ELECTROTEK
CONCEPTS

408 North Cedar Bluff Road
Suite 500
Knoxville, TN 37923-3641
Tel: 865-470-9222
FAX: 865-470-9223
www.electrotek.com

Date: Feb 7, 2003

DSS Fault Current Calculation Procedures

Roger C. Dugan
Sr. Consultant

This document describes the methods used for fault calculations and the effects and impacts of using the various DSS commands and COM interfaces methods and properties.

Introduction

The DSS builds a general nodal admittance (Y) matrix to describe the system. Fault current calculations are performed in two basic ways:

1. Simulation of a specific fault determined by the placement and definition of one or more Fault objects on the feeder.
2. A general fault study solution mode.

More detail follows:

Fault Simulations

A fault simulation is often a dynamics simulation, but doesn't have to be. A Fault object may be applied in conventional SNAPSHOT power flow mode, but the solution will be slightly different than in other modes. When the voltage falls below the nominal range, power consuming and power producing devices switch to constant impedance models. Convergence can sometimes be a problem, but is generally not.

In dynamics mode, the generators are converted to a dynamic model after a conventional power flow solution. Then the fault(s) are applied and the solution proceeds. Multiple faults can be applied simultaneously.

A Fault object is simply a multi-phase resistor branch (two-terminal) in which the second terminal defaults to being connected to ground. The dynamic mode is used for simulation of overcurrent protection devices.

Fault Study Mode

The fault study mode works differently:

1. The system Y matrix is built including all loads as admittances.
2. All generators are converted to their dynamic model (or Thevinen equivalent).
3. All Fault objects are disabled (removed from the circuit).
4. A direct solution of the Y matrix equations is performed including source injections and generator injections. The resulting open-circuit voltage is stored for future use.
5. The Thevinen short circuit impedance matrix at each bus is computed. The inverse is also computed. Both become part of the Bus object.
6. The short circuit currents for a variety of fault conditions are computed for each bus using the Thevinen model. Only the "all-phase" short circuit current is computed at the time the Solve command is issued. The remainder are computed when the Show Faults command is issued.

Using the Fault Object

(This material is basically copied from the Users Manual).

A Fault is a constant-impedance power delivery circuit element almost like any other. The user simply applies the fault and solves. The result is the best approximation to the actual faulted condition that can be achieved for this fault given the load models, which may or may not be accurate for voltages outside the normal band. There are some conditions where faults are treated specially. For example, during the Fault Study mode (Mode=FAULTSTUDY), all fault objects are disabled. The Monte Carlo fault mode also alters fault objects.

The Fault model is basically implemented very similarly to the Capacitor or Reactor as a two-terminal power delivery element. The fault is nominally an uncoupled, multiphase resistance branch. You may also specify the fault as a nodal conductance (G) matrix, which will give you complete control over the definition for any complex fault situation, including coupling between the phases. You can use the Fault object to represent a two terminal resistance if you like, keeping in mind that some solution modes (e.g., FAULTSTUDY) may disable the element so that it is no longer in the circuit.

In Monte Carlo Fault mode, the fault resistance is varied by the % standard deviation specified. If the standard deviation is entered as zero (default), the resistance is varied uniformly.

Like the Capacitor, 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 resistance.

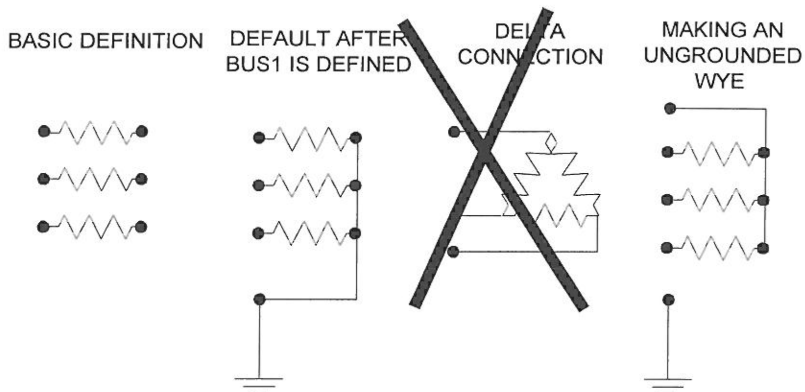


Figure 1. Definition Of Fault Object.

Unlike the capacitor, you cannot specify a delta connection. If you need one, simply make one using the terminal connections. You may also use the Gmatrix command to enter an arbitrary conductance matrix representing any connection you wish.

If you wish a series uncoupled resistor, simply specify an appropriate second bus connection.

If you wish an ungrounded wye resistor, 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 fault

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

Parameters, in order, are:

Bus1= Definition for the connection of the first bus. When this is set, Bus2 is set to the same bus name, except with all terminals connected to node 0 (ground reference). Set Bus 2 at some time later if you wish a different connection.

Bus2= Bus connection for second terminal. Defaults to grounded-wye connection at bus 1. Must be specified to achieve any other connection.

Phases= Number of phases. Default is 1 (SLG fault).

r= Value of resistance in ohms. Using this property sets the resistance in all phases to the same value. To set different values, use Gmatrix. Default is 0.0001 ohms, which is a good approximation of a bolted fault.

%stddev= The percent standard deviation assumed for the fault resistance when using Monte Carlo Fault mode. If set to zero, the resistance is varied uniformly from 0 to the value of **r**. Otherwise, the value given for **r** is assumed to be the mean value.

Gmatrix= Alternate method of defining the fault resistance. Enter nodal admittance matrix in seimens (mhos). Form of the matrix should be (lower triangle):

$$\text{Gmatrix}=(g_{11} \mid -g_{21}\ \ g_{22} \mid -g_{31}\ \ -g_{32}\ \ g_{33})$$

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

Examples of Scripts for Using Fault Objects

A simple script:

```
Compile circuitfile.txt    ! define the circuit
Solve
New fault.f1 bus1=xxx       ! defines a SLG fault
Solve
Show currents      ! see currents in the fault and system
Show voltages       ! voltages throughout the system
```

This script applies a SLG fault at bus "xxx" on phase 1, since all the defaults were taken. The solution mode is generally SNAPSHOT at this point, so this is a quasi-power flow solution. The load and generator models will generally attempt to consume or produce the specified power unless the voltage drops too low. Then they revert to constant impedance models.

An altenative for doing this solution is:

```
Compile circuitfile.txt     !define the circuit
Solve
New fault.f1 bus1=xxx       ! defines a SLG fault
Solve  mode=dynamic  number=1
Show currents      ! see currents in the fault and system
Show voltages       ! voltages throughout the system
```

This script also performs a solution for the specified fault, but switches to dynamic mode first. This forces the generator models to switch to voltage-behind-reactance models like those used in dynamic simulations. They no longer attempt to force a particular power down the line, but simply act as voltage sources. This gives a better answer in most cases, although the previous may be good enough. By doing only 1 solution (number=1), only the initial fault current is computed. This is generally the highest.

If the generators employ a sophisticated dynamics model that changes with time, more solution steps can be executed to see the change in fault current with time.

Fault Study Mode

The general procedure for the fault study mode (mode=FAULTSTUDY) is described in the Introduction. The mathematics that supports this are described here.

The DSS builds a nodal admittance (Y matrix) model of the system. This can be represented schematically as shown in Figure 1. Voltage sources are converted to Norton equivalents and the resulting admittance is incorporated into the system Y matrix as being connected to ground. This includes the Vsource objectss and any Generator objects after they has been converted to a dynamic equivalent.

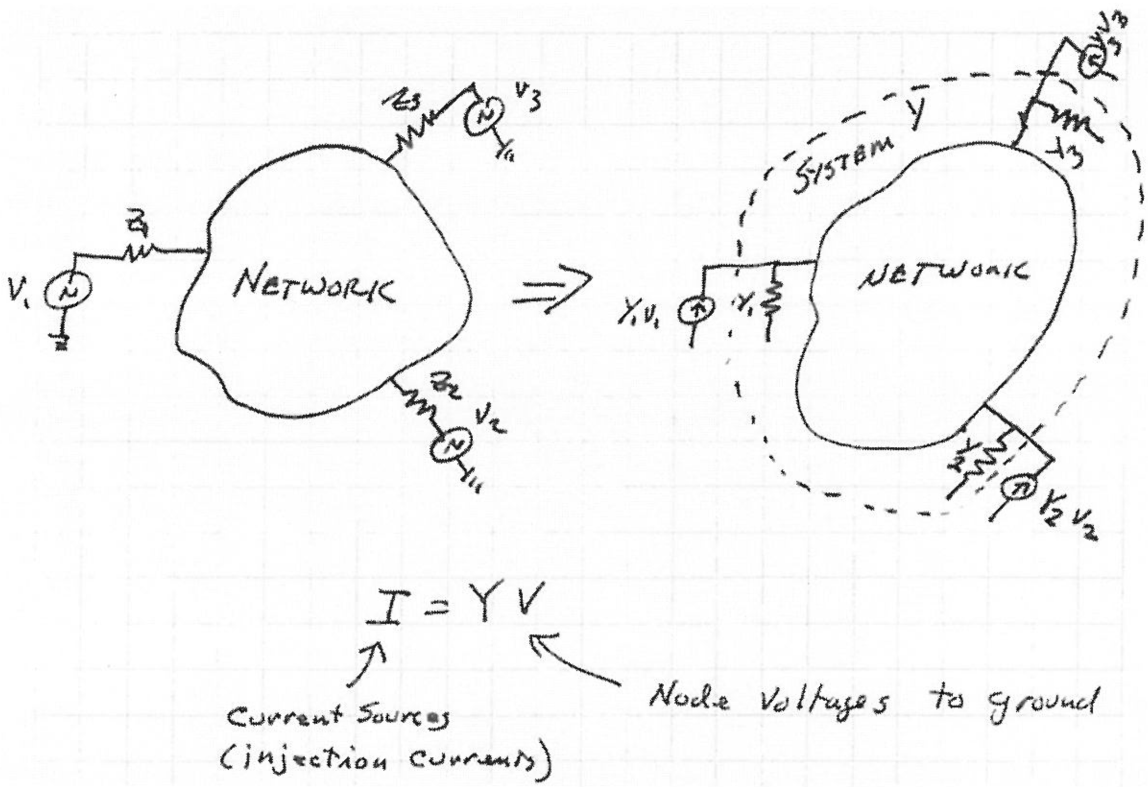


Figure 1.

As indicated, the DSS then solves the equation I=YV for the voltages to ground. The "I" vector is the vector of current injections into the network. In the usual case, this includes the source currents shown as well as the portion of the load currents not included in the Y matrix already.

The fault study is based on a multiphase Thevinen equivalent at each bus (Figure 2). The first step is to compute the open circuit voltage vector, Voc, at each bus. This is done by performing a direct solution immediately after entering the FAULTSTUDY mode. Keep in mind the solution was previously solved in a standard SNAPSHOT mode, which is not necessarily direct. That serves to initialize the generator models (and anything else that might need it). Then the short-circuit impedance matrix, Zsc, is computed for each bus and inverted to form Ysc. Both forms are retained by the program internally for each bus because there are several useful things they can be used for. The Norton form of the equations is also retained by computing the short circuit current vector, Isc, that corresponds to the open circuit voltages.

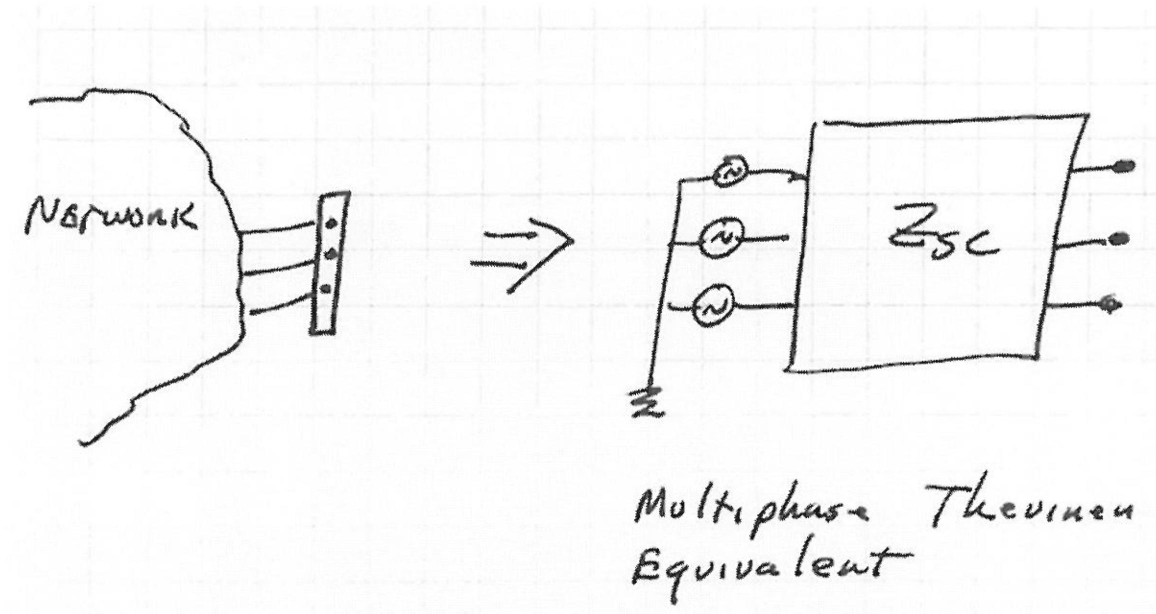


Figure 2.

Since, the system Y matrix already assumes that the voltage sources are shorted, Zsc is determined by injecting a current of 1+j0 amps into each node of the bus under consideration one node at a time. The resulting voltages represent one column of Zsc (See Figure 3).

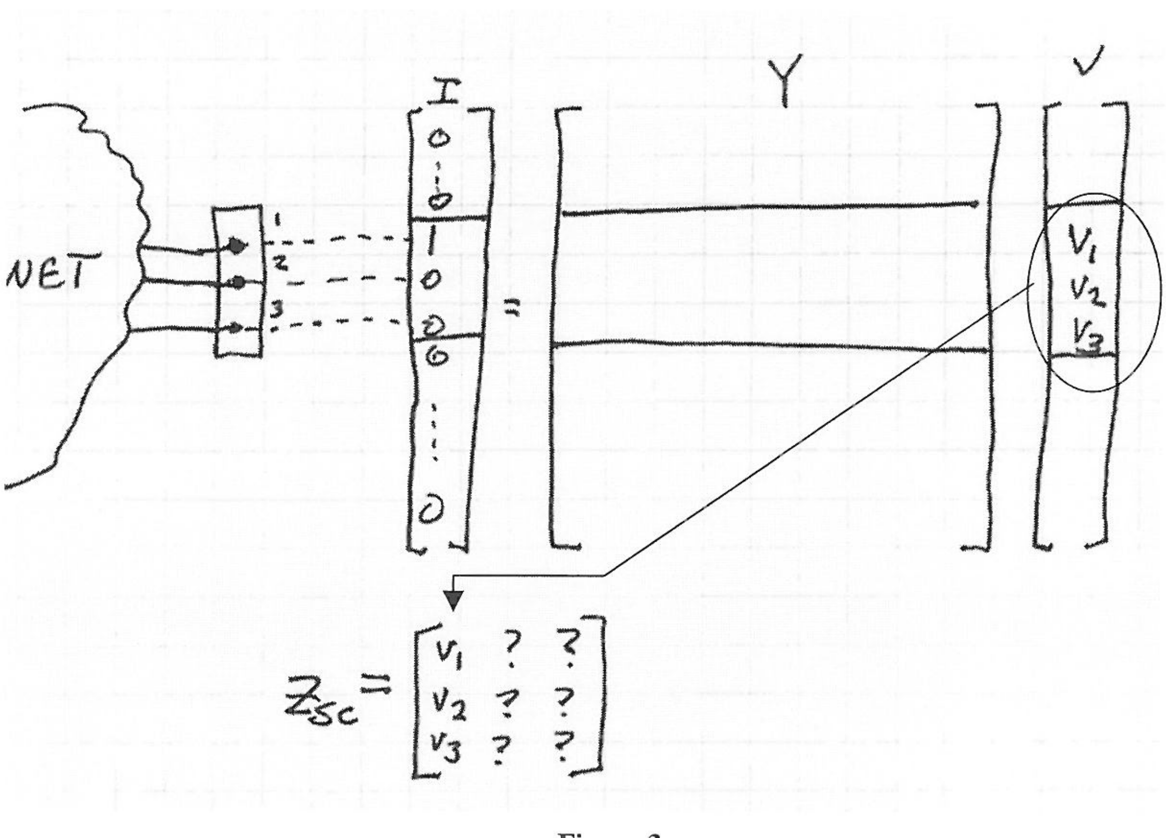


Figure 3

This process is repeated for each node at the bus, including neutral nodes or other non-phase nodes. Now, Zsc is fully populated. It is then inverted to form Ysc. Note, there are circuit conditions where Zsc might be a little bit ill-conditioned. In order to inject 1+j0 into a node, there must be a return path. It is possible to set up a condition where this path does not exist, although the DSS takes steps to minimize this occurrence. Transformers are the main culprit, but a user could conceivably create a coupled inductance matrix with a line or reactor and leave both ends hanging. The DSS adds a tiny conductance to each winding to prevent this, but can't catch everything This is just a fact of life when doing nodal admittance models and is also encountered in EMTP, EMTDC, and SuperHarm. (The DSS is more user friendly than any of these in this regard.)

$$Z_{SC} = \begin{bmatrix} Z_{11} & Z_{12} & Z_{13} \\ Z_{21} & Z_{22} & Z_{23} \\ Z_{31} & Z_{32} & Z_{33} \end{bmatrix} \qquad Y_{SC} = Z_{SC}^{-1}$$

(SYMMETRICAL)

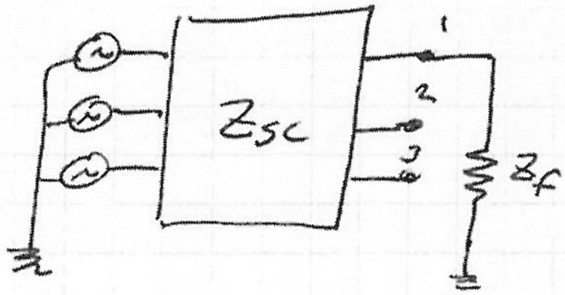
Figure 4.

The Norton currents (short-circuit currents) are now computed as shown in Figure 5. This is the "all-phase" fault current that the DSS reports. That is, it is the short circuit currents that will flow from each node if all nodes at the bus were shorted to Node 0. Note that these currents will not be equal if the impedances are not balanced (or the open circuit voltages not balanced).

$$I_{SC} = Y_{SC} V_{OC}$$

Figure 5.

The quantities reported for single-phase faults (each node-to-ground, one at a time) are computed as shown in Figure 6 for a typical three-phase case. The current is computed from Zsc in a straight forward manner because there is no current flowing from the other nodes at the bus. The unfaulted phase voltages are also of interest and are computed as shown from Voc and the computed fault current. If the base voltage is defined for the bus, the DSS reports the voltages in per unit. Otherwise, they are in L-N actual volts.



$$I_1 = I_f = \frac{V_{oc_1}}{Z_{11} + Z_f}$$

$$I_2 = 0$$

$$I_3 = 0$$

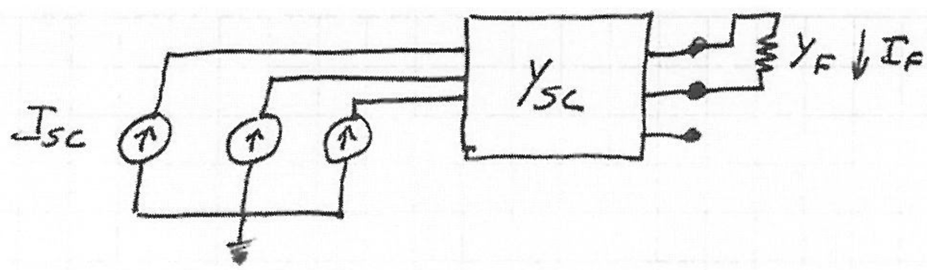
$$V_1 = V_{oc_1} - Z_{11} I_1$$

$$V_2 = V_{oc2} - Z_{21} I_1$$

$$V_3 = V_{oc3} - Z_{31} I_1$$

Figure 5.

The phase-to-phase faults are a little tricky in one sense, but simpler in another. The Norton equivalent at the bus is employed rather than the Thevinen. Figure 6 shows the equations.



$$I_{SC} = \left[Y_{SC} + Y_F \right] V$$

$$\begin{bmatrix} I_{SC1} \\ I_{SC2} \\ I_{SC3} \end{bmatrix} = \left[\begin{bmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ Y_{31} & Y_{32} & Y_{33} \end{bmatrix} + \begin{bmatrix} Y_F & -Y_F & 0 \\ -Y_F & Y_F & 0 \\ 0 & 0 & 0 \end{bmatrix} \right] \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}$$

Figure 6.

408 N. Cedar Bluff Road, Suite 500 • Knoxville, Tennessee 37923 • Tel: (865) 470-9222 • FAX: (865) 470-9223

A fault admittance, Y_F, is connected to the two phases of interest. It is added to the Ysc matrix in the appropriate terms as shown. (Add Y_F to the two diagonals involved and $-Y_F$ to the off-diagonals between the two nodes.) Very simple to code. Then, given Isc, the voltages at the nodes are computed by solving the linear equations. Thus, there is no need for further calculations to determine the voltages appearing at the nodes. The fault current in this specifice case for a fault from node 1 to node 2 is then computed as:

$$I_F = Y_F (V_1 - V_2)$$

The DSS reports only adjacent phase faults (1-2, 2-3, 3-4, etc.). It does not report 1-3, for example. That's a quirk that could eventually be changed if necessary. Of course, any program using the DSS COM interface can extract Zsc and do whatever it pleases.

Monte Carlo Fault Study

This applies to solution mode **MF** (**M**onte Carlo **F**ault mode). This mode is designed to allow studies of behaviors due to random fault events.

To start, the user defines one or more Fault objects in the circuit. Normally for this mode, there will be several fault objects defined. For example, one study might be to determine the response of DG relaying to typical faults on the utility system. In a simple study, one might define all possible types of faults at a single bus (SLG, LL, LLG, 3-Phase). If the system impedance is modeled as unbalanced, it will be necessary to model all 3 possible SLG faults (not necessary of lines are modeled as Z1, Z0).

The Monte Carlo fault mode selects one fault at a time and solves the circuit. One of the faults defined in the active circuit is selected and its resistance value randomized. All other Faults are disabled. A **Direct** solution (non-iterative) is performed, in which the loads are treated as constant impedances. Results are captured by using monitors.

Monitors are reset and generators converted to dynamic models upon changing the mode to MF.

Executes the number of cases specified by the Number option of the Solve or Set command, for example:

```
Compile circuitfile.txt
Solve                          !establish a base solution without the faults
Redirect faultdefinitions.txt    !define faults
Solve mode=MF number=1000
Show mon monitor1                ! displays results
```

This script will solve for 1000 different faults and display the results. The monitor results can be piped back through the COM interface as well (if using the DLL form.)

Internal Algorithm

1. (perform a legitimate solution without the faults)
2. (changing the mode to MF will force generators to switch to voltage behind reactance and reset the monitors)
3. Load model is set to ADMITTANCE
4. Global load multiplier is set to 1.0
5. time is set to zero (hour is used as the case counter)
6. For the number of times specified:
    a. Increment Hour count for monitor sample
    b. Randomly pick a fault
    c. Randomly set the fault resistance
    d. Do a direct solution of I=YV with the selected fault
        i. Rebuild Y matrix
        ii. Get all source injections
        iii. Get all machine injections
        iv. Solve sparse set
    e. Tell the monitors to take a sample
7. Save monitors

Text Interface Commands for Short Circuit Analysis

The first step is to compute the Zsc matrices at each bus. This is accomplished by the following commands:

```
Solve Mode=FaultStudy          (or, mode=F)
```

Or

```
Set Mode=FaultStudy
Solve
```

After this step, the results of the fault study can be viewed by

```
Show Faultstudy          (report)
```

Or

```
Export Faultstudy          (CSV file)
```

The short circuit impedance matrix can be retrieved through the "Result" text string by selecting a particular bus and requesting the appropriate value. One way to select the active bus is:

`Set Bus = xxxx`
(also the Select command and theSet Terminal= option will set the active bus).

Once the bus is selected:

`ZSC` returns the full Zsc matrix in complex R + jX form, comma separated. Values in ohms.

`YSC` returns the full Ysc matrix in complex G + jB form, comma separated. Values in ohms.

`ZSC10` returns both the positive and zero-sequence impedances in ohms.

`ZscRefresh` recomputes the Zsc for the active bus for the present circuit condition.

Keep in mind that Zsc is not normally computed until the Solve command is issued while in FAULTSTUDY mode. If the circuit changes, Zsc will have to be recomputed.

408 N. Cedar Bluff Road, Suite 500 • Knoxville, Tennessee 37923 • Tel: (865) 470-9222 • FAX: (865) 470-9223

COM Interface Commands Related to Short Circuit Analysis

Programs that access the DSS through the COM interface can do pretty much the same as through the text scripts, with perhaps a little more flexibility and less interpretation of text strings.

Use the **SetActiveBus** or **SetActiveBusi** method under the Circuit interface to select the active bus. SetActiveBus uses the name of the bus and returns the index. SetActiveBusi uses the bus index and is faster because it doesn't have to look up the name of the bus. Also, any time you set the active terminal, the associated bus becomes the active bus. You may likewise use the **Buses** collections property to return an interface to the active bus.

Once the bus is selected, the following Bus interface properties give access to the short circuit values:

NumNodes	The number of nodes at the active bus. This gives the order for all the variant arrays and matrices, although it is a good idea to always check the bounds returned because an error might have occurred.
Voc	Returns a variant array of doubles representing the complex values of the open-circuit voltage vector Voc computed as described above.
Isc	Returns a variant array of doubles representing the complex values of the short-circuit current vector Isc (for the Norton equivalent) computed as described above.
ZscMatrix	Returns a variant array of complex number representing the Zsc matrix, column by column.
YscMatrix	Returns a variant array of complex number representing the Ysc matrix, column by column.
Zsc1	Returns a variant array consisting of 2 doubles representing the positive-sequence short circuit impedance at the bus. Determined by averaging the diagonals and the off-diagonals of the Zsc matrix.
Zsc0	Returns a variant array consisting of 2 doubles representing the zero-sequence short circuit impedance at the bus. Determined by averaging the diagonals and the off-diagonals of the Zsc matrix.
ZscRefresh	Recomputes Zsc for the active bus only, representing the present circuit condition.
KVBase	Returns a double representing the defined kv voltage base for the active bus. Use this if you want to convert values to per unit.

Examples with VB

```
Public Sub ShowZscMatrix()
'  Get the Zsc Matrix for a bus and put it in a textbox for viewing.
    Dim Name As String
    Dim Z As Variant
    Dim i As Long, j As Long
    Dim S As String
    Dim n As Long

    Name = Form1.Combo2.Text                                ' Get the name of the bus
    On Error GoTo DSSLikelyDead
    S = ""
    If DSSCircuit.SetActiveBus(Name) > 0 Then                     'Set the bus Active
        Set DSSBus = DSSCircuit.ActiveBus                   ' Establish Bus interface
        With DSSBus
            Z = .ZscMatrix                            ' This transfers the Zsc matrix
            n = .NumNodes                           '  This tells us the order of Zsc
            i = LBound(Z)
            If UBound(Z) = 0 Then                           ' If this is 0, Zsc is nil
                S = S + "Zsc Not computed"
            Else
'  This loop builds a string, S, to display
                Do                          ' The matrix is complex pairs (R + j X)
                    For j = 1 To n
                        S = S + Str(Z(i)) + " +j " + Str(Z(i + 1)) + ", "
                        i = i + 2
                    Next j
                    S = S + vbCrLf                        ' Stick a CRLF after each row
                Loop Until i > UBound(Z)           'Loop until we run out of numbers
            End If
        End With
        TextForm.Text1.Text = S                       ' Put the String in a Text box
        TextForm.Show vbModal
    Else
        MsgBox "Bus: '" + Name + "' Not Found."
    End If
    Exit Sub
DSSLikelyDead:
    MsgBox "OOPS!"
End Sub
```

```
Public Sub ShowZsc10()
' This routine shows the positive and zero sequence impedances at a bus
' It also computes the MVA SC due to the positive-sequence impedance
' based on the base voltage at the bus

    Dim Name As String
    Dim Z As Variant
    Dim i As Long, j As Long
    Dim S As String
    Dim kVBase As Double
    Dim cZ As Complex
    Dim MVA As Double

    Name = Form1.Combo2.Text
    On Error GoTo DSSLikelyDead
    S = ""                                                       ' Init our string
    If DSSCircuit.SetActiveBus(Name) > 0 Then                   ' Set the bus active
        Set DSSBus = DSSCircuit.ActiveBus    ' Establish connection to Bus interface
        With DSSBus                            ' DSSBus is public of type DSSLIB.BUS
            Z = .Zsc1                    ' This gets Z1, which is a two-element array
            i = LBound(Z)            ' Always use Lbound and Ubound on variant arrays
            S = S + "Z1= " + Str(Z(i)) + " +j " + Str(Z(i + 1)) + vbCrLf
'  Now stick the variants into a local complex variable so we can us complex math
'  functions
            cZ.Re = Z(i)
            cZ.im = Z(i + 1)
            kVBase = .kVBase                            ' Get the base kV for the bus
            MVA = kVBase ^ 2 / CabsFunc(cZ)                         ' Compute the MVA

            S = S + "MVA SC 1= " + Str(MVA) + vbCrLf + vbCrLf    ' Add to our string
            Z = .Zsc0                                                        ' Get Z0
            i = LBound(Z)
            S = S + "Z0= " + Str(Z(i)) + " +j " + Str(Z(i + 1)) + vbCrLf
        End With
        TextForm.Text1.Text = S                              ' Copy S to the Text box
        TextForm.Show vbModal
    Else
        MsgBox "Bus: '" + Name + "' Not Found."
    End If
    Exit Sub
DSSLikelyDead:
    MsgBox "OOPS!"
End Sub
```

```
Public Sub RefreshZsc()
' this routine simply refreshes Zsc at the bus for the present
' circuit condition

    Dim Name As String

    Name = Form1.Combo2.Text                              ' Get the name of the bus

    On Error GoTo DSSLikelyDead         ' Always a good idea to have an error handler

    If DSSCircuit.SetActiveBus(Name) > 0 Then
        Set DSSBus = DSSCircuit.ActiveBus                    ' Establish interface
        With DSSBus
            If .ZscRefresh Then                               ' Returns a boolean
                {OK code}
            Else
               {Not OK code}
            End If
        End With
    Else
        MsgBox "Bus: '" + Name + "' Not Found."
    End If
    Exit Sub
DSSLikelyDead:
    MsgBox "OOPS!"
End Sub
```


408 N. Cedar Bluff Road, Suite 500 • Knoxville, Tennessee 37923 • Tel: (865) 470-9222 • FAX: (865) 470-9223