OpenDSS Training Tutorial

This tutorial walks the user through the install and setup of OpenDSS and OpenDSSDirect, explained using a very simple example circuit modeled from the OpenDSS user interface, and analyzed using OpenDSSDirect in a Jupyter Notebook. The commands in OpenDSS represent a basic circuit with only the minimum number of parameters set for each circuit element. One can reference the OpenDSS Manual for a far more detailed settings and additional parameters. [One can find the supporting files for this tutorial here.](https://github.com/NREL/CIFF/tree/Tutorial) One must have OpenDSS and Python installed to work through this tutorial. An installation guide can be found at the above link as well.

Questions regarding this training can be directed towards Erik Pohl ([erik.pohl@nrel.gov](mailto:erik.pohl@nrel.gov)) or Killian McKenna ([killian.mckenna@nrel.gov](mailto:killian.mckenna@nrel.gov)).

1. Intro to OpenDSS
   1. OpenDSS

“The Open Distribution System Simulator (OpenDSS, or simply, DSS) is a comprehensive electrical system simulation tool for electric utility distribution systems.”

* Open-source load flow tool used for feeder analysis, DER integration studies, and grid modernization.
* Limited graphical user interface (script-driven) but extremely flexible. Can be driven and read from MS Office VBA, MATLAB, **Python**, C#, R, and others for external analytical capabilities and improved data visualization (e.g. EMeRGE). Designed to allow expansion!
* Exports data in easy to read/import csv format.
  1. Solution Capabilities (plus many more):
* Unbalanced, multiphase powerflow
* Radial or meshed feeders
* Fault analysis
* Harmonics analysis
* Flicker analysis
  1. Model components (plus many more):
* A comprehensive device/asset library (cables, conductors, transformers, cap banks, switches, reclosers, fuses, etc.
* Controls (regs, LTC, switched caps, DER smart inverters, energy storage dispatch, protective devices and relays, etc.)

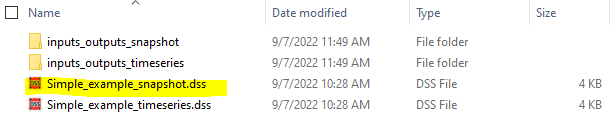
1. OpenDSS Model Build – Voltage and Loss Assessment
   1. Getting Started

To begin, go to the link above and download the files needed for this tutorial. Click on the green “Code” button and then “Download Zip.” Move this zip file to a folder of your choice.

Graphical user interface, text, application

Description automatically generated

Open the folder OpenDSS\_model\_files and then double-click on the Simple\_example\_snapshot.dss file.



Once OpenDSS launches, your screen should look like this.

Graphical user interface, text, application, email

Description automatically generated

A few helpful notes to start navigating OpenDSS:

* Use ! or // to comment a line out
* Use /\* … \*/ to comment out multiple lines
* Use ~ to continue a command on next line
* Yellow screen indicates unsaved changes. Use ctrl+S to save
* Only run what is highlighted with ctrl+D
* General format of objects: **new object\_class.object\_name parameter1= parameter2= …**
  1. Defining your circuit

The oneline below shows the simple circuit that we will be modeling in this training. The circuit includes a substation transformer, three conductors, two loads and a distributed generator.



One should begin every script with the **clear** command to clear all of the OpenDSS registers. One can set the base frequency with command **set DefaultBaseFrequency=50** or by selecting **Set** in the toolbar at the top of your screen. Setting a data path with **set datapath** ensures that all files and figured imported and exported from your script go to the correct location. Lastly, we need to define our circuit, and give it a name.

clear

set DefaultBaseFrequency=50

set datapath = C:\Users\epohl\Desktop\CIFF\_Dehli\_Presentation\Git\_repo\ciff\OpenDSS\_model\_files\inputs\_outputs\_snapshot

new circuit.simple\_example\_snapshot basekv=132 pu=1.0 angle=0 frequency=50 phases=3

* basekV = L-L voltage of your source bus
* pu = balanced per unit voltage of your source bus
* angle = Base angle, degrees, of the first phase
* frequency= frequency of the source
* phases = number of phases
* One can optionally define a source impedance or SC currents to determine the stiffness of their source (important for fault/arc-flash studies). See pg 142-143 in the OpenDSS Manual.
  1. Build a Substation Transformer

The process for defining a transformer is the same for a substation transformer, as for any transformer.

new transformer.substation phases=3 windings=2 buses=(sourcebus, A) conns=(wye, wye)

~ kvs=(132, 33) kvas=(30000, 30000) %loadloss=0.1 xhl=12.5

* Phases = number of phases. Default is 3
* Windings= number of windings. Default is 2
* Buses = busses connected to transformer in order of winding 1 (source-side) followed by winding 2 (load-side
* Conns=Connection of windings, wye or delta. Default is wye.
* Kvs=rated voltage of windings
* Kvas=kva rating (OA) of windings
* %loadloss= Percent Losses at rated load.. Causes the %r values to be set for windings 1 and 2.
* Xhl= percent reactance high-to-low (winding1 to winding 2)
  1. Define Linecodes

Linecodes are library objects that provide the impedance characteristics for lines and cables. One assigns a linecode to a line object when adding lines to their model. You can add as many linecodes as you would like and can keep this library in a separate .dss file using the redirect command (covered later on).

new linecode.795\_AAC nphases=3 R1=0.0746 X1=0.373 R0=0.1796 X0=0.9451 units=km

~ normamps=810 emergamps=972

* R1,X1 = Positive-sequence resistance and reactance
* R0,X0 = Zero-sequence resistance and reactance
* Units=Resistance distance unit (i.e. Ohms/km is entered as km)
* Normamps=normal rated current
* Emergamps=emergency rated current
  1. Build Lines

Add lines to your model using the line codes we just created.

new line.lineA-B bus1=A bus2=B length=5 phases=3 units=km linecode=1\_0\_ACSR

* Bus1=start bus
* Bus2=end bus
* Length=conductor segment length
* Phases=number of phases
* Units=length unit
* Linecode=the desired linecode (“linecode.” class name is omitted)
  1. Build a Distribution Transformer

To build a distribution transformer, we follow the exact same process as our substation transformer, but with the appropriate high-side and low-side voltages. Note the slightly different syntax shown below, that allows the user to define each winding individually, as opposed to the array style we used for our substation transformer. Both methods accomplish the same thing. User preference.

new transformer.dist1 phases=3 windings=2 XHL=2

~ wdg=1 bus=C conn=wye kV=33 3 kva= 1000 %r=0.55 XHT=1

~ wdg=2 bus=D conn=wye kV=0.4 kva= 1000 %r=0.55 XLT=1

* 1. Add Loads

One can add loads to any defined bus. A load can be defined using any of the following combinations: 1. kW and PF, 2. Kw and kvar or 3. kVA and PF. Loads are assumed to be balanced. To simulate unbalanced loads, use three single-phase loads.

new load.primary\_load1 bus1=B phases=3 kv=33 kw=6750 kvar=3269 model=1 !7500kVA at pf=0.90

* Bus1= Name of bus to which the load is connected. Include node definitions if the terminal conductors are connected abnormally. 3-phase Wye-connected loads have 4 conductors; Delta-connected have 3. Wye-connected loads, in general, have one more conductor than phases. 1-phase Delta has 2 conductors; 2-phase has 3. The remaining Delta, or line-line, connections have the same number of conductors as phases.
* Phases=number of phases for load
* Kv= Base voltage for load. For 2- or 3-phase loads, specified in phase-to-phase kV. For all other loads, the actual kV across the load branch. If wye (star) connected, then specify phase-to-neutral (L-N). If delta or phase-to-phase connected, specify the phase-to-phase (L-L) kV.
* kW= nominal active power, kW, for the load. Total of all phases. See kVA.
* Kvar= Base kvar. If this is specified, supercedes PF.
* Model= Integer defining how the load will vary with voltage. 1 = Constant P and constant Q. See page 169 in OpenDSS Manual for alternative models.
  1. Set Solution and Control Modes

OpenDSS has several built-in solution modes to address different types of analyses. In this training, we will be using the Snapshot and Daily modes.

By setting our solution mode to **snapshot** OpenDSS will solve a single snapshot powerflow for the present conditions and time will not advance.

Setting the solution mode to **daily** will iterate through each timestep in your loadshape and solve the circuit each time.

Our control mode will be set to **static**. Is static mode time does not advance. Control actions are executed in order of shortest time to act until all actions are cleared from the control queue. Use this mode for power flow solutions which may require several regulator tap changes per solution.

set controlmode=static

set mode=snapshot

* 1. Add Monitors/Meters

Some basic results can be seen in the summary window on the left side of your window. To plot, export, or visualize any values one needs to place monitors and meters at the desired locations in your circuit.

* + 1. Monitors

Monitor objects are circuit elements that can be connected to a terminal on another circuit element to collect samples of various metrics. In this example we collect powers, voltages, currents, and tap positions of our LTC.

To do this, we must place three monitors at our substation transformer. Each monitor will collect a different metric, designated by the **mode** parameter.

new monitor.substationpowers element=transformer.substation terminal=1 mode=1 ppolar=no

new monitor.substationVI element=transformer.substation terminal=1 mode=0

* Element = name of element to which the monitor is connected.
* Terminal = No. of terminal to which the monitor is connected.
* Mode = integer to describe what the monitor will save (0: V and I each phase 1: kw/kvars each phase, 2: transformer taps)
* Ppolar = If yes, will return power as magnitude and angle. If no, will return power as complex kW and kvar.
  + 1. Energymeters

Similar to a monitor, an EnergyMeter object is connected to a terminal of a circuit element to collect certain values. It has, however, more capabilities than a meter in the real world, as it can access values at other places in the circuit as well. EnergyMeters can measure power and energy values, at it’s location, but also losses and overload values across the entire circuit. One does not specify the values that are collected in the command line.

new energymeter.substation element=transformer.substation terminal=1

!new energymeter.mloadB element=line.lineA-B terminal=1

* 1. Running the Model

Before running the model one must first set their voltage bases to allow for per unit calculations.

set voltagebases=[132 33 0.4]

calcvoltagebases

One must end each script with the **solve** command to run the powerflow. One then must press **ctrl+a** to select the full script, and then **ctrl+d** to run it.

solve

To verify the simulation has run, one will see at the bottom of their window the following bar, showing that the circuit was solved, as well as the number of iterations run to converge.



* 1. Snapshot Results

One can access results of the powerflow in a variety of ways. First, the summary tab on the left side of the window (left image below), shows some of the basic metrics.

In SnapShot mode, one first needs to instruct all meters and monitors to take a measurement at the end of the simulation. This is done with the following lines.

monitor.substationVI.action=take

monitor.substationpowers.action=take

energymeter.Grid.action=take

Results can be accessed in a variety of ways, depending on the user preferences.

1. The **visualize** command presents a bus diagram for any circuit element to see currents or voltages at each terminal.
2. the **export** command will create a csv file containing the designated values
3. the **show** command will display a txt file containing the values.
   * 1. Summary Results

The summary window on the left side of your screen can provide some high-level metrics to begin one’s analysis.

Text, letter

Description automatically generated

* + 1. Powers/Currents/Bus Diagrams

One can then use the dropdown menus at the top of the screen or the **visualize** command to see, currents, voltages, or powers, for any circuit element. The 3ph currents for line A-B are shown below.

visualize what={currents} element=line.lineA-B

Diagram

Description automatically generated

One important item to note, is the diagram notation shown for LineA-B currents. It is standard practice to show current injection on both sides of this bus. One will notice that the different phase angles provide the appropriate signs to designate current flow direction. In the example above, current is flowing from left to right, despite the inward arrows on both sides of the bus.

Use one of the following command to access currents for all circuit elements presented in either a txt or csv file.

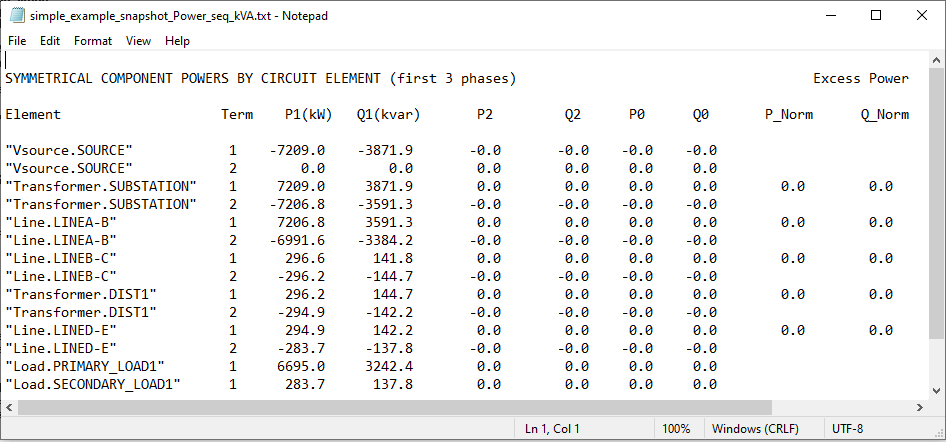
export currents

show currents

Graphical user interface, text, application, table

Description automatically generated

show powers



* + 1. Voltage

Use the same commands as above to see voltages at each bus.

show voltages

Text, table

Description automatically generated

**NOTE** the undervoltages on buses c, d, and e!

As mentioned, energymeter objects have the ability to collect values from the entire circuit. One useful application of this functionality, is viewing a voltage profile. One can either add the command below (after the **solve**) or one can select Plot 🡪 Profile from the toolbar at the top of your window. Primary conductors are shown as solid lines, and secondaries are shown as dotted lines. By default, OpenDSS plots L-N pu voltages. One can specify L-L or even just primary conductors with the second or third line below. The red lines on these plots represent allowable voltage tolerances (+/- 0.05 pu).

plot profile phases=all

plot profile phases=LLall

plot profile phases=LLprimary

Chart, line chart

Description automatically generated

Use the command below to show voltage drop across each circuit element.

show deltaV

Table

Description automatically generated

* + 1. Losses

show losses

Table

Description automatically generated with medium confidence

* 1. Fixing Voltage/Loss Issues
* Adjust source bus or LTC setpoint (see below)
* Install a capacitor bank (see below)
* Reconductor

For each of the methods above, plot our voltage profile again and show losses.

* + 1. Define Regulator Control/LTC

If one wishes to model an autotransformer (i.e. load-tap changer at the substation transformer) one would also need to specify the max and min tap positions as a pu voltage to the substation transformer object above.

Note the math used to calculate the ptratio is in RPN (Reverse Polish Notation).

new transformer.substation phases=3 windings=2 buses=(sourcebus, A) conns=(wye, wye)

~ kvs=(132, 33) kvas=(30000, 30000) %loadloss=0.1 xhl=12.5 sub=yes maxtap=1.05 mintap=0.85

new regcontrol.substation transformer=substation winding=2 tapwinding=1 vreg=(120)

~ ptratio=(33000 3 sqrt / 120 /) band=2

Table

Description automatically generated

Graphical user interface, chart, line chart

Description automatically generated

* + 1. Define a Primary Capacitor Bank

Placing a capacitor bank can both reduce losses and improve voltage. Use the following command to place a capacitor bank.

new capacitor.cap bus1=B phases=3 Kvar=1200 kV=33 Conn=wye basefreq=50

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, line chart

Description automatically generated

* + 1. Reconductoring

As we still have a slight undervoltage, one can reconductor lineA-B simply by changing the line code to a larger conductor. If we reconductor this line with 795 AAC, we dramatically reduce voltage drop and conductor losses.

new line.lineA-B bus1=A bus2=B length=5 phases=3 units=km linecode=795\_AAC

Table

Description automatically generated

Chart, line chart

Description automatically generated

* 1. Plot Circuit Diagram

To visualize our circuit as we may be used to seeing, we can use the following commands to plot our simple example.

Set marktransformers=yes

Set TransMarkerSize=3

plot Circuit

Graphical user interface

Description automatically generated

* 1. Add LoadShapes for TimeSeries Simulation

Open the second .dss file in the downloaded folder called Simple\_example\_timeseries.dss.

Loadshapes are needed for sequential-time power flow solutions. A LoadShape object is a list of multipliers ranging from 0-1 that applied to the base kW values of the loads. Loadshapes can be daily, yeaerly, etc. One can list out the multipliers directly in the command line. In this example, we will use a .txt file with the multipliers listed on a separate line each. One must save the .txt files in the same directory as their .dss file or specify the full path in the command.

One can define multiple load shapes and apply them to different loads.

new loadshape.daily\_primary npts=24 interval=1.0 csvfile=daily.txt

* Npts= Number of points to expect when defining the curve
* Interval= time interval of the data, Hr. Default=1.0. If the load shape has non-uniformly spaced points, define the interval as 0.0.
* csvfile= Name of a CSV file containing active power load shape data, one interval to a line. For variable interval data enter one (hour, multiplier) point to a line with the values separated by a comma. Otherwise there is simply one value to a line.

We also need to assign this load shape to each of the loads in our model. One must add the following command to each load instance defined above.

new load.primary\_load1 bus1=B phases=3 kv=33 kw=5000 kvar=1640 model=1

~ daily=daily\_primary status=variable

* daily= Name of Daily load shape.
* Status= {fixed| variable}. Default is variable. If fixed, then the load is not modified by multipliers; it is fixed at its defined base value.

This can be done manually, one at a time, however this can be impractical when there are a large number of loads in your model. Instead one can use the **BatchEdit** command to edit multiple objects of the same type in one line. One could also apply a loadshape to distributed generators.

BatchEdit Load..\* daily=daily\_primary status=variable

Use the following command to plot the loadshape defined above.

plot loadshape object=daily\_primary

One must also change the mode from snapshot to daily.

set mode=daily

* 1. Timeseries Results

Viewing results for timeseries simulations generally works the same way as for a snapshot simulation with a few minor differences. One no longer needs to instruct the meters and monitors to take measurements as they do this automatically at the end of each timestep.

The show and visualize commands will show the last timestep results, as will the summary box in the main OpenDSS window. These are commented out in this .dss file. The exported csv files for our monitors will now show a value for each timestep in the simulation.

To plot timeseries for voltage and powers for our two substation monitors use the following commands.

plot monitor object=substationpowers channels=(1 2 3 4 5 6)

plot monitor object=substationVI channels=(1 3 5)

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

1. OpenDSSDirect for Solar Integration (Jupyter Notebook)

OpenDSSDirect provides a means of “driving” the OpenDSS program from a python script. We will be using a Jupyter notebook to analyze our circuit.

Open a command line interface and navigate to the folder containing the file OpenDSSDirect\_Tutorial.ipynb using the following command. One must replace C:\Users\epohl\Desktop\OpenDSS\_Tutorial with the path to your folder.

cd C:\Users\epohl\Desktop\OpenDSS\_Tutorial

Then type the following command to launch a jupyter notebook in your browser.

jupyter notebook

Table

Description automatically generated with low confidence

Click on OpenDSSDirect\_Tutorial.ipynb