SDP_PF: Implementation of Applications of a Semidefinite Programming Relaxation of the Power Flow Equations

Version 1.0

Daniel K. Molzahn*

December 17, 2014

Abstract

This report documents the implementation of applications for a semidefinite programming relaxation of the power flow equations. The code integrates with the software package MATPOWER. When a rank condition is satisfied, the semidefinite relaxation yields the global solution to the optimal power flow (OPF) problem in polynomial time. Through the use of matrix decomposition techniques, practically sized systems are computationally tractable. Additional functionality includes implementation of a sufficient condition for global optimality of an OPF solution, sufficient conditions for power flow insolvability including the ability to respect generator reactive power limits, and calculation of voltage stability margins.

*University of Michigan Department of Electrical Engineering and Computer Science. molzahn@umich.edu

Publications

The function `sdpopf_solver` implements SDPOPF, an optimal power flow solver based on semidefinite programming. A detailed description of the theory incorporated in this code is available in reference [1] and the references therein. Please cite reference [1] in work that uses SDPOPF.

[1] D.K. Molzahn, J.T. Holzer, B.C. Lesieutre, and C.L. DeMarco, "Implementation of a Large-Scale Optimal Power Flow Solver Based on Semidefinite Programming," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 3987-3998, November 2013.

Also included with this code is the function `testGlobalOpt`, which implements a sufficient condition for global optimality of a given OPF solution. Please cite reference [2] in work that uses `testGlobalOpt`.

[2] D.K. Molzahn, B.C. Lesieutre, and C.L. DeMarco, "A Sufficient Condition for Global Optimality of Solutions to the Optimal Power Flow Problem," *IEEE Transactions on Power Systems (Letters)*, vol. 29, no. 2, pp. 978-979, March 2014.

The function `insolvablepf` implements a sufficient condition for the insolvability of a specified set of power flow equations. Please cite reference [3] in work that uses `insolvablepf`.

[3] D.K. Molzahn, B.C. Lesieutre, and C.L. DeMarco, "A Sufficient Condition for Power Flow Insolvability with Applications to Voltage Stability Margins," *IEEE Transactions on Power Systems*, vol. 28, no. 3, pp. 2592-2601, August 2013.

The function `insolvablepf` does not incorporate generator reactive power limits. A mixed-integer semidefinite programming formulation implemented by the function `insolvablepf_limitQ` incorporates generator reactive power limits. Please cite reference [4] in work that uses `insolvablepf_limitQ`.

An alternative approach to proving power flow insolvability uses the Positivstellensatz theorem from the field of real algebraic geometry and sum of squares programming to generate infeasibility certificates. The function `insolvablepfsos` implements the proposed method for the power flow equations without considering reactive power limited generators. The function `insolvablepfsos_limitQ` implements the proposed method for the power flow equations with consideration of upper limits on generators' reactive power outputs. Please cite reference [4] in work that uses `insolvablepfsos` or `insolvablepfsos_limitQ`.

[4] D.K. Molzahn, V. Dawar, B.C. Lesieutre, and C.L. DeMarco, "Sufficient Conditions for Power Flow Insolvability Considering Reactive Power Limited Generators with Applications to Voltage Stability Margins,"*Bulk Power System Dynamics and Control - IX. Optimization, Security and Control of the Emerging Power Grid, 2013 IREP Symposium*, 25-30 August 2013.

1 Introduction

The optimal power flow (OPF) problem seeks decision variable values that yield an optimal operating point for an electric power system in terms of a specified objective function, subject to both network equality constraints (i.e., the power flow equations) and engineering inequality constraints (e.g., limits on voltage magnitudes, active and reactive power generations, and flows on transmission lines and transformers).

The OPF problem is nonconvex due to the nonlinear power flow equations [5], may have local solutions [6], and is, in general, NP-hard [7]. Nonconvexity of the OPF problem has made solution techniques an ongoing research topic. Many OPF solution techniques have been proposed, including successive quadratic programs, Lagrangian relaxation, genetic algorithms, particle swarm optimization, and interior point methods (for relevant survey papers, see [8–12]).

Recently, significant research attention has focused on a semidefinite programming relaxation of the optimal power flow problem [7, 13]. Semidefinite programming is a type of convex optimization that minimizes a linear objective function over the intersection of a cone of positive semidefinite matrices (i.e., symmetric matrices constrained to have all non-negative eigenvalues) and an affine plane. Semidefinite programming has been successful in solving or approximating the solutions of many practical problems that are otherwise computationally challenging, including NP-hard optimization problems. Overviews of semidefinite programming theory and practice are available in references [14–16].

By appropriate selection of $\mathbf{A}_i$ matrices, the power flow equations in rectangular voltage coordinates can be written in the form $x^T \mathbf{A}_i x = c_i$, where x is a vector of orthogonal voltage components.

$$x = \begin{bmatrix} V_{d1} & V_{d2} & \dots & V_{dn} & V_{q1} & V_{q2} & \dots & V_{qn} \end{bmatrix}^T \tag{1}$$

By defining the matrix $\mathbf{W} = xx^T$, the power flow equations can then be rewritten as the combination of the linear equations $\text{trace}\,(\mathbf{A}_i \mathbf{W}) = c_i$ and the condition $\text{rank}\,(\mathbf{W}) = 1$.

The nonconvexity in this formulation of the power flow equations is entirely due to the rank condition. The semidefinite relaxation of the power flow equations does not enforce the rank condition; rather, the constraint $\mathbf{W} \succeq 0$, where the symbol $\succeq$ indicates that the corresponding matrix is positive semidefinite, is used to define a convex feasible space. If the solution to the resulting semidefinite relaxation satisfies the rank condition (i.e., the semidefinite relaxation is "exact"), a voltage profile satisfying the power flow equations can be obtained from an eigenvector associated with a non-zero eigenvalue of the solution's $\mathbf{W}$ matrix [7]. The voltage profile obtained from a semidefinite programming relaxation of the optimal power flow is guaranteed to be the globally optimal solution if the rank condition is satisfied. No prior OPF solution method offers a guarantee of finding the global solution; semidefinite programming approaches thus have a substantial advantage over traditional solution techniques.

Existing literature [7] claims that the rank condition is satisfied for most practical power system models, including the IEEE test systems [17]. However, the rank condition

is not always satisfied, which means that semidefinite relaxations do not give physically meaningful solutions for all realistic power system models [18, 19]. Recent research has investigated the conditions under which the rank condition is satisfied; to date, sufficient conditions for rank condition satisfaction include requirements on power injection and voltage magnitude limits and either radial networks (typical of distribution system models) or or unrealistically dense placement of controllable phase shifting transformers [20–23].

In addition to theoretical concerns regarding satisfaction of the rank condition, practical computational issues are also of interest. Semidefinite programming relaxations of the OPF problem constrain a $2n \times 2n$ symmetric matrix to be positive semidefinite, where n is the number buses in the system. The semidefinite program size thus grows as the square of the number of buses, which makes solution of OPF problems by semidefinite programming computationally intractable for large systems. Recent work using matrix completion [24–26] reduces the computational burden inherent in solving large systems by taking advantage of the sparse matrix structure created by realistic power system models. Sojoudi and Lavaei [22], Bai and Wei [27], and Jabr [28] present formulations that decompose the single large $2n \times 2n$ positive semidefinite matrix constraint into positive semidefinite constraints on many smaller matrices. If the matrices from these decompositions satisfy a rank condition, the $2n \times 2n$ matrix also satisfies the rank condition and the globally optimal solution can be obtained.

This code incorporates several computational and modeling advances in applying the semidefinite relaxation to practical power systems. Modeling advances include allowing multiple generators at the same bus and incorporating flow limits on parallel lines. Computational advances include enhancements to the existing matrix decomposition algorithms. A first enhancement is a heuristic algorithm for combining some of the many small matrices resulting from the decomposition. Since linking constraints are required between terms of the decomposed matrices that refer to the same term in the $2n \times 2n$ matrix, it is not always advantageous to create the smallest possible matrices. Combining matrices eliminates some of these linking constraints, which results in a factor of two to three improvement in computational speed over existing matrix decomposition algorithms [1].

A second enhancement presented is a technique for recovering the optimal voltage profile from the decomposed matrices. None of the existing literature describes a method for actually obtaining the optimal voltage profile from a solution to a decomposed formulation.

A final enhancement modifies the maximal clique decomposition as formulated by Jabr [28] to allow for application to general power systems. This formulation uses a Cholesky factorization of the absolute value of the imaginary part of the bus admittance matrix to form a chordal extension of the network. However, this matrix may not be positive definite (for instance, in networks with sufficiently large shunt capacitances), thus preventing formation of a Cholesky factorization. This code uses an alternative matrix that is always positive definite and gives an equivalent chordal extension in order to extend this decomposition for general networks.

Additionally, this code includes the function `testGlobalOpt`, which uses the Karush–Kuhn–Tucker (KKT) conditions of the semidefinite relaxation of the OPF problem to create a sufficient condition for global optimality of an OPF solution. If an OPF solution

from any solver (e.g., the interior point solvers in MATPOWER [12]) satisfies the sufficient condition, the solution is guaranteed to be globally optimal.

Finally, this code includes several functions that implement sufficient conditions for power flow insolvability. The function `insolvablepf` gives a sufficient condition for insolvability of the power flow equations (neglecting generator reactive power limits). This function uses the semidefinite programming relaxation of the power flow equations to minimize the slack bus voltage with proportional changes in PV bus voltages. If the minimum achievable slack bus voltage in the semidefinite relaxation is greater than the specified slack bus voltage, the power flow equations are insolvable. Conversely, if the minimum slack bus voltage is less than or equal to the specified slack bus voltage, solvability of the power flow equations is indeterminate. This function also provides a controlled voltage margin and a power injection margin to the power flow solvability boundary. See [3] for further details.

The function `insolvablepf_limitQ` gives a sufficient condition for insolvability of the power flow equations considering generator reactive power limits. This function uses a mixed-integer semidefinite programming formulation to maximize the power injections in a uniform, constant power factor profile. If the factor by which the power injections can be changed is less than one, no power flow solution exists that satisfies the generator reactive power limits. This function also gives a power injection margin as a measure of the distance to the power flow solvability boundary considering generator reactive power limits. See [4] for further details.

Using the fact that the power flow equations are polynomials in the rectangular voltage components V_d and V_q, the function `insolvablepfsos` applies the Positivstellensatz theorem from the field of real algebraic geometry and sum of squares programming to generate infeasibility certificates. An infeasibility certificate proves that the power flow equations are insolvable. Using a similar approach, the function `insolvablepfsos_limitQ` considers upper limits on reactive power generation through a polynomial formulation of these limits. See [4] for further details.

2 Dependencies

This code is implemented as an add-on to the power systems software package MATPOWER[1] [12]. MATPOWER is a package of MATLAB M-files that solve the power flow and optimal power flow problems.

Other required software packages include the optimization modeling tool YALMIP[2] [29] and a semidefinite programming solver compatible with YALMIP, such as SeDuMi[3] [30] or SDPT3[4] [31]. (Any YALMIP compatible solver should work, but this code was tested most thoroughly with SeDuMi version 1.3 and SDPT3 version 4.0.)

[1] MATPOWER is available at `http://www.pserc.cornell.edu/matpower/`

[2] YALMIP is available at `http://users.isy.liu.se/johanl/yalmip/`

[3] SeDuMi is available at `http://sedumi.ie.lehigh.edu/`

[4] SDPT3 is available at `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`

3 Installation

1. Install YALMIP.
2. Install a semidefinite programming solver compatible with YALMIP, such as SeDuMi or SDPT3. Successful installation of YALMIP and a semidefinite programming solver can be tested using the command `yalmiptest`; the solver is correctly installed if the "SDP" test indicates a correct solution.
3. Install MATPOWER. Ensure that the folder containing the MATPOWER files is on the MATLAB path.
4. Unzip the files in `sdp_pf.zip` into a folder on the MATLAB path.
5. Solve an optimal power flow problem using the standard MATPOWER commands with the option `opf.ac.solver` set to `'SDPOPF'` or the new `runsdpopf` command, e.g., `runsdpopf('case9')`.

4 Outputs

The optimal power flow solver `sdpopf_solver` outputs three variables: `results`, `success`, and `raw`. The output `results` is a MATPOWER case structure (`mpc`) with the usual `baseMVA`, `bus`, `branch`, `gen`, and `gencost` fields. There are three additional fields in this structure:

- `f` : The final objective function value. This is a lower bound on the the system operating cost and equals the minimum operating cost if the rank condition is satisfied.
- `mineigratio` : A measure of the satisfaction of the rank condition.

  The rank condition is satisfied for the dual problem if the dimension of the nullspaces of all decomposed **A** matrices are less than or equal to two. If the optimal voltage profile is obtained from the dual problem, `mineigratio` is the minimum ratio between the third smallest and second smallest eigenvalues (i.e., between an eigenvalue that should not be zero and one that should be zero when the rank condition is satisfied) among all matrices corresponding to the maximal cliques.

  The rank condition is satisfied for the primal problem if the ranks of all decomposed **W** matrices are less than or equal to two. If the optimal voltage profile is obtained from the primal problem, `mineigratio` is the minimum ratio between the second largest and third largest eigenvalues (i.e., between an eigenvalue that should not be zero and one that should be zero when the rank condition is satisfied) among all matrices corresponding to the maximal cliques.
- `zero_eval` : A measure of solution consistency in recovering the optimal voltage profile when matrix decomposition is used.

The optimal voltage profile is recovered from the semidefinite programming solution using a vector in the nullspace of a matrix (see [1] for more details). This matrix has a non-empty nullspace (i.e., it has a zero eigenvalue) for semidefinite program solutions that satisfy the rank condition. The output `zero_eval` is the smallest magnitude eigenvalue of this matrix. If `zero_eval` is non-zero, the optimal voltage profile can not be constructed in a consistent manner from the decomposed matrices.

The output `success` is a binary variable indicating whether SDPOPF found a solution that satisfies the rank condition. The rank condition is considered satisfied (and thus `success` set to 1) when `mineigratio` is greater than the option `sdp_pf.mineigratio_tol` and the magnitude of `zero_eval` is less than the option `sdp_pf.zeroeval_tol`.

The output `raw` contains the variables returned by the SDP solver in the following fields.

- `xr` : Final value of optimization variables
  - `A` : Cell array of matrix variables for the dual problem
  - `W` : Cell array of matrix variables for the primal problem
- `pimul` : Constraint Lagrange multipliers on
  - `lambdaeq_sdp` : Active power equalities
  - `psiU_sdp` : Generator active power upper inequalities
  - `psiL_sdp` : Generator active power lower inequalities
  - `gammaeq_sdp` : Reactive power equalities
  - `gammaU_sdp` : Reactive power upper inequalities
  - `gammaL_sdp` : Reactive power lower inequalities
  - `muU_sdp` : Square of voltage magnitude upper inequalities
  - `muL_sdp` : Square of voltage magnitude lower inequalities
- `info` : solver specific termination code

5 Solver Options

SDPOPF has several options that control the semidefinite programming relaxation of the optimal power flow problem.

Option	Default	Description

`yalmip.opts`	*empty*	Specifies a struct of YALMIP solver options passed to `yalmip_options` to override defaults, applied after overrides from `yalmip.opt_fname`. The YALMIP options struct specifies the desired solver and solver parameters. For example, the following code sets the solver to SeDuMi with a tolerance parameter `eps` of 1×10^{-9}.
		
		`mpopts = mpoption;`
		`mpopts.yalmip.opts = sdpsettings;`
		`mpopts.yalmip.opts.solver = 'sedumi';`
		`mpopts.yalmip.opts.sedumi.eps = 0;`
		
		The following code sets the solver to SDPT3 with tolerance parameters equal to 1×10^{-9}.
		
		`mpopts = mpoption;`
		`mpopts.yalmip.opts = sdpsettings;`
		`mpopts.yalmip.opts.solver = 'sdpt3';`
		`mpopts.yalmip.opts.sdpt3.gaptol = 1e-9;`
		`mpopts.yalmip.opts.sdpt3.inftol = 1e-9;`
		`mpopts.yalmip.opts.sdpt3.steptol = 1e-9;`
		
		See `help YALMIP_OPTIONS` for further details.

`yalmip.opt_fname`	`''`	Specifies the name of a user-supplied function passed as the `FNAME` argument to `yalmip_options` to override the defaults. See `help YALMIP_OPTIONS` for further details.
`sdp_pf.max_number_of_cliques`	0.1	Maximum number of maximal cliques (stopping criterion for maximal clique combination). • 0 : No maximum • (0, 1) : Specified proportion of the original number of maximal cliques • ≥ 1 : Specified number of maximal cliques

`sdp_pf.eps_r`	1×10^{-4}	Minimum branch series resistance. The semidefinite relaxation requires a non-zero series resistance on each branch. This parameter specifies a floor on this resistance.
`sdp_pf.recover_voltage`	4	Specifies the method for recovering the voltage from the semidefinite program solution. • 1 : From the dual problem (**A** matrices) • 2 : From the primal problem (**W** matrices) • 3 : Whichever of the primal or dual problem has the smallest difference between the specified power injections at PQ buses and the PQ injections calculated from the resulting voltage profile, measured with a 2-norm. • 4 : Whichever of the primal or dual problem has the largest minimum eigenvalue ratio, indicating the closest to satisfying the rank condition.
`sdp_pf.recover_injections`	2	Specifies the method for recovering the active and reactive power injections and the line flows from the semidefinite program solution. • 1 : Calculate from the voltage profile. • 2 : Recover from the primal problem (**W** matrices).
`sdp_pf.min_Pgen_diff`	1	Large power system models with tight inequality constraints may cause numerical problems in the semidefinite program solver. To avoid these problems, specify a minimum tolerance in MW for active power generation inequality constraints. If the difference between a generator's upper and lower active power generation bounds is smaller than this tolerance, the active power generation is fixed to the midpoint of the generation range.
`sdp_pf.min_Qgen_diff`	1	See `sdp_pf.min_Pgen_diff`; this setting controls a similar minimum generation tolerance in MVAr for reactive power inequality constraints.
`sdp_pf.max_line_limit`	9900	Line flow limits greater than this value are considered unlimited.

`sdp_pf.max_gen_limit`	99998	Generators with reactive power generation limits greater than this value are considered unlimited.
`sdp_pf.ndisplay`	1	When `verbose` is greater than 1, diagnostic and timing information will display as SDPOPF executes. `sdp_pf.ndisplay` controls how frequently this diagnostic information is displayed.
`sdp_pf.choldense`	10	Controls the density parameter of the minimum degree ordering function `amd` in the Cholesky decomposition used to create a chordal extension of the power system graph. See the MATLAB function `amd` for further details.
`sdp_pf.cholaggressive`	1	Controls the aggressive parameter of the minimum degree ordering function `amd` in the Cholesky decomposition used to create a chordal extension of the power system graph. See the MATLAB function `amd` for further details.
`sdp_pf.bind_lagrange`	1×10^{-3}	A binding constraint is required to extract the optimal voltage profile from the solution to the semidefinite programming relaxation. Binding constraints are identified by non-zero corresponding Lagrange multipliers. A Lagrange multiplier is considered non-zero, and therefore binding, if its magnitude is greater than `sdp_pf.bind_lagrange`.
`sdp_pf.zeroeval_tol`	1×10^{-5}	Maximum magnitude of `zero_eval` to consider SDPOPF successful. Also used in `testGlobalOpt` for satisfaction tolerances for the complementarity and positive semidefinite feasibility conditions for global optimality of a candidate OPF solution.
`sdp_pf.mineigratio_tol`	1×10^{5}	Minimum value of `mineigratio` to consider SDPOPF successful.

6 Functions

Several new functions were required to implement SDPOPF.

- `[RESULTS,SUCCESS,RAW] = SDPOPF_SOLVER(OM, MPOPT)`

  This is the main SDPOPF function. It implements a semidefinite programming relaxation of the power flow equations with matrix decomposition techniques for exploiting power system sparsity [1].

- `[RESULTS, SUCCESS] = RUNSDPOPF(CASEDATA, MPOPT, FNAME, SOLVEDCASE)`

  This is a function for calling SDPOPF in MATPOWER.

- `[SDPOPT] = YALMIP_OPTIONS(OVERRIDES, FNAME)`
  `[SDPOPT] = YALMIP_OPTIONS(OVERRIDES, MPOPT)`

  This function is used to set the YALMIP solver options (i.e., a `SDPSETTINGS` variable). Overrides can be provided directly as a struct in the first argument, as a user-supplied function name `FNAME` in the second argument, or as a struct or function name in options `yalmip.opts` or `yalmip.opt_fname`, respectively, of the MATPOWER options struct `MPOPT`. See `help YALMIP_OPTIONS` for further details.

- `[MAXCLIQUES,E] = COMBINEMAXCLIQUES(MAXCLIQUES, E, MAXNUMBEROFCLIQUES, NDISPLAY)`

  This function combines the maximal cliques to reduce computation time. This function uses the heuristic that an additional variable in a matrix is equivalent to an additional linking constraint between overlapping maximal cliques. See [1] for more details.

- `[QUANTITY] = RECOVERFROMW(SDPMAT, WREF_DD, WREF_QQ, WREF_DQ, MATIDX_DD, MATIDX_QQ, MATIDX_DQ, W, SDPVAR, MAXCLIQUE)`

  This function calculates `trace(sdpmat*W)` for the decomposed matrices. This function is used for recovering power injections and line flows from the primal problem when the option `sdp_pf.recover_injections` is set to 2.

- `[SDPVEC] = MAT2VEC(SDPMAT, WREF_DD, WREF_QQ, WREF_DQ, MATIDX_DD, MATIDX_QQ, MATIDX_DQ)`

  This function is used in the formation of the matrix completion decomposition for the semidefinite programming relaxation of the optimal power flow. It converts a $2n \times 2n$ symmetric matrix `sdpmat` into a list form. For each nonzero element of `sdpmat`, the list form in `sdpvec` gives the appropriate decomposed matrix, the location in the matrix, and the value for that element.

- `[A] = ADDTOA(SDPMAT, WREF_DD, WREF_QQ, WREF_DQ, MATIDX_DD, MATIDX_QQ, MATIDX_DQ, A, SDPVAR, MAXCLIQUE)`

  This function adds a matrix multiplied by a SDP variable to the $\mathbf{A}$ matrices in the dual problem.

- `[E] = PRIM(AADJ)`

  This is an implementation of Prim's algorithm for calculating a minimum spanning tree from a graph adjacency matrix. This implementation can incorporate negative edge weights. A maximal spanning tree can be created by specifying the negative of the graph adjacency matrix.

- `[MAXCLIQUE,ISCHORDAL] = MAXCARDSEARCH(AADJ)`

  This function determines the maximal cliques for a chordal graph described by the adjacency matrix `Aadj`. It also tests the graph for chordality. The algorithms for determining the maximal cliques and testing for chordality are described in [32].

- `[COST] = COMBINECOST(MAXCLIQUES, MAXCLIQUEIDX)`

  This function calculates the cost of combining two maximal cliques in terms of the number of scalar variables and linking constraints that will be required after combining the maximal cliques specified in `maxcliquesidx`. This function implements the clique combination heuristic described in [1]. A negative cost indicates that the heuristic predicts decreased computational costs after combining the specified maximal cliques.

- `[AINC] = MAKEINCIDENCE(BUS, BRANCH)`

  This function builds the bus incidence matrix. This matrix has size `nline` by `nbus`, with each row having two nonzero elements: $+1$ in the entry for the "from" bus of the corresponding line and -1 in the entry for the "to" bus of the corresponding line.

- `[YK, YK_, MK, YLINEFT, YLINETF, Y_LINEFT, Y_LINETF, YL, YL_]`
  `= MAKESDPMAT(MPC)`

  This creates functions that return the matrices used in the semidefinite programming relaxation of the optimal power flow problem.

- `[MPC] = CASE3SC`

  OPF data for 3-bus system used in [18]. The semidefinite relaxation of the OPF problem successfully solves `case3sc` with a value of 60 MVA for the line-flow limit on the line from bus 3 to bus 2. The semidefinite relaxation fails to give a physically meaningful solution to `case3sc` with a value of 50 MVA for the line-flow limit on this line. See [18] for further details.

- `[MPC] = CASE2LOCAL`

  OPF data for 2-bus system used in [19]. The semidefinite relaxation fails to give a physically meaningful solution to `case2local` for values of the upper voltage magnitude limit at bus two that are between 0.985 and 1.04. See [19] for further details.

7 Additional Functionality

Three functions provide additional tools that are based on the semidefinite programming relaxation of the power flow equations.

- `[GLOBALOPT, COMP, APSD] = TESTGLOBALOPT(MPC, MPOPT)`

  The function `testGlobalOpt` uses the KKT conditions of the semidefinite relaxation of the OPF problem to create a sufficient condition for global optimality of a specified OPF solution. The sufficient condition requires satisfaction of both a complementarity condition ($\text{trace}\,(\mathbf{AW}) = 0$) and a feasibility condition ($\mathbf{A} \succeq 0$). Failure to satisfy these conditions may result when the semidefinite relaxation does not satisfy the rank condition (i.e., $\text{rank}\,(\mathbf{W}) > 2$) [18, 19], in which case the solution may still

be globally optimal. Alternatively, failure to satisfy the sufficient condition for global optimality may indicate that a better solution exists. See [2] for details.

Specify an OPF solution in the input `mpc` that was solved with the associated options variable `mpopt` (i.e., `mpc = runopf(mpc, mpopt)`). The function `testGlobalOpt` outputs information on satisfaction of the complementarity and feasibility conditions in `comp` and `Apsd`, respectively. The output `comp` is the value of trace $(\mathbf{AW})$. The output `Apsd` is a flag indicating if the matrix $\mathbf{A} + \epsilon\mathbf{I}$ has a Cholesky factorization, where ϵ is specified by `sdp_pf.zeroeval_tol`, which is both necessary and sufficient for positive definiteness. If `comp` is equal to zero (within tolerances specified by the option `sqrt(sdp_pf.zeroeval_tol)`) and `Apsd` is equal to one, the output `globalopt` is equal to one; otherwise, `globalopt` equals zero.

Note that tight solution tolerances are typically required for good numeric results. Additionally, enforcing small minimum branch resistances may result in satisfaction of the complementarity and feasibility conditions for problems that would not otherwise satisfy these conditions.

- `[INSOLVABLE, VSLACK_MIN, SIGMA, ETA, MINEIGRATIO]`
  `= INSOLVABLEPF(MPC, MPOPT)`

  The function `insolvablepf` evaluates a sufficient condition for insolvability of the power flow equations (neglecting generator reactive power limits). A semidefinite programming relaxation of the power flow equations minimizes the slack bus voltage (with proportional changes in voltage at PV buses). If the minimum achievable slack bus voltage is greater than the specified slack bus voltage, no power flow solutions exist. The converse does not necessarily hold; a power flow solution may not exist even if the minimum slack bus voltage is less than or equal to the specified slack bus voltage.

  As a by-product of the optimization problem used to evaluate the insolvability condition, `insolvablepf` yields two voltage stability margins to the power flow solvability boundary. The output `sigma` is a margin in terms of controlled voltages which gives the factor by which the slack bus voltage can be changed (with proportional changes in PV bus voltages) while maintaining the possibility of power flow solution existence. The output `eta` is a margin in terms of power injections which gives the factor by which the power injections can be uniformly changed at constant power factor while maintaining the possibility of power flow solution existence.

  Due to incorporation of the matrix completion decomposition described in [1], this function is suitable for large systems.

  See [3] for further details.

- `[INSOLVABLE, ETA, MINEIGRATIO] = INSOLVABLEPF_LIMITQ(MPC, MPOPT)`

  The function `insolvablepf_limitQ` evaluates a sufficient condition for power flow insolvability considering generator reactive power limits. Generator reactive power

limits are incorporated using a mixed-integer semidefinite programming relaxation of the power flow equations. Using the semidefinite programming relaxation of the power flow equations, this function maximizes the power injections in a uniform, constant power factor injection profile. The result of this optimization problem, `eta`, is the largest factor by which the power injections can be changed in the direction of this profile while maintaining the possibility of power flow solution existence. Thus, `eta` is a margin of the distance to the power flow solvability boundary, considering generator reactive power limits. See [4] for further details.

Note that due to the computational limitations of YALMIP's branch-and-bound solver, this function is not suitable for large systems. This function has been successfully run on test systems with sizes up to the 57-bus system; 57 buses is therefore chosen as a limit on system size. This limit can be modified using the variable `maxSystemSize`.

- `[INSOLVABLE] = INSOLVABLEPFSOS(MPC,MPOPT)`

  The function `insolvablepfsos` implements a sufficient condition for power flow insolvability using sum of squares programming to generate an infeasibility certificate for the power flow equations. An infeasibility certificate proves insolvability of the power flow equations. Note that absence of an infeasibility certificate does not necessarily mean that a solution does exist (that is, insolvable = 0 does not imply solution existence). This implementation uses constant (degree zero) polynomials. See [4] for more details.

- `[INSOLVABLE] = INSOLVABLEPFSOS_LIMITQ(MPC,MPOPT)`

  The function `insolvablepfsos_limitQ` implements a sufficient condition for power flow insolvability considering gnenerators with upper limits on reactive power output. Sum of squares programming is used to generate an infeasibility certificate for the power flow equations. An infeasibility certificate proves insolvability of the power flow equations. Note that absence of an infeasibility certificate does not necessarily mean that a solution does exist (that is, insolvable = 0 does not imply solution existence). This implementation uses constant (degree zero) polynomials. See [4] for more details.

8 Limitations and Future Work

SDPOPF has several important differences from other OPF solvers that deserve special attention. First, although line flow constraints specified in terms of current flow are theoretically possible, they are not yet implemented in this code (i.e., MATPOWER option `opf.flow_lim` is restricted to `'S'` and `'P'` for apparent power and active power line flow limits, respectively). The ability to specify flow limits based on line currents may be added in a future release of `SDP_PF`.

Second, since non-convex objective functions would eliminate the convexity of the semidefinite relaxation, only convex objective functions are permissible. Quadratic and piecewise-linear objective functions of active power generation are currently implemented in SDPOPF. (Note that the current implementation does not allow buses with both piecewise-linear generator cost functions and quadratic generator cost functions.) More flexible objective function specifications, such functions of reactive power generation, are theoretically possible, but not planned for implementation at this time.

Third, DC lines are not explicitly implemented in SDPOPF. It is theoretically possible to model DC lines in the semidefinite relaxation as linked positive and negative power injections at the DC line terminal buses, but this functionality is not implemented nor planned for implementation at this time. SDPOPF will throw an error for input files with `dcline` fields.

Fourth, limits on branch angle differences are not implemented in SDPOPF. Future work includes incorporating angle difference constraints using the method described in [33].

Fifth, SDPOPF does not handle user-defined cost functions or constraints defined in MATPOWER. That is, some of the MATPOWER functionalities detailed in Section 5.3 (dispatchable loads, generator capability curves, and branch angle difference limits) and in Chapter 6 (customized callbacks, reserve requirements, interface flow limits, DC transmission lines, etc.) of the MATPOWER manual [34] are not implemented in SDPOPF. It may be possible to add some of these functionalities through modifications to the `sdpopf_solver.m` file.

A noteworthy difference in behavior between the current version of SDPOPF and other MATPOWER solvers is in the handling of dispatchable loads. In other solvers, dispatchable loads are specified as generators with negative generation limits. An additional constraint enforces constant power factor at the dispatchable load. In SDPOPF, dispatchable loads can still be specified with negative generation limits, but the constant power factor constraint is not enforced; the dispatchable load can demand any reactive power within the specified reactive power limits. This behavior may be changed in a future version of `SDP_PF` to be consistent with other solvers.

Sixth, note that recovering the optimal voltage profile requires a binding constraint. SDPOPF can currently recover solutions with binding voltage magnitude or line-flow constraints, which is sufficient for the majority of practical OPF problems. However, it is possible to construct cases with solutions that have neither binding voltage magnitude nor binding line-flow constraints. Since there is at least one binding constraint at the optimal solution for an OPF problem, it is possible, in principle, to recover the optimal voltage profile for any OPF problem. These cases are expected to be rare.

Finally, note that even with the computational advances from reference [1] the semidefinite program solver typically has significantly higher processing and memory requirements than other OPF solvers, particularly for large system models. With MATPOWER option `verbose` set to 2, SDPOPF will display diagnostic information that is useful for gauging the progress of SDPOPF for large system models.

9 Conclusion

This document has introduced a package of code implementing applications of a semidefinite programming relaxation of the power flow equations. When a rank condition is satisfied, the semidefinite programming relaxation obtains the globally optimal solution to an OPF problem. This code is an implementation of the research published in [1], incorporating both the modeling and computational advances contained therein. Modeling advances include the capability to limit flows on parallel branches and allow multiple generators at the same bus. Computational advances in matrix decomposition algorithms extend the applicability of these algorithms and significantly reduce computation time.

Additional functionality includes `testGlobalOpt`, an implementation of a sufficient condition for global optimality of an OPF solution, `insolvablepf`, an implementation of a sufficient condition for power flow insolvability, `insolvablepf_limitQ`, an implementation of a sufficient condition for power flow insolvability considering generator reactive power limits, `insolvablepfsos`, an implementation of method for generating infeasibility certificates for the power flow equations, and `insolvablepfsos_limitQ`, an implementation of a method for generating infeasibility certificates for the power flow equations with consideration of upper limits on reactive power generation.

Acknowledgements

The author would like to thank Dr. Bernard Lesieutre, Dr. Christopher DeMarco, Jesse Holzer, Alex Borden, and Vikas Dawar at the University of Wisconsin-Madison and Ray Zimmerman at Cornell University for their insights and assistance preparing and testing this code.

References

[1] D. Molzahn, J. Holzer, B. Lesieutre, and C. DeMarco, "Implementation of a Large-Scale Optimal Power Flow Solver Based on Semidefinite Programming," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 3987–3998, 2013.

[2] D. Molzahn, B. Lesieutre, and C. DeMarco, "A Sufficient Condition for Global Optimality of Solutions to the Optimal Power Flow Problem," *IEEE Transactions on Power Systems*, vol. 29, pp. 978–979, March 2014.

[3] D. K. Molzahn, B. C. Lesieutre, and C. L. DeMarco, "A Sufficient Condition for Power Flow Insolvability with Applications to Voltage Stability Margins," *IEEE Transactions on Power Systems*, vol. 28, no. 3, pp. 2592–2601, 2013.

[4] D. K. Molzahn, V. Dawar, B. C. Lesieutre, and C. L. DeMarco, "Sufficient Conditions for Power Flow Insolvability Considering Reactive Power Limited Generators with Applications to Voltage Stability Margins," in *Bulk Power System Dynamics and*

Control - IX. Optimization, Security and Control of the Emerging Power Grid, 2013 IREP Symposium, August 25-30 2013.

[5] B. Lesieutre and I. Hiskens, "Convexity of the Set of Feasible Injections and Revenue Adequacy in FTR Markets," *IEEE Transactions on Power Systems*, vol. 20, pp. 1790–1798, November 2005.

[6] W. Bukhsh, A. Grothey, K. McKinnon, and P. Trodden, "Local Solutions of the Optimal Power Flow Problem," vol. 28, no. 4, pp. 4780–4788, 2013.

[7] J. Lavaei and S. Low, "Zero Duality Gap in Optimal Power Flow Problem," *IEEE Transactions on Power Systems*, vol. 27, pp. 92–107, February 2012.

[8] M. Huneault and F. Galiana, "A Survey of the Optimal Power Flow Literature," *IEEE Transactions on Power Systems*, vol. 6, pp. 762–770, May 1991.

[9] J. Momoh, R. Adapa, and M. El-Hawary, "A Review of Selected Optimal Power Flow Literature to 1993. I. Nonlinear and Quadratic Programming Approaches," *IEEE Transactions on Power Systems*, vol. 14, pp. 96–104, February 1999.

[10] J. Momoh, M. El-Hawary, and R. Adapa, "A Review of Selected Optimal Power Flow Literature to 1993. II. Newton, Linear Programming and Interior Point Methods," *IEEE Transactions on Power Systems*, vol. 14, pp. 105–111, February 1999.

[11] Z. Qiu, G. Deconinck, and R. Belmans, "A Literature Survey of Optimal Power Flow Problems in the Electricity Market Context," in *Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES*, pp. 1–6, March 2009.

[12] R. Zimmerman, C. Murillo-Sánchez, and R. Thomas, "MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education," *IEEE Transactions on Power Systems*, no. 99, pp. 1–8, 2011.

[13] X. Bai, H. Wei, K. Fujisawa, and Y. Wang, "Semidefinite Programming for Optimal Power Flow Problems," *International Journal of Electrical Power & Energy Systems*, vol. 30, no. 6-7, pp. 383–392, 2008.

[14] A. Nemirovski, "Lectures on Modern Convex Optimization," 2012.

[15] L. Vandenberghe and S. Boyd, "Semidefinite Programming," *SIAM Review*, pp. 49–95, March 1996.

[16] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge Univ Pr, 2004.

[17] "Power Systems Test Case Archive. University of Washington Department of Electrical Engineering [Online] Available at: `http://www.ee.washington.edu/research/pstca/`,"

[18] B. C. Lesieutre, D. K. Molzahn, A. R. Borden, and C. L. DeMarco, "Examining the Limits of the Application of Semidefinite Programming to Power Flow Problems," in *49th Annual Allerton Conference on Communication, Control, and Computing, 2011*, September 28-30 2011.

[19] W. Bukhsh, A. Grothey, K. McKinnon, and P. Trodden, "Local Solutions of Optimal Power Flow," Tech. Rep. ERGO-11-017, University of Edinburgh School of Mathematics, Edinburgh Research Group in Optimization, 2011. [Online]. Available: `http://www.maths.ed.ac.uk/ERGO/pubs/ERGO-11-017.html`.

[20] B. Zhang and D. Tse, "Geometry of Feasible Injection Region of Power Networks," in *49th Annual Allerton Conference on Communication, Control, and Computing, 2011*, September 28-30 2011.

[21] S. Bose, D. Gayme, S. Low, and K. Chandy, "Optimal Power Flow Over Tree Networks," in *49th Annual Allerton Conference on Communication, Control, and Computing, 2011*, September 28-30 2011.

[22] S. Sojoudi and J. Lavaei, "Physics of Power Networks Makes Hard Optimization Problems Easy To Solve," in *To appear in 2012 IEEE Power & Energy Society General Meeting*, July 22-27 2012.

[23] J. Lavaei, D. Tse, and B. Zhang, "Geometry of Power Flows in Tree Networks," in *To appear in 2012 IEEE Power & Energy Society General Meeting*, July 22-27 2012.

[24] M. Fukuda, M. Kojima, K. Murota, K. Nakata, *et al.*, "Exploiting Sparsity in Semidefinite Programming via Matrix Completion I: General Framework," *SIAM Journal on Optimization*, vol. 11, no. 3, pp. 647–674, 2001.

[25] K. Nakata, K. Fujisawa, M. Fukuda, M. Kojima, and K. Murota, "Exploiting Sparsity in Semidefinite Programming via Matrix Completion II: Implementation and Numerical Results," *Mathematical Programming*, vol. 95, no. 2, pp. 303–327, 2003.

[26] S. Kim, M. Kojima, M. Mevissen, and M. Yamashita, "Exploiting Sparsity in Linear and Nonlinear Matrix Inequalities via Positive Semidefinite Matrix Completion," *Mathematical Programming*, vol. 129, no. 1, pp. 33–68, 2011.

[27] X. Bai and H. Wei, "A Semidefinite Programming Method with Graph Partitioning Technique for Optimal Power Flow Problems," *International Journal of Electrical Power & Energy Systems*, 2011.

[28] R. Jabr, "Exploiting Sparsity in SDP Relaxations of the OPF Problem," *IEEE Transactions on Power Systems*, vol. 27, pp. 1138–1139, May 2012.

[29] J. Lofberg, "YALMIP: A Toolbox for Modeling and Optimization in MATLAB," in *IEEE International Symposium on Computer Aided Control Systems Design, 2004*, pp. 284–289, IEEE, 2004.

[30] J. Sturm, "Using SeDuMi 1.02, A MATLAB Toolbox for Optimization Over Symmetric Cones," *Optimization Methods and Software*, vol. 11, no. 1, pp. 625–653, 1999.

[31] R. Tütüncü, K. Toh, and M. Todd, "Solving Semidefinite-Quadratic-Linear Programs using SDPT3," *Mathematical Programming*, vol. 95, no. 2, pp. 189–217, 2003.

[32] R. Tarjan and M. Yannakakis, "Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs," *SIAM Journal on Computing*, vol. 13, p. 566, 1984.

[33] R. Madani, S. Sojoudi, and J. Lavaei, "Convex Relaxation for Optimal Power Flow Problem: Mesh Networks," *IEEE Transactions on Power Systems*.

[34] R. D. Zimmerman and C. E. Murillo-Sánchez, "MATPOWER 5.0 User's Manual," December 17, 2014.