MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education

Ray Daniel Zimmerman, *Member, IEEE*, Carlos Edmundo Murillo-Sánchez, *Member, IEEE*, and Robert John Thomas, *Life Fellow, IEEE*

***Abstract*—MATPOWER is an open-source Matlab-based power system simulation package that provides a high-level set of power flow, optimal power flow (OPF), and other tools targeted toward researchers, educators, and students. The OPF architecture is designed to be extensible, making it easy to add user-defined variables, costs, and constraints to the standard OPF problem. This paper presents the details of the network modeling and problem formulations used by MATPOWER, including its extensible OPF architecture. This structure is used internally to implement several extensions to the standard OPF problem, including piece-wise linear cost functions, dispatchable loads, generator capability curves, and branch angle difference limits. Simulation results are presented for a number of test cases comparing the performance of several available OPF solvers and demonstrating MATPOWER's ability to solve large-scale AC and DC OPF problems.**

***Index Terms*—Load flow analysis, optimal power flow, optimization methods, power engineering, power engineering education, power system economics, power system simulation, power systems, simulation software, software tools.**

I. Introduction

THIS paper describes MATPOWER, an open-source Matlab power system simulation package [1]. It is used widely in research and education for AC and DC power flow and optimal power flow (OPF) simulations. It also includes tools for running OPF-based auction markets and co-optimizing reserves and energy. Included in the distribution are numerous example power flow and OPF cases, ranging from a trivial four-bus example to real-world cases with a few thousand buses.

MATPOWER consists of a set of Matlab M-files designed to give the best performance possible while keeping the code simple to understand and customize. Matlab has become a popular tool for scientific computing, combining a high-level language ideal for matrix and vector computations, a cross-platform runtime with robust math libraries, an integrated development environment and GUI with excellent visualization capabilities, and an active community of users and developers. As a high-level scientific computing language, it is well suited for the numerical computation typical of steady-state power system simulations.

The initial motivation for the development of the Matlab-based power flow and OPF code that would eventually become MATPOWER arose from the computational requirements of the PowerWeb platform [3], [4]. As a web-based market simulation platform used to test electricity markets, PowerWeb requires a "smart market" auction clearing software that uses an OPF to compute the allocations and pricing. Having the clear potential to be useful to other researchers and educators, the software was released in 1997 via the Internet as an open-source power system simulation package, now distributed under the GNU GPL [2]. Even beyond its initial release, much of the ongoing development of MATPOWER continued to be driven in large part by the needs of the PowerWeb project. This at least partially explains the lack of a graphical user interface used by some related tools such as PSAT [5].

While it is often employed as an end-user tool for simply running one-shot simulations defined via an input case file, the package can also be quite valuable as a library of functions for use in custom code developed for one's own research. At this lower level, MATPOWER provides easy-to-use functions for forming standard network Y_{bus} and B matrices, calculating power transfer and line outage distribution factors (PTDFs and LODFs), and efficiently computing first and second derivatives of the power flow equations, among other things. At a higher level, the structure of the OPF implementation is explicitly designed to be extensible [6], allowing for the addition of user-defined variables, costs, and linear constraints.

The default OPF solver is a high-performance primal-dual interior point solver implemented in pure-Matlab. This solver has application to general nonlinear optimization problems outside of MATPOWER and comes with a convenience wrapper function to make it trivial to set up and solve linear programming (LP) and quadratic programming (QP) problems.

To help ensure the quality of the code, MATPOWER includes an extensive suite of automated tests. Some may find the testing framework useful for creating automated tests for their own Matlab programs.

A number of Matlab-based software packages related to power system simulation have been developed by others. A nice summary of their features is presented in [5]. The primary distinguishing characteristics of MATPOWER, aside from

Manuscript received December 22, 2009; revised April 19, 2010; accepted May 17, 2010. Date of publication June 21, 2010; date of current version January 21, 2011. This work was supported in part by the Consortium for Electric Reliability Technology Solutions and the Office of Electricity Delivery and Energy Reliability, Transmission Reliability Program of the U.S. Department of Energy under the National Energy Technology Laboratory Cooperative Agreement No. DE-FC26-09NT43321. Paper no. TPWRS-00995-2009.

R. D. Zimmerman and R. J. Thomas are with the Department of Applied Economics and Management and the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14853 USA (e-mail: rz10@cornell.edu; rjt1@cornell.edu).

C. E. Murillo-Sánchez is with the Universidad Autónoma de Manizales, and with Universidad Nacional de Colombia, both in Manizales, Colombia (e-mail: carlos_murillo@ieee.org).

Digital Object Identifier 10.1109/TPWRS.2010.2051168

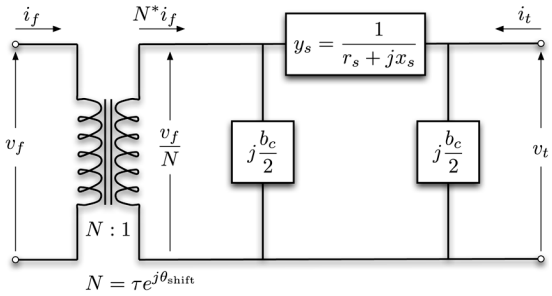

Fig. 1. Branch model.

being one of the first to be publicly and freely available as open-source, are the extensible architecture of the OPF formulation and its ease of use as a toolbox of functions to incorporate into one's own programs. It is also compatible with Octave.

This paper describes the MATPOWER package as it stands at version 4, detailing the component modeling in Section II, the power flow and optimal power flow formulations in Sections III and IV, and some additional functionality in Section V. Some example results and conclusions are presented in Section VI.

II. MODELING

MATPOWER employs all of the standard steady-state models typically used for power flow analysis. The AC models are described first, then the simplified DC models. Internally, the magnitudes of all values are expressed in per unit and angles of complex quantities are expressed in radians. Due to the strengths of the Matlab programming language in handling matrices and vectors, the models and equations are presented here in matrix and vector form.

A. Data Formats

The data files used by MATPOWER are Matlab M-files or MAT-files which define and return a single Matlab struct. The M-file format is plain text that can be edited using any standard text editor. The fields of the struct are baseMVA, bus, branch, gen, and optionally gencost, where baseMVA is a scalar and the rest are matrices. In the matrices, each row corresponds to a single bus, branch, or generator. The columns are similar to the columns in the standard IEEE CDF and PTI formats. The number of rows in bus, branch, and gen are n_b, n_l, and n_g, respectively.

B. Branches

All transmission lines, transformers, and phase shifters are modeled with a common branch model, consisting of a standard π transmission line model, with series impedance $z_s = r_s + jx_s$ and total charging capacitance b_c, in series with an ideal phase shifting transformer. The transformer, whose tap ratio has magnitude τ and phase shift angle θ_{shift}, is located at the *from* end of the branch, as shown in Fig. 1.

The complex current injections i_f and i_t at the *from* and *to* ends of the branch, respectively, can be expressed in terms of the 2×2 branch admittance matrix Y_{br} and the respective terminal voltages v_f and v_t

$$\begin{bmatrix} i_f \\ i_t \end{bmatrix} = Y_{br} \begin{bmatrix} v_f \\ v_t \end{bmatrix}. \tag{1}$$

With the series admittance element in the π model denoted by $y_s = 1/z_s$, the branch admittance matrix can be written

$$Y_{br} = \begin{bmatrix} \left(y_s + j\frac{b_c}{2}\right)\frac{1}{\tau^2} & -y_s \frac{1}{\tau e^{-j\theta_{\text{shift}}}} \\ -y_s \frac{1}{\tau e^{j\theta_{\text{shift}}}} & y_s + j\frac{b_c}{2} \end{bmatrix}. \tag{2}$$

If the four elements of this matrix for branch i are labeled as follows:

$$Y_{br}^i = \begin{bmatrix} y_{ff}^i & y_{ft}^i \\ y_{tf}^i & y_{tt}^i \end{bmatrix} \tag{3}$$

then four $n_l \times 1$ vectors Y_{ff}, Y_{ft}, Y_{tf}, and Y_{tt} can be constructed, where the ith element of each comes from the corresponding element of Y_{br}^i. Furthermore, the $n_l \times n_b$ sparse connection matrices C_f and C_t used in building the system admittance matrices can be defined as follows. The (i, j)th element of C_f and the (i, k)th element of C_t are equal to 1 for each branch i, where branch i connects from bus j to bus k. All other elements of C_f and C_t are zero.

C. Generators

A generator is modeled as a complex power injection at a specific bus. For generator i, the injection is

$$s_g^i = p_g^i + jq_g^i. \tag{4}$$

Let $S_g = P_g + jQ_g$ be the $n_g \times 1$ vector of these generator injections. A sparse $n_b \times n_g$ generator connection matrix C_g can be defined such that its (i, j)th element is 1 if generator j is located at bus i and 0 otherwise. The $n_b \times 1$ vector of all bus injections from generators can then be expressed as

$$S_{g,\text{bus}} = C_g \cdot S_g. \tag{5}$$

D. Loads

Constant power loads are modeled as a specified quantity of real and reactive power consumed at a bus. For bus i, the load is

$$s_d^i = p_d^i + jq_d^i \tag{6}$$

and $S_d = P_d + jQ_d$ denotes the $n_b \times 1$ vector of complex loads at all buses. Constant impedance and constant current loads are not implemented directly, but the constant impedance portions can be modeled as a shunt element described below. Dispatchable loads are modeled as negative generators and appear as negative values in S_g.

E. Shunt Elements

A shunt connected element such as a capacitor or inductor is modeled as a fixed impedance to ground at a bus. The admittance of the shunt element at bus i is given as

$$y_{sh}^i = g_{sh}^i + jb_{sh}^i \tag{7}$$

and $Y_{sh} = G_{sh} + jB_{sh}$ denotes the $n_b \times 1$ vector of shunt admittances at all buses.

F. Network Equations

For a network with n_b buses, all constant impedance elements of the model are incorporated into a complex $n_b \times n_b$ bus ad-

mittance matrix Y_{bus} that relates the complex nodal current injections I_{bus} to the complex node voltages V:

$$I_{\text{bus}} = Y_{\text{bus}} V. \tag{8}$$

Similarly, for a network with n_l branches, the $n_l \times n_b$ system branch admittance matrices Y_f and Y_t relate the bus voltages to the $n_l \times 1$ vectors I_f and I_t of branch currents at the *from* and *to* ends of all branches, respectively:

$$I_f = Y_f V \tag{9}$$
$$I_t = Y_t V. \tag{10}$$

If $[\cdot]$ is used to denote an operator that takes an $n \times 1$ vector and creates the corresponding $n \times n$ diagonal matrix with the vector elements on the diagonal, these system admittance matrices can be formed as follows:

$$Y_f = [Y_{ff}]\, C_f + [Y_{ft}]\, C_t \tag{11}$$
$$Y_t = [Y_{tf}]\, C_f + [Y_{tt}]\, C_t \tag{12}$$
$$Y_{\text{bus}} = C_f^{\mathsf{T}} Y_f + C_t^{\mathsf{T}} Y_t + [Y_{sh}]. \tag{13}$$

The current injections of (8)–(10) can be used to compute the corresponding complex power injections as functions of the complex bus voltages V:

$$S_{\text{bus}}(V) = [V]\, I_{\text{bus}}^* = [V]\, Y_{\text{bus}}^* V^* \tag{14}$$
$$S_f(V) = [C_f V]\, I_f^* = [C_f V]\, Y_f^* V^* \tag{15}$$
$$S_t(V) = [C_t V]\, I_t^* = [C_t V]\, Y_t^* V^*. \tag{16}$$

The nodal bus injections are then matched to the injections from loads and generators to form the AC nodal power balance equations, expressed as a function of the complex bus voltages and generator injections in complex matrix form as

$$g_S(V, S_g) = S_{\text{bus}}(V) + S_d - C_g S_g = 0. \tag{17}$$

G. DC Modeling

The DC formulation [11] (with more detailed derivations in [1]) is based on the same parameters, but with the following three additional simplifying assumptions.

- Branches can be considered lossless. In particular, branch resistances r_s and charging capacitances b_c are negligible:

$$y_s = \frac{1}{r_s + j x_s} \approx \frac{1}{j x_s}, \quad b_c \approx 0. \tag{18}$$

- All bus voltage magnitudes are close to 1 p.u.

$$v_i \approx e^{j\theta_i}. \tag{19}$$

- Voltage angle differences across branches are small enough that

$$\sin(\theta_f - \theta_t - \theta_{\text{shift}}) \approx \theta_f - \theta_t - \theta_{\text{shift}}. \tag{20}$$

By combining (1) and (2) with (18) and (19), the complex current flow in a branch can be approximated as

$$i_f \approx \frac{1}{j x_s \tau} \left(\frac{1}{\tau} e^{j\theta_f} - e^{j(\theta_t + \theta_{\text{shift}})} \right). \tag{21}$$

Furthermore, using (19) and this approximate current to compute the complex power flow, then extracting the real part and applying the last of the DC modeling assumptions from (20) yields

$$p_f \approx \frac{1}{x_s \tau} (\theta_f - \theta_t - \theta_{\text{shift}}). \tag{22}$$

As expected, given the lossless assumption, a similar derivation for p_t leads to $p_t = -p_f$.

The relationship between the real power flows and voltage angles for an individual branch i can then be summarized as

$$\begin{bmatrix} p_f \\ p_t \end{bmatrix} = B_{br}^i \begin{bmatrix} \theta_f \\ \theta_t \end{bmatrix} + P_{\text{shift}}^i \tag{23}$$

where $B_{br}^i = \begin{bmatrix} b_i & -b_i \\ -b_i & b_i \end{bmatrix}$, $P_{\text{shift}}^i = \theta_{\text{shift}}^i \begin{bmatrix} -b_i \\ b_i \end{bmatrix}$, and b_i is defined in terms of the series reactance and tap ratio for that branch as $b_i = 1/x_s^i \tau^i$.

With a DC model, the linear network equations relate real power to bus voltage angles, versus complex currents to complex bus voltages in the AC case. Let the $n_l \times 1$ vector B_{ff} be constructed similar to Y_{ff}, where the ith element is b_i and let $P_{f,\text{shift}}$ be the $n_l \times 1$ vector whose ith element is equal to $-\theta_{\text{shift}}^i b_i$. Then the nodal real power injections can be expressed as a linear function of Θ, the $n_b \times 1$ vector of bus voltage angles

$$P_{\text{bus}}(\Theta) = B_{\text{bus}} \Theta + P_{\text{bus,shift}} \tag{24}$$

where $P_{\text{bus,shift}} = (C_f - C_t)^{\mathsf{T}} P_{f,\text{shift}}$. Similarly, the branch flows at the *from* ends of each branch are linear functions of the bus voltage angles

$$P_f(\Theta) = B_f \Theta + P_{f,\text{shift}} \tag{25}$$

and, due to the lossless assumption, the flows at the *to* ends are given by $P_t = -P_f$. The construction of the system B matrices is analogous to the system Y matrices for the AC model:

$$B_f = [B_{ff}]\, (C_f - C_t) \tag{26}$$
$$B_{\text{bus}} = (C_f - C_t)^{\mathsf{T}} B_f. \tag{27}$$

The DC nodal power balance equations for the system can be expressed in matrix form as

$$g_P(\Theta, P_g) = B_{\text{bus}} \Theta + P_{\text{bus,shift}} + P_d + G_{sh} - C_g P_g = 0 \tag{28}$$

where G_{sh} approximates the amount of power consumed by the constant impedance shunt elements under the voltage assumption of (19).

III. Power Flow

The standard power flow or loadflow problem involves solving for the set of voltages and flows in a network corresponding to a specified pattern of load and generation. MATPOWER includes solvers for both AC and DC power flow problems, both of which involve solving a set of equations of the form

$$g(x) = 0 \tag{29}$$

constructed by expressing a subset of the nodal power balance equations as functions of unknown voltage quantities.

All of MATPOWER's solvers exploit the sparsity of the problem and, except for Gauss-Seidel, scale well to very large systems. Currently, none of them include any automatic updating of transformer taps or other techniques to attempt to satisfy typical OPF constraints, such as generator, voltage, or branch flow limits.

A. AC Power Flow

In MATPOWER, by convention, a single generator bus is typically chosen as a reference bus to serve the roles of both a voltage angle reference and a real power slack. The voltage angle at the reference bus has a known value, but the real power generation at the slack bus is taken as unknown to avoid overspecifying the problem. The remaining generator buses are classified as PV buses, with the values of voltage magnitude and generator real power injection given. Since the loads P_d and Q_d are also given, all non-generator buses are PQ buses, with real and reactive injections fully specified. Let $\mathcal{I}_{\mathrm{ref}}$, $\mathcal{I}_{\mathrm{PV}}$, and $\mathcal{I}_{\mathrm{PQ}}$ denote the sets of bus indices of the reference bus, PV buses, and PQ buses, respectively.

In the traditional formulation of the AC power flow problem, the power balance equation in (17) is split into its real and reactive components, expressed as functions of the voltage angles Θ and magnitudes V_m and generator injections P_g and Q_g, where the load injections are assumed constant and given:

$$g_P(\Theta, V_m, P_g) = P_{\mathrm{bus}}(\Theta, V_m) + P_d - C_g P_g = 0 \quad (30)$$

$$g_Q(\Theta, V_m, Q_g) = Q_{\mathrm{bus}}(\Theta, V_m) + Q_d - C_g Q_g = 0. \quad (31)$$

For the AC power flow problem, the function $g(x)$ from (29) is formed by taking the left-hand side of the real power balance equations (30) for all non-slack buses and the reactive power balance equations (31) for all PQ buses and plugging in the reference angle, the loads and the known generator injections and voltage magnitudes:

$$g(x) = \begin{bmatrix} g_P^{\{i\}}(\Theta, V_m, P_g) \\ g_Q^{\{j\}}(\Theta, V_m, Q_g) \end{bmatrix} \quad \begin{matrix} \forall i \in \mathcal{I}_{\mathrm{PV}} \cup \mathcal{I}_{\mathrm{PQ}} \\ \forall j \in \mathcal{I}_{\mathrm{PQ}}. \end{matrix} \quad (32)$$

The vector x consists of the remaining unknown voltage quantities, namely the voltage angles at all non-reference buses and the voltage magnitudes at PQ buses:

$$x = \begin{bmatrix} \theta_{\{i\}} \\ v_m^{\{j\}} \end{bmatrix} \quad \begin{matrix} \forall i \notin \mathcal{I}_{\mathrm{ref}} \\ \forall j \in \mathcal{I}_{\mathrm{PQ}}. \end{matrix} \quad (33)$$

This yields a system of nonlinear equations with $n_{pv} + 2n_{pq}$ equations and unknowns, where n_{pv} and n_{pq} are the number of PV and PQ buses, respectively. After solving for x, the remaining real power balance equation can be used to compute the generator real power injection at the slack bus. Similarly, the remaining $n_{pv} + 1$ reactive power balance equations yield the generator reactive power injections.

MATPOWER includes four different algorithms for solving the AC power flow problem. The default solver is based on a standard Newton's method [7] using a polar form and a full Jacobian updated at each iteration. Each Newton step involves computing the mismatch $g(x)$, forming the Jacobian based on the sensitivities of these mismatches to changes in x and solving for an updated value of x by factorizing this Jacobian. This method is described in detail in many textbooks.

Also included are solvers based on variations of the fast-decoupled method [8], specifically, the XB and BX methods described in [9]. These solvers greatly reduce the amount of computation per iteration, by updating the voltage magnitudes and angles separately based on constant approximate Jacobians which are factored only once at the beginning of the solution process. These per-iteration savings, however, come at the cost of more iterations. The fourth algorithm is the standard Gauss-Seidel method from Glimm and Stagg [10]. It has numerous disadvantages relative to the Newton method and is included primarily for academic interest.

By default, the AC power flow solvers simply solve the problem described above, ignoring any generator limits, branch flow limits, voltage magnitude limits, etc. However, there is an option that allows for the generator reactive power limits to be respected at the expense of the voltage setpoint. This is done by adding an outer loop around the AC power flow solution. If any generator has a violated reactive power limit, its reactive injection is fixed at the limit, the corresponding bus is converted to a PQ bus, and the power flow is solved again. This procedure is repeated until there are no more violations.

B. DC Power Flow

For the DC power flow problem [11], the vector x consists of the set of voltage angles at non-reference buses

$$x = \left[\theta_{\{i\}}\right], \quad \forall i \notin \mathcal{I}_{\mathrm{ref}} \quad (34)$$

and (29) takes the form

$$B_{dc} x - P_{dc} = 0 \quad (35)$$

where B_{dc} is the $(n_b - 1) \times (n_b - 1)$ matrix obtained by simply eliminating from B_{bus} the row and column corresponding to the slack bus and reference angle, respectively. Given that the generator injections P_g are specified at all but the slack bus, P_{dc} can be formed directly from the non-slack rows of the last four terms of (28).

The voltage angles in x are computed by a direct solution of the set of linear equations. The branch flows and slack bus generator injection are then calculated directly from the bus voltage angles via (25) and the appropriate row in (28), respectively.

C. Linear Shift Factors

The DC power flow model can also be used to compute the sensitivities of branch flows to changes in nodal real power injections, sometimes called injection shift factors (ISF) or generation shift factors [11]. These $n_l \times n_b$ sensitivity matrices, also called power transfer distribution factors or PTDFs, carry an implicit assumption about the slack distribution. If H is used to denote a PTDF matrix, then the element in row i and column j,h_{ij}, represents the change in the real power flow in branch i given a unit increase in the power injected at bus j, *with the assumption* that the additional unit of power is extracted according to some specified slack distribution:

$$\Delta P_f = H \Delta P_{\mathrm{bus}}. \quad (36)$$

This slack distribution can be expressed as an $n_b \times 1$ vector w of non-negative weights whose elements sum to 1. Each element specifies the proportion of the slack taken up at each bus. For the special case of a single slack bus k, w is equal to the vector e_k. The corresponding PTDF matrix H_k can be constructed by first creating the $n_l \times (n_b - 1)$ matrix

$$\tilde{H}_k = \tilde{B}_f \cdot B_{dc}^{-1} \tag{37}$$

then inserting a column of zeros at column k. Here $\tilde{B}_f$ and B_{dc} are obtained from B_f and B_{bus}, respectively, by eliminating their reference bus columns and, in the case of B_{dc}, removing row k corresponding to the slack bus.

The PTDF matrix H_w, corresponding to a general slack distribution w, can be obtained from any other PTDF, such as H_k, by subtracting w from each column, equivalent to the following simple matrix multiplication:

$$H_w = H_k(I - w \cdot 1^{\mathsf{T}}). \tag{38}$$

These same linear shift factors may also be used to compute sensitivities of branch flows to branch outages, known as line outage distribution factors or LODFs [12]. Given a PTDF matrix H_w, the corresponding $n_l \times n_l$ LODF matrix L can be constructed as follows, where l_{ij} is the element in row i and column j, representing the change in flow in branch i (as a fraction of its initial flow) for an outage of branch j.

First, let H represent the matrix of sensitivities of branch flows to branch flows, found by multplying the PTDF matrix by the node-branch incidence matrix:

$$H = H_w(C_f - C_t)^{\mathsf{T}}. \tag{39}$$

If h_{ij} is the sensitivity of flow in branch i with respect to flow in branch j, then l_{ij} can be expressed as

$$l_{ij} = \begin{cases} \frac{h_{ij}}{1-h_{jj}} & i \neq j \\ -1 & i = j. \end{cases} \tag{40}$$

MATPOWER includes functions for computing both the DC PTDF matrix and the corresponding LODF matrix for either a single slack bus k or a general slack distribution vector w.

IV. Optimal Power Flow

MATPOWER includes code to solve both AC and DC versions of the optimal power flow problem. The standard version of each takes the following form:

$$\min_x \quad f(x) \tag{41}$$

$$\text{subject to} \quad g(x) = 0 \tag{42}$$

$$h(x) \leq 0 \tag{43}$$

$$x_{\min} \leq x \leq x_{\max}. \tag{44}$$

A. Standard AC OPF

The optimization vector x for the standard AC OPF problem consists of the $n_b \times 1$ vectors of voltage angles Θ and magnitudes V_m and the $n_g \times 1$ vectors of generator real and reactive power injections P_g and Q_g:

$$x = \begin{bmatrix} \Theta \\ V_m \\ P_g \\ Q_g \end{bmatrix}. \tag{45}$$

The objective function (41) is simply a summation of individual polynomial cost functions f_P^i and f_Q^i of real and reactive power injections, respectively, for each generator:

$$\min_{\Theta, V_m, P_g, Q_g} \sum_{i=1}^{n_g} f_P^i(p_g^i) + f_Q^i(q_g^i). \tag{46}$$

The equality constraints in (42) are simply the full set of $2 \cdot n_b$ nonlinear real and reactive power balance equations from (30) and (31). The inequality constraints (43) consist of two sets of n_l branch flow limits as nonlinear functions of the bus voltage angles and magnitudes, one for the *from* end and one for the *to* end of each branch:

$$h_f(\Theta, V_m) = |F_f(\Theta, V_m)| - F_{\max} \leq 0 \tag{47}$$

$$h_t(\Theta, V_m) = |F_t(\Theta, V_m)| - F_{\max} \leq 0. \tag{48}$$

The flows are typically apparent power flows expressed in MVA, but can be real power or current flows, yielding the following three possible forms for the flow constraints:

$$F_f(\Theta, V_m) = \begin{cases} S_f(\Theta, V_m), & \text{apparent power} \\ P_f(\Theta, V_m), & \text{real power} \\ I_f(\Theta, V_m), & \text{current} \end{cases} \tag{49}$$

where I_f is defined in (9), S_f in (15), $P_f = \Re\{S_f\}$, and the vector of flow limits $F_{\max}$ has the appropriate units for the type of constraint. It is likewise for $F_t(\Theta, V_m)$.

The variable limits (44) include an equality constraint on any reference bus angle and upper and lower limits on all bus voltage magnitudes and real and reactive generator injections:

$$\theta_i^{\text{ref}} \leq \theta_i \leq \theta_i^{\text{ref}}, \qquad i \in \mathcal{I}_{\text{ref}} \tag{50}$$

$$v_m^{i,\min} \leq v_m^i \leq v_m^{i,\max}, \quad i = 1 \ldots n_b \tag{51}$$

$$p_g^{i,\min} \leq p_g^i \leq p_g^{i,\max}, \quad i = 1 \ldots n_g \tag{52}$$

$$q_g^{i,\min} \leq q_g^i \leq q_g^{i,\max}, \quad i = 1 \ldots n_g. \tag{53}$$

B. Standard DC OPF

When using DC network modeling assumptions and limiting polynomial costs to second order, the standard OPF problem above can be simplified to a quadratic program, with linear constraints and a quadratic cost function. In this case, the voltage magnitudes and reactive powers are eliminated from the problem completely and real power flows are modeled as linear functions of the voltage angles. The optimization variable is

$$x = \begin{bmatrix} \Theta \\ P_g \end{bmatrix} \tag{54}$$

and the overall problem reduces to (55)–(60) at the bottom of the page.

C. Extended OPF Formulation

MATPOWER employs an extensible OPF structure [6] to allow the user to modify or augment the problem formulation without rewriting the portions that are shared with the standard OPF formulation. This is done through optional input parameters, preserving the ability to use pre-compiled solvers. The standard formulation is modified by introducing additional optional user-defined costs f_u, constraints, and variables z and can be written in the following form:

$$\min_{x,z} \quad f(x) + f_u(x,z) \tag{61}$$

$$\text{subject to} \quad g(x) = 0 \tag{62}$$

$$h(x) \le 0 \tag{63}$$

$$x_{\min} \le x \le x_{\max} \tag{64}$$

$$l \le A \begin{bmatrix} x \\ z \end{bmatrix} \le u \tag{65}$$

$$z_{\min} \le z \le z_{\max}. \tag{66}$$

The user-defined cost function f_u is specified in terms a set of parameters in a pre-defined form described in detail in [6]. This form provides the flexibility to handle a wide range of costs, from simple linear functions of the optimization variables to scaled quadratic penalties on quantities, such as voltages, lying outside a desired range, to functions of linear combinations of variables, inspired by the requirements of price coordination terms found in the decomposition of large loosely coupled problems encountered in our own research.

D. Standard Extensions

In addition to making this extensible OPF structure available to end users, MATPOWER also takes advantage of it internally to implement several additional capabilities.

1) Piecewise Linear Costs: The standard OPF formulation in (41)–(44) does not directly handle the non-smooth piecewise linear cost functions that typically arise from discrete bids and offers in electricity markets. When such cost functions are convex, however, they can be modeled using a constrained cost variable (CCV) method. The piecewise linear cost function $c(x)$ is replaced by a helper variable y and a set of linear constraints that form a convex "basin" requiring the cost variable y to lie in the epigraph of the function $c(x)$.

A convex n-segment piecewise linear cost function

$$c(x) = \begin{cases} m_1(x - x_1) + c_1, & x \le x_1 \\ m_2(x - x_2) + c_2, & x_1 < x \le x_2 \\ \vdots & \vdots \\ m_n(x - x_n) + c_n, & x_{n-1} < x \end{cases} \tag{67}$$

can be defined by a sequence of points (x_j, c_j), $j = 0 \ldots n$, where m_j denotes the slope of the jth segment

$$m_j = \frac{c_j - c_{j-1}}{x_j - x_{j-1}}, \quad j = 1 \ldots n \tag{68}$$

and $x_0 < x_1 < \cdots < x_n$ and $m_1 \le m_2 \le \cdots < m_n$.

The "basin" corresponding to this cost function is formed by the following n constraints on the helper cost variable y:

$$y \ge m_j(x - x_j) + c_j, \quad j = 1 \ldots n. \tag{69}$$

The cost term added to the objective function in place of $c(x)$ is simply the variable y. For an AC or DC OPF, MATPOWER uses this CCV approach internally to automatically generate the appropriate helper variable, cost term, and corresponding set of constraints for any piecewise linear generator costs.

2) Dispatchable Loads: A simple approach to dispatchable or price-sensitive loads is to model them as negative real power injections with associated negative costs. This is done by specifying a generator with a negative output, ranging from a minimum injection equal to the negative of the largest possible load to a maximum injection of zero. With this model, if the negative cost corresponds to a benefit for consumption, minimizing the cost $f(x)$ of generation is equivalent to maximizing social welfare.

With an AC network model, there is also the question of reactive dispatch for such loads. In MATPOWER, it is assumed that dispatchable loads maintain a constant power factor and an additional equality constraint is automatically added to enforce this requirement for any "negative generator" being used to model a dispatchable load.

3) Generator Capability Curves: The typical AC OPF formulation includes simple box constraints on a generator's real and reactive injections. However, the true P-Q capability curves of physical generators usually involve some tradeoff between real and reactive capability. If the user provides the parameters defining this tradeoff for a generator, MATPOWER automatically constructs the corresponding constraints.

4) Branch Angle Difference Limits: The difference between the bus voltage angle θ_f at the *from* end of a branch and the

$$\min_{\Theta, P_g} \quad \sum_{i=1}^{n_g} f_P^i(p_g^i) \tag{55}$$

$$\text{subject to} \quad g_P(\Theta, P_g) = B_{\text{bus}}\Theta + P_{\text{bus,shift}} + P_d + G_{sh} - C_g P_g = 0 \tag{56}$$

$$h_f(\Theta) = B_f\Theta + P_{f,\text{shift}} - F_{\max} \le 0 \tag{57}$$

$$h_t(\Theta) = -B_f\Theta - P_{f,\text{shift}} - F_{\max} \le 0 \tag{58}$$

$$\theta_i^{\text{ref}} \le \theta_i \le \theta_i^{\text{ref}}, \quad i \in \mathcal{I}_{\text{ref}} \tag{59}$$

$$p_g^{i,\min} \le p_g^i \le p_g^{i,\max}, \quad i = 1 \ldots n_g \tag{60}$$

TABLE I
OPF TEST CASES

Case Name	Cost	Sizes			AC OPF		DC OPF		Binding Constraints		
		n_b	n_g	n_l	n_x	n_{gh}	n_x	n_{gh}	$V_m^{\min}$	$\lvert S_{f/t}^{\max}\rvert$	$\lvert P_{f/t}^{\max}\rvert$
case9	Q	9	3	9	24	36	12	27	0	0	0
case30	Q	30	6	41	72	142	36	112	0	2	0
case_npcc36	L	36	43	121	158	314	79	278	0	0	0
case118	Q	118	54	186	344	608	172	490	0	0	0
case300	Q	300	69	411	738	1,422	369	1,122	0	0	0
case2383wp	L	2,383	327	2,896	5,420	10,558	2,710	8,175	2	6	5
case2736sp	L	2,736	270	3,269	6,012	12,010	3,006	9,274	0	1	1
case3120sp	L	3,120	298	3,693	6,836	13,626	3,418	10,506	0	8	10
case2935	Q	2,935	1,024	7,028	7,918	6,314	3,959	3,379	19	7	9
case21k	L	21,084	2,692	25,001	54,091	111,784	30,315	90,700	3	64	40
case42k	L	42,168	5,384	50,001	107,027	222,796	59,475	180,628	3	137	66

angle θ_t at the *to* end can be bounded above and below to act as a proxy for a transient stability limit, for example. If these limits are provided, MATPOWER creates the corresponding constraints on the voltage angle variables.

E. Solvers

Early versions of MATPOWER relied on Matlab's Optimization Toolbox [13] to provide the NLP and QP solvers needed to solve the AC and DC OPF problems, respectively. While they worked reasonably well for very small systems, they did not scale well to larger networks. Eventually, optional packages with additional solvers were added to improve performance, typically relying on Matlab extension (MEX) files implemented in Fortran or C and pre-compiled for each machine architecture. For DC optimal power flow, there is a MEX build [14] of the high performance BPMPD solver [15] for LP/QP problems. For the AC OPF problem, the MINOPF [16] and TSPOPF [17] packages provide solvers suitable for much larger systems. The former is based on MINOS [18] and the latter includes the primal-dual interior point and trust region based augmented Lagrangian methods described in [19].

Beginning with version 4, MATPOWER also includes its own primal-dual interior point solver (MIPS) implemented in pure-Matlab code, derived from the MEX implementation of the corresponding algorithms in [19]. If no optional packages are installed, the MIPS solver will be used by default for both the AC OPF and as the QP solver used by the DC OPF. The AC OPF solver also employs a unique technique for efficiently forming the required Hessians via a few simple matrix operations. The MIPS solver has application to general nonlinear optimization problems outside of MATPOWER and comes with a convenience wrapper function to make it trivial to set up and solve LP and QP problems.

V. ADDITIONAL FUNCTIONALITY

As mentioned earlier, MATPOWER was birthed out of a need for an OPF-based electricity auction clearing mechanism for a "smart market". In this context, offers to sell and bids to buy power from generators and loads define the "costs" for the OPF that determines the allocations and prices used to clear the auction. MATPOWER includes code that takes bids and offers for real or reactive power, sets up and runs the corresponding OPF, and returns the cleared bids and offers.

The standard OPF formulation described above includes no mechanism for completely shutting down generators which are very expensive to operate. Instead they are simply dispatched at their minimum generation limits. MATPOWER includes the capability to run an OPF combined with a unit de-commitment for a single time period, which allows it to shut down these expensive units and find a least cost commitment and dispatch using an algorithm similar to dynamic programming.

In some cases, it may be desirable to further constrain an OPF solution with the requirement to hold a specified level of capacity in reserve to cover contingencies. MATPOWER includes OPF extensions that allow it to co-optimize energy and reserves, subject to a set of fixed zonal reserve requirements. This code also serves as an example of how to customize the standard OPF with additional variables, costs, and constraints.

VI. RESULTS AND CONCLUSIONS

Several example cases are used to compare the performance of the various OPF solvers on example networks ranging in size from nine buses and three generators to tens of thousands of buses, thousands of generators and tens of thousands of additional user variables and constraints. Table I summarizes the test cases in terms of the order of the cost function (quadratic or linear), numbers of buses, generators and branches (n_b, n_g, and n_l), numbers of variables and constraints (n_x and n_{gh}) for both AC and DC OPF formulations, and the number of binding lower voltage limits ($V_m^{\min}$) and branch flow limits ($\lvert S_{f/t}^{\max}\rvert$) for the AC problem and flow limits ($\lvert P_{f/t}^{\max}\rvert$) for the DC case.

For each case, six different AC OPF solvers and four different DC OPF solvers were used to solve the problem on a laptop with a 2.33-GHz Intel Core 2 Duo processor running Matlab 7.9. Table II gives the run times in seconds for the solvers which were successful, with the fastest time highlighted in bold for each example. The first algorithm listed for each is from Matlab's Optimization Toolbox, `fmincon` in the case of the AC OPF and `linprog` or `quadprog` for the DC problem. Next are the standard and step-controlled variants of the pure-Matlab implementation of the primal-dual interior point method, and last are some of the C and Fortran-based MEX solvers distributed as MATPOWER optional packages.

For small systems, the clear winners are MINOPF for AC and BPMPD for DC, both Fortran-based MEX files. For larger systems, the primal-dual interior point solvers have the clear advantage, with the pure-Matlab implementation offering respectable performance relative to the C-based MEX versions.

MATPOWER provides a high-level set of power flow and optimal power flow tools for researchers, educators, and students. The optimal power flow is extensible, allowing for easy

TABLE II
OPF RUN TIMES

Case Name	AC OPF Times (in seconds)						DC OPF Times (in seconds)			
	Opt Tbx	pure Matlab		MEX			Opt Tbx	pure Matlab		MEX
		MIPS	MIPS-sc	MINOPF	PDIPM	SC-PDIPM		MIPS	MIPS-sc	BPMPD
case9	0.147	0.080	0.120	**0.012**	0.024	0.027	0.014	0.019	0.020	**0.009**
case30	0.935	0.133	0.169	0.064	**0.056**	0.062	0.016	0.022	0.035	**0.012**
case_npcc36	13.5	0.247	0.332	0.346	**0.150**	0.216	0.070	0.027	0.031	**0.020**
case118	–	0.313	0.492	0.428	**0.232**	0.322	0.244	0.051	0.065	**0.023**
case300	–	0.795	–	4.07	**0.713**	4.04	0.369	0.077	0.138	**0.043**
case2383wp	–	8.24	10.2	94.8	**7.80**	8.94	9.00	**5.62**	5.91	–
case2736sp	–	8.36	10.4	86.8	**7.69**	8.84	–	**1.74**	1.84	–
case3120sp	–	14.1	17.7	384.9	**12.8**	15.0	9.96	**6.98**	7.32	7.95
case2935	–	**15.2**	18.1	1,262	15.3	17.5	–	**5.28**	5.68	–
case21k	–	822.1	1,027	–	**578.8**	610.4	–	**256.9**	263.8	–
case42k	–	5,232	5,933	–	**3,700**	3,701	–	1,241	**1,224**	–

modification of the problem formulation. The performance of the included OPF solvers, along with others available as optional plug-ins, scales quite well to very large systems. At the time of writing, there have been well over 20 000 downloads of MATPOWER, with about 50% primarily for education, 43% for research, and 7% for industry and other.

ACKNOWLEDGMENT

The authors would like to thank everyone who has contributed to MATPOWER, especially R. Bo, A. Borghetti, C. DeMarco, D. Gan, R. Korab, M. Lin, D. Mitarotonda, J. S. Thorp, A. Ward, P. Wei, and B. Wollenberg, and all of the users who extend MATPOWER's contributions through their own work.

REFERENCES

[1] R. D. Zimmerman and C. Murillo-Sánchez, MATPOWER User's Manual. [Online]. Available: http://www.pserc.cornell.edu/matpower/.
[2] GNU General Public License. [Online]. Available: http://www.gnu.org/licenses/gpl.html.
[3] R. D. Zimmerman, R. J. Thomas, D. Gan, and C. Murillo-Sánchez, "A web-based platform for experimental investigation of electric power auctions," *Decision Support Syst.*, vol. 24, no. 3-4, pp. 193–205, 1999.
[4] R. D. Zimmerman and R. J. Thomas, "PowerWeb: A tool for evaluating economic and reliability impacts of electric power market designs," in *Proc. IEEE PES Power Systems Conference and Exposition, 2004*, Oct. 10–13, 2004, vol. 3, pp. 1562–1567.
[5] F. Milano, "An open source power system analysis toolbox," *IEEE Trans. Power Syst.*, vol. 20, no. 3, pp. 1199–1206, Aug. 2005.
[6] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER's extensible optimal power flow architecture," in *Proc. IEEE Power and Energy Soc. General Meeting, 2009*, Jul. 26–30, 2009, pp. 1–7.
[7] W. F. Tinney and C. E. Hart, "Power flow solution by Newton's method," *IEEE Trans. Power App. Syst.*, vol. PAS-86, pp. 1449–1460, Nov. 1967.
[8] B. Stott and O. Alsaç, "Fast decoupled load flow," *IEEE Trans. Power App. Syst.*, vol. PAS-93, pp. 859–869, May 1974.
[9] R. A. M. van Amerongen, "A general-purpose version of the fast decoupled load flow," *IEEE Trans. Power Syst.*, vol. 4, no. 2, pp. 760–770, May 1989.
[10] A. F. Glimm and G. W. Stagg, "Automatic calculation of load flows," *AIEE Trans. (Power App. Syst.)*, vol. 76, pp. 817–828, Oct. 1957.
[11] A. J. Wood and B. F. Wollenberg, *Power Generation, Operation, and Control*, 2nd ed. New York: Wiley, 1996.
[12] T. Guler, G. Gross, and M. Liu, "Generalized line outage distribution factors," *IEEE Trans. Power Syst.*, vol. 22, no. 2, pp. 879–881, May 2007.
[13] Optimization Toolbox 4 Users's Guide, The MathWorks, Inc., 2008. [Online]. Available: http://www.mathworks.com/access/helpdesk/help/pdf_doc/optim/optim_tb.pdf.
[14] BPMPD_MEX. [Online]. Available: http://www.pserc.cornell.edu/bpmpd/.
[15] C. Mészáros, "The efficient implementation of interior point methods for linear programming and their applications," Ph.D. dissertation, Eötvös Loránd Univ. Sciences, Budapest, Hungary, 1996.
[16] MINOPF. [Online]. Available: http://www.pserc.cornell.edu/minopf/.
[17] TSPOPF. [Online]. Available: http://www.pserc.cornell.edu/tspopf/.
[18] B. A. Murtagh and M. A. Saunders, MINOS 5.5 User's Guide, Stanford Univ. Systems Optimization Laboratory, Tech. Rep. SOL83-20R.
[19] H. Wang, C. E. Murillo-Sánchez, R. D. Zimmerman, and R. J. Thomas, "On computational issues of market-based optimal power flow," *IEEE Trans. Power Syst.*, vol. 22, no. 3, pp. 1185–1193, Aug. 2007.

Ray Daniel Zimmerman (M'89) is a Senior Research Associate in electrical engineering and applied economics and management at Cornell University, Ithaca, NY. He is the lead developer of the PowerWeb electricity market simulation platform and the MATPOWER power system simulation software. His current research interests center on the interactions between the economic and engineering aspects of electric power system operations and planning. Other interests include software tools for education and research.

Carlos Edmundo Murillo-Sánchez (M'99) received the electronics engineering degree from ITESM, Monterrey, México, in 1987, the M.Sc. degree in electrical engineering from the University of Wisconsin-Madison in 1991, and the Ph.D. degree in electrical engineering from Cornell University, Ithaca, NY, in 1999.

He is a Professor of engineering at the Universidad Autónoma de Manizales, Manizales, Colombia, and a Professor at the Universidad Nacional de Colombia, also in Manizales. He is a founding member of the Colombian Automation Society (Asociación Colombiana de Automática). His interests include power systems operation and control, control systems applications, optimization, simulation, and mechatronics.

Robert John Thomas (LF'08) is Professor Emeritus of Electrical Engineering at Cornell University, Ithaca, NY. He has held sabbatical positions with the U.S. Department of Energy Office of Electric Energy Systems (EES) in Washington, DC, and at the National Science Foundation as the first Program Director for the Power Systems Program in the Engineering Directorates Division of Electrical Systems Engineering. His current technical research interests are broadly in the areas of analysis and control of nonlinear continuous and discrete time systems with applications to large-scale electric power systems.

Prof. Thomas is a member of Tau Beta Pi, Eta Kappa Nu, Sigma Xi, and ASEE.