UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

# Distributed Algorithm for Optimal Power Flow on Multiphase Distribution Networks

Dissertation presented by
**Alexandre LATERRE**

for obtaining the Master's degree in
**Mathematical Engineering**

Supervisor(s)
**Anthony PAPAVASILIOU, Gauthier DE MAERE D'AERTRYCKE**

Reader(s)
**Emmanuel DE JAEGER**

Academic year 2015-2016

# Table of Contents

# Notations

| | |
|---|---|
| $A$ | A matrix, usually of size $m \times n$ unless stated otherwise. |
| $A_{ij}$ | The $(i,\ j)$ entry of the matrix A. |
| $A^T$ | The transpose of the matrix $A$. |
| $A^*$ | The hermitian of the matrix $A$ (conjugate transpose). |
| $\mathrm{Tr}(A)$ | The trace of a square matrix $A \in \mathbb{C}^{n\times n}$, defined by $\mathrm{Tr}(A) := \sum_{i=1}^{N} A_{ii}$. |
| $\mathrm{diag}(A)$ | The vector composed of $A$'s diagonal elements, if $A \in \mathbb{C}^{n\times n}$, $\mathrm{diag}(A) \in \mathbb{C}^n$. |
| $A \succeq 0$ | The hermitian matrix $A \in \mathbb{C}^{n\times n}$ is positive semi-definite, i.e $z^T A z \geq 0$, $\forall z \in \mathbb{C}^n$. |
| $I$ | The identity matrix. |
| $\mathbf{0}$ | The zero matrix, all its entries are zero. |
| $\mathrm{Re}\{z\}$ | The real part of the complex number $z$. |
| $\mathrm{Im}\{z\}$ | The imaginary part of the complex number $z$. |
| $\lvert z\rvert$ | Magnitude of the complex number $z$. |
| $\angle z$ | Argument of the complex number $z$. |
| $x = y$ | Two complex numbers are equal if and only if both their real and imaginary parts are equal. |
| $\langle x, y\rangle$ | The inner product of two matrices (vectors), $x,\ y \in \mathbb{C}^{m\times n}$, defined by $\mathrm{Re}\{\mathrm{Tr}(x^* y)\}$. |
| $\lVert x\rVert_2$ | The norm of a matrix (vector) $x \in \mathbb{C}^{m\times n}$ is defined as $\sqrt{\langle x, x\rangle}$. |
| $\lvert\mathcal{N}\rvert$ | The cardinality of the set $\mathcal{N}$, i.e. the number of elements of the set $\mathcal{N}$. |

# Introduction

The optimal power flow (OPF) problem is fundamental in power system operations and planning because it underlies many applications such as economic dispatch, unit commitment, state estimation, volt/var control, demand response, etc. This problem seeks to control power generation/demand to optimise certain objectives such as minimising the generation cost or power loss in the network, subject to power flow equations and operational constraints. This optimisation problem has been extensively studied since Carpentier's first formulation in 1962 [Car62]. Numerous algorithms have been proposed for solving this highly non-convex problem, including linear programming, quadratic programming, nonlinear programming, interior point methods, neural networks, fuzzy logic, genetic algorithms, etc.

It's becoming increasingly important for distribution networks due to the emergence of high penetration of distributed generation and controllable loads such as electric vehicles. This continued growth of highly volatile renewable sources on distribution systems calls for real-time feedback control. Solving the OPF problems in such environment has at least two challenges :

- First, the exact solution to the OPF problem is very difficult (NP-hard) to obtain for general networks due to its non-convex constraints, namely magnitude constraints on complex-valued bus voltages, and non-linear equality constraints corresponding to Kirchhoff's Laws which govern power flows in electrical networks. There are generally three ways to tackle the non-convexity: (i) use linear approximations of the power flow equations; (ii) employ nonlinear solver to find local optimum; (iii) exploit convex relaxations of the non-convex constraints. After a short discussion about the first two approaches, the rest of this paper will be focused on the third.
- Secondly, most algorithms in the literature are centralised and meant for applications in today's energy management systems (central schedule of a relatively small number of generators). However, in future, the growing number of controllable devices will make a central approach impracticable because of its computation cost. In this thesis, a fully decentralised algorithm able to solve the OPF problem is proposed. Through optimisation decomposition, the original OPF problem is decomposed into several local subproblems that can be solved simultaneously.

Due to these challenges, the current practice in the electricity industry is to use the so-called DC-OPF approximation. In contrast, the original non-convex OPF is usually called the AC-OPF (alternating current). DC-OPF uses a linearization of AC-OPF by exploiting some physical properties of the power flows in typical power systems, such as tight bounds on voltage magnitudes at buses and small voltage angle differences between buses. However, such an approximation completely ignores important aspects of power flow physics, such as the reactive power and voltage magnitude. To partially remedy this drawback, the current practice is to solve the DC-OPF problem and then to solve a set of power flow equations with the DC-OPF solution to compute feasible reactive powers and voltages. However,

it is clear such an approach cannot guarantee any optimality of the AC power flow solution obtained. The rest of this paper will be focused on the convex relaxations of the non-convex constraints of AC-OPF, hereafter denoted by OPF. The structure of this thesis is the following:

- The first part focuses on recent advances in the convex relaxation of the optimal power flow problem in the special case of distribution networks. Most of those works assume a single-phase network while distribution networks are typically multiphase and unbalanced [Ker01b]. Although the single-phase formulations will not be directly used in the sequel, it provides some useful background for the understanding of the multi-phase formulation.

- The second part of this thesis is dedicated to the elaboration of a decentralised algorithm based on the alternating direction method of multipliers (ADMM), briefly described. Unlike existing approaches, the problem structure is exploited in order to decompose the OPF problem in such a way that the subproblems at each iteration reduce to either a closed-form solution or an eigen-decomposition of a $6 \times 6$ hermitian matrix; which significantly reduce the computational time. Moreover, since the method is completely decentralized, and needs no global coordination other than synchronizing iterations, the problem can be solved extremely efficiently in parallel.

- In the third part of this thesis, we demonstrate the effectiveness of the algorithm by testing it on made-up networks and real distribution networks. Specially, we show that the proposed convex relaxation of the optimal power flow problem is generally exact for the IEEE test feeders.

- Finally, we briefly review the main results presented in this work and discuss some directions for future research (Local Stopping Criteria, fast ADMM, multi-step programming).

# Part I

# Formulation of the Optimal Power Flow Problem

# 1 | The OPF Problem on Single-Phase Distribution Networks

In this chapter, we describe the formulation of the constraints which govern power flows in electrical networks in the case of single-phase distribution networks. We show that two different models can be used to formulate the OPF problem, namely the bus injection model and the branch flow model. In particular, the tow formulations are showed to be strictly equivalent.

It's interesting to consider both models because some relaxations are much easier to formulate in one model than the other. For instance, the semidefinite relaxation has a much cleaner formulation in the bus injection model. Whereas the branch flow model has a convenient recursive structure that allow a more efficient computation [CB90]. Moreover, it also play a crucial role in proving sufficient conditions for the exactness of the convex relaxation (see e.g.[GLTL12]).

## 1.1 Bus Injection Model

The bus injection model aims at formulating the OPF problem according to nodal variables, such as the voltage and the net power injection (generation minus load). In this section, we describe two different relaxations, a semidefinite relaxation and a second-order cone relaxation of the OPF problem [Low13]. We show the equivalence of those two relaxations and point out that in case of radial networks, one should always use the second-order cone relaxation [BLTH14].

### 1.1.1 Formulation of the OPF problem

A distribution network is composed of buses and branches connecting these buses. Moreover, it's assumed to be radial, i.e. has a tree topology. The root is called the substation node and holds the responsibility for drawing the power from the the transmission network to the distribution network for power balance. The substation bus is indexed by 0 and the others buses from 1 to $n$. Let's denote $\mathcal{N}$ the set of all buses and $\mathcal{N}^+$ the set of all buses except the substation node. The set of all lines is $\mathcal{E}$. We say that $(i,j) \in \mathcal{E}$ if $i \to j$. And if $i \to j$, or $i \to j$, then $i \sim j$, otherwise $i \nsim j$.

For each line in the network $(i,j) \in \mathcal{E}$, let $y_{ij} = g_{ij} - ib_{ij}$ denote its admittance and $z_{ij} = r_{ij} + ix_{ij}$ its impedance such that $y_{ij}z_{ij} = 1$. For each bus $i \in \mathcal{N}$, let $V_i$ denote its voltage and $I_i$ denote its current injection. A bus $i \in \mathcal{N}$ can have a generator, a load, both or neither. The spot loads are specified and the generations are variables to be determined. Let $s_i = p_i + iq_i$ denote the power injection at node $i$ where $p_i$ and $q_i$ are the active and reactive power injections respectively (generation minus load). The substation node is assumed to have a fixed voltage and a flexible power injection. A letter without

subscript denotes a vector of the corresponding quantity, e.g. $V = (V_1, \dots, V_n)^T$.

Given a network $(\mathcal{N}, \mathcal{E})$, the admittances $y$ and the substation voltage $V_0$, the other variables $(s, V, I, s_0)$ must satisfy the following constraints :

- Current balance and Ohm's law :

$$I_i = \sum_{j:j\sim i} (V_i - V_j)\, y_{ij} \qquad \forall i \in \mathcal{N}$$

- Power Balance :

$$s_i = V_i I_i^* \qquad \forall i \in \mathcal{N}$$

Those two sets of equations can be combined to get rid off the variables $I_i$.

$$s_i = V_i \sum_{j:j\sim i} \left(V_i^* - V_j^*\right) y_{ij}^* \qquad \forall i \in \mathcal{N} \tag{1.1}$$

(1.1) is referred as the power flow equations. For each bus $i \in \mathcal{N}$, there are two operational constraints. First, the constraint on the net power injection $s_i$ can be captured by some feasible power injection set $\mathcal{I}_i$, such that $s_i \in \mathcal{I}_i$. A common set is for example:

$$\mathcal{I}_1 := \{p + iq \in \mathbb{C} \mid \underline{p}_i \le p \le \bar{p}_i,\ \underline{q}_i \le q \le \bar{q}_i, \}$$

In that case, if we combine these constraints with (1.1), we obtain :

$$\underline{s}_i \le V_i \sum_{j:j\sim i} \left(V_i^* - V_j^*\right) y_{ij}^* \le \bar{s}_i \qquad \forall i \in \mathcal{N} \tag{1.2}$$

It's usually assumed that there is no limit on the power injection at the substation node, i.e. $-\underline{s}_0 = \bar{s}_0 = +\infty$. However, such assumption is not essential to our model. Secondly, the voltage magnitude needs to be maintained within a predefined range. This is captured by specifying lower and upper bounds on the voltage magnitude, i.e.

$$\underline{V}_i \le |V_i| \le \bar{V}_i \qquad \forall i \in \mathcal{N} \tag{1.3}$$

It's common practice to allow a 5% voltage deviation from the nominal value $V_0^{ref}$. Since the voltage at the substation node is assumed to be fixed at the nominal value, $\underline{V}_0 = \bar{V}_0 = V_0^{ref}$.

The constraints (1.3), (1.2) define a feasible set of the optimal power flow problem :

$$\mathbb{V} := \{V \mid V \text{ satisfies (1.3) and (1.2)}\}$$

Besides the enforcement of the aforementioned constraints, the optimal power flow aims at minimising an objective function. Typical objective functions include generation cost or total power loss. Hereafter, We assume there exists for each node $i \in \mathcal{N}$, a real-valued function $f_i(s_i)$ defined on $\mathbb{R}$ which represents the local objective of node $i$. Notice that those functions depend only on the power injection $s_i$ since the power loss and the generation cost can be expressed using only the power injection. Then

the total objective function is given by

$$C(s, s_0) = \sum_{i \in \mathcal{N}} f_i \left( \mathrm{Re}\{s_i\} \right)$$

Since the power injection can be expressed in terms of the voltage $V$. The cost function $C$ is in reality a function of the voltage $V$. For exemple, if we want to minimize the real power loss :

$$C(V) = \sum_{i \in \mathcal{N}} \mathrm{Re}\{s_i\} = \sum_{i \in \mathcal{N}} \sum_{j : i \sim j} \mathrm{Re}\Big\{ V_i \left( V_i^* - V_j^* \right) y_{ij}^* \Big\}$$

The OPF problem in the bus injection model can be formulated using the aforementioned constraints and objective function.

**BIM-OPF :**

$$\min_{V} \quad C(V) \tag{1.4a}$$

$$\text{s.t.} \quad V \in \mathbb{V} \tag{1.4b}$$

### 1.1.2 Convex Relaxation of The OPF Problem

Since the functions $f_i$ are usually assumed to be convex, the challenge in solving the OPF problem comes from the non-convex quadratic equality constraints (1.1). In this subsection, we describe a convex relaxation of the optimal power flow problem. To do so, we enlarge the feasible set of OPF to a convex set and characterise it in terms of partial matrices. These characterisations lead naturally to an SDP and SOCP relaxation of the OPF problem.

#### Mathematical Tools

Fix an undirected graph $G = (\mathcal{N}, \mathcal{E})$. A **$G$-partial matrix** $X_G$ (or a partial matrix if $G$ is clear from the context), is a collection of complex numbers such that

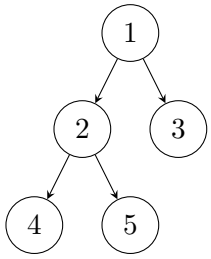
$$X_G := \left( [X_G]_{ii} \in \mathbb{C} \ \forall i \in \mathcal{N}, \ [X_G]_{ij} \in \mathbb{C} \ \forall (i,j) \in \mathcal{E} \right)$$

One can treat a partial matrix $X_G$ as entries of an $n \times n$ matrix $X$ whose entries $X_{ij}$ are unspecified if $(i,j) \notin \mathcal{E}$. Given a partial matrix $X_G$, we call an $n \times n$ matrix $X$, a **completion** of $X_G$ if $X_{ii} = [X_G]_{ii}$, $i \in \mathcal{N}$ and $X_{ij} = [X_G]_{ij}$, $(i,j) \in \mathcal{E}$, i.e. $X$ agrees with $X_G$ on $G$.

Consider any $n \times n$ matrix $X$. Given $k \leq n$ nodes, let $X(n_1, \ldots, n_k)$ denote the $k \times k$ principal sub matrix of $X$ defined by:

$$[X(n_1, \ldots, n_k)]_{ij} := X_{ij} \qquad \forall i, j \in \{n_1, \ldots, n_k\}$$

In particular, any maximal clique $q = (n_1, \ldots, n_k)$ of $G$ with $k$ nodes defines a fully specified $k \times k$ principal sub-matrix denoted by $X(q)$. Figure (1.1) shows an example of a partial matrix and a principal sub-matrix.




| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **1** | ✓ | ✓ | ✓ | ✗ | ✗ |
| **2** | ✓ | ✓ | ✗ | ✓ | ✓ |
| **3** | ✓ | ✗ | ✓ | ✗ | ✗ |
| **4** | ✗ | ✓ | ✗ | ✓ | ✗ |
| **5** | ✗ | ✓ | ✗ | ✗ | ✓ |

| | 1 | 3 |
|---|---|---|
| **1** | ✓ | ✓ |
| **3** | ✓ | ✓ |

Figure 1.1: Example of a partial matrix $W_G$ (center) associated with the graph $G$ (left). The matrix on the right is the principal submatrix $W_G(C)$ for the clique $C := \{1, 3\}$.

We can extend the notion of hermitian to partial matrices. We say a partial matrix $W_G$ is hermitian, denoted by $X_G = X_G^*$, if $[X_G]_{ii} = [X_G]_{ii}^*$, $i \in \mathcal{N}$ and $[X_G]_{ij} = [X_G]_{ji}^*$, $\forall (i,j) \in \mathcal{E}$. The extension of the notion of positive-semidefinite (psd) and rank-1 matrices will be explained later.

**SDP Relaxation**

Let's Define a partial matrix $W_G$ such that

$$[W_G]_{ij} = V_i V_j^* \qquad i \sim j \text{ or } i = j$$

The constraints of the OPF problem (1.2), (1.3) can be rewritten in terms of the partial matrix $W_G$

$$\underline{s}_i \leq \sum_{j:j\sim i} \left( [W_G]_{ii} - [W_G]_{ij} \right) y_{ij}^* \leq \bar{s}_i \qquad \forall i \in \mathcal{N} \tag{1.5a}$$

$$\underline{V}_i^2 \leq [W_G]_{ii} \leq \bar{V}_i^2 \qquad \forall i \in \mathcal{N} \tag{1.5b}$$

It's clear that any completion $W$ of $W_G$ would also satisfy the constraints (1.5a)-(1.5b), since $y_{ij} = 0$, if $(i,j) \notin \mathcal{E}$. Moreover if $W = VV^*$, with $V$ a voltage vector, then $W$ must obviously be a rank-1 matrix but also positive-semidefinite. Indeed, by definition, a matrix $W \in \mathbb{C}^{n\times n}$ is psd if $\forall x \in \mathbb{C}^n$,

$$x^* M x \geq 0$$

if $W = VV^*$, then $x^* M x = (V^* x)^* (V^* x) \geq 0$.

The OPF problem can be rewritten in terms of a $n \times n$ Hermitian matrix $W$ and its partial matrix $W_G$ defined on $\mathcal{G}$.

**Problem $\mathcal{P}_1$**

$$\min_{W} \quad C(W_G) \tag{1.6a}$$

$$\text{s.t.} \quad W_G \text{ satisfy (1.5a) and (1.5b)} \tag{1.6b}$$

$$W \succeq 0, \quad \text{rank } W = 1 \tag{1.6c}$$

Given $V \in \mathbb{V}$, $W = VV^*$ is feasible for $\mathcal{P}_1$. In the same way, if $W$ is feasible for $\mathcal{P}_1$, then it has a unique spectral decomposition $W = VV^*$ with $V \in \mathbb{V}$. Problem $\mathcal{P}_1$ is therefore equivalent to the OPF problem. Unfortunately $\mathcal{P}_1$ is an rank-constrained SDP and then hard to solve (NP-hard). The non-convex rank constraint can be relaxed to obtain the following **SDP** relaxation :

**Problem $\mathcal{R}_1$**

$$\min_{W} \quad C(W_G) \tag{1.7a}$$
$$\text{s.t.} \quad W_G \text{ satisfy (1.5a) and (1.5b)} \tag{1.7b}$$
$$W \succeq 0 \tag{1.7c}$$

This relaxation can be solved in polynomial time using for example an interior point method. Denote $W^\star$, an optimal solution of $\mathcal{R}_1$. If $W^\star$ is rank-1 then it also solves $\mathcal{P}_1$. We say the relaxation $\mathcal{R}_1$ is exact with respect to $\mathcal{P}_1$ if there exists an optimal solution of $R_1$ that satisfies the rank constraint in $\mathcal{P}_1$. Notice that if the relaxation $\mathcal{R}_1$ has multiple solutions, we say that the relaxation is exact as long as there exists at least one rank-1 solution of $\mathcal{R}_1$.

### SOCP relaxation

Instead of working with a completion matrix $W$ of $W_G$, we would like to seek additional conditions on the partial matrix that guarantee that it has a psd rank-1 completion $W$ from which a voltage vector $V$ can be recovered.

To this end, we need to extend the notion of psd to partial matrices. Denote by $W(\mathcal{K})$ a principal sub-matrix of an $n \times n$ matrix $W$ for a set $\mathcal{K} \subseteq \{1, \dots, n\}$ containing the rows and columns considered in the sub-matrix is given by:

$$W(\mathcal{K}) := P_\mathcal{K}\, W\, P_\mathcal{K}^T$$

where $P_\mathcal{K}$ is a diagonal matrix where the rows and columns not contained in $\mathcal{K}$ have been removed. For instance, $\mathcal{K} = \{1, 2, 4\}$,

$$W_\mathcal{K} = P_\mathcal{K}\, W\, P_\mathcal{K}^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 4 \\ 5 & 6 & 8 \\ 13 & 14 & 16 \end{pmatrix}$$

Let's show that a $n \times n$ matrix W is psd if and only of all its principal sub-matrices (including $W$ itself) are psd. By definition, $W$ is a psd matrix if for all vector $x \in \mathbb{C}^n$, $0 \leq x^* W x$. Expressed in terms of the partial matrix $W(\mathcal{K})$, we get :

$$x^* W x = x^*\, P_\mathcal{K}^T\, W(\mathcal{K})\, P_\mathcal{K}\, x = (P_\mathcal{K}\, x)^*\, W(\mathcal{K})\, (P_\mathcal{K}\, x) = y^*\, W(\mathcal{K})\, y \geq 0$$

This result holds for every $y \in \mathbb{C}^{|\mathcal{K}|}$ since $y$ can always be expressed as $P_\mathcal{K}\, x$ with $x \in \mathbb{C}^n$ and for every partition $\mathcal{K} \subseteq \{1, \dots, n\}$.

Thanks to this result, we extend the notion of psd to partial matrices. A partial matrix is psd if all its "principal sub-matrices" that are fully specified are psd. Formally, $W_G$ is psd, denoted by $W_G \succeq 0$, if $W_G(C) \succeq 0$ for all clique $C$ of $G$. In the same way, we say that a partial matrix is rank-1 if $X_G(C)$ is rank-1 for all maximal clique $C$ of $G$. Notice that for a tree, the maximal cardinality of a clique is two, otherwise there would be cycles in the graph. Therefore all the principal sub-matrices of a radial network are $2 \times 2$ matrices.

We say that a partial matrix $W_G$ satisfies the cycle condition if, for every cycle in $G$,

$$\sum_{(i,j)\in \text{ Cycle}} \angle[W_G]_{ij} = 0 \tag{1.8}$$

The following theorem characterises when a definite matrix $W$ is rank-1 in terms of its restriction on $G$, i.e. the partial matrix $W_G$ (proof available in [SLC12]).

**Theorem 1.** *Fix a graph $G$ on $n$ nodes. Given an $n \times n$ positive or negative semidefinite matrix $W$, the following are equivalent:*

1. *rank $W = 1$*
2. *rank $W_G(i,j) = 1$ for all $(i,j) \in \mathcal{E}$ and the partial matrix $W_G$ satisfies the cycle condition* (1.8).

The rank condition is a property for the whole matrix $W$. Theorem 1 characterises this condition in terms of the partial matrix $W_G$, defined on the graph $G$. This is important because $W_G$ is typically much smaller than $W$ and can be much more efficiently computed for large sparse networks. Note also that for radial networks, the cycle condition always holds since there is no cycle in a tree.

As we now explain, theorem 1 allows us to solve simpler problems in terms of partial matrices. For any $e = (i,j) \in \mathcal{E}$, let's define $W_G(e)$ as the $2 \times 2$ principal sub-matrix of $W_G$ defined by the clique $e$.

**Problem $\mathcal{P}_2$**

$$\begin{aligned} \min_{W_G} \quad & C(W_G) && (1.9a)\\ \text{s.t.} \quad & W_G \text{ satisfy (1.5a), (1.5b) and (1.8)} && (1.9b)\\ & W_G(e) \succeq 0,\ \text{rank } W_G(e) = 1 \qquad \forall e \in \mathcal{E} && (1.9c) \end{aligned}$$

Fortunately, the cycle condition always holds for radial networks. Nevertheless, the problem still remains non-convex due to the rank constraint. If we relax it, we obtain the following relaxation :

**Problem $\mathcal{R}_2$**

$$\begin{aligned} \min_{W_G} \quad & C(W_G) && (1.10a)\\ \text{s.t.} \quad & W_G \text{ satisfy (1.5a) and (1.5b)} && (1.10b)\\ & W_G(e) \succeq 0 \qquad \forall e \in \mathcal{E} && (1.10c) \end{aligned}$$

Moreover $\forall e = (i,j) \in \mathcal{E}$, and for an Hermitian matrix $W_G$, the condition $W_G(e) \succeq 0$ can be reformulated using the Sylvester's Criterion for hermitian matrices :

$$0 \preceq W_G(e) = \begin{pmatrix} [W_G]_{ii} & [W_G]_{ij} \\ [W_G]_{ij}^* & [W_G]_{jj} \end{pmatrix} \quad \Longleftrightarrow \quad \begin{cases} [W_G]_{ii} \geq 0, \quad [W_G]_{jj} \geq 0 \\ [W_G]_{ii}\,[W_G]_{jj} \geq |[W_G]_{ij}|^2 \end{cases}$$

This is a second-order cone constraint, and hence the problem (1.10) can be solved as an **SOCP**. If an optimal solution of $\mathcal{R}_2$ satisfies that $W_G(e)$ is rank-1 for all $e \in \mathcal{E}$, then the relaxation is said to be exact with respect to $\mathcal{P}_2$.

**Equivalence of the two relaxations**

The following corollary links the two formulations in terms of the optimal value of their objective function [Low13].

**Corollary 1.1.** *Let $C^*, C_{sdp}, C_{socp}$ be the optimal values of OPF, $\mathcal{R}_1$ (SDP relaxation) and $\mathcal{R}_2$ (SOCP relaxation) respectively.*

1. *If $G$ is radial, then $C^* \geq C_{sdp} = C_{socp}$*
2. *If $G$ has cycles, then $C^* \geq C_{sdp} \geq C_{socp}$*

Indeed, for mesh networks, the SOCP relaxation requires the relaxation of both the cycle condition and the rank constraint whereas the SDP relaxation only requires the relaxation of the rank constraint. Therefore, since the feasible set of the SOCP relaxation is larger than the feasible set of the SDP relaxation, we get that $C_{sdp} \geq C_{socp}$.

In the special case of radial networks, the cycle condition always holds, and the two formulations relaxe the same rank constraint. Hence, they are strictly equivalent for radial networks, and we obtain that $C_{sdp} = C_{socp}$.

**Computational aspect**

Although $\mathcal{R}_1$ (SDP) and $\mathcal{R}_2$ (SOCP) are convex and hence can be solved in polynomial time, SOCP usually requires a much smaller computational effort than SDP for large sparse networks, like distribution networks. Indeed, $G$ is a subgraph of the complete graph defined on $\mathcal{N}$, and hence the number of complex variables is the smallest in $\mathcal{R}_2$ ($|W_G|$) and the largest in $\mathcal{R}_1$ ($|\mathcal{N}|^2$).

Most importantly, corollary 1.1 suggests that, when $G$ is a tree, we should always solve the SOCP formulation. When $G$ has cycles, there is a tradeoff between computational effort and exactness in deciding between solving the SOCP or the SDP formulation.

### 1.1.3 How to recover the voltage vector

Consider the optimization problem $\mathcal{P}_1$ (1.6), so that any optimal solution $W^\star$ is a psd rank-1 matrix. We can show that we can recover a voltage vector $V^\star$ from $W^\star$. This voltage vector is unique and an optimal solution of BIM-OPF(1.4).

It's straightforward that for any $V \in \mathbb{V}$, the point $\psi(V) = W = VV^*$ is feasible for (1.6). It remains to prove that the map $\psi$ is bijective, i.e injective and surjective.

**Injective:** Let's define the two voltage vectors $V$ and $V'$. if $\psi(V) = \psi(V')$ then $V_i V_j^* = V_i' V_j'^*$ for $i \sim j$. Hence, $V_i = V_i'$ implies $V_j = V_j'$ if $i \sim j$. But since $V_0 = V_0'$ and the network is connected, $V = V'$.

**Surjective:** We can prove that for any feasible solution $W$ of (1.6), there exists a $V$ such that $W_{ij} = V_i V_j^*$ for $i = j$ and $i \sim j$. Such $V$ is given by :

$$V_i = \sqrt{W_{ii}} \exp\left( i \left( \angle V_0 - \sum_{(j,k)\in\mathcal{P}_i} \angle W_{jk} \right) \right) \qquad \forall i \in \mathcal{N} \tag{1.11}$$

where $\mathcal{P}_i$ denotes the unique path form the substation bus 0 to the bus $i$. It's not difficult to verify that such $V$ satisfies $\psi(V) = W$, which complete the proof.

■

If the relaxation $\mathcal{R}_2$ doesn't yield an exact solution of BIM-OPF (1.6), i.e. the optimal solution $W$ of $\mathcal{R}_2$ is not a rank-1 matrix, the aforementioned recovering algorithm doesn't apply. To partially remedy this drawback, a solution could be to project the solution $W$ onto its closest rank-1 matrix $\tilde{W}$ defined by

$$\tilde{W} := U_{1:}\, \Sigma_{11}\, V_{1:}^*$$

where the matrix $U$ and $V$ are unitary matrices coming from the singular value decomposition of $W$. $\Sigma$ is a diagonal matrix containing the singular values in descending order. Then a voltage vector $V$ can be recovered from $\tilde{W}$ using formula (1.11). Although such solution is not guaranteed to be feasible for (1.4), i.e. $V \notin \mathbb{V}$, we can employ a nonlinear solver with $V$ as starting point to find a local optimum of the BIM-OPF problem. However, it is clear that such approach cannot guarantee any global optimality.

## 1.2 Branch Flow Model

The branch Flow Model (BFM) focuses on both nodal variables and branch variables such as the current and the power on the branches. It has received far less attention than the Bus Injection Model, and has been used mainly for the analysis of distribution networks because of its convenient recursive structure. Nevertheless, it plays an important role in proving sufficient conditions for efficient recovery of the optimal solution of the OPF problem from its convex relaxations. Moreover, the variables in the BFM correspond to physical quantities such as the branch power and current, which can be more convenient depending on the application.

### 1.2.1 Formulation of the OPF problem

The BFM of [FL12] adopts a directed connected graph $\mathcal{G} := (\mathcal{N}, \mathcal{E})$ to represent a power network where each node in $\mathcal{N}$ represents a bus and each edge $e \in \mathcal{E}$ represents a line connecting two buses. The orientations of the edges are arbitrary.

As before, denote $V_i$ as the complex voltage at node $i \in \mathcal{N}$, $s_i$ as the net power injection (generation minus load) at node $i \in \mathcal{N}$. For each edge $e = (i, j) \in \mathcal{E}$, let $z_{ij}$ be the complex impedance on the line $(i, j)$, let $I_{ij}$ be the complex current from node $i$ to $j$, and $S_{ij} = P_{ij} + iQ_{ij}$ the sending-end complex power from buses $i$ to $j$, where $P_{ij}$ is the active power and $Q_{ij}$ is the reactive power.

The Branch flow model is defined by the following set of equations :

$$V_i - V_j = z_{ij} I_{ij} \qquad \forall (i,j) \in \mathcal{E} \tag{1.12a}$$

$$S_{ij} = V_i \, I_{ij}^* \qquad \forall (i,j) \in \mathcal{E} \tag{1.12b}$$

$$s_j = \sum_{k:j\to k} S_{jk} - \sum_{i:i\to j} \left( S_{ij} - z_{ij} |I_{ij}|^2 \right) \qquad \forall j \in \mathcal{N} \tag{1.12c}$$

where (1.12a) describes Ohm's Law, (1.12b) the branch power and (1.12c) the power balance. And where $z_{ij}|I_{ij}|^2$ represents the line loss so that $S_{ij} - z_{ij}|I_{ij}|^2$ is the receiving-end complex power at bus $j$ from $i$. The power injection $s_j$ must satisfy :

$$\underline{s}_j \leq s_j \leq \bar{s}_j \qquad \forall j \in \mathcal{N} \tag{1.13}$$

where $\underline{s}_j$ and $\bar{s}_j$ are limits on the net power injection at node $j$. Note that we assume there is no limit on the net power injection at the substation node for power balance, i.e. $-\underline{s}_0 = \bar{s}_0 = +\infty$. We can combine these constraints with the power balance constraints (1.12c) :

$$\underline{s}_j \leq \sum_{k:j\to k} S_{jk} - \sum_{i:i\to j} \left( S_{ij} - z_{ij} |I_{ij}|^2 \right) \leq \bar{s}_j \qquad \forall j \in \mathcal{N} \tag{1.14}$$

Finally, the voltage magnitude must be contained within a predefined range :

$$\underline{V}_j \leq V_j \leq \bar{V}_j \qquad \forall j \in \mathcal{N} \tag{1.15}$$

where $\underline{V}_j$ and $\bar{V}_j$ are given. And as in the bus injection model, the voltage of the substation node is fixed at a given value, i.e. $\underline{V}_0 = \bar{V}_0 = V_0^{ref}$. Denote $x := (S, I, V) \in \mathbb{C}^{3|\mathcal{N}|-2}$ the variables of the Branch Flow Model.

The objective function $C$ of the OPF problem can be expressed in terms of $x$. For example, if we want to minimize the total power loss :

$$C(x) = \sum_{j\in\mathcal{N}} \mathrm{Re}\{s_j\} = \sum_{j\in\mathcal{N}} \mathrm{Re}\left\{ \sum_{k:j\to k} S_{jk} - \sum_{i:i\to j} \left( S_{ij} - z_{ij} |I_{ij}|^2 \right) \right\}$$

.

The OPF problem can be written in the Branch Flow model by :

**BFM-OPF**

$$\min_x \quad C(x) \tag{1.16a}$$

$$\text{s.t.} \quad x \text{ satisfies (1.12a), (1.12b), (1.14), and (1.15)} \tag{1.16b}$$

Since (1.12c) are quadratic equality constraints, the feasible set is generally non-convex. And the OPF is as before a non-convex problem.

### 1.2.2 Convex Relaxation of the OPF Problem

An SOCP relaxation of (1.16) is developed in [FL12]. It consists of two steps. Firstly, we transform (1.12a) and (1.12b) in order to remove the phase angles from the complex voltages $V$ and currents $I$.

Ohm's law can be transformed into

$$\begin{aligned} V_j &= V_i - z_{ij} I_{ij} && \forall (i,j) \in \mathcal{E} \\ V_j V_j^* &= \left(V_i - z_{ij} I_{ij}\right)\left(V_i - z_{ij} I_{ij}\right)^* && \forall (i,j) \in \mathcal{E} \\ v_j &= v_i - V_i I_{ij}^* z_{ij}^* - z_{ij} I_{ij} V_i^* + z_{ij} I_{ij} I_{ij}^* z_{ij}^* && \forall (i,j) \in \mathcal{E} \\ v_j &= v_i - S_{ij} z_{ij}^* - z_{ij} S_{ij}^* + z_{ij} l_{ij} z_{ij}^* && \forall (i,j) \in \mathcal{E} \\ v_j &= v_i - 2\,\mathrm{Re}\Big\{ z_{ij}^* S_{ij} \Big\} + z_{ij} l_{ij} z_{ij}^* && \forall (i,j) \in \mathcal{E} \end{aligned} \tag{1.17}$$

where we define $l_{ij} = I_{ij}\, I_{ij}^*$, $\forall (i,j) \in \mathcal{E}$, and $v_i = V_i V_i^*$, $\forall i \in \mathcal{N}$. The constraints (1.12b) can be transformed into

$$\begin{aligned} S_{ij} &= V_i\, I_{ij}^* && \forall (i,j) \in \mathcal{E} \\ S_{ij} S_{ij}^* &= V_i\, I_{ij}^* I_{ij}\, V_i^* && \forall (i,j) \in \mathcal{E} \\ |S_{ij}|^2 &= v_i l_{ij} && \forall (i,j) \in \mathcal{E} \end{aligned} \tag{1.18}$$

The quadratic equalities (1.18) are still non-convex; but we can relax them to inequalities:

$$|S_{ij}|^2 \le v_i l_{ij} \qquad \forall (i,j) \in \mathcal{E} \tag{1.19}$$

Let $x := (S, l, v)$ denote the new variables of the optimal power flow problem. From these developments, we can define two following relaxations (non-convex and convex):

**Problem** $R_{BFM}^{nc}$

$$\min_x \quad C(x) \tag{1.20a}$$

$$\text{s.t.} \quad x \text{ satisfies (1.14), (1.15), (1.17), and (1.18)} \tag{1.20b}$$

**Problem** $R_{BFM}$

$$\min_x \quad C(x) \tag{1.21a}$$

$$\text{s.t.} \quad x \text{ satisfies (1.14), (1.15), (1.17), and (1.19)} \tag{1.21b}$$

Where (1.20) is non-convex and (1.21) is convex. Note that the relaxation (1.21) is an SOCP. Indeed, the set of constraints (1.19) can be reformulated using a second-order cone $\forall (i,j) \in \mathcal{E}$ :

$$\left\| \begin{array}{c} 2P_{ij} \\ 2Q_{ij} \\ l_{ij} - v_i \end{array} \right\|_2 \le l_{ij} + v_i \quad \Longleftrightarrow \quad \begin{array}{l} \sqrt{4P_{ij}^2 + 4Q_{ij}^2 + \left(l_{ij} - v_i\right)^2} \le l_{ij} + v_i \\ 4P_{ij}^2 + 4Q_{ij}^2 + \left(l_{ij} - v_i\right)^2 \le \left(l_{ij} + v_i\right)^2 \\ |S_{ij}|^2 = P_{ij}^2 + Q_{ij}^2 \le l_{ij} v_i \end{array}$$

where $P_{ij}$ and $Q_{ij}$ are the active and reactive power respectively along the line $(i,j) \in \mathcal{E}$. Thus the relaxation $\mathcal{R}_{BFM}$ problem can be efficiently computed.

### 1.2.3 Exactness of the relaxation

Whether the solution of the SOCP relaxation (1.21) yields an optimal solution for BFM-OPF depends on two factors :

- Whether the solution of (1.21) satisfies the equality constraint (1.18).
- Whether the phase angles of $V$ and $I$ can be recovered from such a solution.

Only the second issue is discussed here. The first issue will be investigated in the next section.

#### Angle Recovering Algorithm

The condition under which the recovery of the phase angles is described in [FL12], theorem 2. Here, we briefly introduce the recovering algorithm.

For a vector $\theta \in [-\pi, \pi)^n$, let's define the mapping $f_\theta(S, \ell, v) = (S, I, V)$ where :

$$\begin{aligned} V_i &:= \sqrt{v_i}\, \exp(\mathbf{i}\theta_i) && \text{for } i \in \mathcal{N} \\ I_{ij} &:= \sqrt{\ell_{ij}}\, \exp\left(\mathbf{i}\left(\theta_i - \angle S_{ij}\right)\right) && \text{for } (i,j) \in \mathcal{E} \end{aligned}$$

$\theta_i$ can be interpreted as the phase of the voltage at node $i \in \mathcal{N}$. The goal is then to find $\theta$ such that if $x := (S, l, v)$ is optimal for (1.20), $f_\theta(x)$ is optimal for the OPF problem. Whether such $\theta$ exists depends on the topology of the network. Let's define $\beta(x) \in \mathbb{R}^{|\mathcal{E}|}$ by

$$\beta_{ij}(x) := \angle\left(v_i - z_{ij}^* S_{ij}\right) \qquad \forall (i,j) \in \mathcal{E}$$

which is the phase angle difference across each line $i \to j \in \mathcal{E}$. Then the system simply consists of finding a vector $\theta$ such that $\theta_i - \theta_j = \beta_{ij}(x)$. Let's define the $|\mathcal{N}| \times |\mathcal{E}|$ incidence matrix $C$ of the graph $G$ :

$$C_{ie} := \begin{cases} 1 & \text{if edge } e \in \mathcal{E} \text{ leaves node } i \in \mathcal{N} \\ -1 & \text{if edge } e \in \mathcal{E} \text{ enters node } i \in \mathcal{N} \\ 0 & \text{Otherwise} \end{cases}$$

The first row of $C$ correspond the the substation node. Let's define the reduced matrix $B$ obtained by removing the first row of $C$ and taking its transpose. The condition under which we can recover the phase angles is then

$$\exists\, \theta \text{ that solves } B\theta = \beta(x) \qquad \mod 2\pi \tag{1.22}$$

If such solution exist, it's unique in $[-\pi, \pi)^n$. Moreover the condition for the existence of a solution to (1.22) has a simple interpretation : The voltage angle differences $\beta_{ij}(x)$ must sum to zero (mode $2\pi$) around any cycle (see [FL12] for further explanations). In the special case of radial networks, the cycle condition always holds. Hence the recovering of the phase angles can be computed by solving the system (1.22).

## 1.3 Exactness and Equivalence of the Two Models

For radial networks, it has been shown that one should always solve the second-order cone relaxation in the bus injection model and in the branch flow model. In this section, we show there exists a bijection between the feasible set of the OPF in the bus injection model and in the branch flow model, establishing the equivalence of theses models and their second-order cone relaxations. Then we briefly discuss the conditions under which the SOCP relaxations are exact.

### 1.3.1 Equivalence of the SOCP relaxations

Recall that the two SOCP relaxations of the optimal power flow problem in the bus injection model (1.23) and in the branch flow model (1.24) are given by:

**BIM-SOCP**

$$
\begin{aligned}
\min \quad & C(W_G) && && \text{(1.23a)}\\
\text{over} \quad & W_G \\
\text{s.t.} \quad & s_i = \sum_{j:j\sim i} \left([W_G]_{ii} - [W_G]_{ij}\right) y_{ij}^* && \forall j \in \mathcal{N} && \text{(1.23b)}\\
& \underline{s}_j \le s_j \le \bar{s}_j && \forall j \in \mathcal{N} && \text{(1.23c)}\\
& \underline{V}_j \le [W_G]_{jj} \le \bar{V}_j && \forall j \in \mathcal{N} && \text{(1.23d)}\\
& [W_G]_{ii}\,[W_G]_{jj} \ge |[W_G]_{ij}|^2 && \forall (i,j) \in \mathcal{E} && \text{(1.23e)}
\end{aligned}
$$

**BFM-SOCP**

$$
\begin{aligned}
\min \quad & C(S,l,v,s) && && \text{(1.24a)}\\
\text{over} \quad & S,l,v,s \\
\text{s.t.} \quad & v_j = v_i - 2\,\mathrm{Re}\left\{z_{ij}^* S_{ij}\right\} + z_{ij} l_{ij} z_{ij}^* && \forall (i,j) \in \mathcal{E} && \text{(1.24b)}\\
& s_j + \sum_{i:i\to j} \left(S_{ij} - z_{ij}|I_{ij}|^2\right) = \sum_{k:j\to k} S_{jk} && \forall j \in \mathcal{N} && \text{(1.24c)}\\
& \underline{V}_j \le V_j \le \bar{V}_j && \forall j \in \mathcal{N} && \text{(1.24d)}\\
& \underline{s}_j \le s_j \le \bar{s}_j && \forall j \in \mathcal{N} && \text{(1.24e)}\\
& |S_{ij}|^2 \le v_i l_{ij} && \forall (i,j) \in \mathcal{E} && \text{(1.24f)}
\end{aligned}
$$

For a hermitian partial matrix $W_G$ defined on the graph $G$, define the vector $x := (S,l,v) \in \mathbb{R}^{3|N|-2}$ such that $x = (S,l,v) =: g(W_G)$. The mapping function $g$ is given for $i \in \mathcal{N}$ and for $(i,j) \in \mathcal{E}$ by

$$
g(W_G) := \begin{cases}
v_i &:= V_i V_i^* = [W_G]_{ii} \\
S_{ij} &:= V_i I_{ij}^* = V_i\ (V_i - V_j)^*\, y_{ij}^* = y_{ij}^* \left([W_G]_{ii} - [W_G]_{ij}\right) \\
l_{ij} &:= I_{ij} I_{ij}^* = y_{ij}\,(V_i - V_j)\,(V_i - V_j)^*\, y_{ij}^* \\
&= |y_{ij}|^2 \left([W_G]_{ii} - [W_G]_{ij} - [W_G]_{ji} + [W_G]_{jj}\right)
\end{cases}
$$

And the inverse mapping $g^{-1}$ from $\mathbb{R}^{3|N|-2}$ to the set of Hermitian G-partial matrix is defined such that $g^{-1}(x) := W_G$.

$$g^{-1}(x) := \begin{cases} [W_G]_{ii} & := \quad v_i \\ [W_G]_{ij} & := \quad v_i - z_{ij}^* S_{ij} = [W_G]_{ji}^* \end{cases}$$

Using this mapping function $g$, we can show that the constraints in one model can be expressed in terms of the variables in the other model. By doing so, we end up with the constraints of the other model, and which proves that the feasible set of two formulations are equivalent.

From the definition of $(S, \ell, v)$ in terms of the G-partial matrix $W_G$, we can easily verified that:

$$v_j = v_i - 2\,\mathrm{Re}\Big\{z_{ij}^* S_{ij}\Big\} + |z_{ij}|^2 \ell_{ij} \qquad \forall (i,j) \in \mathcal{E}$$

The power flow constraints can be transformed for $i \in \mathcal{N}$ :

$$\begin{aligned} s_i &= \sum_{j:i\sim j} \left([W_G]_{ii} - [W_G]_{ij}\right) y_{ij}^* \\ &= \sum_{j:i\to j} \left([W_G]_{ii} - [W_G]_{ij}\right) y_{ij}^* + \sum_{k:k\to i} \left([W_G]_{ii} - [W_G]_{ik}\right) y_{ik}^* \\ &= \sum_{j:i\to j} S_{ij} + \sum_{k:k\to i} \Big(|z_{ki}|^2 l_{ki} - \left([W_G]_{kk} - [W_G]_{ki}\right)\Big) y_{ki}^* \\ &= \sum_{j:i\to j} S_{ij} + \sum_{k:k\to i} \left(z_{ki} l_{ki} - S_{ki}\right) \\ &= \sum_{j:i\to j} S_{ij} - \sum_{k:k\to i} \left(S_{ki} - z_{ki} l_{ki}\right) \end{aligned}$$

which is exactly the power balance constraint in the branch flow model. The voltage constraints can also be transformed for $i \in \mathcal{N}$ as

$$\underline{V}_i^2 \le [W_G]_{ii} \le \bar{V}_i^2 \quad \Longleftrightarrow \quad \underline{v}_i \le v_i \le \bar{v}_i \qquad \forall i \in \mathcal{N}$$

where $\underline{v}_i = \underline{V}_i^2$ and $\bar{v}_i = \bar{V}_i^2$. The rank-constraints can be reformulated for $(i,j) \in \mathcal{E}$ as

$$\begin{aligned} & \text{rank } \left(W_G\left(i,j\right)\right) = 1 \\ \Leftrightarrow\ & [W_G]_{jj} = \frac{[W_G]_{ij}[W_G]_{ji}}{[W_G]_{ii}} \\ \Leftrightarrow\ & [W_G]_{jj} - [W_G]_{ij} - [W_G]_{ji} + [W_G]_{ii} = \frac{\left([W_G]_{ii} - [W_G]_{ij}\right)\left([W_G]_{ii} - [W_G]_{ji}\right)}{[W_G]_{ii}} \\ \Leftrightarrow\ & l_{ij} = \frac{|S_{ij}|^2}{v_i} \end{aligned}$$

And finally, in the same way, the constraints regarding the positive semidefinitness of the G-partial matrix $W_G$ can be reformulated into

$$W_G\left(\{i,j\}\right) \succeq 0 \quad \Longleftrightarrow \quad \ell_{ij} \ge \frac{|S_{ij}|^2}{v_i}$$

In conclusion, by using the mapping function $g$, we saw that each constraint in one model can be reformulated in terms of the variables of the other. Moreover, those constraints are exactly the constraints considered in the other model, and hence proves the equivalence between the bus injection model and the branch flow model as well as the equivalence between their second-order cone relaxations.

### 1.3.2 Exactness of the SOCP relaxations for radial networks

Since the SOCP relaxations in the branch flow model and in the bus injection model are equivalent, the conditions under which they are exact are also equivalent but expressed in terms of different variables.

**Branch Flow Model** We saw that whether the solution of the SOCP relaxation in the Branch Flow Model (1.21) yields an optimal solution for BFM-OPF depends on two factors :

- Whether the solution of (1.21) satisfies the equality constraint (1.18).
- Whether the phase angles of $V$ and $I$ can be recovered from such a solution.

We already proved that in the special case of radial networks, the angle recovering condition always holds and hence the exactness of the SOCP relaxation depends only on whether the solution satisfies the equality constraint :

$$l_{ij} = \frac{|S_{ij}|^2}{v_i} \qquad (i,j) \in \mathcal{E}$$

**Bus Injection Model** In the Bus Injection Model, The exactness of the SOCP relaxation for the BIM-OPF depends on whether the G-partial matrix $W_G$ is rank-1 or not.

Up to date sufficient conditions that have been derived for the exactness of these SOCP relaxations doesn't hold in practice [ZT13]. For instance some conditions require some/all buses to be able to draw infinite power, and the condition in [LZT12] requires a fixed voltage at every bus. A more elaborated condition based on the branch flow model has been described in [GLTL12] to prove the exactness of the SOCP relaxations. This condition can be checked a priori (prior solving the relaxation), and generally holds for real networks, even with high penetration of distributed generation. Moreover, they showed that the feasible set of OPF problem can be slightly modified to force the solution to be exact. Surprisingly, with this modification, only feasible points that are "close" to the voltage upper and lower bounds are eliminated.

## 1.4 Conclusion

This first chapter gave a short review of recent advances in convexification methods for solving the Optimal power flow problem, which were developed in the last few years. These methods proved to be very promising, since they can be computed very efficiently. However, there are still theoretical issues (e.g. conditions for exactness).

# 2 | The OPF Problem on Three-Phase Distribution Networks

## 2.1 OPF Formulation

In this first section, we briefly introduce the OPF formulation for an unbalanced multiphase distribution network. The formulation is a generalisation of the Branch Flow Model described in section 1.2, and is based on [GL14] and [PL15]. We choose to work with the branch flow model because of its numerical stability. BIM-SDP is indeed ill-conditioned due to subtractions of voltages that are close in value. Using alternating variables, BFM avoids these subtractions and is therefore numerically more stable. Moreover, the variables in the BFM correspond to physical quantities such as the branch power and current, which are more convenient for interpretation.

### 2.1.1 Notations

A distribution network is modelled as a directed graph $\mathcal{G} := (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} := \{0, 1, \dots, n\}$ represents the set of buses and $\mathcal{E}$ the set of lines connecting the buses. Since $\mathcal{G}$ represents a distribution network, we expect him to be radial. We index the root of the tree by 0 and denote it as the substation node, which draws power from the transmission network to the distribution network for power balance. Let $\mathcal{N}_+$ denote the set of non-substation buses.

Each bus $i$ has a unique ancestor $A_i$ and a set of children buses denoted by $C_i$. For convenience, we adopt the following graph orientation : Every line points towards the root. Hence, each line $i \in \mathcal{E}$ connects the bus $i$ to its unique ancestor $A_i$. Since the line set $\mathcal{E} := \{1, \dots, n\} = \mathcal{N}_+$, we will use $\mathcal{N}_+$ as the set of lines in the sequel. Denote by $a, b, c$ the three phases of the network. Each bus $i \in \mathcal{N}$ has a set of phases $\Phi_i \subseteq \{a, b, c\}$. the set of phases of bus $i$ is a subset of the phases of its ancestor and a superset for the phases of its children $j \in C_i$, i.e. $\Phi_i \subseteq \Phi_{A_i}$, $\Phi_j \subseteq \Phi_i \ \forall j \in C_i$.

For each bus $i \in \mathcal{N}$ and each phase $\phi \in \Phi_i$, denote $V_i^\phi$ the complex voltage and $s_i^\phi = p_i^\phi + iq_i^\phi$ the complex power injection (generation minus load). And denote the vectors $V_i := \left(V_i^\phi, \forall \phi \in \Phi_i\right) \in \mathbb{C}^{|\Phi_i| \times 1}$, and $s_i := \left(s_i^\phi, \forall \phi \in \Phi_i\right) \in \mathbb{C}^{|\Phi_i| \times 1}$.

For each line $i \in \mathcal{N}_+$ connecting bus $i$ and its ancestor $A_i$, the set of phases is $\Phi_{A_i} \cap \Phi_i = \Phi_i$ (since $\Phi_i \subseteq \Phi_{A_i}$). Denote for each phase $\phi \in \Phi_i$, $I_i^\phi \in \mathbb{C}$ the complex branch current form bus $i$ to its ancestor $A_i$, and $I_i := \left(I_i^\phi, \forall \phi \in \Phi_i\right) \in \mathbb{C}^{|\Phi_i| \times 1}$. And let's define the sending-end complex power $S_i := V_i I_i^* \in \mathbb{C}^{|\Phi_i| \times |\Phi_i|}$ from node $i$ to its ancestor $A_i$.

Finally, let's define some variables useful for the formulation of the OPF problem in the branch flow model : $v_i := V_i V_i^* \in \mathbb{C}^{|\Phi_i|\times|\Phi_i|}$, $\forall i \in \mathcal{N}$, $\ell_i := I_i I_i^* \in \mathbb{C}^{|\Phi_i|\times|\Phi_i|}$, $\forall i \in \mathcal{N}_+$. A variable without a subscript denotes the set of variables with appropriate components :

| | |
|---|---|
| $v := (v_i, i \in \mathcal{N})$ | $s := (s_i, i \in \mathcal{N})$ |
| $\ell := (\ell_i, i \in \mathcal{N}_+)$ | $S := (S_i, i \in \mathcal{N}_+)$ |

The notations are summarized on figure (2.1). Note that the impedance $z_{ij}$ for each line $(i,j) \in \mathcal{E}$ is a symmetric $|\Phi_i \cap \Phi_j| \times |\Phi_i \cap \Phi_j|$ complex matrix that consists of self-impedances on the diagonal and mutual impedances (between phases) on the off-diagonals. Note that for balanced system, the impedance matrix is diagonal. Hence the voltages and currents are independent form each each other, i.e. they have no cross-terms and can be treated as three de-coupled single phase networks.

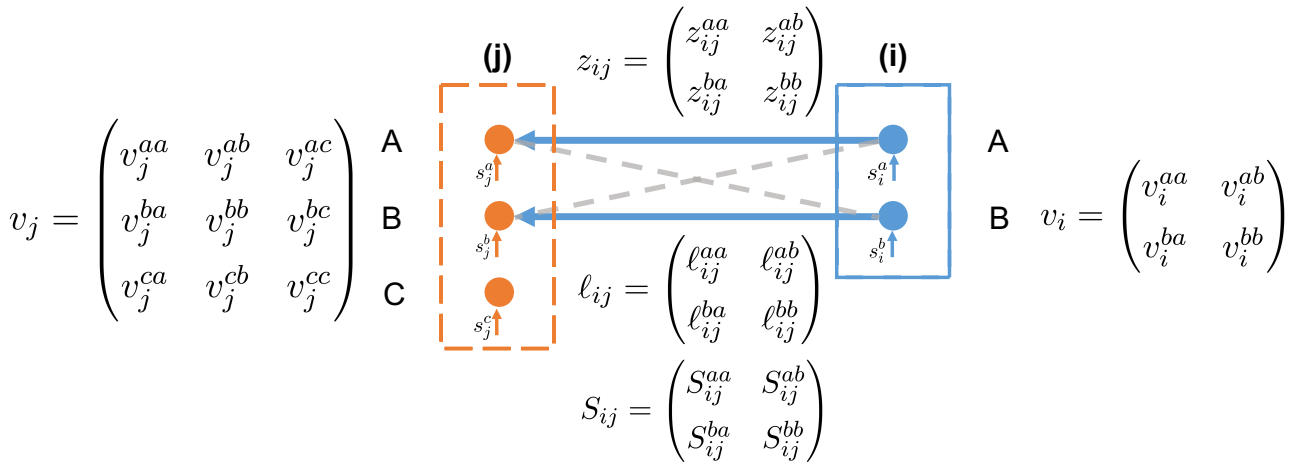



Figure 2.1: Notations

### 2.1.2 Objective Function

The optimal power flow problem aims at minimizing a certain objective function. Typical objective functions iclude generation cost or total power loss. Moreover the solution must satisfy power flow equations and operational constraints. We assume there exists for each node $i \in \mathcal{N}$ and each phase $\phi \in \Phi_i$, a real-valued function $f_i^\phi\left(s_i^\phi\right)$ defined on $\mathbb{R}$ which represents the local objective of node $i$. Then the total objective function is given by

$$f(s) := \sum_{i\in\mathcal{N}} f_i(s_i) := \sum_{i\in\mathcal{N}} \sum_{\phi\in\Phi_i} f_i^\phi(s_i^\phi). \tag{2.1}$$

If we want to minimize the total line loss of real power, the objective function for each node $i$ and for each phase $\phi \in \Phi_i$ is

$$f_i^\phi(s_i^\phi) = p_i^\phi$$

If we want to minimize the generation cost, the objective function could be

$$f_i^\phi(s_i^\phi) = \frac{\alpha_i^\phi}{2}(p_i^\phi)^2 + \beta_i^\phi p_i^\phi,$$

where $\alpha_i^\phi, \beta_i^\phi > 0$ are fixed parameters that depend on the characteristic of the generator located at node $i$.

### 2.1.3 Constraints

For each bus $i \in \mathcal{N}$, there are two technical constraints on each phase $\phi \in \Phi_i$. The first constraint concerns the complex power injection $s_i^\phi$, and can be captured by some feasible power injection set $\mathcal{I}_i^\phi$, such that :

$$s_i^\phi \in \mathcal{I}_i^\phi \quad \text{for } \phi \in \Phi_i \text{ and } i \in \mathcal{N} \tag{2.2}$$

The feasible power injection is determined by the load and the generation attached at each phase $\phi \in \Phi_i$. A common feasible set is when real power can vary within $[\underline{p}_i, \bar{p}_i]$ and reactive power can vary within $[\underline{q}_i, \bar{q}_i]$, the injection region $\mathcal{I}_i^\phi$ is

$$\mathcal{I}_i^\phi = \{p + \mathbf{i}q \in \mathbb{C} \mid p \in [\underline{p}_i, \bar{p}_i], q \in [\underline{q}_i, \bar{q}_i]\} \subseteq \mathbb{C} \tag{2.3}$$

Since the substation node is responsible for drawing power from the transmission network to the distribution network for power balance, we assume its power injection $s_0$ to be unconstrained, thus $-\underline{p}_0 = -\underline{q}_0 = \bar{p}_0 = \bar{q}_0 = +\infty$.

The second technical constraint concerns the voltage magnitude. It needs to be maintained within a fixed region. Since the diagonal of $v_i$ is the voltage magnitude square for each phase $\phi \in \Phi_i$, we can formulate the constraint as :

$$\underline{v}_i^\phi \leq v_i^{\phi\phi} \leq \bar{v}_i^\phi \quad i \in \mathcal{N} \tag{2.4}$$

where $v_i^{\phi\phi}$ denotes the $\phi_{th}$ diagonal element of $v_i$. The voltage magnitude is typically allowed to deviate by a $5-10\%$ from the nominal value of the network for each phase $\phi \in \Phi_i$. The voltage magnitude of the substation node is assumed to be fixed, i.e. $\underline{v}_0 = \bar{v}_0$.

Besides operational constraints, power flows are characterised by two sets of equations :

- Ohm's law :

$$V_i - \mathcal{P}_i\left(V_{A_i}\right) = z_i I_i \tag{2.5}$$

- Power Balance :

$$\operatorname{diag}\left(\sum_{j \in C_i} \mathcal{P}_i\left(S_j - z_j l_j\right)\right) + s_i = \operatorname{diag}\left(S_i\right) \tag{2.6}$$

where the operator $\mathcal{P}_i(x)$ defines the projection of $x$ on the set of phases $\Phi_i$. For example, $P_i\left(v_{A_i}\right)$ considers only the entries of $v_{A_i}$ for the phases present at node $i$ and $A_i$. $\mathcal{P}_i(S_j - z_j\ell_j)$ denotes the lifting of $S_j - z_j\ell_j$ to the phases $\Phi_i$ and filling the missing phases with zero, e.g. if $\Phi_{A_i} = \{a, b, c\}$, $\Phi_i = \{a, b\}$ and $\Phi_j = \{a\}$, then

$$\mathcal{P}_i(v_{A_i}) := \begin{pmatrix} v_{A_i}^{aa} & v_{A_i}^{ab} \\ v_{A_i}^{ba} & v_{A_i}^{bb} \end{pmatrix} \qquad \mathcal{P}_i(S_j - z_j\ell_j) := \begin{pmatrix} S_j^{aa} - z_j^{aa}\ell_j^{aa} & 0 \\ 0 & 0 \end{pmatrix}$$

The term $\operatorname{diag}\left(S_j\right)$ represents the sending-end power on the branch connecting $i$ to its ancestor $A_i$. if

we multiply both sides of (2.5) by their hermitian, we obtain :

$$\begin{aligned} & \mathcal{P}_i\left(V_{A_i}\right) = V_i - z_i I_i \\ \Rightarrow\ & \mathcal{P}_i\left(V_{A_i}\right) \mathcal{P}_i\left(V_{A_i}^*\right) = \left(V_i - z_i I_i\right)\left(V_i - z_i I_i\right)^* \\ \Rightarrow\ & \mathcal{P}_i\left(v_{A_i}\right) = v_i - z_i I_i V_i^* - V_i I_i^* z_i^* + z_i I_i I_i^* z_i^* \\ \Rightarrow\ & \mathcal{P}_i\left(v_{A_i}\right) = v_i - \left(z_i S_i^* + S_i z_i^*\right) + z_i l_i z_i^* \end{aligned} \tag{2.7}$$

### 2.1.4 Formulation

Given a radial network $\mathcal{G}$, the impedance of each line $i \in \mathcal{N}_+$ and the nominal voltage, the branch flow model for an unbalanced network is defined by the following set of equations :

$$\mathcal{P}_i(v_{A_i}) = v_i - \left(z_i S_i^* + S_i z_i^*\right) + z_i \ell_i z_i^* \qquad i \in \mathcal{N}_+ \tag{2.8a}$$

$$s_i + \operatorname{diag}\left(\sum_{j \in C_i} \mathcal{P}_i\left(S_j - z_j l_j\right)\right) = \operatorname{diag}\left(S_i\right) \qquad i \in \mathcal{N} \tag{2.8b}$$

$$\begin{pmatrix} v_i & S_i \\ S_i^H & \ell_i \end{pmatrix} \succeq 0 \qquad i \in \mathcal{N} \tag{2.8c}$$

$$\operatorname{rank} \begin{pmatrix} v_i & S_i \\ S_i^* & \ell_i \end{pmatrix} = 1 \qquad i \in \mathcal{N} \tag{2.8d}$$

The rank constraint (2.8d) and the positive semidefinite constraint (2.8c) come from the definition of the variables $(v, \ell, S)$

$$\begin{pmatrix} v_i & S_i \\ S_i^* & \ell_i \end{pmatrix} = \begin{bmatrix} V_i \\ I_i \end{bmatrix} \begin{bmatrix} V_i \\ I_i \end{bmatrix}^* \tag{2.9}$$

This matrix must be rank-1 and positive semidefinite in order to recover the voltage vectors $V_i$ and the current vectors $I_i$. Because of the rank constraint, the problem is non-convex; by relaxing it, we obtain the following **SDP** relaxation:

**R-OPF :**

$$\min \quad \sum_{i \in \mathcal{N}} \sum_{\phi \in \Phi_i} f_i^\phi(s_i^\phi) \tag{2.10a}$$

$$\text{over} \quad v, s, S, \ell$$

$$\text{s.t.} \quad \mathcal{P}_i(v_{A_i}) = v_i - \left(z_i S_i^* + S_i z_i^*\right) + z_i \ell_i z_i^* \qquad i \in \mathcal{N}_+ \tag{2.10b}$$

$$s_i + \operatorname{diag}\left(\sum_{j \in C_i} \mathcal{P}_i\left(S_j - z_j l_j\right)\right) = \operatorname{diag}\left(S_i\right) \qquad i \in \mathcal{N} \tag{2.10c}$$

$$\begin{pmatrix} v_i & S_i \\ S_i^* & \ell_i \end{pmatrix} \succeq 0 \qquad i \in \mathcal{N} \tag{2.10d}$$

$$s_i^\phi \in \mathcal{I}_i^\phi \qquad \phi \in \Phi_i,\ i \in \mathcal{N} \tag{2.10e}$$

$$\underline{v}_i^\phi \leq v_i^{\phi\phi} \leq \overline{v}_i^\phi \qquad \phi \in \Phi_i,\ i \in \mathcal{N} \tag{2.10f}$$

## 2.2 How to recover the original variables

Given a vector $(v, \ell, S)$ that satisfies (2.8), it is proved in [GL14] that the bus voltages $V_i$ and branch currents $I_i$ can be uniquely determined if the network is a tree.

**Lemma 1.** *[GL14] Let $v_i \in \mathbb{H}^{|\Phi_i|\times|\Phi_i|}$ for $i \in \mathcal{N}$. Le $S_{ij} \in \mathbb{C}^{|\Phi_i|\times|\Phi_i|}$ and $\ell_{ij} \in \mathbb{H}^{|\Phi_i|\times|\Phi_i|}$ for $i \to j$. If $(v, S, \ell)$ is such that :*

- $v_0 = V_0^{ref} V_0^{ref\,*}$ *for some nominal voltage* $V_0^{ref} \in \mathbb{C}^{|\Phi_0|}$.
- $diag\,(v_i)$ *is nonzero component-wise for* $i \in \mathcal{N}$.
- $(v, S, \ell)$ *satisfies Ohm's Law* (2.7)
- $\begin{pmatrix} v_i & S_i \\ S_i^* & \ell_i \end{pmatrix} \succeq 0$ *and is rank-1* $\forall i \in \mathcal{N}$

*Then the algorithm* (1) *computes the unique* $(V, I)$ *that satisfies* $V_0 = V_0^{ref}$, $v_i = V_i V_i^*$ $\forall i \in \mathcal{N}$, $\ell_i = I_i I_i^*$, *and* $S_i = V_i I_i^*$ $\forall i \in \mathcal{N}_+$.

**Algorithm 1:** Recover $(V, I)$ from $(v, S, \ell)$

**input** : $(v, S, \ell)$ that satisfies the conditions in Lemma (1)

**output**: $(V, I)$

$V_0 \longleftarrow V_0^{ref}$

$\mathcal{N}_{visit} \longleftarrow \{0\}$

**while** $\mathcal{N}_{visit} \neq \mathcal{N}$ **do**

  find $i \to j$ such that $i \in \mathcal{N}_{visit}$ and $j \notin \mathcal{N}_{visit}$

  compute

$$I_{ij} \leftarrow \frac{1}{\mathrm{Tr}(v_i)} S_{ij}^* V_i$$
$$V_j \leftarrow V_i - z_{ij} I_{ij}$$
$$\mathcal{N}_{visit} \leftarrow \mathcal{N}_{visit} \cup \{j\}$$

**end**

## 2.3 Conclusion

This second chapter described an SOCP relaxation of the optimal power flow problem in the special case of multiphase radial networks. Even if this formulation is very similar to the case of single-phase networks, there are no theoretical works proving the exactness of the relaxation. However, the relaxation is generally exact for real distribution networks as shown in section (6.4)

# Part II

# Distributed Algorithm

# 3 | Alternating Direction Method of Multipliers

The alternating direction method of multipliers (ADMM) is a convex optimization algorithm dating back to the early 80's [GR75, GM76]. The method has experienced a revival in recent years due to its applicability to optimization problems arising from "big data" and image processing applications, and its relative ease with which it may be implemented in parallel and distributed computational environnements. The alternating direction method of multipliers takes the form of a decomposition-coordination procedure, in which the solutions to small local sub-problems are coordinated to find a solution to a large global problem. The method turns out to be equivalent or at least closely related to many other algorithms such as the Douglas-Rachford splitting method, proximal methods, Dykstra's alternating projections method, etc. This chapter aims at providing an accessible introduction to this method, including analytical results and two short applications (based on [BPC+11]).

## 3.1 Precursors

In this section, we briefly review the two optimization methods that are precursors to ADMM. Although those methods will not be used in the sequel, it provides some useful backgrounds and motivations.

### 3.1.1 Dual Ascent

Let's consider a simple equality-constrained convex optimization problem

$$\min_{x \in \mathbb{R}^n} \quad f(x) \quad \text{s.t. } Ax = b \tag{3.1}$$

where $A \in \mathbb{R}^{m \times n}$ and $f(x) : \mathbb{R}^n \to \mathbb{R}$ is convex. The Lagrangian function of this problem is

$$L(x, \lambda) = f(x) + \langle \lambda, Ax - b \rangle$$

where $\lambda \in \mathbb{R}^m$ is the dual variable. The dual function is therefore

$$g(\lambda) = \inf_x L(x, \lambda)$$

And the dual problem consists in maximizing this dual function. In the dual ascent method, we solve the dual problem using gradient ascent. Assuming that $g$ is differentiable, the dual ascent method

consists of iterating the updates

$$x^{k+1} = \underset{x}{\operatorname{argmin}}\, L(x, \lambda^k)$$
$$\lambda^{k+1} = \lambda^k + \alpha_k \left(Ax^{k+1} - b\right)$$

where $\{\alpha_k\}$ is a sequence of positive scalar paremeters bounded away from 0. The main benefit of the dual ascent method is that it can lead to a decentralized algorithm in some cases. For example, suppose the objective function $f$ is separable, i.e.

$$f(x) = \sum_{i=1}^{N} f_i(x_i)$$

where $x = (x_1, \ldots, x_N)$ and the variables $x_i \in \mathbb{R}^{n_i}$ are subvectors of $x$. Partitioning the matrix $A$ such that

$$Ax = \sum_{i=1}^{N} A_i x_i$$

yields a new formulation of the Lagrangian function

$$L(x, \lambda) = \sum_{i=1}^{N} L_i(x_i, \lambda) = \sum_{i=1}^{N} f_i(x_i) + \langle \lambda, A_i x_i - b \rangle$$

Meaning that the x-minimization can be split into $N$ separable subproblems that can be solved in parallel. Explicitly, the algorithm is

$$x_i^{k+1} = \underset{x_i}{\operatorname{argmin}}\, L_i(x_i, \lambda^k)$$
$$\lambda^{k+1} = \lambda^k + \alpha_k \left(Ax^{k+1} - b\right)$$

In this case, we refer to the dual ascent method as the **dual decomposition**.

### 3.1.2 Method of Multipliers

Augmented Lagrangian methods were developed to bring robustness to the dual ascent method. In particular, to yield convergence without assumptions such as strict convexity of the objective function. The augmented Lagrangian for (3.1) is

$$L_\rho(x, \lambda) = f(x) + \langle \lambda, Ax - b \rangle + \frac{\rho}{2} \|Ax - b\|_2^2$$

where $\rho > 0$ is called the penalty parameter. The augmented Lagrangian can be viewed as a classic Lagrangian associated with the problem

$$\min_{x \in \mathbb{R}^n} \quad f(x) + \frac{\rho}{2} \|Ax - b\|_2^2 \quad \text{s.t. } Ax = b$$

which is clearly equivalent to the original problem (3.1). Applying the dual ascent method as before

yields the following algorithm

$$x^{k+1} = \underset{x_i}{\operatorname{argmin}}\, L_\rho(x, \lambda^k)$$
$$\lambda^{k+1} = \lambda^k + \rho\left(Ax^{k+1} - b\right)$$

which is known as the **method of multipliers**. By using $\rho$ as step-size in the dual update, we ensure the dual feasibility of the iterates $\left\{x^k, \lambda^k\right\}$. The method of multipliers converges under far more general conditions than the dual ascent, including cases when $f$ is not strictly convex. The drawback is that even if the objective function $f$ is separable, the augmented Lagrangian is not separable. This means that the basic method of multipliers cannot be used for decomposition.

## 3.2 Alternating Direction Method of Multipliers

The ADMM algorithm combines the decomposition of dual ascent type algorithms and the good convergence properties of the method of multipliers. It intends to solve the following problem

$$\min \quad f(x) + g(z) \quad \text{s.t. } Ax + Bz = c \tag{3.2}$$

with variables $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$, where $A \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{p\times m}$, and $f$ and $g$ are convex functions on $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. We let $f$ and $g$ take not only values in $\mathbb{R}$ but also the value $+\infty$, so that constraints may be "embedded" in $f$ and $g$. As in the method of multipliers, the augmented Lagrangian of (3.2) is

$$L_\rho(x, z, \lambda) = f(x) + g(z) + \langle \lambda, Ax + Bz - c\rangle + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 \tag{3.3}$$

And the method of multipliers for (3.2) is

$$\left(x^{k+1}, z^{k+1}\right) = \underset{x,\, z}{\operatorname{argmin}}\, L_\rho(x, z, \lambda^k) \tag{3.4}$$
$$\lambda^{k+1} = \lambda^k + \rho\left(Ax^{k+1} + Bz^{k+1} - c\right) \tag{3.5}$$

Here, the augmented Lagrangian is jointly minimized with respect to the primal variables. On the other hand, the ADMM method updates in an **alternating** way the primal variables. One can view the ADMM as an approximate version of the classical augmented Lagrangian method (3.4) in which a single pass of "Gauss-Seidel" block minimization substitues the full minimization of the augmented Lagrangian. ADMM method consists of the iterations

$$x^{k+1} = \underset{x}{\operatorname{argmin}}\, L_\rho(x, z^k, \lambda^k) \tag{3.6}$$
$$z^{k+1} = \underset{z}{\operatorname{argmin}}\, L_\rho(x^{k+1}, z, \lambda^k) \tag{3.7}$$
$$\lambda^{k+1} = \lambda^k + \rho\left(Ax^{k+1} + Bz^{k+1} - c\right) \tag{3.8}$$

As the method of multipliers, the dual variable update uses a step-size equal to the penalty parameter $\rho$. Equations (3.6) and (3.7) can be reformulated by combining the linear and the quadratic term in

the augmented Lagrangian function and scaling the dual variable. We have

$$\begin{aligned}\lambda^T r + (\rho/2)\|r\|_2^2 &= \lambda^T r + (\rho/2) r^T r = (\rho/2)\left(r^T r + (2/\rho)\lambda^T r + (1/\rho^2)\lambda^T\lambda\right) - (1/2\rho)\lambda^T\lambda \\ &= (\rho/2)\|r + (1/\rho)\lambda\|_2^2 - (1/2\rho)\|\lambda\|_2^2 \\ &= \frac{\rho}{2}\|r+u\|_2^2 - \frac{\rho}{2}\|u\|_2^2\end{aligned}$$

where $u = (\lambda/\rho)$, the scaled dual variable. If we define $r$ as the primal residual $Ax + Bz - c$, the ADMM method can be expressed as :

$$\begin{aligned}x^{k+1} &= \underset{x}{\operatorname{argmin}}\, f(x) + \frac{\rho}{2}\left\|Ax + Bz^k - c + u^k\right\|_2^2 \\ z^{k+1} &= \underset{z}{\operatorname{argmin}}\, g(z) + \frac{\rho}{2}\left\|Ax^{k+1} + Bz - c + u^k\right\|_2^2 \\ u^{k+1} &= u^k + Ax^{k+1} + Bz^{k+1} - c\end{aligned}$$

The two formulations are equivalent but the formulas in the scaled version are generally shorter than the unscaled version.

## 3.3 Convergence Results

In this section, we briefly introduce the convergence results of the ADMM method. See [BPC+11, EY12, NLR+15], for a more comprehensive discussion. Compared to the dual decomposition, ADMM is guaranteed to converge to an optimal solution under less restrictive conditions.

**Assumption 1** The function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ are closed, proper, and convex, i.e. the epigraphs of $f$ and $g$ are closed non-empty convex sets.

**Assumption 2** The unaugmented Lagrangian $L_0$ has a saddle point, i.e there exist $(x^\star, z^\star, \lambda^\star)$ such that

$$L_0(x^\star, z^\star, \lambda) \le L_0(x^\star, z^\star, \lambda^\star) \le L_0(x, z, \lambda^\star)$$

for all $x, z, \lambda$.

**Proposition [BPC+11]** Under assumptions 1 and 2, the ADMM iterates satisfy the following :

- Residual convergence $lim_{k\to\infty} r^k = 0$, i.e. the iterates approach feasibility.
- Objective convergence $lim_{k\to\infty} f(x^k) + g(z^k) = p^\star$, i.e. the objective function approaches the optimal value.
- Dual variables convergence $lim_{k\to\infty}\lambda^k = \lambda^\star$, where $\lambda^\star$ is the dual optimal point.

Where $r^k = Ax^k + Bz^k - c$. Note that $x^k$ and $z^k$ don't need to converge to optimal values, although such results can be shown under additional assumptions.

### 3.3.1 Optimality Condition and Stopping Criterion

The optimality conditions for the ADMM problem (3.2) are primal and dual feasbilities

$$Ax^\star + Bz^\star - c = 0 \tag{3.9}$$
$$0 \in \partial f(x^\star) + A^T\lambda^\star \tag{3.10}$$
$$0 \in \partial g(z^\star) + B^T\lambda^\star \tag{3.11}$$

where $\partial$ denote the sudbifferential operator. Since $z^{k+1}$ minimize $L_\rho(x^{k+1}, z, \lambda^k)$ then

$$\begin{aligned} 0 &\in \partial g(z^{k+1}) + B^T\lambda^k + \rho B^T\left(Ax^{k+1} + Bz^{k+1} - c\right) \\ &= \partial g(z^{k+1}) + B^T\left(\lambda^k + \rho r^{k+1}\right) \\ &= \partial g(z^{k+1}) + B^T\lambda^{k+1} \end{aligned}$$

Meaning that $z^{k+1}$ and $\lambda^{k+1}$ always satisfy (3.11) when the penalty parameter $\rho$ is chosen as step size. Since $x^{k+1}$ minimizes $L_\rho(x, z^k, \lambda^k)$, we have that

$$\begin{aligned} 0 &\in \partial f(x^{k+1}) + A^T\lambda^k + \rho A^T\left(Ax^{k+1} + Bz^k - c\right) \\ &= \partial f(x^{k+1}) + A^T\left(\lambda^k + \rho r^{k+1} + \rho B(z^k - z^{k+1})\right) \\ &= \partial f(x^{k+1}) + A^T\lambda^{k+1} + \rho A^T B(z^k - z^{k+1}) \end{aligned}$$

or equivalently,

$$\rho A^T B(z^{k+1} - z^k) \in \partial f(x^{k+1}) + A^T\lambda^{k+1}$$

The quantity

$$s^{k+1} = \rho A^T B(z^{k+1} - z^k)$$

can be viewed as a residual for the dual feasibility. We will refer to $s^{k+1}$ as the **dual residual** and $r^{k+1}$ as the **primal residual** at iteration $k+1$. Since 3.11 always holds, the optimal conditions sum up to the convergence of the primal and dual residuals $r^k$ and $s^k$, respectively. It's proven in [BPC+11] that those two residuals converge to zero as ADMM proceeds. This result suggests that a reasonable termination criterion should have small primal and dual residuals, i.e.

$$\left\|r^k\right\|_2 \leq \epsilon^{\text{primal}} \qquad \left\|s^k\right\|_2 \leq \epsilon^{\text{dual}}$$

where $\epsilon^{\text{primal}} > 0$ and $\epsilon^{\text{dual}} > 0$ are the feasibility tolerances for the primal and dual feasibility conditions.

**Rate of Convergence** As far as the rate of convergence is concerned, only a linear rate of convergence was recently established in the literature [NLR+15, GOSB14], though this rate does not require strict convexity. Therefore, compared to second-order methods that are able to achieve a high accuracy via expensive iterations, ADMM relies on low-complex iterations and can achieve a modest accuracy in tens of iterations. Nevertheless, inspired by Nesterov's scheme for accelerating gradient methods [Nes], great efforts have been devoted to accelerating ADMM and attaining high accuracy in a reasonable number of iterations [GOSB14].

## 3.4 Applications

The problems addressed in this section illustrate why ADMM is a natural fit for consensus problems as well as common optimization problems.

### 3.4.1 $l_1$-regularized linear regression

The first application is the $l_1$-regularized linear regression, also called Lasso. This involves solving

$$\min_{x\in\mathbb{R}^n} \quad \frac{1}{2}\|Ax-b\|_2^2 + \lambda\|x\|_1$$

with $\lambda > 0$, the regularization parameter. In the ADMM form, the lasso problem can be written as :

$$\min_{x,z\in\mathbb{R}^n} \quad \frac{1}{2}\|Ax-b\|_2^2 + \lambda\|z\|_1 \quad \text{s.t.} \quad x - z = 0$$

The ADMM algorithm (scaled version) becomes

$$x^{k+1} := \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} \quad \frac{1}{2}\|Ax-b\|_2^2 + \frac{\rho}{2}\left\|x - z^k + u^k\right\|_2^2 \tag{3.12}$$

$$z^{k+1} := \underset{z\in\mathbb{R}^n}{\operatorname{argmin}} \quad \lambda\|z\|_1 + \frac{\rho}{2}\left\|x^{k+1} - z + u^k\right\|_2^2 \tag{3.13}$$

$$u^{k+1} := u^k + x^{k+1} - z^{k+1} \tag{3.14}$$

Fortunately, the primal updates (3.12) and (3.13) enjoy a closed-form solution

$$x^{k+1} := \left(A^TA + \rho I\right)^{-1}\left(A^Tb + \rho\left(z^k - u^k\right)\right) \tag{3.15}$$

$$z^{k+1} := S_{\lambda/\rho}\left(x^{k+1} + u^k\right) \tag{3.16}$$

$$u^{k+1} := u^k + x^{k+1} - z^{k+1} \tag{3.17}$$

where $S_k(a)$ is a shrinkage operator : $S_k(a) = a\left(1 - \frac{k}{|a|}\right)^+$. Note that we can cache an initial factorization of (3.15) to make subsequent iterations much cheaper.

### 3.4.2 Consensus

Let's consider a simple consensus problem that could involve the following optimization problem

$$\min_{x\in\mathbb{R}^n} \quad f(x) = \sum_{i=1}^{N} f_i(x) \tag{3.18}$$

where $f_i : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ are convex. Each of the objective function $f_i$ can encode constraints by assigning $f_i(x) = +\infty$ when $x$ is unfeasible. The problem can be rewritten with local variables $x_i \in \mathbb{R}^n$ and a global variable $z \in \mathbb{R}^n$ :

$$\min_{x_i,z\in\mathbb{R}^n} \quad \sum_{i=1}^{N} f_i(x_i) \quad \text{s.t.} \quad x_i - z = 0 \quad \forall i = 1\dots N \tag{3.19}$$

The ADMM algorithm can be written as

$$x_i^{k+1} := \underset{x_i \in \mathbb{R}^n}{\operatorname{argmin}} \quad f_i(x_i) + y_i^{kT}(x_i - z^k) + \frac{\rho}{2}\left\|x_i - z^k\right\|_2^2 \tag{3.20}$$

$$z^{k+1} := \underset{z \in \mathbb{R}^n}{\operatorname{argmin}} \quad y_i^{kT}(x_i^{k+1} - z) + \frac{\rho}{2}\left\|x_i^{k+1} - z\right\|_2^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x_i^{k+1} + \frac{1}{\rho}y_i^k\right) \tag{3.21}$$

$$y_i^{k+1} := y_i^k + \rho\left(x_i^{k+1} - z_i^{k+1}\right) \tag{3.22}$$

This formulation can be simplified. We know from (3.21) and (3.22) that

$$z^{k+1} = \bar{x}^{k+1} + \frac{1}{\rho}\bar{y}^k$$
$$\bar{y}^{k+1} = \bar{y}^k + \rho\left(\bar{x}^{k+1} - z^{k+1}\right)$$

Inserting the first equation into the second one gives $\bar{y}^{k+1} = 0$ after the first iteration. Since then $z^{k+1} = \bar{x}^k$, ADMM can be written as

$$x_i^{k+1} := \underset{x_i \in \mathbb{R}^n}{\operatorname{argmin}} \quad f_i(x_i) + y_i^{kT}(x_i - \bar{x}^k) + \frac{\rho}{2}\left\|x_i - \bar{x}^k\right\|_2^2 \tag{3.23}$$

$$y_i^{k+1} := y_i^k + \rho\left(x_i^{k+1} - \bar{x}^{k+1}\right) \tag{3.24}$$

We can view ADMM as a method for solving problems in which the objective and constraints are distributed across multiple processors. And each processor handles only its own local variables. The primal local variables are updated according to their own objective function plus a linear and quadratic term in such a way that the variables converge to a common value, which is the solution of the full problem. If we look at the primal residual and its squared norm at iterate $k$

$$r^k = \sum_{i=1}^{N}(x_i^k - z^k) = \sum_{i=1}^{N}(x_i^k - \bar{x}^k)$$
$$\left\|r^k\right\|_2^2 = \sum_{i=1}^{N}\left\|x_i^k - \bar{x}^k\right\|_2^2$$

We see that this norm corresponds to $N$ times the standard deviation of the primal variables $x_i$, a natural measure of (lack of) consensus.

# 4 | A Distributed Algorithm for the Optimal Power Flow Problem

In this chapter, we firstly design an ADMM based algorithm for a broad class of optimization problems, of which the relaxation of the OPF problem is a special case. We develop then the algorithm for the special case of the OPF problem. This approach is inspired from [PL15].

## 4.1 Distributed Algorithm

Let's consider the following optimization problem :

$$\begin{aligned}
\min \quad & \sum_{i\in\mathcal{N}} f_i(x_i) && && (4.1a)\\
\text{over} \quad & \{x_i \mid i \in \mathcal{N}\} && && (4.1b)\\
\text{s.t.} \quad & \sum_{j\in N_i} A_{ij}x_j = 0 && \text{for} \quad i \in \mathcal{N} && (4.1c)\\
& x_i \in \cap_{r=0}^{R_i} \mathcal{K}_{ir} && \text{for} \quad i \in \mathcal{N}, && (4.1d)
\end{aligned}$$

where $x_i$ is a complex vector for each $i \in \mathcal{N}$, the objectif function is composed of the sum of convex functions $f_i(x_i)$ and the sets $\mathcal{K}_{ir}$ are convex non-empty sets. $A_{ij}$ are matrices linking the variables together, with appropriate dimensions ($j \in N_i, i \in \mathcal{N}$). A broad class of graphical optimization problems can be formulated as (4.1). Specifically, each node $i \in \mathcal{N}$ is associated with some local variables stacked as $x_i$, which belongs to an intersection of $R_i + 1$ local feasible sets $\mathcal{K}_{ir}$ and has a cost objective function $f_i(x_i)$. Variables in node $i$ are coupled with variables from their neighbour nodes in $N_i$ through linear constraints (4.1c). The objective is then to solve a minimal total cost across all the nodes.

The goal is to develop a distributed algorithm that solves (4.1) such that each node $i$ solves its own subproblem and only exchanges information with its neighbour. In order to apply the ADMM algorithm to (4.1), we need to reformulate it into the standard form of ADMM. We introduce two sets of slack variables $x$ and $y$ :

- $x_{ir}$ represents a copy of the original variable $x_i$ for $1 \leq r \leq R_i$. For convenience, we refer the original $x_i$ by $x_{i0}$.
- $y_{ij}$ represents the variables in node $i$ observed at node $j$, for $j \in N_i$.

Then (4.1) can be reformulated as

$$\begin{aligned}
\min \quad & \sum_{i\in\mathcal{N}} f_i(x_{i0}) && && \text{(4.2a)}\\
\text{over} \quad & x = \{x_{ir} \mid 0 \le r \le R_i,\ i \in \mathcal{N}\} && &&\\
& y = \{y_{ij} \mid j \in N_i,\ i \in \mathcal{N}\} && &&\\
\text{s.t.} \quad & \sum_{j\in N_i} A_{ij} y_{ji} = 0 && \forall\, i \in \mathcal{N} && \text{(4.2b)}\\
& x_{ir} \in \mathcal{K}_{ir} && \forall\, 0 \le r \le R_i,\ i \in \mathcal{N} && \text{(4.2c)}\\
& x_{ir} = y_{ii} && \forall\, 1 \le r \le R_i,\ i \in \mathcal{N} && \text{(4.2d)}\\
& x_{i0} = y_{ij} && \forall\, j \in N_i,\ i \in \mathcal{N} && \text{(4.2e)}
\end{aligned}$$

where $x$ and $y$ represent the two groups of variables in standard ADMM. Note that the consensus constraints (4.2d) and (4.2e) force all the duplicates $x_{ir}$ and $y_{ij}$ to be the same. Thus its solution $x_{i0}$ is also optimal to the original problem (4.1), and (4.2) has the benefit of falling into the general ADMM form.

Following the ADMM procedure, we relax the consensus constraints (4.2d) and (4.2e), whose Lagrangian multipliers are denoted by $\lambda_{ir}$ and $\mu_{ij}$, respectively. For a given penalty parameter $\rho$, the generalised augmented Lagrangian can be written as

$$\begin{aligned}
L_\rho(x, y, \lambda, \mu) = \sum_{i\in\mathcal{N}} \Bigg( & \sum_{r=1}^{R_i} \left( \langle \lambda_{ir},\ x_{ir} - y_{ii} \rangle + \frac{\rho}{2} \|x_{ir} - y_{ii}\|_{n_{ir}}^2 \right) + \\
& f_i(x_{i0}) + \sum_{j\in N_i} \left( \langle \mu_{ij}, x_{i0} - y_{ij} \rangle + \frac{\rho}{2} \|x_{i0} - y_{ij}\|_{m_{ij}}^2 \right) \Bigg)
\end{aligned} \tag{4.3}$$

where the parameter $n_{ir}$ and $m_{ij}$ depend on the problem we will show how to design them in section 4.2. Next, we show that both the $x$-update and $y$-update can be solved in a distributed manner, i.e. both of them can be decomposed into local subproblems that can be solved in parallel by each node $i$ with only neighbourhood communications.

First, we define the set of local variables for each node $i$, denoted by $\mathcal{A}_i$, which includes its own duplicates $x_{ir}$ with the associated multiplier $\lambda_{ir}$ for $0 \le r \le R_i$, and the "observations" $y_{ji}$ of variables from its neighbour $N_i$ with the associated multiplier $\mu_{ji}$, i.e.

$$\mathcal{A}_i := \{x_{ir}, \lambda_{ir} \mid 0 \le r \le R_i\} \cup \{y_{ji}, \mu_{ji} \mid j \in N_i\}. \tag{4.4}$$

Next, we show how does each node $i$ update $\{x_{ir} \mid 0 \le r \le R_i\}$ in the $x$-update and $\{y_{ji} \mid j \in N_i\}$ in the $y$-update.

### 4.1.1 x-update

In the $x$-update, the optimization subproblem that updates $x^{k+1}$ is

$$\min_{x\in\mathcal{X}} L_\rho(x, y^k, \lambda^k, \mu^k), \tag{4.5}$$

Where $\mathcal{X}$ is the feasible set of $x$. The objective can be written as a sum of local objectives as shown below

$$\begin{aligned} L_\rho(x, y^k, \lambda^k, \mu^k) &= \sum_{i\in\mathcal{N}} \left( \sum_{r=1}^{R_i} \left( \langle \lambda_{ir}^k, x_{ir} - y_{ii}^k \rangle + \frac{\rho}{2} \|x_{ir} - y_{ii}^k\|_{n_{ir}}^2 \right) + \right. \\ &\quad \left. f_i(x_{i0}) + \sum_{j\in N_i} \left( \langle \mu_{ij}^k, x_{i0} - y_{ij}^k \rangle + \frac{\rho}{2} \|x_{i0} - y_{ij}^k\|_{m_{ij}}^2 \right) \right) \\ &= \sum_{i\in\mathcal{N}} \sum_{r=0}^{R_i} H_{ir}(x_{ir}) - \sum_{i\in\mathcal{N}} \left( \sum_{r=0}^{R_i} \langle \lambda_{ir}^k, y_{ii}^k \rangle + \sum_{j\in N_i} \langle \mu_{ij}^k, y_{ij}^k \rangle \right) \end{aligned}$$

where

$$H_{ir}(x_{ir}) = \begin{cases} f_i(x_{i0}) + \sum_{j\in N_i} \left( \langle \mu_{ij}^k, x_{i0} \rangle + \frac{\rho}{2} \|x_{i0} - y_{ij}^k\|_{m_{ij}}^2 \right) & r = 0 \\ \langle \lambda_{ir}^k, x_{ir} \rangle + \frac{\rho}{2} \|x_{ir} - y_{ii}^k\|_{n_{ir}}^2 & r > 0 \end{cases} \tag{4.6}$$

Note that the last term of the objective function is independent of $x$ and can therefore be dropped since we minimize according to $x$. Then the problem (4.5) in the $x$-update can be written explicitly as

$$\begin{aligned} \min \quad & \sum_{i\in\mathcal{N}} \sum_{r=0}^{R_i} H_{ir}(x_{ir}) \\ \text{over} \quad & x = \{x_{ir} \mid 0 \le r \le R_i, i \in \mathcal{N}\} \\ \text{s.t.} \quad & x_{ir} \in \mathcal{K}_{ir} \qquad\qquad 0 \le r \le R_i,\ i \in \mathcal{N} \end{aligned}$$

where both the objective and constraint are separable for $0 \le r \le R_i$ and $i \in \mathcal{N}$. Thus it can be decomposed into $\sum_{i\in\mathcal{N}}(R_i+1)$ independent problems that can be solved in parallel. There are $R_i+1$ problems associated with each node $i$ and the $r_{th}$ one can be simply written as

$$\min_{x_{ir}\in\mathcal{K}_{ir}} \quad H_{ir}(x_{ir}) \tag{4.8}$$

whose solution is the new update of variables $x_{ir}$ for node $i$. In the above problem, the constants $y_{ij}^k, \mu_{ij}^k \in \mathcal{A}_j$ are not local to $i$ and are stored in $i$'s neighbours $j \in N_i$. Therefore, each node $i$ needs to collect $(y_{ij}, \mu_{ij})$ from all of its neighbours prior to solving (4.8).

### 4.1.2 y-update

In the $y$-update, the optimization problem that updates $y^{k+1}$ is

$$\min_{y\in\mathcal{Y}} \quad L_\rho(x^{k+1}, y, \lambda^k, \mu^k) \tag{4.9}$$

where the constraint set $\mathcal{Y}$ can be represented as a Cartesian product of $|\mathcal{N}|$ disjoint sets, i.e.

$$\mathcal{Y} := \otimes_{i\in\mathcal{N}} \{y_{ji}, j \in N_i \mid \sum_{j\in N_i} A_{ij} y_{ji} = 0\}.$$

The objective can be written as a sum of local objectives as below.

$$\begin{aligned} L_\rho(x^{k+1}, y, \lambda^k, \mu^k) \;=\; & \sum_{i\in\mathcal{N}} \left( \sum_{r=1}^{R_i} \left( \langle \lambda_{ir}^k, x_{ir}^{k+1} - y_{ii} \rangle + \frac{\rho}{2} \|x_{ir}^{k+1} - y_{ii}\|_{n_{ir}}^2 \right) + \right. \\ & \left. f_i(x_{i0}^{k+1}) + \sum_{j\in N_i} \left( \langle \mu_{ji}^k, x_{j0}^{k+1} - y_{ji} \rangle + \frac{\rho}{2} \|x_{j0}^{k+1} - y_{ji}\|_{m_{ji}}^2 \right) \right) \\ =\; & \sum_{i\in\mathcal{N}} G_i(\{y_{ji} \mid j \in N_i\}) + \sum_{i\in\mathcal{N}} \left( f_i(x_{i0}^{k+1}) + \sum_{r=0}^{R_i} \langle \lambda_{ir}^k, x_{ir}^{k+1} \rangle + \sum_{j\in N_i} \langle \mu_{ji}^k, x_{j0}^{k+1} \rangle \right), \end{aligned}$$

where the last term is independent of $y$ and

$$G_i(\{y_{ji} \mid j \in N_i\}) = \sum_{r=1}^{R_i} \left( -\langle \lambda_{ir}^k, y_{ii} \rangle + \frac{\rho}{2} \|x_{ir}^{k+1} - y_{ii}\|_{n_{ir}}^2 \right) + \sum_{j\in N_i} \left( -\langle \mu_{ji}^k, y_{ji} \rangle + \frac{\rho}{2} \|x_{j0}^{k+1} - y_{ji}\|_{m_{ji}}^2 \right)$$

Then the problem (4.9) in the $y$-update can be written explicitly as

$$\begin{aligned} \min \quad & \sum_{i\in\mathcal{N}} G_i(\{y_{ji} \mid j \in N_i\}) \\ \text{over} \quad & y = \{\{y_{ji} \mid j \in N_i\} \mid i \in \mathcal{N}\} \\ \text{s.t.} \quad & \sum_{j\in N_i} A_{ij} y_{ji} = 0 \qquad\qquad \forall i \in \mathcal{N} \end{aligned}$$

which can be decomposed into $|\mathcal{N}|$ subproblems. The subproblem associated with node $i$ is

$$\begin{aligned} \min \quad & G_i(\{y_{ji} \mid j \in N_i\}) && \text{(4.10a)} \\ \text{over} \quad & \{y_{ji} \mid j \in N_i\} && \text{(4.10b)} \\ \text{s.t.} \quad & \sum_{j\in N_i} A_{ij} y_{ji} = 0 && \text{(4.10c)} \end{aligned}$$

whose solution is the new update of $\{y_{ji} \mid j \in N_i\} \in \mathcal{A}_i$. In (4.10), the constants $x_{j0} \in \mathcal{A}_j$ are stored in $i$'s neighbor $j \in N_i$. Hence, each node $i$ needs to collect $x_{j0}$ from all of its neighbour prior to solving (4.10).

The problem (4.10) can be solved with a closed form solution. If we stack the real and imaginary part of the variables $\{y_{ji} \mid j \in N_i\}$ in a vector with appropriate dimensions and denote it as $z$. Then (4.10) takes the following form:

$$\begin{aligned} \min \quad & \frac{1}{2} z^T Q z + c^T z && \text{(4.11a)} \\ \text{over} \quad & z && \text{(4.11b)} \\ \text{s.t.} \quad & Az = 0 && \text{(4.11c)} \end{aligned}$$

where $Q$ is a positive diagonal matrix, $A$ is a full row rank real matrix, and $c$ is a real vector. $Q, c, A$ are derived from (4.10). The Lagrangian function is given by :

$$\mathcal{L}(z, \lambda) = \frac{1}{2} z^T Q z + c^T z + \lambda^T A z$$

To achieve optimality, we need that $\nabla_z \mathcal{L}(z, \lambda) = 0$, and since $Az = 0$, we have :

$$\begin{aligned} & Qz + c + A^T\lambda = 0 \\ \Rightarrow \quad & z + Q^{-1}c + Q^{-1}A^T\lambda = 0 \\ \Rightarrow \quad & Az + AQ^{-1}c + AQ^{-1}A^T\lambda = 0 \\ \Rightarrow \quad & AQ^{-1}c + AQ^{-1}A^T\lambda = 0 \\ \Rightarrow \quad & \lambda = -\left(AQ^{-1}A^T\right)^{-1} AQ^{-1}c \end{aligned}$$

By injecting the optimal value of $\lambda$ into the first equation, we get :

$$\begin{aligned} & Qz + c + A^T\lambda = 0 \\ \Rightarrow \quad & Qz + c - A^T\left(AQ^{-1}A^T\right)^{-1} AQ^{-1}c = 0 \\ \Rightarrow \quad & z = \left(Q^{-1}A^T\left(AQ^{-1}A^T\right)^{-1} AQ^{-1} - Q^{-1}\right)c \end{aligned}$$

### 4.1.3 dual update

The update of the dual variables consists in a gradient ascent with the penalty parameter $\rho$ as the step-size. The update can be written as

$$\lambda_{ir}^{k+1} = \lambda_{ir}^{k} + \rho\left(x_{ir}^{k+1} - y_{ii}^{k+1}\right) \qquad 0 \le r \le R_i,\ \forall i \in \mathcal{N} \tag{4.12a}$$

$$\mu_{ij}^{k+1} = \mu_{ij}^{k} + \rho\left(x_{i0}^{k+1} - y_{ij}^{k+1}\right) \qquad \forall j \in \mathcal{N}_i,\ \forall i \in \mathcal{N} \tag{4.12b}$$

In (4.12), the constants $y_{ij}^{k+1}$ are stored in $i$'s neighbours $j \in \mathcal{N}_i$. Hence, each node $i$ needs to collect $y_{ij}$ from all its neighbours prior to updating its dual variables.

### 4.1.4 Conclusion

In summary, the original problem (4.1) is decomposed into local subproblems that can be solved in a distributed manner using ADMM. At each iteration, each node $i$ solves (4.8) in the $x$-update and (4.11) in the $y$-update. There exists a closed form solution to the subproblem (4.11) in the $y$-update, and hence whether the original problem (4.1) can be solved efficiently in a distributed manner depends on the existence of efficient solutions to the subproblems (4.8) in the $x$-update, which depends on the realization of both the objectives $f_i(x_i)$ and the constraint sets $\mathcal{K}_{ir}$.

## 4.2 Application to the OPF Problem

In this section, we show the ROPF problem (2.10) is a special case of (4.1), and hence can be solved in a distributed manner using the method described in section 4.1. In particular, we show that the corresponding subproblems in the $x$-update can be solved efficiently.

### 4.2.1 ADMM Formulation

In order to apply the aforementioned method, let's show that the relaxation of the OPF problem can be reformulated according to the previous model (4.1). Let's consider

$$x_i := \{\{x_{i0}, x_{i1}\} \mid x_{i0} := \{S_{i0}, \ell_{i0}, v_{i0}, s_{i0}\},\ x_{i1} := \{v_{i1}\}\} \tag{4.13}$$

$$\mathcal{K}_{i0} := \left\{ x_{i0} \mid \begin{pmatrix} v_{i0} & S_{i0} \\ S_{i0}^* & \ell_{i0} \end{pmatrix} \in \mathbb{S}_+, \left\{ s_{i0}^\phi \in \mathcal{I}_i^\phi \mid \phi \in \Phi_i \right\} \right\} \tag{4.14}$$

$$\mathcal{K}_{i1} := \{x_{i1} \mid \underline{v}_i^\phi \leq v_{i1}^{\phi\phi} \leq \overline{v}_i^\phi, \phi \in \Phi_i\} \tag{4.15}$$

Then (2.10) takes the form of (4.1) with $R_i = 1$ for all $i \in \mathcal{N}$, where (2.10b)–(2.10c) correspond to (4.1c) and (2.10d)–(2.10f) correspond to (4.1d). Following the procedure in section 4.1, we introduce two sets of slack variables: $x$ and $y$.

The reformulation of (2.10) according to (4.2) is then :

$$\min \quad \sum_{i\in\mathcal{N}} \sum_{\phi\in\Phi_i} f_i^\phi \left( s_{i0}^{\phi\,(x)} \right) \tag{4.16a}$$

$$\text{over} \quad x := \{x_{ir} \mid 0 \leq r \leq 1,\ i \in \mathcal{N}\}$$
$$y := \{y_{ji} \mid j \in N_i,\ i \in \mathcal{N}\}$$

$$\text{s.t.} \quad \mathcal{P}_i \left( v_{A_i i}^{(y)} \right) = v_{ii}^{(y)} - z_i S_{ii}^{*\,(y)} - S_{ii}^{(y)} z_i^* + z_i \ell_{ii}^{(y)} z_i^* \qquad i \in \mathcal{N}_+ \tag{4.16b}$$

$$s_{ii}^{(y)} + \operatorname{diag}\left( \sum_{i\in C_i} \mathcal{P}_i \left( S_{ji}^{(y)} - z_j \ell_{ji}^{(y)} \right) \right) = \operatorname{diag}\left( S_{ii}^{(y)} \right) \qquad i \in \mathcal{N} \tag{4.16c}$$

$$\begin{pmatrix} v_{i0}^{(x)} & S_{i0}^{(x)} \\ S_{i0}^{*\,(x)} & \ell_{i0}^{(x)} \end{pmatrix} \succeq 0 \qquad i \in \mathcal{N}_+ \tag{4.16d}$$

$$s_{i0}^{\phi\,(x)} \in \mathcal{I}_i^\phi \qquad \phi \in \Phi_i \qquad i \in \mathcal{N}_+ \tag{4.16e}$$

$$\underline{v}_i^\phi \leq v_{i1}^{\phi\phi\,(x)} \leq \overline{v}_i^\phi \qquad \phi \in \Phi_i \qquad i \in \mathcal{N}_+ \tag{4.16f}$$

$$x_{ir} - y_{ii} = 0 \qquad r = 1 \qquad i \in \mathcal{N} \tag{4.16g}$$

$$x_{i0} - y_{ij} = 0 \qquad j \in N_i \qquad i \in \mathcal{N} \tag{4.16h}$$

where we put superscript $(\,\cdot\,)^{(x)}$ and $(\,\cdot\,)^{(y)}$ on each variable to denote whether the variable is updated in the $x$-update or $y$-update. The x-update considers the constraints (4.16d)-(4.16f) (form the

feasible set $\mathcal{X}$), whereas the y-update considers the constraints (4.16b) and (4.16c) (form the feasible set $\mathcal{Y}$). And the consensus constraints will be relaxed as explained before.

Note that each node $i$ does not need full information of its neighbours. Specifically, for each node $i$, only voltage information $v_{A_i i}^{(y)}$ is needed from its parent $A_i$, and branch power $S_{ji}^{(y)}$ and current $\ell_{ji}^{(y)}$ information from its children $j \in C_i$. Thus, $y_{ij}$ contains only partial information, i.e.

$$y_{ij} := \begin{cases} (S_{ii}^{(y)}, \ell_{ii}^{(y)}, v_{ii}^{(y)}, s_{ii}^{(y)}) & j = i \\ (S_{iA_i}^{(y)}, \ell_{iA_i}^{(y)}) & j = A_i \\ (v_{ij}^{(y)}) & j \in C_i \end{cases}$$

Meaning that, for each node and besides local variables, $y$ holds some variables corresponding to the voltage of their parent and the branch power and current of the lines connecting them to their children. On the other hand, $x_{i0}$ needs only to hold all the local variables and $x_{i1}$ only needs to have a duplicate of $v_i$, i.e.

$$x_{ir} := \begin{cases} (S_{i0}^{(x)}, \ell_{i0}^{(x)}, v_{i0}^{(x)}, s_{i0}^{(x)}) & r = 0 \\ (v_{i1}^{(x)}) & r = 1 \end{cases}$$

As a result, $x_{ir}$, $y_{ii}$ and $x_{i0}$, $y_{ij}$ do not consist of the same components. Here, we abuse notations in both (4.16g) and (4.16h), which are composed of components that appear in both items, i.e.

$$x_{i0} - y_{ij} := \begin{cases} (S_{i0}^{(x)} - S_{ii}^{(y)}, \ell_{i0}^{(x)} - \ell_{ii}^{(y)}, v_{i0}^{(x)} - v_{ii}^{(y)}, s_{i0}^{(x)} - s_{ii}^{(y)}) & j = i \\ (S_{i0}^{(x)} - S_{iA_i}^{(y)}, \ell_{i0}^{(x)} - \ell_{iA_i}^{(y)}) & j = A_i \\ (v_{i0}^{(x)} - v_{ij}^{(y)}) & j \in C_i \end{cases}$$

$$x_{i1} - y_{ii} := \Big\{ (v_{i1}^{(x)} - v_{ii}^{(y)})$$

In the context of our application, let's define $\|x_{i0} - y_{ij}\|_{m_{ij}}^2$ by :

$$\begin{aligned} \|x_{io} - y_{ij}\|_{m_{ij}}^2 &= (2|C_i| + 3)\,\|S_{i0} - S_{ii}\|_2^2 + (|C_i| + 1)\,\|\ell_{i0} - \ell_{ii}\|_2^2 + 2\|v_{i0} - v_{ii}\|_2^2 + \|s_{i0} - s_{ii}\|_2^2 & j = i \\ &= \|S_{i0} - S_{iA_i}\|_2^2 + \|\ell_{i0} - \ell_{iA_i}\|_2^2 & j = A_i \\ &= \|v_{i0} - v_{ij}\|_2^2 & j \in C_i \end{aligned}$$

This tailor-made norm helps us to simplify the objective function of the $x$-update (see next section).

### 4.2.2 x-update

In the $x$-update, there are 2 subproblems associated with each bus $i$, the update of $x_{i0}$ and $x_{i1}$. Remember that the problem associated with the $x_{i0}$-update is given by

$$\begin{aligned}
\min \quad & H_{i0}(x_{i0}) && \text{(4.17a)}\\
\text{over} \quad & x_{i0} = \{v_{i0}, \ell_{i0}, S_{i0}, s_{i0}\} && \text{(4.17b)}\\
\text{s.t.} \quad & \begin{pmatrix} v_{i0} & S_{i0} \\ S_{i0}^* & \ell_{i0} \end{pmatrix} \succeq 0 && \text{(4.17c)}\\
& s_{i0}^{\phi} \in \mathcal{I}_i^{\phi} \qquad \phi \in \Phi_i, && \text{(4.17d)}
\end{aligned}$$

where,

$$H_{i0}(x_{i0}) = f_{i0}(x_{i0}) + \sum_{j \in \mathcal{N}_i} \left( \langle \mu_{ij}, x_{i0} \rangle + \frac{\rho}{2} \|x_{i0} - y_{ij}\|_{M_{ij}}^2 \right)$$

Since the objective function only depends on the power injection, we assume that $f_{i0}(x_{i0}) = f_{i0}(s_{i0})$. Then, we can develops $H_{i0}(x_{i0})$ in term of each variable $\{S_{i0}, l_{i0}, v_{i0}, s_{i0}\}$, such that

$$H_{i0}(x_{i0}) = H_{i0}^{(1)}(S_{i0}) + H_{i0}^{(2)}(\ell_{i0}) + H_{i0}^{(3)}(v_{i0}) + H_{i0}^{(4)}(s_{i0})$$

Remember that any terme independent of $x_{i0}$ can be removed without changing the optimal solution. In that case, those functions can be expressed by (See appendix A.2 for the full development) :

$$\begin{aligned}
H_{i0}^{(1)}(S_{i0}) &= 2\ \frac{\rho}{2}\left(|C_i| + 2\right) \left\| S_{i0} - \hat{S}_i \right\|_2^2 \\
H_{i0}^{(2)}(\ell_{i0}) &= \frac{\rho}{2}\left(|C_i| + 2\right) \|\ell_{i0} - \hat{s}_i\|_2^2 \\
H_{i0}^{(2)}(v_{i0}) &= \frac{\rho}{2}\left(|C_i| + 2\right) \|v_{i0} - \hat{s}_i\|_2^2 \\
H_{i0}^{(4)}(s_{i0}) &= f_{i0}(s_{i0}) + \frac{\rho}{2} \|s_{i0} - \hat{s}_i\|_2^2
\end{aligned}$$

where the values $\left\{\hat{S}_i,\ \hat{\ell}_i,\ \hat{v}_i,\ \hat{s}_i\right\}$ are some constant parameters that depend on local primal/dual variables but also on primal/dual variables located at i's neighbours. At the end, the objective function can be written as :

$$H_{i0}(x_{i0}) = f_{i0}(x_{i0}) + \sum_{j \in \mathcal{N}_i} \left( \langle \mu_{ij}, x_{i0} \rangle + \frac{\rho}{2} \|x_{i0} - y_{ij}\|_{M_{ij}}^2 \right) \tag{4.18}$$

$$= \underbrace{\frac{\rho}{2}\left(|C_i| + 2\right) \left\| \begin{pmatrix} v_{i0} & S_{i0} \\ S_{i0}^* & l_{i0} \end{pmatrix} - \begin{pmatrix} \hat{v}_i & \hat{S}_i \\ \hat{S}_i^* & \hat{\ell}_i \end{pmatrix} \right\|_2^2}_{H_i^{(1)}(S_{i0}, \ell_{i0}, v_{i0})} + \underbrace{f_{i0}(s_{i0}) + \frac{\rho}{2} \|s_{i0} - \hat{s}_i\|_2^2}_{H_i^{(2)}(s_{i0})} \tag{4.19}$$

Hence, the objective function $H_{i0}(x_{i0})$ can be decomposed into two parts, where the first part $H_i^{(1)}(S_{i0}^{(x)}, \ell_{i0}^{(x)}, v_{i0}^{(x)})$ involves variables $(S_{i0}^{(x)}, \ell_{i0}^{(x)}, v_{i0}^{(x)})$ and the second part $H_i^{(2)}(s_{i0}^{(x)})$ involves $s_{i0}^{(x)}$. Note that the constraint can also be separated into two independent constraints. The problem can be decomposed into two subproblems, where the first one solves the optimal $(S_{i0}^{(x)}, \ell_{i0}^{(x)}, v_{i0}^{(x)})$ and the second one solves the optimal $s_{i0}^{(x)}$.

The first subproblem can be written explicitly as

$$\begin{aligned} \min \quad & H_i^{(1)}\left(S_{i0}, \ell_{i0}, v_{i0}\right) \\ \text{over} \quad & S_{i0}, \ell_{i0}, v_{i0} \\ \text{s.t.} \quad & \begin{pmatrix} v_{i0} & S_{i0} \\ S_{i0}^* & l_{i0} \end{pmatrix} \succeq 0 \end{aligned}$$

which can be solved using eigen-decomposition of a 6×6 matrix via the following theorem (see appendix [REF] for the proof).

**Theorem 2.** *Suppose $W \in \mathbb{S}^n$ and denote $X(W) := \arg\min_{X \in \mathbb{S}_+^n} \|X - W\|_2^2$. Then $X(W) = \sum_{i:\lambda_i>0} \lambda_i u_i u_i^*$, where $\lambda_i, u_i$ are the $i_{th}$ eigenvalue and orthonormal eigenvector of matrix $W$, respectively.*

The second problem is

$$\begin{aligned} \min \quad & f_i\left(s_{i0}\right) + \frac{\rho}{2}\|s_{i0} - \hat{s}_i\|_2^2 && \text{(4.20a)} \\ \text{over} \quad & s_{i0} && \text{(4.20b)} \\ \text{s.t.} \quad & s_{i0}^{\phi} \in I_i^{\phi} \qquad \forall \phi \in \Phi_i && \text{(4.20c)} \end{aligned}$$

Recall that if $f_i(s_{i0}) = \sum_{\phi \in \Phi_i} f_i^{\phi}(s_{i0}^{\phi})$, then both the objective and constraint are separable for each phase $\phi \in \Phi_i$. Therefore, the problem can be further decomposed into $|\Phi_i|$ subproblems as below.

$$\begin{aligned} \min \quad & f_i^{\phi}\left(s_{i0}^{\phi}\right) + \frac{\rho}{2}\left\|s_{i0}^{\phi} - \hat{s}_i^{\phi}\right\|_2^2 \\ \text{over} \quad & s_{i0}^{\phi} \\ \text{s.t.} \quad & s_{i0}^{\phi} \in I_i^{\phi} \end{aligned}$$

Let's define a closed form solution for this problem if the objective function takes the form : $f_i^{\phi} = \frac{\alpha_i}{2} p_i^2 + \beta_i p_i$, where $\alpha_i \geq 0$. Note that this form includes the possibility to minimize the total line loss and/or the generation cost (quadratic cost). And assume that

$$\mathcal{I}_i^{\phi} = \left\{ p + iq \mid p \in [\underline{p}_i, \bar{p}_i],\ q \in [\underline{q}_i, \bar{q}_i] \right\}$$

In this case, the optimisation problem can be rewritten as :

$$\begin{aligned} \min \quad & \frac{a_2}{2} p^2 + a_1 p + \frac{b_2}{2} q^2 + b_1 q \\ \text{over} \quad & p,\ q \\ \text{s.t.} \quad & \underline{p} \leq p \leq \bar{p} \\ & \underline{q} \leq q \leq \bar{q} \end{aligned}$$

And the closed-form solution is given by :

$$p^* = \left[ -\frac{a_1}{a_2} \right]_{\underline{p}}^{\bar{p}} \qquad q^* = \left[ -\frac{b_1}{b_2} \right]_{\underline{q}}^{\bar{q}}$$

where $[\,.\,]_a^b = \min\{b, \max\{a,\ .\ \}\}$.

The update of $x_{i1}$ consists of only one component $v_{i1}$. It can be written explicitly as

$$\begin{aligned} \min \quad & H_{i1}(v_{i1}) \\ \text{over} \quad & v_{i1} \\ \text{s.t.} \quad & \underline{v}_i^{\phi} \le v_{i1}^{\phi\phi} \le \bar{v}_i^{\phi} \qquad \phi \in \Phi_i \end{aligned}$$

where

$$\begin{aligned} H_{i1}(x_{i1}) &= \langle \lambda_{i1}, x_{i1} \rangle + \frac{\rho}{2} \|x_{i1} - y_{ii}\|_{n_{i1}}^2 \\ &= \langle \lambda_{i1}, v_{i1} \rangle + \frac{\rho}{2} \|v_{i1} - v_{ii}\|_2^2 \\ &= \frac{\rho}{2} \left\| v_{i1} - v_{ii} + \frac{\lambda_{i1}}{\rho} \right\|_2^2 + \text{Cst} \end{aligned}$$

Giving us this equivalent formulation

$$\min \quad \frac{1}{2} \left\| v_{i1} - v_{ii} + \frac{\lambda_{i1}}{\rho} \right\|_2^2 \tag{4.21a}$$

$$\text{over} \quad v_{i1} \tag{4.21b}$$

$$\text{s.t.} \quad \underline{v}_i^{\phi} \le v_{i1}^{\phi\phi} \le \bar{v}_i^{\phi} \qquad \phi \in \Phi_i \tag{4.21c}$$

This problem enjoys a closed-form solution, given by

$$v_{i1}^{\phi_1\phi_2} = \begin{cases} \left[ v_{ii} - \frac{\lambda_{i1}}{\rho} \right]_{\underline{v}_i^{\phi_1}}^{\bar{v}^{\phi_1}} & \text{if} \quad \phi_1 = \phi_2 \\ v_{ii} - \frac{\lambda_{i1}}{\rho} & \text{if} \quad \phi_1 \neq \phi_2 \end{cases}$$

In brief, the $x$-update for each bus $i$ can be solved through a closed form solution and an eigen-decomposition of a $6 \times 6$ matrix. In the above problems, some parameters are not local to $i$ and are stored in $i$'s neighbours $j \in \mathcal{N}_i$. Therefore, each node $i$ needs to collect them from all its neighbours prior to solving (4.17) and (4.20).

### 4.2.3 y-update

In the $y$-update, each node $i$ must solved its own subproblem of the form (4.10). In the special case of the optimal power flow, the subproblem associated with each node $i$ can be written as

$$\begin{aligned} \min \quad & G_i\Big(\{y_{ji} \mid j \in \mathcal{N}_i\}\Big) \\ \text{over} \quad & \{y_{ji} \mid j \in \mathcal{N}_i\} \\ \text{s.t.} \quad & \mathcal{P}_i(v_{A_i i}) = v_{ii} - z_i S_{ii}^* - S_{ii} z_i^* + z_i \ell_{ii} z_i^* \\ & s_{ii} = -\text{diag}\left(\sum_{j \in C_i} \mathcal{P}_i(S_{ji} - z_j \ell_{ji}) - S_{ii}\right) \end{aligned}$$

where

$$G_i\Big(\{y_{ji} \mid j \in \mathcal{N}_i\}\Big) = -\langle \lambda_{i1}, v_{ii} \rangle + \frac{\rho}{2}\|v_{i1} - v_{ii}\|_2^2 + \sum_{j \in \mathcal{N}_i} -\langle \mu_{ji}, y_{ji} \rangle + \frac{\rho}{2}\|x_{j0} - y_{ji}\|_{M_{ji}}^2$$

#### Objective Function

We can decomposed this objective function according to each variable contained in $y$ such that :

$$\begin{aligned} G_i\Big(\{y_{ji} \mid j \in \mathcal{N}_i\}\Big) = \; & G_i^{(1)}(S_{ii}) + G_i^{(2)}(\ell_{ii}) + G_i^{(3)}(v_{ii}) + G_i^{(4)}(s_{ii}) \\ & + \sum_{j \in C_i} G_i^{(5,j)}(S_{ji}) + \sum_{j \in C_i} G_i^{(6,j)}(\ell_{ji}) \\ & + \; G_i^{(7)}(v_{A_i i}) \end{aligned}$$

See appendix A.3 for the complete development. All those functions are composed of a quadratic and linear term, and can therefore take the form of (4.11). Indeed, if we stack the real and imaginary part of the variables $\{y_{ji} \mid j \in N_i\}$ in a vector with appropriate dimensions and denote it as $z$. Then, we can reformulate the objective function in the following form :

$$\frac{1}{2} z^T Q z + c^T z$$

If we denote $A(:)$ as the vectorisation of the matrix $A$ which converts the matrix into a column vector, the inner product between two matrices can be reformulated in terms of their vectorization :

$$\langle A, B \rangle = \text{Re}\Big\{\text{Tr}(A^* B)\Big\} = \text{Re}\{A(:)^* B(:)\} = \text{Re}\Big\{A(:)^T\Big\}\,\text{Re}\{B(:)\} + \text{Im}\Big\{A(:)^T\Big\}\,\text{Im}\{B(:)\}$$

By using this formula, we can develop the objective function (see appendix [REF] for the complete development). The matrix $Q$ and the vector $c$ are given by :

$$
z = \begin{bmatrix}
\operatorname{Re}\{S_{ii}(:)\} \\
\operatorname{Re}\{\ell_{ii}(:)\} \\
\operatorname{Re}\{v_{ii}(:)\} \\
\operatorname{Re}\{s_{ii}(:)\} \\
\operatorname{Re}\{v_{A_i i}(:)\} \\
\operatorname{Re}\{S_{ji}(:)\} \\
\operatorname{Re}\{\ell_{ji}(:)\} \\
\\
\operatorname{Im}\{S_{ii}(:)\} \\
\operatorname{Im}\{\ell_{ii}(:)\} \\
\operatorname{Im}\{v_{ii}(:)\} \\
\operatorname{Im}\{s_{ii}(:)\} \\
\operatorname{Im}\{v_{A_i i}(:)\} \\
\operatorname{Im}\{S_{ji}(:)\} \\
\operatorname{Im}\{\ell_{ji}(:)\}
\end{bmatrix}
\quad
c = -\begin{bmatrix}
\operatorname{Re}\left\{\mu_{ii}^{(1)}(:) + \rho\left(2|C_i|+3\right) S_{i0}^{+}(:)\right\} \\
\operatorname{Re}\left\{\mu_{ii}^{(2)}(:) + \rho\left(|C_i|+1\right) \ell_{i0}^{+}(:)\right\} \\
\operatorname{Re}\left\{\lambda_{i1}(:) + \mu_{ii}^{(3)}(:) + \rho v_{i1}^{+}(:) + 2\rho v_{i0}^{+}(:)\right\} \\
\operatorname{Re}\left\{\mu_{ii}^{(4)} + \rho s_{i0}^{+}\right\} \\
\operatorname{Re}\left\{\mu_{A_i i}^{(3)}(:) + \rho v_{A_i 0}^{+}(:)\right\} \\
\operatorname{Re}\left\{\mu_{ji}^{(1)}(:) + \rho S_{j0}^{+}(:)\right\} \\
\operatorname{Re}\left\{\mu_{ji}^{(2)}(:) + \rho \ell_{j0}^{+}(:)\right\} \\
\\
\operatorname{Im}\left\{\mu_{ii}^{(1)}(:) + \rho\left(2|C_i|+3\right) S_{i0}^{+}(:)\right\} \\
\operatorname{Im}\left\{\mu_{ii}^{(2)}(:) + \rho\left(|C_i|+1\right) \ell_{i0}^{+}(:)\right\} \\
\operatorname{Im}\left\{\lambda_{i1}(:) + \mu_{ii}^{(3)}(:) + \rho v_{i1}^{+}(:) + 2\rho v_{i0}^{+}(:)\right\} \\
\operatorname{Im}\left\{\mu_{ii}^{(4)} + \rho s_{i0}^{+}\right\} \\
\operatorname{Im}\left\{\mu_{A_i i}^{(3)}(:) + \rho v_{A_i 0}^{+}(:)\right\} \\
\operatorname{Im}\left\{\mu_{ji}^{(1)}(:) + \rho S_{j0}^{+}(:)\right\} \\
\operatorname{Im}\left\{\mu_{ji}^{(2)}(:) + \rho \ell_{j0}^{+}(:)\right\}
\end{bmatrix}
\quad
Q = \rho \ \operatorname{diag}\begin{bmatrix}
\left(2|C_i|+3\right) I_{|\Phi_i|^2} \\
\left(|C_i|+1\right) I_{|\Phi_i|^2} \\
3 I_{|\Phi_i|^2} \\
I_{|\Phi_i|} \\
I_{|\Phi_i|^2} \\
I_{|\Phi_i|^2} \\
I_{|\Phi_i|^2} \\
\\
\left(2|C_i|+3\right) I_{|\Phi_i|^2} \\
\left(|C_i|+1\right) I_{|\Phi_i|^2} \\
3 I_{|\Phi_i|^2} \\
I_{|\Phi_i|} \\
I_{|\Phi_i|^2} \\
I_{|\Phi_i|^2} \\
I_{|\Phi_i|^2}
\end{bmatrix}
$$

**Constraints**

The two constraints associated with the $y$-update must also be reformulated in the form of $Az = 0$. The kronecker product is frequently used together with the vectorization to express matrix multiplications. In particular,

$$\operatorname{vec}(ABC) = \left(C^T \otimes A\right) \operatorname{vec}(B) \tag{4.22}$$

Let's express the fist constraint according to the vectorization of $S, \ell, v$ and $v_A$ (For notational convenience, the indice $i$ is skipped).

$$\mathcal{P}_i(v_A) = v - \underbrace{\left(zS^* + Sz^*\right)}_{(1)} + \underbrace{z\ell z^*}_{(2)}$$

Since the variables are all complex, this equality must hold for the real part and the imaginary part. Define for each complex variable $z$ the following decomposition, $z = z_1 + i\, z_2$. Then the real and imaginary parts of (1) are

$$\operatorname{vec}\left(\operatorname{Re}\{zS^* + Sz^*\}\right) = \underbrace{\left((z_1 \otimes I) + (I \otimes z_1)P\right)}_{M_1} \operatorname{vec}(S_1) + \underbrace{\left((z_2 \otimes I) + (I \otimes z_2)P\right)}_{M_2} \operatorname{vec}(S_2)$$

$$\operatorname{vec}\left(\operatorname{Im}\{zS^* + Sz^*\}\right) = \underbrace{\left((I \otimes z_2)P - (z_2 \otimes I)\right)}_{M_3} \operatorname{vec}(S_1) + \underbrace{\left((z_1 \otimes I) - (I \otimes z_1)P\right)}_{M_4} \operatorname{vec}(S_2)$$

where $P$ is defined such that $\text{vec}\left(A^T\right) = P\,\text{vec}\,(A)$. In the same way, the real and imaginary parts of (2) are :

$$\text{vec}\left(\text{Re}\{z\ell z^*\}\right) = \underbrace{\left((z_1 \otimes z_1) + (z_2 \otimes z_2)\right)}_{P_1}\text{vec}\,(\ell_1) + \underbrace{\left((z_2 \otimes z_1) - (z_1 \otimes z_2)\right)}_{P_2}\text{vec}\,(\ell_2)$$
$$\text{vec}\left(\text{Im}\{z\ell z^*\}\right) = \underbrace{\left((z_1 \otimes z_2) - (z_2 \otimes z1)\right)}_{-P_2}\text{vec}\,(\ell_1) + \underbrace{\left((z_1 \otimes z_1) + (z_2 \otimes z_2)\right)}_{P_1}\text{vec}\,(\ell_2)$$

**Real Part**

$$\text{vec}\left(\mathcal{P}\left(v_{A_i i,1}\right)\right) = \text{vec}\,(v_{ii,1}) - M_1\text{vec}\,(S_{ii,1}) - M_2\text{vec}\,(S_{ii,2}) + P_1\text{vec}\,(\ell_{ii,1}) - P_2\text{vec}\,(\ell_{ii,2}) \tag{4.23}$$

**Imaginary Part**

$$\text{vec}\left(\mathcal{P}\left(v_{A_i i,2}\right)\right) = \text{vec}\,(v_{ii,2}) - M_3\text{vec}\,(S_{ii,1}) - M_4\text{vec}\,(S_{ii,2}) - P_2\text{vec}\,(\ell_{ii,1}) + P_1\text{vec}\,(\ell_{ii,2}) \tag{4.24}$$

The same method can be used to expressed the second constraint in term of the vectorisation of the variables.

$$s_{ii} + \text{diag}\left(\sum_{j\in C_i} S_{ji} - z_j\ell_{ji}\right) = \text{diag}\,(S_{ii})$$

As before, the equality must hold for the real and imaginary part. Let's define $M$ such that $\text{diag}(A) = M\text{vec}\,(A)$. Then, the real and imaginary parts of $z_j\, l_{ji}$ can be reformulated using (4.22)

$$\text{vec}\left(\text{Re}\{z_j\ell_{ji}\}\right) = \text{vec}\,(z_{j,1}\ell_{ji,1} - z_{j,2}\ell_{ji,2}) = \underbrace{(I \otimes z_{j,1})}_{N_{j,1}}\text{vec}\,(\ell_{ji,1}) - \underbrace{(I \otimes z_{j,2})}_{N_{j,2}}\text{vec}\,(\ell_{ji,2})$$
$$\text{vec}\left(\text{Im}\{z_j\ell_{ji}\}\right) = \text{vec}\,(z_{j,2}\ell_{ji,1} + z_{j,1}\ell_{ji,2}) = \underbrace{(I \otimes z_{j,2})}_{N_{j,2}}\text{vec}\,(\ell_{ji,1}) + \underbrace{(I \otimes z_{j,1})}_{N_{j,1}}\text{vec}\,(\ell_{ji,2})$$

**Real Part**

$$s_{ii,1} + M\sum_{j\in C_i}\left(\text{vec}\,(S_{ji,1}) - N_{j,1}\text{vec}\,(\ell_{ji,1}) + N_{j,2}\text{vec}\,(\ell_{ji,2})\right) = M\,\text{vec}\,(S_{ii,1}) \tag{4.25}$$

**Imaginary Part**

$$s_{ii,2} + M\sum_{j\in C_i}\left(\text{vec}\,(S_{ji,2}) - N_{j,2}\text{vec}\,(\ell_{ji,1}) - N_{j,1}\text{vec}\,(\ell_{ji,2})\right) = M\,\text{vec}\,(S_{ii,2}) \tag{4.26}$$

We can know combined the constraints (4.23)-(4.26) to form the matrix $A$ such that $Az = 0$.

$$A = \begin{bmatrix} -M_1 & P_1 & \mathbf{I} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & -M_2 & P_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -M_3 & -P_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -M_4 & P_1 & \mathbf{I} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} \\ & & & & & & & & & & & & & \\ -M & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & M & -MN_{j,1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & MN_{j,2} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -MN_{j,2} & -M & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & M & -MN_{j,1} \end{bmatrix}$$

**A small Improvement** Note that the algorithm will converge towards a solution such that the voltages $v$ and the branch currents $\ell$ are hermitian matrices. This constraint is enforced in the $x$-update but not in the $y$-update. That's a part of the algorithm's job to force the voltages and the branch currents contained in $y$ to be hermitian. Nevertheless we could argue that if we also enforce this constraint in the $y$-update, there certainly will be a decrease of the number of iterations since both $x$ and $y$ are already looking for variables satisfying this constraint. Hence, less work should be done to find a agreement between $x$ and $y$ since they already agree on this constraint.

To force a matrix $X = X_1 + iX_2$ to be hermitian, we need that $X_1 = X_1^T$ and $X_2 = -X_2^T$. We can reformulate those constraints in term of the vectorization of $X_1$ and $X_2$.

$$\begin{aligned} X_1 = X_1^T &\Leftrightarrow \quad \text{vec}\,(X_1) = \text{vec}\left(X_1^T\right) &\Leftrightarrow \quad (\mathbf{I} - P)\text{vec}\,(X_1) = \mathbf{0} \\ X_2 = -X_2^T &\Leftrightarrow \quad \text{vec}\,(X_2) = -\text{vec}\left(X_2^T\right) &\Leftrightarrow \quad (\mathbf{I} + P)\text{vec}\,(X_2) = \mathbf{0} \end{aligned}$$

Where $P$ is defined such that $\text{vec}\left(X^T\right) = P\text{vec}\,(X)$. Fortunately, those constraints are also of the form $Az = 0$, and can therefore be added to the matrix $A$ already built. We are now able to enforce for each node $i$ that its matrices $v_{ii}$, $l_{ii}$, $v_{A_i i}$ and $\ell_{ji}$, $\forall j \in C_i$, must be hermitian.

#### Solution

It has been shown that the y-update can be reformulated into the form of (4.11). We can now apply the analytical solution of this problem :

$$z = \underbrace{\left(Q^{-1}A^T\left(AQ^{-1}A^T\right)^{-1}AQ^{-1} - Q^{-1}\right)}_{T} c$$

where $Q$ is a positive diagonal matrix, $A$ is a full row rank real matrix, and $c$ is a real vector.

Note that only the vector $c$ changes over time, whereas $A$ and $Q$ don't change from one iteration to another. Thus, they can be formed at the beginning of the iterative process for each node in order to avoid repeating their construction and their inversion at each iteration. Only the matrix $T$ must be stored locally for each node $i \in \mathcal{N}$. In the end, the resolution of the $y$-update comes down to a simple multiplication between a square matrix and a vector.

### 4.2.4 Dual Update

The update of the dual variables is much more straightforward than the update of the primal variables. It consists of a gradient ascent. Each node $i$ must perform the following updates :

$$\begin{aligned} \mu_{ii}^{(1)+} &= \mu_{ii}^{(1)} + \rho\left(S_{i0}^+ - S_{ii}^+\right) & \mu_{ji}^{(1)+} &= \mu_{ji}^{(1)} + \rho\left(S_{j0}^+ - S_{ji}^+\right) & \forall j \in C_i \\ \mu_{ii}^{(2)+} &= \mu_{ii}^{(2)} + \rho\left(\ell_{i0}^+ - \ell_{ii}^+\right) & \mu_{ji}^{(2)+} &= \mu_{ji}^{(2)} + \rho\left(\ell_{j0}^+ - \ell_{ji}^+\right) & \forall j \in C_i \\ \mu_{ii}^{(3)+} &= \mu_{ii}^{(3)} + \rho\left(v_{i0}^+ - v_{ii}^+\right) & \mu_{A_i i}^+ &= \mu_{A_i i} + \rho\left(v_{A_i 0}^+ - v_{A_i i}^+\right) & \\ \mu_{ii}^{(4)+} &= \mu_{ii}^{(4)} + \rho\left(s_{i0}^+ - s_{ii}^+\right) & \lambda_i^+ &= \lambda_i + \rho\left(v_{i1}^+ - v_{i0}^+\right) & \end{aligned}$$

Note that some of the constant are stored in $i$'s neighbours $j \in \mathcal{N}_i$. Hence, each node $i$ needs to collect $S_{j0}$, $\ell_{j0}$ from its children and $v_{A_i 0}$ from its parent prior to solving the dual update.

### 4.2.5 Conclusion

In summary, the original relaxation of the OPF problem has been decomposed into local subproblems that can be solved in a distributed manner using ADMM. The problem structure has been exploited in such a way that the subproblems ($x$-update and $y$-update) are reduced to either a closed-form solution, or a eigen-decomposition of a $6 \times 6$ hermitian matrix, which significantly reduce the convergence time. Moreover, the nodes only exchange limited messages with their immediate neighbours.

# Part III

# Numerical Results

# 5 | Implementation

This section addresses the implementation of the algorithm described in section 4.2. In particular, we explained how the starting point is chosen, how a network is represented, how to quantify the exactness of the solution obtained, etc.

## 5.1 Structure of the network

Each node in the network is denoted by an index $i$ from zero for the substation node to $n$. A network is represented as a set of nodes referencing each other as their ancestor or their children, in a way of a linked list. The complete structure of a node is illustrated on figure (5.1). There are three different categories: (a) local variables related to values associated with the node, as its complex voltage, its power injection, etc. (b) some parameters related to the topology of the network, to operational constraints, etc. (c) variables that are local observations of its neighbours' variables.

This structure enables an easy implementation in a peer-to-peer architecture. In this architecture, each device contains its on structure and its own processor, which carries out the required local optimisation and exchanges messages with its neighbours on the network. In this setting, the devices do not need to divulge their objectives or constraints. They only need to support a simple protocol for interacting with their neighbours. The decentralized algorithm elaborated in section 4.2 ensures that each device has no information about the rest of the network, and only exchanges limited message with its immediate neighbours. Hence, whether the network contains ten or one million nodes, the structure of a node doesn't change and the computation associated with it remains the same.

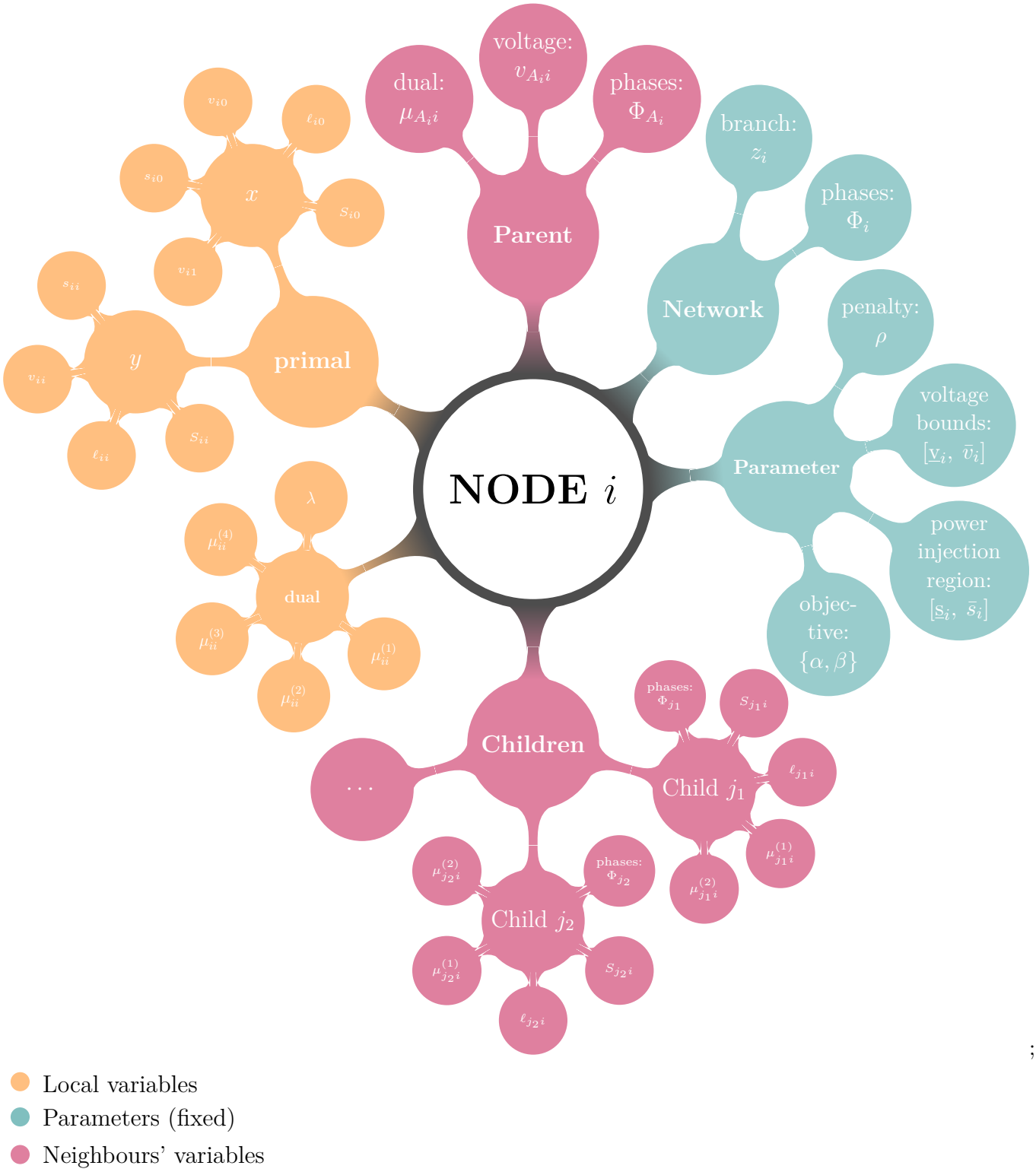
## 5.2 Initialization

A good starting point usually reduces the number of iterations for a given tolerance. Hereafter, I describe an initialisation procedure inspired from [PL15]. The procedure works on the nodal variables $V_i$ and $I_i$ for each node $i \in \mathcal{N}$, the variables $(S, l, v)$ are then deduced. The initialisation procedure find a feasible solution assuming zero impedance on the lines. In this case, the two major constraints of the OPF problem can be simplified and transformed into:

$$\begin{cases} \mathcal{P}_i(v_{A_i}) = v_i - (z_i S_i^* + S_i z_i^*) + z_i \ell_i z_i^* \\ s_i + \operatorname{diag}\left(\sum_{j \in C_i} \mathcal{P}_i\left(S_j - z_j l_j\right)\right) = \operatorname{diag}\left(S_i\right) \end{cases} \Longrightarrow \begin{cases} \mathcal{P}_i(v_{A_i}) = v_i & i \in \mathcal{N}_+ \\ s_i + \operatorname{diag}\left(\sum_{j \in C_i} \mathcal{P}_i\left(S_j\right)\right) = \operatorname{diag}\left(S_i\right) & i \in \mathcal{N} \end{cases}$$

Hence, each node has the same voltage values at the beginning, i.e.

$$V_i^{\phi_1} = V_0^{ref}, \;\; V_i^{\phi_2} = V_0^{ref} e^{\frac{2\pi}{3}}, \;\; V_i^{\phi_3} = V_0^{ref} e^{-i\frac{2\pi}{3}} \qquad \forall i \in \mathcal{N}$$




;

Figure 5.1: Structure of a node

For each node $i \in \mathcal{N}$, we initialise the power injection $s_i$ using any feasible point in the injection region $\mathcal{I}_i$. When a node $i \in \mathcal{N}$ doesn't have any children, the second constraint can be simplified even further :

$$s_i = \operatorname{diag}(S_i)$$

hence, by definition of the complex branch power, $I_i^\phi = \left(\frac{S_i^{\phi\phi}}{V_i^\phi}\right)^* = \left(\frac{s_i^\phi}{V_i^\phi}\right)^*$, for all $\phi \in \Phi_i$, and $i \in \mathcal{N}$, such that $C_i = \emptyset$. If a node $i \in \mathcal{N}$ has some children then, using the current balance, we obtain :

$$I_i^\phi = \left(\frac{s_i^\phi}{V_i^\phi}\right)^* + \sum_{j \in C_i} I_j^\phi$$

Thanks to this result, we can define an algorithm to initialise the current $I_i$ for each node $i \in \mathcal{N}$:

**Algorithm 2:** Initialisation of the current $I$

**input** : $(v, S, \ell)$ that satisfies the conditions in Lemma (1)

**output**: $(V, I)$

`initCurrent(`*arg1, arg2, arg3*`)`

$I_i^\phi \longleftarrow \left(\frac{s_i^\phi}{V_i^\phi}\right)^*$

$\mathcal{N}_{visit} \longleftarrow \{0\}$

**while** $\mathcal{N}_{visit} \neq \mathcal{N}$ **do**

find $i \to j$ such that $i \in \mathcal{N}_{visit}$ and $j \notin \mathcal{N}_{visit}$

compute

$$I_{ij} \leftarrow \frac{1}{\operatorname{Tr}(v_i)} S_{ij}^* V_i$$
$$V_j \leftarrow V_i - z_{ij} I_{ij}$$
$$\mathcal{N}_{visit} \leftarrow \mathcal{N}_{visit} \cup \{j\}$$

**end**

Now, we can to compute the quantity $v_{i0} = V_i V_i^*$, $\ell_{i0} = I_i I_i^*$, $S_{i0} = V_i I_i^*$, and $s_{i0} = s_i$, for $i \in \mathcal{N}$. Finally, it remains to set the variables $y$ at the same values :

$$\begin{cases} y_{ij} = x_{i0} & \forall j \in \mathcal{N}_i,\ i \in \mathcal{N} \\ x_{i1} = x_{i0} & \forall i \in \mathcal{N} \end{cases}$$

## 5.3 Stopping Criterion

Even if there is no general rule for the stoping criterion, a simple stopping criterion for the ADMM method is

$$\left\|r^k\right\|_2 \leq \epsilon^{\text{primal}} \qquad \left\|s^k\right\|_2 \leq \epsilon^{\text{dual}}$$

where $\epsilon^{\text{primal}} > 0$ and $\epsilon^{\text{dual}} > 0$ are the feasibility tolerances for the primal and dual residuals. Both quantities can be normalised according to the network size :

$$\epsilon^{\text{primal}} = \epsilon^{\text{dual}} = \epsilon^{\text{abs}} \sqrt{|\mathcal{N}|}$$

where $\epsilon^{\text{abs}} > 0$ is some absolute parameter. A good value of $\epsilon^{\text{abs}}$ in practice turns around $10^{-4}$, $10^{-5}$. In the context of this project, I took $\epsilon^{\text{abs}} = 10^{-5}$.

## 5.4 Dynamic Penalty Parameter

The ADMM method is based on the relaxation of constraints linking the variables $x$ and $y$. The violation of the primal feasibility is then penalised using a parameter $\rho$. Many examples in the literature (see e.g. [BPC+11]) show that the value of $\rho$ can have a dramatic effect on the rate of convergence in practice. A standard extension is to use different penalty parameters $\rho^k$ for each iteration. The objective is twofold: improve the convergence in practice, and making the algorithm less dependent on the initial choice of the penalty parameter. Such approach has been studied in the context of the method of multipliers, and it has been shown that a superlinear convergence might be achieved. In the case of ADMM, this approach makes the proof of the convergence more difficult, however it holds if we assume a fixed value of $\rho$ after a given number of iterations.

It's shown in [SYP15] that this standard approach often works well in practice:

$$\rho^{k+1} := \begin{cases} \alpha^{inc}\rho^k & \text{if } \left\|r^k\right\|_2 > \beta\left\|s^k\right\|_2 \\ \frac{\rho^k}{\alpha^{dec}} & \text{if } \left\|s^k\right\|_2 > \beta\left\|r^k\right\|_2 \\ \rho^k & \text{otherwise} \end{cases} \tag{5.1}$$

where $\alpha^{inc}$, $\alpha^{dec} > 1$, and $\beta > 1$ are parameters. The goal of this approach is to keep both primal and dual residual norms within a factor $\beta$ from each other while they both converge to zero. Indeed, recall that the primal and dual residuals in the standard form are given by :

$$r^{k+1} = Ax^{k+1} + Bz^{k+1} - c$$
$$s^{k+1} = \rho A^T B(z^{k+1} - z^k)$$

The ADMM method suggests that a large value of the penalty parameter involves a large penalty on the violation of primal feasibility and tends to produce small primal residuals. However the definition of the dual residual $s^{k+1}$ suggests that small values of the penalty parameter $\rho$ tend to reduce the dual residual but at the expense of the primal feasibility that may increase.

The scheme (5.1) increases the penalty parameter $\rho$ by a factor $\alpha^{inc}$ when the primal residual appears larger than the dual residual, and reduce $\rho$ by $\alpha_{dec}$ when the primal residual appears too small compared to the dual residual.

Notice that when varying the penalty parameter, some predefined parameters must be modified. Recall that the solution of the $y$-update implies a simple matrix-vector multiplication where the vector changes at each iteration whereas the matrix is fixed and can be cached at the beginning of the iterative process.

$$z^\star = \left(Q^{-1}A^T\left(AQ^{-1}A^T\right)^{-1}AQ^{-1} - Q^{-1}\right)c$$

Since $Q$ depends on $\rho$, The matrix $T$ is in fact a function of the penalty parameter $\rho$. Fortunately, we can extract the penalty parameter from the closed-form solution. By definition of the diagonal matrix

$Q$, let's define $\tilde{Q}$ such that $Q = \rho\tilde{Q}$, then

$$\begin{aligned} z^\star &= \left(Q^{-1}A^T\left(AQ^{-1}A^T\right)^{-1}AQ^{-1} - Q^{-1}\right)c \\ &= \left(\frac{1}{\rho}\tilde{Q}^{-1}A^T\left(A\frac{1}{\rho}\tilde{Q}^{-1}A^T\right)^{-1}A\frac{1}{\rho}\tilde{Q}^{-1} - \frac{1}{\rho}\tilde{Q}^{-1}\right)c \\ &= \frac{1}{\rho}\underbrace{\left(\tilde{Q}^{-1}A^T\left(A\tilde{Q}^{-1}A^T\right)^{-1}A\tilde{Q}^{-1} - \tilde{Q}^{-1}\right)}_{\tilde{T}}c \end{aligned}$$

Since the matrix $\tilde{T}$ doesn't change from one iteration to another, we can stored the matrix $\tilde{T}$ of each node into its local memory at the start of the algorithm.

## 5.5 Measure of the exactness of the solution

To quantify how close a solution of the SDP problem is to rank-1, we need to define a metric. Due to finite numerical precision, even if the solution of the relaxation is exact, the matrices (2.9) are only approximately rank-1. To quantify how close those matrix are to rank one, we can look at their ratios $|\lambda_2/\lambda_1|$, where $\lambda_1$ and $\lambda_2$ are their two largest eigenvalues, with $|\lambda_1| > |\lambda_2|$. The smaller those ratios are, the closer the matrices are to rank one. We only keep the largest ratio over all the matrices to quantify how close are solution is to rank one (i.e. exact solution).

## 5.6 Conclusion

This chapter briefly described the key points behind the implementation of the decentralised algorithm. The pertinence of those choices is discussed in the next chapter.

# 6 | Numerical Results

In this chapter, we report numerical results on different test networks. They are chosen to illustrate the effectiveness of the algorithm and a variety of the ideas developed in the previous sections. In the first part of this chapter, I apply the algorithm on a small made-up network in order to explore the behaviour of the algorithm and to verify the exactness of the algorithm. In the second part of this chapter, the algorithm is applied to real networks [Ker01a]. And we show that the SDP relaxation of the OPF problem is generally exact for real networks.

## 6.1 First Test

The network chosen to illustrate the effectiveness of the algorithm and its behaviour is depicted on figure 6.1 and a part of the datas associated with the network is available on table 6.1 (for the rest, see appendix A.4). The substation node (node 0) and node 1 have three phases, whereas nodes 2 and 3 have two and one phase, respectively. Table 6.1 describes the spot load for each node and the maximum complex power that can be produced. Since the substation node is responsible for drawing the power from the transmission network to the distribution network for power balance, there is no upper or lower bound associated with node 0. The voltage magnitude is allowed to deviate by a factor of 10% from the nominal value (in this case, 50 V). Finally the objective function consists in minimizing the total power loss over the network.

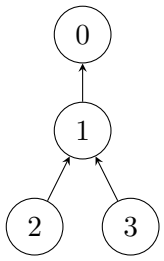



Figure 6.1: Test network

| | Node | **Active Power** (W) | | | **Reactive Power** (VAR) | | |
|---|---|---|---|---|---|---|---|
| Generation | | Ph1 | Ph2 | Ph3 | Ph1 | Ph2 | Ph3 |
| | 1 | 3 | 0 | 5 | 3 | 0 | 5 |
| | 2 | 3 | 3 | · | 0 | 0 | · |
| | 3 | · | · | 1 | · | · | 1 |
| Spot Load | | Ph1 | Ph2 | Ph3 | Ph1 | Ph2 | Ph3 |
| | 1 | 6.1 | 2 | 3 | 3.1 | 0.23 | 4.6 |
| | 2 | 3.45 | 1 | · | 0 | 3 | · |
| | 3 | · | · | 1.28 | · | · | 1 |

Table 6.1: Spot load and generation capacity for each node of the test network

**Centalized Solver**

The relaxation of the OPF problem has also been modelled using a popular modelling framework for convex optimization (CVX, see [GB14]). The optimization solver used, is based on a primal-dual interior point method, and works on the centralized formulation of the OPF problem. The objective was twofold : (a) verify the exactness of the solution obtained by the decentralized algorithm, (b) compare the computation time for solving the relaxation.

**Computation Time**

The ADMM method takes the form of a decomposition-coordination procedure, in which the solutions to small local subproblems are coordinated to find a solution to a large global problem. This method makes it possible to split our problem into $|\mathcal{N}|$ subproblems, which can be solved in parallel. In this context, we assumed that each node contains its own processor, which carries out the required optimization. However because of the coordination and the communications with their immediate neighbours, the nodes have to wait for the termination of the update of their neighbours before being able to perform the next iteration. The computation time is calculated as follow : each node performs its $x$-update and the maximum computation time is stored in $t_x^k$. In the same way, we store the maximum computation time for both the $y$-update and the dual update. Then the computation time at iteration $k$ is given by, $t^k = t_x^k + t_y^k + t_{dual}^k$ (see figure 6.2).

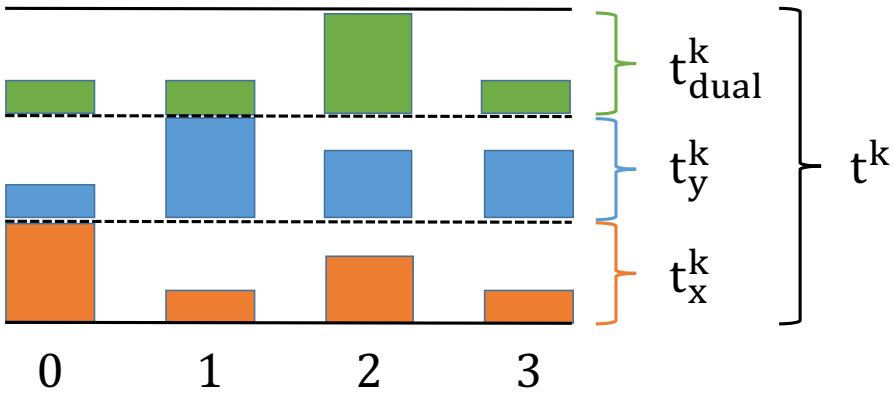



Figure 6.2: Computation time at iteration $k$, $t^k$

### 6.1.1 The Computation of the Solution

Recall that the optimality conditions for the ADMM method sum up to the convergence of the primal and dual residuals $r^k$ and $s^k$, respectively. The natural stopping criteria chosen in this project is that both primal and dual residuals are below $10^{-5}\sqrt{|\mathcal{N}|}$, where $|\mathcal{N}|$ is the number of nodes in the network. Figure 6.3 shows the the evolution of $r^k/\sqrt{|\mathcal{N}|}$ and $s^k/\sqrt{|\mathcal{N}|}$ versus iteration $k$. The optimal value of the objective function $f^\star$ was computed a priori using the centralized solver with the highest tolerance level (*cvx_precision high*, see [GB14]). The total number of iterations is 464, for a computation time of 0.658s, and a average computation time per iteration of 1.4181 $10^{-3}$s. The speed per iteration is mainly due to the closed-form solutions derived for all sub-problems. If instead, we were using an iterative solver (CVX) for each of the subproblems, the computation time would have been multiplied by at least one hundred. This highlights the necessity to derive closed-form solution for each of the

subproblems.

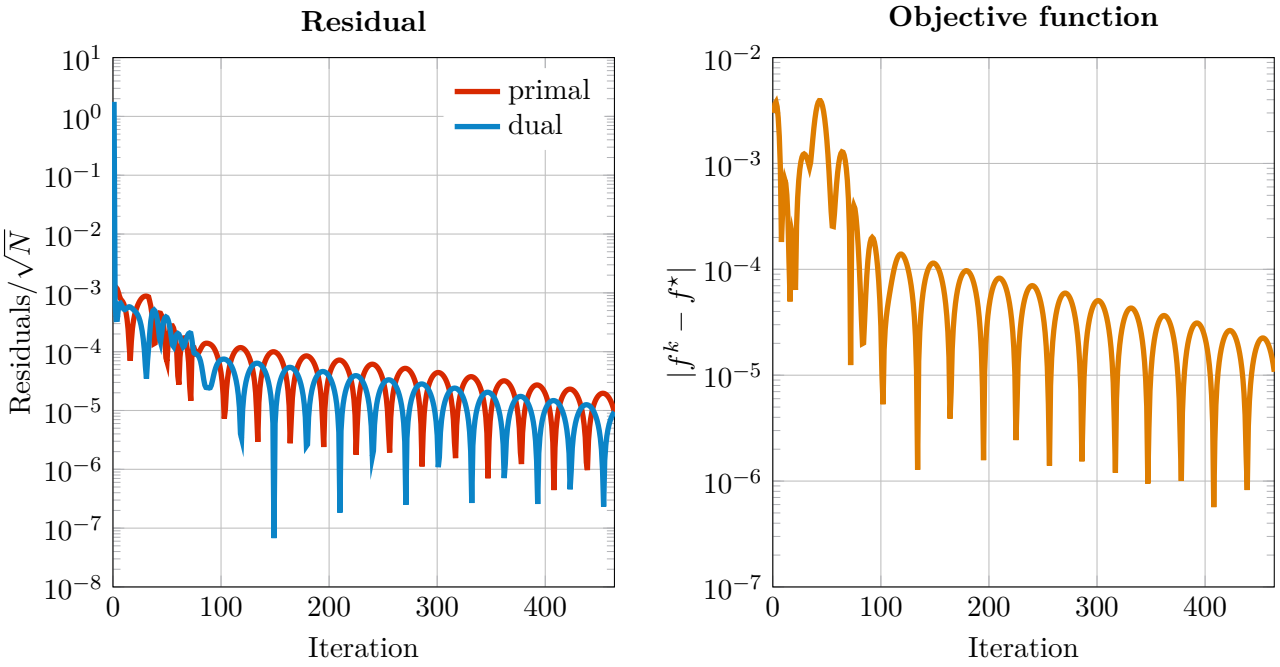



Figure 6.3: On the left, we see the evolution of both primal and dual residuals, $r^k/\sqrt{|\mathcal{N}|}$ and $s^k/\sqrt{|\mathcal{N}|}$. On the right, we can see the absolute error between the value of the objective function at iterate $k$, $f^k$, and the optimal objective function, $f^\star$.

Interestingly, the computation time of the solver working on the centralized formulation is 0.98s for a total of 28 iterations. This shows that ADMM can be competitive with the best known methods (interior point methods) even for a small instance of the problem, as long as the computation is performed in parallel and not serially.

Figure 6.3 shows the general behaviour of the evolution of the residuals. There are several important aspects to note. Firstly, we observe a linear rate of convergence as the theory predicts. Secondly, the pattern describes by the evolution of the residuals shows clear bumps of both the primal and the dual residuals. But even more interestingly, when one residual is a the top of its bump, the other one is at the bottom of its curve. Indeed, the primal residual is fighting for the equality between the $x$ and the $y$ variables, whereas the dual update is trying to stabilise the iterates $y^k$. When the value of the primal residual decreases, it bothers the dual residual because he has to change the $y$ variable, and quite logically it produces a increase of the dual residual. They both play this game in an alternating way (remember what the A stands for in ADMM) until a given tolerance level is reached. This is similar to a negotiation in which both parties discuss, debate, confront their point of view and step by step find an agreement.

### 6.1.2 The Solution

Recall that the relaxation of the OPF problem is exact if the matrices 2.9 are rank-1. To quantify how close those matrices are to rank one, we can look at their ratios $|\lambda_2/\lambda_1|$, where $\lambda_1$ and $\lambda_2$ are their two largest eigenvalues, with $|\lambda_1| > |\lambda_2|$. The smaller those ratios are, the closer the matrices are to rank one. We only keep the largest ratio over all the matrices to quantify how close our solution is to rank

one (i.e. exact solution). With a ratio of $4.6526\,10^{-13}$, we can take for granted that the relaxation of the OPF problem for our test network is exact.

Since we obtained a rank-1 solution, the recovering algorithm described in section 2.2 can be applied for each node $i \in \mathcal{N}^+$. We extract from the voltage magnitude $v_i$, the complex branch power $S_i$ and the current magnitude $l_i$, the complex voltage $V_i$ and the complex current $I_i$ for each node $i \in \mathcal{N}^+$. The complete description of the solution is available in the tables (A.1)-(A.3) in appendix A.5.

The interpretation of the solution is vey simple : If a node has the capacity to fulfil its own demand using local generators, it's more beneficial to produce locally since it doesn't involve any power loss. However if the local capacity isn't sufficient to satisfy the demand, external generators can be used. Those generators are chosen in order for the path from the supplier to the consumer to be the lowest in energy consumption, while of course satisfying the power flow equations.

## 6.2 How to Tune the Algorithm

### 6.2.1 Initialization

The goal of the initialisation procedure described in section 5.2 is to find a feasible solution assuming zero impedance on the lines. By using such starting point, we hope to reduce the number of iterations required to achieve a given tolerance level. Figure 6.4 shows the evolution of the residuals. The same pattern is observed, except that at the beginning both primal and dual residuals are much smaller. Nevertheless, the number of iterations is quite identical : 462, instead of 464. What we observe is that the gain from using the initialisation procedure is catch up by the algorithm using a basic starting point, in two or three iterations. Note that this behaviour isn't specific to this particular instance of the problem. In average, the initialisation procedure reduces the number of iterations by less than 5%. therefore the initialisation procedure isn't vital but it can't harm since the starting point is easy to compute.

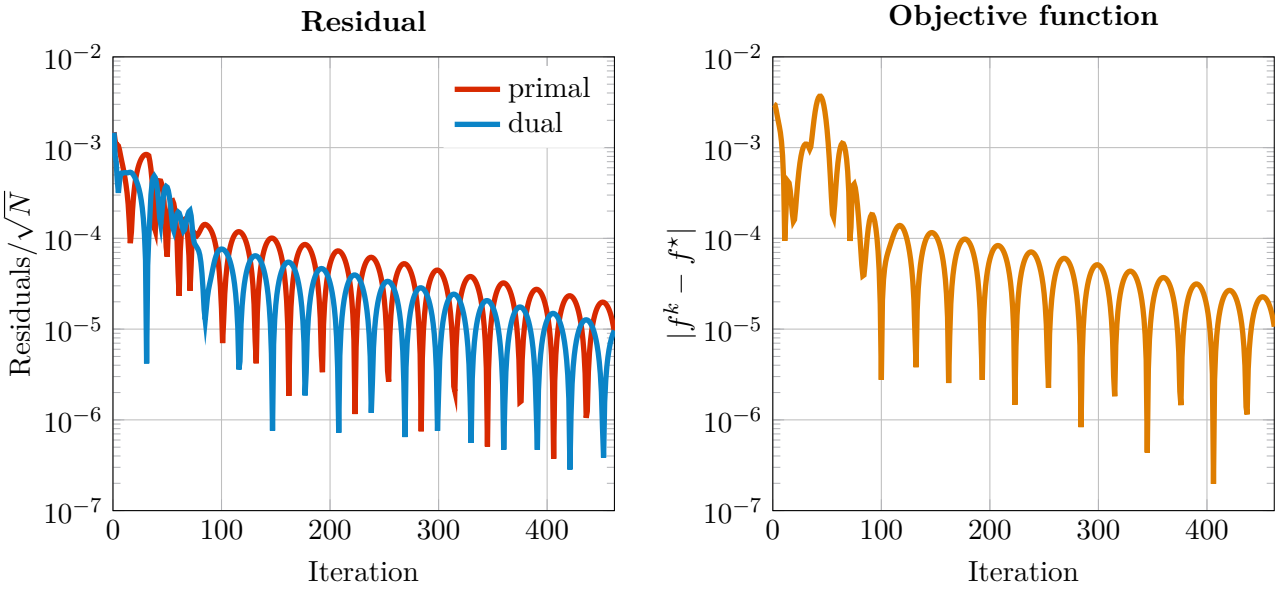



Figure 6.4: The evolution of both primal and dual residuals, and the absolute error between the value of the objective function $f^k$, and the optimal objective function, $f^\star$, when we use the initialisation procedure.

### 6.2.2 The Penalty Parameter, aka the selfishness parameter

Many examples in the literature (see e.g. [BPC+11]) show that the value of $\rho$ can have a dramatic effect on the rate of convergence in practice. To illustrate the influence of the penalty parameter on the convergence of the method in practice, I choose a more complex network composed of 30 nodes. Figure 6.5 shows the evolution of the residuals for this network and for a penalty parameter of 1.

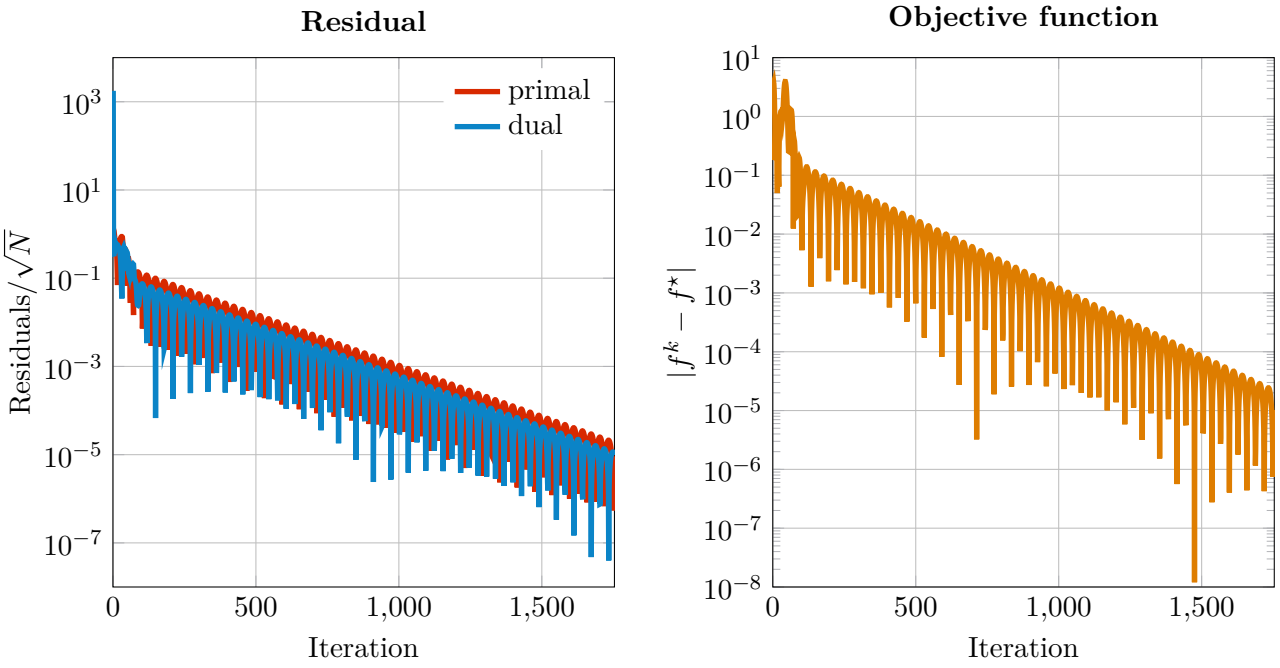



Figure 6.5: number of iterations : 1754, total CPU time : 1.98s

On figure 6.6, we see the evolution of the residuals when the penalty parameter is set to 100, instead of 1 in the previous example. Since the violation of the primal feasibility is penalised using this parameter $\rho$, we observe low values of the primal residual but at the expense of the dual residual. Due to the high value of the penalty parameter, the primal is totally self-centred and try to have the smallest residual as possible without considering the dual. Actually he persists quite a long time on this path ($\sim$ 3500 iterations). Nevertheless, at some point, he understands that optimality cannot be achieved with this logic. After more than 4000 iterations, the primal residual changes its strategy and accepts to take a step backward in order to jointly decrease with the dual residual.

This observation Highlights the importance of the penalty parameter on the rate of convergence in practice. To make the algorithm less dependent on the initial choice of the penalty parameter, we described in section 5.4 a dynamic scheme allowing us to update the value of the penalty parameter according to the difference between the primal and the dual residuals. If the primal and the dual residuals are too far away from each other then the penalty parameter is updated in order to force them to stick together. Figure 6.7 shows the result of this scheme, with an initial penalty parameter set to 100. The value of the penalty parameter quickly decreases from 100 to end up with values close to one. When it decreases, it produces an increase of the primal residual because the primal feasibility becomes less penalised. Those decreases of the penalty parameter correspond to the abrupt increases of the primal residual on figure 6.7. Notice that the number of iterations (2397) is bigger than if we had used a fixed, but more suitable penalty parameter, in this case $\rho = 1$. In the case of a good initial penalty parameter, the dynamic update of $\rho$ will not play any role since the two residuals will stay

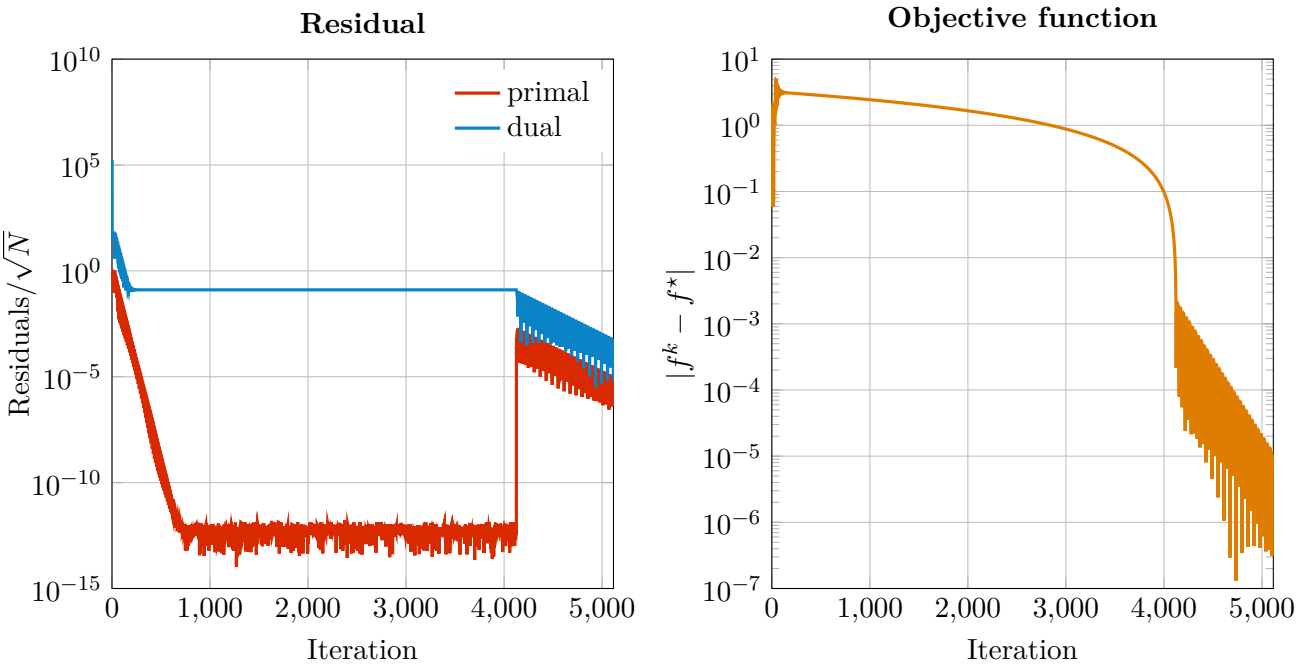



Figure 6.6: number of iterations : 5116, total CPU time : 3.92s

close from each other. Therefore, there is no downside risk in using the dynamic update of the penalty parameter. If $\rho$ is correctly chosen, it doesn't change anything, if not, then the number of iterations can be drastically reduced. In brief, one should always use this procedure.

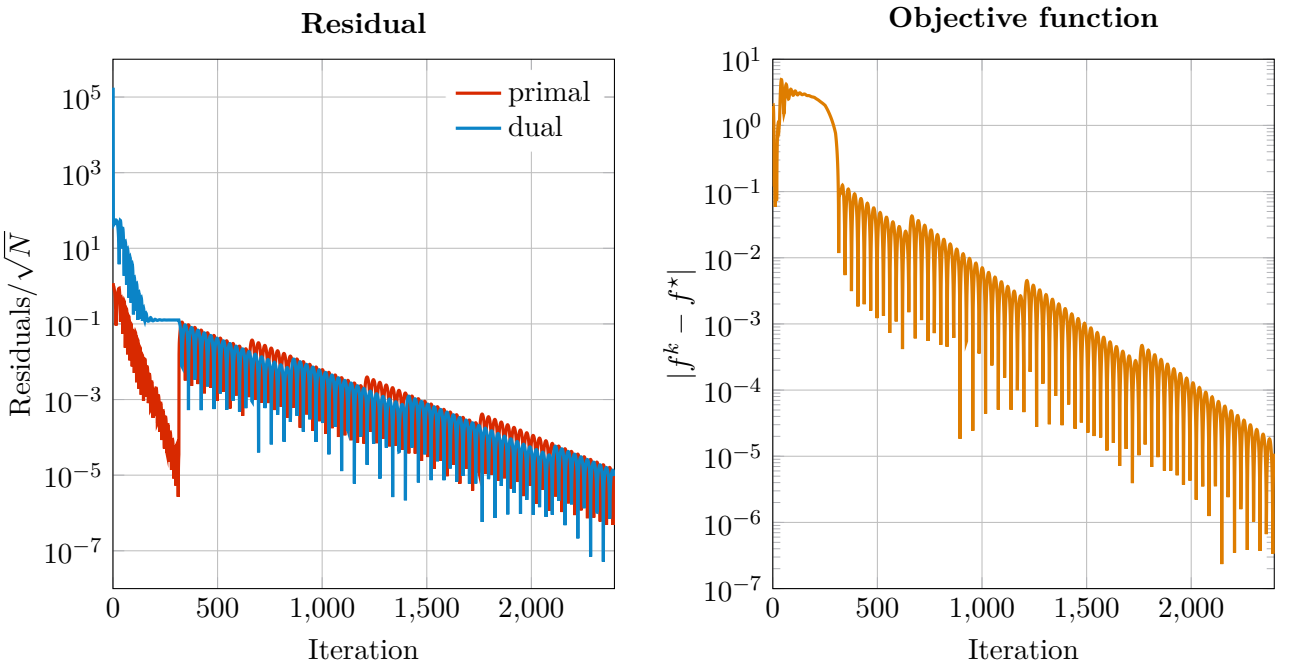



Figure 6.7: number of iterations : 2397, total CPU time : 2.17s

## 6.3 Impact of Network Topology

In this section, I would like to determine the impact of the size of the network on the number of iterations required to achieve a tolerance level of $10^{-5}\sqrt{|\mathcal{N}|}$, where $|\mathcal{N}|$ is the size of the network. In particular, we simulate the algorithm on two extreme cases : (a) a line network (see figure 6.8) and a fat tree network (see figure 6.9). The nodes of those networks have three phases, the spot load and the generation capacity for each node are randomly determined. Figure 6.10 shows the average number of iterations per network size (10 trials per network size).

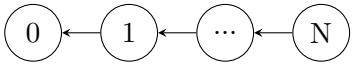



Figure 6.8: Line network

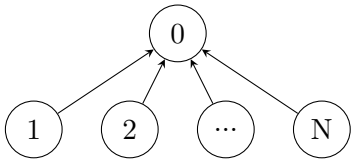



Figure 6.9: Fat tree network

There are several observations to make. Firstly, we observe that for line networks, the number of iterations increases especially as the network size increases. For fat tree networks, the trend is still present but less obvious compared to line networks. We can conclude that the size of the network isn't necessarily the most important factor influencing the number of iterations. Indeed, even if the fat tree network has the same size than the line network, the number of iterations is much smaller. The diameter of the network (longest shortest path between any two nodes in the network) seems a much more relevant factor when considering the number of iterations. Let's suppose that the number of iterations takes the linear form $a \cdot size + b \cdot diameter$. The parameters $a = 14.508$, $b = 95.645$ minimize the least square error for the data on figure 6.10. This shows that the diameter of the network is a much more important factor than the network size to understand the variation of the number of iterations from one network to another. This can be easily explained : In fact, the ADMM method relies on the communication between the nodes of the network in order to find an agreement, which corresponds to the global optimum. But when the network has a big diameter, the information takes much more time to travel from one node to the others. hence the time to reach a consensus is much longer.

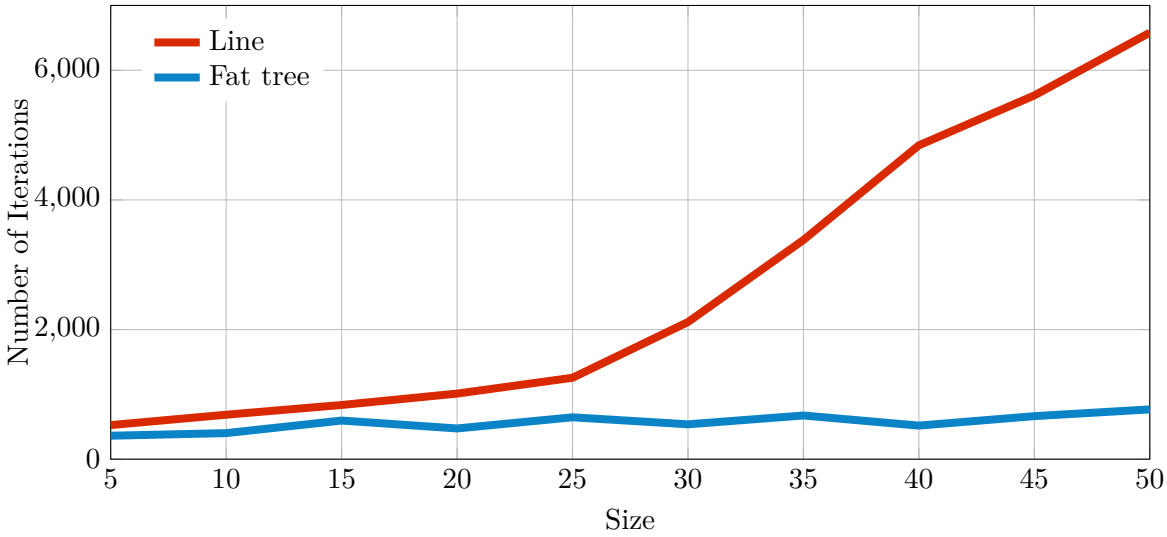



Figure 6.10: Evolution of the number of iterations for different network sizes

## 6.4 Case Study : IEEE test feeders

Besides theoretical results, we would like to test the algorithm on real networks. The systems described in [Ker01a], are real networks and were designed to evaluate and benchmark algorithms in solving unbalanced three-phase radial systems[1]. The four networks used contain 13, 34, 37, and 123 buses respectively. The IEEE13 test feeder operates at a nominal voltage of 4.16kV. The IEEE34 feeder is located in Arizona and operates at a nominal voltage of 24.9kV. the IEEE37 feeder is located in California and operates at a nominal voltage of 4.8kV. Finally, the IEEE123 has also a nominal voltage of 4.8kVand is shown on figure 6.11. In order to apply our model to those networks some simplifications have been made :

- Transformers are modelled as lines with appropriate impedances.
- The voltage of the regulators is assumed to be fixed at the nominal voltage.
- Circuit switches are considered open or short lines according to their status in the dataset.
- Distributed load on a line is split and considered as two identical loads at each end of the line.

The substation node is modelled as a fixed voltage bus with an infinite power injection for power balance. Each line in the network is characterised by a length and an impedance. Each bus is modelled as a load bus whose voltage magnitude at each phase can vary within [0.95 1.05] (per unit) and the spot loads are also specified (active and reactive power). There are no controllable devices in those networks, hence the OPF problem degenerates to a power flow problem, which is easy to solve. In order to demonstrate the effectiveness of the algorithm, we replace all the capacitors with inverters, whose reactive power injection ranges form 0 to the maximum ratings specified by the original capacitors. The objective function consists in minimizing the total power loss over the whole network.

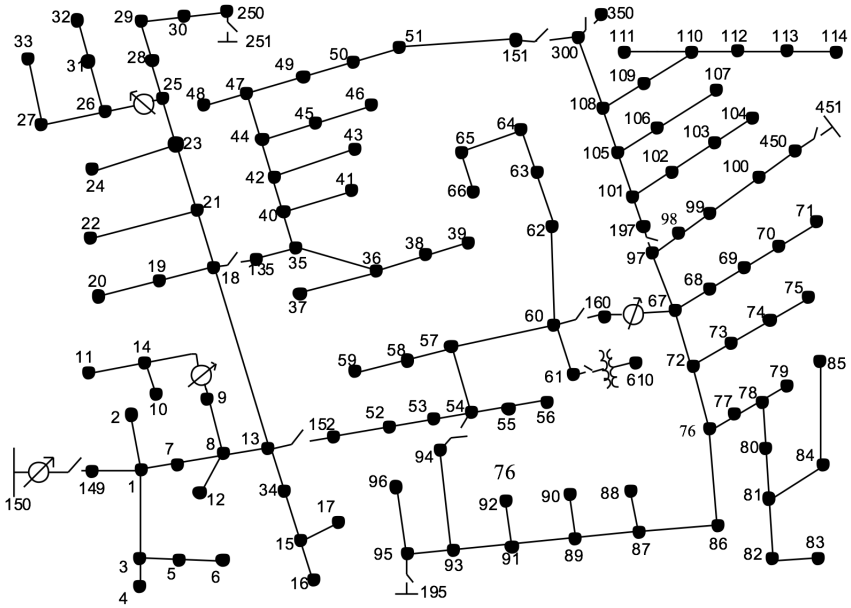



Figure 6.11: IEEE123 test feeder (California)

[1]source : `https://ewh.ieee.org/soc/pes/dsacom/testfeeders/`

Table 6.2 summarises the results for the four networks. They have been obtained using the decentralized solver with the intialization procedure and the dynamic update of the penalty parameter (initial value of the penalty parameter : 3). We observe that the number of iterations is consistent with the previous conclusion : the number of iterations is mainly driven by the diameter of the network. Notice that three out of four of those networks get a rank-1 solution (the value of the ratio entry is closed to zero). Meaning that the OPF relaxation is exact for those networks. These results are confirmed by other studies showing the exactness of SDP relaxations on real networks, not just for the IEEE test feeders [PL15], [GL14]. As far as the CPU time is concerned, we see that the proposed algorithm always beats the CVX solver since we assumed a parallel implementation of the algorithm (The average CPU time per iteration turns around $10^{-3}s$).

| Network | Diameter | #Iter | CPU time [s] | CVX [s] | value | ratio |
|---|---|---|---|---|---|---|
| IEEE13 | 6 | 762 | 0.657 | 0.95 | 187.78 | $2.3\ 10^{-11}$ |
| IEEE34 | 20 | 2512 | 2.121 | 4.78 | $-123.01$ | 0.978 |
| IEEE37 | 16 | 2302 | 2.001 | 4.71 | 332.59 | $8.2\ 10^{-9}$ |
| IEEE123 | 30 | 4550 | 3.664 | 18.34 | 27.67 | $3.6\ 10^{-12}$ |

Table 6.2: Results of the simulation

## 6.5 Conclusion

In this chapter, we gave some numerical results proving the effectiveness of the proposed algorithm. In particular, we showed that ADMM can be competitive with the best known methods even for a small instance of the problem. When considering the number of iterations, the diameter of the network seems to be a much more important factor than the size of the network. Finally, and quite fortunately, the SDP relaxation of the OPF problem is apparently exact for three out of four the networks on which the algorithm was applied.

# Extensions

In this section, some directions for future research are discussed. Especially, we show that battery storages can be included in our model without rethinking the entire algorithm.

## Integration of Battery Energy Storages

Two of the most popular forms of renewable generation, wind and photovoltaic face large fluctuations in their generation profile. This variation means that peak generation and peak customer demand are often greatly divergent. Battery energy storage can help to buffer wind and PV generation, by capturing a portion of the energy produced during light load and exporting it back onto the network as required. Even if the modelisation of battery energy storages is beyond the scope of this master thesis, it's interesting to know if such devices can be integrate in our model without rethinking the entire algorithm. In the sequel, a modification of the OPF formulation is proposed in order to integrate battery energy storages.

Battery storages force the problem to consider a new dimension, the time. Indeed, there can be several variables associated with a battery, for example the battery level and the temperature. Those state variables add a temporal dimension to the problem, and link the problems in time. Suppose, the spot load and the spot price are available for each time $t \in [1, 2, \dots T]$, where $[1, 2, \dots T]$ is the time frame considered. For each bus $i \in \mathcal{N}^+$, denote $q_i^t$ the battery level at time $t$. Then the power balance constraints have to be modified to take into account the charge or the discharge of the batteries:

$$s_i^t + \text{diag}\left(\sum_{j \in C_i} \mathcal{P}_i \left(S_j^t - z_j l_j^t\right)\right) = \text{diag}\left(S_i^t\right) + \xi\left(q_i^{t+1} - q_i^t\right) \qquad \forall i \in \mathcal{N} \text{ and } t \in [1, \dots T-1]$$

where $\xi$ is a constant output factor, and the variable $q_i^1$ is given. Note that Ohm's law doesn't need any modifications. There are also two technical constraints to consider. The first constraint is a ramp rate constraint on the battery level:

$$R_i^- \leq q_i^{t+1} - q_i^t \leq R_i^+ \qquad \forall i \in \mathcal{N} \text{ and } t \in [1 \dots T-1]$$

Meaning that the charge of the discharge of the battery between $t$ and $t+1$ is limited by a ramp rate. And the second constraint forces the battery level to be maintained within a given range $[Q_i^-, Q_i^+]$:

$$Q_i^- \leq q_i^t \leq Q_i^+ \qquad \forall i \in \mathcal{N} \text{ and } t \in [1 \dots T]$$

If there is no battery at bus $i \in \mathcal{N}$, then $Q_i^-, Q_i^+ = 0$. Besides satisfying operational constraints, the OPF problem consists in minimizing an objective function, $f$. Hereafter, we assume there exists for

each node $i \in \mathcal{N}$, a real-valued function $f_i^t$ (convex) defined on $\mathbb{R}$ which represents the local objective of node $i$ at time $t$. Moreover suppose that the value of the objective function at time $t$ depends exclusively on the net complex power injection $s^t$ and the battery level at time $t$ and $t+1$. Indeed, typical objective functions include cost generation or power loss which depend exclusively on the power injection. But it can also include a cost for the charge and for the discharge of the battery. Then the total objective function is given by :

$$f^t\left(s^t, q^t, q^{t+1}\right) = \sum_{i \in \mathcal{N}} f_i^t\left(s_i^t, q_i^t, q_i^{t+1}\right)$$

The relaxation of the optimal power flow problem can be reformulated as :

$$\begin{aligned}
\min \quad & \sum_{t=0}^{T} \sum_{i \in \mathcal{N}} f_i^t\left(s_i^t, q_i^t, q_i^{t+1}\right) && && (6.1a)\\
\text{over} \quad & v, s, S, \ell, q \\
\text{s.t.} \quad & \mathcal{P}_i(v_{A_i}^t) = v_i^t - \left(z_i {S_i^t}^* + S_i z_i^*\right) + z_i \ell_i^t z_i^* && i \in \mathcal{N}^+,\ t \in [1 \dots T] && (6.1b)\\
& s_i^t + \operatorname{diag}\left(\sum_{j \in C_i} \mathcal{P}_i\left(S_j^t - z_j l_j^t\right)\right) = \operatorname{diag}\left(S_i^t\right) + \xi\left(q_i^{t+1} - q_i^t\right) && i \in \mathcal{N},\ t \in [1 \dots T] && (6.1c)\\
& R_i^- \le q_i^{t+1} - q_i^t \le R_i^+ && i \in \mathcal{N},\ t \in [1 \dots T-1] && (6.1d)\\
& Q_i^- \le q_i^t \le Q_i^+ && i \in \mathcal{N},\ t \in [1 \dots T] && (6.1e)\\
& \underline{v}_i \le v_i^t \le \overline{v}_i && i \in \mathcal{N},\ t \in [1 \dots T] && (6.1f)\\
& s_i^t \in \mathcal{I}_i^t && i \in \mathcal{N},\ t \in [1 \dots T] && (6.1g)\\
& \begin{pmatrix} v_i^t & S_i^t \\ {S_i^t}^* & \ell_i^t \end{pmatrix} \succeq 0 && i \in \mathcal{N},\ t \in [1 \dots T] && (6.1h)
\end{aligned}$$

We can see that the number of variables has been multiplied by $T$. However, our decentralized algorithm can be applied with just minor changes. Note that the variable $q_i$ doesn't need to be shared with neighbours's buses. Therefore, we only need to add the variables $q_{i0}^t$ and $q_{ii}^t$, in the $x$ variables and the $y$ variables respectively, for each time $t$. The variables $x_{i0}$ and $y_{ii}$ change into:

$$\begin{cases}
x_{i0} & := & \left\{S_{i0}^t,\ \ell_{i0}^t,\ v_{i0}^t,\ s_{i0}^t,\ q_{i0}^t \qquad \forall t \in [1 \dots T]\right\} \\
y_{ii} & := & \left\{S_{ii}^t,\ \ell_{ii}^t,\ v_{ii}^t,\ s_{ii}^t,\ q_{ii}^t \qquad \forall t \in [1 \dots T]\right\}
\end{cases}$$

and we have to add the consensus constraint :

$$q_{i0}^t = q_{ii}^t \qquad i \in \mathcal{N},\ t \in [1 \dots T]$$

Let's discuss in more details the modifications to operate in the $x$-update and the $y$-update in order to apply our algoithm.

**$x$-update** Since the variables $S_{i0}$, $l_{i0}$ and $v_{i0}$ have no time-dependencies, the update of the variables $S_{i0}$, $l_{i0}$ and $v_{i0}$ doesn't change at all and can be split into $T$ sub-problems for each bus $i \in \mathcal{N}$. This

gives us the following $T$ sub-problems.

$$\begin{aligned} \min \quad & \frac{\rho}{2}\left(|C_i|+2\right)\left\|\begin{pmatrix} v_{i0}^t & S_{i0}^t \\ S_{i0}^{t\,*} & l_{i0}^t \end{pmatrix} - \begin{pmatrix} \hat{v}_i^t & \hat{S}_i^t \\ \hat{S}_i^{t\,*} & \hat{\ell}_i^t \end{pmatrix}\right\|_2^2 \\ \text{over} \quad & S_{i0}^t, \ell_{i0}^t, v_{i0}^t \\ \text{s.t.} \quad & \begin{pmatrix} v_{i0}^t & S_{i0}^t \\ S_{i0}^{t\,*} & l_{i0}^t \end{pmatrix} \in \mathbb{S}_+ \end{aligned}$$

where $\hat{v}_i^t$, $\hat{S}_i^t$ and $\hat{\ell}_i^t$ are some fixed parameters. This optimization problem can be solved using eigen-decomposition of a $6 \times 6$ matrix. On the opposite, due to the time-dependencies associated with the battery levels, the second part of the $x$-update can't be split into $T$ sub-problems for each bus $i$ and the following problem must be solved:

$$\begin{aligned} \min \quad & \sum_{t=1}^{T} f_i^t\left(s_{i0}^t, q_{i0}^t, q_{i0}^{t+1}\right) + \frac{\rho}{2}\left\|s_{i0}^t - \hat{s}_i^t\right\|_2^2 + \frac{\rho}{2}\left\|q_{i0}^t - \hat{q}_i^t\right\|_2^2 & \\ \text{over} \quad & s_{i0}^t,\ q_{i0}^t & \forall t \in [1 \dots T] \\ \text{s.t.} \quad & R_i^- \le q_{i0}^{t+1} - q_{i0}^t \le R_i^+ & t \in [1 \dots T-1] \\ & Q_i^- \le q_{i0}^t \le Q_i^+ & t \in [1 \dots T] \\ & s_{i0}^t \in I_i^t & \forall t \in [1 \dots T] \end{aligned}$$

Unfortunately, a closed-form solution is generally impossible to derive for such problem. Although it could be possible to solve it efficiently using an industrial solver.

**$y$-update** The optimisation problem associated with the $y$-update is very similar except that the number of variables has drastically increased. Indeed, since the problem cannot be spilt into $T$ sub-problems because of the time-depencies, all the variables $y_{ii}^t$, $y_{ij}^t\ \forall j \in C_i$ and $y_{A_i i}^t$ are part of the same optimisation problem. Fortunately, this optimization problem can still be reformulated under the form :

$$\begin{aligned} \min_z \quad & \frac{1}{2} z^T Q z + c^T z \\ \text{s.t.} \quad & Az = 0 \end{aligned}$$

Where $z$ stacks the real and the imaginary parts of the variables $\left\{y_{ji}^t,\ \forall j \in N_i,\ \forall t \in [1 \dots N]\right\}$, and where the constraint $Az = 0$ includes the power balance constraints and the Ohm's Law. This problem enjoys a closed-form solution, given by :

$$z = \underbrace{\left(Q^{-1}A^T\left(AQ^{-1}A^T\right)^{-1}AQ^{-1} - Q^{-1}\right)}_{M} c$$

Note that even if the number of variables has been multiplied by a factor $T$, the matrix $M$ doesn't change from one iteration to another and so can be formed at the beginning of the iterative process. This saves us the necessity to recompute at each iteration the matrix $M$. In the end, the resolution of the $y$-update comes down to a simple multiplication between a square matrix and a vector.

**Conclusion** In brief, the proposed algorithm can easily integrate battery storages at the expense of a multiplication of the number of variables per node by the size of the time frame. Moreover, whether the SDP relaxation of the OPF problem can be solved efficiently depends on the existence of a closed-form solution or at least an efficient way to compute the $x$-update, which depends on the local objective functions $f_i$.

## Local Stopping Criteria

The stopping criterion and the algorithm proposed to update the penalty parameter $\rho$ require a global coordination, due to the computation of the global primal and dual residuals at each iteration. Theoretically speaking, those residuals could be computed in a decentralised way, using for example gossip algorithms [Sha09]. However such approach would requires many rounds of gossip between each iteration, and so a significant increase of the runtime. This approach has therefore not been implemented in this project. However, it could be interesting to investigate a method by which the stopping criterion could be independently chosen by individual devices based only on local information such as the primal and dual residuals of the node and its neighbours. In case of multistage programming, another approach is to run the algorithm continuously without any stopping criteria. The devices would iterate indefinitely until a given time, at which point they shift their moving horizon forward one time step and rerun the algorithm for this new time frame.

## Fast Alternating Direction Method of Multipliers

Compared to second-order methods that are able to achieve high accuracy via expensive iterations, ADMM relies on low-complex iterations and can achieve a modest accuracy in tens of iterations. While ADMM result in simple algorithms, they often perform poorly when high accuracy is required. Inspired by Nesterovs scheme for accelerating gradient methods [Nes], great efforts have been devoted to accelerating ADMM and attaining high accuracy in a reasonable number of iterations [GOSB14]. The proposed method is simply ADMM with a predictor-corrector type acceleration step. This predictor-corrector step is unfortunately stable only in the special case where the objective function is strongly convex. The accelerated version can still be applied to weakly convex problems, but the stability must be enforced using a restart rule. This restart rule relies on a combination of the primal and the dual residual, which makes the method easy to implement. An interesting research would be to investigate the potential application of this fast version of the ADMM method in order to reduce the rate of convergence in practice.

# General Conclusion

In the first part of this master thesis, we gave a short review of the recent advances in convexification methods for solving the optimal power flow problem for multiphase unbalanced distribution networks. For the last five years, those methods have been intensively studied and proved to be very promising, since they can be computed very efficiently. However, there are still theoretical issues regarding the exactness of such relaxations for radial networks as well as for general mesh networks.

After having formulated a convex relaxation of the optimal power flow problem, we developed a distributed algorithm based on the alternating direction method of multiplier (ADMM). This method has experienced a revival in recent years due to its applicability to optimization problems arising from "big data" and image processing applications, and its relative ease with which it may be implemented in parallel. The alternating direction method of multipliers takes the form of a decomposition-coordination procedure, in which the solutions to small local sub-problems are coordinated to find a solution to a large global problem. In this context, We presented a fully decentralized method based on message passing between buses, relying solely on peer to peer communication between buses that exchange energy. The method is completely decentralized, and needs no global coordination other than synchronizing iterations. At each iteration, every device passes simple messages to its network neighbours and then solves a local optimization problem that minimizes the sum of its own objective function and a simple regularization term that only depends on the messages it received from its network neighbours in the previous iteration. Moreover, the problem structure has been exploited in such a way that the subproblems are reduced to either a closed-form solution, or an eigen-decomposition of a hermitian matrix, which significantly reduce the convergence time.

We then demonstrated, in the third part, the scalability of the distributed algorithm proposed by testing it on some made-up networks as well as real multiphase distribution networks (IEEE test feeders). To show the efficiency of the proposed algorithm, we also compared the computation time of solving the decentralized algorithm with a popular optimization solver (CVX, see [GB14]), based on a primal-dual interior point method. Subsequently, we run the algorithm on networks of different topology to understand the factors that affect the convergence rate. We found out that the diameter of the network seems to be the main factor driving the number of iterations, more than the size of the distribution network. Preliminary simulations showed that the proposed convex relaxation of the optimal power flow problem is exact for three out four of the IEEE test distribution systems.

Finally, we concluded on some possible extensions for future considerations.

# Bibliography


[BLTH14] Subhonmesh Bose, Steven H. Low, Thanchanok Teeraratkul, and Babak Hassibi. Equivalent relaxations of optimal power flow. *CoRR*, abs/1401.1876, 2014.

[BPC+11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[Car62] J. Carpentier. Contribution to the economic dispatch problem. *Bulletin de la societe Francoise des Electriciens*, 3(8):431–447, 1962.

[CB90] H. D. Chiang and M. E. Baran. On the existence and uniqueness of load flow solution for radial distribution power networks. *IEEE Transactions on Circuits and Systems*, 37(3):410–416, Mar 1990.

[EY12] Jonathan Eckstein and W Yao. Augmented lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results. *RUTCOR Research Reports*, 32, 2012.

[FL12] Masoud Farivar and Steven H. Low. Branch flow model: Relaxations and convexification. *CoRR*, abs/1204.4865, 2012.

[GB14] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[GL14] L. Gan and S. H. Low. Convex Relaxations and Linear Approximation for Optimal Power Flow in Multiphase Radial Networks. June 2014.

[GLTL12] L. Gan, N. Li, U. Topcu, and S. H. Low. Exact Convex Relaxation of Optimal Power Flow in Tree Networks. August 2012.

[GM76] Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17 – 40, 1976.

[GOSB14] Tom Goldstein, Brendan O'Donoghue, Simon Setzer, and Richard Baraniuk. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences*, 7(3):1588–1623, 2014.

[GR75] Marroco A. Glowinski R. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *ESAIM: Mathematical Modelling and Numerical Analysis*, 9(R2):41–76, 1975.

[Ker01a] W. H. Kersting. Radial distribution test feeders. In *Power Engineering Society Winter Meeting, 2001. IEEE*, volume 2, pages 908–912 vol.2, 2001.

[Ker01b] W.H. Kersting. *Distribution System Modeling and Analysis*. The electric power engineering series. CRC Press, 2001.

[Low13] S. H. Low. Convex relaxation of optimal power flow: A tutorial. In *Bulk Power System Dynamics and Control - IX Optimization, Security and Control of the Emerging Power Grid (IREP), 2013 IREP Symposium*, pages 1–15, Aug 2013.

[LZT12] A. Y. S. Lam, B. Zhang, and D. N. Tse. Distributed algorithms for optimal power flow problem. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 430–437, Dec 2012.

[Nes] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $o(1/k^2)$.

[NLR+15] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. I. Jordan. A General Analysis of the Convergence of ADMM. February 2015.

[PL15] Q. Peng and S. Low. Distributed Algorithm for Optimal Power Flow on Unbalanced Multiphase Distribution Networks. December 2015.

[Sha09] Devavrat Shah. Gossip algorithms. *Found. Trends Netw.*, 3(1):1–125, January 2009.

[SLC12] B. Subhonmesh, S. H. Low, and K. M. Chandy. Equivalence of branch flow and bus injection models. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pages 1893–1899, Oct 2012.

[SYP15] Changkyu Song, Sejong Yoon, and Vladimir Pavlovic. Fast ADMM algorithm for distributed optimization with adaptive penalty. *CoRR*, abs/1506.08928, 2015.

[ZT13] B. Zhang and D. Tse. Geometry of injection regions of power networks. *IEEE Transactions on Power Systems*, 28(2):788–797, May 2013.

# A | Developments

## A.1 Proof of Theorem 2

*Proof.* Let $\Lambda := \text{diag}(\lambda_i,\ 1 \leq i \leq n)$, denote the diagonal matrix containing the eigenvalues of the matrix $W$, and let $U := \{u_i,\ 1 \leq i \leq n\}$ denote the unitary matrix containing the eigenvectors of $W$. Since $W \in \mathbb{S}^n$, $U^{-1} = U^*$ and $W = U\Lambda U^*$. Then,

$$\begin{aligned}\|X - W\|_2^2 &= \text{Tr}\left((X-W)^*(X-W)\right)\\ &= \text{Tr}\left((X-W)(X-W)\right)\\ &= \text{Tr}\left(U^*(X-W)UU^*(X-W)U\right)\\ &= \text{Tr}\left((U^*XU-\Lambda)(U^*XU-\Lambda)\right)\end{aligned}$$

Let's define $\hat{X} := U^*XU$. Since $X \in \mathbb{S}_+^n$, $\hat{X} \in \mathbb{S}_+^n$. Then,

$$\|X - W\|_2^2 = \sum_{i=1}^{n}(\hat{x}_{ii} - \lambda_i)^2 + \sum_{i \neq j}|\hat{x}_{ij}|$$

In order to minimize this quantity, $\hat{x}_{ij} = 0$ for $i \neq j$. And since $\hat{x}_{ii}$ must be positive for $1 \leq i \leq n$ (because $\hat{X} \in \mathbb{S}_+^n$), we obtain:

$$\hat{x}_{ii} = \begin{cases} \lambda_i & \text{for } i : \lambda_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

which means $X = U\hat{X}U^* = \sum_{i:\lambda_i>0} \lambda_i u_i u_i^*$. □

## A.2 Development of the objective function in the $x$-update

The objective function associated with the $x$-update can be developed in term of each variable $\{S_{i0}, l_{i0}, v_{i0}, s_{i0}\}$, such that

$$H_{i0}(x_{i0}) = H_{i0}^{(1)}(S_{i0}) + H_{i0}^{(2)}(\ell_{i0}) + H_{i0}^{(3)}(v_{i0}) + H_{i0}^{(4)}(s_{i0})$$

In that case, those functions can be expressed by :

$$
\begin{aligned}
H_{i0}^{(1)}\left(S_{i0}\right) &= \sum_{j\in\mathcal{N}_i}\left(\left\langle \mu_{ij}^{(1)}, S_{i0}\right\rangle + \frac{\rho}{2}\|S_{i0}-S_{ij}\|_{M_{ij}}^2\right)\\
&= \left\langle \mu_{ii}^{(1)}, S_{i0}\right\rangle + \frac{\rho}{2}\left(2|C_i|+3\right)\|S_{i0}-S_{ii}\|_2^2 + \left\langle \mu_{iA_i}^{(1)}, S_{i0}\right\rangle + \|S_{i0}-S_{iA_i}\|_2^2\\
&= \left\langle \mu_{ii}^{(1)}, S_{i0}\right\rangle + \frac{\rho}{2}\left(2|C_i|+3\right)\left(\|S_{i0}\|_2^2 - 2\left\langle S_{i0}, S_{ii}\right\rangle\right) + \left\langle \mu_{iA_i}^{(1)}, S_{i0}\right\rangle + \frac{\rho}{2}\left(\|S_{i0}\|_2^2 - 2\left\langle S_{i0}, S_{iA_i}\right\rangle\right)\\
&= 2\,\frac{\rho}{2}\left(|C_i|+2\right)\|S_{i0}\|_2^2 - \rho\left\langle S_{i0}, \left(2|C_i|+3\right)S_{ii} + S_{iA_i} - \frac{\mu_{ii}^{(1)}+\mu_{iA_i}^{(1)}}{\rho}\right\rangle\\
&= 2\,\frac{\rho}{2}\left(|C_i|+2\right)\left(\|S_{i0}\|_2^2 - 2\left\langle S_{i0}, \underbrace{\frac{1}{2(|C_i|+2)}\left(\left(2|C_i|+3\right)S_{ii} + S_{iA_i} - \frac{\mu_{ii}^{(1)}+\mu_{iA_i}^{(1)}}{\rho}\right)}_{\hat{S}_i}\right\rangle\right)\\
&= 2\,\frac{\rho}{2}\left(|C_i|+2\right)\left\|S_{i0}-\hat{S}_i\right\|_2^2
\end{aligned}
$$

$$
\begin{aligned}
H_{i0}^{(2)}\left(\ell_{i0}\right) &= \sum_{j\in\mathcal{N}_i}\left(\left\langle \mu_{ij}^{(2)}, \ell_{i0}\right\rangle + \frac{\rho}{2}\|\ell_{i0}-\ell_{ij}\|_{M_{ij}}^2\right)\\
&= \left\langle \mu_{ii}^{(2)}, \ell_{ii}\right\rangle + \frac{\rho}{2}\left(|C_i|+1\right)\|\ell_{i0}-\ell_{ii}\|_2^2 + \left\langle \mu_{iA_i}^{(2)}, \ell_{i0}\right\rangle + \frac{\rho}{2}\|\ell_{i0}-\ell_{iA_i}\|_2^2\\
&= \left\langle \mu_{ii}^{(2)}, \ell_{ii}\right\rangle + \frac{\rho}{2}\left(|C_i|+1\right)\left(\|\ell_{i0}\|_2^2 - 2\left\langle \ell_{i0}, \ell_{ii}\right\rangle\right) + \left\langle \mu_{iA_i}^{(2)}, \ell_{i0}\right\rangle + \frac{\rho}{2}\left(\|\ell_{i0}\|_2^2 - 2\left\langle \ell_{i0}, \ell_{iA_i}\right\rangle\right)\\
&= \frac{\rho}{2}\left(|C_i|+2\right)\left(\|\ell_{i0}\|_2^2 - 2\left\langle \ell_{i0}, \underbrace{\frac{1}{|C_i|+2}\left(\left(|C_i|+1\right)\ell_{ii} + \ell_{iA_i} - \frac{\mu_{ii}^{(2)}+\mu_{iA_i}^{(2)}}{\rho}\right)}_{\hat{\ell}_i}\right\rangle\right)\\
&= \frac{\rho}{2}\left(|C_i|+2\right)\left\|\ell_{i0}-\hat{\ell}_i\right\|_2^2
\end{aligned}
$$

$$
\begin{aligned}
H_{i0}^{(3)}\left(v_{i0}\right) &= \sum_{j\in\mathcal{N}_i}\left(\left\langle \mu_{ij}^{(3)}, v_{i0}\right\rangle + \frac{\rho}{2}\|v_{i0}-v_{ij}\|_{M_{ij}}^2\right)\\
&= \left\langle \mu_{ii}^{(3)}, v_{i0}\right\rangle + 2\,\frac{\rho}{2}\|v_{i0}-v_{ii}\|_2^2 + \sum_{j\in C_i}\left\langle \mu_{ij}^{(3)}, v_{i0}\right\rangle + \frac{\rho}{2}\|v_{i0}-v_{ij}\|_2^2\\
&= \left\langle \mu_{ii}^{(3)}, v_{i0}\right\rangle + 2\,\frac{\rho}{2}\left(\left\langle v_{i0}, v_{i0}\right\rangle - 2\left\langle v_{i0}, v_{ii}\right\rangle\right) + \sum_{j\in C_i}\left\langle \mu_{ij}^{(3)}, v_{i0}\right\rangle + \frac{\rho}{2}\left(\left\langle v_{i0}, v_{i0}\right\rangle - 2\left\langle v_{i0}, v_{ij}\right\rangle\right)\\
&= \frac{\rho}{2}\left(|C_i|+2\right)\left\langle v_{i0}, v_{i0}\right\rangle + \left\langle v_{i0}, \mu_{ii}^{(3)} + \sum_{j\in C_i}\mu_{ij}^{(3)}\right\rangle - 2\,\frac{\rho}{2}\left\langle v_{i0}, 2v_{ii} + \sum_{j\in C_i} v_{ij}\right\rangle\\
&= \frac{\rho}{2}\left(|C_i|+2\right)\left(\left\langle v_{i0}, v_{i0}\right\rangle - 2\left\langle v_{i0}, \underbrace{\frac{1}{|C_i|+2}\left(2v_{ii} + \sum_{j\in C_i} v_{ij} - \frac{\mu_{ii}^{(3)}+\sum_{j\in C_i}\mu_{ij}^{(3)}}{\rho}\right)}_{\hat{v}_i}\right\rangle\right)\\
&= \frac{\rho}{2}\left(|C_i|+2\right)\|v_{i0}-\hat{v}_i\|_2^2
\end{aligned}
$$

$$
\begin{aligned}
H_{i0}^{(4)}\left(s_{i0}\right) &= f_{i0}(s_{i0}) + \sum_{j\in\mathcal{N}_i}\left(\left\langle \mu_{ij}^{(4)}, s_{i0}\right\rangle + \frac{\rho}{2}\|s_{i0}-s_{ij}\|_{M_{ij}}^2\right) \\
&= f_{i0}(s_{i0}) + \left\langle \mu_{ii}^{(4)}, s_{i0}\right\rangle + \|s_{i0}-s_{ii}\|_2^2 \\
&= f_{i0}(s_{i0}) + \left\langle \mu_{ii}^{(4)}, s_{i0}\right\rangle + \frac{\rho}{2}\left(\langle s_{i0}, s_{i0}\rangle - 2\left\langle s_{i0}, s_{ii}\right\rangle\right) \\
&= f_{i0}(s_{i0})\frac{\rho}{2}\left(\langle s_{i0}, s_{i0}\rangle - 2\left\langle s_{i0}, \underbrace{s_{ii} - \frac{\mu_{ii}^{(4)}}{\rho}}_{\hat{s}_i}\right\rangle\right) \\
&= f_{i0}(s_{i0}) + \frac{\rho}{2}\|s_{i0}-\hat{s}_i\|_2^2
\end{aligned}
$$

## A.3 Development of the objective function in the $y$-update

We can decomposed the objective function in the $y$-update according to each variable contained in $y$ :

$$
\begin{aligned}
G_i^{(1)}\left(S_{ii}\right) &= -\left\langle \mu_{ii}^{(1)}, S_{ii}\right\rangle + \frac{\rho}{2}\left(2|C_i|+3\right)\left\|S_{i0}^{+}-S_{ii}\right\|_2^2 \\
&= \frac{\rho}{2}\left(2|C_i|+3\right)\left\langle S_{ii}, S_{ii}\right\rangle - \left\langle S_{ii}, \mu_{ii}^{(1)} + \rho\left(2|C_i|+3\right)S_{i0}^{+}\right\rangle \\
G_i^{(2)}\left(\ell_{ii}\right) &= -\left\langle \mu_{ii}^{(2)}, \ell_{ii}\right\rangle + \frac{\rho}{2}\left(|C_i|+1\right)\left\|\ell_{i0}^{+}-\ell_{ii}\right\|_2^2 \\
&= \frac{\rho}{2}\left(|C_i|+1\right)\left\langle \ell_{ii}, \ell_{ii}\right\rangle - \left\langle \ell_{ii}, \mu_{ii}^{(2)} + \rho\left(|C_i|+1\right)\ell_{i0}^{+}\right\rangle \\
G_i^{(3)}\left(v_{ii}\right) &= -\left\langle \lambda_{i1}, v_{ii}\right\rangle + \frac{\rho}{2}\left\|v_{i1}^{+}-v_{ii}\right\|_2^2 - \left\langle \mu_{ii}^{(3)}, v_{ii}\right\rangle + 2\,\frac{\rho}{2}\left\|v_{i0}^{+}-v_{ii}\right\|_2^2 \\
&= 3\,\frac{\rho}{2}\left\langle v_{ii}, v_{ii}\right\rangle - \left\langle v_{ii}, \lambda_{i1} + \mu_{ii}^{(3)} + \rho v_{i1}^{+} + 2\rho v_{i0}^{+}\right\rangle \\
G_i^{(4)}\left(s_{ii}\right) &= -\left\langle \mu_{ii}^{(4)}, s_{ii}\right\rangle + \frac{\rho}{2}\left\|s_{i0}^{+}-s_{ii}\right\|_2^2 \\
&= \frac{\rho}{2}\left\langle s_{ii}, s_{ii}\right\rangle - \left\langle s_{ii}, \mu_{ii}^{(4)} + \rho s_{i0}^{+}\right\rangle \\
G_i^{(5,j)}\left(S_{ji}\right) &= -\left\langle \mu_{ji}^{(1)}, S_{ji}\right\rangle + \frac{\rho}{2}\left\|S_{j0}^{+}-S_{ji}\right\|_2^2 \\
&= \frac{\rho}{2}\left\langle S_{ji}, S_{ji}\right\rangle - \left\langle S_{ji}, \mu_{ji}^{(1)} + \rho S_{j0}^{+}\right\rangle \\
G_i^{(6,j)}\left(\ell_{ji}\right) &= -\left\langle \mu_{ji}^{(2)}, \ell_{ji}\right\rangle + \frac{\rho}{2}\left\|\ell_{j0}^{+}-\ell_{ji}\right\|_2^2 \\
&= \frac{\rho}{2}\left\langle \ell_{ji}, \ell_{ji}\right\rangle - \left\langle \ell_{ji}, \mu_{ji}^{(2)} + \rho\ell_{j0}^{+}\right\rangle \\
G_i^{(6,j)}\left(v_{A_ii}\right) &= -\left\langle \mu_{A_ii}^{(3)}, v_{A_ii}\right\rangle + \frac{\rho}{2}\left\|v_{A_i0}^{+}-v_{A_ii}\right\|_2^2 \\
&= \frac{\rho}{2}\left\langle v_{A_ii}, v_{A_ii}\right\rangle - \left\langle v_{A_ii}, \mu_{A_ii}^{(3)} + \rho v_{A_i0}^{+}\right\rangle
\end{aligned}
$$

## A.4 Impedance associated with the test network

$$z_1 = \begin{pmatrix} 1.9 + 1.87i & 0.456 + 0.24i & 0.34 + 0.167i \\ \cdot & 1.34 + 1.325i & 0.023 + 0.0067i \\ \cdot & \cdot & 1.576 + 1.234i \end{pmatrix}$$

$$z_2 = \begin{pmatrix} 1.67 + 1.87i & 0.46 + 0.024i & 0 \\ \cdot & 2.43 + 1.844i & 0 \\ \cdot & \cdot & 0 \end{pmatrix}$$

$$z_3 = \begin{pmatrix} 0 & 0 & 0 \\ \cdot & 0 & 0 \\ \cdot & \cdot & 2.37 + 1.87i \end{pmatrix}$$

## A.5 Solution for the test network

| Node | **Active Power** (W) | | | **Reactive Power** (VAR) | | |
|---|---|---|---|---|---|---|
| | Ph1 | Ph2 | Ph3 | Ph1 | Ph2 | Ph3 |
| 0 | 3.55 | 1.52 | 0.37 | 0.108 | 3.24 | -0.39 |
| 1 | -3.1 | -2 | -0.095 | -0.1 | -0.23 | 0.4 |
| 2 | -0.45 | 0.49 | 0 | 0 | -3 | 0 |
| 3 | 0 | 0 | -0.28 | 0 | 0 | 0 |

Table A.1: Net Complex Power Injection

| | | **Active Power** (W) | | | **Reactive Power** (VAR) | | |
|---|---|---|---|---|---|---|---|
| | | Ph1 | Ph2 | Ph3 | Ph1 | Ph2 | Ph3 |
| Node 1 | Ph1 | -3.5499 | 3.5613 | 0.53335 | -0.10031 | 0.29592 | 0.12377 |
| | Ph2 | 1.6795 | -1.5155 | -0.15803 | 3.1295 | -3.2367 | -0.52427 |
| | Ph3 | 1.8727 | -2.0479 | -0.37555 | 3.0259 | 2.9374 | 0.39994 |
| Node 2 | Ph1 | -0.45 | 2.3684 | · | 0 | 1.9149 | · |
| | Ph2 | 0.22199 | 0.49341 | · | 0.39052 | -3 | · |
| | Ph3 | · | · | · | · | · | · |
| Node 3 | Ph1 | · | · | · | · | · | · |
| | Ph2 | · | · | · | · | · | · |
| | Ph3 | · | · | -0.28 | · | · | 0 |

Table A.2: Complex branch power

| Node | **Voltage** (V) | | | **Current** (A) | | |
|---|---|---|---|---|---|---|
| | Ph1 | Ph2 | Ph3 | Ph1 | Ph2 | Ph3 |
| 0 | 50∠0 | 50∠-120 | 50 ∠120 | · | · | · |
| 1 | 49.89∠0 | 49.9 ∠-119.8 | 50 ∠120.13 | 0.0711 ∠178.38 | 0.072 ∠-4.74 | 0.109 ∠-13.06 |
| 2 | 49.9∠0 | 49.82∠-119.6 | · | 0.009 ∠180 | 0.061 ∠- 38.96 | · |
| 3 | · | · | 49.98 ∠120.17 | · | · | 0.038 ∠-14.75 |

Table A.3: Complex voltage and complex current

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**