IOWA STATE UNIVERSITY
ECpE Department

OpenDSS – Define A Circuit

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

Contents

- General object description
- Line
- Source and other objects
- Power delivery elements
- Power conversion elements
- Control elements
- Meter elements

General Object Property Description

These are objects common to all circuits in the OpenDSS (there may be several circuits loaded into memory at any one time). Any circuit can reference the data contained in the General DSS Objects.

The following describes each object class and the parameters that may be defined through the command interface.

Note that all DSS objects have a Like parameter. When another element of the same class is very similar to a new one being created, use the Like parameter to start the definition then change the parameters that differ. Issue the Like=nnnn parameter first. Command lines are parsed and executed from left to right.

General Object Property Description

-- LineCode

LineCode objects are general library objects that contain impedance characteristics for lines and cables. The term "line code" is an old term that simply refers to a code that was made up by programmers to describe a line construction. In most distribution analysis programs, one can describe a line by its LineCode and its length. LineCode objects were defined in a separate file.

This collection of objects emulates the old linecode files, except that the concept is a little more powerful. Ultimately, the impedance of a line is described by its series impedance matrix and nodal capacitive admittance matrix. These matrices may be specified directly or they can be generated by specifying the symmetrical component data. Note that the impedances of lines may be specified directly and one does not need to use a line code, although the linecode will be more convenient most of the time. There may be hundreds of lines, but only a few different kinds of line constructions.

General Object Property Description

-- LineCode

LineCode can also perform a Kron reduction, reducing out the last conductor in the impedance matrices, which is assumed to be a neutral conductor. This applies only if the impedance is specified as a matrix. If the impedance is defined as symmetrical components, this function does not apply because symmetrical component values already assume the reduction.

By specifying the values of Rg, Xg, and rho, the DSS will take the base frequency impedance matrix values and adjust the earth return component for frequency. Skin effect in the conductors is not modified. To represent skin effect, you have to define the geometry.

This assumes the impedance matrix is constructed as follows:

$$Z = R + jX = \begin{bmatrix} Z_{11} + Zg & Z_{12} + Zg & Z_{13} + Zg \\ Z_{21} + Zg & Z_{22} + Zg & Z_{23} + Zg \\ Z_{31} + Zg & Z_{32} + Zg & Z_{33} + Zg \end{bmatrix}$$

General Object Property Description

-- LineCode

Where, using the 1st term of Carson's formulation,

$$Rg = \mu_0 \frac{\omega}{8} \quad \Omega / m$$

$$Xg = \mu_0 \frac{\omega}{2\pi} \ln\left(658.5\sqrt{\frac{\rho}{f}}\right) \quad \Omega / m$$

$$Xii = \mu_0 \frac{\omega}{2\pi} \ln\left(\frac{1}{GMRi}\right) \quad \Omega / m$$

$$Xij = \mu_0 \frac{\omega}{2\pi} \ln\left(\frac{1}{dij}\right) \quad \Omega / m$$

$$\mu_0 = 4\pi \times 10^{-7}$$

General Object Property Description

-- LineCode

The LineCode properties, in order, are: (not case sensitive)

Nphases = Number of phases. Default = 3.
R1 = Positive-Sequence resistance, ohms per unit length.
X1 = Positive-Sequence reactance, ohms per unit length.
R0 = Zero-Sequence resistance, ohms per unit length.
X0 = Zero-Sequence reactance, ohms per unit length.
C1 = Positive-Sequence capacitance, nanofarads per unit length
C0 = Zero-Sequence capacitance, nanofarads per unit length
Units = {mi | km | kft | m | ft | in | cm} Length units. If not specified, it is assumed that the units correspond to the length being used in the Line models.
Rmatrix = Series resistance matrix, ohms per unit length..
Xmatrix = Series reactance matrix, ohms per unit length.
Cmatrix = Shunt nodal capacitance matrix, nanofarads per unit length.
BaseFreq = Base Frequency at which the impedance values are specified. Default = 60.0 Hz.
Normamps = Normal ampacity, amps.
Emergamps = Emergency ampacity, amps.
Faultrate = Number of faults per year per unit length. This is the default for this general line construction.

General Object Property Description

-- LineCode

Pctperm = Percent of the fault that become permanent (requiring a line crew to repair and a sustained interruption).
Kron = Y/N. Default=N. Perform Kron reduction on the impedance matrix after it is formed, reducing order by 1. Do this only on initial definition after matrices are defined. Ignored for symmetrical components.
Rg = Carson earth return resistance per unit length used to compute impedance values at base frequency. See description above. For making better adjustments of line impedance values for frequency for harmonics studies. Default= 0.01805 ohms per 1000 ft at 60 Hz. If you do not wish to adjust the earth return impedance for frequency, set both Rg and Xg to zero. Generally avoid Kron reduction if you will be solving at frequencies other than the base frequency and wish to adjust the earth return impedance.

General Object Property Description

-- LineCode

Xg = Carson earth return reactance per unit length used to compute impedance values at base frequency. See description above. For making better adjustments of line impedance values for frequency for harmonics studies. Default= 0.155081 ohms per 1000 ft at 60 Hz. If you do not wish to adjust the earth return impedance for frequency, set both Rg and Xg to zero. Generally avoid Kron reduction if you will be solving at frequencies other than the base frequency and wish to adjust the earth return impedance.

Rho = Earth resistivity used to compute earth correction factor. Default=100 meter ohms.

Like = Name of an existing LineCode object to build this like.

General Object Property Description

-- LineGeometry

This class of data is used to define the positions of the conductors.

Nconds = Number of conductors in this geometry. Default is 3. Triggers memory allocations. Define first!

Nphases = Number of phases. Default =3; All other conductors are considered neutrals and might be reduced out.

Cond = Set this to number of the conductor you wish to define. Default is 1.

Wire = Code from WireData. MUST BE PREVIOUSLY DEFINED. no default.

X = x coordinate.

H = Height of conductor.

Units = Units for x and h: {mi|kft|km|m|Ft|in|cm } Initial default is "ft", but defaults to last unit defined.

Normamps = Normal ampacity, amperes for the line. Defaults to first conductor if not specified.

Emergamps = Emergency ampacity, amperes. Defaults to first conductor if not specified.

Reduce = { Yes | No} Default = no. Reduce to Nphases (Kron Reduction). Reduce out neutrals.

Like = Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

General Object Property Description

-- Line Constants Examples

Define the wire data:

```
New Wiredata.ACSR336 GMR=0.0255000 DIAM=0.7410000 RAC=0.3060000
~ NormAmps=530.0000
~ Runits=mi radunits=in gmrunits=ft
New Wiredata.ACSR1/0 GMR=0.0044600 DIAM=0.3980000 RAC=1.120000
~ NormAmps=230.0000
~ Runits=mi radunits=in gmrunits=ft
```

Define the Geometry data:

```
New Linegeometry.HC2_336_1neut_0Mess nconds=4 nphases=3
~ cond=1 Wire=acsr336 x=-1.2909 h=13.716 units=m
~ cond=2 Wire=acsr336 x=-0.502 h=13.716 !units=m
~ cond=3 Wire=acsr336 x=0.5737 h=13.716 !units=m
~ cond=4 Wire= ACSR1/0 x=0 h=14.648 ! units=m ! neutral
```

Define a 300-ft line section:

```
New Line.Line1 Bus1=xxx Bus2=yyy
~ Geometry= HC2_336_1neut_0Mess
~ Length=300 units=ft
```

General Object Property Description

-- Line Constants Examples

Check out the line constants at 60 and 600 Hz in per km values. This command shows line constants for all defined geometries:

```
Show lineconstants freq=60 units=km
Show lineconstants freq=600 units=km
```

If the number of conductors = 3, this Show command will also give you the sequence impedances. If your geometry has more than 3 conductors and you want to see the sequence impedance, define

```
nconds = (...whatever...)
nphases = 3
Reduce=Yes
```

This will force the impedance matrices to be reduced to 3x3 and the Show LineConstants command will automatically give the sequence impedances. Note: make sure the phase conductors are defined first in the geometry definition.

General Object Property Description

-- LoadShape

A Loadshape object is very important for all types of sequential power flow solutions. A LoadShape object consists of a series of multipliers, typically ranging from 0.0 to 1.0 that are applied to the base kW values of the load to represent variation of the load over some time period.

Load shapes are generally fixed interval, but may also be variable interval. For the latter, both the time, hr, and the multiplier must be specified.

All loadshapes, whether they be daily, yearly, or some arbitrary duty cycle, are maintained in this class. Each load simply refers to the appropriate shape by name.

The loadshape arrays may be entered directly in command line, or the load shapes may be stored in one of three different types of files from which the shapes are loaded into memory.

General Object Property Description

-- LoadShape

The properties for LoadShape objects are:

Npts = Number of points to expect when defining the curve

Interval = time interval of the data, Hr. Default=1.0. If the load shape has non-uniformly spaced points, specify the interval as 0.0.

mInterval = Specify Interval in minutes.

sInterval = Specify Interval in seconds.

Mult = Array of multiplier values. Looking for Npts values.

Hour = Array of hour values corresponding to the multipliers.

Qmult = Array of multiplier values. Same property rules as the Mult property. If specified, the multiplier is applied to the Load (or Generator) kvar value. If omitted, the value of Mult is applied to the kvar value.

Mean = Mean of the multiplier array.

Stddev = Standard Deviation (see Mean, above).

General Object Property Description

-- LoadShape

The next three properties instruct the LoadShape object to get its data from a file. Three different formats are allowed. If Interval>0 then only the multiplier is entered. For variable interval data, set Interval=0.0 and enter both the time (in hours) and multiplier, in that order for each interval.

Csvfile = Name of a CSV file containing load shape data, one interval to a line.

Sngfile = Name of a binary file of single-precision floating point values containing the load shape data.

Dblfile = Name of a binary file of double-precision floating point values containing the load shape data.

Action= {Normalize | DblSave | SngSave} After defining load curve data, setting action=normalize will modify the multipliers so that the peak is 1.0. The mean and std deviation are recomputed. Setting action=DblSave or SngSave will cause the present mult and qmult values to be written to either a packed file of double or single, respectively.

UseActual = {Yes | No* | True | False*} If true, signals to Load, Generator, or other objects to use the value of the multiplier as the actual kW, kvar value rather than a multiplier.

General Object Property Description

-- LoadShape

Pmax, Qmax= If you define the LoadShape object with UseActual=Yes, when you define any of the Duty, Daily, or Yearly properties of a load or generator, the kW property is redefined to the kW and kvar values at the time of the peak kW in the loadshape. This will be the value used for the initial Snapshot solution. If you define more than one loadshape object, the last one overrides any previous definition, as with all OpenDSS properties. You can query the Pmax and Qmax properties of the Loadshape object to see what was computed.

Pbase = Base P value for normalization. Default is zero, meaning the peak will be used.

Like = Name of an existing loadshape object to base this one on.

General Object Property Description

-- GrowthShape

A GrowthShape object is similar to a Loadshape object. However, it is intended to represent the growth in load year-by-year and the way the curve is specified is entirely different. You must enter the growth for the first year. Thereafter, only the years where there is a change must be entered. Otherwise it is assumed the growth stays the same.

Growth rate is specified by specifying the multiplier for the previous year's load. Thus, if the load grows 2.5% in 1999, the multiplier for that year will be specified as 1.025.

The parameters are:

Npts = Number of points to expect when defining the curve.
Year = Array of year values corresponding to the multiplier values. Enter only those years in which the multiplier changes.
Mult= Array of Multiplier values corresponding to the year values. Enter multiplier by which the load will grow in this year.

General Object Property Description

-- GrowthShape

Csvfile = Name of a csv file containing one (year, mult) point per line. Separate the year from the multiplier by a comma.
Sngfile = Name of a file of single-precision numbers containing (year, mult) points packed.
Dblfile = Name of a file of single-precision numbers containing (year, mult) points packed.
Like = Name of an existing GrowthShape object to base this one on.

General Object Property Description
-- TCC_Curve

A TCC_Curve object is defined similarly to Loadshape and Growthshape objects in that they all are defined by curves consisting of arrays of points. Intended to model time-current characteristics for overcurrent relays, TCC_Curve objects are also used for other relay types requiring time curves. Both the time array and the C array must be entered.

The properties are:

Npts = Number of points to expect when defining the curve.
C_Array = Array of current (or voltage or whatever) values corresponding to time values in T_Array (see T_Array).
T_Array = Array of time values in sec. Typical array syntax:
t_array = (1, 2, 3, 4, ...)
You may also substitute a file designation: t_array = (file=filename). The specified file has one value per line.
Like = Name of an existing GrowthShape object to base this one on.

General Object Property Description

-- WireData

This class of data defines the raw conductor data that is used to compute the impedance for a line geometry.

Note that you can use whatever units you want for any of the dimensional data – be sure to declare the units. Otherwise, the units are all assumed to match, which would be very rare for conductor data. Conductor data is usually supplied in a hodge-podge of units. Everything is converted to meters internally to the DSS.

The parameters are:

Rdc = dc Resistance, ohms per unit length (see Runits). Defaults to Rac if not specified.
Rac = Resistance at 60 Hz per unit length. Defaults to Rdc if not specified.
Runits = Length units for resistance: ohms per {mi|kft|km|m|Ft|in|cm } Default=none.
GMRac = GMR at 60 Hz. Defaults to .7788*radius if not specified.
GMRunits = Units for GMR: {mi|kft|km|m|Ft|in|cm } Default=none.
Radius = Outside radius of conductor. Defaults to GMR/0.7788 if not specified.
…

General Object Property Description

-- WireData

The parameters are:

…

Normamps = Normal ampacity, amperes. Defaults to Emergency amps/1.5 if not specified.
Emergamps = Emergency ampacity, amperes. Defaults to 1.5 * Normal Amps if not specified.
Diam = Diameter; Alternative method for entering radius.
Like = Make like another object of this class.

Contents

- General object description
- Line
- Source and other objects
- Power delivery elements
- Power conversion elements
- Control elements
- Meter elements

Modeling A Line in OpenDSS

1. Using Wire and Line Geometry Data

- WireData defines the ***raw conductor data***.
- LineGeometry is used to define the ***positions*** of the conductors.

Example:

```
// Step 1: define the wires:
New Wiredata.ACSR336 GMR=0.0244000 DIAM=0.7210000 RAC=0.3060000
~ NormAmps=530.0000
~ Runits=mi radunits=in gmrunits=ft
New Wiredata.ACSR4/06/1 GMR=0.0081400 DIAM=0.5630000 RAC=0.5920
~ NormAmps=230.0000
~ Runits=mi radunits=in gmrunits=ft

// Step 2: define the Geometry:
New Linegeometry.HC2_336_1neut_0Mess nconds=4 nphases=3 Reduce=Yes
~ cond=1 Wire=acsr336      x=0    h=29 units=ft
~ cond=2 Wire=acsr336      x=2.5  h=29 units=ft
~ cond=3 Wire=acsr336      x=7    h=29 units=ft
~ cond=4 Wire= ACSR4/06/1  x=4    h=25 units=ft

// Step 3: define a 300-ft line section:
New Line.L1 Bus1=n Bus2=m
~ Geometry= HC2_336_1neut_0Mess
~ Length=1 units=mi
~ EarthModel=Carson
```

Wire name

Geometrical Mean Radius

Diameter

Resistance at 60 Hz per unit length

Normal ampacity

Units

Geometry name

Number of conductors

Number of phases

Conductor #

Reduce from Nconds to Nphases by Kron Reduction

Specify a pre-defined Wire

X coordinate

Height of conductor

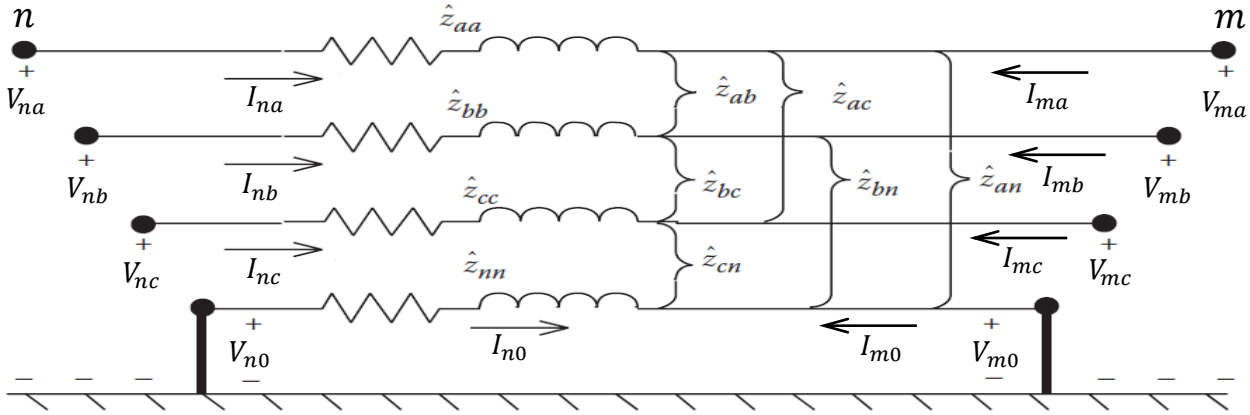
Specify a predefined Geometry

Use Carson's equations

Modeling A Line in OpenDSS

1. Using Wire and Line Geometry Data

How to build the primitive Y matrix for a line?



$$\hat{z}_{ii} = r_i + 0.09530 + j0.12134(\ln\frac{1}{GMR_i} + 7.93402)\ \Omega/\text{mile} \qquad (4.41)$$

$$\hat{z}_{ij} = 0.09530 + j0.12134(\ln\frac{1}{D_{ij}} + 7.93402)\Omega/\text{mile} \qquad (4.42)$$

$$\boldsymbol{Z}_{primitive} = \begin{bmatrix} [\hat{z}_{ij}] & [\hat{z}_{i0}] \\ [\hat{z}_{0j}] & [\hat{z}_{00}] \end{bmatrix} \qquad (4.46)$$

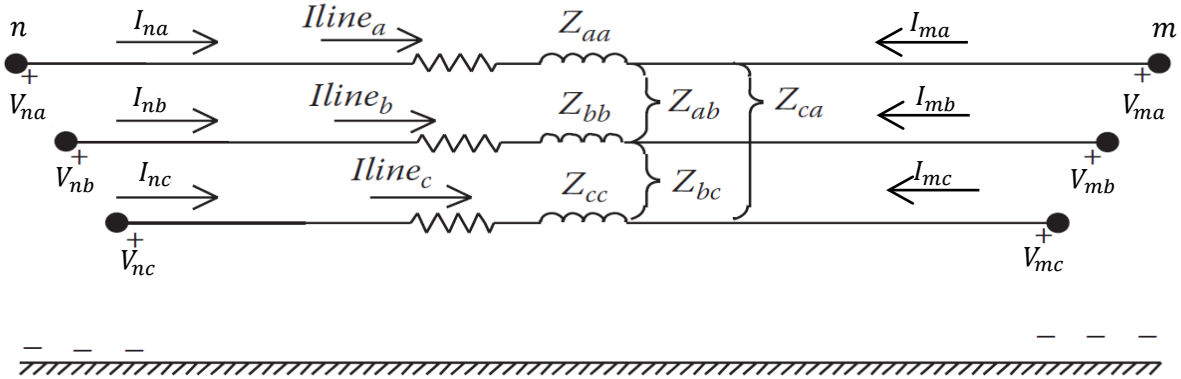
$$\boldsymbol{Z}_{abc} = [\hat{z}_{ij}] - [\hat{z}_{i0}] * [\hat{z}_{00}]^{-1} * [\hat{z}_{0j}] \quad \text{(“Kron” reduction)} \qquad (4.55)$$

where, $\boldsymbol{Z}_{abc} = \begin{bmatrix} Z_{aa} & Z_{ab} & Z_{ac} \\ Z_{ba} & Z_{bb} & Z_{bc} \\ Z_{ca} & Z_{cb} & Z_{cc} \end{bmatrix}$

Modeling A Line in OpenDSS

1. Using Wire and Line Geometry Data

How to build the primitive Y matrix for a line?



$$\boldsymbol{V}^n_{abc} = \boldsymbol{V}^m_{abc} + \boldsymbol{Z}_{abc}\boldsymbol{I}^n_{abc}$$

$$\boldsymbol{I}^n_{abc} = \boldsymbol{Z}^{-1}_{abc}(\boldsymbol{V}^n_{abc} - \boldsymbol{V}^m_{abc})$$

$$\boldsymbol{I}^n_{abc} = [\boldsymbol{Z}^{-1}_{abc} \quad -\boldsymbol{Z}^{-1}_{abc}] * \begin{bmatrix}\boldsymbol{V}^n_{abc}\\ \boldsymbol{V}^m_{abc}\end{bmatrix}$$

$$\boldsymbol{V}^m_{abc} = \boldsymbol{V}^n_{abc} + \boldsymbol{Z}_{abc}\boldsymbol{I}^m_{abc}$$

$$\boldsymbol{I}^m_{abc} = \boldsymbol{Z}^{-1}_{abc}(\boldsymbol{V}^m_{abc} - \boldsymbol{V}^n_{abc})$$

$$\boldsymbol{I}^m_{abc} = [-\boldsymbol{Z}^{-1}_{abc} \quad \boldsymbol{Z}^{-1}_{abc}] * \begin{bmatrix}\boldsymbol{V}^n_{abc}\\ \boldsymbol{V}^m_{abc}\end{bmatrix}$$

where, $\boldsymbol{V}^n_{abc} = \begin{bmatrix}V_{na}\\ V_{nb}\\ V_{nc}\end{bmatrix}, \boldsymbol{V}^m_{abc} = \begin{bmatrix}V_{ma}\\ V_{mb}\\ V_{mc}\end{bmatrix}, \boldsymbol{I}^n_{abc} = \begin{bmatrix}I_{na}\\ I_{nb}\\ I_{nc}\end{bmatrix}, \boldsymbol{I}^m_{abc} = \begin{bmatrix}I_{ma}\\ I_{mb}\\ I_{mc}\end{bmatrix}.$

Modeling A Line in OpenDSS

1. Using Wire and Line Geometry Data

How to build the primitive Y matrix for a line?

$$\begin{cases} \boldsymbol{I}^n_{abc} = [\boldsymbol{Z}^{-1}_{abc} \quad -\boldsymbol{Z}^{-1}_{abc}] * \begin{bmatrix} \boldsymbol{V}^n_{abc} \\ \boldsymbol{V}^m_{abc} \end{bmatrix} \\ \boldsymbol{I}^m_{abc} = [-\boldsymbol{Z}^{-1}_{abc} \quad \boldsymbol{Z}^{-1}_{abc}] * \begin{bmatrix} \boldsymbol{V}^n_{abc} \\ \boldsymbol{V}^m_{abc} \end{bmatrix} \end{cases} \Rightarrow \begin{bmatrix} \boldsymbol{I}^n_{abc} \\ \boldsymbol{I}^m_{abc} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Z}^{-1}_{abc} & -\boldsymbol{Z}^{-1}_{abc} \\ -\boldsymbol{Z}^{-1}_{abc} & \boldsymbol{Z}^{-1}_{abc} \end{bmatrix} * \begin{bmatrix} \boldsymbol{V}^n_{abc} \\ \boldsymbol{V}^m_{abc} \end{bmatrix}$$

$$\Rightarrow \boldsymbol{I} = \boldsymbol{Y}_{prim} * \boldsymbol{V}$$

Where, $\boldsymbol{Y}_{prim} = \begin{bmatrix} \boldsymbol{Z}^{-1}_{abc} & -\boldsymbol{Z}^{-1}_{abc} \\ -\boldsymbol{Z}^{-1}_{abc} & \boldsymbol{Z}^{-1}_{abc} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Y}_{abc} & -\boldsymbol{Y}_{abc} \\ -\boldsymbol{Y}_{abc} & \boldsymbol{Y}_{abc} \end{bmatrix}, \boldsymbol{I} = \begin{bmatrix} \boldsymbol{I}^n_{abc} \\ \boldsymbol{I}^m_{abc} \end{bmatrix}, \boldsymbol{V} = \begin{bmatrix} \boldsymbol{V}^n_{abc} \\ \boldsymbol{V}^m_{abc} \end{bmatrix}$

In a detailed form:

$$\begin{bmatrix} I_{na} \\ I_{nb} \\ I_{nc} \\ I_{ma} \\ I_{mb} \\ I_{mc} \end{bmatrix} = \begin{bmatrix} Y_{na,na} & Y_{na,nb} & Y_{na,nc} & Y_{na,ma} & Y_{na,mb} & Y_{na,mc} \\ Y_{nb,na} & Y_{nb,nb} & Y_{nb,nc} & Y_{nb,ma} & Y_{nb,mb} & Y_{nb,mc} \\ Y_{nc,na} & Y_{nc,nb} & Y_{nc,nc} & Y_{nc,ma} & Y_{nc,mb} & Y_{nc,mc} \\ Y_{ma,na} & Y_{ma,nb} & Y_{ma,nc} & Y_{ma,ma} & Y_{ma,mb} & Y_{ma,mc} \\ Y_{mb,na} & Y_{mb,nb} & Y_{mb,nc} & Y_{mb,ma} & Y_{mb,mb} & Y_{mb,mc} \\ Y_{mc,na} & Y_{mc,nb} & Y_{mc,nc} & Y_{mc,ma} & Y_{mc,mb} & Y_{mc,mc} \end{bmatrix} * \begin{bmatrix} V_{na} \\ V_{nb} \\ V_{nc} \\ V_{ma} \\ V_{mb} \\ V_{mc} \end{bmatrix}$$

Modeling A Line in OpenDSS

1. Using Wire and Line Geometry Data

Verifying the steps of computing $\boldsymbol{Y}_{prim}$ (ignoring the shunt capacitance):
Computed $\boldsymbol{Y}_{prim}$ using the aforementioned equations:

$$\boldsymbol{Y}_{prim} = \begin{bmatrix} 0.4866 - j1.009 & -0.2364 + j0.3463 & -0.08530 + j0.2250 & -0.4866 + j1.009 & 0.2364 - j0.3463 & 0.08530 - j0.2250 \\ -0.2364 + j0.3463 & 0.5441 - j1.0455 & -0.1448 + j0.2746 & 0.2364 - j0.3463 & -0.5441 + j1.0455 & 0.1448 - j0.2746 \\ -0.08530 + j0.2250 & -0.1448 + j0.2746 & 0.4372 - j0.9737 & 0.08530 - j0.2250 & 0.1448 - j0.2746 & -0.4372 + j0.9737 \\ -0.4866 + j1.009 & 0.2364 - j0.3463 & 0.08530 - j0.2250 & 0.4866 - j1.009 & -0.2364 + j0.3463 & -0.08530 + j0.2250 \\ 0.2364 - j0.3463 & -0.5441 + j1.0455 & 0.1448 - j0.2746 & -0.2364 + j0.3463 & 0.5441 - j1.0455 & -0.1448 + j0.2746 \\ 0.08530 - j0.2250 & 0.1448 - j0.2746 & -0.4372 + j0.9737 & -0.08530 + j0.2250 & -0.1448 + j0.2746 & 0.4372 - j0.9737 \end{bmatrix}$$

Exported $\boldsymbol{Y}_{prim}$ from OpenDSS:

0.4866	-1.009	-0.236	0.3463	-0.085	0.225	-0.487	1.0089	0.2364	-0.3463	0.0853	-0.225
-0.236	0.3463	0.5442	-1.046	-0.145	0.2746	0.2364	-0.346	-0.544	1.0455	0.1448	-0.2746
-0.085	0.225	-0.145	0.2746	0.4372	-0.974	0.0853	-0.225	0.1448	-0.2746	-0.437	0.9738
-0.487	1.0089	0.2364	-0.346	0.0853	-0.225	0.4866	-1.009	-0.236	0.3463	-0.085	0.225
0.2364	-0.346	-0.544	1.0455	0.1448	-0.275	-0.236	0.3463	0.5442	-1.0455	-0.145	0.2746
0.0853	-0.225	0.1448	-0.275	-0.437	0.9738	-0.085	0.225	-0.145	0.2746	0.4372	-0.9738

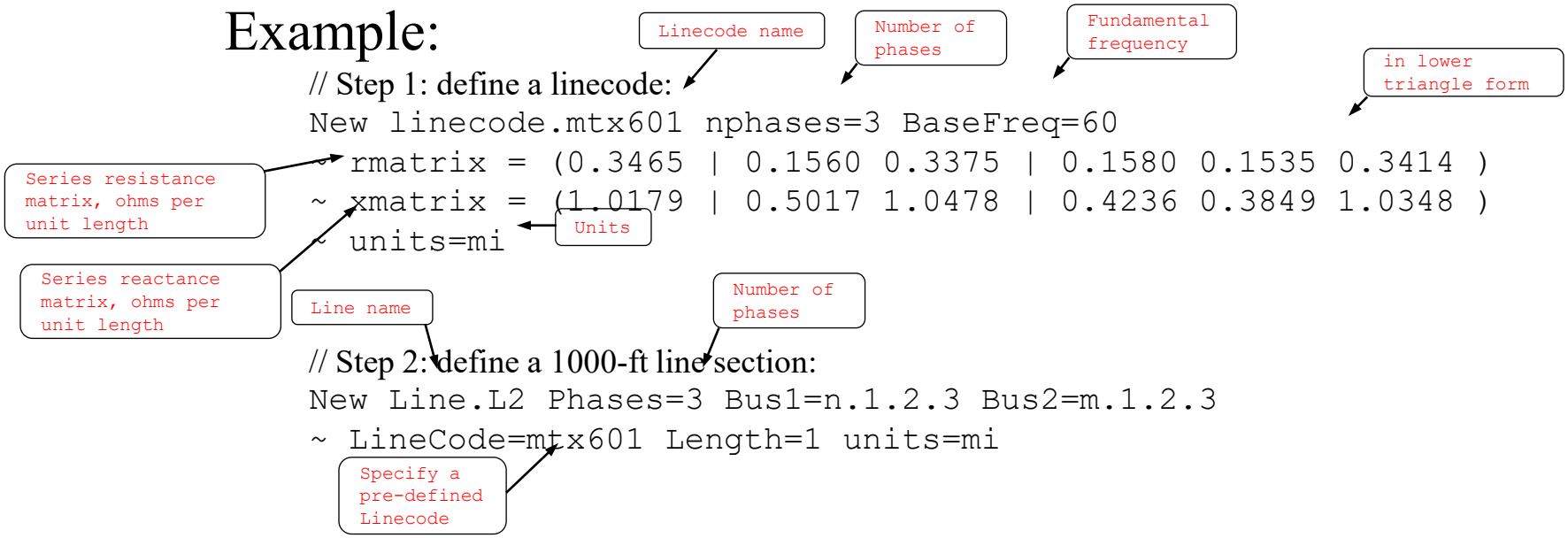
G B

Modeling A Line in OpenDSS

2. Using Linecode

- LineCode is a general ***library*** that contains impedance characteristics for lines.

Example:



Modeling A Line in OpenDSS

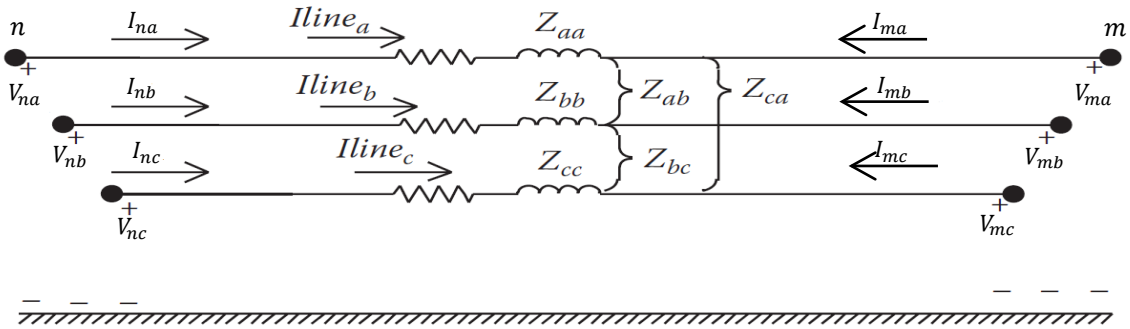
2. Using Linecode

How to build the primitive Y matrix for a line?

$\hat{z}_{ii} = rmatrix_{ii} * Length$

$\hat{z}_{ij} = rmatrix_{ij} * Length \qquad i, j = a, b, c$

Thus, we can obtain $\boldsymbol{Z}_{abc}$.



$$\boldsymbol{V}^n_{abc} = \boldsymbol{V}^m_{abc} + \boldsymbol{Z}_{abc}\boldsymbol{I}^n_{abc}$$

$$\boldsymbol{I}^n_{abc} = \boldsymbol{Z}^{-1}_{abc}(\boldsymbol{V}^n_{abc} - \boldsymbol{V}^m_{abc})$$

$$\boldsymbol{I}^n_{abc} = [\boldsymbol{Z}^{-1}_{abc} \quad -\boldsymbol{Z}^{-1}_{abc}] * \begin{bmatrix} \boldsymbol{V}^n_{abc} \\ \boldsymbol{V}^m_{abc} \end{bmatrix}$$

$$\boldsymbol{V}^m_{abc} = \boldsymbol{V}^n_{abc} + \boldsymbol{Z}_{abc}\boldsymbol{I}^m_{abc}$$

$$\boldsymbol{I}^m_{abc} = \boldsymbol{Z}^{-1}_{abc}(\boldsymbol{V}^m_{abc} - \boldsymbol{V}^n_{abc})$$

$$\boldsymbol{I}^m_{abc} = [-\boldsymbol{Z}^{-1}_{abc} \quad \boldsymbol{Z}^{-1}_{abc}] * \begin{bmatrix} \boldsymbol{V}^n_{abc} \\ \boldsymbol{V}^m_{abc} \end{bmatrix}$$

where, $\boldsymbol{V}^n_{abc} = \begin{bmatrix} V_{na} \\ V_{nb} \\ V_{nc} \end{bmatrix}, \boldsymbol{V}^m_{abc} = \begin{bmatrix} V_{ma} \\ V_{mb} \\ V_{mc} \end{bmatrix}, \boldsymbol{I}^n_{abc} = \begin{bmatrix} I_{na} \\ I_{nb} \\ I_{nc} \end{bmatrix}, \boldsymbol{I}^m_{abc} = \begin{bmatrix} I_{ma} \\ I_{mb} \\ I_{mc} \end{bmatrix}.$

Modeling A Line in OpenDSS

2. Using Linecode

How to build the primitive Y matrix for a line?

$$\begin{cases} \boldsymbol{I}^{n}_{abc} = [\boldsymbol{Z}^{-1}_{abc} \quad -\boldsymbol{Z}^{-1}_{abc}] * \begin{bmatrix} \boldsymbol{V}^{n}_{abc} \\ \boldsymbol{V}^{m}_{abc} \end{bmatrix} \\ \boldsymbol{I}^{m}_{abc} = [-\boldsymbol{Z}^{-1}_{abc} \quad \boldsymbol{Z}^{-1}_{abc}] * \begin{bmatrix} \boldsymbol{V}^{n}_{abc} \\ \boldsymbol{V}^{m}_{abc} \end{bmatrix} \end{cases} \Rightarrow \begin{bmatrix} \boldsymbol{I}^{n}_{abc} \\ \boldsymbol{I}^{m}_{abc} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Z}^{-1}_{abc} & -\boldsymbol{Z}^{-1}_{abc} \\ -\boldsymbol{Z}^{-1}_{abc} & \boldsymbol{Z}^{-1}_{abc} \end{bmatrix} * \begin{bmatrix} \boldsymbol{V}^{n}_{abc} \\ \boldsymbol{V}^{m}_{abc} \end{bmatrix}$$

$$\Rightarrow \boldsymbol{I} = \boldsymbol{Y}_{prim} * \boldsymbol{V}$$

Where, $\boldsymbol{Y}_{prim} = \begin{bmatrix} \boldsymbol{Z}^{-1}_{abc} & -\boldsymbol{Z}^{-1}_{abc} \\ -\boldsymbol{Z}^{-1}_{abc} & \boldsymbol{Z}^{-1}_{abc} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Y}_{abc} & -\boldsymbol{Y}_{abc} \\ -\boldsymbol{Y}_{abc} & \boldsymbol{Y}_{abc} \end{bmatrix}, \boldsymbol{I} = \begin{bmatrix} \boldsymbol{I}^{n}_{abc} \\ \boldsymbol{I}^{m}_{abc} \end{bmatrix}, \boldsymbol{V} = \begin{bmatrix} \boldsymbol{V}^{n}_{abc} \\ \boldsymbol{V}^{m}_{abc} \end{bmatrix}$

In a detailed form:

$$\begin{bmatrix} I_{na} \\ I_{nb} \\ I_{nc} \\ I_{ma} \\ I_{mb} \\ I_{mc} \end{bmatrix} = \begin{bmatrix} Y_{na,na} & Y_{na,nb} & Y_{na,nc} & Y_{na,ma} & Y_{na,mb} & Y_{na,mc} \\ Y_{nb,na} & Y_{nb,nb} & Y_{nb,nc} & Y_{nb,ma} & Y_{nb,mb} & Y_{nb,mc} \\ Y_{nc,na} & Y_{nc,nb} & Y_{nc,nc} & Y_{nc,ma} & Y_{nc,mb} & Y_{nc,mc} \\ Y_{ma,na} & Y_{ma,nb} & Y_{ma,nc} & Y_{ma,ma} & Y_{ma,mb} & Y_{ma,mc} \\ Y_{mb,na} & Y_{mb,nb} & Y_{mb,nc} & Y_{mb,ma} & Y_{mb,mb} & Y_{mb,mc} \\ Y_{mc,na} & Y_{mc,nb} & Y_{mc,nc} & Y_{mc,ma} & Y_{mc,mb} & Y_{mc,mc} \end{bmatrix} * \begin{bmatrix} V_{na} \\ V_{nb} \\ V_{nc} \\ V_{ma} \\ V_{mb} \\ V_{mc} \end{bmatrix}$$

Modeling A Line in OpenDSS

2. Using Linecode

Verifying the steps of computing $\boldsymbol{Y}_{prim}$ (ignoring the shunt capacitance):

Computed $\boldsymbol{Y}_{prim}$ using the aforementioned equations:

$$\boldsymbol{Y}_{prim} = \begin{bmatrix} 0.4338 - j1.2502 & -0.1840 + j0.46220 & -0.1008 + j0.34550 & -0.4338 + j1.2502 & 0.1840 - 0j.46220 & 0.1008 - j0.34550 \\ -0.1840 + j0.46220 & 0.3798 - j1.1847 & -0.04780 + j0.26390 & 0.1840 - j0.46220 & -0.3798 + j1.1847 & 0.04780 - j0.26390 \\ -0.1008 + j0.34550 & -0.04780 + j0.26390 & 0.3359 - j1.1176 & 0.1008 - j0.34550 & 0.04780 - j0.26390 & -0.3359 + j1.1176 \\ -0.4338 + j1.2502 & 0.1840 - j0.46220 & 0.1008 - j0.34550 & 0.4338 - j1.2502 & -0.1840 + j0.46220 & -0.1008 + j0.34550 \\ 0.1840 - j0.46220 & -0.3798 + j1.1847 & 0.04780 - j0.26390 & -0.1840 + j0.46220 & 0.3798 - j1.1847 & -0.04780 + j0.26390 \\ 0.1008 - j0.34550 & 0.04780 - j0.26390 & -0.3359 + j1.1176 & -0.1008 + j0.34550 & -0.04780 + j0.26390 & 0.3359 - j1.1176 \end{bmatrix}$$

Exported $\boldsymbol{Y}_{prim}$ from OpenDSS:

0.4338	-1.25	-0.184	0.4622	-0.101	0.3455	-0.434	1.2502	0.184	-0.4622	0.1008	-0.3455
-0.184	0.4622	0.3798	-1.185	-0.048	0.2639	0.184	-0.462	-0.38	1.18471	0.0478	-0.2639
-0.101	0.3455	-0.048	0.2639	0.3359	-1.118	0.1008	-0.346	0.0478	-0.2639	-0.336	1.11765
-0.434	1.2502	0.184	-0.462	0.1008	-0.346	0.4338	-1.25	-0.184	0.46223	-0.101	0.34554
0.184	-0.462	-0.38	1.1847	0.0478	-0.264	-0.184	0.4622	0.3798	-1.1847	-0.048	0.2639
0.1008	-0.346	0.0478	-0.264	-0.336	1.1176	-0.101	0.3455	-0.048	0.2639	0.3359	-1.1176

G B

Contents

- General object description
- Line
- Source and other objects
- Power delivery elements
- Power conversion elements
- Control elements
- Meter elements

Vsource Object

Voltage source. This is a special power conversion element. It is special because voltage sources are used to initialize the power flow solution with all other injection sources set to zero.

A Vsource object is a two-terminal, multi-phase Thevenin equivalent. That is, it is a voltage source behind an impedance. The data are specified as it would commonly be for a power system source equivalent: Line-line voltage (kV) and short circuit MVA.

The most common way to use a voltage source object is with the first terminal connected to the bus of interest with the second terminal connected to ground (voltage reference). In this usage, the connection of the second terminal may be omitted. In 2009, the voltage source was changed from a single-terminal device to a two-terminal device. This allows for the connection of a voltage source between two buses, which is convenient for some types of studies.

Vsource Object

The properties are:

Bus1 = Name of bus to which the source's first terminal is connected. Remember to specify the node order if the terminals are connected in some unusual manner. Side effect: The processing of this property results in the setting of the Bus2 property so that all conductors in terminal 2 are connected to ground.
Bus2 = Name of bus to which the source's second terminal is connected. If omitted, the second terminal is connected to ground (node 0) at the bus designated by the Bus1 property.
Basekv = base or rated Line-to-line kV.
Pu = Actual per unit at which the source is operating. Assumed balanced for all phases.
Angle = Base angle, degrees, of the first phase.
Frequency = frequency of the source.
Phases = Number of phases. Default = 3.0.
Mvasc3 = 3-phase short circuit MVA= kVBase^2/Z_{SC}
Mvasc1 = 1-phase short circuit MVA.
…

Vsource Object

The properties are, in order:

…
x1r1 = Ratio of X1/R1. Default = 4.0.
x0r0 = Ratio of X0/R0. Default = 3.0.
Isc3 = Alternate method of defining the source impedance. 3-phase short circuit current, amps. Default is 10000.
Isc1 = Alternate method of defining the source impedance. single-phase short circuit current, amps. Default is 10500.
R1 = Alternate method of defining the source impedance. Positive-sequence resistance, ohms. Default is 1.65.
X1 = Alternate method of defining the source impedance. Positive-sequence reactance, ohms. Default is 6.6.
R0 = Alternate method of defining the source impedance. Zero-sequence resistance, ohms. Default is 1.9.
X0 = Alternate method of defining the source impedance. Zero-sequence reactance, ohms. Default is 5.7.
ScanType = {pos*| zero | none} Maintain specified symmetrical component sequence to assume for Harmonic mode solution. Default is positive sequence.
…

Vsource Object

The properties are, in order:

…

Sequence = {pos*| neg | zero} Set the phase angle relationships for the specified symmetrical component sequence for solution modes other than Harmonics. Default is positive sequence.

Spectrum = Name of harmonic spectrum for this source. Default is "defaultvsource", which is defined when the DSS starts.

Z1 = Positive-sequence impedance, ohms, as a 2-element array representing a complex number.

Z2 Negative-sequence impedance, ohms, as a 2-element array representing a complex number.

Z0 = Zero-sequence impedance, ohms, as a 2-element array representing a complex number.

BaseFreq = Base Frequency for impedance specifications. Default is 60 Hz.

Like = Name of an existing Vsource object on which to base this one.

Isource Object

Current source. This is a one-terminal current source object that can be connected to any bus. Its most common use is likely to be used to represent harmonic sources and to be used in frequency response scans of circuit models. You can perform positive- or zero-sequence scans.

You can generally attach as many Isource objects to a bus as you want. An Isource is assumed to be ideal and its Yprim matrix is zero.

The properties are:

```
phases = Number of phases. Defaults to 3. For 3 or less, phase
shift defaults to 120 degrees.
bus1 = Name of bus to which source is connected.
       bus1=busname
       bus1=busname.1.2.3
Amps = Magnitude of current source, each phase, in Amps.
angle = Phase angle in degrees of first phase: e.g.,Angle=10.3.
Phase shift between phases defaults to 120 degrees when number
of phases <= 3
...
```

Isource Object

...

frequency = Source frequency. Defaults to circuit fundamental frequency.
Scantype = {pos*| zero | none} Maintain specified sequence for harmonic solution. Default is positive sequence. Otherwise, angle between phases rotates with harmonic.
Sequence = {pos*| neg | zero} Set the phase angles for the specified symmetrical component sequence for solution modes other than Harmonics. Default is positive sequence.
Spectrum = Harmonic spectrum assumed for this source. Default is "default".

Inherited properties:

basefreq = Base Frequency for ratings.
enabled = {Yes|No or True|False} Indicates whether this element is enabled.
like = Make like another object.

Fault Object

A Fault object is a resistor network that can be configured in a variety of ways. It is nominally designed with the same philosophy as the Capacitor. That is, it is a two-terminal device in which the second terminal defaults to ground. This is often what is desired when simulating a fault. However, the OpenDSS Fault object may be configured to do much more and can be configured to represent any type of fault. For example, it can be connected between transmission overbuild and distribution underbuild to simulate the transmission falling on the distribution circuit.

A Fault object is a standard Power Delivery component. You can have as many Fault objects on the circuit as you wish (some may cause the power flow solution to diverge – if so, switch to a direct solution or a dynamic solution). For Monte Carlo fault mode (MF) you will distribute Fault objects all over the circuit is some proportion. The solver will enable one at a time for each solution.

Fault Object

Since the Fault object is nothing more than a resistor network, you may use it for purposes other than modeling short circuit faults – anything that requires a resistor model. With the Gmatrix property a very complex resistive network can be modeled. For example, you may wish to represent a fault that has one resistance L-L and another L-ground. Note that it is specified as a nodal conductance matrix.

In time mode simulations such as Dynamics, the initiation of the fault can be delayed (Ontime property) and it will automatically clear itself when the current drops below a certain level(MinAmps property) if the fault is declared to be Temporary.

The properties are:

```
phases = Number of Phases. Default is 1.
Bus1 = Name of first bus.
Bus2 = Name of 2nd bus. Defaults to all phases connected to
first bus, node 0, if not specified. (Shunt Wye to ground
connection)
R = Resistance, each phase, ohms. Default is 0.0001.
...
```

Fault Object

The properties are:

…

Gmatrix = Use this to specify a nodal conductance (G) matrix to represent some arbitrary resistance network.
MinAmps = Minimum amps that can sustain a temporary fault. Default is 5.
ONtime = Time (sec) at which the fault is established for time varying simulations. Default is 0.0 (on at the beginning of the simulation).
pctperm = Percent of failures that become permanent. (not used)
Temporary = {Yes | No} Default is No. Designate whether the fault is temporary. For Timevarying simulations, the fault will be removed if the current through the fault drops below the MINAMPS criteria.
%stddev = Percent standard deviation in resistance to assume for Monte Carlo fault (MF) solution mode for GAUSSIAN distribution. Default is 0 (no variation from mean).

Contents

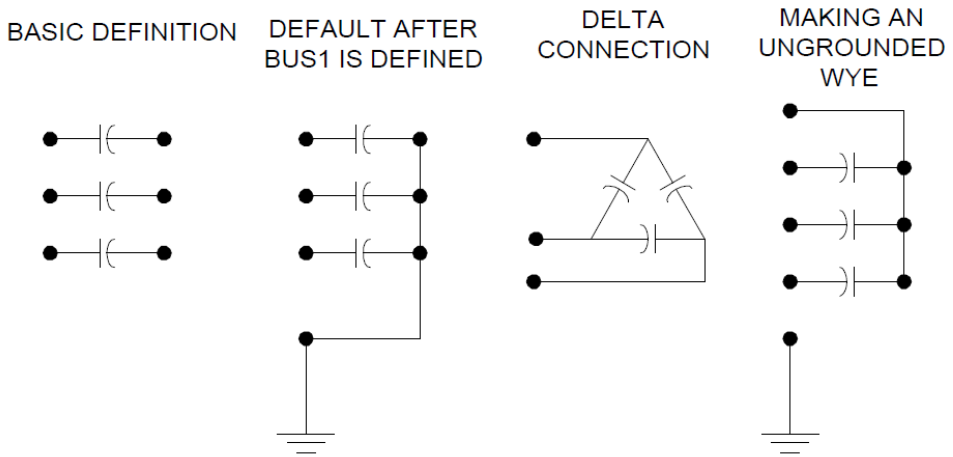
- General object description
- Line
- Source and other objects
- Power delivery elements
- Power conversion elements
- Control elements
- Meter elements

Capacitor Object

The capacitor model is basically implemented as a two-terminal power delivery element. However, if you don't specify a connection for the second bus, it will default to the 0 node (ground reference) of the same bus to which the first terminal is connected. That is, it defaults to a grounded wye (star) shunt capacitor bank.

If you specify the connection to be "delta" then the second terminal is eliminated.

If you wish a series capacitor, simply specify a second bus connection.



Capacitor Object

If you wish an ungrounded wye capacitor, set all the second terminal conductors to an empty node on the first terminal bus, e.g.:

```
Bus1=B1 bus2 = B1.4.4.4 ! for a 3-phase capacitor
```

Of course, any other connection is possibly by explicitly specifying the nodes.

While the Capacitor object may be treated as if it were a single capacitor bank, it is actually implemented as a multistep tuned filter bank. Many of the properties can be specified as arrays. See Numsteps property.

If you wish to represent a filter bank, specify either the XL or Harm property. When a Capcontrol object switches a capacitor, it does so by incrementing or decrementing the active step.

The properties are:

```
Bus1 = Definition for the connection of the first bus.
```

…

Capacitor Object

The properties are:

…

Bus2 = Bus connection for second terminal.
Phases = Number of phases. Default is 3.
Kvar = Most common of three ways to define a power capacitor. Rated kvar at rated kV, total of all phases. Each phase is assumed equal.
Kv = Rated kV of the capacitor (not necessarily same as bus rating).
Conn = Connection of bank. One of {wye | ln} for wye connected banks or {delta | ll} for delta (line-line) connected banks. Default is wye (or straight-through for series capacitor).
Cmatrix = Alternate method of defining a capacitor bank.
Cuf = Alternate method of defining a capacitor bank.
R = ARRAY of series resistance in each phase (line), ohms. Default is 0.0
XL = ARRAY of series inductive reactance(s) in each phase (line) for filter, ohms at base frequency. Use this OR "h" property to define filter. Default is 0.0.

…

Capacitor Object

The properties are:

…

Harm = ARRAY of harmonics to which each step is tuned. Zero is interpreted as meaning zero reactance (no filter). Default is zero.

Numsteps = Number of steps in this capacitor bank. Default = 1. Forces reallocation of the capacitance, reactor, and states array.

states = ARRAY of integers {1|0} states representing the state of each step (on|off). Defaults to 1 when reallocated (on). Capcontrol will modify this array as it turns steps on or off.

Normamps = Normal current rating. Automatically computed if kvar is specified. Otherwise, you need to specify if you wish to use it.

Emergamps = Overload rating. Defaults to 135% of Normamps.

Faultrate = Annual failure rate. Failure events per year. Default is 0.0005.

Pctperm = Percent of faults that are permanent. Default is 100.0.

Basefreq = Base frequency, Hz. Default is 60.0

Like = Name of another Capacitor object on which to base this one.

Line Object

The Line element is used to model most multi-phase, two-port lines or cables. It is a "Pi" model with shunt capacitance. This is a Power Delivery element described by its impedance. Impedances may be specified by symmetrical component values or by matrix values. Alternatively, you may simply refer to an existing LineCode object from which the impedance values will be copied. Or you may specify an existing Geometry object and the line impedances will be computed.

You can define the line impedance at a base frequency directly in a Line object definition or you can import the impedance definition from a LineCode object. Both of these definitions of impedance are quite similar except that the LineCode object can perform Kron reduction. (See LineCode Object).

If the Geometry property is specified, all previous definitions are ignored. The DSS will compute the impedance matrices from the specified geometry each time the frequency changes.

Line Object

Whichever definition is the most recent applies, as with nearly all DSS functions.

Note the **units** property; you can declare any length measurement in whatever units you please. Internally, everything is converted to meters. Just be sure to declare the units. Otherwise, they are assumed to be compatible with other data or irrelevant.

The default Line object is a 1000-ft overhead line with 336 MCM ACSR conductor on a 8-ft crossarm. It is defined by symmetrical component values:

```
R1 = 0.0580 ohms per 1000 ft
X1 = 0.1206 ohms per 1000 ft
R0 = 0.1784 ohms per 1000 ft
X0 = 0.4047 ohms per 1000 ft
C1 = 3.4e-9 nF per 1000ft
C0 = 1.6e-9 nF per 1000ft
```

The corresponding values of Rg and Xg for Rho=100 and 60 Hz are:

Line Object

Rg = 0.01805 ohms per 1000 ft
Xg = 0.155081 ohms per 1000 ft

The properties, are:

Bus1 = Name of bus for terminal 1. Node order definitions optional.
Bus2 = Name of bus for terminal 2.
Linecode = Name of an existing LineCode object containing impedance definitions.
Length = Length multiplier to be applied to the impedance data.
Phases = No. of phases. Default = 3.
R1 = positive-sequence resistance, ohms per unit length.
X1 = positive-sequence reactance, ohms per unit length.
R0 = zero-sequence resistance, ohms per unit length.
X0 = zero-sequence reactance, ohms per unit length.
C1 = positive-sequence capacitance, nanofarads per unit length. Setting any of R1, R0, X1, X0, C1, C0 forces the program to use the symmetrical component line definition.
C0 = zero-sequence capacitance, nanofarads per unit length.
B1 = Alternate way to enter C1, microS per unit length.
B0 = Alternate way to enter C0, microS per unit length.

...

Line Object

The properties, are:

…

Normamps = Normal ampacity, amps.
Emergamps = Emergency ampacity, amps. Usually the one-hour rating.
Faultrate = Number of faults per year per unit length. This is the default for this general line construction.
Pctperm = Percent of the faults that become permanent (requiring a line crew to repair and a sustained interruption).
Repair = Hours to repair.
BaseFreq = Base Frequency at which the impedance values are specified. Default = 60.0 Hz.
Rmatrix = Series resistance matrix, ohms per unit length. See Command Language for syntax. Lower triangle form is acceptable.
Xmatrix = Series reactance matrix, ohms per unit length.
Cmatrix = Shunt nodal capacitance matrix, nanofarads per unit length.
Switch = {y/n | T/F} Default= no/false. Designates this line as a switch for graphics and algorithmic purposes.

…

Line Object

The properties, are:

…

Rg = Carson earth return resistance per unit length used to compute impedance values at base frequency.
Xg = Carson earth return reactance per unit length used to compute impedance values at base frequency.
Rho = Earth resistivity used to compute earth correction factor. Overrides Line geometry definition if specified. Default=100 meter ohms.
Geometry = Geometry code for LineGeometry Object. Supercedes any previous definition of line impedance. Line constants are computed for each frequency change or rho change.
EarthModel = One of {Carson | FullCarson | Deri}. Default is the global value established with the Set EarthModel option. See the Options Help on EarthModel option. This is used to override the global value for this line. This option applies only when the "geometry" property is used.
Units = Length Units = {none | mi | kft | km | m | Ft | in | cm } Default is None - assumes length units match impedance units.
Like = Name of an existing Line object to build this like.

Reactor Object

The Reactor element is implemented with basically the same philosophy as a Capacitor (and a Fault object). It is a constant impedance element that may be configured into a variety of connections. Like the Capacitor, the second terminal defaults to a Wye connection to ground if not specified. In that case, it is flagged as a Shunt element. It is a constant impedance element entirely represented by its primitive Y matrix.

The reactor may be conceived as a series R-L or a parallel R-L connection (see Parallel property). By default it is an inductance with series resistance. You may also specify a resistor, Rp, in parallel with the entire branch. These options allow the Reactor impedance characteristic to have different frequency response characteristics, which are often useful for some Harmonicsmode simulations, particularly for filters. In the case of a shunt reactor, the Rp value is used for no-load losses.

Reactor Object

Zero-impedance devices cannot exist in OpenDSS. However, either R or X can be zero in this model. A Reactor element can be used for source equivalent or other equivalent impedances where the capacitance of a Line element is unnecessary. Use a 1-phase reactor to model neutral reactors in transformers, generators, or loads. See the Tech Notes on the OpenDSS Wiki for further details.

By default, the Reactor has no coupling between the phases. Shunt reactors would typically be defined by kV and kvar properties, similar to a capacitor. Series reactors without mutual coupling would be defined by the R and X properties. Of course, either could be used in all circumstances. Note that if the connection is specified as "delta" only the first terminal matters; there is no second terminal. This applies only to shunt reactors. By leaving the Conn property as wye and specifying both terminal connections explicitly, nearly any reactor configuration can be achieved. Mutual coupling between phases can be achieved by specifying Rmatrix and Xmatrix properties. Note that the matrix specification is mutually exclusive with the other means of specifying the reactance values. Of course, the Rmatrix and Xmatrix properties may be defined with zero mutual coupling.

Reactor Object

The properties, are:

phases = Number of phases.
bus1 = Name of first bus.
bus2 = Name of 2nd bus. Defaults to all phases connected to first bus, node 0. (Shunt Wye Connection).
kv = For 2, 3-phase, kV phase-phase. Otherwise specify actual coil rating.
kvar = Total kvar, all phases. Evenly divided among phases. Only determines X. Specify R separately
conn = {wye | delta |LN |LL} Default is wye, which is equivalent to LN. If Delta, then only one terminal.
Parallel = {Yes | No} Default=No. Indicates whether Rmatrix and **Xmatrix** are to be considered to be in parallel. This makes a significant difference in harmonic studies. Default is series. For other models, specify R and Rp.
R = Resistance (in series with reactance), each phase, ohms.
Rmatrix = Resistance matrix, lower triangle, ohms at base frequency. Order of the matrix is the number of phases. Mutually exclusive to specifying parameters by kvar or R.
Rp = Resistance in parallel with R and X (the entire branch). Assumed infinite if not specified.

...

Reactor Object

The properties, are:

…

X = Reactance, each phase, ohms at base frequency.
Xmatrix = Reactance matrix, lower triangle, ohms at base frequency. Order of the matrix is the number of phases. Mutually exclusive to specifying parameters by kvar or X.
Z = Alternative way of defining R and X properties. Enter a 2-element array representing R +jX in ohms.
Z1 = Positive-sequence impedance, ohms, as a 2-element array representing a complex number.
Z2 = Negative-sequence impedance, ohms, as a 2-element array representing a complex number.
Z0 = Zero-sequence impedance, ohms, as a 2-element array representing a complex number.

Properties inherited from the circuit element class:

normamps = Normal rated current.
emergamps = Maximum current.
repair = Hours to repair.
faultrate = No. of failures per year.
Pctperm = Percent of failures that become permanent.

…

Reactor Object

Properties inherited from the circuit element class:

…

basefreq = Base Frequency for ratings.
enabled = {Yes|No or True|False} Indicates whether this element is enabled.
like = Make like another object.

Transformer Object

The Transfomer is implemented as a multi-terminal (two or more) power delivery element.

A transfomer consists of two or more *Windings*, connected in somewhat arbitrary fashion (with a default Wye-Delta connection). You can specify the parameters one winding at a time or use arrays to set all the winding values at once. Use the "wdg=…" parameter to select a winding for editing.

Transformers have one or more *phases*. The number of conductors per terminal is always one more than the number of phases. For wye- or star-connected windings, the extra conductor is the neutral point. For delta-connected windings, the extra terminal is open internally (you normally leave this connected to node 0).

The properties, are:

```
Phases = Number of phases. Default is 3.
Windings = Number of windings. Default is 2.
```

…

Transformer Object

The properties, are:

…

Note: For defining the winding values one winding at a time, use the following parameters. Always start the winding definition with "wdg = …" when using this method of defining transformer parameters. The remainder of the tags are optional as usual if you keep them in order.

Wdg = Integer representing the winding which will become the active winding for subsequent data.

Bus = Definition for the connection of this winding (each winding is connected to one terminal of the transformer and, hence, to one bus).

Conn = Connection of this winding. One of {wye | ln} for wye connected banks or {delta | ll} for delta (line-line) connected banks. Default is wye.

Kv = Rated voltage of this winding, kV. For transformers designated 2- or 3-phase, enter phase-to-phase kV. For all other designations, enter actual winding kV rating. Two-phase transformers are assumed to be employed in a 3-phase system. Default is 12.47 kV.

…

Transformer Object

The properties, are:

…

Kva = Base kVA rating (OA rating) of this winding.
Tap = Per unit tap on which this winding is set.
%R = Percent resistance of this winding on the rated kVA base. (Reactance is between two windings and is specified separately -- see below.)
rneut = Neutral resistance to ground in ohms for this winding. Ignored if delta winding. For open ungrounded neutral, set to a negative number. Default is -1 (capable of being ungrounded).
xneut = Neutral reactance in ohms for this winding. Ignored if delta winding. Assumed to be in series with neutral resistance. Default is 0.
Note: Use the following properties to set the winding values using arrays (setting of wdg= … is ignored). The names of these properties are simply the plural form of the property name above.
Buses = Array of bus definitions for windings [1, 2. …].
Conns = Array of winding connections for windings [1, 2. …].
KVs = Array of kV ratings following rules stated above for the kV field for windings [1,2,…].

…

Transformer Object

The properties, are:

…

KVAs = Array of base kVA ratings for windings [1,2,…].
Taps = Array of per unit taps for windings [1,2,…].
%Rs = Array of percent resistances for windings [1, 2. …]
Use the following propertis to define the reactances of the transformer. For 2- and 3-winding transformers, you may use the conventional XHL, XLT, and XHT (or X12, X23, X13) parameters. You may also put the values in an array (xscarray), which is required for higher phase order transformers. There are always n*(n-1)/2 different short circuit reactances, where n is the number of windings. Always use the kVA base of the first winding for entering impedances. Impedance values are entered in percent.
XHL (or X12) = Percent reactance high-to-low (winding 1 to winding 2).
XLT (or X23) = Percent reactance low-to-tertiary (winding 2 to winding 3).
XHT (or X13) = Percent reactance high-to-tertiary (winding 1 to winding 3).

…

Transformer Object

The properties, are:

…

General transformer rating data:
Thermal = Thermal time constant, hrs. Default is 2.
n = Thermal exponent, n, from IEEE/ANSI C57. Default is 0.8.
m = Thermal exponent, m, from IEEE/ANSI C57. Default is 0.8.
flrise = Full-load temperature rise, degrees centigrade. Default is 65.
hsrise = Hot-spot temperatire rise, degrees centigrade. Default is 15.
%Loadloss = Percent Losses at rated load.. Causes the %r values to be set for windings 1 and 2.
%Noloadloss = Percent No load losses at nominal voltage. Default is 0. Causes a resistive branch to be added in parallel with the magnetizing inductance.
%imag = Percent magnetizing current. Default is 0. An inductance is used to represent the magnetizing current. This is embedded within the transformer model as the primitive Y matrix is being computed.

…

Transformer Object

The properties, are:

…

Ppm_Antifloat = Parts per million for anti floating reactance to be connected from each terminal to ground. Default is 1. That is, the diagonal of the primitive Y matrix is increased by a factor of 1.000001.
NormHKVA = Normal maximum kVA rating for H winding (1). Usually 100 - 110% of maximum nameplate rating.
EmergHKVA = Emergency maximum kVA rating for H winding (1). Usually 140-150% of maximum nameplate rating. This is the amount of loading that will cause 1% loss of life in one day.
Faultrate = Failure rate for transformer. Defaults to 0.007 per year. All are considered permanent.
Basefreq = Base frequency, Hz. Default is 60.0
Like = Name of another Transformer object on which to base this one.
Sub = Yes/No. Designates whether this transformer is to be treated as a substation. Default is No.

Contents

- General object description
- Line
- Source and other objects
- Power delivery elements
- Power conversion elements
- Control elements
- Meter elements

Load Object

A Load is a complicated Power Conversion element that is at the heart of many analyses. It is basically defined by its nominal kW and PF or its kW and kvar. Then it may be modified by a number of multipliers, including the global circuit load multiplier, yearly load shape, daily load shape, and a duty-cycle load shape.

The default is for the load to be a current injection source. Thus, its primitive Y matrix contains only the impedance that might exist from the neutral of a wye-connected load to ground. However, if the load model is switched to Admittance from PowerFlow (see Set LoadModel command), the load is converted to an admittance and included in the system Y matrix. This would be the model used for fault studies where convergence might not be achieved because of low voltages.

Loads are assumed balanced for the number of phases specified. If you would like unbalanced loads, enter separate single-phase loads.

Load Object

There are three legal ways to specify the base load:

1. kW, PF
2. kw, kvar
3. kVA, PF

If you send these properties in the order shown, the definition should work. If you deviate from these procedures, the result may or may not be what you want. (To determine if it has accomplished the desired effect, execute the Dump command for the desired load(s) and observe the settings.)

The properties are:

Bus1 = Name of bus to which the load is connected. Include node definitions if the terminal conductors are connected abnormally.
Phases = No. of phases this load.
Kv = Base voltage for load. For 2- or 3-phase loads, specified in phase-to-phase kV. For all other loads, the actual kV across the load branch. If wye (star) connected, then specify phase-to-neutral (L-N). If delta or phase-to-phase connected, specify the phase-to-phase (L-L) kV.

...

Load Object

The properties are:

…

Kw = nominal active power, kW, for the load. Total of all phases. See kVA.
Pf = nominal Power Factor for load. Negative PF is leading. Specify either PF or kvar (see below). If both are specified, the last one specified takes precedence.
Model = Integer defining how the load will vary with voltage (see "Power Conversion Elements" for more details). The load models currently implemented are:
1: Constant P and constant Q (Default): Commonly used for power flow studies
2: Constant Z (or constant impedance)
3: Constant P and quadratic Q
4: Exponential
5: Constant I (or constant current magnitude) Sometimes used for rectifier load
6: Constant P and fixed Q (at the nominal value)
7: Constant P and quadratic Q (i.e., fixed reactance)
8: ZIP (see ZIPV)

…

Load Object

The properties are:

…

Yearly = Name of Yearly load shape.
Daily = Name of Daily load shape.
Duty = name of Duty cycleload shape. Defaults to Daily load shape if not defined.
Growth = Name of Growth Shape Growth factor defaults to the circuit's default growth rate if not defined. (see Set %Growth command)
Conn = {wye | y | LN} for Wye (Line-Neutral) connection; {delta | LL} for Delta (Line-Line)
connection. Default = wye.
Kvar = Base kvar. If this is specified, supercedes PF. (see PF)
Rneut = Neutral resistance, ohms. If entered as negative, non-zero number, neutral is assumed open, or ungrounded. Ignored for delta or line-line connected loads.
Xneut = Neutral reactance, ohms. Ignored for delta or line-line connected loads. Assumed to be in series with Rneut value.
Status = {fixed| variable}. Default is variable. If fixed, then the load is not modified by multipliers; it is fixed at its defined base value.

…

Load Object

The properties are:

…

Class = Integer number segregating the load according to a particular class.
Vminpu = Default = 0.95. Minimum per unit voltage for which the MODEL is assumed to apply. Below this value, the load model reverts to a constant impedance model.
Vmaxpu = Default = 1.05. Maximum per unit voltage for which the MODEL is assumed to apply. Above this value, the load model reverts to a constant impedance model.
VminNorm = Minimum per unit voltage for load EEN evaluations, Normal limit.
VminEmerg = Minimum per unit voltage for load UE evaluations, Emergency limit. Default = 0, which defaults to system "vminemerg" property (see Set Command under Executive).
XfkVA = Default = 0.0. Rated kVA of service transformer for allocating loads based on connected kVA at a bus.
AllocationFactor = Default = 0.5. Allocation factor for allocating loads based on connected kVA at a bus.

…

Load Object

The properties are:

…

kVA = Definition of the Base load in kVA, total all phases. This is intended to be used in combination with the power factor (PF) to determine the actual load. Legal ways to define base load (kW and kvar):
kW, PF
kW, kvar
kVA, PF
XFKVA * Allocationfactor, PF
kWh/(kWhdays*24) * Cfactor, PF
%mean = Percent mean value for load to use for monte carlo studies if no loadshape is assigned to this load. Default is 50.
%stddev = Percent Std deviation value for load to use for monte carlo studies if no loadshape is assigned to this load. Default is 10.
CVRwatts = Exponential paramater that defines the relationship between voltage (V) and active power (P) based on the following:P/P0 = (V/V0)^CVRwatts. P0 is the nominal power of the load at the base voltage V0.

…

Load Object

The properties are:

…

CVRvars = Exponential paramater that defines the relationship between voltage (V0) and reactive power (Q0) based on the following: Q/Q0 = (V/V0)^CVRwars. Q0 is the nominal power of the load at the base voltage V0.
kWh = kWh billed for this period. Default is 0. See help on kVA and Cfactor and kWhDays.
kWhDays = Length of kWh billing period in days (24 hr days). Default is 30. Average demand is computed using this value.
CFactor = Factor relating average kW to peak kW. Default is 4.0. See kWh and kWhdays. See kVA.
CVRCurve = Default is NONE. Curve describing both watt and var factors as a function of time.
NumCust = Number of customers, this load. Default is 1.
spectrum = Name of harmonic current spectrum for this load. Default is "defaultload", which is defined when the DSS starts.

…

Load Object

The properties are:

…

ZIPV = Array of 7 coefficients:
1. First 3 are ZIP weighting factors for active power (should sum to 1)
2. Next 3 are ZIP weighting factors for reactive power (should sum to 1)
3. Last 1 is cut-off voltage in p.u. of base kV; load is 0 below this cut-off
No defaults; all coefficients must be specified if using model=8.

%SeriesRL = Percent of load that is series R-L for Harmonic studies. Default is 50. Remainder is assumed to be parallel R and L. This has a significant impact on the amount of damping observed in Harmonics solutions.

Basefreq = Base frequency for which this load is defined. Default is 60.0.

Like = Name of another Load object on which to base this one.

Generator Object

A Generator is a Power Conversion element similar to a Load object. Its rating is basically defined by its nominal kW and PF or its kW and kvar. Then it may be modified by a number of multipliers, including the global circuit load multiplier, yearly load shape, daily load shape, and a dutycycle load shape.

For power flow studies, the generator is essentially a negative load that can be dispatched. For Harmonics mode, the generator is converted to a voltage source behind the value specified for Xd" that approximately matches the power flow solution. For dynamics mode, the generator is converted to a voltage source behind an impedance with the impedance dependent on the model chosen.

If the dispatch value (DispValue property) is 0, the generator always follows the appropriate dispatch curve, which is simply a Loadshape object. If DispValue>0 then the generator only comes on when the global circuit load multiplier exceeds DispValue. When the generator is on, it always follows the dispatch curve appropriate for the type of solution being performed.

Generator Object

If you want to model a generator that is fully on whenever it is dispatched on, simply designate "Status=Fixed". The default is "Status=Variable" (i.e., it follows a dispatch curve. You could also define a dispatch curve that is always 1.0.

Generators have their own energy meters that record:

1. Total kwh
2. Total kvarh
3. Max kW
4. Max kVA
5. Hours in operation
6. $ (Price signal * energy generated)

Generator meters reset with the circuit energy meters and take a sample with the circuit energy meters as well. The Energy meters also used trapezoidal integration so that they are compatible with Load-Duration simulations.

Generator Object

Generator power models for power flow simulations are:

1. Constant P, Q (* dispatch curve, if appropriate).
2. Constant Z (For simple, approximate solution)
3. Constant P, |V| somewhat like a standard power flow with voltage magnitudes and
angles as the variables instead of P and Q.
4. Constant P, fixed Q. P follows dispatch; Q is always the same.
5. Constant P, fixed reactance. P follows dispatch, Q is computed as if it were a fixed
reactance.
6. User-written model
7. Current-limited constant P, Q model (like some inverters).

Most of the time you will use #1 for planning studies, assuming you want to specify a specific power. All load models can follow Loadshapes. Some follow only the P component while the Type 1 can follow both a P and Q characteristic.

Generator Object

The default is for the generator to be a current injection source (Norton equivalent). Thus, its primitive Y matrix is similar to a Load object with a nominal equivalent admittance at rated voltage included in the primitive Y matrix and the injection current representing the amount required to compensate the Norton equivalent to achieve the desired terminal current. However, if the generator model is switched to Admittance from PowerFlow (see Set Mode command), the generator terminal current is computed simply from equivalent admittance that is included in the system Y matrix.

Generator powers are assumed balanced over the number of phases specified. If you would like unbalanced generators, enter separate single-phase generators.

The properties, in numerical order, are:

```
bus1 = Name of bus to which the generator is connected.
Phases = No. of phases this generator.
```

...

Generator Object

The properties, in numerical order, are:

…

Kv = Base voltage for generator. For 2- or 3-phase generators, specified in phase-tophase kV. For all other generators, the actual kV across the generator branch. If wye (star) connected, specify the phase-to-neutral (L-N) kV. If delta or phase-tophase connected, specify the phase-to-phase (L-L) kV.
Kw = nominal kW for generator. Total of all phases.
Pf = nominal Power Factor for generator. Negative PF is leading (absorbing vars). Specify either PF or kvar (see below). If both are specified, the last one specified takes precedence.
Model= Integer defining how the generator will vary with voltage. Presently defined models are:
1: Generator injects a constant kW at specified power factor.
2: Generator is modeled as a constant admittance.
3: Const kW, constant kV. Somewhat like a conventional transmission power flow P-V generator.
4: Const kW, Fixed Q (Q never varies)

…

Generator Object

The properties, in numerical order, are:

…

5: Const kW, Fixed Q(as a constant reactance)
6: Compute load injection from User-written Model.(see usage of Xd, Xdp)
7: Constant kW, kvar, but current is limited when voltage is below Vminpu
Yearly = Name of Yearly load shape.
Daily = Name of Daily load shape.
Duty = name of Duty cycle load shape. Defaults to Daily load shape if not defined.
Dispvalue = Dispatch value. If = 0.0 then Generator follows dispatch curves. If > 0 then Generator is ON only when the global load multiplier exceeds this value. Then the generator follows dispatch curves (see also Status)
Conn = {wye | y | LN} for Wye (Line-Neutral) connection; {delta | LL} for Delta (Line-Line) connection. Default = wye.
Kvar = Base kvar. If this is specified, will supercede PF. (see PF)

…

Generator Object

The properties, in numerical order, are:

…

Rneut = Neutral resistance ohms. If entered as negative, non-zero number, neutral is assumed open, or ungrounded. Ignored for delta or line-line connected generators. Default is 0.
Xneut = Neutral reactance, ohms. Ignored for delta or line-line connected generators. Assumed to be in series with Rneut value.
Status = {fixed | variable}. If Fixed, then dispatch multipliers do not apply. The generator is always at full power when it is ON. Default is Variable (follows curves).
Class = Integer number segregating the generator according to a particular class.
Maxkvar = Maximum kvar limit for Model = 3. Defaults to twice the specified load kvar. Always reset this if you change PF or kvar properties.
Minkvar = Minimum kvar limit for Model = 3. Enter a negative number if generator can absorb vars. Defaults to negative of Maxkvar. Always reset this if you change PF or kvar properties.

…

Generator Object

The properties, in numerical order, are:

…

Pvfactor = Convergence deceleration factor for P-V generator model (Model=3). Default is 0.1. If the circuit converges easily, you may want to use a higher number such as 1.0. Use a lower number if solution diverges. Use Debugtrace=yes to create a file that will trace the convergence of a generator model.
Debugtrace = {Yes | No } Default is no. Turn this on to capture the progress of the generator model for each iteration. Creates a separate file for each generator named "GEN_name.CSV".
Vminpu = Default = 0.95. Minimum per unit voltage for which the Model is assumed to apply. Below this value, the generator model reverts to a constant impedance model. For Model 7, this is used to determine the upper current limit. For example, if Vminpu is 0.90 then the current limit is (1/0.90) = 111%.
Vmaxpu = Default = 1.05. Maximum per unit voltage for which the Model is assumed to apply. Above this value, the generator model reverts to a constant impedance model.

…

Generator Object

The properties, in numerical order, are:

…

ForceON = {Yes | No} Forces generator ON despite requirements of other dispatch modes. Stays ON until this property is set to NO, or an internal algorithm cancels the forced ON state.
kVA = kVA rating of electrical machine. Defaults to 1.2* kW if not specified. Applied to machine or inverter definition for Dynamics mode solutions.
MVA = MVA rating of electrical machine. Alternative to using kVA=.
Xd = Per unit synchronous reactance of machine. Presently used only for Thevinen impedance for power flow calcs of user models (model=6). Typically use a value from 0.4 to 1.0. Default is 1.0
Xdp = Per unit transient reactance of the machine. Used for Dynamics mode and Fault studies. Default is 0.27.For user models, this value is used for the Thevinen/Norton impedance for Dynamics Mode.
H = Per unit mass constant of the machine. MW-sec/MVA. Default is 1.0.
D = Damping constant. Usual range is 0 to 4. Default is 1.0. Adjust to get damping.

…

Generator Object

The properties, in numerical order, are:

…

UserModel = Name of DLL containing user-written model, which computes the terminal currents for Dynamics studies, overriding the default model. Set to "none" to negate previous setting.

UserData = String (in quotes or parentheses) that gets passed to user-written model for defining the data required for that model.

ShaftModel = Name of user-written DLL containing a Shaft model, which models the prime mover and determines the power on the shaft for Dynamics studies. Models additional mass elements other than the single-mass model in the DSS default model. Set to "none" to negate previous setting.

ShaftData = String that gets passed to user-written shaft dynamic model for defining the data for that model.

spectrum = Name of harmonic voltage or current spectrum for this generator. Voltage behind Xd" for machine - default. Current injection for inverter. Default value is "default", which is defined when the DSS starts.

Basefreq= Base frequency for which this generator is defined. Default is 60.0.

Like= Name of another Generator object on which to base this one.

Storage Object

Storage object is a recently added model into OpenDss, there is no description about it now. If you want to know more about storage object, you can search in the OpenDss forum or contact Roger Dugan.

However, the properties of storage object in summarized into a table, please refer to the OpenDss manual.

Contents

- General object description
- Line
- Source and other objects
- Power delivery elements
- Power conversion elements
- Control elements
- Meter elements

CapControl Object

One of the distinctive capabilities of the OpenDSS is that control elements are modeled separately from the power-carrying elements. This provides significant flexibility to create new models. Initially, control objects reflected only standard utility distribution system control devices. More recently, new, innovative models have been developed. Currently, there are nine controls provided with the program with more planned soon. This section gives some idea of how the control objects work and how you might exploit them.

Control elements are polled after a power flow solution has been obtained. If any control needs to operate, it places a message on the Control Queue. When it comes time for the control to operate, the message is popped off the queue and the command executed. This allows for the simulation of controls that have time delays in their operation.

CapControl Object

The typical execution of the Solve command for a single snapshot power flow solution is:

1. Initialize Snapshot (_InitSnap)
2. Repeat until converged:
   a. Solve Circuit (_SolveNoControl)
   b. Sample control devices (_SampleControls)
   c. Do control actions, if any (_DoControlActions)

The names in parentheses after the step is the corresponding command (see command reference above) that you would use if you wanted to "roll your own" solution method.

Control elements typically have a Sample function that samples the voltage and current at the terminal that the control is monitoring. Each element of the Control element class also has a DoPendingAction function that is called from the control queue at the appropriated time. The Control element then takes the prescribed action or decides that it doesn't need to (as is the case for many distribution system controls).

CapControl Object

The capacitor control monitors the voltage and current at a terminal of a PDelement or a PCelement and sends switching messages to a Capacitor object. The CapControl contains essential features of many typical utility capacitor controls.

Example:

```
New CapControl.C1ctrl element=Line.L1 Capacitor=C1
~ Type=Current ON=400 OFF=300 Delay=30
```

This control monitors the current in Line.L1 and switches Capacitor.C1 based on current magnitude. This is one of the simpler controls to implement. The capacitor will switch ON when the current in the monitored phase (defaults to 1) after a delay of 30 s. If the current drops below 400 A before 30 s has elapsed, the control resets (ignores the message from the control queue). After energization, the capacitor switches off when the current drops below 300 with the default DelayOFF.

The properties are:

…

CapControl Object

The properties are:

Element = Full object name of the circuit element, typically a line or transformer, to which the capacitor control's PT and/or CT are connected.There is no default; must be specified.

Capacitor = Name of Capacitor element which the CapControl controls. No Default; Must be specified.Do not specify the full object name; "Capacitor" is assumed for the object class. Example: Capacitor=cap1

Type = {Current | voltage | kvar | PF | time } Control type. Specify the ONsetting and OFFsetting appropriately with the type of control. (See help for ONsetting)

CTPhase Number of the phase being monitored for CURRENT control or one of {AVG | MAX | MIN} for all phases. Default=1. If delta or L-L connection, enter the first or the two phases being monitored [1-2, 2-3, 3-1]. Must be less than the number of phases.

Does not apply to kvar control which uses all phases by default.

CTratio = Ratio of the CT from line amps to control ampere setting for current and kvar control types.

...

CapControl Object

The properties are:

…

DeadTime = Dead time after capacitor is turned OFF before it can be turned back ON. Default is 300 sec.

Delay = Time delay, in seconds, from when the control is armed before it sends out the switching command to turn ON. The control may reset before the action actually occurs. This is used to determine which capacity control will act first. Default is 15. You may specify any floating point number to achieve a model of whatever condition is necessary.

DelayOFF = Time delay, in seconds, for control to turn OFF when present state is ON. Default is 15.

EventLog = {Yes/True* | No/False} Default is YES for CapControl. Log control actions to Eventlog.

OFFsetting = Value at which the control arms to switch the capacitor OFF. (See help for ONsetting)For Time control, is OK to have Off time the next day (< On time)

…

CapControl Object

The properties are:

…

ONsetting = Value at which the control arms to switch the capacitor ON (or ratchet up a step).

Type of Control: (1) Current: Line Amps / Ctratio; (2) Voltage: Line-Neutral (or Line-Line for delta) Volts / Ptratio; (3) kvar: Total kvar, all phases (3-phase for pos seq model). This is directional; (4) PF: Power Factor, Total power in monitored terminal. Negative for Leading; (5) Time: Hrs from Midnight as a floating point number (decimal). 7:30am would be entered as 7.5.

PTPhase = Number of the phase being monitored for VOLTAGE control or one of {AVG | MAX | MIN} for all phases. Default=1. If delta or L-L connection, enter the first or the two phases being monitored [1-2, 2-3, 3-1]. Must be less than the number of phases. Does not apply to kvar control which uses all phases by default.

…

CapControl Object

The properties are:

…

PTratio = Ratio of the PT that converts the monitored voltage to the control voltage. Default is 60. If the capacitor is Wye, the 1st phase line-to-neutral voltage is monitored. Else, the line-to-line voltage (1st - 2nd phase) is monitored.

terminal = Number of the terminal of the circuit element to which the CapControl is connected. 1 or 2, typically. Default is 1.

Vbus = Name of bus to use for voltage override function. Default is bus at monitored terminal. Sometimes it is useful to monitor a bus in another location to emulate various DMS control algorithms.

Vmax = Maximum voltage, in volts. If the voltage across the capacitor divided by the PTRATIO is greater than this voltage, the capacitor will switch OFF regardless of other control settings. Default is 126 (goes with a PT ratio of 60 for 12.47 kV system).

…

CapControl Object

The properties are:

…

Vmin = Minimum voltage, in volts. If the voltage across the capacitor divided by the PTRATIO is less than this voltage, the capacitor will switch ON regardless of other control settings. Default is 115 (goes with a PT ratio of 60 for 12.47 kV system).

VoltOverride = {Yes | No} Default is No. Switch to indicate whether VOLTAGE OVERRIDE is to be considered. Vmax and Vmin must be set to reasonable values if this property is Yes.

RegControl Object

This control is designed to emulate a standard utility voltage regulator or LTC control. It is attached to a particular winding of a transformer as the winding to monitor. It generally also adjusts the taps in that winding but could also be directed to control the taps in another winding.

The control has line drop compensator modeling by setting the R, X, CTprim, and ptratio properties. The control can also monitor the voltage at a remote bus to emulate various Smart Grid devices. This is a useful function for performing volt/var optimization.

To understand how regulator controls work, refer to W. H. Kersting's book on Distribution System Modeling. A simple example of a regulator on a 12.47 kV system:

```
New RegControl.Reg1 Transformer=T1 Winding=2 Vreg=122 band=3
ptratio=60
```

RegControl Object

With a line-drop compensator, the definition might look like

```
New RegControl.Reg1 Transformer=T1 Winding=2 Vreg=122 band=3
~ ptratio=60 CTprim=300 R=2 X=0
```

Controlling a bus at the end of the feeder to 118 V:

```
New RegControl.Reg1 Transformer=T1 Winding=2 Vreg=118 band=2
bus=MyEndBus
```

The properties are:

transformer = Name of Transformer element to which the RegControl is connected. Do not specify the full object name; "Transformer" is assumed for the object class. Example: Transformer=Xfmr1

winding = Number of the winding of the transformer element that the RegControl is monitoring. 1 or 2, typically. Side Effect: Sets TAPWINDING property to the same winding.

vreg = Voltage regulator setting, in VOLTS, for the winding being controlled. Multiplying this value times the ptratio should yield the voltage across the WINDING of the controlled transformer. Default is 120.0

...

RegControl Object

The properties are:

…

band = Bandwidth in VOLTS for the controlled bus (see help for ptratio property). Default is 3.0
delay = Time delay, in seconds, from when the voltage goes out of band to when the tap changing begins. This is used to determine which regulator control will act first. Default is 15. You may specify any floating point number to achieve a model of whatever condition is necessary.
ptratio = Ratio of the PT that converts the controlled winding voltage to the regulator voltage. Default is 60. If the winding is Wye, the line-to-neutral voltage is used. Else, the line-to-line voltage is used.
CTprim = Rating, in Amperes, of the primary CT rating for converting the line amps to control amps.The typical default secondary ampere rating is 0.2 Amps (check with manufacturer specs).
R = R setting on the line drop compensator in the regulator, expressed in VOLTS.

…

RegControl Object

The properties are:

…

X = X setting on the line drop compensator in the regulator, expressed in VOLTS PTphase For multi-phase transformers, the number of the phase being monitored or one of { MAX | MIN} for all phases. Default=1. Must be less than or equal to the number of phases. Ignored for regulated bus.
tapwinding = Winding containing the actual taps, if different than the WINDING property. Defaults to the same winding as specified by the WINDING property.
bus = Name of a bus (busname.nodename) in the system to use as the controlled bus instead of the bus to which the transformer winding is connected or the R and X line drop compensator settings.
debugtrace = {Yes | No* } Default is no. Turn this on to capture the progress of the regulator model for each control iteration. Creates a separate file for each RegControl named "REG_name.CSV".
EventLog = {Yes/True* | No/False} Default is YES for regulator control. Log control actions toEventlog.

…

RegControl Object

The properties are:

…

inversetime = {Yes | No* } Default is no. The time delay is adjusted inversely proportional to the amount the voltage is outside the band down to 10%.
maxtapchange = Maximum allowable tap change per control iteration in STATIC control mode. Default is 16.
Set this to 1 to better approximate actual control action.
Set this to 0 to fix the tap in the current position.
revband = Bandwidth for operating in the reverse direction.
revDelay = Time Delay in seconds (s) for executing the reversing action once the threshold for reversing has been exceeded. Default is 60 s.
reversible = {Yes |No*} Indicates whether or not the regulator can be switched to regulate in the reverse direction. Default is No.Typically applies only to line regulators and not to LTC on a substation transformer.
revNeutral = {Yes | No*} Default is no. Set this to Yes if you want the regulator to go to neutral in the reverse direction.

…

RegControl Object

The properties are:

…

revR = R line drop compensator setting for reverse direction.
revThreshold = kW reverse power threshold for reversing the direction of the regulator. Default is 100.0 kw.
revvreg = Voltage setting in volts for operation in the reverse direction.
revX = X line drop compensator setting for reverse direction.
tapdelay = Delay in sec between tap changes. Default is 2. This is how long it takes between changes after the first change.
TapNum = An integer number indicating the tap position that the controlled transformer winding tap position is currently at, or is being set to.
vlimit = Voltage Limit for bus to which regulated winding is connected (e.g. first customer). Default is 0.0. Set to a value greater then zero to activate this function.

Contents

- General object description
- Line
- Source and other objects
- Power delivery elements
- Power conversion elements
- Control elements
- Meter elements

EnergyMeter Object

An EnergyMeter object is an intelligent meter connected to a terminal of a circuit element. It simulates the behavior of an actual energy meter. However, it has more capability because it can access values at other places in the circuit rather than simply at the location at which it is installed. It measures not only power and energy values at its location, but losses and overload values within a defined region of the circuit.

The operation of the object is simple. It has several registers that accumulate certain values. At the beginning of a study, the registers are cleared (reset) to zero. At the end of each subsequent solution, the meter is instructed to take a sample. Energy values are then integrated using the interval of time that has passed since the previous solution.

EnergyMeter Object
-- Registers

There are two types of registers:

1. Energy Accumulators (for energy values)
2. Maximum power values ("drag hand" registers).

The energy registers may use trapezoidal integration (system option), which allows for somewhat arbitrary time step sizes between solutions with less integration error. This is important for using load duration curves approximated with straight lines, for example.

The present definitions of the registers are, for example for a 22 kV system:

```
Hour
"kWh"
"kvarh"
"MaxkW"
"MaxkVA"
"ZonekWh"
"Zonekvarh"
"ZoneMaxkW"
"ZoneMaxkVA"
"OverloadkWhNormal"
"OverloadkWhEmerg"
...
```

EnergyMeter Object

-- Registers

Registers are frequently added for various purposes. You can view the present meters simply by solving and taking a sample. Then execute a Show Meters command. The Aux registers listed in the example above are used for keep track of losses at up to 7 different voltage levels in the meter zone.

EnergyMeter Object
-- Meter Zones

The EnergyMeter object uses the concept of a zone. This is an area of the circuit for which the meter is responsible. It can compute energies, losses, etc for any power delivery object and Load object in its zone (Generator objects have their own intrinsic meters).

A zone is a collection of circuit elements "downline" from the meter. This concept is nominally applicable to radial circuits, but also has some applicability to meshed circuits. The zones are automatically determined according to the following rules:

Start with the circuit element in which the meter is located. Ignore the terminal on which meter is connected. This terminal is the start of the zone. Begin tracing with the other terminal(s). (Note: These rules imply that the meter should usually be placed at the source end of the circuit element it is monitoring.)

EnergyMeter Object
-- Meter Zones

Trace out the circuit, finding all other circuit elements (loads and power delivery elements) connected to the zone. Continue tracing out every branch of the circuit. Stop tracing a branch when:

- The end of the circuit branch is reached
- A circuit element containing another EnergyMeter object is encountered
- An OPEN terminal is encountered. (all phases in the terminal are open.)
- A disabled device is encountered.
- A circuit element already included in another zone is encountered.
- There are no more circuit elements to consider.

Zones are automatically updated after a change in the circuit unless the ZONELOCK option (Set command) is set to true (Yes). Then zones remain fixed after initial determination.

In order to apply the Reconductor command, both lines must be in the same meter zone. The upline/downline orientation of the line segments is established when the zones are built. Otherwise, the DSS has no concept of radiality.

EnergyMeter Object

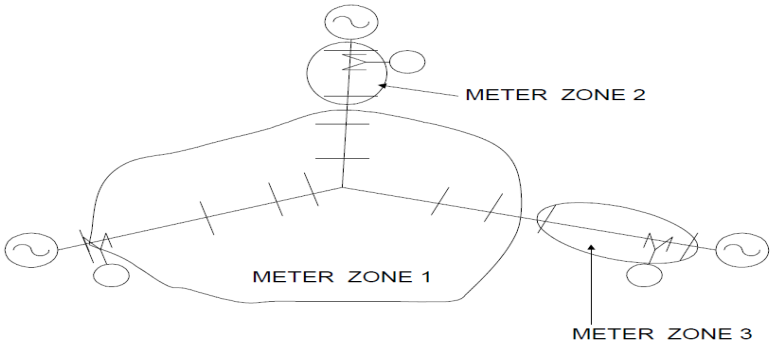
-- Meter Zones

The Parent property available in the Lines interface in the COM interface is set when the zone is established. This allows users to programmatically trace back up the circuit toward the Energymeter. The total number of customers served downline is determined during the establishment of the meter zone.

EnergyMeter Object
-- Zones on Meshed Networks

While the concept of zones nominally applies to radial circuits, judicious placement of energy meters can make the concept useful for meshed networks as well. Keep in mind that there can be many EnergyMeter objects defined in the circuit. Their placement does not necessarily have to represent reality; they are for the reporting of power and energy quantities throughout the system.

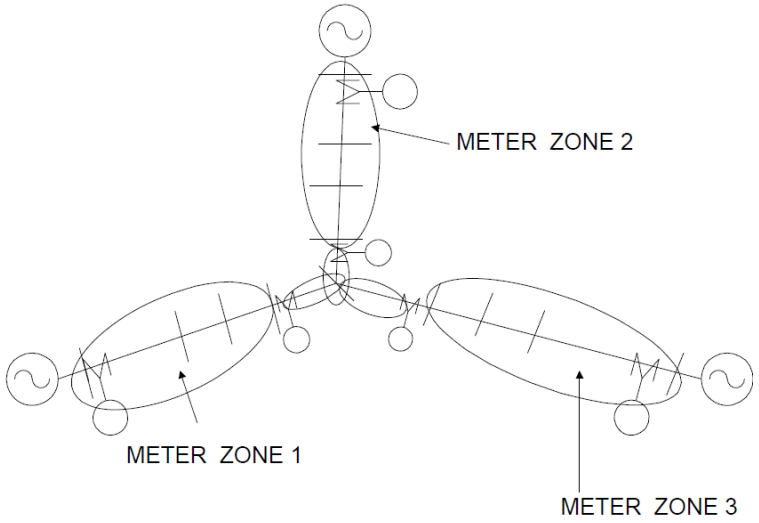
The automatic algorithm for determining zones will determine zones consistently for meshed networks, although the zones themselves may not be radial. If there are several meters on the network that could be monitoring the same zone, the first one defined will have access to all the elements except the ones containing the other meters. The others will have only one element in their zone, as in the Figure below.



EnergyMeter Object

-- Zones on Meshed Networks

Don't put any load at tie point or take care in processing meter information so that it isn't counted more than once. All three of the meters added would see the center-most bus in Figure below.



EnergyMeter Object
-- Sampling

The sampling algorithms are as follows:

Local Energy and Power Values: Simply compute the power into the terminal on which the meter is installed and integrate using the interval between the present solution and the previous solution. This operation uses the voltage and current computed from the present solution.

Losses in Zone: Accumulate the kW losses in each power delivery element in the zone.

Load in Zone: While sampling the losses in each power delivery element, accumulate the power in all loads connected to the downline bus(es) of the element.

Overload Energy in Zone: For each power delivery element in the zone, compute the amount of power exceeding the rating of the element compared to both normal and emergency ratings.

EEN and UE in Zone: For each load in the zone marked as exceeding normal or unserved, compute the present power. Integrate to get energies.

EnergyMeter Object
-- EEN and UE Definitions

EEN refers to load energy considered unserved because the current or voltage exceeds Normal ratings. UE refers to load energy considered unserved because the power (actually the current) exceeds Emergency, or maximum, ratings.

On radial systems (default), a load is marked as unserved with respect to either normal or emergency ratings if either:
The voltage at the load bus is below minimum ratings; the current in any power delivery element supplying the load exceeds the current ratings.

Either the entire load or just the portion above rating at the bus that is considered unserved is counted as unserved, depending on whether the Excess option or the Total option have been specified.

EnergyMeter Object

-- EnergyMeter EEN and UE Registers

Register 9 through 12 on the EnergyMeter are confusing to both new and experienced OpenDSS users. Hopefully, this explanation will help shed some light on the contents of these registers.

EEN = Energy Exceeding Normal: The energy served over a selected period of time above the "Normal" rating of power delivery devices (lines, transformers, switches, etc.). It is assumed that there is sufficient engineering margin in the power system to continue to operate, but that a failure (1st contingency) will require curtailment of load. This is the primary quantity used for measure of risk.

UE = Unserved Energy: The energy over a selected period of time projected to be above the Emergency, or Maximum, rating of power delivery equipment. It is assumed that some load will have to be curtailed to bring the power down to a manageable level.

EnergyMeter Object

-- EnergyMeter EEN and UE Registers

These concepts have evolved since 1994 as we began to look for some means to measure risk in planning, particularly as it applies to distributed generation. We quickly learned that distribution planners were not comfortable with pure probabilistic planning. They prefer methods based on concrete limits. Therefore, we adapted some traditional planning concepts to achieve a measure of risk. Many traditional methods used two limits for power delivery equipment: Normal and Emergency. Planning studies are triggered when the peak demand exceeds the Normal limits. Exceeding the Emergency limit requires immediate curtailment. The usual intent was to get something built before the Emergency limit was projected to be exceeded. By using power demand alone, new construction is frequently very conservative. Distributed generation muddies the waters for planning with demand alone because it is often unclear how much capacity is actually achieved by DG.

Our extension of the concept was to use energy in addition to simply power to determine the risk. One would defer investing in new infrastructure until the energy exceeding the limits was sufficient to justify it.

EnergyMeter Object

-- Registers 9 and 10: Overload EEN and UE

As the Energymeter element sweeps through its zone, it queries each series power delivery element encountered for the kW above Normal and Emergency limits. The value recorded in these two registers is the largest kW amount encountered. In other words, the value is the maximum overload in terms of actual kW.

Most of the time, this is what you want. However, there are cases where data errors can skew the results. For example, if the data show a 15 kVA transformer serving an apartment building with over 150 kVA load, this will consistently show up as a 135 kVA overload, which might be the largest overload in the problem. However, you would not build a new feeder because a small transformer is overloaded. You would change out the transformer (or correct the data error). Such errors are common in distribution data where it may take a while for transformer changeouts to get properly entered.

EnergyMeter Object

-- Registers 9 and 10: Overload EEN and UE

Excess or Total: There are two ways these registers can record the results. They can simply record the excess kW over the limits. This is the default behavior. However, they can also record the total kW flowing through the element. The latter method is for cases where it is assumed that the feeder branch in question must be switched off completely if an overload occurs. This reflects a more conservative planning approach.

Registers 11 and 12: Load EEN and UE

These two registers record a different approach to compute EEN and UE values: They ask each Load element in the zone if it is "unserved." The nominal criterion is undervoltage.

(1) If you set Option=Voltage for an Energymeter object, only the voltage is used. The global options NormVminpu and EmergVminpu are used for this value. (2) If the voltage is between NormVminpu and EmergVminpu, the EEN is proportioned to how far below the NormVminpu it is.

EnergyMeter Object

-- Registers 9 and 10: Overload EEN and UE

(3) If the voltage is less than EmergVminpu, the UE value for the load is computed in proportion to the degee it is below EmergVminpu, continuing on the same slope as the EEN calculation. In multiphase elements, the lowest phase voltage is used.

The default behavior (Option=Combined) is to consider both line overload and undervoltage. If a line, or other power delivery element, serving the load is overloaded, the load is considered unserved in the same percentage that the line is overloaded. This actually takes precedence over the voltage criteria, which assumes that any undervoltage is due to the line overload. A line is considered to serve the load if it is between the EnergyMeter and the load. Note that some inaccuracies can occur if the meter zone is not properly defined, such as if loops exist.

EnergyMeter Object
-- Properties

The properties are:

Element = Name of an existing circuit element to which the monitor is to be connected. Note that there may be more than one circuit element with the same name (not wise, but it is allowed). The monitor will be placed at the first one found in the list.

Terminal = No. of the terminal to which the monitor will be connected, normally the source end.

Action = Optional action to execute.

Clear = reset all registers to zero

Save = Saves (appends) the present register values to a file. File name is MTR_metername.CSV, where metername is the name of the energy meter.

Take = Takes a sample at the present solution.

Option = Options: Enter a string ARRAY of any combination of the following. Options processed left-to-right:

(E)xcess : (default) UE/EEN is estimate of only energy exceeding capacity

(T)otal : UE/EEN is total energy after capacity exceeded.

(R)adial : (default) Treats zone as a radial circuit

…

EnergyMeter Object
-- Properties

The properties are:

…

(M)esh : Treats zone as meshed network (not radial).

(C)ombined: (default) Load UE or EEN are computed from both overload and undervoltage criteria.

(V)oltage: Load UE/EEN are computed only from the undervoltage criteria.

Example: option=(E, R, C)

In a meshed network, the overload registers represent the total of the power delivery element overloads and the load UE/EEN registers will contain only those loads that are "unserved", which are those with low voltages. In a radial circuit, the overload registers record the max overload (absolute magnitude, not percent) in the zone. Loads become unserved either with low voltage or if any line in their path to the source is overloaded.

KWNorm = Upper limit on kW load in the zone, Normal configuration. Default is 0.0 (ignored). If specified, overrides limits on individual lines for overload EEN. KW above this limit for the entire zone is considered EEN.

…

EnergyMeter Object
-- Properties

The properties are:

…

KWEmerg = Upper limit on kW load in the zone, Emergency configuration. Default is 0.0 (ignored). If specified, overrides limits on individual lines for overload UE. KW above this limit for the entire zone is considered UE.

Peakcurrent = ARRAY of current magnitudes representing the peak currents measured at this location for the load allocation function (for loads defined with xfkva=). Default is (400, 400, 400). Enter one current for each phase.

Zonelist = ARRAY of full element names for this meter's zone. Default is for meter to find it's own zone. If specified, DSS uses this list instead. It can access the names in a singlecolumn text file. Examples:

```
Zonelist =[line.L1, transformer.T1, Line.L3]
Zonelist =(file=branchlist.txt)
```

LocalOnly = {Yes | No} Default is NO. If Yes, meter considers only the monitored element for EEN and UE calcs. Uses whole zone for losses.

…

EnergyMeter Object
-- Properties

The properties are:

…

Mask = Mask for adding registers whenever all meters are totalized. Array of floating point numbers representing the multiplier to be used for summing each register from this meter. Default = (1, 1, 1, 1, ...). You only have to enter as many as are changed (positional). Useful when two meters monitor same energy, etc.

Losses = {Yes | No} Default is YES. Compute Zone losses. If NO, then no losses at all are computed.

LineLosses = {Yes | No} Default is YES. Compute Line losses. If NO, then none of the line losses are computed.

XfmrLosses = {Yes | No} Default is YES. Compute Transformer losses. If NO, transformers are ignored in loss calculations.

SeqLosses = {Yes | No} Default is YES. Compute Sequence losses in lines and segregate by line mode losses and zero mode losses.

3PhaseLosses = {Yes | No} Default is YES. Compute Line losses and segregate by 3-phase and other (1- and 2-phase) line losses.

…

EnergyMeter Object
-- Properties

The properties are:

…

VbaseLosses = {Yes | No} Default is YES. Compute losses and segregate by voltage base. If NO, then voltage-based tabulation is not reported. Make sure the voltage bases of the buses are assigned BEFORE defining the EnergyMeter to ensure that it will automatically pick up the voltage bases. Or, issue the CalcVoltageBases command after defining the EnergyMeter. Each voltage base has four(4) loss registers: total, line, load, and no-load, respectively. There are sufficient registers to report losses in five (5) different voltage levels.

BaseFreq = Base frequency for ratings.

Enabled = {Yes|No or True|False} Indicates whether the element is enabled.

Like= Name of another EnergyMeter object on which to base this one.

Monitor Object

-- Properties

A monitor is a benign circuit element that is connected to a terminal of another circuit element. It takes a sample when instructed, recording the time and the complex values of voltage and current, or power, at all phases. Other quantities may be saved depending on the setting of the Mode property. The data are saved in a file stream (a separate one for each monitor) at the conclusion of each step of a multistep solution (e.g., daily or yearly, or harmonics) or each solution in a Monte Carlo calculation. In essence, it works like a real power monitor. The data in the file may be converted to CSV form and, for example, brought into Excel. You may accomplish this by either the Show Monitor command or the Export Monitor command.

For Monte Carlo runs, the hour is set to the number of the solution and seconds is set to zero. For Harmonic solutions, the first two fields are changed to Frequency and Harmonic.

Monitors may be connected to both power delivery elements and power conversion elements.

Monitor Object

-- Properties

The parameters are:

Element = Name of an existing circuit element to which the monitor is to be connected. Note that there may be more than one circuit element with the same name (not wise, but it is allowed). The monitor will be placed at the first one found in the list.

Terminal = No. of the terminal to which the monitor will be connected.

Mode = Integer bitmask code to describe what it is that the monitor will save. Monitors can save two basic types of quantities: 1) Voltage and current; 2) Power. The Mode codes are defined as follows:

0: Standard mode - V and I,each phase, complex
1: Power each phase, complex (kw and kvars)
2: Transformer taps (connect Monitor to a transformer winding)
3: State variables (connect Montor to a PCElement)
+16: Sequence components: V012, I012
+32: Magnitude only
+64: Pos Seq only or Average of phases, if not 3 phases

For example, Mode=33 will save the magnitude of the power (kVA) only in each phase. Mode=112 saves Positive sequence voltages and currents, magnitudes only.

Monitor Object

-- Properties

The parameters are:

```
...
Action = {clear | save} parsing of this property forces clearing
of the monitor's buffer, or saving to disk.
```

Thank you!