Optimization with PYPOWER

You can run an Optimal Power Flow using the PYPOWER OPF

AC OPF

`pandapower.runopp`*(net, verbose=False, calculate_voltage_angles=True, check_connectivity=True, suppress_warnings=True, switch_rx_ratio=2, delta=1e-10, init='flat', numba=True, trafo3w_losses='hv', consider_line_temperature=False, **kwargs)*

Runs the pandapower Optimal Power Flow. Flexibilities, constraints and cost parameters are defined in the pandapower element tables.

Flexibilities can be defined in net.sgen / net.gen /net.load / net.storage /net.ext_grid net.sgen.controllable if a static generator is controllable. If False, the active and reactive power are assigned as in a normal power flow. If True, the following flexibilities apply:

- net.gen.min_p_mw / net.gen.max_p_mw
- net.gen.min_q_mvar / net.gen.max_q_mvar
- net.sgen.min_p_mw / net.sgen.max_p_mw
- net.sgen.min_q_mvar / net.sgen.max_q_mvar
- net.dcline.max_p_mw
- net.dcline.min_q_to_mvar / net.dcline.max_q_to_mvar / net.dcline.min_q_from_mvar / net.dcline.max_q_from_mvar
- net.ext_grid.min_p_mw / net.ext_grid.max_p_mw
- net.ext_grid.min_q_mvar / net.ext_grid.max_q_mvar
- net.load.min_p_mw / net.load.max_p_mw
- net.load.min_q_mvar / net.load.max_q_mvar
- net.storage.min_p_mw / net.storage.max_p_mw
- net.storage.min_q_mvar / net.storage.max_q_mvar

Controllable loads behave just like controllable static generators. It must be stated if they are controllable. Otherwise, they are not respected as flexibilities. Dc lines are controllable per default

> **Network constraints can be defined for buses, lines and transformers the elements in the following columns:**

- net.bus.min_vm_pu / net.bus.max_vm_pu
- net.line.max_loading_percent
- net.trafo.max_loading_percent
- net.trafo3w.max_loading_percent

If the external grid ist controllable, the voltage setpoint of the external grid can be optimized within the

voltage constraints by the OPF. The same applies to the voltage setpoints of the controllable generator elements.

How these costs are combined into a cost function depends on the cost_function parameter.

INPUT:

net - The pandapower format network

OPTIONAL:

verbose (bool, False) - If True, some basic information is printed

suppress_warnings (bool, True) - suppress warnings in pypower

> If set to True, warnings are disabled during the loadflow. Because of the way data is processed in pypower, ComplexWarnings are raised during the loadflow. These warnings are suppressed by this option, however keep in mind all other pypower warnings are suppressed, too.

init (str, “flat”) - init of starting opf vector. Options are “flat” or “pf”

> Starting solution vector (x0) for opf calculations is determined by this flag. Options are: “flat” (default): starting vector is (upper bound - lower bound) / 2 “pf”: a power flow is executed prior to the opf and the pf solution is the starting vector. This may improve convergence, but takes a longer runtime (which are probably neglectible for opf calculations)

delta (float, 1e-10) - power tolerance

trafo3w_losses (str, “hv”) - defines where open loop losses of three-winding transformers are considered. Valid options are “hv”, “mv”, “lv” for HV/MV/LV side or “star” for the star point.

consider_line_temperature (bool, False) - adjustment of line impedance based on provided line temperature. If True, net.line must contain a column “temperature_degree_celsius”. The temperature dependency coefficient alpha must be provided in the net.line.alpha column, otherwise the default value of 0.004 is used

kwargs - Pypower / Matpower keyword arguments:

- OPF_VIOLATION (5e-6) constraint violation tolerance
- PDIPM_COSTTOL (1e-6) optimality tolerance
- PDIPM_GRADTOL (1e-6) gradient tolerance
- PDIPM_COMPTOL (1e-6) complementarity condition (inequality) tolerance
- PDIPM_FEASTOL (set to OPF_VIOLATION if not specified) feasibiliy (equality) tolerance
- PDIPM_MAX_IT (150) maximum number of iterations
- SCPDIPM_RED_IT(20) maximum number of step size reductions per iteration

The internal solver uses the interior point method. By default, the initial state is the center of the operational constraints. Another option would be to initialize the optimisation with a valid loadflow solution. For optimiation of a timeseries, this warm start possibilty could imply a significant speedup. This is not yet provided in the actual version, but could be an useful extension in the future. Another parametrisation for the AC OPF is, if voltage angles should be considered, which is the same option than for the loadflow calculation with pandapower.runpp:

References:

- "On the Computation and Application of Multi-period Security-Constrained Optimal Power Flow for Real-time Electricity Market Operations", Cornell University, May 2007.
- H. Wang, C. E. Murillo-Sanchez, R. D. Zimmerman, R. J. Thomas, "On Computational Issues of Market-Based Optimal Power Flow", IEEE Transactions on Power Systems, Vol. 22, No. 3, Aug. 2007, pp. 1185-1193.
- R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-State Operations, Planning and Analysis Tools for Power Systems Research and Education," Power Systems, IEEE Transactions on, vol. 26, no. 1, pp. 12-19, Feb. 2011.

DC OPF

The dc optimal power flow is a linearized optimization of the grid state. It offers two cost function options, that are fitting special use cases. To understand the usage, the DC OPF tutorial is recommended.

`pandapower.rundcopp`*(net, verbose=False, check_connectivity=True, suppress_warnings=True, switch_rx_ratio=0.5, delta=1e-10, trafo3w_losses='hv', **kwargs)*

Runs the pandapower Optimal Power Flow. Flexibilities, constraints and cost parameters are defined in the pandapower element tables.

Flexibilities for generators can be defined in net.sgen / net.gen. net.sgen.controllable / net.gen.controllable signals if a generator is controllable. If False, the active and reactive power are assigned as in a normal power flow. If yes, the following flexibilities apply:

- net.sgen.min_p_mw / net.sgen.max_p_mw
- net.gen.min_p_mw / net.gen.max_p_mw
- net.load.min_p_mw / net.load.max_p_mw

Network constraints can be defined for buses, lines and transformers the elements in the following columns: - net.line.max_loading_percent - net.trafo.max_loading_percent - net.trafo3w.max_loading_percent

INPUT:

net - The pandapower format network

OPTIONAL:

verbose (bool, False) - If True, some basic information is printed

suppress_warnings (bool, True) - suppress warnings in pypower

> If set to True, warnings are disabled during the loadflow. Because of the way data is processed in pypower, ComplexWarnings are raised during the loadflow. These warnings are suppressed by this option, however keep in mind all other pypower warnings are suppressed, too.

delta (float, 1e-10) - power tolerance

trafo3w_losses (str, “hv”) - defines where open loop losses of three-winding transformers are considered. Valid options are “hv”, “mv”, “lv” for HV/MV/LV side or “star” for the star point.

Flexibilities, costs and constraints (except voltage constraints) are handled as in the Optimisation problem. Voltage constraints are not considered in the DC OPF, since voltage magnitutes are not part of the linearized power flow equations.

Note

If you are interested in the pypower casefile that pandapower is using for power flow, you can find it in net[“_ppc_opf”]. However all necessary informations are written into the pandpower format net, so the pandapower user should not usually have to deal with pypower.