TechNote OpenDSS Custom Scripting

From OpenDSSWiki

Contents

Custom Scripting of OpenDSS Time Simulations

(Version 7.3.3 and later)
May 18, 2010

Back to Tech Notes

This document describes how to do simple scripting of discrete-event sequential-time simulations. This is basically for solution modes that have not yet been programmed into the OpenDSS, but can also apply to some of the pre-programmed modes.

Existing Solution Modes.

As of this writing, the following solution modes have been programmed into the OpenDSS: (This listing can be seen in the help on the **Set Mode**= option.)

Mode	Description
Snapshot	Solves one power flow for the presently-defined circuit condition.
DIrect	Non-iterative snapshot solution using only the present value of the system Y matrix. Useful for debugging difficult cases were iterative snapshot power flow solution fails.
Daily	Nominally for performing 24-hr solutions following a Daily load shape.
Yearly	Nominally for performing 8760-hr solutions following a Yearly loadshape. Automatically handles EnergyMeters objects with demand interval sampling, etc.
DUtycycle	Nominally for performing sequential power flow solutions in 1s timesteps. Use this for wind plant, solar ramping, car crushers, etc.. Follows Duty loadshape.
Time	This mode is new with this version. It differs from the plain Snapshot mode in following Loadshapes and automatically sampling Monitor objects. (Also see **Set LoadShapeClass** option)
DYnamic	For performing machine dynamics response to events such as faults. Usually at a very small time step (<1ms).

Harmonic	Starting with an initial power flow solution, performs a solution at each frequency currently defined by any object in the circuit having a spectrum (see Load and Spectrum object, for example). Requires monitors to capture results.
M1	(Monte Carlo 1 power flow solution method)
M2	(Monte Carlo 2 power flow solutions method)
M3	(Monte Carlo 3 power flow solution method)
Faultstudy	For performing a short circuit current study of the present circuit. Combined with **Show Faultstudy** to show the results. Note: this is different than applying a Fault object in Snapshot or Direct modes.
MF	This mode is for performing a monte carlo fault study. Picks one of the defined Fault objects, randomizes the fault resistance, and solves in Direct mode.
Peakday	(Currently not supported)
LD1	(load-duration 1) An alternate method of performing yearly simulations using load-duration curves.
LD2	(load-duration 2) An alternate method of performing yearly simulations using load-duration curves.
AutoAdd	This executes a simple algorithm for suggesting optimal sites for capacitors and distributed generators of a specified size. (see AddType option)

The specific algorithms for these solution modes may be found in SolutionAlgs.pas. Each of these was developed in response to a specific need.

You can theoretically interact with any of these modes by setting Number=1 after setting the mode. Then after each solution (solve command) you sample the results and take some action.

Scripting Custom Simulations

There are two basic ways to script custom simulations in the OpenDSS:

1. Generate long, detailed text scripts and have the OpenDSS read them from a file (**Redirect** command). You could also paste the script into a window on the OpenDSS.EXE user interface and execute directly (or selectively).
2. Write a program in Matlab, VBA, etc to drive the OpenDSSEngine's COM interface. This allows you to generate the script on the fly, code loops, and directly access several features of the OpenDSS through the ever-expanding COM interface.

Note that there is no looping facility in the OpenDSS scripting language. If you want to do repetitive things with only small changes, you will have to generate the complete sequential script with sections repeated. This can appear to be rather inelegant, but works just fine. Alternatively, you may write some code in your preferred programming language to drive the COM interface and code the loops there. This will look better. Unless you are one of those text editor whizzes, you will probably have to write some code in either approach if the script is long. So it's six-in-one and a half-dozen in the other as far as the OpenDSS is concerned.

If the algorithm you are working on is of general use to others, we can add it to the SolutionAlgs.Pas module and add a solution mode to access it.

Snapshot Mode Scripting

In **Snapshot** mode, the OpenDSS performs one power flow solution for each invoking of the **Solve** command. All properties of circuit elements are at their most recently-defined values. Time is not advanced nor are Monitor objects and Energymeter objects, or anything else, automatically sampled. **Load** and **Generator** objects do not follow loadshapes.

Basically, if anything is to happen, you have to script it.

The default control mode is **Static**. Thus, capacitor and regulator controls, switches, etc., may change when you execute a Solve command, but Load and Generator objects remain at their defined model values (kW, kvar, PF, etc.).

For example, consider a simulation of compensating for a solar PV ramp down with distributed storage units that was performed recently:

```
Compile  C:\Myfolder\Master.DSS   ! Define the system model

Solve                   ! Initial Base solution
Set time=(0,0)          ! Make sure Time is at 0h 0s
Set Controlmode=OFF     ! We'll move the regulators manually
Sample                  ! All monitors get base solution

! Start the ramp down after 1 sec
Set sec=1
Solve      ! solution at t=1s
Sample     ! All monitors take a sample
! by t=2s, generator is down 10%
Set sec=2
Generator.PV1.kW=(2500 250 -)  ! decrement generator power 10%
Solve
Sample
! Assume Generator control detects deficit in output and sends out
! message to Storage units at t=3s
Set sec=3
Generator.PV1.kW=(2500 500 -)  ! decrement generator power another 10%
Solve
Sample
Set sec = 3.020834372 ! Message arrives at Unit 1
storage.jo0235001304.state=discharging %discharge=11.9
Solve
Sample
Set sec = 3.022028115 ! Message arrives at Unit 2
storage.jo0235000257.state=discharging %discharge=11.9
Solve
Sample
Set sec = 3.023158858 ! Message arrives at Unit 3
storage.jo0235000265.state=discharging %discharge=11.9
Solve
Sample
Set sec = 3.024604602 ! Message arrives at Unit 4
storage.jo0235000268_1.state=discharging %discharge=11.9
Solve
Sample
…etc…
```

This script starts by defining the circuit using the **Compile** command. The remainder of the script is related to making modifications to selected elements in the circuit and controlling the solution.

The next step is to perform a base case solution. This will cause all control devices to find a satisfactory setting. Since the rest of the simulation takes place in a short time frame – too short for another regulator tap change or capacitor switch to take place – the controls are subsequently disabled by setting **Controlmode=OFF**.

A "sample" command is issued to record the base case solution.

In this problem we know beforehand that the 2500-kW solar PV generator ramps down at 10% per second. It is assumed the generator monitor/control samples the output once per second and, as the generation starts to fall off, the control sends messages to the distributed storage units in an attempt to compensate.

At each step, the script:

1. sets the time using the **Set Sec**= option
2. modifies the generator output, if appropriate. This is done in this example using the inline math facility (RPN format) in OpenDSS so it is obvious what is being done, but the script could simply set the value directly.
3. executes a **Solve** command, which solves the new power flow,
4. records the result in the Monitors defined in the circuit definition using the **Sample** command.

During this simulation the Load object setting remain unchanged. (The solution might change slightly due to different voltages in the network.) Loads do not follow any Loadshape values in this solution mode. (See Time Mode Scripting below).

An external application was used to compute the latency in the communications system. This is converted to the appropriate time in the simulation script. Thus, the first message arrives at sec = **3.020834372**. The messages arrive at irregular intervals. This is of no consequence in this simulation because time is only used for reporting the results in the Monitor records.

As each message arrives, its effect is simulated by setting the Storage elements to discharging at a specified discharge rate. In this case, it is assumed the storage controller/generator monitor determines that the discharge rate should be 11.9%. Note that the Storage elements remain ON in dispatching mode for subsequent solutions so it is not necessary to redispatch them until a different dispatch level is desired. So when the message arrives at Unit 2 a few ms later, only the new Storage unit need be redispatched.

There are many similar problems that can be scripted in this simple manner. The key thing to remember is that in the default Snapshot mode, everything must be explicitly scripted.

Time Mode Scripting

This mode (**Set Mode=Time**) was added at this revision of the program to add a few automatic time-dependent features to the simple Snapshot model. Specifically, in this mode when the Solve command is issued,

1. Load, Generator, etc objects follow one of {**Daily, Year, Duty**} loadshapes as time is changed. You specify which one using the **Set LoadShapeClass**= option. **None** is also an option.
2. Monitors are automatically sampled. (If you use Energymeters, you will have to sample them separately).
3. Time is incremented by the stepsize after the solution has been performed. You may override that by subsequently setting time to a different value if you wish.

This gives you a little more flexibility in the types of simulations you may wish to script. You can more easily simulate longer time durations with the loads and generatos following the loadshapes associated with them.

The default number of solutions for this mode is 1. However, you can change that as you see fit, with time incrementing at the end of each solution.

Recall that nearly everything is reset when you change modes using the **Set Mode**= option. Use the **Get Stepsize**, **Get Number**, etc commands (Get is the opposite of the Set command) to see default values after the mode is entered.

For example, the previous script can be simplified somewhat if we assume a Duty cycle loadshape has been defined to describe the generator load shape:

```
Compile  C:\Myfolder\Master.DSS   ! Define the system model

Solve          ! Initial Base solution in Snapshot mode

! Define a solar ramp function on 1s intervals
New Loadshape.SolarRamp npts=300 Interval=(1.0 3600 /)
~ mult=(file=solarramp.csv)

! assign loadshape to PV generator
Generator.PV1.Duty=SolarRamp

Set mode=Time  LoadshapeClass=Duty
Set Stepsize=1s
! The ramp starts down after 1 sec
Solve      ! solution at t=0 and sample and inc time
Solve      ! solution at t=1s;  t=2s
! Generator automatically follows loadshape
Solve      ! solution at t=2s; t=3s
! Assume Generator control detects deficit in output and sends out
! messages to Storage units at t=3s
Solve      ! solution at t=3s; t is set to 4s
! override time for message arrivals…
Set sec = 3.020834372 ! Message arrives at Unit 1
storage.jo0235001304.state=discharging %discharge=11.9
Solve     ! solution at t=3.020834372; time incremented to 4.020834372
Set sec = 3.022028115 ! Message arrives at Unit 2
storage.jo0235000257.state=discharging %discharge=11.9
Solve
Set sec = 3.023158858 ! Message arrives at Unit 3
storage.jo0235000265.state=discharging %discharge=11.9
Solve
Set sec = 3.024604602 ! Message arrives at Unit 4
storage.jo0235000268_1.state=discharging %discharge=11.9
Solve
… etc …
! go to 4.0s solution; next generator monitor sample
Set sec=4
Solve
! now simulate arrival of new messages at Storage elements
Set sec = 4.020834372 ! Message arrives at Unit 1
! dispatch twice as high because generator output has dropped
! another 10%
storage.jo0235001304.state=discharging %discharge=(11.9 2 *)
Solve     ! solution at t=3.020834372; time incremented to 4.020834372
… etc …''
```

In the initial base solution, the solar PV generator is at full output and the loads are constant as well. Next, we define a **Loadshape** object that embodies the solar ramping characteristic we will to simulate and assign it to the **Duty** property of the PV generator. Note that the units on the **Loadshape Interval** property is hours, so we do a little in-line math to specify that the points in the solarramp.csv file are in 1s intervals.

Then we switch the solution mode to the general Time mode and set the **LoadShapeClass=Duty**. This means that for the duration of this simulation, **Load** and **Generator** objects will follow the **Loadshape** object that has been associated with their respective **Duty** property.

We set the stepsize to 1s, although one might argue that it doesn't really matter much in this example because later, we will be setting time explicitly to a different value. However, we let the time increment automatically for the first 3 solutions.

The first Solve is at time 0 since it is the first after the mode change. The **SolveGeneralTime** procedure in SolutionAlgs.Pas auto-increments the time after the solution so that time is 1s at the end of this statement. The generator ramp starts down after t=1s and we assume the device monitoring the output once per second will notice the decline after the t=2 solution and send out dispatch messages at t=3.

After the t=3s solution, the time is auto-incremented to 4s. However, we override that to simulate the arrival of the first message at a storage unit a few milliseconds later. Time incremented after each solution, but the incrementing is overridden for the remainder of the simulation.

At the conclusion of the simulations, the Monitor objects will each contain a number of records at irregular time intervals. These can be seen in the usual way with either the Show, Export , or Plot commands.

Other Ways to "Roll Your Own" Simulation

At the top of the Help form under Executive, there are a number of commands beginning with the underscore ('_') character. These commands give you access to the various steps of the solution process so you can customize your own solution process if you desire. These are shown in Figure 1.

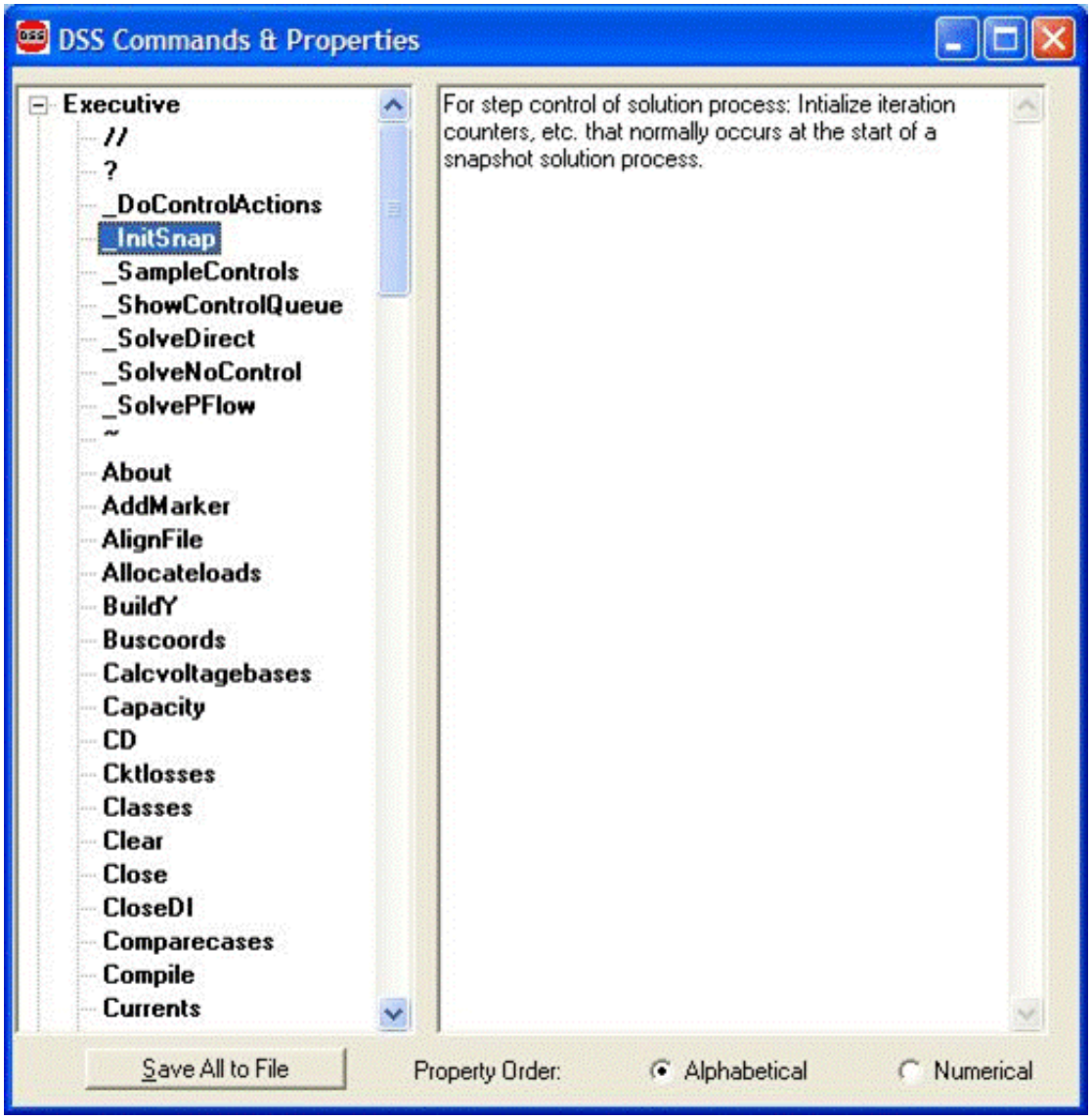

Figure 1. Help form screen capture.

The OpenDSS **Solve** command will typically invoke at some point the **SolveSnap** function in the program. For the basic power flow solution the **SolveSnap** function executes the following steps:

1. Initialize Snapshot (**_InitSnap**)
2. Repeat until converged:
3. Solve Circuit (**_SolveNoControl**)
4. Sample control devices (**_SampleControls**)
5. Do control actions, if any (**_DoControlActions**)

In _SolveNoControl you may get either an iterative power flow solution or a direct solution. You will usually get an iterative one unless the mode is Harmonic or Dynamic, in which cases the default is direct solution. You can specifically require a particular solution by **_SolveDirect** or **_SolvePFlow**.

Using these commands, you can basically design your own solution process. Of course, you would be required to do your own management of time and of sampling.

These functions are also available as direct calls through the COM interface. In addition, there is access to other quantities such as the convergence flag, total iterations, etc. It generally makes more sense to “roll your own” solution algorithm using a program that accesses the COM interface, but if you want to do it in a script, you can.