DPVProt Documentation

Release 0.1

Tom McDermott

Oct 01, 2021

CONTENTS:

CHAPTER

ONE

DPVPROT SCRIPT DOCUMENTATION

1.1 AtpReduction.py

Writes a reduced-order ATP model, based on the model reduction of a full-size OpenDSS model. Must be called from RunReduction.py

Public Functions:

WriteAtp does the work

param atpfile the name of the ATP file to write

type atpfile str

param pairs link information passed from RunReduction.py

type pairs array

param retained names of retained buses, passed from RunReduction.py

type retained set

param buses bus information, passed from RunReduction.py

type buses dict

param seqz sequence impedancs of retained branches, passed from RunReduction.py

type seqz dict

param foundCapacitors indicates whether feeder capacitors are in ReducedCapacitors.dss

type foundCapacitors boolean

param bNoCaps indicates to ignore shunt capacitance in line pi-section models

type bNoCaps boolean

returns writes a message if ATP file not produced

rtype str

1.2 comtrade.py

Local version of David Parrini's COMTRADE 1991 - 2013 library.

From https://github.com/dparrini/python-comtrade with patches. Should make a pull request.

Public Functions:

main does the work

1.3 ParseSeqZ.py

Tabulate seqz outputs from OpenDSS

The script reads a local buslist.dat for the bus names of interest. Only the first comma-separated value is of interest, so the same buslist.dat file used with RunFaults.py also works with this file.

Reads the seqz.csv file exported from an OpenDSS faultstudy solution.

The output is a summary of the impedances and fault currents at each bus of interest.

Public Functions:

main does the work

param kVLL the line-to-line voltage [kV] to calculate fault current from impedance

type kVLL str

param Rf the maximum fault resistance [Ohms] for a single-line-to-ground fault

type Rf float

1.4 plot_opendss_feeder.py

Plots the OpenDSS feeder from a JSON file

Reads the feeder components and coordinates from a local ReducedNetwork.json file. Creates j1reduced.pdf high-quality plot, then shows a screen plot that may be saved to PNG.

ReducedNetwork.json is output from RunReduction.py. It contains a networkx graph, with supplemental node and link data to describe the OpenDSS components. In this file, a node is as defined by networkx, not as defined by OpenDSS.

The node id is the OpenDSS bus name, and the node ndata is:

- shunts (str): an array of fully-qualified names of loads, capacitors and DER attached to this node
- nomkv (float): nominal line-to-line voltage [kV] at this location, if known
- kw (float): total load kw at this node
- kvar (float): total load kvar at this node
- capkvar (float): total kvar of shunt capacitors at this node
- derkva (float): the total kva of DER (all PV in this project) at this node
- source (boolean): true only if the circuit source is connected here
- phase (str): include a, b, and/or c if those phases are present at this node
- busnum (int): the sequential bus number of this node in the corresponding ATP model

- x (float): horizontal coordinate of this node location, arbitrary units
- y (float): vertical coordinate of this node location, arbitrary units

The link data is:

- eclass (str): the OpenDSS class name, line or transformer
- ename (str): the OpenDSS instance name, with its class
- source (str): the node id corresponding to bus1 from OpenDSS
- target (str): the node id corresponding to bus2 from OpenDSS

The edata for each line link may contain:

- r1 (float): positive sequence resistance, in Ohms
- x1 (float): positive sequence reactance, in Ohms
- r0 (float): zero sequence resistance, in Ohms
- x0 (float): zero sequence reactance, in Ohms
- c1 (float): positive sequence capacitance, in nF
- c0 (float): zero sequence capacitance, in nF
- len (float): length of the line, in km. This may be zero for a switch.
- phases (str): include A, B, and/or C if those phases are present in this link

The edata for each transformer link may contain:

- phs1 (str): include A, B, and/or C if those phases are present on the primary
- phs2 (str): include A, B, and/or C if those phases are present on the secondary
- conn1 (str): w if the primary is wye, d if delta
- conn2 (str): w if the secondary is wye, d if delta
- kv1 (float): primary winding kV rating, line-to-neutral for single-phase, line-to-line otherwise
- kv2 (float): secondary winding kV rating, line-to-neutral for single-phase, line-to-line otherwise
- kva1 (float): kva rating of the primary
- kva2 (float): kva rating of the secondary
- xhl (float): percent reactance
- r1 (float): equivalent positive sequence resistance, in Ohms from the primary
- x1 (float): equivalent positive sequence reactance, in Ohms from the primary
- r0 (float): equivalent zero sequence resistance, in Ohms from the primary
- x0 (float): equivalent zero sequence reactance, in Ohms from the primary
- phases (str): include A, B, and/or C if those phases are present in this link

Consult the networkx module documentation for more information about the json file format.

Public Functions:

main does the work

param arg1 base file name, don't add the extension, defaults to ReducedNetwork

type arg1 str

param arg2 choose 1 to include text node labels (default), or 0 not to

type arg2 int

returns writes information and warnings about missing nodes or edges to the console

rtype str

1.5 RelayPerformance.py

Summarize protection performance from a set of OpenDSS dynamic solutions.

First, use a local dofaults.bat, which invokes RunFaults.py, to produce events.out. This script appends a cumulative summary of the fault results to the file dofaults.rpt. The summary includes total number of cleared and uncleared faults, the total number of utility devices that tripped, failed to trip, or false-tripped, and the total number of DER devices that tripped. Considering faults that cleared, the minimum, maximum, mean, and standard deviation of the clearing time is reported.

Also writes the header and data rows for a summary table in LaTex format, to the file dofaults.tex.

Public Functions:

main does the work

param csv_name the fault output file name, defaults to events.out

type csv_name str

returns writes the same summary to console as appended to dofaults.rpt

rtype str

1.6 RelaySummary.py

Summarize protection performance from a set of OpenDSS dynamic solutions.

Use RunEventStudy.py to produce all events.out files, renamed to pv*.out. The script expects to read a set of files pv_%%_$$.out, where %% is an integer PV penetration percentage, and $$ indicates the scheme. $$ should be cat1, cat2, cat3, dist, or td21 (not included in report).

Sample invocation from pareto_data/IEEE8500 directory:

python ....RelaySummary.py IEEE8500 IEEE8500

Public Functions:

main does the work

param ckt_name the name of the circuit for plot titles

type ckt_name str

param file_root the name of PNG and PDF plot files, if desired

type file_root str

param max_pv the percentage of PV penetration that 100 actually corresponds to. Defaults to 100

type max_pv int

returns writes a TeX row of summary values

rtype str

1.7 RunFaults.py

Run the OpenDSS event study and tabulates results. The event study consists of a series of dynamic fault simulations, defined in a local EventStudy.dss file. This file needs to define the feeder model (typically full-size, not reduced, with DER and relays. The simulation time should be long enough to ensure that all relay and fault events have settled out for each fault. It must also include a monitor for at least the fault current. Other monitors may exist for plotting, but the script runs faster if they are commented out.

The script reads a local buslist.dat to determine the fault types and locations. Each line of this file contains comma-separate values as follows:

- busname (str): the OpenDSS to fault
- phase (str): A, B, and/or C for the phases present. Three-phase, single-phase, and line-to-line faults will be applied as appropriate
- Rf (float): the maximum fault resistance, in Ohms, for the SLGF. Bolted faults are also created, currently hard-coded to 0.01 Ohms
- targets (str): a comma-separated list of selective devices that are expected to trip to clear the fault. Typically, this is one utility device and several DER devices.

The script writes scripted_fault.dss for each fault to simulate, then invokes opendsscmd on EventStudy.dss, which needs to include scripted_fault.dss.

Writes events.out with summary information.

Appends to dofaults.rpt with summary information, ending with a list of devices that didn't behave properly in at least one fault scenario. The user may focus investigation on these devices.

The summary information for each fault is:

- Bus (str): the faulted bus name
- Nphases (int): the number of faulted phases.
- Rf (float): the fault resistance in Ohms
- If (float): the fault current in kA
- Status (str): the convergence status of the OpenDSS dynamic simulation, should always be SOLVED
- Cleared (boolean): indicate whether the fault was cleared
- Tcleared (float): the fault clearing time, in s
- Nopen (int): the number of devices locked open
- Nreclosed (int): the number of utility devices that tripped and reclosed
- Nfailed (int): the number of utility devices that failed to trip
- Nfalse (int): the number of utility devices that false-tripped, e.g., sympathetic or non-directional
- Npv (int): the number of PV devices that tripped (and locked open)
- Sopen (str): a quoted comma-separated list of utility devices that locked open
- Sreclosed (str): a quoted comma-separated list of utility devices that tripped and reclosed
- Sfailed (str): a quoted comma-separated list of utility devices that failed to trip
- Sfalse (str): a quoted comma-separated list of utility devices that false-tripped
- Spv (str): a quoted comma-separated list of DER devices that tripped (and locked open)

Public Functions:

main does the work

param cktname the root name (not the file name) of the OpenDSS circuit. The script needs this to find summary and monitor outputs from each fault simulation.

type cktname str

returns writes a progress message as each fault is simulated

rtype str

1.8 RunReduction.py

Reduce full-size OpenDSS model to smaller OpenDSS and ATP models. The name of the circuit, cktname, is the first command-line argument and it must match the circuit name from the original OpenDSS model.

Reads buspairs.dat, created by the user. Each line contains two comma-separated bus names, with phasing if appropriate, for links to be retained in the reduced model. The set of all bus names referenced in this file will be retained in the reduced model. The retained buses should include breakers, reclosers, capacitors, large DER, large loads, line regulators, major junction points, and remote ends of feeder branches. The OpenDSS circuit plots can help identify the important buses to retain. In buspairs.dat, annotations may be added to the end of each line, using // to indicate such a comment.

Reads ReducedCapacitors.dss, created by the user. If there are any capacitors in the original model, they should be copied into this new file. Their buses should also be retained in buspairs.dat. They will appear explicitly in the generated ATP model.

Reads these files created by solving local ReductionStudy.dss in OpenDSS:

- bus data from cktname + '_Buses.Txt'
- element data from cktname + '_Elements.Txt'
- bus voltage and distance profile from cktname + '_EXP_Profile.CSV'
- short-circuit impedances from cktname + '_EXP_SEQZ.CSV'
- power flow solution from cktname + '_Power_elem_kVA.txt'
- feeder meter zone from cktname + '_ZoneOut_feeder.txt'

To create these files:

- copy TOCRelays.dss UtilityRelays.dss
- opendsscmd ReductionStudy.dss

Writes ReducedNetwork.dss with OpenDSS components between retained buses.

Writes ReducedXY.dss with coordinates of retained buses.

Writes ReducedNetwork.json for plotting and topology analysis. The format is documented with plot_opendss_feeder.py.

Writes ReducedNetwork.atpmap to help compare OpenDSS and ATP results on the reduced model. Each line has three space-separated tokens:

- the retained OpenDSS bus name
- the retained ATP bus name, which will be a sequential number
- A, B, and/or C to indicate the phases present at the retained bus

Writes atpfile with ATP components.

To use the reduced model in OpenDSS, the user should create a file like Reduced.dss, which includes ReducedNetwork.dss, ReducedCapacitors.dss, and ReducedXY.dss. The Reduced.dss file should also create the circuit, relays, reclosers, substation transformer, substation regulator if applicable, and an energymeter at the feeder breaker, looking out into the circuit. It should create any large DER to be studied initially; later, the user might use ScalePV.py with BasePV.json to add more DER/PV locations of different sizes. The Reduced.dss file should contain edit or batchedit commands to adjust base component settings and values to accommodate the DER and obtain a good power flow solution. The file should end with commands that calculate voltage bases and set the load multiplier.

To use the reduced model in ATP, the user might specify atpfile like ReducedNetwork.atp in a different directory than the OpenDSS files. This file might then be included from a master ATP file, which must also include the ATP solution parameters, substation model, DER model, etc. The user needs an ATP license and ATP documentation to complete these steps.

Public Functions:

main does the work

param cktname the root name of the OpenDSS circuit to be reduced

type cktname str

param atpfile the name of the ATP file to write

type atpfile str

returns writes total PV kW to console

rtype str

1.9 RunSettings.py

Finds the load current and source impedances at OpenDSS reclosers, then applies heuristics to recommend distance and incremental distance relay settings.

Do 'opendsscmd SettingStudy.dss' first. The local SettingStudy.dss file needs to define the OpenDSS model and run a power flow solution at nominal load, with no DER. Then it runs a faultstudy solution at no-load, with controls, protection and DER all disabled. Output files from both solutions are used to determine the loads, source impedance, and recommended settings for each utility device.

Reads a local recloser_buses.dat file. Each line contains a comma-separated OpenDSS bus name, and the name of an OpenDSS recloser at the bus. The script uses the source impedance at the bus, the load current through the recloser, and network topology information to recommend settings. The reclosers to coordinate are based on a networkx topology constructed from the OpenDSS energymeter zone.

The OpenDSS solution files read are the same as in RunReduction.py:

- bus data from cktname + '_Buses.Txt'
- element data from cktname + '_Elements.Txt'
- bus voltage and distance profile from cktname + '_EXP_Profile.CSV'
- short-circuit impedances from cktname + '_EXP_SEQZ.CSV'
- power flow solution from cktname + '_Power_elem_kVA.txt'
- feeder meter zone from cktname + '_ZoneOut_feeder.txt'

The analysis method is to build a networkx topology graph, with links to the power flow solution and faultstudy impedances from OpenDSS. Each recloser can determine the next upstream device by tracing back to the source.

Furthermore, each faultable bus can find its closest upstream recloser, which is expected to clear the fault, by tracing back to the source. When this fault-bus trace encounters a recloser, that recloser keeps the faultable bus with highest X1 value as the remote bus, which the recloser should see for backup purposes. These graph traces only consider devices at the same voltage level, so they don't try to coordinate through transformers. When all these traces finish, each recloser has its expected load current, and the necessary equivalent source impedances for coordination.

The output report begins with a summary of each recloser from recloser_buses.dat:

- The bus name
- The recloser name
- The full-load current [kA] and nominal voltage [kV] at the bus
- The positive and zero-sequence equivalent source impedances [Ohms] at this recloser, called Zs1 and Zs0
- The positive and zero-sequence equivalent source impedances [Ohms] at the next downstream recloser, called Zn1 and Zn0
- The positive and zero-sequence equivalent source impedances [Ohms] at a remote downstream bus, called Zr1 and Zr0
- Nzone, the number of faultable buses in this recloser's primary protection zone. May include first bus in the next zone.
- NextRec, the name of the nearest downstream recloser considered for coordination. There may be other downstream reclosers, farther away electrically.
- RemoteBus, the name of the remote bus considered in reach setting

The next two output tables contain Zone 1 and Zone 2 impedance settings, positive and negative sequence, recommended for each recloser. The first of these tables is in rectangular coordinates, the second in polar coordinates. Zone 1 is the impedance difference between this recloser and the next downstream. Zone 2 is the impedance difference between this recloser and the remote bus. However, these are full-reach settings. To use them, Zone 1 should be used with a multiplier of 0.8 to 0.9, while Zone 2 should be increased with a multiplier of at least 1.25 to ensure full coverage. Furthermore, in OpenDSS each Zone would be implemented with a separate recloser device, attached at the same location.

The last output table contains the settings used in this project. It is based on a single Zone, full reach to the remote bus or nearest downstream recloser, whichever is farthest, and a minimum pickup of 1.2 times full load current. The settings are formatted in the syntax for OpenDSS relay parameters. To use these settings, coordination should be achieved with define time delays, where the time delays decrease with the recloser's distance from the substation.

Public Functions:

main does the work

param cktname the root name (not the file name) of the OpenDSS circuit, used to find outputs from SettingStudy.dss

type cktname str

returns writes progress messages to console, ending with the settings report

rtype str

1.10 ScalePV.py

Change sizes of PV, assign IEEE 1547 category.

Reads the list of PV installations from local BasePV.json file.

Writes each PV as a VCCS, resizes the interconnection transformer, and writes undervoltage relay with Category I, II, or III settings. Output to local protected_pv.dss

Writes the distance relay for each PV to local DistanceRelaysPV.dss, to be included from local DistanceRelays.dss that has the utility distance relays. Writes the incremental distance relay for each PV to local TD21RelaysPV.dss, to be included from local TD21Relays.dss that has the utility TD21 relays. The relay settings are based on rated current and base impedance. The impedance setting, looking out into the grid, is 0.9 times the base impedance at an angle of 10 degrees. The minimum current to trip is 1.1 times rated current. The time to trip is fixed at 0.5 s delay plus 0.032 s breaker time. The reset time is fixed at 5.0 s. There is one trip to lockout. Phase and ground settings are the same.

BasePV.json contains an array of VCCS elements, as follows:

- Name (str): the name of the existing pvsystem to replace with VCCS
- Bus (str): the name of the existing bus where pvsystem is connected
- Xfmr (str): the name of the existing interconnection transformer
- Prated (float): the nominal pv rating in Watts, doesn't have to match existing pvsystem
- Vrated (float): the nominal line-to-line (for three-phase) or line-to-neutral voltage, in Volts
- Phases (int): choose 3 or 1

Public Functions:

main does the work

param arg1 scale factor applied to BasePV.json

type arg1 float

param arg2 IEEE 1547 disturbance category, 1..3

type arg2 int

returns writes total PV kW to console

rtype str

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

d

INDEX

D

M