A Novel Distributed Approach for Optimal Power Flow Problem in Smart Grids

UMBERTO MOSCA

Degree project in
Automatic Control
Master's Thesis
Stockholm, Sweden, January 2013

XR-EE-RT 2013:001
# Contents

1 Introduction 3  
1.1 Energy distribution ............................................. 3  
1.1.1 Structure of electrical network .......................... 3  
1.1.2 Control in electrical networks ............................. 5  
1.2 Optimal Power Flow ............................................ 7  

2 Background 9  
2.1 Related work ................................................... 9  

3 Mathematical Background 13  
3.1 Decomposition ................................................... 13  
3.2 ADMM .............................................................. 16  
3.2.1 Problems with quadratic objective ....................... 19  
3.2.2 Constrained convex optimization ......................... 19  
3.2.3 Global variable consensus optimization ............... 20  
3.2.4 Global variable consensus optimization with regular-
ization ................................................................. 22  
3.2.5 General form consensus optimization ................... 23  
3.3 F-Lipschitz ....................................................... 24  

4 Mathematical Modeling of problem 27  

5 Solution Procedure 34  
5.1 Centralized formulation ....................................... 35  
5.2 Distributed formulation ....................................... 37  
5.2.1 Partially distributed formulation ........................ 37  
5.2.2 Distributed formulation .................................... 40  

6 Numerical results 46  

7 Conclusion and future work 58
Abstract

In this thesis we study a classical problem of the electrical field; the optimal power flow (OPF) in an electrical network. Given a power grid, the problem is to find the optimal production of generators respecting all the constraints imposed by physics, like Kirchhoff’ equations and power bounds on each part of network. The goal of a power flow problem is to obtain complete voltage angle and magnitude information for each bus in a power system for operating conditions. Solving this problem in a centralized manner for a very large networks becomes difficult due to computational limitations and become undesirable due to safety reasons. The development of computational ability in each component of the network has opened new horizons, linking the electrical and ICT engineering. With the rapid development of smart grid infrastructures, the OPF problem is becoming very important. Scalability and the fast convergent properties of the associated solution methods are highly desirable in a practical point of view. One of the main challenge in the OPF problem is the decoupling of the constraints enforced by the Kirchhoff’ laws. Our contribution has been to propose a new formulation of the problem so that the bigger problem can be decomposable into a number of subproblems (one for each node), which relies on only the local information available. As a result, our proposed protocols are scalable. Moreover, we adopt the state-of-the-art alternating direction method of multipliers (ADMM), which blends fast convergent properties into the proposed protocol. We also propose a partially distributed protocol based on ADMM, which relies on an intelligent central controller to handle the associated constraints of the OPF problem. In this case, the computational burden at nodes are very small, thus, the nodes can be unintelligent. Finally, we provide numerical experiments to illustrate the behavior of proposed algorithms.
Chapter 1

Introduction

1.1 Energy distribution

Electric energy is an essential ingredient for the development of any country. It is a serviceable form of energy, because it can be generated centrally in bulk and transmitted economically over long distances. Further it can be adapted easily to domestic and industrial applications, like lighting purposes and drives. Over the years some notable change has occurred in the general power system structure. Generation power plants have been moved away from the load center. Complex transmission networks have been built to link these generation plants among themselves and to the load centers, as well as to neighboring utilities. The overall combination is advantageous, in that the construction and operation of large generation plants provides an economy of scale, and their interconnection to the entire network of load ensures a higher level of reliability of supply for each load. Electric energy today constitutes about 30% of the total annual energy consumption on a worldwide basis. Transportation can be expected to go electric in a big way in the long run, when non-conventional energy resources are well developed. The demand for increasing amounts of energy has spurred the construction of large, complex power systems, comprising of many generating plant and intricate, widespread networks for transmission and distribution.

1.1.1 Structure of electrical network

Electricity, unlike oil and gas, cannot be stored economically (except in very small quantities in batteries) and the electric utility can exercise a little control over the load at any time. The power system must, therefore, be capable of matching the output from generators to the demand at any time at a specified voltage and frequency. Many components influence the energy
demand during the day, like weather, season, time of the day, holyday and a purely randomly varying component of relatively small amplitude. The average load determines the energy consumption over the day, while the peak load along with considerations of standby capacity determines plant capacity for meeting the load. The unvarying (or slowly varying over many hours) portion of the electric demand is known as the base load and is generally served best by large facilities (which are therefore efficient due to economies of scale) with low variable costs for fuel and operations. Such facilities might be nuclear, coal-fired power stations or hydroelectric, but other renewable energy sources such as concentrated solar thermal and geothermal powers have the potential to provide base load power. Renewable energy sources such as solar photovoltaic, wind, wave, and tidal are, due to their intermittency, not considered "base load" but can still add power to the grid. The remaining power demand, if any, is supplied by peaking power plants, which are typically smaller, faster-responding, and higher cost sources, such as combined cycle or combustion turbine plants fueled by natural gas. The necessity to conserve fossil fuels, to protect environment and the need of sustainable development has forced to search for unconventional sources of electric energy. Renewable and non-conventional forms of energy will play an increasingly important role in the future. Generating stations, transmission lines and the distribution system are the main components of an electric power system. Generating station and a distribution system are connected through transmission lines, which also connect one power system to another. A distribution system connects all the loads in a particular area to the transmission lines. As the transmission capability of a line is proportional to the square of its voltage, research is continuously being carried out to raise transmission voltages. The reduced current flowing through the line reduces the losses in the conductors. According to Joule’s Law, energy losses are directly proportional to the square of the current. Energy is usually transmitted within a grid with three-phase AC. Single-phase AC is used only for distribution to end users since it is not usable for large polyphase induction motors. For very long distance (over 600 Km) is convenient to transmit power by DC transmission. It also obviates some of technical problems associates with very long distance AC transmissions. The first step down of voltage from transmission level is at the power substation. It is stepped down and sent to smaller substations in towns and neighborhoods. Sub transmission circuits are usually arranged in loops so that a single line failure does not cut off service to a large number of customers for more than a short time. The next step down in voltage is at the distribution substation. Finally, the distribution system, taking power from the distribution transformer stations, supplies power to the domestical/commercial/industrial consumers.
1.1.2 Control in electrical networks

The electric power system has undergone extensive change over the past several decades and has become substantially more complex, dynamic, and uncertain as new market rules, business practices, regulatory policies, and electric generation, transmission, distribution, storage, and end-use technologies have been tried and adopted. Since the early 1970’s many utilities have built computerized control center to aid in the operation of their systems. SCADA (Supervisory control and data acquisition functions) were the first to be implemented in these centers. Measurements from the power system are continuously sent to a central location in real time and compared to estimated values from a state estimator program. The verified quantities are then checked for system security and for the reliability of the network configuration to supply the load. Other online functions implemented in the systems are economic dispatch and load frequency control. The first one computes the most economical distribution of generations, given the list of available generators. The second one supervises energy interchanges and controls the system frequency in response to imbalance between the system’s generation and its load. Due to their complexity, some other useful functions are used off-line, as tools for analysis. Effective power system planning, operations, and communications requires power system operators to analyze vast
amounts of data, such as automated and computer-assisted control data, grid telemetry, market information, environmental information, and others. With the advent of Smart Grid technologies, data will become richer and denser. More data can yield more precise forecasts and enable more robust active control. With S.G., we add intelligence to an electric power transmission system, we will have independent processors in each component and at each substation and power plant. These processors must have a robust operating system and be able to act as independent agents that can communicate and cooperate with others, forming a large distributed computing platform. Each agent must be connected to sensors associated with its own component or its own substation so that it can assess its own operating conditions and report them to its neighboring agents via the communications paths. The Smart Grid will manage and deliver electrical energy through a combined centralized and distributed system, in which many nodes are capable of producing, consuming, and storing electrical energy; [1]

Microgrids are power generation/distribution systems in which users and generators are in close proximity. This results in relatively low voltage grids (few hundred kVA). Generation is often done using renewable generation sources such as photovoltaic cells or wind turbines. Power generation can also be accomplished through small microturbines and gas/diesel generators. Storage devices such as battery banks represent another important power source for microgrids. These units can be used in places such as office buildings, parks and homes as distributed power sources. All the microgrids in the network can work in a cooperative way to meet the overall load demand in the network. Sensors at each node of the grid will automatically provide continuous data about energy use, flow, and system status. Managing transactions on this network will involve extensive communications networks, real-time data monitoring and analysis, and distributed and hierarchical control schemes. The general problem of energy management which faces the power utilities to satisfy customer demand in a safe, reliable and cost effective manner is a very complex one. It requires much insight into workings of the power system, for sure, but also a good working knowledge of mathematical optimization theory. Many problems of power system management and control have been formulated and cover a large part of mathematical programming discipline, ranging from very long term (ten to fifteen years for generation and transmission planning), to very short term (a few minutes for dispatching and few milliseconds for power imbalance). Due to their complexities, each problem is usually treated separately. They are usually performed in a hierarchy, from long term to short term, with output of long term tasks serving as targets for the short term tasks. The short term functions are grouped under the category of power system operation. Some of
the more important tasks in this group are economic dispatch, minimum
loss dispatching, minimum load shedding and minimum deviation from op-
erating point. These problems and others, which are subject to the load flow
equation as constraints, share a common nonlinear programming formulation
called optimal power flow, denoted OPF. Optimal power flow modeling is an
important tool for determining the most efficient and economical operation
of existing power systems as well as for planning future expansion. However,
full-scale nonlinear OPF has not been widely adopted in real-time operation
of large-scale power systems because too complex to be solved in a short
time. Instead, system operators often use simplified OPF tools. Historically,
this is due both to the lack of powerful computing hardware in the industry
and to the lack of efficient and robust OPF algorithms. With the advent of
fast, low-cost computers, speed has now become a secondary concern, while
algorithm robustness and scalability are the primary issues. The full OPF
serves two purposes. In operations, it periodically sets optimal target val-
ues for the electrical variables of the power network, following the system’s
varying load. Based on the OPF’s optimal values, the variables can then dis-
patched every few minutes to follow smart variations in load, using simpler
algorithms. A second application for OPF is in the system planning, where it
is used to study the effects of parameter variations on power’s system optimal
operation.

1.2 Optimal Power Flow

Active/reactive power dispatch problems have been the research subject
of power system community since in the early 1960’s. The optimal power
flow problem was proposed by Carpentier in the 1962 based on the eco-
nomic dispatch problem. Since then, the OPF problem has been extensively
studied in the literature and numerous algorithms have been proposed for
solving this highly nonconvex problem. Many such methods are based on
the Karush-Kuhn-Tucker (KKT) necessary conditions, which can only guar-
antee a locally optimal solution due to nonconvexity of the OPF problem.
The theory developed by KKT optimality conditions is able to characterize
the solution. However, solving the KKT system to obtain even a suboptimal
solution is often a difficult task except for some special problems. Based on
present requirements, it is desirable in practice to aim at scalable solution
methods with fast convergence properties.

The performance of solution methods in the power system analysis area
dependent on many factors, such as the nature of the system model, the
type of nonlinearities, the type and the number of constraints, etc. The
OPF problem can be considered as one of the most useful tool for power system analysis. In General, the OPF results in is a non linear programming problem that determines the optimal control set points of the system to minimize a desired objective function, subject to certain system constraints. The primary goal of a generic OPF is to minimize the costs of meeting the load demand for a power system while maintaining the security of the system. Typical objective functions include the minimum operational cost and the minimum active power losses. The cost associated with the power system usually is the cost for generating power (megawatts, MW) at each generator. The objective function usually depends on parameters with a direct economic impact (e.g., power generation or load) and parameters without a direct cost impact (e.g., bus voltage magnitude and phase). For generators, the OPF will control generator outputs (in MW) as well as generator voltage (in Volts, V). For the transmission system, the OPF may control the tap ratio or phase shift angle for variable transformers, switched shunt control, and all other flexible AC transmission system devices.

In addition to optimizing the costs of generation, the problem formulation also considers the physical constraints of the network. Such constraints are necessary to maintain system’s security, e.g., avoiding overloading of the power lines. The maintenance of system security requires keeping each device in the power system within its desired operational range at the steady-state. This includes limits on the maximum and the minimum outputs for generators, limits on the maximum MVA flows on transmission lines and transformers, and limits on the system bus voltages. Apart from those inequality constraints, the OPF problem incorporates equality constraints as well. The equality constraint set typically consists of both active and reactive power balance equation at each bus of the network.

It should be noted that the OPF only addresses steady-state operation of the power system. Topics such as transient stability, dynamic stability, and steady-state contingency analysis are not addressed. The OPF problem can be formulated in many different ways, including additional features useful for the management of a real grid. It can calculate the redistribution and the release of the loads in case of an overload or an emergency. It also can provide a preventive dispatch if the system constraints (physical limit) are about to be violated. Moreover, the OPF can calculate the stress in each transmission line and many others.
Chapter 2

Background

In this chapter, we will make a brief review of methods used to solve the OPF problem formulations. Since it was introduced, OPF problem has been extensively studied in the literature and numerous algorithms have been proposed for solving this highly nonconvex problem including linear programming, Newton Raphson, quadratic programming, nonlinear programming, Lagrange relaxation based approaches, interior point methods and, in recent years, also artificial intelligence methods, like artificial neural network, fuzzy logic, genetic algorithm, evolutionary programming and particle swarm optimization. Some of these methods are based on the KKT necessary conditions, which can only guarantee a locally optimal solution due to nonconvexity of the OPF problem [15].

2.1 Related work

Let us now discuss some relevant papers that address the OPF considered in this thesis. Standard linear programming (LP) formulation, of course, requires the linearization of the objective function and the nonnegativity of the variables. In [2] authors proposed a LP based OPF for minimization of transmission losses and generator reactive margins of the Spanish power system. The objective function consists of minimizing transmission losses and generator reactive outputs. The problem constraints are the power flow equations and the bounds of the power system variables (generator reactive outputs, bus voltages, transformer taps, and branch power flows). The method is an iterative process that linearizes in each iteration both the objective function and the constraints. In particular, the objective function is approximated by a piecewise linear function. The discrete nature of shunt reactors and capacitors is modeled by integer variables. This approach is bet-
ter than the ones, which linearizes only the objective function [2] once. In [3] authors have proposed two methods, based on Newton method. The two methods are compared with the Newton-based full AC load flow method and the fast decoupled load flow in three test systems to demonstrate their convergence characteristics. In [4] authors have presented a semi smooth Newton algorithms for solving OPF problems. It treated general inequality constraints and bounded constraints separately. By introducing a diagonal matrix and the nonlinear complementarity function, the KKT system of the OPF is transformed equivalently to a system of no smooth bounded constrained equations. Comparing with the classical OPF methods this treatment has two advantages. First, advantage is that it handles the inequality constraints in OPF problems. Second, it reduces the number of dual variables in the KKT system. Based on the reformulated equations, the paper proposes a algorithm which has nice global and local convergence property. Furthermore the paper presents a decoupled semi smooth Newton-type algorithm. The decoupled method solves the system of equations via solving two lower dimension problems. Therefore the method saves the computing cost in theory.

OPF problems have also been formulated as quadratic programming, where the objective function is quadratic and constraints are linear. Authors in [5] presented an extension of basic Kuhn-Tucker conditions and employing a generalized Quadratic-Based model for OPF. The conditions for feasibility, convergence and optimality are included in the construction of the OPF algorithm. It is also capable of using hierarchical structures to include multiple objective functions and selectable constraints. The generalized algorithm using sensitivity of objective functions yields a global optimal solution. The algorithm has been tested using different objective functions on actual power systems, and optimal solutions reached in relatively few iterations. Another method used in literature to solve the problem is the interior point method. An implementation of an interior point method to the optimal reactive dispatch problem is described in [6]. Numerical results presented in [6] shows that this technique can be very effective to relatively large scale optimal power flow applications. One advantage of this method is that the number of iterations is not very sensitive to network size or number of control variables.

The reference [7] has presented an artificial intelligence method, (Fuzzy Logic approach). The fuzzy formulation is then converted into another formulation that is solved using a standard OPF method. This method formulated the contingency constrained OPF problem in a decomposed form that allows post-contingency corrective rescheduling. Contingencies refer to disturbances, such as line outages or generator outages that can cause sudden
and large changes in both the configuration and the state of a system. Contingencies often result in severe violations of operating constraints. Consequently, planning for contingencies forms an important aspect of secure operation. The function of a contingency constrained optimal power flow is to schedule power system controls to achieve operation at a desired security level, while minimizing the generator fuel costs and any other economic factors. The proposed approach in [7] can yield Pareto curves that can guide the system operator regarding the tradeoff between cost and security against contingencies.

Another solution method for OPF is based on genetic algorithm. Those methods belong to the category of random search algorithms, which simulate the evolution process based on the theory of survival of the fittest [8]. The genetic algorithm emulates the optimization techniques found in nature. This optimization algorithm does not require the strict continuity of classical search techniques, but allows non-linearities and discontinuities to appear in the solution space. The application of this algorithm to the economic dispatch problem uses the payoff information of an objective function to determine optimality. An application of Genetic Algorithm to solve an economic dispatch problem is found in [8]. The algorithm utilizes payoff information of candidate solutions to evaluate their optimality.

Artificial intelligence methods can work with nonlinearities and discontinuities found commonly in physical systems, like power systems. They are characterized by highly nonlinear and computationally difficult environment with a need for optimality. Instead classical Lagrangian techniques in economic dispatch require the cost curve of generator must be convex and most industrial algorithms require the incremental cost curves to be piecewise-linear. The input-output characteristics produced by generator operation can be made to approximate this requirement. But the loss of accuracy induced by these approximations is not desirable. Thus, with artificial intelligence methods, the constraints of classical Lagrangian techniques on unit curves are bypassed.

Reference [9] presented a solution of OPF problem of a power system via a simple particle swarm optimization (PSO) algorithm. This method is dynamic in nature and it overcomes the shortcomings of other evolutionary computation techniques such as premature convergence and provides high quality solutions. The objective is to minimize the fuel cost and keep the power outputs of generators, bus voltages, shunt capacitors/reactors and transformers tap setting in their secure limits. PSO is an optimization method based on population, and it can be used to solve many complex optimization problems, which are nonlinear and non differentiable. The most important merit of PSO is its fast convergence speed. PSO has been applied to various powers
system optimization problems with impressive success [9].

In summary we can say that the classical methods, have the following disadvantages: they are limited to handle the dynamic of systems, can converge incorrectly to local solution and they have difficulty to solve large system due to too big matrices. In most cases, mathematical formulations have to be simplified to get the solutions because the real power system problems are too inherently complex due to nonconvexity. Moreover, those methods have poor convergence and the optimality is not guaranteed. In contrast, methods based on artificial intelligence are relatively versatile for handling various qualitative constraints, and can support multi objective cost function. Their main disadvantage is higher computational time. The algorithm we propose is based on ADMM method. We can consider it belonging to the first category described above. But, this formulation can solve some disadvantages of this category using some mathematical expedients. At the end, we are able to obtain an execution time independent of the network size.
Chapter 3

Mathematical Background

In this chapter we present some classical as well as state-of-the-art distributed optimization methods. In particular, we discuss general dual decomposition techniques and alternating direction method of multipliers (ADMM). These methods are used in later chapters in this thesis for distributed algorithm developments of the optimal power flow (OPF) problem in Smart Grid. For the completeness, we also discuss in this chapter the recent F-Lipschitz framework for distributed optimization, though it is not directly applicable in the context of our considered OPF problem.

3.1 Decomposition

The material presented in this section is essentially based from [11]. Decomposition is a general approach to solving a problem by breaking it up into smaller ones and solving each of the smaller ones separately, either in parallel or sequentially. As a general example of such a problem, suppose the variable \( x \) can be partitioned into subvectors \( x_1, \ldots, x_k \), the objective is a sum of functions of \( x_i \), and each constraint involves only variables from one of the subvectors \( x_i \). Then evidently we can solve each problem involving \( x_i \) separately (and in parallel), and then re-assemble the solution \( x \). Of course this is a trivial but a more interesting situation occurs when there is some coupling or interaction between the subvectors, so the problems cannot be solved independently. For these cases there are techniques that solve the overall problem by iteratively solving a sequence of smaller problems.

For example, the variables might be partitioned into subvectors, some of which are local (i.e., appear in one subproblem only) and some of which are complicating (i.e., appear in more than one subproblem). This decomposition structure can be represented by a hypergraph. The nodes are associated
with the subproblems, which involve local variables, objective terms, and local constraints. The hyperedges or nets are associated with complicating variables or constraints. If a hyperedge is adjacent to only two nodes, we call it a link. A link corresponds to a shared variable or constraint between the two subproblems represented by the nodes. Now we describe decomposition with a general structure. We have K subsystems. Subsystem \( i \) has private variables \( x_i \in \mathbb{R}^{n_i} \), public variables \( y_i \in \mathbb{R}^{p_i} \), objective function \( f : \mathbb{R}^{n_i} \times \mathbb{R}^{p_i} \to \mathbb{R} \), and local constraint set \( C_i \subseteq \mathbb{R}^{n_i} \times \mathbb{R}^{p_i} \). The overall objective is \( \sum_{i=0}^{K} f(x_i, y_i) \), and the local constraints are \((x_i, y_i) \in C_i\). These subsystems are coupled through constraints that require various subsets of the components of the public variables to be equal. To describe this we collect all the public variables together into one vector variable \( y = (y_1, \ldots, y_K) \in \mathbb{R}^p \), where \( p = p_1 + \ldots + p_K \) is the sum of the total public variables. Note that \( p_i \) is the number of public variable associated with subsystem \( i \). Suppose there are \( N \) nets. We introduce a vector \( z \in \mathbb{R}^N \) that gives the common values of the public variables on the nets. We can express the coupling constraints as \( y_i = E_i z \), \( i = 1, \ldots, K \).

The problem is of the form:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{K} f_i(x_i, y_i) \\
\text{subject to} & \quad (x_i, y_i) \in C_i, \quad i = 1, \ldots, K \\
& \quad y_i = E_i z, \quad i = 1, \ldots, K,
\end{align*}
\]

where the variables are \( x = (x_1, \ldots, x_K) \), \( y \), and \( z \). By applying the dual decomposition we write the partial Lagrangian of problem:
\[ L(x, y, z, \lambda) = \sum_{i=1}^{K} f_i(x_i, y_i) + \lambda^T (y - E z) \]  
(3.5)

\[ = \sum_{i=1}^{K} (f_i(x_i, y_i) + \lambda_i^T y_i) - \lambda^T E z , \]  
(3.6)

where \( \lambda \in \mathbb{R}^p \) is the Lagrange multiplier associated with \( y = E z \), and \( \lambda_i \) is the subvector of \( \lambda \) associated with the \( i \)th subsystem. To find the dual function we need minimization over \( (x, y, z) \). Note that the minimization with respect to \( z \) indicates that \( E^T \lambda = 0 \) (Otherwise, the dual function is unbounded below), which is a constraint on dual variables. Moreover, we can independently perform the minimization at each subsystem with respect to \( (x_i, y_i) \). Let \( g_i(\lambda_i) \) denote the optimal value of the following problem

\[
\text{minimize} \quad f_i(x_i, y_i) + \lambda_i^T y_i \\
\text{subject to} \quad (x_i, y_i) \in C_i ,
\]  
(3.7)

where the variable is \( (x_i, y_i) \). Thus, the dual problem can be formally expressed as

\[
\text{maximize} \quad \sum_{i=1}^{K} g_i(\lambda_i) \\
\text{subject to} \quad E^T \lambda = 0 ,
\]  
(3.8)

where the variable is \( \lambda \). We use projected subgradient method to solve the dual problem 3.8, and is given by the following algorithm [11]:

\[
\begin{align*}
\text{given} \quad & \text{an initial value of} \quad y \quad \text{such that} \quad E^T y = 0 \\
\text{repeat} & \\
1. \quad & \text{Solve subproblems} \quad 3.7 \quad \text{to obtain} \quad x_i, \ y_i \\
2. \quad & \text{Compute average} \quad \hat{z} := (E^T E)^{-1} E^T y \\
3. \quad & \text{Update prices} \quad v := v + \alpha_k (y - E \hat{z})
\end{align*}
\]

In the first step, the algorithm optimize the subsystems, solving problems (3.7) in an independent way. The subsystems’ updates are carried out independently. In the second step instead is taken an average, over each net, of the vector \( y \), which collect all copies of the public variables. Each component of \( z \) is updated by averaging the values of all its copies in the different subsystems, this because \( E’E \) a diagonal matrix, which shows the number of neighbors of each node. Finally, the vector \( y \) computed in the last step, is the projection of the current values of the public variables onto the set of feasible, or consistent values of public variables.
3.2 ADMM

The material presented in this section is based on [10]. We refer the reader to [10] for details. The main motivation for using ADMM is that it combines the benefits of dual decomposition and augmented Lagrangian methods for constrained optimizations. The result is a distributed algorithm with fast (compared to the subgradient method) convergence properties. We start with a simple convex constrained optimization problem. Then, we describe the dual ascent algorithm and augmented Lagrangian method for solving this problem, which are the key ingredients for ADMM. Next we present the ADMM algorithm and its applicability in the context of some important problem formulations, which are used in later chapters for addressing OPF problem. Consider the following convex constrained optimization problem

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad Ax = b ,
\end{align*}
\]

where the variable is \( x \). The associated lagrangian is

\[
L(x, y) = f(x) + y^T(Ax - b) ,
\]

where \( y \) is the dual variable or Lagrange multiplier associated with the equality constraint. The dual problem is given by

\[
\begin{align*}
\text{maximize} & \quad g(y) ,
\end{align*}
\]

where \( g(y) = \inf_x L(x, y) \).

In the dual ascent method, we solve the dual problem using gradient ascent. Assuming that \( g \) is differentiable, the gradient \( \nabla g \) of \( g \) at \( y \) is given by

\[
\nabla g(y) = Ax^+ - b ,
\]

where

\[
x^+ = \arg \min_x L(x, y) .
\]

Thus, the dual ascent algorithm is given by

\[
\begin{align*}
x^{k+1} &= \arg \min_x L(x, y^k) \\
y^{k+1} &= y^k + \alpha^k(Ax^{k+1} - b) ,
\end{align*}
\]

where \( \alpha^k \) is the step size. Note that \( Ax^{k+1} - b \) corresponds to the gradient of \( g \) at \( y^k \), i.e., \( \nabla g(y^k) \). The first step is an \( x \)-minimization step, and the second step is a dual variable update. The dual variable \( y \) can be interpreted as a
vector of prices. This algorithm is called dual ascent since, with appropriate choice of $\alpha$, the dual function increases in each step.

As we have noticed in Section 3.1, the great advantage of the dual ascent method is that, in some cases, the algorithm leads to decentralized algorithms. For example, suppose that the objective $f$ in problem (3.9) is separable as follows

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

(3.16)

where $x = (x_1, \ldots, x_N)$. Moreover assume that $A$ is partitioned as $A = [A_1 \ldots A_N]$. Then the Lagrangian is given by

$$L(x, y) = \sum_{i=1}^{N} L_i(x_i, y) - y^T b ,$$

(3.17)

where

$$L_i(x_i, y) = f_i(x_i) + y^T A_i x_i .$$

(3.18)

This means that the $x$-minimization step (3.14) splits into $N$ separate problems that can be solved in parallel. In particular, we have the following iterations:

$$x^{k+1}_i = \arg \min_{x_i} L_i(x_i, y^k)$$

(3.19)

$$y^{k+1} = y^k + \rho (Ax^{k+1} - b) .$$

(3.20)

However, in the case of convergence, the dual ascent algorithm discussed above heavily relies on assumptions like strict convexity or finiteness of $f$ [10]. Let us now describe the augmented lagrangian method, which gracefully achieves convergence without such assumptions, and that more robust. Here we consider the following equivalent problem formulation, instead of original problem (3.9)

$$\begin{align*}
\text{minimize} & \quad f(x) + \frac{\rho}{2} ||Ax - b||_2^2 \\
\text{subject to} & \quad Ax = b ,
\end{align*}$$

(3.21)

where the variable is $x$ and $\rho$ is a positive scalar. We denote by $L_\rho(x, y)$ the Lagrangian associated with problem (3.21), and is given by

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

(3.22)

where $y$ represents the dual variables associated with the equality constraint. Applying dual ascent to the modified problem (3.21) yields the algorithm:

$$x^{k+1} = \arg \min_{x} L_\rho(x, y^k)$$

(3.23)

$$y^{k+1} = y^k + \rho (Ax^{k+1} - b) .$$

(3.24)
which is known as the method of multipliers. Note that the $x$-minimization step uses the augmented Lagrangian $L_\rho(x, y)$, instead of $L(x, y)$ in (3.10). Moreover, the penalty parameter $\rho$ is used in the place of $\alpha_k$ in (3.15). The conditions for convergence of the method of multipliers (3.23)-(3.24) is far more general compared to the dual ascent method (3.14)-(3.15) [10]. However, when $f$ is separable (e.g., see (3.16)), the augmented Lagrangian $L_\rho$ is not separable, because of the quadratic term $||Ax - b||^2_2$. As a result, the $x$-minimization step (3.23) cannot be performed in parallel for each $x_i$.

We next describe the ADMM method, which can be considered as a blend between the dual ascent algorithm and the method of multipliers. Let us consider the problem of the form

$$\begin{align*}
\text{minimize} & \quad f(x) + g(z) \\
\text{subject to} & \quad Ax + Bz = c ,
\end{align*}$$

(3.25)

with variables $x \in R^n$ and $z \in R^m$. The augmented Lagrangian for problem (3.25) is given by

$$L_\rho(x, z, y) = f(x) + g(z) + y^T (Ax + Bz - c) + (\rho/2)||Ax + Bz - c||^2_2 ,$$

(3.26)

where $y$ denotes the dual variables as usual. Note that the direct application of method of multiplier method (3.23)-(3.24) for problem (3.25) results

$$\begin{align*}
(x^{k+1}, z^{k+1}) &= \arg \min_{x, z} L_\rho(x, z, y^k) \\
y^{k+1} &= y^k + \rho (Ax^{k+1} + Bz^{k+1} - c) .
\end{align*}$$

(3.27)

(3.28)

In contrast, ADMM split the $(x, z)$-minimization step (3.27) into two sequential updates, namely $x$-minimization and $z$-minimization. Specifically, ADMM consist of the iteration

$$\begin{align*}
x^{k+1} &= \arg \min_{x} L_\rho(x, z^k, y^k) \\
z^{k+1} &= \arg \min_{z} L_\rho(x^{k+1}, z, y^k) \\
y^{k+1} &= y^k + \rho (Ax^{k+1} + Bz^{k+1} - c) .
\end{align*}$$

(3.29)

(3.30)

(3.31)

where $\rho$ is a positive scalar. There are many convergence results for ADMM in the literature. Here, we do not go into explicit details and refer the reader for [cite boyd]. However, under the assumptions 1) $f$ and $g$ are closed, proper, and convex, 2) the augmented Lagrangian has a saddle point, we list 3 interesting convergence properties of ADMM algorithm [10]:

1. **Iterates approach feasibility**: $Ax^k + Bz^k - c \to 0$ as $k \to \infty$
2. **Objective function of the iterates approaches optimal value:** \( f(x^k) + g(z^k) \to p^* \) as \( k \to \infty \), where \( p^* \) is the optimal value.

3. **Dual variable convergence:** \( y^k \to y^* \) as \( k \to \infty \), where \( y^* \) is the dual optimal point.

Compared to interior point algorithms, which are based on the Newton’s method, the convergence of ADMM algorithm is noticeably slow. However, the convergence is faster compared to the classical dual ascent methods. ADMM algorithm can typically produce results with modest accuracy within hundreds of iterations. As a result, ADMM algorithm is suitable for applications, which do not require results with a very high accuracy.

In the rest of this section we give some important problem formulations, which are used in later chapters.

### 3.2.1 Problems with quadratic objective

Here we consider the case of quadratic objective terms. With this particular structure of \( f \) and \( g \), we can carry out the \( x \) and \( z \) updates more efficiently.

Suppose \( f \) is given by the (convex) quadratic function

\[
    f(x) = \frac{1}{2}x^TPx + qx + r,
    \tag{3.32}
\]

where \( P \) is a positive semidefinite \( n \times n \) matrices. This includes the cases when \( f \) is linear or constant, by setting \( P \), or both \( P \) and \( q \), to zero. Then, assuming \( P + \rho A^T A \) is invertible, we have

\[
    x^+ = (P + \rho A^T)^{-1}(\rho A^Tv - q). \tag{3.33}
\]

The computing the \( x \)-update solving a linear system can substantially improve performance.

### 3.2.2 Constrained convex optimization

The generic constrained convex optimization problem is

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in C,
\end{align*}
\tag{3.34}
\]

with variable \( x \in \mathbb{R}^n \), where \( f \) and \( C \) are convex. This problem can be rewritten in ADMM form as
minimize \( f(x) + g(z) \)
subject to \( x - z = 0 \), \hspace{1cm} (3.35)

where \( g \) is the indicator function of \( C \).

\[ g(z) = \begin{cases} 0 & \text{if } z \in C \\ \infty & \text{if } z \notin C \end{cases} \]

the associated lagrangian is

\[ L_\rho(x, z, y) = f(x) + g(z) + y^T(x - z) + (\rho/2)||x - z||_2^2, \]

so we can write

\[ x^{k+1} = \text{arg min}_x L_\rho(x, z^k, y^k) \hspace{1cm} (3.36) \]
\[ z^{k+1} = \text{arg min}_z L_\rho(x^{k+1}, z, y^k) = \pi_C(x^{k+1} + y^k/\rho) \hspace{1cm} (3.37) \]
\[ y^{k+1} = y^k + \rho(x^{k+1} - z^{k+1}) \hspace{1cm} (3.38) \]

the \( x \)-update involves minimizing \( f \) plus a convex quadratic function evaluation of the proximal operator associated with \( f \). The \( z \) update is Euclidean projection onto \( C \).

### 3.2.3 Global variable consensus optimization

Consensus is a generic optimization problem who can be solved via, ADMM-based methods in a distributed way. For more details you can refer to [10]. We first consider the case with a single global variable, with the objective and constraint terms split into \( N \) parts:

\[ \text{minimize } f(x) = \sum_{i=0}^{N} f_i(x), \hspace{1cm} (3.39) \]

where \( x \in \mathbb{R}^n \) and \( f_i : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\} \) are convex. Each term can also encode constraints by assigning \( f_i(x) = \infty \) when a constraint is violated. The goal is to solve the problem above in such a way that each term can be handled by its own processing element. This problem can be rewritten with local variables \( x_i \in \mathbb{R}^n \) and a common global variable \( z \):

\[ \text{minimize } \sum_{i=0}^{N} f_i(x) \]
\[ \text{subject to } x_i - z = 0 \quad i = 1...N, \hspace{1cm} (3.40) \]

where the variables are \( \{x_i\}_{i=1,...,N} \) and \( z \).
This is called the global consensus problem, since the constraint is that all the local variables should agree be equal. The resulting ADMM algorithm is the following:

$$x_i^{k+1} = \arg \min_{x_i^k} (f_i(x_i^k) + y_i^k(x_i^k - z^k) + \frac{\rho}{2}||x_i^k - z^k||^2_2)$$  (3.41)

$$z^{k+1} = \frac{1}{N} \sum_{i=1}^{N} (x_i^{k+1} + (1/\rho)y_i^k)$$  (3.42)

$$y_i^{k+1} = y_i^k + \rho(x_i^{k+1} - z^{k+1}).$$  (3.43)

The first and last steps are carried out independently for each $i = 1, \ldots, N$. The dual variables are separately updated to drive the variables into consensus, and quadratic regularization helps pull the variables toward their average value while still attempting to minimize each local $f_i$. We can interpret consensus ADMM as a method for solving problems in which the objective and constraints are distributed across multiple processors. Each processor only has to handle its own objective and constraint term, plus a quadratic term which is updated each iteration. This term is updated in such a way that the variables converge to a common value, which is the solution of the full problem.
3.2.4 Global variable consensus optimization with regularization

We can make a simple variation on the global variable consensus problem adding an objective term $g$, often representing a simple constraint or regularization. The associated problem is:

$$\text{minimize } \sum_{i=0}^{N} f_i(x) + g(z),$$

(3.44)

with variables $x$. We can equivalently reformulate the problem above as

$$\text{minimize } \sum_{i=0}^{N} f_i(x) + g(z)$$

subject to $x_i - z = 0 \quad i = 1,...,N,$

(3.45)

where the variables are $\{x_i\}_{i=1,...,N}$ and $z$. The resulting ADMM algorithm applied to problem 3.44 is as follows:

$$x_i^{k+1} = \arg \min_{x_i} (f_i(x_i) + y_i^{kT} (x_i - z^k) + (\rho/2)||x_i - z^k||^2_2)$$

(3.46)

$$z^{k+1} = \arg \min_{z} (g(z) + \sum_{i=1}^{N} (-y_i^{kT} z + (\rho/2)||x_i^{k+1} - z||^2_2))$$

(3.47)

$$y_i^{k+1} = y_i^k + \rho (x_i^{k+1} - z^{k+1}).$$

(3.48)

Now we consider a more general form of the consensus minimization problem, in which we have local variables $x_i \in \mathbb{R}^{n_i}$, $i = 1,...,N$, with the objective $f_1(x_1) + + f_N(x_N)$ separable in the $x_i$. Each of these local variables consists...
of a selection of the components of the global variable $z \in R^n$; that is, each component of each local variable corresponds to some global variable component $z_g$. Reach consensus between the local variables and the global variable means that each local variable is just a partial or total copy of the global variable $z$. In the global variable consensus, we have $x_i = z$. In the other case each local vector only contains a small number of the global variables.

3.2.5 General form consensus optimization

The general form consensus problem is

$$\minimize \sum_{i=0}^{N} f_i(x) \quad \text{subject to} \quad x_i - z_i = 0 \quad i = 1...N,$$

with variables $x_1, ..., x_N$ and $z$. Intuitively, $z_i \in R^n$ is the fraction of the global variable $z$ that the local variable $x_i$ should be; It is a selection of the global state $z$ and the local variables $x_i$ can influence only these component of $z$ Then ADMM consists of iterations:

$$x_i^{k+1} = \min_{x_i} (f_i(x_i) + y_i^k x_i + (\rho/2)||x_i - z_i^k||^2_2)$$

$$z_i^{k+1} = \min_{z_i} \sum_{i=1}^{m} -y_i^k z_i + (\rho/2)||x_i^{k+1} - z_i^k||^2_2$$

$$y_i^{k+1} = y_i^k + \rho(x_i^{k+1} - z_i^{k+1}),$$

where the $x_i$ and $y_i$ updates can be carried out independently in parallel for each $i$. As in the global consensus case, the general form consensus problem can be generalized by allowing the global variable nodes to handle an objective term.
This section was written based on the work presented in [12]. For a more detailed discussion can then refer to this work.

F-Lipschitz is a new optimization method developed primarily to improve the performance of the calculation in wsn (wireless sensor network), where it is required fast optimization of the node’s variables with minimal exchange of information among them. Compared to traditional Lagrangian methods, the convergence time of centralized F-Lipschitz problems is better. These methods are computationally much more expensive, particularly for distributed optimization in wireless sensor networks. To compute the optimal solution for centralized optimization problems, F-Lipschitz algorithms do not require Lagrangian methods, but superlinear iterations based on a solution of a system of equations given by the constraints. Also distributed F-Lipschitz algorithms converge fast, respect to traditional Lagrangian method because they don’t require Lagrangian decomposition and parallelization methods, but simple asynchronous iterative methods.

F-Lipschitz optimization defines a class of optimization problems for which all the constraints, also the inequalities, are satisfied at equality when computed at the optimal solution. The most insidious part of the method is to formulate conditions, called *qualifying properties*, that ensure that this happens, before calculating the solution. The main disadvantage of this method is just in difficulty to check such conditions. The functional $f_0$ can be composed of a linear or non-linear, decomposable or a non-decomposable function.

An F-Lipschitz optimization problem is defined as:

$$\begin{align*}
\text{maximize} & \quad f_0(x) \\
\text{subject to} & \quad x_i \leq f_i(x), \quad i = i, \ldots, l \\
& \quad x_i = h_i(x), \quad i = l + 1, \ldots, n \\
& \quad x \in D,
\end{align*}$$

(3.53)

where $D \in \mathbb{R}^n$ is a non empty, convex and compact set. It is a box constraint. $l \leq n$; The objective function and constraints must be continuous differentiable functions such that:

$$\begin{align*}
f_0(x) & : D \rightarrow \mathbb{R}^m, \quad m \geq 1 \\
f_i(x) & : D \rightarrow \mathbb{R}, \quad i = 1, \ldots, l \\
h_i(x) & : D \rightarrow \mathbb{R}, \quad i = l + 1, \ldots, n
\end{align*}$$

(3.54)

and the following three properties are satisfied:
∇f₀(x) ≥ 0 (f₀(x) is strictly increasing),

and

∇j fᵢ(x) ≤ 0 ∀i ≠ j, ∀x ∈ D
∇j hᵢ(x) ≤ 0 ∀i ≠ j, ∀x ∈ D

or

∇i f₀(x) = ∇j f₀(x) ∀i ≠ j, ∀x ∈ D
∇fᵢ(x) ≥ 0 ∀i ≠ j, ∀x ∈ D
∇hᵢ(x) ≥ 0 ∀i ≠ j, ∀x ∈ D

and

|fᵢ(x) − fᵢ(y)| ≤ αᵢ||x − y||, i = 1, ..., l ∀x, y ∈ D
|hᵢ(x) − hᵢ(x)| ≤ αᵢ||x − y||, i = l + 1, ..., n ∀x, y ∈ D

with αᵢ ∈ [0, 1), ∀i.

All these properties are called qualifying properties of an F-Lipschitz optimization problem. They are sets of assumptions on f₀, f and D which ensure that the problem is Fast-Lipschitz.

The main theorem of F-Lipschitz optimization says that:
"Let problem 3.53 be weak, F-Lipschitz and feasible. Then, the problem has a Pareto optimal solution x, uniquely defined by:

\[ xᵢ^* = fᵢ(x^*) \quad ∀ \quad i = 1, ..., n. \] (3.56)

A Pareto optimal solution is a vector for which is impossible to improve one component without decreasing another component. Pareto efficiency, or Pareto optimality, is a concept in economics with applications in engineering. In a Pareto efficient allocation, no one can be made better off without making at least one individual worse off. Given an initial allocation of goods among a set of individuals, or a set of values in a vector, a change to a different allocation that makes at least one individual better off without making any other individual worse off is called a Pareto improvement. An allocation or a set of values, is defined as Pareto efficient or Pareto optimal when no further Pareto improvements can be made.

The optimal distribuite solutions can be found by iterating:

\[ xᵢ^{k+1} = fᵢ(x^{k}) \quad ∀ \quad i = 1, ..., l \]
\[ xᵢ^{k+1} = hᵢ(x^{k}) \quad ∀ \quad i = l + 1, ..., n \] (3.57)
Iterating this equation, every node \( i \) of the network collects asynchronously the decision variables \( x \) at time \( k \) and update its decision variable. In order to proceed the algorithm requires that the state variables \( x \) are to be exchanged between the nodes.

In the equations 3.57, \( k \) is an index associated with iterations. To obtain a specific precision \( \epsilon \) of the optimal solution \( x^* \), an upper bound to the number of iterations needed is:

\[
\bar{k} = \frac{ln|\epsilon| - ln|d|}{ln|\alpha|}
\]

(3.58)

where \( d = \max_{x,y \in D} ||x - y||_{max} \) and \( \alpha \) is the modulus of \([f(x)^T h(x)^T]^T\).
Chapter 4
Mathematical Modeling of problem

Active/reactive power dispatch problems are fundamental in the field of electrical engineering. It is usually formulated as an optimal power flow (OPF) problem. The OPF problem is an important class of problems in the power industry. The problem is to determine generators power set points so that the overall cost of power generation is minimized, while respecting limits on the generator’s capacity and transmission power flow constraints. The purpose of this chapter is to derive the flow model, which is widely used to characterize a power system’s behavior around the normal steady state operation and to formally formulate the OPF problem. We draw inspiration from the electrical model used in Matpower [13]. It is a tool for Matlab which was used in our work to get some examples of electrical networks used in our algorithm’s tests. The problem can be formulated as a minimization problem with a set of nonlinear algebraic equality and inequality constraints. These constraints represent both Kirchhoff’s laws and network and generator operation limits. The objective function is the total cost of real generation. The costs may be defined as polynomials or as piecewise-linear functions of generator output. In the formulation of the power flow problem, four variables are associated to each network node:

- $v_k$ Voltage magnitude of node $k$
- $\theta_k$ Voltage angle of node $k$
- $P_{kj}$ Active power flow from node $k$ to node $j$
- $Q_{kj}$ Reactive power flow from node $k$ to node $j$
Depending on which of the above four variables are known and which ones are unknown and have to be calculated, can be defined three basic types of nodes (or buses): PQ, PU, Uθ. In the first the variables are \( v_k \) and \( \theta_k \), in the second are \( Q_{kj} \) and \( \theta_k \) and in the last are \( P_{kj} \) and \( Q_{kj} \). The PQ buses are normally used to represent load buses without voltage control, while PU buses are used to represent generation buses with voltage control in power flow calculations. Finally, Uθ bus, also called reference bus or slack bus. It has double functions in the basic formulation of the power flow problem, it serves the roles of both a voltage angle reference and a real power slack. The voltage angle at the reference bus has a known value, but the real power generation is taken as unknown. The slack bus is selected to provide or take active and reactive power to or from the transmission line to provide losses since these are unknown till the final solution of the problem. The slack bus is the only bus at which the system reference phase angle is defined. From this, the various angular differences can be calculated in the power flow equations. In practice, the slack bus is the connection node of our network with other external power systems. In normal power systems the far most common are PQ-buses or load buses, while there is only one reference bus. To guarantee that the OPF can be solved, the slack bus is assigned a zero phase angle by setting its phase angle upper and lower limits to zero. The problem can be formulated schematically as:

\[
\begin{align*}
\text{minimize} & \quad \text{costs of active and reactive generation} \\
\text{subject to} & \quad \text{active power balance equations} \\
& \quad \text{reactive power balance equations} \\
& \quad \text{apparent power flow limit of line, from and to side} \\
& \quad \text{bus voltage limits} \\
& \quad \text{active and reactive power generation limits}
\end{align*}
\tag{4.1}
\]

where the variables are listed above.

Generator cost functions usually are represented as quadratic functions:

\[
C(P_G) = a + b P_G + c P_G^2,
\tag{4.2}
\]

where \( a, b, c \) are three positive constants. That means that the cost function of generators is convex. The convex quadratic cost functions make this OPF formulation a problem that can be solved with a quadratic programming (QP) algorithms. In Matpower Model, all transmission lines, transformers and phase shifters are modeled with a common branch model, consisting of a standard \( \pi \) transmission line model, with series impedance \( z_{ij} \) and total
charging capacitance $b_c$. In addition there is also an ideal phase shifting transformer (figure 4.1). The standard AC Optimal Power Flow problem involves the minimization of total variable generation costs subject to nonlinear balance, branch flow, and production constraints for real and reactive power. An AC power flow is modeled by using values for both resistance and reactance. In practice, AC OPF problems are typically approximated by a DC OPF problem that focuses only on real power constraints in linearized form [13]. The model considered, can admit the presence of a transformer in the line. In the case of our simplified model, transformers are not present. Therefore, we use transformation ratio $N = 1$ and phase shift angle $\theta_{\text{shift}} = 0$. 
In the electrical model, generators and loads are represented as in Figure 4.2. The general case for load current can be written as:

$$I_{\text{load}}^j = I_{\text{load}}^j(v_j) \quad (4.3)$$

where the function $I_{\text{load}}^j(v_j)$ describe the load characteristics. Generators are in load flow analysis modeled as current injections. In steady state a generator is commonly controlled so that the active power injected into the bus and the voltage at the generator terminals are kept constant. For a deeper analysis on transmission lines and networks, we refer the reader to [14]. The DC OPF power flow model assumes that only the angles of the complex bus voltages vary, and that the variation is small. Voltage magnitudes are assumed to be constant. Transmission lines are assumed to have no resistance, and therefore no losses. This is a reasonable first approximation for the real power system, which can be considered only slightly non-linear in normal steady state operation. Furthermore, we assume that in electricity markets the loads are usually inelastic, meaning that they do not change as much as the price changes. In other words, the function given in 4.3 is constant. When this is the case, the OPF objective is to minimize total generation cost. We restrict our self to the simplest model of the power supply to test the proposed distributed algorithm. Extensions of the work to complex models including more electrical components and additional constraints are left for future research.

The power system can be modeled as a directed graph. Consider a connected directed graph $G = (V, E)$ as an abstraction of an electrical power network (Figure 4.4). The system consists of $N$ buses. We assume that each bus has a local generator and a local load. In Figure 4.3 is shown a graphical representation used to refer the nodes. An edge from node $i$ to node $j$ is denoted as $e_{ij}$ and $z_{ij} = r_{ij} + jx_{ij}$ is the impedance of the transmission line corresponding to edge $e_{ij}$ (see Figure 4.4). Since $r_{ij}$ is often negligible
compared to $x_{ij}$, we assume that $r_{ij}=0$ in our DC flow model. In the power system analysis, power is defined as $\text{Re}(vi^*)$, where $v$ and $i$ represents the voltage and current, respectively. $\text{Re}(a)$ and $\text{Im}(a)$ is the real and imaginary part of a complex number $a$. Complex power $S$, reactive power $Q$ and active power $P$ are defined as

$$S = vi^* \quad Q = \text{Im}(vi^*) \quad P = \text{Re}(vi^*) .$$

Remembering the triangle of powers we have

$$S = P + jQ .$$

Let $S_{ij}$ denote the complex power flow from node $i$ to node $j$, and $v_i$ denote the generator voltage at node $i$. Magnitude-phase representation is given by:

$$v_i = |v_i|e^{j\theta_i}, \quad v_j = |v_j|e^{j\theta_j} ,$$

and active and reactive powers are given by

$$P_{ij} = -P_{ji} = \frac{|v_i||v_j|}{x_{ij}} \sin(\theta_i - \theta_j) ,$$

$$Q_{ij} = \frac{|v_i|^2}{x_{ij}} - \frac{|v_i||v_j|}{x_{ij}} \cos(\theta_i - \theta_j) ,$$

$$Q_{ji} = \frac{|v_j|^2}{x_{ji}} - \frac{|v_j||v_i|}{x_{ji}} \cos(\theta_i - \theta_j) .$$
The active power flow from node $i$ to node $j$ is mainly dependent on $\theta_i - \theta_j$, and the reactive power flow from node $i$ to node $j$ is mainly dependent on $|v_i| - |v_j|$. Under normal operating conditions, we have $|v_i| = |v_j|$, and $\theta_i - \theta_j$ is typically small. In this case, there is reasonably good decoupling between the control of active power flow $P_{ij}$, $P_{ji}$ and reactive power flow $Q_{ij}$, $Q_{ji}$.

In the DC flow model we further assume that only the voltage phases $\theta_i$, $\theta_j$ vary, and that variation is small. Voltage magnitudes $|v_i|$, $|v_j|$ are assumed to be constant ($|v_i|$, $|v_j|$ = 1 (p.u.)). (In the field of electrical engineering, a per-unit system (p.u.) is the expression of system quantities as fractions of a defined base unit quantity. The base voltage might be the nominal voltage of a bus as we used here, or nominal current or power). In this case, the reactive power flow $Q_{ij}$ is negligible, and we are only considering only the active power flow $P_{ij}$. It is a scalar value. Under the assumptions and simplifications above, the power flow from node $i$ to node $j$ is given by

$$P_{ij} = \frac{1}{x_{ij}}(\theta_i - \theta_j)$$ (4.9)

In our problem we consider constraints on the maximum flow in each edge. Therefore we need to write the equation above in a matrix form, obtaining the powers flow in each edge of the graph (in each transmission lines)

$$P_{edge} = F\theta.$$ (4.10)

Here, $\theta = (\theta_1, \ldots, \theta_n)$ and the matrix $F$ is constructed in such a way as to operate the phase difference between all pairs of nodes in the net, taking into account the impedance of the edge that connects them. In this way we obtain $P_{edge}$ which collects all the flows in all branches of the network. The total power flowing into bus $i$, $P_i$, must equal the power generated by generator $i$ minus the power absorbed by the local load $P_{Li}$. Therefore, $P_i$, must equal the sum of the power flowing away from bus $i$ on all transmission lines, i.e.,

$$P_i = \sum_j P_{ij}.$$ (4.11)

The equation 4.11 can be compactly expressed as

$$P = B\theta,$$ (4.12)

where $P = (P_1, ..., P_n)$, and the $i$th $j$th component of $B$ is given by

$$b_{ij} = \begin{cases} \sum_j (1/x_{ij}) & i = j \\ (-1/x_{ij}) & e_{ij} \in E \\ 0 & e_{ij} \notin E \end{cases}.$$ (4.13)
Here the matrix $B$ is singular. It can be thought as a weighted Laplacian matrix of the graph. Now, we can formulate the General OPF problem as follows:

$$\text{maximize } C(P_G) = \sum_{i=1}^{N} C_i(P_{Gi})$$

subject to

$$B\theta = P_G - P_L$$
$$P_{min} \leq F\theta \leq P_{max}$$
$$P_{Gmin} \leq P_G \leq P_{Gmax}$$
$$\theta_{imin} \leq \theta_i \leq \theta_{imax}$$

The variables are $P_G = (P_{G1}, ..., P_{GN}), \theta = (\theta_1, ..., \theta_N)$, and the problem data are $P_L = (P_{L1}, ..., P_{LN}), P_{min} = (P_{1min}, ..., P_{Nmin}), P_{max} = (P_{1max}, ..., P_{Nmax}), P_{Gmin} = (P_{G1min}, ..., P_{Gmin}), P_{Gmax} = (P_{G1max}, ..., P_{Gmax}), \theta_{min} = (\theta_{1min}, ..., \theta_{Nmin})$ and $\theta_{max} = (\theta_{1max}, ..., \theta_{Nmax})$. Here $P_{imin}, P_{imax}$ represent the lower and upper limits of the power flows on each transmission lines, $P_{Gimin}$ and $P_{Gimax}$ represent the minimum and maximum active powers output for generator $i$, likewise $P_L$ represent the local loads for node $i$ and $\theta_{imin}, \theta_{imax}$ represent the lower and upper phase limit of node $i$.

The General OPF problem seeks to find the optimal generated active power $P_G$ such that the total generation cost is minimized, subject to the power flow equation and physical constraints of the generation and transmission systems. The objective function of equation 4.14 represents the total generation cost of all the generators, and the cost for generator $i$ to generating $P_{Gi}$ unit of active power is usually in the form:

$$C_i(P_{Gi}) = a_i + b_i P_{Gi} + c_i P_{Gi}^2,$$

where $a_i$, $b_i$, and $c_i$ are positive coefficients.

In the next chapter we develop algorithms for solving the OPF problem 4.14.
Chapter 5

Solution Procedure

In this chapter we provide distributed algorithms for OPF problem 4.14 formulated in the previous chapter. We first summarize a centralized method that is used as a benchmark for our proposed algorithms. In particular, we start by briefly explaining the method used by matpower ISCED to solve the problem. We next provide distributed solution procedures for the OPF problem based on ADDM method. Specifically, we give two equivalent formulations of the original OPF problem 4.14. The first one results in a partially distributed algorithm where we need to rely on another entity who must take care of all the constraints of problem 4.14, including power flow balance equation and others. The second formulation gracefully resolve the coupling constraints of the original formulation by introducing new interface variables together with consistency constraints to guarantee the convergence of interface variables to their common net values (see Section 3.1). The result is a fully decentralized algorithm. Consequently, each node need to know only the neighbors’ s data and it is not required to know all the network data. Moreover, this proposed method requires only message passing among neighbors. Therefore, the algorithm is scalable and can be applied in the case of very large networks. Roughly speaking this means that the growth of the network does not increase the computational load of the individual nodes.
5.1 Centralized formulation

In this section, we briefly explain the method used by Matpower [13] to solve the problem in a centralized manner. Matpower uses four different algorithms for solving the power flow problem. The default solver is based on a standard Newton’s method. With this method typically five iterations are required for a solution. Memory and time requirements vary approximately in direct proportion to the problem size. We start from the problem 4.14 and rewrite it for clarity, i.e.,

\[
\begin{align*}
\text{minimize} \quad & C(P_G) = \sum_{i=1}^{N} C_i(P_G^i) \\
\text{subject to} \quad & B\theta = P_G - P_L \\
& P_{\text{min}} \leq F\theta \leq P_{\text{max}} \\
& P_{\text{Gmin}} \leq P_G \leq P_{\text{Gmax}} \\
& \theta_{\text{min}} \leq \theta_i \leq \theta_{\text{max}}.
\end{align*}
\]

where the variables are $P_G$ and $\theta$. For notation simplicity, we rewrite the optimization problem in standard form. In the standard form problem, we adopt the convention that the right-hand side of the inequality and equality constraints are zero. This can always be arranged by subtracting any nonzero right-hand side. To do this, we introduce new functions defined as follows:

\[
h(P_G, \theta) = B\theta - P_G + P_L
\]

\[
g(P_G, \theta) = \begin{bmatrix}
F\theta - P_{\text{max}} \\
P_G - P_{\text{Gmax}} \\
\theta - \theta_{\text{max}} \\
P_{\text{min}} - F\theta \\
P_{\text{Gmin}} - P_G \\
\theta_{\text{min}} - \theta
\end{bmatrix}
\]

and

\[
f(P_G, \theta) = C(P_G)
\]

where $f, g, h$ are convex and twice continuously differentiable. Now, problem 5.1 can be equivalent reformulated as

\[
\begin{align*}
\text{minimize} \quad & f(P_G, \theta) \\
\text{subject to} \quad & h(P_G, \theta) = 0 \\
& g(P_G, \theta) \leq 0,
\end{align*}
\]

where the variables are $P_G$ and $\theta$. The Lagrangian associated with problem 5.3 is given by:

\[
L(P_G, \theta) = f(P_G, \theta) + \mu^T h(P_G, \theta) + \lambda^T g(P_G, \theta)
\]
Where \( \mu \) and \( \lambda \) are vectors of the Lagrange multipliers associated with constraints. We also put \( z = [P_G, \theta, \mu, \lambda]^T \).

From this, according to optimization theory, the Kuhn-Tucker necessary conditions of optimality are:

\[
\begin{align*}
  f(P_G^*, \theta^*) &= 0 \\
  g(P_G^*, \theta^*) &\leq 0 \\
  h(P_G^*, \theta^*) &= 0 \\
  \lambda^* &\geq 0
\end{align*}
\]

\( (5.5) \)

where \( (P_G^*, \theta^*, \mu^*, \lambda^*) \) is the primal and dual optimal solution and \( \nabla_{P_G, \theta} \) represents the gradient of the accompanying function with respect to primal variables \( (P_G, \theta) \).

Interior-point methods solve the problem 5.3 (or the KKT conditions). The basic idea of method is to decompose the problem into a sequence of equality constrained problems, and then solve each of it by Newton’s Method. Among the convex optimization algorithms, linear equality constrained quadratic problems are the simplest. For these problems the KKT conditions are a set of linear equations, which can be solved analytically. Newton’s method is the next level in the hierarchy. We can think of Newton’s method as a technique for solving a linear equality constrained optimization problem, with twice differentiable objective, by reducing it to a sequence of linear equality constrained quadratic problems. Interior-point methods form the next level in the hierarchy: They solve an optimization problem with linear equality and inequality constraints by reducing it to a sequence of linear equality constrained problems. Many problems are already in the form 5.3, and satisfy the assumption that the objective and constraint functions are twice differentiable and many others problems can be easily reformulated in the required form. [15, Section 11].
5.2 Distributed formulation

In this section, we first consider a partial distributed algorithm implementation where we rely on a central controller to handle all the problem constraints. Then, we propose a fully distributed algorithm, where there is no a central controller to perform any variable updates. Unlike the partial distributed methods, all the algorithm variable update can be carried out in fully distributed fashion with communication among local entities only.

5.2.1 Partially distributed formulation

Each node has a generator that produces $P_G$ and a fixed load $P_L$. Moreover, recall that the variables for each node are $P_G$ and $\theta_i$. Thus for generator $i$, we have its (local) variables $(P_G, \theta_i)$. We denote by $x_i$ the local variables at generator $i$, i.e.,

$$x_i = (P_G, \theta_i)$$

Let us introduce a new variable $z$ where the components represents the global consensus of all the variables $(P_G, \theta_1, P_G, \theta_2, \ldots, P_G, \theta_N)$. Specifically, we let

$$z = (z_{P_1}, z_{\theta_1}, z_{P_2}, z_{\theta_2}, \ldots, z_{P_N}, z_{\theta_N}) = (P_{G_1}, \theta_{1}, P_{G_2}, \theta_{2}, \ldots, P_{G_N}, \theta_{N})$$

In other words, each component of each local variable corresponds to some global variable component of $z$ (see figure 5.1). For latter use, we let

$$z_i = (z_{P_i}, z_{\theta_i})$$

which corresponds to the global copies of local variables $(P_G, \theta_i)$. The vector $z_i$ can be easily obtained by using a selection matrix $E_{select}$ as follows:

$$z_i = E_{select}^i z$$

It is useful to define the linear transformation $s(z) = Pz$, where $P$ is a permutation matrix such that

$$s(z) = (z_{P_1}, \ldots, z_{P_N}, z_{\theta_1}, \ldots, z_{\theta_N}) = (z_P, z_\theta)$$

and

$$z_P = (z_{P_1}, \ldots, z_{P_N}) \quad \text{and} \quad z_\theta = (z_{\theta_1}, \ldots, z_{\theta_N})$$

Note that $s(z)$ and $z$ contain the same components, but grouped according to different orders. Now we are ready to reformulate the problem 4.14 For
Figure 5.1: Local objective terms are on the left; global variable components are on the right. Each edge in the bipartite graph is a consistency constraint, linking a local variable and a global variable component.

clarity let us rewrite the original problem:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{N} C_i(P_{G_i}) \\
\text{subject to} & \quad B\theta = P_G - P_L \\
& \quad P_{\text{min}} \leq F\theta \leq P_{\text{max}} \\
& \quad P_{G\text{min}} \leq P_G \leq P_{G\text{max}} \\
& \quad \theta_{\text{min}} \leq \theta \leq \theta_{\text{max}} ,
\end{align*}
\]

where the variables are \( P_G = (P_{G_1}, \ldots, P_{G_N}) \) and \( \theta = (\theta_1, \ldots, \theta_N) \). The problem above can be equivalently formulated as

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{N} C_i(P_{G_i}) \\
\text{subject to} & \quad B\theta = P_G - P_L \\
& \quad P_{\text{min}} \leq z\theta \leq P_{\text{max}} \\
& \quad P_{G\text{min}} \leq zP \leq P_{G\text{max}} \\
& \quad \theta_{\text{min}} \leq z\theta_i \leq \theta_{\text{max}} \\
& \quad (P_{G_i}, \theta_i) = (zP_i, z\theta_i), \quad i = 1, \ldots, N ,
\end{align*}
\]
where the variables are \( P_{Gi}, \theta_i, z_{Pi}, \) and \( z_{\theta_i}, i = 1, \ldots, N \). We can compactly express problem 5.9 as follows:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{N} C_i(P_{Gi}) \\
\text{subject to} & \quad [-C_g \ B] s(z) + P_L = 0 \\
& \quad P_{min} \leq [0 \ F] s(z) \leq P_{max} \\
& \quad x_i \leq s(z) \leq x_{max} \\
& \quad x_i = z_i, \ i = 1, \ldots, N ,
\end{align*}
\]

where the variables are \( x = (x_1, \ldots, x_N) \) and \( z \). Note that the objective function \( C_i(P_{Gi}) = f_i(x_i) \). The problem data have their usual meaning with

\[
\begin{align*}
x_{min} &= (P_{G1min}, \ldots, P_{Gmin}, \theta_{1min}, \ldots, \theta_{Nmin}) \\
x_{max} &= (P_{G1max}, \ldots, P_{Gmax}, \theta_{1max}, \ldots, \theta_{Nmax}).
\end{align*}
\]

Note that the matrix \( C_g \) of problem 5.10 above is the generation connected matrix and we have,

\[
C_g = I ,
\]

where \( I \in R^{N \times N} \) is the identity matrix in the problem 5.10 above. In general, \( C_g \) is defined as follow. If its \((i,j)th\) element is 1, the generator \( j \) is located at bus \( i \) and 0 otherwise. The matrix \( C_g \) is used to select only the generators active on the nodes. If all nodes have a generator, the matrix becomes an identity matrix. The last constraint of problem 5.10 is the consensus constraint, see Section 3.2.5, problem 3.45 and problem 3.49 Now we can express the problem in general consensus form (see section 3.2.5) as follows:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{N} f_i(x_i) + g(z) \\
\text{subject to} & \quad x_i = z_i, \ i = 1, \ldots, N ,
\end{align*}
\]

where the variables are \( x = (x_1, \ldots, x_N) \) and \( z \), and \( g \) is the indicator function of the set \( C \) defined as

\[
C = \left\{ z \mid [-C_g \ B] P z + P_L = 0, P_{min} \leq [0 \ F] P z \leq P_{max}, x_{min} \leq P z \leq x_{max} \right\}.
\]

By applying ADMM to problem 5.12, we obtain the following iterations:

\[
\begin{align*}
x_i^{k+1} &= \arg \min_{x_i} \left( f_i(x_i) + y_i^{kT}(x_i - z_i^k) + (\rho/2)||x_i - z_i^k||^2_2 \right) \\
z^{k+1} &= \arg \min_{z \in C} \sum_{i=1}^{N} -y_i^{kT} z_i + (\rho/2)||x_i^{k+1} - z_i||^2_2 \\
y_i^{k+1} &= y_i^k + \rho (x_i^{k+1} - z_i^{k+1})
\end{align*}
\]
As we can see from the equations above, in this implementation, the individual nodes calculate their own production taking into account the cost function, without considering constraints. This step can be carried out in parallel from all nodes of the network. The constraints involved instead, in updating the global variable $z$. It is updated taking into account the constraints and the state of all nodes. Finally, the dual variable $y$ is updated in a distributed fashion.

5.2.2 Distributed formulation

In this section we develop a new equivalent formulation of problem 5.1, in order to obtain fully distributed implementation among the nodes. The key idea is based by introducing interface or public variables to each nodes in the system to decouple the coupling constraints and by introducing consistency constraints to enforce the global agreement of the public variables. The global opinion of public variables are modeled by a (global) vector $z$. In this formulation, updating the global variable $z$ can be implemented in a distributed fashion, unlike the $z$ update in 5.13. Roughly speaking, the $z$ update is simply an averaging of the public variables of neighbors. Let us rewrite the original problem 5.1 for clarity.

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{N} C_i(P_{G_i}) \\
\text{subject to} & \quad B\theta = P_G - P_L \\
& \quad P_{min} \leq F\theta \leq P_{max} \\
& \quad P_{Gmin} \leq P_G \leq P_{Gmax} \\
& \quad \theta_{min} \leq \theta \leq \theta_{max} ,
\end{align*}
\]

(5.14)

where the variables are $P_G = (P_{G1}, \ldots, P_{GN})$ and $\theta = (\theta_1, \ldots, \theta_N)$. We show next, how to eliminate coupling of the problem above. Note that only the first two constraints of problem above are coupled via matrices $B$ and $F$. Recall that the matrices $B$ and $F$ are dependent on the structure of the network. See Chapter 4, relations given in 4.10, 4.12, and 4.13. For example, consider the network, composed by 4 nodes shown in figure 5.2. Here the matrix $B$ and $F$, can be written as:

\[
B = \begin{bmatrix}
b_{11} & b_{12} & b_{13} & 0 \\
b_{21} & b_{22} & 0 & b_{24} \\
b_{31} & 0 & b_{33} & b_{34} \\
0 & b_{42} & b_{43} & b_{44}
\end{bmatrix}
\]
Figure 5.2: Example network representation

\[ F = \begin{bmatrix} f_{11} & f_{12} & 0 & 0 \\ f_{21} & 0 & f_{23} & 0 \\ 0 & f_{32} & 0 & f_{34} \\ 0 & 0 & f_{43} & f_{44} \end{bmatrix} \]

Note that, each row, with index \( i \), contains a nonzero element in column \( j \), only if \( j \)th node is directly connected to node \( i \). This indicates that each node needs to know only the information of the neighbors and not of the entire network. To facilitate the decoupling of the problem let us now consider the decomposition structure of figure 5.2. Different variants of the decomposition structures are shown in figure 5.3 and figure 5.4. The subsystems here correspond to generator nodes and the arrows correspond to the coupling between the subsystems. We refer to the different arrow groups represented by different color as nets. Next, we introduce a global (scalar) variable \( \theta_n \) to each net, i.e., we have the global variable vector \( z \) given by

\[ z = (\theta_1, \ldots, \theta_N) \]

Moreover, we introduce \( p_i \) interface (scalar) variables to node \( i \), where \( p_i \) is the number of nets connected to subsystem \( i \). For example, each node in figure 5.2 has 3 interface (scalar) variables, because each has connections to three nets. We combine these interface variables of node \( i \) to the vector \( x_i \in \mathbb{R}^{p_i} \). Thus, node \( i \) has its variables \((P_{G_i}, x_i)\), where \( P_{G_i} \) is refereed to as the local variable and \( x_i \) is the interface variable. It is also useful to define a vector \( x \) that collects all the interface variables, i.e.,

\[ x = (x_1, \ldots, x_N) \]

(5.15)

Note that (scalar) components of the interface variable \( x \in \mathbb{R}^{\sum_{i=1}^{N} p_i} \) are related to (scalar) components of global variable \( z \), see figure 5.3). We clearly
see that the vector $x$ is easily obtained by multiplying $z$ by an appropriate matrix $E \in R^{\sum_{i=1}^{N} p_i \times N}$ as shown in section 3.1, equation 3.2, i.e.,

$$x = Ez$$

(5.16)

Moreover, we have

$$x_i = E_i z$$

(5.17)

for some appropriately chosen matrix $E_i \in R^{p_i \times N}$, see section 3.1. Now we can use the newly introduced interface variables $x$ and global variables $z$ to decouple the problem. In order to proceed, let us first consider the Kirchhoff’s equations, i.e., first constraint of problem 5.14. By using the interface variables and the global variables, we can decouple Kirchhoff’s equations equivalently as follows:

$$B_ix_i = P_{G_i} - P_{L_i} , \quad i = 1 \ldots, N$$

(5.18)

$$x_i = E_i z , \quad i = 1 \ldots, N$$

(5.19)

where the row vector $B_i \in R^{1 \times p_i}$ is constructed by using $i$th row of $B$ and removing the columns that corresponds to zero elements. It is worth noting that any node $i$ can construct $B_i$ from local information. For example, in the case of our example network (see figure 5.2), we have

$$B_1 = [b_{11} \ b_{12} \ b_{13}] \quad B_2 = [b_{21} \ b_{22} \ b_{24}]$$

(5.20)
Similar decomposition can be performed in the case of the second constraint of problem 5.14 as well. By using the interface variables and the global variables, we can equivalently express the second constraint as follows:

\[ P_{i_{\text{min}}} \leq F_i x_i \leq P_{i_{\text{max}}} , \quad i = 1 \ldots , N \]  \hspace{1cm} (5.22)

\[ x_i = E_i z , \quad i = 1 \ldots , N , \]  \hspace{1cm} (5.23)

where \( F_i \in \mathbb{R}^{1 \times p_i} \), can be constructed by using local information by node \( i \). For example, in our example network (see figure 5.2), we have

\[ F_1 = [f_{11} \ f_{12} \ 0] \quad F_2 = [0 \ f_{32} \ f_{34}] \]  \hspace{1cm} (5.24)

\[ F_3 = [f_{21} \ f_{23} \ 0] \quad F_4 = [0 \ f_{43} \ f_{44}] . \]  \hspace{1cm} (5.25)

By using 5.18, 5.19, 5.22, and 5.23, now we can equivalently express the

\[ B_3 = \begin{bmatrix} b_{31} & b_{33} & b_{34} \end{bmatrix} \quad B_4 = \begin{bmatrix} b_{42} & b_{43} & b_{44} \end{bmatrix} . \]  \hspace{1cm} (5.21)

Figure 5.4: Decomposition structure: Each subsystem needs to know is the value of the phase of its neighbors and not of the whole network only.
original problem 5.14 as follows:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{N} C_i(P_{G_i}) \\
\text{subject to} & \quad B_i x_i = P_{G_i} - P_{L_i}, \quad i = 1, \ldots, N \\
& \quad P_{l_{\text{min}}} \leq F_i x_i \leq P_{l_{\text{max}}}, \quad i = 1, \ldots, N \\
& \quad P_{G_i,\text{min}} \leq P_{G_i} \leq P_{G_i,\text{max}}, \quad i = 1, \ldots, N \\
& \quad x_{i_{\text{min}}} \leq x_i \leq x_{i_{\text{max}}}, \quad i = 1, \ldots, N \\
& \quad x_i = E_i z, \quad i = 1, \ldots, N,
\end{align*}
\]  

(5.26)

where the variables are \(P_{G_i}, x_i, i = 1, \ldots, N\) and \(z = (\theta_1, \ldots, \theta_N)\) and the problem data \(x_{i_{\text{min}}} \in R^{p_i}\) represents the lower phase limits of its neighbors, and \(x_{i_{\text{max}}} \in R^{p_i}\) represents the upper phase limits of its neighbors. Now we can apply ADMM to problem 5.26. Given the dual variables \(\{y^k_i \in R^{p_i}\}_i=1,...,N\) and the global variable \(z^k \in R^N\), the \(x\)-update in the \(k\)th iteration of ADMM is distributed among the nodes. In particular, \(x^{k+1}_i\) is obtained as a part of the solution of the following problem solved at node \(i\):

\[
\begin{align*}
\text{minimize} & \quad C_i(P_{G_i}) + y^k_i^T (x_i - E_i z^k) + \frac{(\rho/2)}{2} ||x_i - E_i z^k||^2_2 \\
\text{subject to} & \quad B_i x_i = P_{G_i} - P_{L_i} \\
& \quad P_{l_{\text{min}}} \leq F_i x_i \leq P_{l_{\text{max}}} \\
& \quad P_{G_i,\text{min}} \leq P_{G_i} \leq P_{G_i,\text{max}} \\
& \quad x_{i_{\text{min}}} \leq x_i \leq x_{i_{\text{max}}},
\end{align*}
\]  

(5.27)

where the variables are \((P_{G_i}, x_i)\). We denote by \((P^{k+1}_{G_i}, x^{k+1}_i)\) the solution of problem 5.27 in the \(k\)th iteration of ADMM.

Given the dual variables \(\{y^k_i \in R^{p_i}\}_i=1,...,N\) and the interface variable \(x^{k+1} \in R^{\sum_{i=1}^{N}p_i}\), the \(z\)-update in the \(k\)th iteration of ADMM is obtained as the solution of the unconstrained problem

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{N} y^k_i^T (x^{k+1}_i - E_i z) + \frac{(\rho/2)}{2} \sum_{i=1}^{N} ||x^{k+1}_i - E_i z||^2_2, \\
\end{align*}
\]  

(5.28)

where the variable is \(z = (\theta_1, \ldots, \theta_N)\). We denote by \(z^{k+1}\) the solution. Note that \(z^{k+1}\) can be expressed in closed form as

\[
\begin{align*}
z^{k+1} &= (\sum_{i=1}^{N} E_i^T E_i)^{-1} \sum_{i=1}^{N} E_i^T (x^{k+1}_i + (1/\rho)y^k_i) \\
&= (E^T E)^{-1} \sum_{i=1}^{N} E_i^T (x^{k+1}_i + (1/\rho)y^k_i) \\
&= \text{diag}((1/n_1), \ldots, (1/n_N)) \sum_{i=1}^{N} E_i^T (x^{k+1}_i + (1/\rho)y^k_i).
\end{align*}
\]  

(5.29-5.31)

The relation (5.29) follows by making the gradient of problem 5.28 objective function equals to 0, (5.30) follows from (3.3), and (5.30) follows from the properties of \(E\), where \(n_i\) represents the number of subsystems associated
to net $i$. For example, in our 4-node example (see figure 5.2), we have $n_i = 3$, $i = 1, 2, 3, 4$. If we carefully analyze (5.31), we note that $z$-update is simply obtained as follows: each subsystem $i$ update its global net value $\theta_i$ simply by averaging $n_i$ scalars, where each scalar is the sum of net $i$’s interface variables and scaled versions of dual variables. This is interesting, because $z$-update can be implemented in a distributed fashion.

Given the previous iteration dual variables $\{y^k_i\}_{i=1,...,N}$, the interface variable $x^{k+1} \in R^{\sum_{i=1}^N p_i}$ and the global variable $z^{k+1}$, $y$-update or the dual variable update in the $k$th iteration of ADMM is as follows:

$$y^{k+1}_i := y^k_i + \rho (x_i - E_i z^{k+1}) .$$

Note that the iteration above can also be implemented in a distributed fashion.

The main advantage as we will see in the next chapter is that the formulation above is scalable, because $x$, $y$, and $z$ updates are not affected by the size of the network. Indeed, these updates rely only on the information from adjacent nodes.

45
Chapter 6

Numerical results

In this chapter, we numerically illustrate the behavior of solution methods discussed in section 5.1, 5.2.1, and 5.2.2 by implementing them. All the simulations have been carried out on a personal computer with the following specifications: Windows 7 Enterprise, CPU Intel (R) i7-2600@ 3.4 GHz and 8 GB of RAM. Networks and related data were inspired from the sample data of the simulator Matpower[13]. First we show the results obtained on the network composed of 4 nodes, shown in figure 6.1. Results obtained on larger networks are provided later in this section.

Figure 6.1: 4-node network topology
Figure 6.2: Total objective value versus ADMM iteration
Figure 6.3: $G_1$ output power (Watts) versus ADMM iterations
Figure 6.4: $\theta_3$ (degrees) versus ADMM iterations
After, we shows the evolution of the objective value of problem 5.1 in the case of partial distributed algorithm (figure 6.2(a)) and fully distributed algorithm (figure 6.2(b)). Here the X-axis of the figures represent ADMM iterations. Different curves in each figure correspond to different values. We note that the behavior of two algorithms are very similar and both algorithms converges to the optimal objective value found by Matpower. Results show that lager $\rho$ values give faster convergence compared to smaller $\rho$ values.

Figure 6.3 shows the evolution of the output $P_{G_1}$ of the generator 1 in the case of partial distributed algorithm (figure 6.3(a)) and fully distributed algorithm (figure 6.3(b)). Results show, that the value of $\rho$ greatly influences the speed of convergence. For high values of $\rho$, we get a much faster convergence. Even though, we have not shown exhaustively, for the generators of other nodes, i.e., node 1, 2, 3, the behavior is almost the same.

Figure 6.4 shows the evolution of the phase 3, (i.e., $\theta_3$) in the case of partial distributed algorithm (figure 6.4(a)) and fully distributed algorithm (figure 6.4(b)). The influence of $\rho$ on the convergence can be seen in the case of phases as well. Note that the $\theta_3$ convergence of the proposed algorithms is different from the value given by Matpower centralized method. This behavior is not surprising, because there can be multiple solutions for phases. In other words, the objective function of problem 5.1 is not strictly convex in $\theta$, which is an indication of existence of multiple solutions for $\theta$. Figure 6.5 shows all the generators’output power for all the node. The rate of convergence is the same for each node.
Figure 6.6: Overall consensus error versus ADMM iterations
Figure 6.7: Consensus error in fully distributed formulation versus ADMM iterations
Figure 6.8: Error percentage of the power respect benchmark versus ADMM iterations.
Figures 6.6(a), 6.6(b) show the trend of consensus error. In particular we define the overall error as follows in the case of semi distributed algorithm

\[ e_1 = \sum_i ||x_i - z_i||_2^2 \]

where, \(x_i\) and \(z_i\) are respectively the local copy for node \(i\) and the global variable (see section 5.2.2). For the fully distributed case instead, we define the overall error as

\[ e_2 = \sum_i ||\theta_i - z_{\theta_i}||_2^2 \]

It represents the difference between the local copy and the average of all copy of that variable (see equation 3.45). We show again the consensus error in figure 6.7(a), where is plotted the error trend for each local copy of the phases variables. Figure 6.7(b) shows the error of the three copy of the variable \(\theta_3\) used by node 3 and its neighbors. The error is given by:

\[ e_3 = ||y_i - \theta_2||_2^2 \]

where \(y_i\) is the local copy of \(\theta_2\) at either subsystem \(i = 1\) or \(2\) or \(3\).

Figures 6.8(a) and 6.8(b) plot the error % of the power respect to the benchmark given by Matpower [13]. This error is given by:

\[ e_4 = \sum_i 100\frac{||P_i - P_{benchmark}||}{P_{benchmark}} \]

Again we can see how the value of \(\rho\) influences the error.
Figure 6.9: Topology of other networks used
Figure 6.10: Total objective function versus ADMM iterations

(a) 9 Nodes

(b) 14 Nodes
Table 6.1: Time needed for 200 iteration, varying number of node (seconds)

<table>
<thead>
<tr>
<th>variable</th>
<th>4 nodes</th>
<th>9 nodes</th>
<th>14 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>x update</td>
<td>49,55</td>
<td>49,41</td>
<td>45,96</td>
</tr>
<tr>
<td>z update</td>
<td>0,01</td>
<td>0,04</td>
<td>0,15</td>
</tr>
<tr>
<td>y update</td>
<td>0,01</td>
<td>0,00</td>
<td>0,00</td>
</tr>
</tbody>
</table>

To get inside into scalability of the fully distributed algorithm, we consider also two other network, composed respectively of 9 and 14 nodes. In the figures 6.10 is possible to see the trends of functional for these new networks varying the parameter $\rho$. The results again shows fast convergence properties for larger $\rho$.

Looking on the table 6.1 we can compare the time required to perform a given number of iterations. We see that the execution time is independent from the number of nodes. The results also show that the time for updating the $x$ the dual variable $y$, and $z$ at each node is independent of the number of nodes. Therefore, we can see the scalability of the proposed fully distributed algorithm. The growth of the size of the network is not important for individual nodes, because they only need to exchange messages with their neighbors.
Chapter 7

Conclusion and future work

In this work, we created a new formulation for the DC-OPF problem. The main advantages of this new formulation are mainly scalability and fast convergence. Here was done a basic implementation of this method. There are numerous ways to continue and extend this work. For example, following the same steps as those proposed in the thesis, a more general AC-OPF problem in this distributed formulation can be investigated. Is of relevant interest how the algorithm works with very large networks (hundreds of nodes), because in these networks the distributed approach to the solution of the optimization problem better shows its advantages. Moreover, would be useful to create an appropriate framework for testing large networks. Is also possible to continue studying this problem by including variable load in the functional or testing new cost functions closer to the real cost of electric generators, such as semi convex or piecewise functions. In general is possible to extend or change the objective function to include new costs related to social welfare or customer benefits.
Bibliography


