Noname manuscript No.
(will be inserted by the editor)

Solving Security Constrained Optimal Power Flow Problems by a Structure Exploiting Interior Point Method

Nai-Yuan Chiang · **Andreas Grothey**

Received: date / Accepted: date

Abstract The aim of this paper is to demonstrate a new approach to solve the linearized (n-1) security constrained optimal power flow (SCOPF) problem by a structure exploiting interior point solver.

Firstly, we present a reformulation of the linearized SCOPF model, in which most matrices that need to be factorized are constant. Hence, most factorizations and a large number of backsolve operations only need to be performed once throughout the interior point methods (IPM) iterations.

However, assembling the Schur complement matrix remains expensive in this scheme. To overcome this, we suggest to use preconditioned iterative method to solve its corresponding linear system. We suggest two main schemes to pick a good and robust preconditioner based on combining different "active" contingency scenarios of the SCOPF model.

These new schemes are implemented within Object-Oriented Parallel Solver (OOPS), which is a modern structure-exploiting primal-dual interior-point implementation. We give results on several SCOPF test problems. The largest example contains 500 buses. We compare the results from the original IPM implementation in OOPS and our new reformulation.

Keywords SCOPF · interior point methods · structure · iterative methods · preconditioner

Nai-Yuan Chiang
School of Mathematics, JCMB, The King's Buildings, University of Edinburgh, UK EH9 3JZ
Tel.: +44-131-6505083
Fax: +44-131-6506553
E-mail: N.Chiang-2@sms.ed.ac.uk

Andreas Grothey
School of Mathematics, JCMB, The King's Buildings, University of Edinburgh, UK EH9 3JZ

1 Introduction

The optimal power flow (OPF) problem describes a minimum cost electricity generation model that takes into account generation level constraints, line flow constraint and bus voltage constraint. However, the normal OPF model is not necessarily secure against equipment failure.

Therefore, attention has turned to an improved model: the Security-Constrained Optimal Power Flow (SCOPF). This model guarantees that the transmission networks can successfully transfer power flow not only under the base network topology, but also for any contingency state caused by losing segments of the network, such as power transmission lines. The downside is the typically large size of the SCOPF model.

On the other hand, interior point methods (IPM) have proven to be robust and successful methods for solving various power system problems (Wei et al 1996; Quintana et al 2000; Capitanescu et al 2007). One advantage of IPM is that they are applicable to large problems and can easily exploit problem structure (Gondzio and Grothey 2009). Structure exploiting IPM have been applied to solving SCOPF problems (Karoui et al 2008; Qiu et al 2005; Petra and Anitescu 2010). In detail, Qiu, Flueck, and Tu (2005) solve the nonlinear SCOPF model with a structure exploiting IPM. They use an iterative method, namely GMRES, to solve the arising Schur complement system. Their preconditioner is derived from the base case scenario of the power system network without any other contingency considered. Petra and Anitescu 2010 also use an iterative method but use random contingencies to build the preconditioner. They showed that the preconditioned method outperforms a direct approach on medium-sized problems but experiences a bottleneck on larger problems.

In this paper, we will present a new approach to solve the large linearized SCOPF problem. Firstly we revisit the standard Schur complement factorization for the SCOPF and present a reformulation in which most matrices that need to be factorized are constant throughout the IPM iteration. Thus we only require a single factorization of these matrices to solve the problem. Nevertheless assembling the Schur complement matrix is still computationaly expensive, so as a second step we use a preconditioned iterative method to solve the Schur complement system in the spirit of Qiu et al (2005) and Petra and Anitescu (2010). We investigate two main different schemes to build the preconditioner by combining the contributions of “active” contingencies.

The paper is organized as follows. In Section 2, the structure of the linearized DC SCOPF model is described above. We give a symmetric transformation of the model. In Section 3, we firstly recall the linear algebra of interior point methods. Then we present a reformulation of the SCOPF, resulting in constant factorizable matrices. In Section 4, numerical results from tehe factorization-based approaches are presented. Comparisons are given between the original IPM approach in OOPS and our new approaches in OOPS. Finally, in the last section, we suggest to use an iterative method to solve the Schur complement system and discuss how to build an effective preconditioner based on active scenarios.

2 Structure Analysis of the SCOPF Model

A power system network consists of buses (nodes) $b \in \mathscr{B}$, transmission lines $l \in \mathscr{L}$ and power generators $g \in \mathscr{G}$. Power generators are allocated at buses, that is, for each generator g, $o_g \in \mathscr{B}$ gives the bus to which it is connected. We use $\mathscr{C}$ to denote the set of contingencies, indexed by c. In this paper, we only consider line failures, hence $\mathscr{C} \subseteq \mathscr{L}$ and c is the index of the contingent transmission line. Parameters $a_{bl,c}$ gives the line/bus incidence under contingency case c, i.e. a_{bl} takes value -1 if b is the start bus of line l; 1 if b is the end bus of l; otherwise takes 0. Hence $\mathbf{A}_c = \{a_{bl,c}\}$ is the node-arc incidence matrix when line c is missing. The lower/upper bound of the real power output of generator g is given by $p_g^{(-,+)}$ and f_l^+ denotes the power flow limit on transmission line l. We assume the voltage level V in the whole network is constant. Parameters o_g, d_b and r_l denote the location of power generator g, power demand at bus b and reactance of line l, respectively. The problem variables are defined as following: p_g is the real power generation at generator g; f_l^c is the line flow in line l under contingency c and δ_b^c is the voltage phase at bus b in contingency scenario c.

With these notations, the linearized SCOPF model can be described as below

$$\text{SCOPF Model:} \quad \min \quad \sum_{g \in \mathscr{G}} c_g p_g \tag{1}$$

$$s.t \quad p_g^- \leq p_g \leq p_g^+, \quad \forall g \in \mathscr{G} \tag{2}$$

$$-f_l^+ \leq f_l^c \leq f_l^+, \quad \forall l \in \mathscr{L}, c \in \mathscr{C} \tag{3}$$

$$\sum_{g|o_g=b} p_g + \sum_{l \in \mathscr{L}} a_{bl} f_l^c = d_b, \quad \forall b \in \mathscr{B}, \forall c \in \mathscr{C} \tag{4}$$

$$f_l^c = \begin{cases} -\frac{V^2}{r_l} \sum_{b \in \mathscr{B}} a_{bl} \delta_b^c, & \forall l \neq c \\ 0, & l = c \end{cases} \tag{5}$$

$$\delta_{b_0}^c = 0, \quad \forall c \in \mathscr{C}. \tag{6}$$

In this model the variables δ_b^c are only fixed up to a constant. Hence we set $\delta_{b_0}^c = 0$ at a designated reference bus b_0.

Constraints (2) and (3) represent the power generation limits and line flow limits, respectively. Whereas constraints (4) and (5) refer to the linearized Kirchhof Current Law and Kirchoff Voltage Law, respectively. This resulting linear model is known as the “DC model” as there is no reactive power flow in the model. Note that in the DC model, it is assumed that there are no transmission line losses.

To analyze the structure of the DC SCOPF model (1) - (6), we note that p_g are the only variables which affect all contingencies. Without consideration of p_g, the rest of the problem may be decomposed into separate sub-problem for each contingency scenario, and each sub-problem is similar in structure to the original OPF problem. In fact, the only difference between the standard OPF and the contingency sub-problem is that in the latter the broken line is missing.

We use subscript 0 to denote the sub-problem without any line outage and that is given by the basic OPF condition. Other subscript numbers denote the corresponding

contingent situation, e.g, “1” denotes the case in which line 1 has tripped. Then, after we reorder the variables of the SCOPF problem by contingencies as

$$f_l^0,\ \delta_b^0,\ f_l^1,\ \delta_b^1\ \ldots,\ f_l^{|\mathscr{C}|},\ \delta_b^{|\mathscr{C}|},\ p_g, \tag{7}$$

the whole constraint coefficient matrix can be reformed as a large sparse bordered block diagonal matrix in which each diagonal block matrix corresponds to one contingency scenario. This coefficient matrix can become very large with increasing size of the power system network and number of contingencies. With every line outage considered in the network, we need to add another diagonal block into the model.

Furthermore, within each contingency c, we always present Kirchoff Voltage Law before Kirchoff Current Law and reorder the variables as

$$f_{l_1}^c,\ f_{l_2}^c,\ \ldots,\ f_{l_{|\mathscr{L}|}}^c,\ \delta_{b_1}^c,\ \delta_{b_2}^c,\ldots,\ \delta_{b_{|\mathscr{B}|}}^c. \tag{8}$$

Moreover, there are lot of fixed variables in the model, e.g the tripped line flow must be forced to zero. All these fixed variables and its corresponding constraints can be removed altomatically by the presolve function in some mathematical modelling language, such as AMPL Fourer et al (2003). However, in order to keep the special structure of the SCOPF model, we do these steps manually by not introducing line flow variables of the collapsed line, f_c^c, and the reference bus voltage phase variables, $\delta_{b_0}^c$ (6), for all the contingency scenarios.

In the following, we use a 3-bus power network as an example to describe the structure of the coefficient matrix of the SCOPF problem. This 3-bus system contains three transmission lines and four generators. Its transmission network is shown in the Figure 1. In this small example, we only consider the OPF scenario and one contingency created by the loss of the line from bus 1 to bus 2.

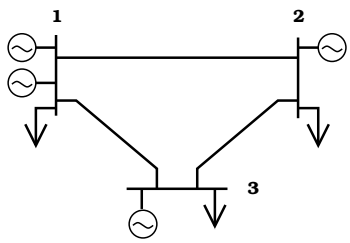

Fig. 1: 3-bus system

With above orderings (7) and (8), the coefficient matrix of the SCOPF modelling can be represented in a bordered block diagonal structure as Figure 2 and the 3-bus example without the fixed columns is visualized in Figure 3 where the black blocks indicate the non-zero elements. In detail, the first six rows in the Figure 3 correspond to the base case while the next five correspond to the single contingency. The first three rows, the 7th and the 8th rows denote the Kirchoff Voltage Law corresponding to the two different scenarios, respectively. The final column blocks $(\mathbf{F}_0, \mathbf{F}_1, \ldots, \mathbf{F}_{|\mathscr{C}|})$ is given by the power generation variables p_g.

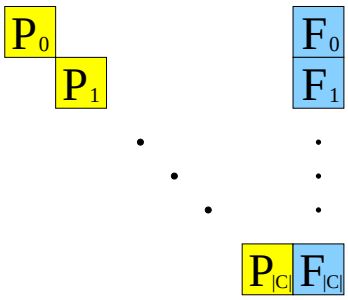

Fig. 2: Block angular matrix

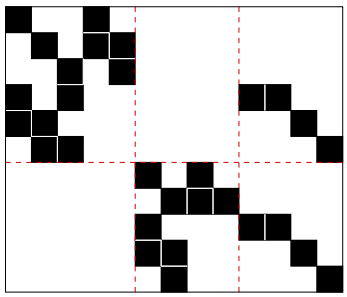

Fig. 3: 3-bus example with the SCOPF Model

Furthermore, we can achieve that the diagonal block matrices become symmetric through simple reformulation. In detail, because of the assumption that there is no loss in the transmission system, the total demand must be equal to the total real power generation. Hence, we may replace the KCL constraint at one specific bus by the total generation constraint $\sum_{g\in\mathscr{G}} p_g = \sum_{b\in\mathscr{B}} d_b$. We choose to eliminate the KCL at the reference bus b_0 and consequently the new reformulated model is given as below:

Modified Model:

$$\min \sum_{g\in\mathscr{G}} c_g p_g \tag{9}$$

$$s.t \quad p_g^- \le p_g \le p_g^+, \quad \forall g \in \mathscr{G} \tag{10}$$

$$-f_l^+ \le f_l^c \le f_l^+, \quad \forall c \in \mathscr{C}, l \in \mathscr{L} \setminus \{c\} \tag{11}$$

$$\sum_{g|o_g=b} p_g + \sum_{l\in\mathscr{L}\setminus\{c\}} a_{bl} f_l^c = d_b, \forall b \in \mathscr{B}\setminus\{b_0\}, c \in \mathscr{C} \tag{12}$$

$$f_l^c = -\frac{V^2}{r_l} \sum_{b\in\mathscr{B}\setminus\{b_0\}} a_{bl}\delta_b^c, \quad \forall c \in \mathscr{C}, l \in \mathscr{L}\setminus\{c\} \tag{13}$$

$$\sum_{g\in\mathscr{G}} p_g = \sum_{b\in\mathscr{B}} d_b. \tag{14}$$

With this modified formulation, the coefficient matrix of SCOPF problem can still has a block-angular form, as shown in the Figure 4. The most advantage of this new formulation is that each block diagonal matrix $\mathbf{P}_c$ is symmetric as

$$\mathbf{P}_c = \begin{bmatrix} \mathbf{R}_c & \mathbf{A}_c^T \\ \mathbf{A}_c & \mathbf{0} \end{bmatrix}, \tag{15}$$

where $\mathbf{R}_c = \text{diag}(\frac{r_1}{V^2}, \frac{r_2}{V^2}, \ldots, \frac{r_{c-1}}{V^2}, \frac{r_{c+1}}{V^2}, \ldots, \frac{r_{|\mathscr{L}|}}{V^2})$ and $\mathbf{A}_c$ is the bus-line node-arc matrix in contingency scenario c. It is worth mentioning that matrices $\mathbf{A}_c$ is represented as the reduced network node-arc matrix in which bus b_0 is eliminated. We do not require the reference bus b_0 since the KCL of the b_0 is replaced by the total generation constraint (14). On the other hand, the matrices $\mathbf{F}_c$, which are in the column

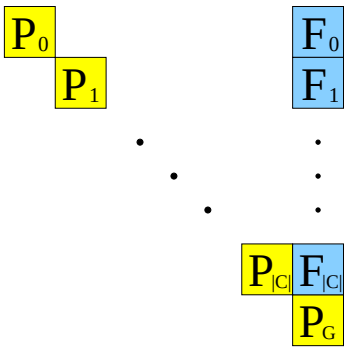

Fig. 4: Constraint matrix of the modified model.

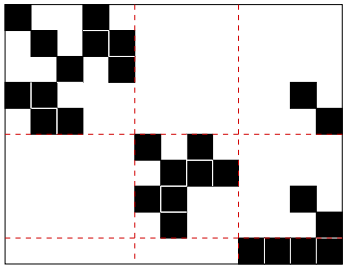

Fig. 5: The modified model for the 3-bus example.

border, have the form

$$\mathbf{F}_c = \begin{bmatrix} \mathbf{0} \\ \mathbf{J} \end{bmatrix}, \tag{16}$$

where $\mathbf{J} \in \mathscr{R}^{(|\mathscr{B}|-1)\times|\mathscr{G}|}$ is the bus-generator incidence matrix with only 0-1 entries. Even more, each column of $\mathbf{J}$ contains only one non-zero element since an individual generator can only be allocated on one bus. We also define matrix $\mathbf{J}$ as the bus-generator incidence matrix without the node corresponding to the reference bus b_0, as how matrix A_c is defined. Note that matrix F_c does not change through different contingency scenarios and it always has the same form (16).

The corresponding coefficient matrix of the modified model for the 3-bus example is shown in the Figure 5. Comparing to the structure of the original formulation, we do not need the 4th and 9th rows from Figure 3, which come from the KCL applied on the reference bus b_0 in both contingency scenario, respectively. Instead, we now have one extra diagonal block at the bottom.

Note that the leading diagonal block entry of the matrices $\mathbf{P}_c$ is the diagonal matrix $\mathbf{R}_c$. Since parameters r_l and V denote the reactance of the transmission line l and voltage level through the system respectively, entries of $\mathbf{R}_c$ are bounded well away from zero and all the r_l have the same order of magnitude. In other words, matrices $\mathbf{R}_c$ should be well-scaled. Moreover, matrix $\mathbf{A}_c$ is a node-arc incidence matrix with full row rank. Therefore, matrices $\mathbf{P}_c$ are invertible and can be stably $\mathbf{LDL^T}$-decomposed without further need of regularization.

3 Linear Algebra of Interior Point Method

Interior point methods (Wright 1997) are a certain powerful tool to solve large or extremely large optimization problems. In this section, we describe the basic background of interior point methods and discuss some details of the linear algebra issues encountered when solving the structured SCOPF problem appeared in the previous

section. Firstly, we consider the linear programming problem in standard form

$$\min\ \mathbf{c}^T\boldsymbol{x}\ \text{ s.t. } \mathbf{A}\boldsymbol{x}=\mathbf{b},\ \boldsymbol{x}\geq 0. \tag{17}$$

Its corresponding logarithmic barrier problem is

$$\min\ \mathbf{c}^T\boldsymbol{x}-\mu\sum_{j=1}^{n}\ln x_j\ \text{ s.t. } \mathbf{A}\boldsymbol{x}=\mathbf{b}, \tag{18}$$

where $\mu>0$ is a barrier parameter.

The generic framework of interior point method is to derive first order optimality conditions of the barrier problem (18) and perform Newton's method towarding their solution while decreasing μ to zero between steps. Here we use the so-called augmented system form to compute Newton direction $(\Delta\boldsymbol{x},\Delta\boldsymbol{y})$ as

$$\begin{bmatrix} -\Theta & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix}\begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{y} \end{bmatrix}=\begin{bmatrix} \xi_d-\mathbf{X}^{-1}\xi_\mu \\ \xi_p \end{bmatrix}, \tag{19}$$

where

$$\begin{aligned} \xi_p &= \mathbf{b}-\mathbf{A}\boldsymbol{x} \\ \xi_d &= \mathbf{c}-\mathbf{A}^T\boldsymbol{y}-\boldsymbol{z}, \\ \xi_\mu &= \mu\mathbf{e}-\mathbf{XZe} \end{aligned} \tag{20}$$

and

$$\Theta=\mathbf{X}^{-1}\mathbf{Z},\quad \mathbf{X}=\mathrm{diag}(x_1,x_2,\ldots,x_n),\quad \mathbf{Z}=\mathrm{diag}(z_1,z_2,\ldots,z_n).$$

The main computational effort of IPM is solving system (19), which is done by obtaining $\mathbf{LDL}^T$ factors of the augmented system matrix $\begin{bmatrix} -\Theta & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix}$. In the following, we explore how the structure of the coefficient matrix affects the IPM implementation.

To apply IPM, we build the augmented system matrix corresponding to the structured coefficient matrix from Figure 4. After reordering rows and columns, the augmented system matrix can be expressed as Figure 6, where block matrices $\Theta_i, i\in\{0,1,\ldots,|\mathscr{C}|,\mathrm{G}\}$ represent the appropriate sections of the diagonal matrix Θ. This reordered matrix is double bordered block diagonal and each diagonal block matrix is itself an augmented system matrix given by a contingency sub-problem.

To simplify the notation, we use the form (21)

$$\Phi=\begin{pmatrix} \Phi_1 & & & & \mathbf{B}_1^T \\ & \Phi_2 & & & \mathbf{B}_2^T \\ & & \ddots & & \vdots \\ & & & \Phi_n & \mathbf{B}_n^T \\ \mathbf{B}_1 & \mathbf{B}_2 & \cdots & \mathbf{B}_n & \Phi_0 \end{pmatrix}, \tag{21}$$

to refer to the reordered augmented system, where $n=|\mathscr{C}|+1, \Phi_i\in\mathscr{R}^{n_i\times n_i}$, $i=0,\ldots,n$ and $\mathbf{B}_i\in\mathscr{R}^{n_0\times n_i}$, $i=1,\ldots,n$. Matrix Φ has $\mathrm{N}=\sum_{i=0}^{n}n_i$ rows and columns.

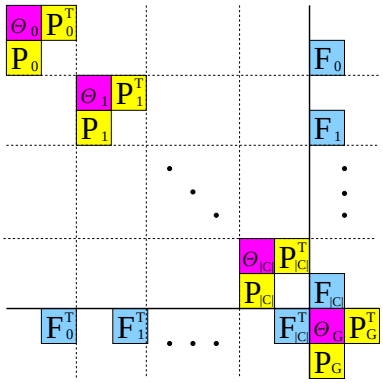

Fig. 6: Reordered augmented system matrix.

3.1 Default Structure-exploiting Technique

The linear algebra implementation in OOPS (Gondzio and Grothey 2009) and also in others' framework (Karoui et al 2008; Qiu et al 2005; Petra and Anitescu 2010) use a block Cholesky type factorization. That is, we have

$$\Phi = \mathbf{L}\mathbf{D}\mathbf{L}^T, \tag{22}$$

where

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_1 & & & & \\ & \mathbf{L}_2 & & & \\ & & \ddots & & \\ & & & \mathbf{L}_n & \\ \mathbf{L}_{n,1} & \mathbf{L}_{n,2} & \cdots & \mathbf{L}_{n,n} & \mathbf{L}_c \end{pmatrix}, \tag{23}$$

$$\mathbf{D} = \begin{pmatrix} \mathbf{D}_1 & & & & \\ & \mathbf{D}_2 & & & \\ & & \ddots & & \\ & & & \mathbf{D}_n & \\ & & & & \mathbf{D}_c \end{pmatrix}, \tag{24}$$

and

$$\Phi_i = \mathbf{L}_i\mathbf{D}_i\mathbf{L}_i^T, \quad \forall i = 1,\ldots,n \tag{25}$$

$$\mathbf{L}_{n,i} = \mathbf{B}_i\mathbf{L}_i^{-T}\mathbf{D}_i^{-1}, \quad \forall i = 1,\ldots,n \tag{26}$$

$$\mathbf{C} = \Phi_0 - \sum_{i=1}^{n} \mathbf{B}_i\Phi_i^{-1}\mathbf{B}_i^T = \mathbf{L}_c\mathbf{D}_c\mathbf{L}_c^T. \tag{27}$$

We use this factorization and its corresponding back-solve sequence to obtain the solution of the system $\Phi \boldsymbol{x} = \mathbf{b}$, where $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \boldsymbol{x}_0)$ and $\mathbf{b} = (\mathbf{b}_1, \ldots, \mathbf{b}_n, \mathbf{b}_0)$. The steps of the back-solve are indicated by the following sequence

$$\boldsymbol{z}_i = \mathbf{L}_i^{-1}\mathbf{b}_i, \quad i = 1, \ldots, n \tag{28}$$

$$\boldsymbol{z}_0 = \mathbf{L}_c^{-1}(\mathbf{b}_0 - \sum_{i=1}^{n} \mathbf{L}_{n,i}\boldsymbol{z}_i), \tag{29}$$

$$\boldsymbol{y}_i = \mathbf{D}_i^{-1}\boldsymbol{z}_i, \quad i = 0, \ldots, n \tag{30}$$

$$\boldsymbol{x}_0 = \mathbf{L}_c^{-T}\boldsymbol{y}_0, \tag{31}$$

$$\boldsymbol{x}_i = \mathbf{L}_i^{-T}(\boldsymbol{y}_i - \mathbf{L}_{n,i}^T\boldsymbol{x}_0), \quad i = 1, \ldots, n \tag{32}$$

From equations (25) - (27), it is obvious that we can avoid factorizing the entire augmented system matrix Φ directly. As an alternative, we find the Cholesky factors of the smaller augmented matrices Φ_i. In addition, an inevitable step in this framework is that we need to build the Schur complement $\mathbf{C}$ and compute its corresponding Cholesky factorization (27). This however is a much smaller system since the dimension of matrix $\mathbf{C}$ is the number of power generators, n_g, plus one. Moreover, it is often better to calculate the contributions $\mathbf{B}_i\Phi_i^{-1}\mathbf{B}_i^T$ of the Schur complement matrix C (27) by terms

$$\mathbf{B}_i\Phi_i^{-1}\mathbf{B}_i^T = (\mathbf{L}_i^{-1}\mathbf{B}_i^T)^T\mathbf{D}_i^{-1}(\mathbf{L}_i^{-1}\mathbf{B}_i^T), \tag{33}$$

The implementation of OOPS (Gondzio and Grothey 2009) saves $\mathbf{V}_i = \mathbf{L}_i^{-1}\mathbf{B}_i^T$ and computes $\mathbf{V}_i^T\mathbf{D}_i^{-1}\mathbf{V}_i$ by getting row-wise access to $\mathbf{V}_i$. Thata is, for all rows $\mathbf{r}_j$ in $\mathbf{V}_i$, OOPS updates the Schur complement: $\mathbf{C} = \mathbf{C} - (\mathbf{r}_j * \mathbf{r}_j^T)/\mathbf{d}_j$, where $\mathbf{d}_j$ is the jth entry of the diagonal matrix $\mathbf{D}_i$.

It is worth mentioning that in this default method, matrix $\Theta_i = \mathbf{X}_i^{-1}\mathbf{Z}_i$ changes between in each IPM iteration because both of the primal variable X and dual variable Z need to be updated. In addition, the values of $\mathbf{X}$ and $\mathbf{Z}$ tend toward zero or infinity when IPM converges. Therefore, the regularization of Φ_i is required when we factorize it. The implementation of OOPS uses a dynamical regularization framework. More details can be found in the reference Altman and Gondzio (1993).

3.2 Structure Exploitation for SCOPF model

The augmented system matrix of the SCOPF problem has the double bordered block diagonal form (21) and each diagonal blocks $\Phi_i = \begin{bmatrix} -\Theta_i & \mathbf{P}_i^T \\ \mathbf{P}_i & \mathbf{0} \end{bmatrix}$. Besides, unlike the general situation, now coefficient matrices $\mathbf{P}_i$ (15), from the modified model, are themselves of augmented system format, symmetric and invertible. An obvious question is how we can take advantage of this fact.

In the following, we introduce a new method in which we can avoid factorizing matrices ϕ_i. Instead, we decompose the smaller matrices $\mathbf{P}_j = \mathbf{LDL}^T$. For instance, solving the linear system $\Phi_i\boldsymbol{x}_i = \mathbf{b}_i$, which is equivalent to finding the solution

$(x_{i,1}, x_{i,2})$ to the system

$$\begin{bmatrix} -\Theta_i & \mathbf{P}_i^T \\ \mathbf{P}_i & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{i,1} \\ \boldsymbol{x}_{i,2} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{i,1} \\ \mathbf{b}_{i,2} \end{bmatrix}, \tag{34}$$

can be replaced by the following sequence

$$x_{i,1} = \mathbf{P}_i^{-1}\mathbf{b}_{i,2} \tag{35}$$

$$x_{i,2} = \mathbf{P}_i^{-1}(\mathbf{b}_{i,1} + \Theta_i \boldsymbol{x}_{i,1}), \tag{36}$$

where $\mathbf{P}_i^{-1} = \mathbf{LDL}^{-1}$ can be obtained from its Cholesky factors. In order to make the notation easier and clear to read, in the following we keep using notation Φ_i^{-1} to indicate solving the linear system (34) by the sequence (35)-(36).

On the other hand, since now we do not have access to the Cholesky factors of matrix Φ_i anymore, the back-solving process (28)-(32) needs to be replaced by process (37)-(41).

$$\boldsymbol{z}_i = \Phi_i^{-1}\mathbf{b}_i, \quad i = 1, \ldots, n \tag{37}$$

$$\boldsymbol{z}_0 = \mathbf{L}_c^{-1}(\mathbf{b}_0 - \sum_{i=1}^{n} \mathbf{B}_i \boldsymbol{z}_i), \tag{38}$$

$$\boldsymbol{y}_0 = \mathbf{D}_c^{-1}\boldsymbol{z}_0, \tag{39}$$

$$\boldsymbol{x}_0 = \mathbf{L}_c^{-T}\boldsymbol{y}_0, \tag{40}$$

$$\boldsymbol{x}_i = \boldsymbol{z}_i - \Phi_i^{-1}\mathbf{B}_i \boldsymbol{x}_0, \quad i = 1, \ldots, n \tag{41}$$

Note that while matrices Φ_i depends on $\Theta_i = \mathbf{X}_i^{-1}\mathbf{Z}_i$ and change in every IPM iteration, $\mathbf{P}_i$ is constant throughout and hence only needs to be factorized once. In addition, it is unecessary to use additional pivoting and regularization to factorize $\mathbf{P}_i$ since it is well-scaled. Therefore, back-solves with $\mathbf{P}_i$ should be very stable and improve the quality of the solution to the system.

With the modified model, we still need to compute the Cholesky factorization of the Schur complement $\mathbf{C}$. As far as we do not have the Cholesky factors L_i, D_i and L_i^T of Φ_i, we cannot use the equation (33) to form matrix $\mathbf{C}$. Alternatively, we introduce three different schemes to build the Schur complement $\mathbf{C}$:

A) We can evaluate $\Phi_i^{-1}\mathbf{B}_i^T$ by solving $n_g + 1$ linear systems according to sequence (35)-(36), where n_g is the number of power generators. After we compute $\Phi_i^{-1}\mathbf{B}_i^T$ by this scheme, we then multiply the solutions by $\mathbf{B}_i$ to obtain the contingency contribution $\mathbf{B}_i\Phi_i^{-1}\mathbf{B}_i^T$. This is the most straightforward method among our three methods, as it computes $\mathbf{B}_i\Phi_i^{-1}\mathbf{B}_i^T$ from right to left, obviously. We use this method to compare with the next two more sophisticated alternatives.

B) As we described in the last section, block matrices $\mathbf{F}_i$ has the same specific form (16) no matter which contingency scenario is. Considering the definition (21) and the corresponding SCOPF structure in Figure 6, we can find that the column border

matrix $\mathbf{B}_i^T$ is equivalent to $\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{F} & \mathbf{0} \end{bmatrix}$, where each column of $\mathbf{F}$ only contains one non-zero element. Note that the last block column of $\mathbf{B}_i^T$ is an one dimension zero column. To simplify the notation, we use $\begin{bmatrix} \mathbf{0} \\ \tilde{\mathbf{F}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{F} & \mathbf{0} \end{bmatrix}$. The contribution $\mathbf{B}_i \Phi_i^{-1} \mathbf{B}_i^T$ can be further derived as

$$\mathbf{B}_i \Phi_i^{-1} \mathbf{B}_i^T = \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{F}}^T \end{bmatrix} \Phi_i^{-1} \begin{bmatrix} \mathbf{0} \\ \tilde{\mathbf{F}} \end{bmatrix}. \tag{42}$$

If we take the special structures of Φ into consideration and recall the solving sequence (35)-(36), we find that now

$$\Phi_i^{-1} \mathbf{B}_i^T = \Phi_i^{-1} \begin{bmatrix} \mathbf{0} \\ \tilde{\mathbf{F}} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_i^{-1} \tilde{\mathbf{F}} \\ \mathbf{P}_i^{-T} \Theta_i \mathbf{P}_i^{-1} \tilde{\mathbf{F}} \end{bmatrix},$$

where the first sub-block matrix of the solution, that is $\mathbf{P}_i^{-1} \tilde{\mathbf{F}}$, is a constant matrix. On the other hand, sub-block $\mathbf{P}_i^{-T} \Theta_i \mathbf{P}_i^{-1} \tilde{\mathbf{F}}$ changes between each IPM iteration because the term Θ_i is changing. Therefore, we only need to compute the corresponding sub-solution $\mathbf{P}_i^{-1} \tilde{\mathbf{F}}$ once. This process corresponds to find $x_{i,1}$ in (35). We may save all these fixed solutions and then use them in the later IPM iterations. That is, when we compute the Schur complement C in latter IPM iterations, we reuse the saved data $\mathbf{P}_c^{-1} \tilde{\mathbf{F}}$ and just need to calculate the term $\mathbf{P}_c^{-T} \Theta_c \mathbf{P}_c^{-1} \tilde{\mathbf{F}}$. Then, we multiply the result by $\begin{bmatrix} \mathbf{0} & \tilde{\mathbf{F}}^T \end{bmatrix}$ to get the contributions

$$\mathbf{B}_i \Phi_i^{-1} \mathbf{B}_i^T = \tilde{\mathbf{F}}^T \mathbf{P}_i^{-T} \Theta_i \mathbf{P}_i^{-1} \tilde{\mathbf{F}}, \tag{43}$$

where $\tilde{\mathbf{F}}$ is very sparse as we described eariler. It is worth mentioning that this method computes $\mathbf{B}_i \Phi_i^{-1} \mathbf{B}_i^T$ from right to left, as what method A does, but the difference is we save the constant results and reuse them in this scheme.

C) As (33) in the default method, we can reformulate (43) as

$$\mathbf{B}_i \Phi_i^{-1} \mathbf{B}_i^T = \mathbf{V}_i^T \Theta_i \mathbf{V}_i, \tag{44}$$

where $\mathbf{V}_i = \mathbf{P}_i^{-1} \tilde{\mathbf{F}}$. Note that $\mathbf{V}_i$ is constant throughout the IPM iterations. Then, we may compute $\mathbf{V}_i^T \Theta_i \mathbf{V}_i$ by getting row-wise access to $\mathbf{V}_i$. However, this may not speed up the process. In fact, unlike the situation in (33) where $\mathbf{L}_i^{-1} \mathbf{B}_i^T$ is typically sparse, matrix $\mathbf{V}_i$ now is obtained by taking columns from $\mathbf{P}_i^{-1}$ which is dense. Hence, we tend to use routing DSYRK of CBLAS to perform this symmetric matrix operation.

In this section, we presented four methods to solve the augmented system in IPM. They are summarized in Table 1. Compared with the default method, we expect that the second method should require less memory since now we only need to factorize smaller matrices $\mathbf{P}_i$. On the other hand, the third and fourth method should require more memory since they need to save partial solutions to build the the Schur complement C. Compared to the second method, the third method should definitely save time on computing the Cholesky factorization and back-solving the linear system. We expect the third method to be the fastest one to solve the linearized SCOPF problems.

In addition, for each $\Phi^{-1}\mathbf{b}$ operation according to (35) and(36), we need to do four back-solves with the L-factors of $\mathbf{P}_i$, consisting of two in (35) and two in (36). That is because we only have $\mathbf{LDL}^T$ access to smaller matrix $\mathbf{P}_i$. However, in Method 1, we only require two backsolves albeit with the larger factors of Φ_i. Hence we are interested in how the Method 1 and Method 2 would behave since they both have their own advantages and disadvantages.

Table 1: Four different methods

	Detail	Factorization	Way to build $\mathbf{B}_c\Phi_c^{-1}\mathbf{B}_c^T$	Save Data
Method 1(Default)	Section 3.1.	Φ_i	Row-wise access to $\mathbf{V}_j = \mathbf{L}_j^{-1}\mathbf{B}_j^T$	N
Method 2	Refer to A)	P_i	Compute (43) from right to left	N
Method 3	Refer to B)	P_i	Compute (43) from right to left	Y
Method 4	Refer to C)	P_i	Row-wise access to $\mathbf{V}_j = \mathbf{P}_i^{-1}\mathbf{F}$	Y

4 Numerical results

To demonstrate the efficiency of our approaches, we used several test networks with different number of buses: 3, 26, 56, 100, 200, 300 and 500. Among these test problems, examples with sizes 26 and 56 are modified from the IEEE test problems. The test models with more than 56 buses are made up randomly using a 2D grid-like distribution network. The number of generators are fixed as one-fourth of the number of buses and other values of parameters, such as reactance, are generated as uniform random data, using the same parameter range as the IEEE test problems. For all the test problems, we ensure that each bus is connected to the network with at least two transmission lines to guarantee that the n-1 SCOPF formulation is meaningful and feasible. The number of variables and constraints in each test problem are shown in Table 2. The last column denotes the number of nonzero elements in the coefficient matrix.

Examples are tested on a 2.33GHz Intel(R) Xeon(R) computer with 4GB RAM, running Redhat Enterprise Linux. The code is compiled with gcc with compile option -O2.

The tolerances of finding the optimal solution is set to 10^{-6}. For all these problems, we compare the time and the number of IPM iterations for solving these examples to convergence. In addition, we also focus on the memory requirement of the these different methods. We use OOPS to test our different IPM implementations. The results of those four methods are represented in Table 3.

Comparing these four different implementations, we find that the second method needs the least amount of memory while the third one is the fastest. As we expected, both the third and the fourth methods require more memory since we need to save additional data. Furthermore, in comparison to Method 1, Method 2 requires less memory for all test problems but spends more time on solving larger systems. In

Table 2: Problem details of the test problems.

buses	generators	contingencies	variables	constraints	nonzeros
3	4	2	17	14	35
26	5	40	2,630	2,626	7,929
56	7	79	10,648	10,642	31,060
100	25	180	50,344	50,320	165,649
200	50	370	210,779	210,730	701,249
300	75	565	488,534	488,460	1,635,824
400	100	760	881,339	881,240	2,960,399
500	125	955	1,389,194	1,389,070	4,674,974

Table 3: Test results from four methods in OOPS.

	Method 1			Method 2		
size	user time(s)	memory	iters	user time(s)	memory	iters
3	<0.01	5.2MB	8	<0.01	5.2MB	8
26	0.21	7.6MB	13	0.17	7.4MB	13
56	1.00	14.3MB	15	0.82	13.5MB	15
100	9.02	54.6MB	20	7.45	42.5MB	20
200	65.97	220MB	27	61.93	161MB	27
300	251.70	531MB	34	261.07	378MB	33
400	955.32	985MB	59	1023.71	699MB	53
500	1552.80	1593MB	49	1877.23	1109MB	47
	Method 3			Method 4		
size	user time(s)	memory	iters	user time(s)	memory	iters
3	<0.01	5.2MB	8	<0.01	5.2MB	8
26	0.17	7.5MB	13	0.16	7.5MB	13
56	0.77	14.1MB	15	0.72	14.0MB	15
100	6.16	53.1MB	20	5.39	53.1MB	20
200	45.23	244MB	27	41.88	244MB	27
300	177.39	667MB	33	167.85	663MB	33
400	655.50	1380MB	53	659.22	1366MB	54
500	1195.77	2467MB	47	1252.20	2466MB	47

fact, the performance of the second method is better than the default method if the network size is less than 300. Method 4 is comparable to the default method in speed only if the size is less than 400. But it require too much more memory. Focusing on the memory consumption side, since the second method only factorizes the smaller dimensioned matrix $\mathbf{P}_i$ and does not save extra information, it consequently needs least memory. For the systems with size more than 100 buses, we found the default method requires less memory than both the third method and fourth method. This is due to the fact that the default algorithm needs to build matrices $\mathbf{V}_i = \mathbf{L}_i^{-1}\mathbf{B}_i^T$. This technique is used to reduce the memory need for large or extremely large problems, in which the matrices $\mathbf{V}_i$ are always sparse. On the other hand, the matrices $\mathbf{V}_i = \mathbf{P}_i^{-1}\mathbf{F}$ are smaller in dimension but very dense, hence it may require less memory in the small test problems but it needs much more memory for the larger problem.

Among these four IPM implementations in OOPS, we note that method 3 is the best one if we only focus on its speed. Hence saving the partial data when we build the Schur complement $\mathbf{C}$ can definitely speed up the solution process.

However, due to saving partial results it requires too much memory and assembling the Schur complement is computationally expensive. Hence it is not well suited for large problems. To overcome this difficulty, in the next section, we use an iterative solver within Method 3 to avoid building $\mathbf{C}$.

5 Preconditioned Iterative Method

In all methods discussed so far in the previous section, we need to build the Schur complement $\mathbf{C} = \Phi_0 - \sum_{j=1}^{n} \tilde{\mathbf{F}}^T \mathbf{P}_c^{-T} \Theta_i \mathbf{P}_i^{-1} \tilde{\mathbf{F}}$ as a dense matrix and factorize it as $\mathbf{C} = \mathbf{L}_C \mathbf{D}_C \mathbf{L}_C^T$. Since forming the inside term $\tilde{\mathbf{F}}^T \mathbf{P}_i^{-T} \Theta_i \mathbf{P}_i^{-1} \tilde{\mathbf{F}}$ explicitly is expensive, we suggest to use an iterative method to solve the linear system $\mathbf{C}\boldsymbol{x}_0 = \mathbf{b}_0 - \sum_{i=1}^{n} \mathbf{B}_i \boldsymbol{z}_i$, as steps (38) to (40) now can be replaced by an iterative solver. The advantage of an iterative solver is that the matrix-vector product $\mathbf{C}x$ can be performed cheaply and quickly by evaluating the value $\mathbf{C}\boldsymbol{x} = \Phi_0 \boldsymbol{x} - \sum_{j=1}^{n} \tilde{\mathbf{F}}^T \mathbf{P}_c^{-T} \Theta_i \mathbf{P}_i^{-1} \tilde{\mathbf{F}} \boldsymbol{x}$ from right to left.

To make an iterative method work, it is crucial to choose a good preconditioner. However, choosing a good preconditioner is still something of a "black art" and the best choice is heavily problem dependent. There are a few suggestions in the literature to cover SCOPF problem. Since $\mathbf{C} = \Phi_0 - \sum_{i=1}^{n} \mathbf{B}_i \Phi_i^{-1} \mathbf{B}_i^T$, involves a sum of all scenarios, an appealing idea is to construct the preconditioner by using a subset of scenarios instead of all of them. This preconditioner is build as

$$\mathbf{M} = \Phi_0 - \frac{n}{|\mathscr{M}|} \sum_{i \in \mathscr{M}} \mathbf{B}_i \Phi_i^{-1} \mathbf{B}_i^T, \tag{45}$$

where $\mathscr{M}$ is a set of indices of the contingency scenario. Note that Qiu et al (2005) just use the base case scenario $i = 0$ without any line failures to build the preconditioner while Petra and Anitescu (2010) use a random sample of contingencies.

When we evaluated the performance of the IPM implementations in the previous sections, we found that some contingency scenarios may bring very large entries into the Schur complement after several IPM iterations. In detail, sometimes the entries of the ith contingency contribution $\tilde{\mathbf{F}}^T \mathbf{P}_i^{-T} \Theta_i \mathbf{P}_i^{-1} \tilde{\mathbf{F}}$ are much larger than the entries in other different contingency scenarios. In fact, when we compute the factor $\Theta_i = \mathbf{X}_i^{-1} \mathbf{Z}_i$ in the saddle point matrix Φ_i, we found that one or several theta value $x_j^{-1} z_j$ are very large compared to others. These very large values are caused by a slack variable corresponding to an active line flow limit constraint in a particular contingency scenario. In addition, this specific contingency scenario typically contains this small slack variable through the whole IPM process even though the theta factor is changing iteration by iteration. We define these kind of contingencies as "active contingencies".

For example, recalling the three bus system, we found that in the contingency case one, when the line from bus one to bus two fails, flow on line from bus one to bus three is close to its upper bound, resulting in the corresponding slack variable to be close

to zero. This results in the corresponding theta value $x_i^{-1}z_i$ to be of order 10^6 times larger than the others. Because of this, it is crucial that these active contingencies should be included in the preconditioner. Now we suggest two different methods to build $\mathscr{M}$ based on active contingencies in (45).

In the first approach,we compute the set $\mathscr{M}$ in each IPM iteration. In each IPM iteration, we check all the contingencies and consider one to be active if and only if the ratio between the largest and the smallest theta factor is larger than 10^6. We do not save the previous contingency scenarios in the set M and reset the set before each IPM iteration. We name this approach the ”active” method.

In the second approach we update the set $\mathscr{M}$ when we find “active” contingencies. We use the same ratio, 10^6 to denote whether or not a contingency is active. However, this time once we add the indices of these contingencies into $\mathscr{M}$, we then never move them out from the set throughout the whole IPM procedure. Therefore, the number of the components of $\mathscr{M}$ is at least non-decreasing and could potentially grow in in pace with the IPM iterations. We call this approach the “cumulative” method.

5.1 Numerical Result

To test our approach, we use preconditioned GMRES as the iterative solver and use the same test problems as in the last section. The tolerances of IPM is 10^{-6}.

Firstly, we solve the model by GMRES with the preconditioner build only by the base case. Its numerical results are shown in the Table 4. Then, we use our two new approaches to build the preconditioner. The corresponding results are shown in the Table 5, where the second column denotes the total number of contingency scenarios and the column ‘FinalNoSce’ denotes the number of active scenarios we have found when IPM converges.

Table 4: Using base case as the preconditioner

	Base case preconditioned		
size	user time(s)	memory	iters
3	<0.01s	5.2MB	8
26	0.252s	7.4MB	13

It is worth mentioning that with the base case preconditioner, we can only solve the 3 and 26 buses systems. The test problems with size more than 56 buses fail because GMRES does not converge to the required tolerance. Even for the problems with 3 or 26 buses, we also need to change the tolerance of GMRES manually for several IPM steps to force GMRES to converge.

However, as shown in the Table 5, almost all the test problems can be solved with our preconditioner. This is true even if we set the GMRES tolerance fixed as 10^{-12}. In addition, we find that the cumulative method performs better than the active method. Most of all, the number of active scenarios is much smaller than the total number.

Table 5: GMRES with different preconditioners

		Cumulative			Active		
size	NoSce	time(s)	iters	FinalNoSce	time(s)	iters	FinalNoSce
3	3	<0.1	8	2	<0.1	8	1
26	41	0.25	13	2	0.25	13	1
56	80	1.30	18	6	1.29	18	3
100	181	8.08	20	7	9.50	28	5
200	371	44.36	27	9	44.98	27	6
300	566	177.49	35	21	168.89	33	16
400[1]	761	395.36	53	20	406.16	52	11
500	956	742.13	47	31	941.42	>70	NAN

For the largest test problem which contain 956 contingency scenarios, it only needs roughly 3% of the scenarios to build the preconditioner.

Comparing to the numerical results from the previous section, we find advantages in using an iterative method rather than a direct method. Compared to the Table 3, the cumulative method is faster than Method 3 and only requires approximately half the solution time of the default method. Moreover, its memory requirements are also much less than the default method and Method 3.

5.2 Conclusions

To summarize, this paper presents results on solving SCOPF problems by several structure-exploiting implementations of IPM and also by iterative method.

Firstly, we suggest a new way of organizing the linear algebra of IPM applied to SCOPF that avoids most factorizations. Then we turn to use iterative method to overcome the difficulty of forming the Schur complement matrix. In addition, we discuss and test several schemes of how to build an efficient and robust preconditioner. We introduce the concept of active contingencies and demonstrate how to use it to build the preconditioners. The numerical tests show us that with our best approach, the full IPM performance can be cut down by over half the time required, compared to the default method.

References

Altman A, Gondzio J (1993) Hopdm – a higher order primal-dual method for large scale linear programming. European Journal of Operational Research 66(1):158 – 160, DOI DOI:10.1016/0377-2217(93)90214-8

[1] We find OOPS cannot solve this problem before exceeding its iteration limit of 70, if we use gcc with compiler option -O2 or -O3. Once we turn to use option -g, OOPS can solve this problem in the 53th IPM iteration. If we consider that OOPS can solve this problem in 53 iterations with option -O2, it should require $522.17/70 * 53 = 395.36$ seconds roughly.

Capitanescu F, Glavic M, Ernst D, Wehenkel L (2007) Interior-point based algorithms for the solution of optimal power flow problems. Electric Power Systems Research 77(5-6):508 – 517, DOI DOI:10.1016/j.epsr.2006.05.003

Fourer R, MGay D, Wkernighan B (2003) AMPL: A Modeling Language for Mathematical Programming, 2nd Ed. Curt Hinrichs, Canada

Gondzio J, Grothey A (2009) Exploiting structure in parallel implementation of interior point methods for optimization. Comput Manag Sci 6(2):135–160

Karoui K, Platbrood L, Crisciu H, Waltz R (2008) New large-scale security constrained optimal power flow program using a new interior point algorithm. In: Electricity Market, 2008. EEM 2008. 5th International Conference on European, pp 1 –6, DOI 10.1109/EEM.2008.4579069

Petra C, Anitescu M (2010) A preconditioning technique for schur complement systems arising in stochastic optimization. Tech. rep., Argonne National Laboratory, Argonne, IL, to appear in Computational Optimization and Applications. Preprint ANL/MCS-P1748-0510

Qiu W, Flueck A, Tu F (2005) A new parallel algorithm for security constrained optimal power flow with a nonlinear interior point method. In: Power Engineering Society General Meeting, 2005. IEEE, pp 447 – 453 Vol. 1, DOI 10.1109/PES.2005.1489574

Quintana V, Torres G, Medina-Palomo J (2000) Interior-point methods and their applications to power systems: a classification of publications and software codes. Power Systems, IEEE Transactions on 15(1):170 –176, DOI 10.1109/59.852117

Wei H, Sasaki H, Yokoyama R (1996) An application of interior point quadratic programming algorithm to power system optimization problems. Power Systems, IEEE Transactions on 11(1):260 –266, DOI 10.1109/59.486104

Wright SJ (1997) Primal-dual Interior-Point Methods. Society for Industrial and Applied Mathematics (SIAM), Philadelphia