Simulating Curtailment with COM

From OpenDSSWiki

Question

I am currently running hourly powerflows in OpenDSS and I want to set up control parameters (such as curtailment if thermal limits are reached) and iterating a powerflow after curtailment for another hour and then another check with control parameters and so on.

I was exploring my options on how to do that: My first one was using matlab to write the control loops and then using the COM interface to communicate with the OpenDSS engine. I was trying to access the values of certain parameters (for example MVA's at a specific line) through the interface in matlab but couldn't succeed.

I know in the manual it says that all of the solutions and scripts come out of the OpenDSS engine. So does that mean that the only option I have is to import the txt file that OpenDSS produces into matlab and then work from that? I was hoping to avoid that since matlab is quite slow at importing datafiles.

My second option was to write the control loops directly in OpenDSS. I couldn't find a way to do that in the current interface, so is the only viable way to create a new class in Pascal for control parameters and add it to the OpenDSS package and work from there?

Answer

August 18, 2010

You should be able to do it all through the COM interface, although if you wanted to write your own control you could start with the GenDispatcher control. It was designed for a problem that is almost the opposite of what you are trying to do. It was designed to increase the output of DG to keep the loading on a line below a kW limit (turning on DG on a feeder to keep substation loading or a line loading below limits), which is probably the wrong direction for you. Perhaps, you could add a property that would be a switch to determine whether the control is to run in curtailment mode or the opposite.

For the COM interface approach (see the Matlab examples (http://electricdss.svn.sourceforge.net/viewvc/electricdss/Examples/Matlab/) on sourceforge.net which have recently been updated) I would, for Daily mode, execute the following to start off:

```
DSSText.Command - 'set mode=daily number=1';
```

Thus, each time you execute a Solve command, it will go one hour and stop. All the Monitor and EnergyMeter elements you have defined will stay open, collecting data after each solve.

While it is stopped you can get the power flowing in a line doing something like this:

```
  DSSCircuit.SetActiveElement('Line.MyLinename');
  PowerArray = DSSCircuit.ActiveCktElement.Powers;
```

This gives an array of complex powers into each conductor of each terminal of the line, so you would have to sum them up to get total power.

You can also get symmetrical component values (sequence values):

```
  SeqPowerArray = DSSCircuit.ActiveCktElement.SeqPowers;
```

which gives you an array of the symmetrical component powers into each terminal. You probably want the positive sequence power.

Since it is usually the currents you are ultimately interested in, you can get the currents and compare them with the rated current for the line. To get the current ratings for a Line you could do:

```
  DSSLines = DSSCircuit.Lines;   % set local variable to the Lines interface
  ...
  DSSLines.Name = 'Line.MyLine';  % set the line to be the active element using the Lines interfac
  NormalRating = DSSLines.NormAmps;
  EmergencyRating = DSSLines.EmergAmps;
```

Now, since Line.MyLine is the active circuit element, you can get the currents by

```
  CurrentArray = DSSCircuit.ActiveCktElement.Currents;   % array of complex currents in each condu
```

Or

```
  SeqCurrentArray = DSSCircuit.ActiveCktElement.SeqCurrents;   % array of complex sequence current
```

Once you have determined the degree of overload, you can directly dispatch whichever Generator elements you wish. For example if you want to change the dispatch of all Generators in the circuit programmatically, you might do it like this, for example:

```
iGen = DSSCircuit.Generators.First;
while iGen > 0
   DSSCircuit.Generators.kW = MyNewGenDispatchkW;
   iGen = DSSCircuit.Generators.Next;
End
```

Do you have MS Excel installed on your computer? It is generally much better about exposing the COM interface than Matlab. Of course, I am also better with VBA than Matlab. I keep a VBA window open while writing Matlab code.

You can also use scripts for obtaining and setting most of these values. Of course, the results come back in text form and must be parsed and converted into floating point numbers and integers. For example,

```
  DSSText.Command = '? Line.MyLine.currents';
  strCurrents = DSSText.Result;
  DSSText.Command = '? Generator.MyGen.kW';
  strGenkW = DSSText.Result;
```

Execute these statements (with the names changed) in the Matlab command window and you can see what you get. Setting the values are straightforward.

```
DSSText.Command = ['Generator.MyGen.kW=' num2str(MyNewGenDispatchkW)];
```

Note that there is no looping in the DSS scripting language. You have to do that through the COM interface.

We are also working on making the DLL interface easier to implement so you can write your own models, but you will still have to be a computer programmer.

Roger Dugan

Retrieved from "http://localhost/mediawiki/index.php?title=Simulating_Curtailment_with_COM&oldid=354"

- This page was last modified on 18 August 2010, at 11:12.
- This page has been accessed 525 times.