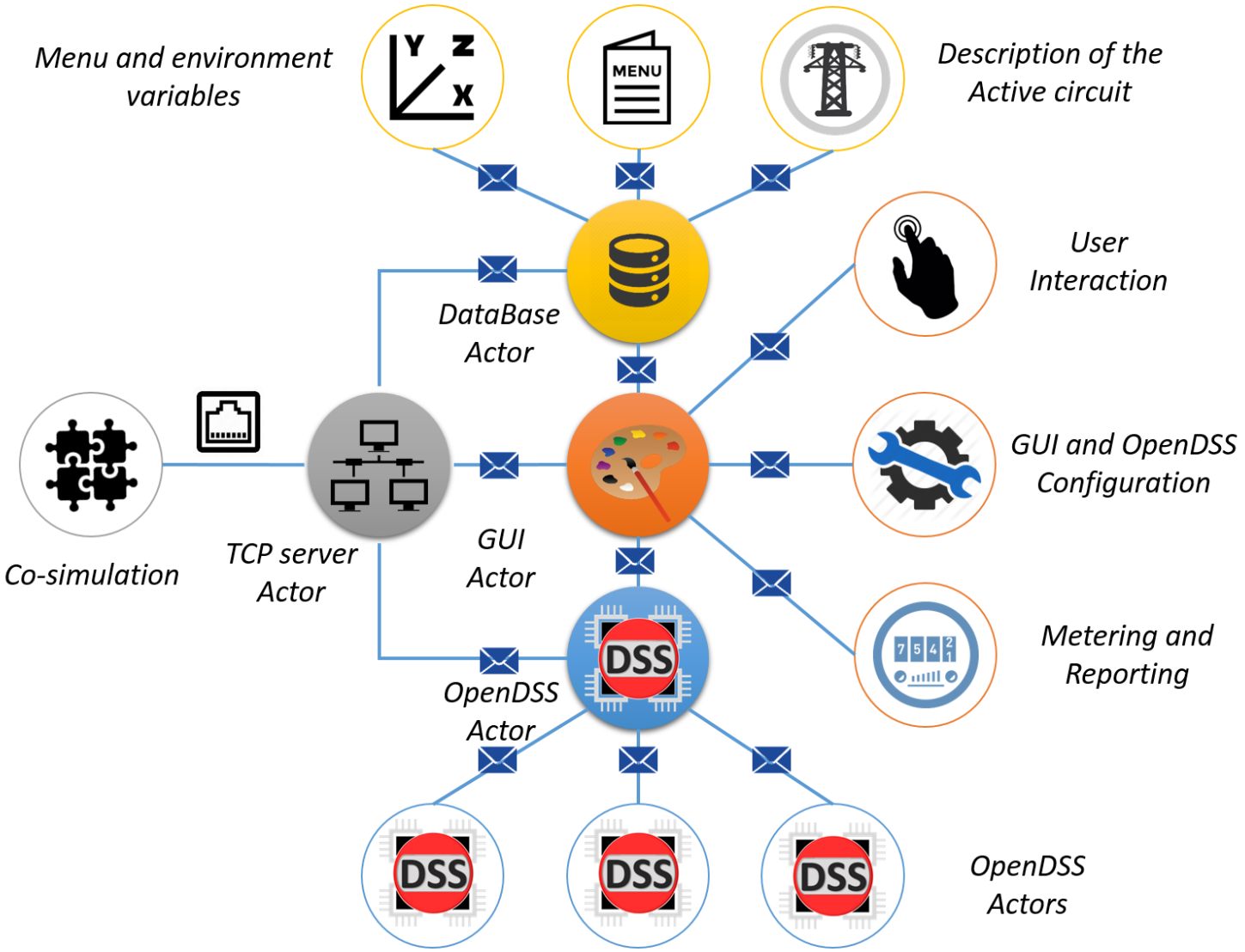
OpenDSS-G YouTube channel

https://www.youtube.com/channel/UCGe58SDH3Iq-EGvnxEOuWaQ

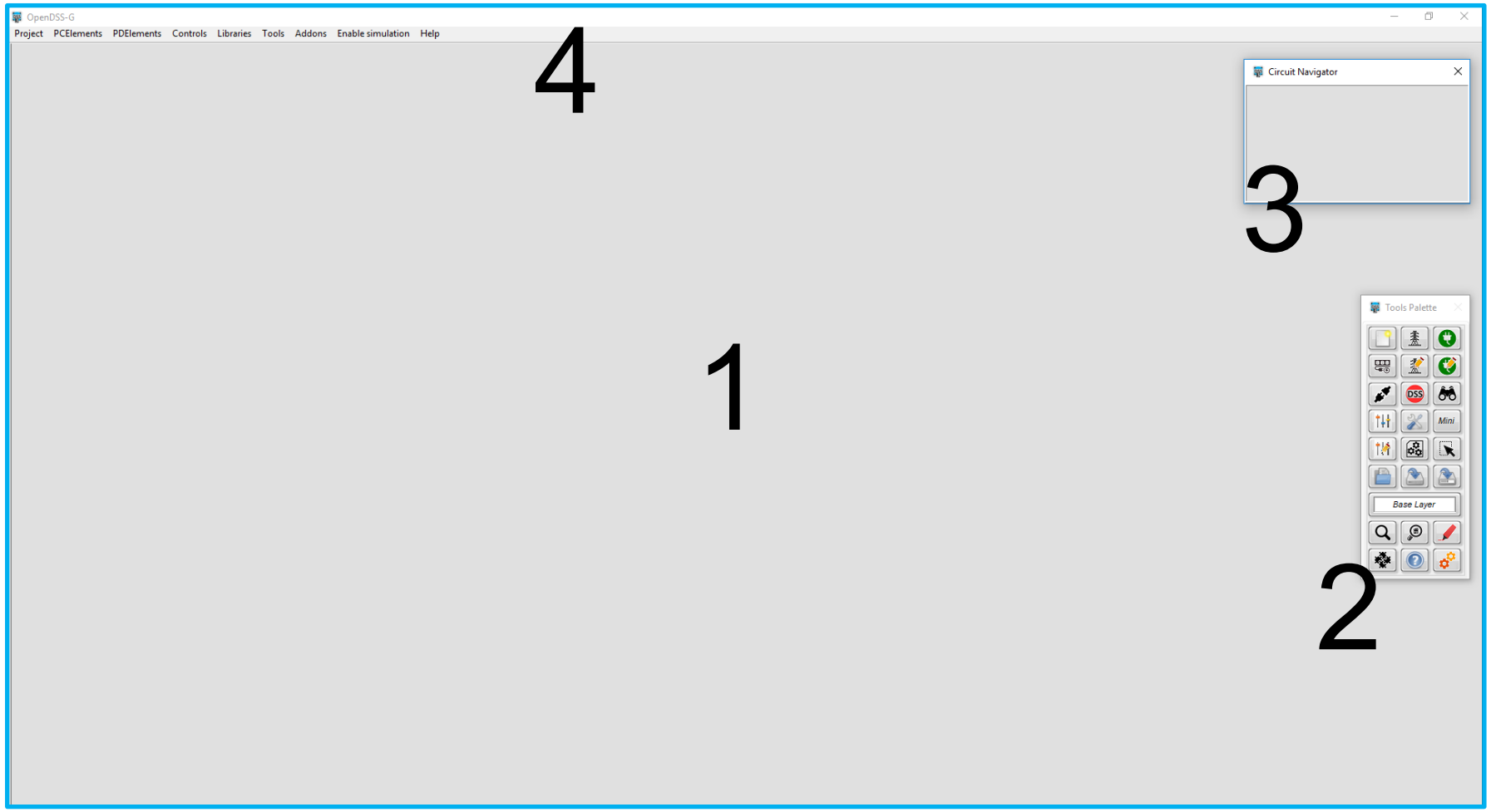
OpenDSS derivative products



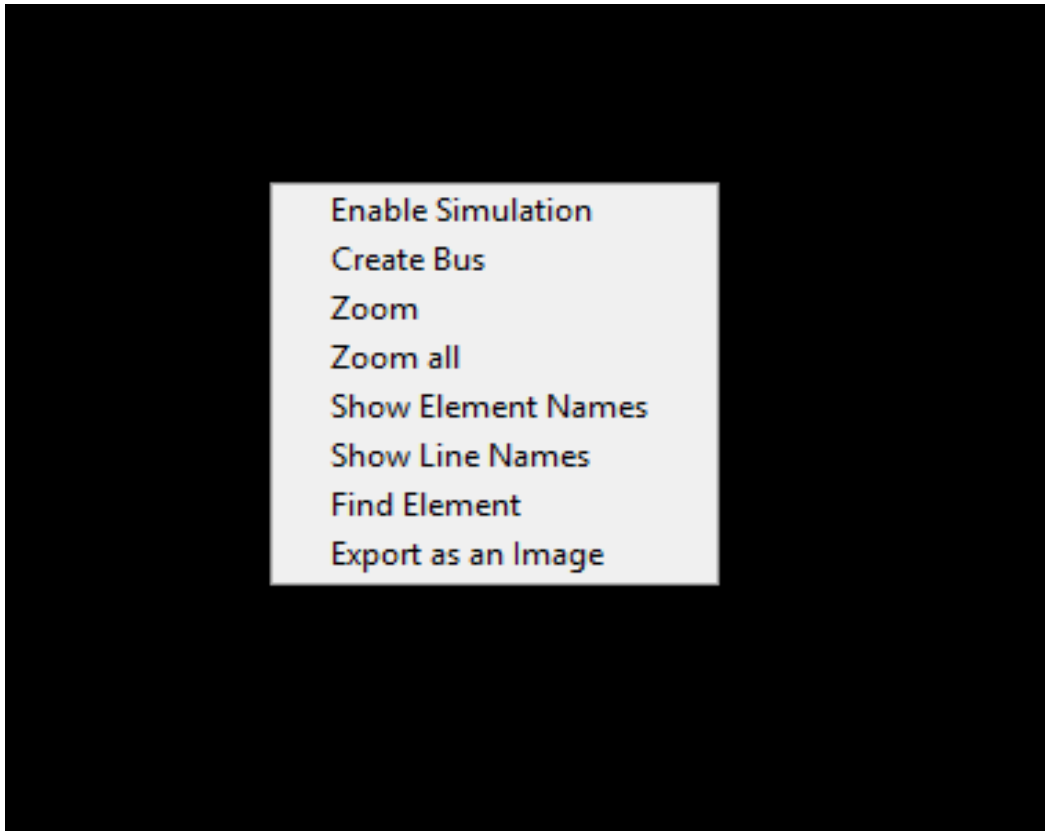
Introduction to OpenDSS-G

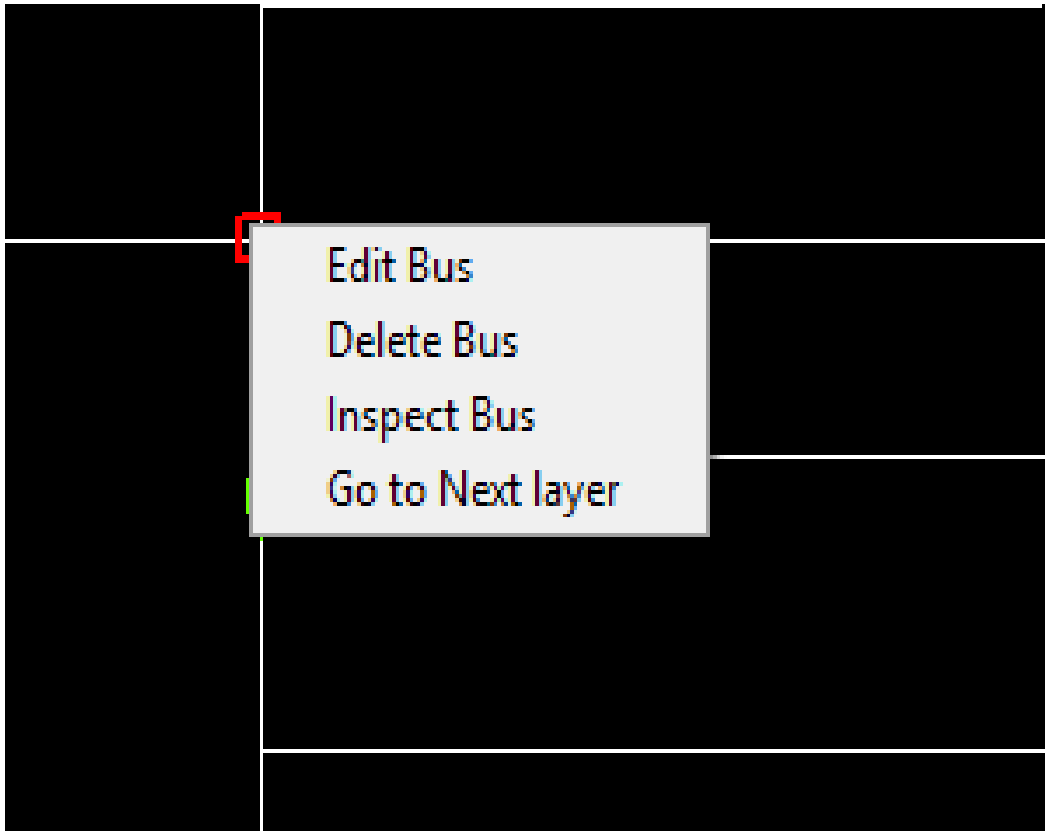


Introduction to OpenDSS-G

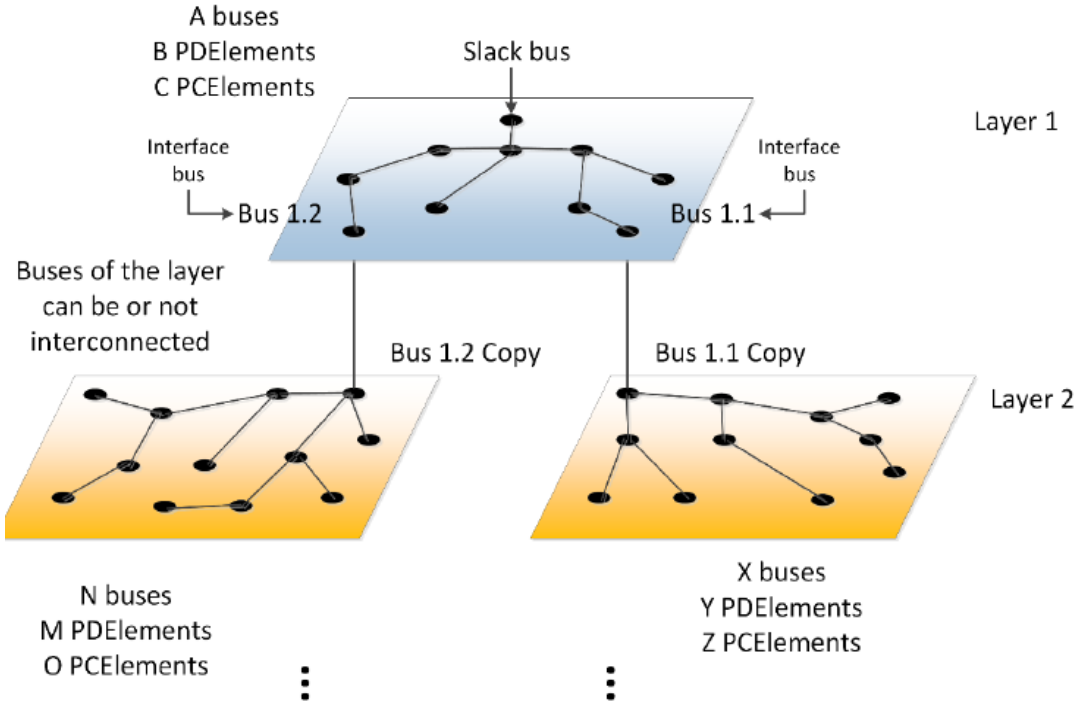


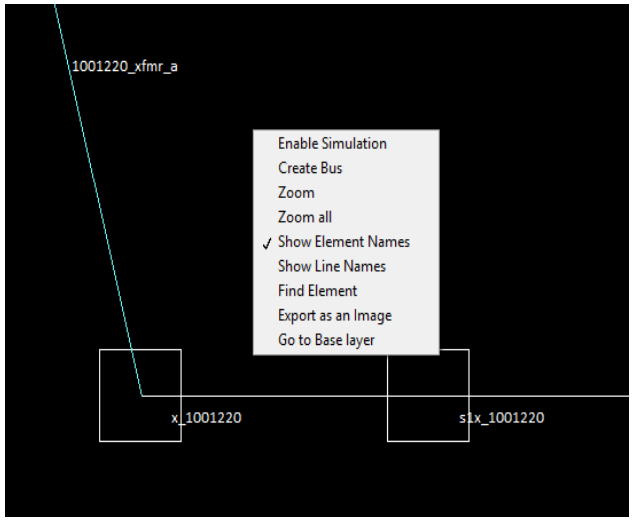
Introduction to OpenDSS-G

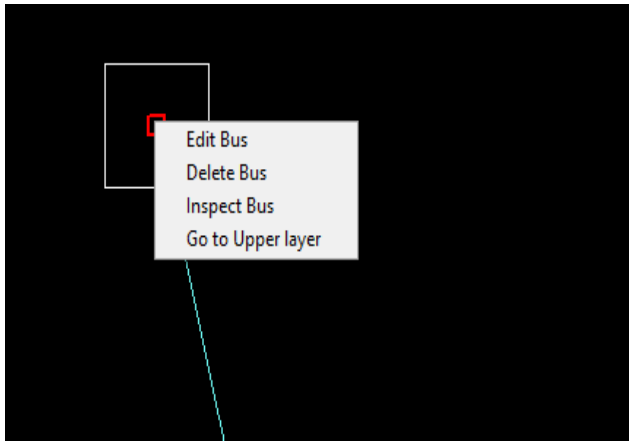




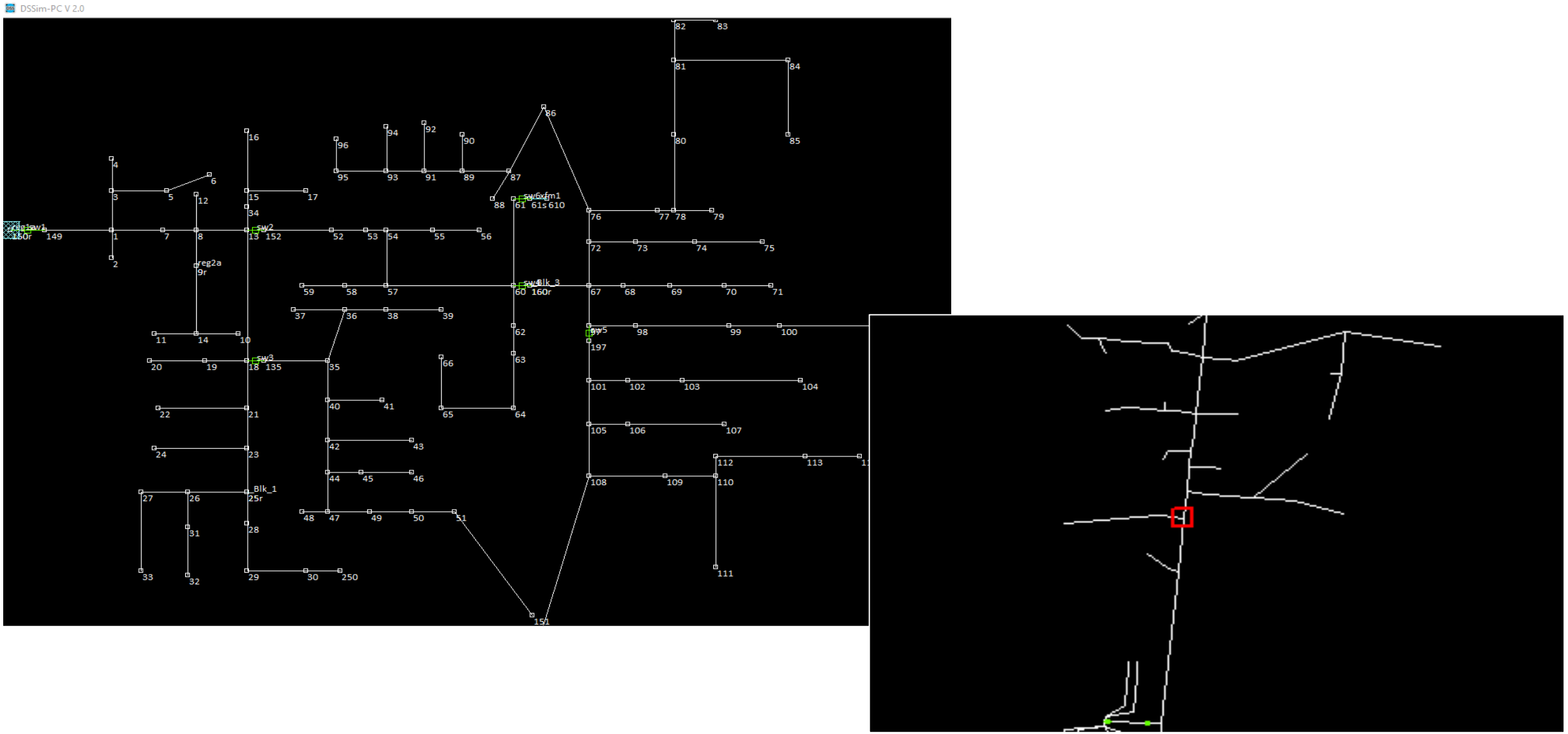
Introduction to OpenDSS-G





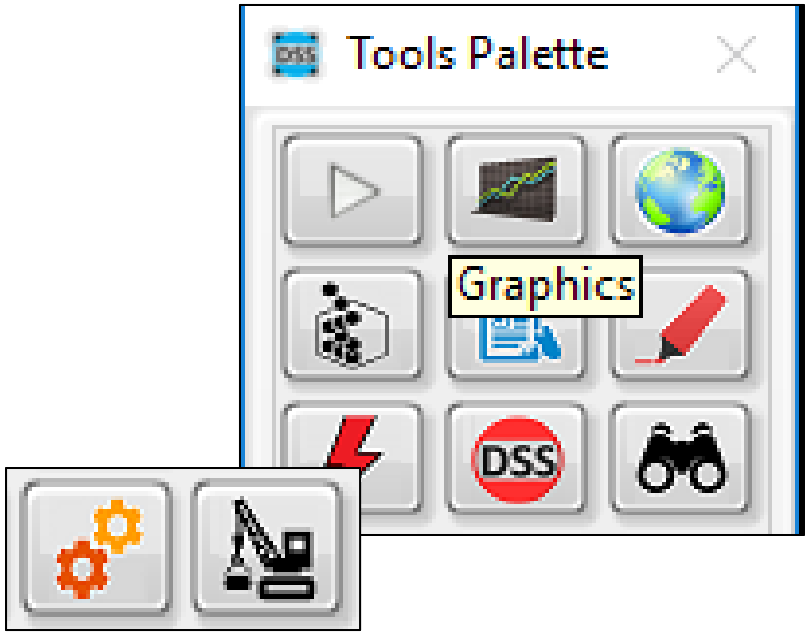


Introduction to OpenDSS-G

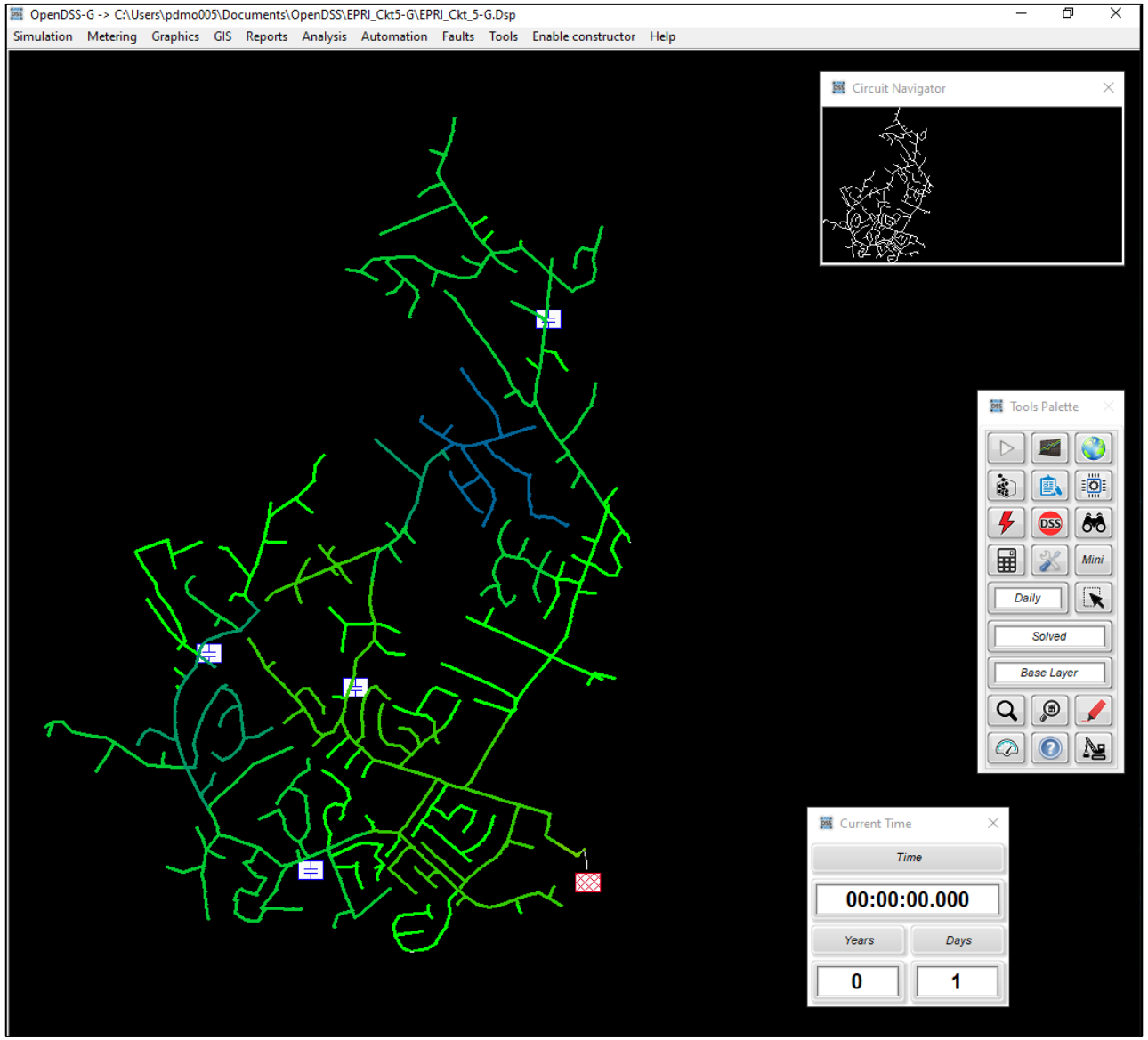


Introduction to OpenDSS-G

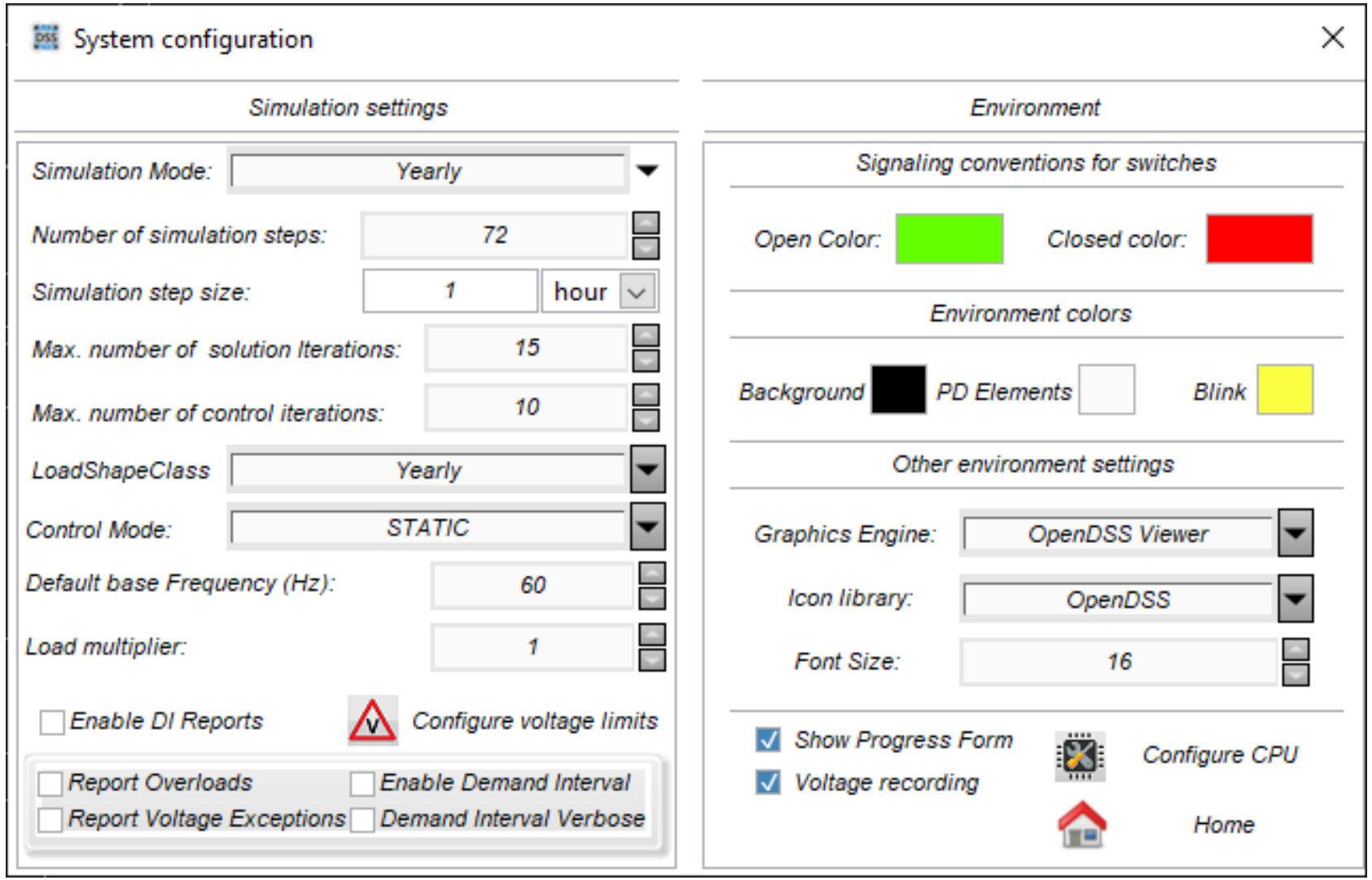
Introduction to OpenDSS-G



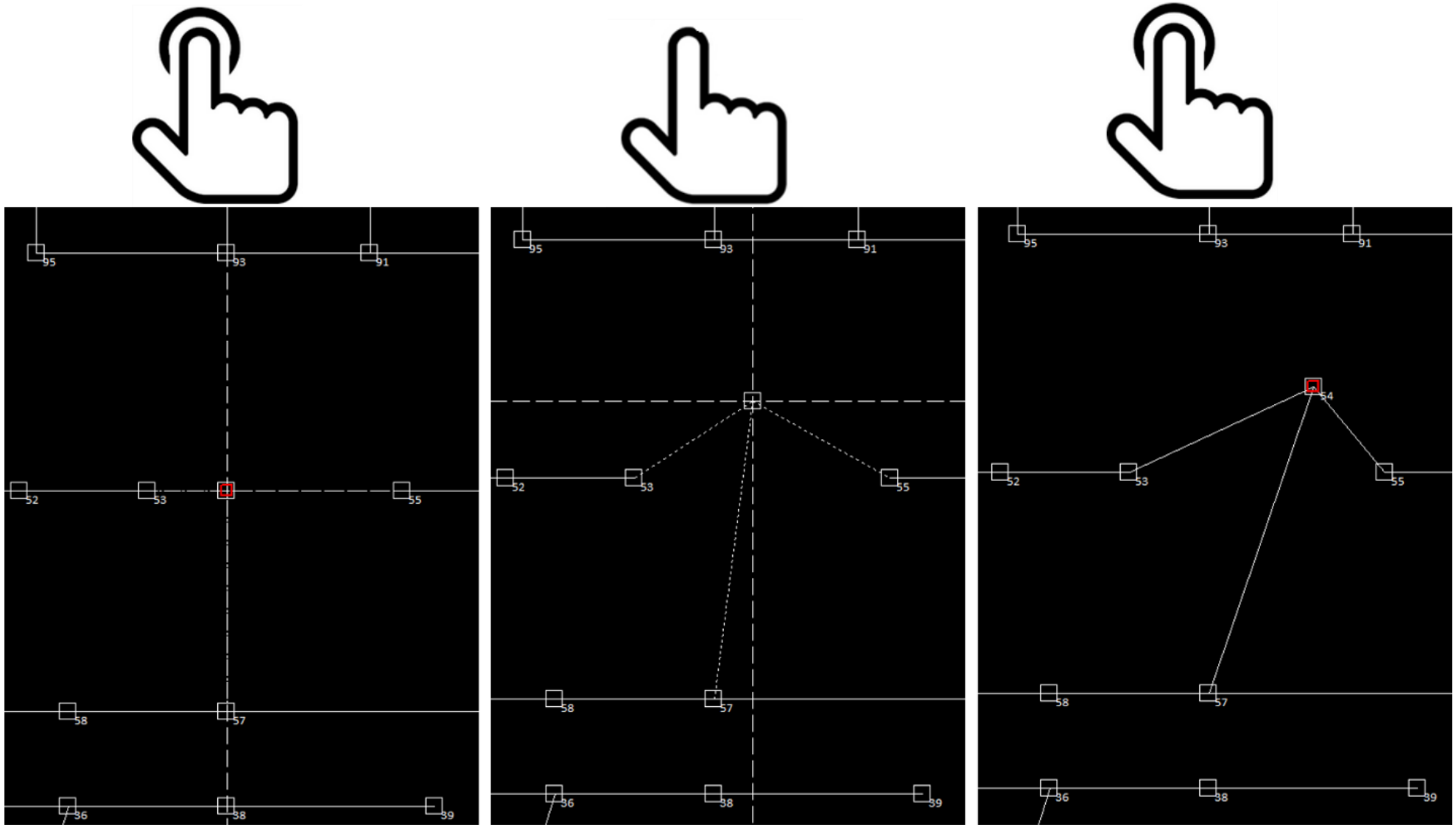
Introduction to OpenDSS-G



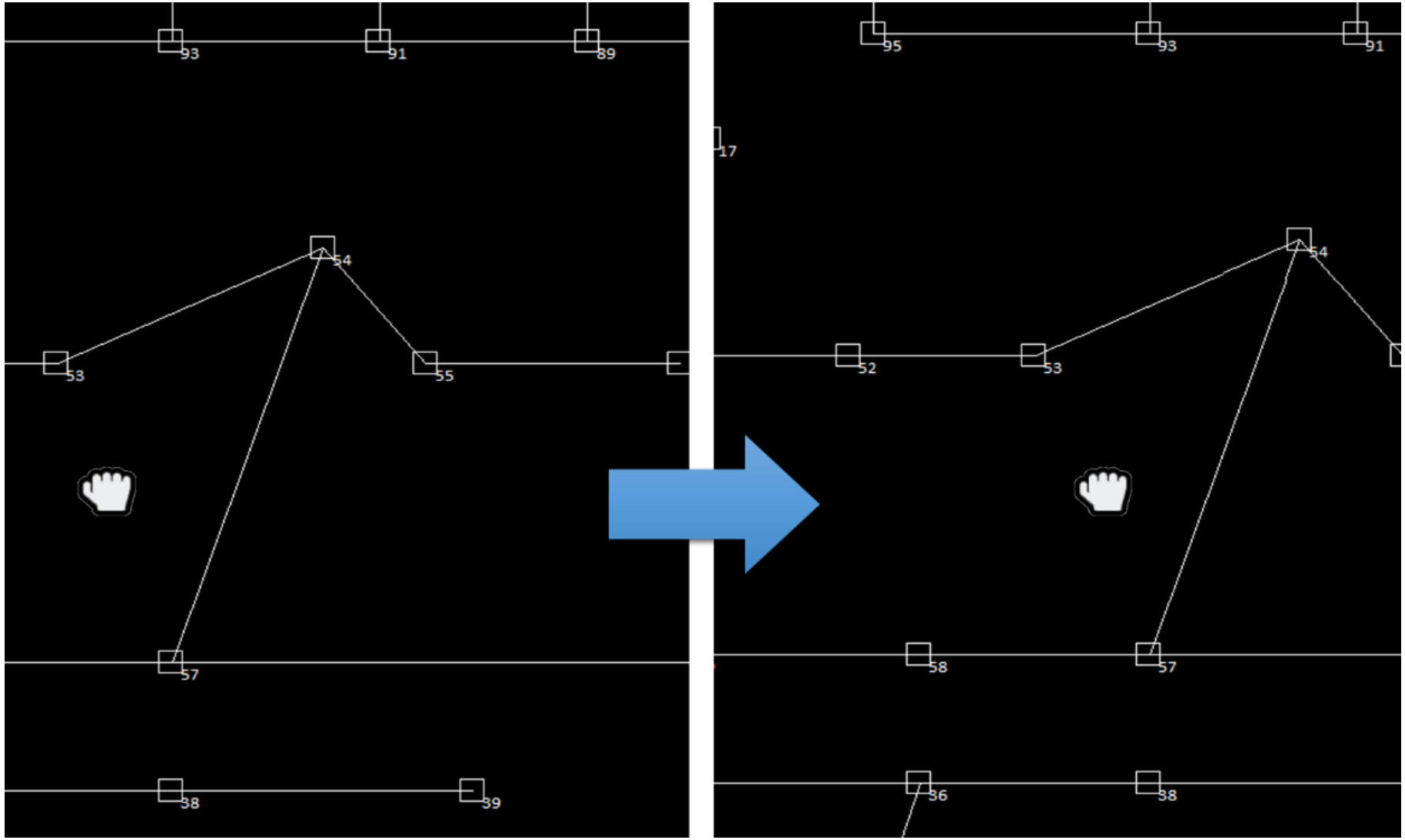
Introduction to OpenDSS-G



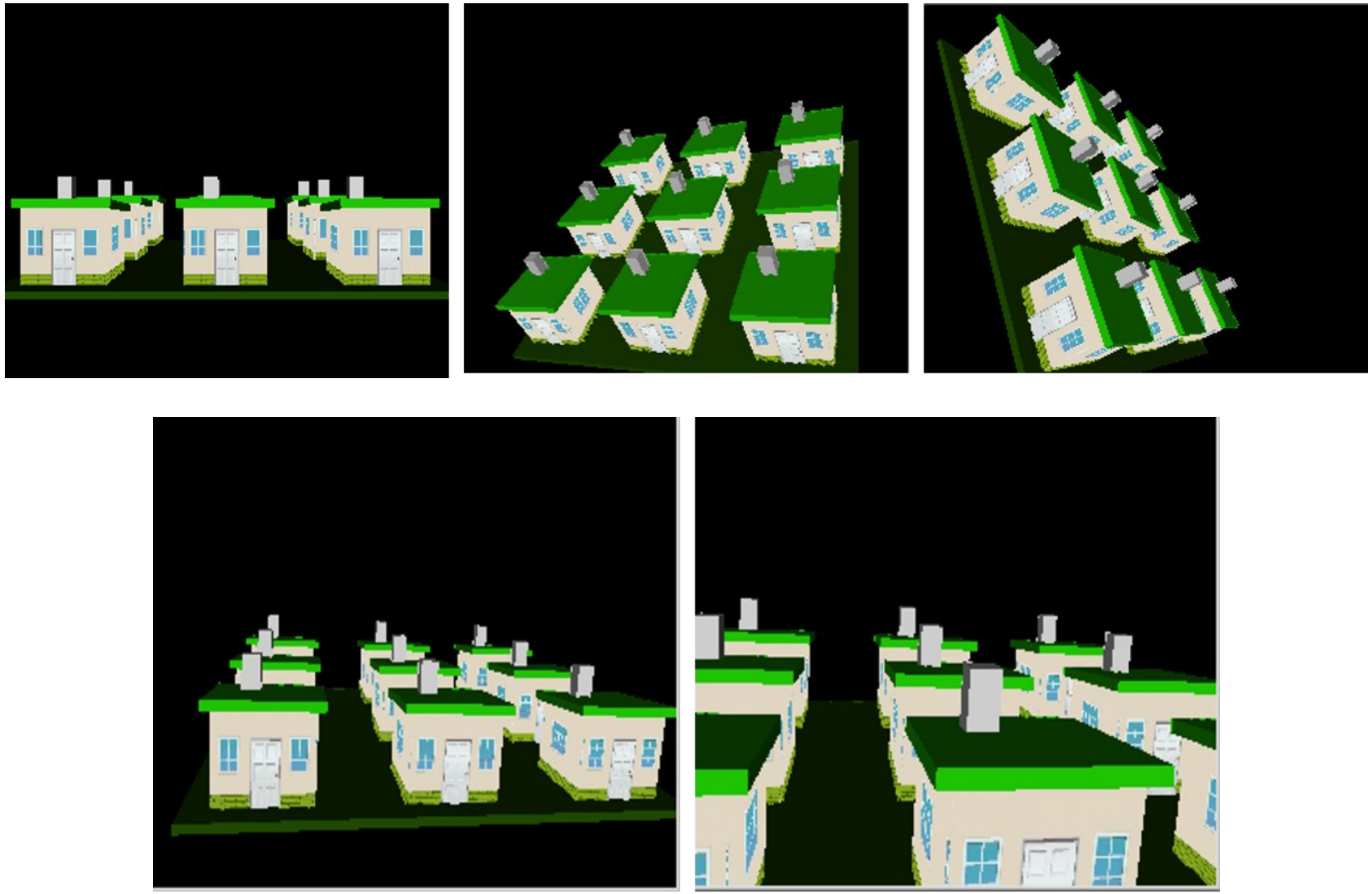
Introduction to OpenDSS-G



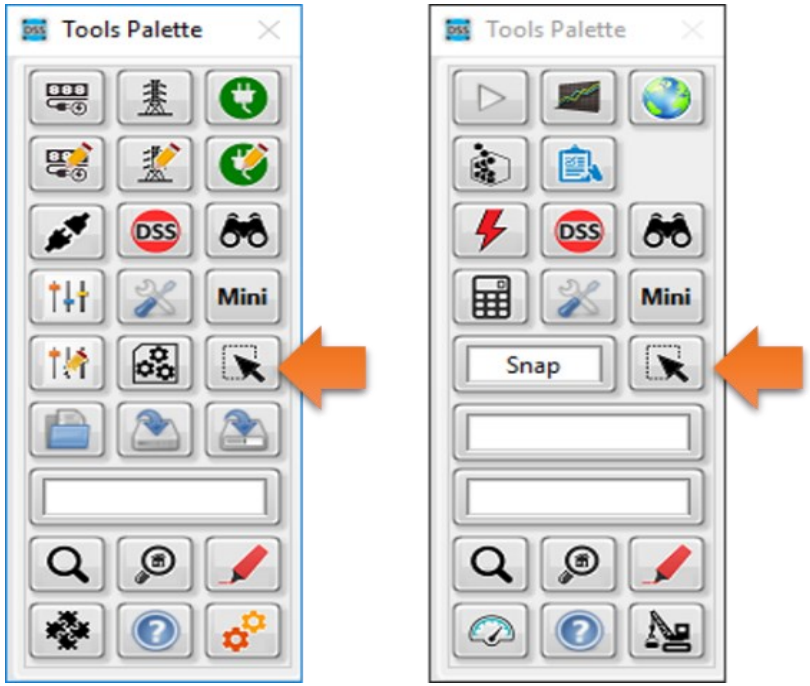
Introduction to OpenDSS-G

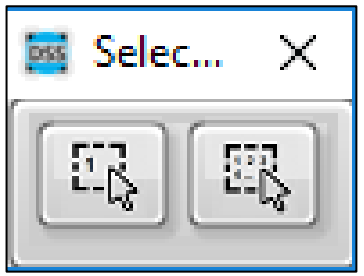


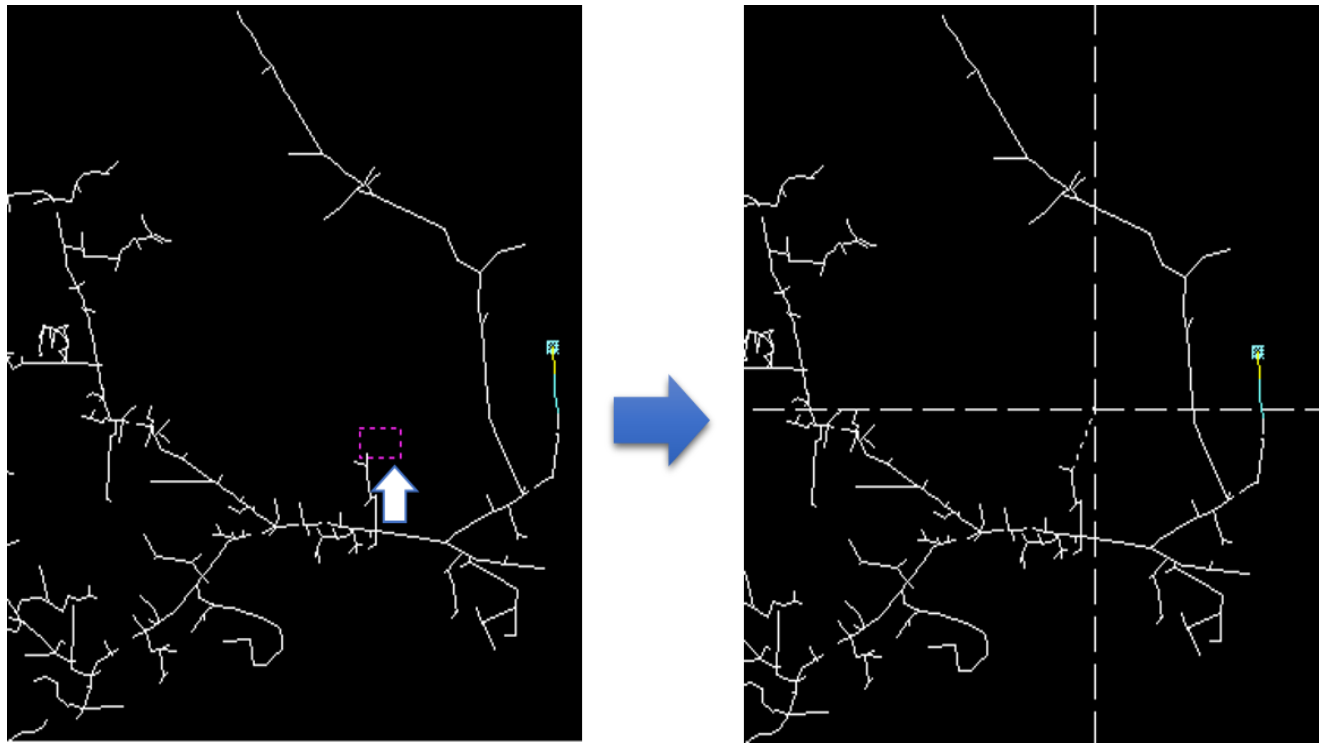
Introduction to OpenDSS-G



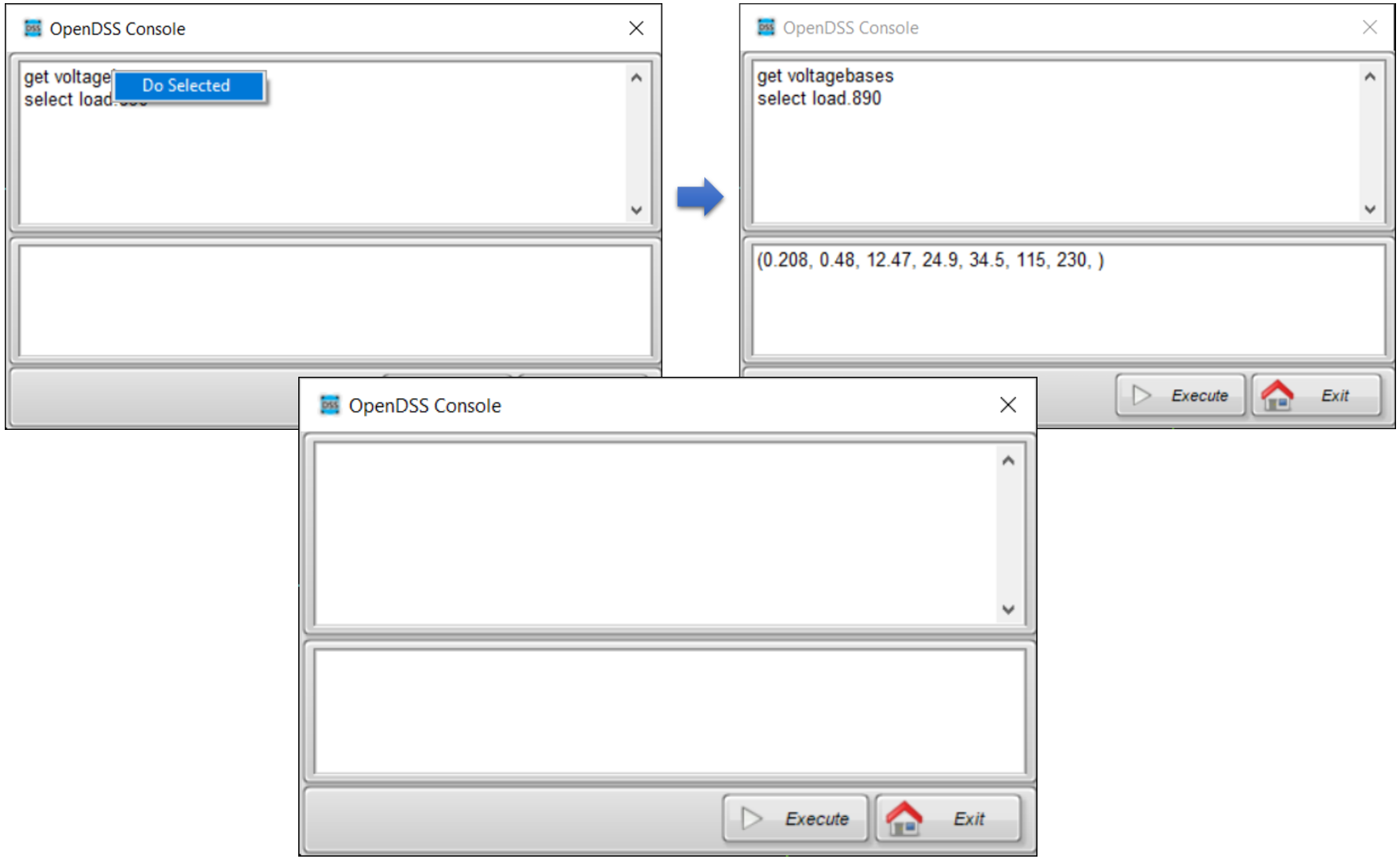
Introduction to OpenDSS-G







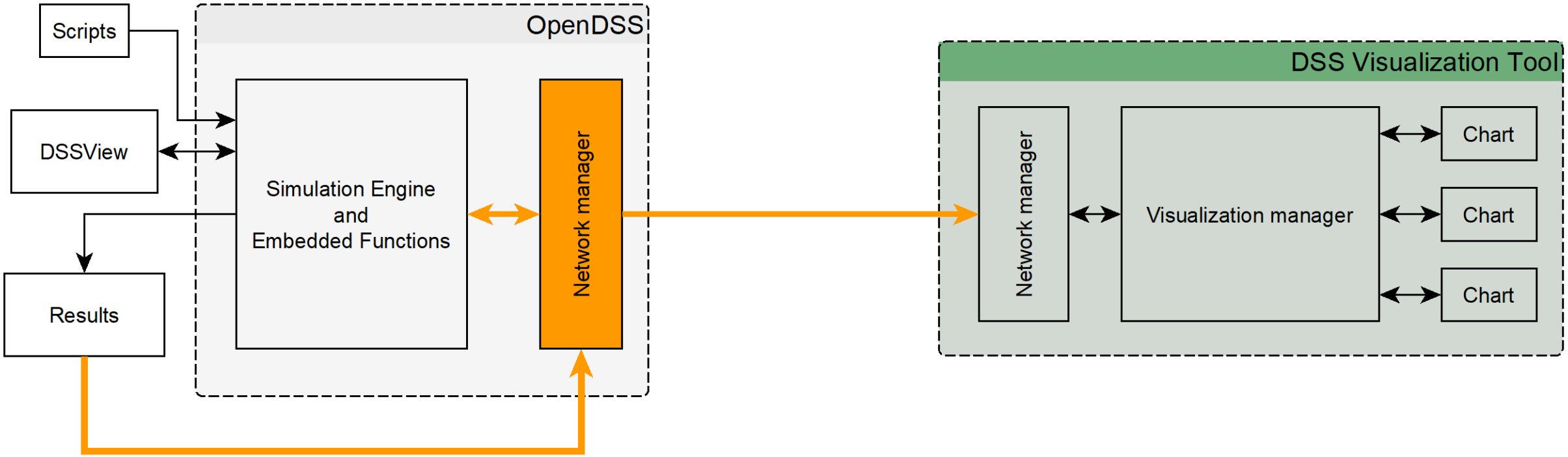
Introduction to OpenDSS-G



Complementary Tools

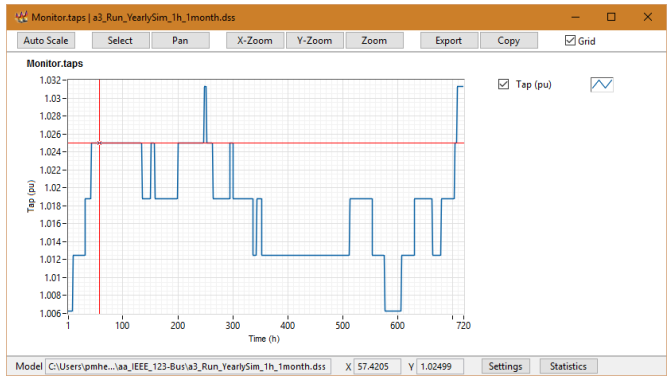
Advanced Graphics Module for OpenDSS (OpenDSS-Viewer)

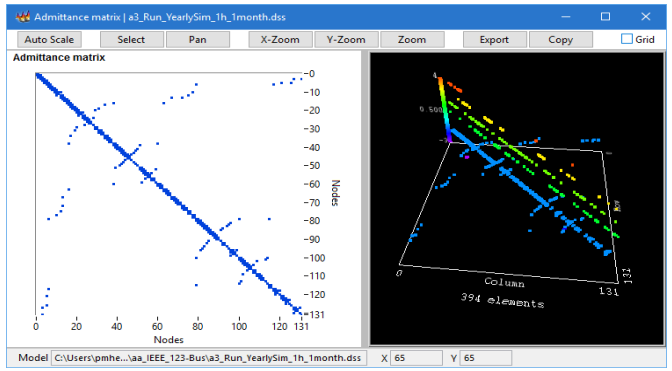
Developed by Miguel Hernandez (EPRI). Enhance the visualization of Distribution System Simulations with a **flexible**, **scalable** and **meaningful** approach.

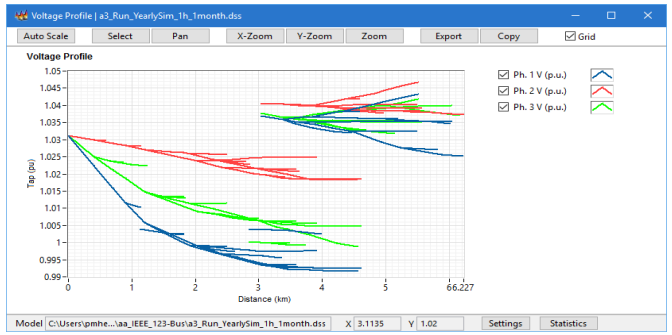


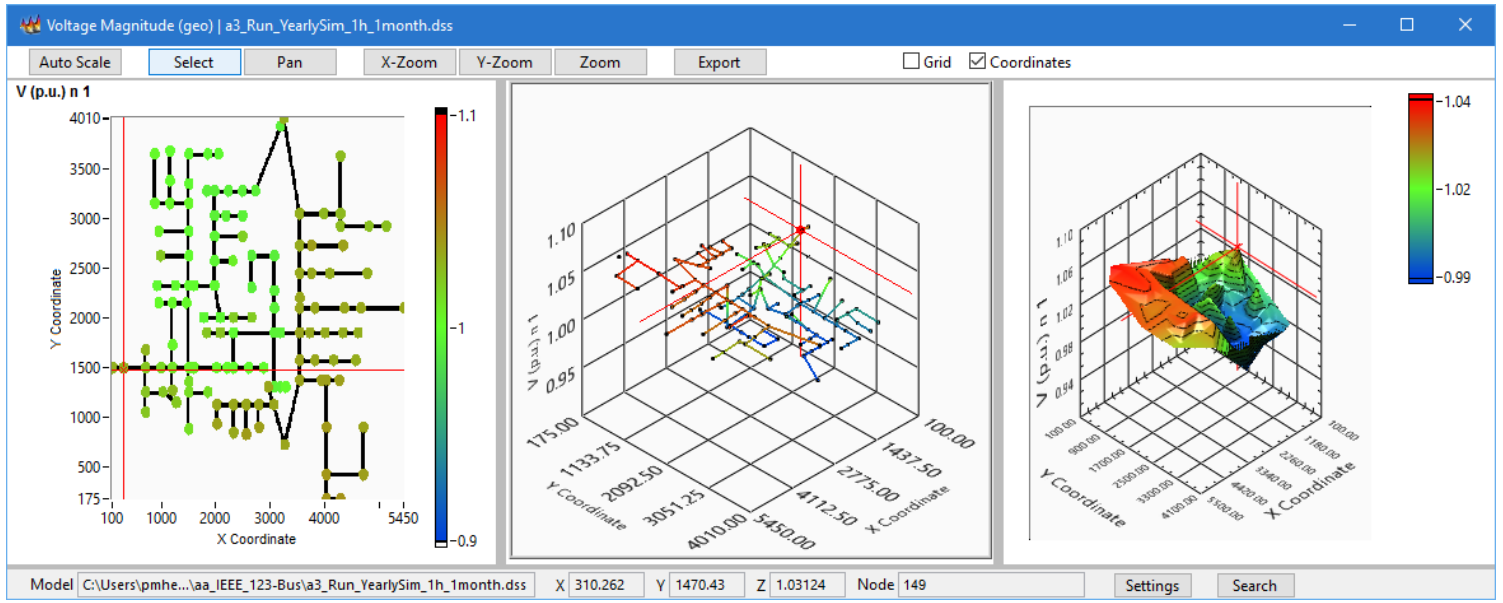
https://epri.box.com/s/o4itfn23txv73iyo3y1ukbs9nmgsspgo

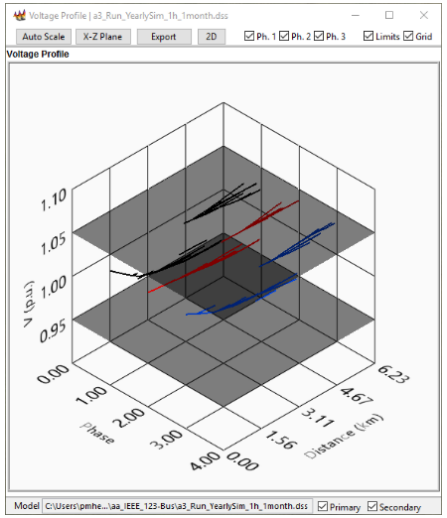
Complementary Tools

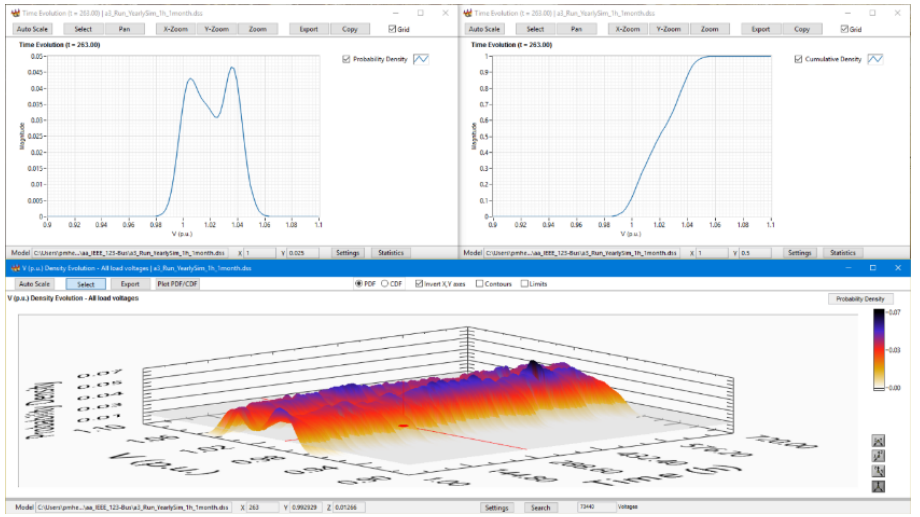








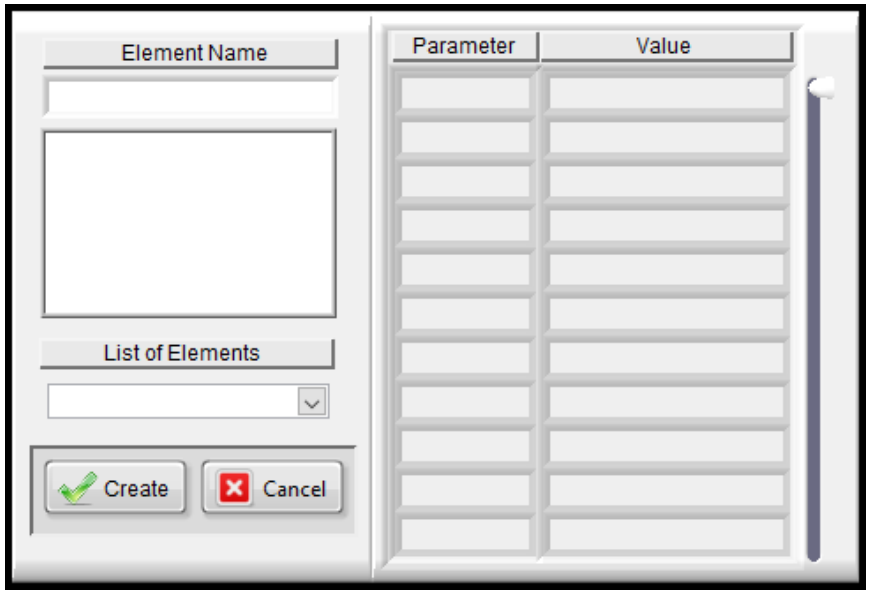




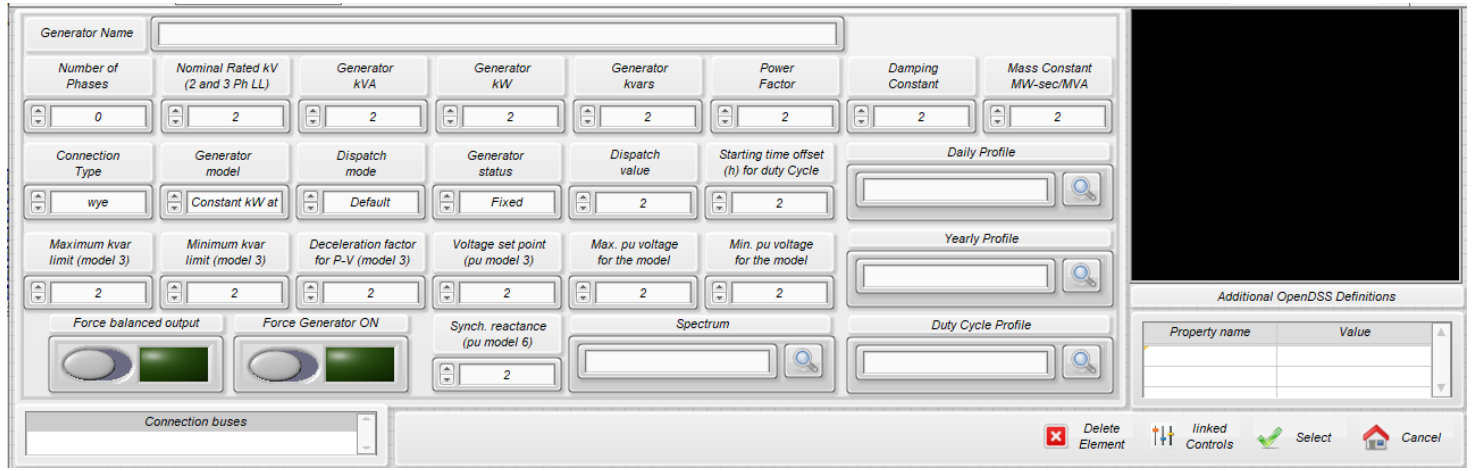
Let's see what's new in version 4.5

The input interface has changed

GUI evolution

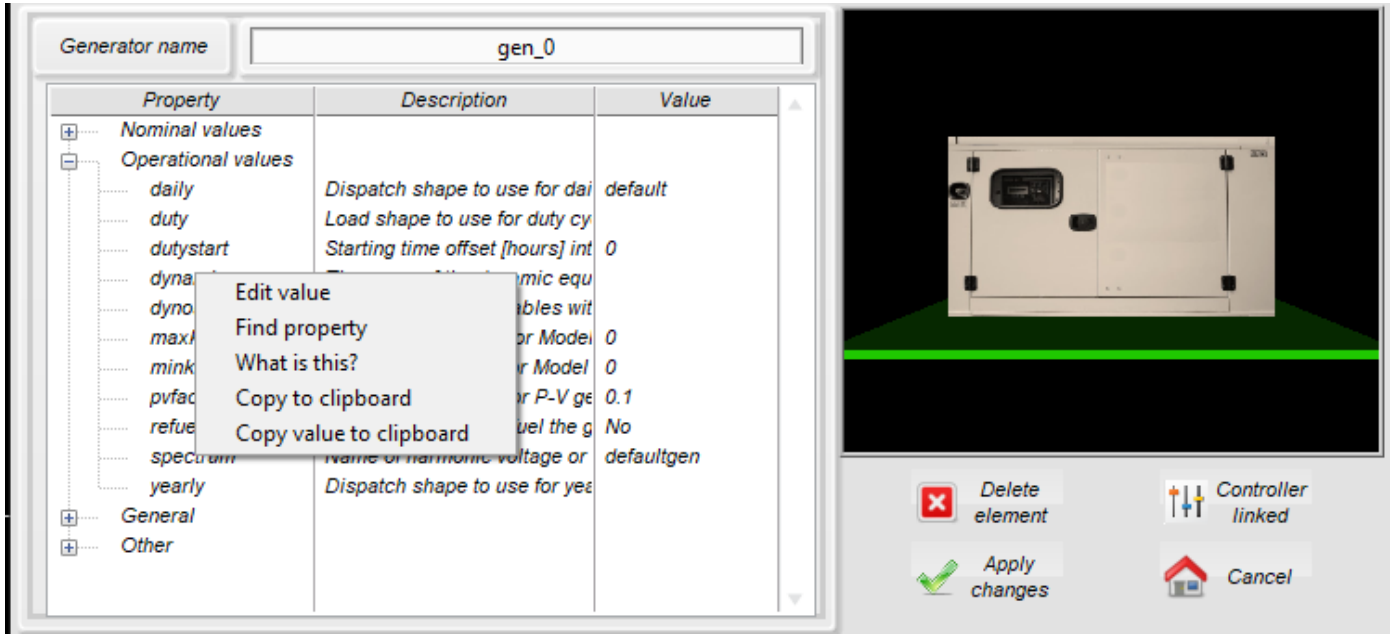


2012



2016

GUI evolution



2022

Questions?

Thanks