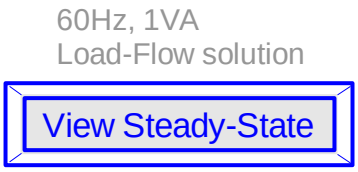
View Steady-State



Jean Mahseredjian, 4/21/2020 1:11:00 AM

1 Introduction

The steady-state and load-flow solution visualization options are available through the View Steady-State device or menu in Design ribbon.

2 Main Data tab

This tab is for selecting signal and device steady-state (or load-flow) visualization options.

After making the appropriate selections, the user must run the visualization command (menu) “Design>Steady-state view>Show data” or “Save results into file”. The ‘View Steady-State’ (VSS) option-device is only for making option selections executed by the above commands for the presentation of phasors from load-flow or steady-state solutions.

Only one VSS device can be present in a design and at the top-circuit level (not in a subcircuit).

The available options are self-explanatory and extra notes are given in tooltips.

This option-device (VSS) can be available on the top circuit level. When the visualization command is executed, it will search for this option-device at the top circuit level and perform the requested selections.

If a signal or a power device is located in a subcricuit (subnetwork) and if its subcircuit is not unique, then to retrieve the correct data for each subcircuit instance, the user must run the ‘Show data’ command at the subcircuit level.

2.1 Parameters/Settings

1. **Solution frequency**: this option specifies the steady-state solution frequency. It is only applicable to the Steady-State solution.
2. **Load-flow solution or Steady-State solution**: The presented phasors will be based on the load-flow or steady-state solution.

3. **Refresh automatically**: When this option is turned on, the presented phasors are automatically refreshed upon the completion of an EMTP load-flow or steady-state solution (simulation).
4. **Show device powers using device selections**: Power devices are given a right-click menu for selecting power visualization options. When this option “Show device powers in steady-state” is turned on (checked), the device powers are automatically refreshed. Existing or new selections will not be refreshed if this option is off.
5. **Power base**: used for the presentation of device powers.
6. **Show signal voltages using signal selections**: Power signals are given a right-click menu for selecting voltage visualization options. When this option is checked, the signal voltage phasors are automatically refreshed. Existing or new selections will not be refreshed if this option is off.
7. **Use RMS values for current phasors**: All shown current phasors will use RMS values.
8. **Use kA for current phasors**: All shown current modules will us the kA unit.
9. **Save results**: see below.

2.2 Save results

When this option is checked, the selected phasors on power signals will be sent to the console (“Write to console” option) or saved into text file or Excel file.
This save operation can be started from the Ribbon Design. There are two options in the Steady-State View section of this Ribbon:

1. Save Voltage results into file
2. Save Current results into file

When a signal with phasor visualization options is located in a subnetwork, then its phasors are saved for all instances of such non-unique subnetwork. The same is applicable to device currents.

3 Options

This tab presents several visualization options. The options are self-explanatory.

4 Scripting

In addition to scripting options for the View Steady-State device as described in the script file load_flow_view.dwj found in Info Scripts\files under the EMTPWorks program folder, it is possible for users to program load-flow or steady-state data extraction using specific methods as documented below.

The available Request object (named RequestObject, for example) fields are:

Signal	Searched signal object.
SignalName	Signal name. This is an option as documented below.
FoundSignalFullName	The actual signal name with path name, as found (returned) by the scripts.
SignalFound	Returns true when the signal data is found.
Signalis3phase	Returns true when the searched signal is 3-phase.
get_pu_from_signal_attribute	The default value is false, use true to retrieve per-unit base from ‘Signal Parameters’ right-click menu. When this parameters is false, the default units are ‘pu’ and the per-unit base is 1.
UseXMLfileUnits	When true, uses the units of XML file for steayd0state or load-low solution. The default units are Volts (V). This parameter overrides get_pu_from_signal_attribute.
Units	Found (returned) units. When this value is ‘pu’ and get_pu_from_signal_attribute is false, then the per-unit base is 1 for 1 kVRMSLL.
puBase	Found (returned) pert-unit base.
LForSS	0 means that searched data is from load-flow solution (default) 1 means the searched data is from steady-state solution

	This field can be set only from the first cell of the Request object array, example: RequestObject[0].LForSS=0
Frequency	Specifies the required frequency from steady-state solution. The default value is taken from View Steady-State Options. This field can be set only from the first cell of the Request object array, example: RequestObject[0].LForSS=50
MessageConsole	Sends informative messages and error messages to the View Steady-State console, when set to true, default is false. This field can be set only from the first cell of the Request object array, example: RequestObject[0].MessageConsole=false
Va_mag	Phase-A voltage magnitude.
Va_phase	Phase-A voltage phase.
Vb_mag	Phase-B voltage magnitude.
Vb_phase	Phase-B voltage phase.
Vc_mag	Phase-C voltage magnitude.
Vc_phase	Phase-C voltage phase.
V1_mag	Positive sequence voltage magnitude, 3-phase signal.
V1_phase	Positive sequence voltage phase, 3-phase signal.
V2_mag	Negative sequence voltage magnitude, 3-phase signal.
V2_phase	Negative sequence voltage phase, 3-phase signal.
V0_mag	Zero sequence voltage magnitude, 3-phase signal.
V0_phase	Zero sequence voltage phase, 3-phase signal.
Device	Searched device object.
DeviceName	Device name. This is an option as documented below.
FoundDeviceFullName	The actual device name with path name, as found (returned) by the scripts.
DeviceFound	Returns true when the device data is found.
PowerCurrentDataFields	An object related to device data for powers and currents. The object fields are listed below. This is applicable when searching for a Device.
.NotApplicable	Becomes true when a selected device does not participate (no data) in load-flow or steady-state solution.
.Ptotalk	Total active power entering the k-side off device.
.Ptotalm	Total active power entering the m-side off device.
.Qtotalk	Total reactive power entering the k-side off device.
.Qtotalm	Total reactive power entering the m-side off device.
.mSideExists	True when data from m-side is available. Some devices provide only k-side data.
.Ikside_module	Array of current modules entering k-side.
.Ikside_phase	Array of current phases entering k-side.
.Imside_module	Array of current modules entering m-side.
.Imside_phase	Array of current phases entering m-side.
.IkSequences	Array of sequence currents entering k-side: use IkSequences[iseq].magnitude() and IkSequences[iseq].angle(). iseq is the sequence counter: 0, 1, 2 Applicable only to 3-phase device case.
.ImSequences	Array of sequence currents entering m-side: use ImSequences[iseq].magnitude() and ImSequences[iseq].angle(). iseq is the sequence counter: 0, 1, 2

	Applicable only to 3-phase device case.
.SequencesExist	True when sequence data exists due to a 3-phase device.

Notes:

1. In the current version, signals and devices must be searched separately (not with the same Request object array).
    a. The RequestObject must provide a sequence (array) of signals or a sequence of devices.
2. The devices are connected between k and m-sides. The k-side is typically the left-side (with the plus sign).
3. For a General signal, the fields Va_mag and Va_phase deliver the voltage phasor. When a 3-phase bus does not contain all the phase signals, then the phasors wil appear separately. If for example, the 3-phase bus is connected only to phase-B, then only Vb_mag and Vb_phase will be populated. If it is connected only to phases B and C, then Vb_mag, Vb_phase, Vc_mag and Vc_phase will be populated.
4. Only fundamental units are used (Amperes and Watts) for device requests. The phase units are degrees.
5. Devices or signals can be targeted using search functions or by selections on the design.

For both signals and devices, the fields Signal and Device are mandatory. In some cases, the requested signal or device may be located in a subnetwork. Let us assume that a device named R1 is located in two subnetworks named DEV1 and DEV2, respectively. If the subnetworks are of the same type (not unique), then there is only one object handle for R1 and the search will return only one object referring either to DEV1 or DEV2. In that case, the reference to device data can be forced by specifying the field DeviceName to be, for example: 'DEV1/R1'. The same is applicable to SignalName specifications for retrieving appropriate data.

A demonstration example is presented here for design shown in Figure 1. It is noticed that SW1 is found on the top circuit and also in the subcircuit SUBSWITCH_Req_.

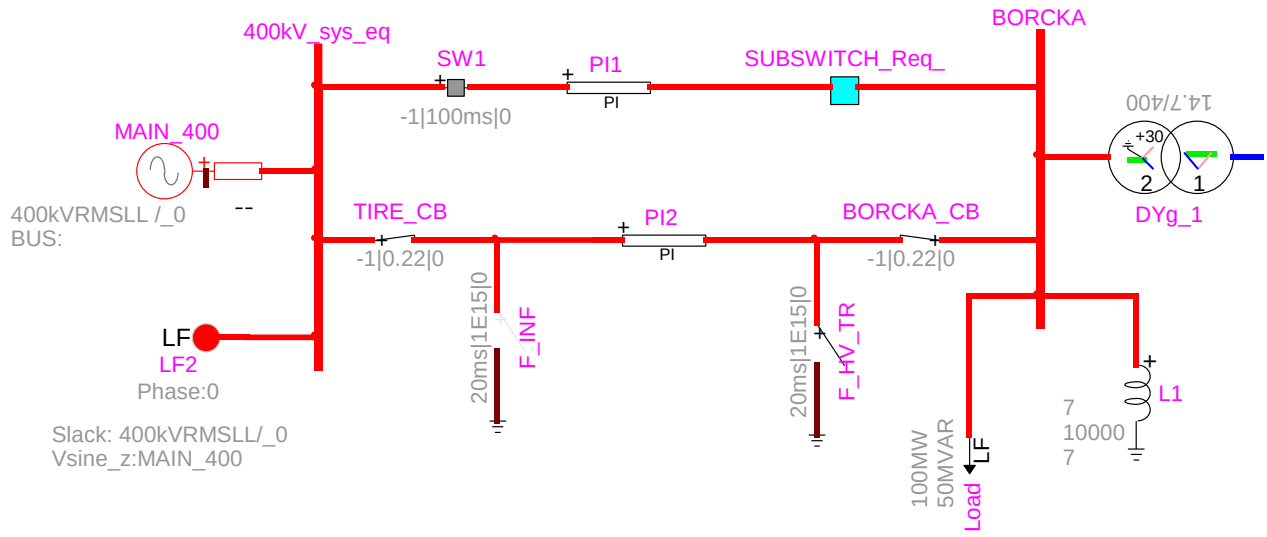


Figure 1 Example for script usage

The following script allows to extract device data for SW1 and also PI1.

```
c = SPRichConsole("console", "main_console");
c.setFont("Arial Black 14 white")
c.clear

//*Needed or making requests
parseScriptFile('get_ss_solution.dwj')

//*Select current circuit
cct=currentCircuit()

//*Create the request object array
var RequestObject = new Array();

//*search this circuit for the target device SW1------------------------
SWsearched=cct.devices(512,-1,3,'Name','SW1');
//*Select first element in this array (should have a length of 2)
SWsearched=SWsearched[0]
//*First request
RequestObject[0]=new DefineSignalDeviceViewSSRequestObject()
RequestObject[0].MessageConsole=false
//Searched device
RequestObject[0].Device=SWsearched
RequestObject[0].DeviceName='SUBSWITCH_Req_/SW1'; //specify which one

//*Search this circuit for the target device PI1------------------------
LineSearched=cct.devices(512,-1,4,'Name','PI1');
LineSearched=LineSearched[0]

//*Second request
RequestObject[1]=new DefineSignalDeviceViewSSRequestObject()
RequestObject[1].Device=LineSearched;

//*Retrieve data, execute the search
get_ss_solution(RequestObject)

//*List all object fields
writeln('First search----------------------------------SW1')
OO=( RequestObject[0])
for ( field in OO ) {
  writeln(field+' = '+ OO[field])
}

OO=( RequestObject[0].PowerCurrentDataFields)
for ( field in OO ) {
  writeln(field+' = '+ OO[field])
}
writeln('Ik1mag'+ RequestObject[0].PowerCurrentDataFields.IkSequences[1].magnitude())
writeln('Ik1phase'+ RequestObject[0].PowerCurrentDataFields.IkSequences[1].angle())
writeln('Ik2mag'+ RequestObject[0].PowerCurrentDataFields.IkSequences[2].magnitude())
writeln('Ik2phase'+ RequestObject[0].PowerCurrentDataFields.IkSequences[2].angle())
writeln('Ik0mag'+ RequestObject[0].PowerCurrentDataFields.IkSequences[0].magnitude())
writeln('Ik2phase'+ RequestObject[0].PowerCurrentDataFields.IkSequences[0].angle())



writeln('Second search----------------------------------PI1')
OO=( RequestObject[1])
for ( field in OO ) {
  writeln(field+' = '+ OO[field])
}

OO=( RequestObject[1].PowerCurrentDataFields)
for ( field in OO ) {
  writeln(field+' = '+ OO[field])
}

writeln('Ik1mag'+ RequestObject[1].PowerCurrentDataFields.IkSequences[1].magnitude())
writeln('Ik1phase'+ RequestObject[1].PowerCurrentDataFields.IkSequences[1].angle())
writeln('Ik2mag'+ RequestObject[1].PowerCurrentDataFields.IkSequences[2].magnitude())
```

```
writeln('Ik2phase'+ RequestObject[1].PowerCurrentDataFields.IkSequences[2].angle())
writeln('Ik0mag'+ RequestObject[1].PowerCurrentDataFields.IkSequences[0].magnitude())
writeln('Ik2phase'+ RequestObject[1].PowerCurrentDataFields.IkSequences[0].angle())
```

The following script is for requesting signal data for the buses BORCKA and G_BORCKA (not shown in Figure 1).

```
c = SPRichConsole("console", "main_console");
c.setFont("Arial Black 14 white")
c.clear

//*Select current circuit
cct=currentCircuit()

//*Needed for making requests
parseScriptFile('get_ss_solution.dwj')

//*search this circuit for two signals
SIGsearched1=cct.signals(512,4,'Name','BORCKA');
SIGsearched2=cct.signals(512,4,'Name','G_BORCKA');

//*Select first element in this array (should have a length of 1)
SIGsearched1=SIGsearched1[0]
SIGsearched2=SIGsearched2[0]

//*Create the request object array
var RequestObject = new Array();

RequestObject[0]=new DefineSignalDeviceViewSSRequestObject()
RequestObject[1]=new DefineSignalDeviceViewSSRequestObject()

RequestObject[0].MessageConsole=false
RequestObject[0].LForSS=0

//*Searched signals
RequestObject[0].Signal=SIGsearched1
RequestObject[0].get_pu_from_signal_attribute=true;

RequestObject[1].Signal=SIGsearched2
RequestObject[1].get_pu_from_signal_attribute=false;
RequestObject[1].UseXMLfileUnits=true;

//*Retrieve data, execute the search
get_ss_solution(RequestObject)


//*List all object fields
writeln('First search----------------------------------')
OO=( RequestObject[0])
for ( field in OO ) {
  writeln(field+' = '+ OO[field])
}

writeln('Second search---------------------------------')
OO=( RequestObject[1])
for ( field in OO ) {
  writeln(field+' = '+ OO[field])
}
```