OpenDSS LOADSHAPE Usage

(Version 7.3.3)

May 15, 2010

---- Recent Updates ----

New Ways to Enter Array Data

To better accommodate large amounts of AMI data, the capability has been added to import packed files of doubles or singles into floating point array properties. For example, you can use the syntax:

```
Mult=[file=myfile.txt]

Mult=[dblfile=myfile.dbl]

Mult=[sngfile=myfile.sng]
```

This syntax will work on just about any property of any class of element in the OpenDSS where an array of numbers is expected.

The addition of the capability to handle the two packed binary file types should significantly speed up the importing of AMI data into Loadshape objects. Also, the way the Mean and Std Deviation calculation is done has been changed to delay calculation until these values are actually needed. This should minimize the computing overhead on reading loadshapes back in.

You can create the .DBL and .SNG files with any language that can create packed binary files. If you are working in VBA, look at **Put** & **Get**. For an example, look for the **WriteToSingleFile** macro in the **SampleDSSDriver.xls** file on the SourceForge website.

However, I also added options to the **Action** property in the Loadshape Object to allow you to use OpenDSS to create the **dbl** and **sng** files. (see the next section)

To go along with changes allowing LoadShapes to be read from files of Singles and files of Doubles, the **CvrtLoadshapes** command has been added to write all presently-loaded LoadShape elements to either **SNG** or **DBL** files (see description of the command below). And the command will also produce a script for reading them back in. So you can load the LoadShapes <u>once</u> from text scripts or files and save them.

Note: this won't save you anything if you are changing the LoadShape objects every time you run. You would be better off creating them initially using either SNG or DBL files. In Matlab, use the fwrite statement with the precision for the array:

```
fwrite(obj,A,'precision')
```

where precision=*single* or *double*, etc.

In Excel VBA, for example:

```
Dim sngTemp as Single
Dim X8760(1 to 8760) as Double  ' Array of multipliers
…  (do stuff)
Open MyFileName.sng For Binary Access Write As #1

   For i = 1 To 8760
      sngTemp = X8760(i)    ' convert double to single
      Put #1, , sngTemp     ' put it to the binary file
   Next i
Close #
```

Saving Sng or Dbl Files from the Loadshape Element

First, you read the LoadShape data into the program from text files or just putting the data into a long array in the OpenDSS script.

Then you would issue this command, for example, to save a file of singles:

```
Loadshape.myloadshape.action=sngsave
```

This will save the presently defined myloadshape values into

```
Myloadshape_P.sng
```

If you defined Qmult, you will also get

```
Myloadshape_Q.sng
```

In the Result window, you will get an example script you can use for importing this data in its new format.

Thus, you can read in your loadshapes one time from text files and save them to a file of singles, for example. (Singles are usually sufficient for LoadShape objects). Then subsequent simulations should go faster. (be sure to edit your scripts after the first time). This should be a big help if loading 8760's for every Load.

Computing Mean and Std Deviation

In previous versions of OpenDSS, the **Mean** and **StdDev** properties of any loadshape was computed upon reading the loadshape out of the array definition. In the interest of faster processing, this has been changed so that these values are only computed when the **Mean** or **StdDev** property values are requested by some process within OpenDSS. Only a few solution modes, such as Monte Carlo load studies, actually use these so they will be only rarely computed.

This should be transparent to the user. If you show the Properties for a LoadShape object, the values are computed.

The CvrtLoadshapes Command

Issuing this command will cause the OpenDSS two write out all presently-loaded LoadShape Objects to either files of Single or files of doubles as specified.

```
cvrtloadshapes type=sng  (this is the default)
cvrtloadshapes type=dbl
```

A separate file is created for each P and each Q loadshape. The naming should be obvious. The command also creates a script file named **ReloadLoadshapes**.**DSS** for reading the loadshapes back in.

This can create a lot of files. Be sure the present folder is really where you want the files. Of course, you can always move the files later and change the reload script accordingly.

--- Handling AMI Data ---

Modifications have been made to the Load, Generator, and Loadshape objects to better facilitate the handling of AMI data that is assumed to be in units of kW and kvar. For example, the Load object will automatically pick up the maximum kW value and set the **kW** property so that the Snapshot power flow solution will represent a non-coincident peak load case. The **UseActual** property of the LoadShape object is key to this.

Using the UseActual Property of LoadShape

Most AMI data is in actual kW and kvar values. To facilitate this in AMI simulations, the **UseActual** property has been added to the LoadShape object. The Help text on the property is:

> *{Yes | No* | True | False*} If true, signals to Load, Generator, or other objects to use the return value as the actual kW, kvar value rather than a multiplier. Nominally for AMI data.*

It is a Boolean property that defaults to No (False). This is default behavior of the LoadShape object that has been the only option up to this time.

The conventional way to use LoadShape objects has been to use the values as a multiplier against the defined load. You may normalize them to 1.0 per unit either when you read them in (**Action=Normalize**) or in some external program. Then the value of the array is multiplied by the **kW** or **kvar** property of the Load, Generator, or other object. You could also define the load as **kw=1** (and **kvar=1** if you are defining **Qmult**) and then define the loadshape in actual values. You can still do this, but the downside is that the initial Snapshot solution is bogus. It is generally preferable to get a snapshot solution for the actual peak values to check your model.

The recent modifications to the LoadShape object allow you to define the loadshape in actual values and get a reasonable snapshot solution for the peak kW of the loadshape. 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.

If you want something else, you will need to redefine **kW** and **kvar** (or **kW**, **PF**) after you set the **Duty**, **Daily**, or **Yearly** properties.

You can query the **Pmax** and **Qmax** properties of the Loadshape object to see what was computed. Keep in mind that the **Qmax** value is the value of kvar at the time of maximum kW, if defined using the **Qmult** property. If you want something different, you may override the computed values by setting either or both of these properties after reading in the loadshapes.

Example: To query the **Pmax** and **Qmax** properties in a VBA app:

```
DSSText.Command = "? Loadshape.Myshape.Pmax"
MyPmaxStr = DSSText.Result
DSSText.Command = "? Loadshape.Myshape.Qmax"
MyQmaxStr = DSSText.Result
```

Then if all your curve data arrays are synchronized, you should be able perform a snapshot solution and proceed with yearly simulations, for example.