

Usage

PYPOWER provides a Command Line Interface (CLI) and a Python Application Programming Interface (API).

Command Line Interface

Following the *Installation* instructions adds `pf` and `opf` to the command path. To print usage info type:

```
$ pf -h
```

All available options will be printed:

```
Usage: pf [options] [casedata]
```

Runs a power flow.

If 'casedata' is provided it specifies the name of the input data file containing the case data.

Options:

```
--version          show program's version number and exit
-h, --help         show this help message and exit
-t, --test         run tests
-c TESTCASE, --testcase=TESTCASE
                   built-in test case (choose from: 'case30_userfcns',
                   'case118', 'case9', 'case30Q', 'case30pwl', 'case6ww',
                   'case57', 'case39', 'case14', 'case9Q', 'case30',
                   'case300', 'case4gs', 'case24_ieee_rts')
-o FNAME, --outfile=FNAME
                   pretty printed output will be appended to a file with
                   the name specified. Defaults to stdout.
-s SOLVEDCASE, --solvedcase=SOLVEDCASE
                   the solved case will be written to a case file with
                   the specified name in PYPOWER format. If solvedcase
                   ends with '.mat' the case is saved as a MAT-file
                   otherwise it saves it as a Python file.
```

Power Flow Options:

```
--pf_alg=PF_ALG    power flow algorithm: 1 - Newton's method, 2 - Fast-
                   Decoupled (XB version), 3 - Fast-Decoupled (BX
                   version), 4 - Gauss Seidel [default: 1]
--pf_tol=PF_TOL     termination tolerance on per unit P & Q mismatch
                   [default: 1e-08]
--pf_max_it=PF_MAX_IT
                   maximum number of iterations for Newton's method
                   [default: 10]
--pf_max_it_fd=PF_MAX_IT_FD
                   maximum number of iterations for fast decoupled method
                   [default: 30]
--pf_max_it_gs=PF_MAX_IT_GS
                   maximum number of iterations for Gauss-Seidel method
                   [default: 1000]
--enforce_q_lims=ENFORCE_Q_LIMS
                   enforce gen reactive power limits, at expense of |V|
                   [default: False]
--pf_dc=PF_DC       use DC power flow formulation, for power flow and OPF:
                   False - use AC formulation & corresponding algorithm
                   opts, True - use DC formulation, ignore AC algorithm
                   options [default: False]
```

Output Options:

```

--verbose=VERBOSE    amount of progress info printed: 0 - print no progress
                      info, 1 - print a little progress info, 2 - print a
                      lot of progress info, 3 - print all progress info
                      [default: 1]
--out_all=OUT_ALL     controls printing of results: -1 - individual flags
                      control what prints, 0 - don't print anything
                      (overrides individual flags, except OUT_RAW), 1 -
                      print everything (overrides individual flags,
                      except OUT_RAW) [default: -1]
--out_sys_sum=OUT_SYS_SUM
                      print system summary [default: True]
--out_area_sum=OUT_AREA_SUM
                      print area summaries [default: False]
--out_bus=OUT_BUS     print bus detail [default: True]
--out_branch=OUT_BRANCH
                      print branch detail [default: True]
--out_gen=OUT_GEN     print generator detail (OUT_BUS also includes gen
                      info) [default: False]
--out_all_lim=OUT_ALL_LIM
                      control constraint info output: -1 - individual flags
                      control what constraint info prints, 0 - no constraint
                      info (overrides individual flags), 1 - binding
                      constraint info (overrides individual flags), 2 - all
                      constraint info (overrides individual flags) [default:
                      -1]
--out_v_lim=OUT_V_LIM
                      control output of voltage limit info: 0 - don't print,
                      1 - print binding constraints only, 2 - print all
                      constraints (same options for OUT_LINE_LIM,
                      OUT_PG_LIM, OUT_QG_LIM) [default: 1]
--out_line_lim=OUT_LINE_LIM
                      control output of line limit info [default: 1]
--out_pg_lim=OUT_PG_LIM
                      control output of gen P limit info [default: 1]
--out_qg_lim=OUT_QG_LIM
                      control output of gen Q limit info [default: 1]
--out_raw=OUT_RAW     print raw data [default: False]
--return_raw_der=RETURN_RAW_DER
                      return constraint and derivative info in
                      results['raw'] (in keys g, dg, df, d2f)) [default: 0]

```

PYPOWER includes a selection of test cases. For example, to run a power flow on the [IEEE 14 bus](#) test case:

```
$ pf -c case14
```

Alternatively, the path to a [PYPOWER case data](#) file can be specified:

```
$ pf /path/to/case14.py
```

The `opf` command has the same calling syntax. For example, to solve an OPF for the [IEEE Reliability Test System](#) and write the solved case to file:

```
$ opf -c case24_ieee_rts --solvedcase=rtsout.py
```

Application Programming Interface

The Python API for PYPOWER can be accessed using the `pypower.api` package:

```
In [1]: from pypower.api import case9, ppoption, runpf, printpf
```

To load the 9 bus test case, solve an AC power flow using the fast-decoupled method and print the results:

```
In [2]: ppc = case9()  
In [3]: ppopt = ppoption(PF_ALG=2)  
In [4]: r = runpf(ppc, ppopt)  
In [5]: printpf(r)
```

For additional information refer to the Python documentation for each of the functions. E.g:

```
In [6]: help runpf
```

Alternatively, refer to the on-line [API documentation](https://rw1.github.io/PYPOWER/usage.html#application-programming-interface).