Energy Management and Peer-to-Peer Trading in the Future Smart Grid

J.A. Logeman

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Energy Management and Peer-to-Peer Trading in the Future Smart Grid

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

J.A. Logeman

March 11, 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Climate change is one of the biggest challenges of this time. One of the necessary changes to combat this challenge is a shift from conventional power resources, such as coal and natural gas, to large scale deployment of renewable resources, such as wind and solar power. However, renewable energy resources are volatile because they depend on the weather, which poses a problem because supply and demand should always be exactly matched. Along with a shift in energy production, also appliances that use non renewable energy are becoming electric, e.g., electric heating and cooling in buildings and electric vehicles (EVs). This causes an increase in electricity demand and is a burden on the main grid. To reduce the demand on the grid, it is important that a part of the energy is produced locally by distributed renewable resources, such as residential solar panels. To maintain a stable operation of the grid, the smart grid paradigm has to be adopted. Advancements in information and communication technologies provide the means for an efficient management of the grid using various services, such as distributed generation and storage.

Because of the conflicting nature of the agents in an electricity network, game theory is a widely used framework to tackle this problem. In a realistic model of a power grid coupling constraints must be taken into account, because the agents have shared resources. Therefore, the economic dispatch problem is modeled as a noncooperative game with coupling constraints, called a generalized game. Three distributed algorithms are derived to reach a generalized Nash equilibrium (GNE). It is shown through simulations that these algorithms have the potential to ensure a safe and efficient operation of the grid, while minimizing the cost of power usage. To reduce the cost even further, a method is presented to make more optimal use of the storage units.

Table of Contents

1 Introduction **1**

1-1 Motivation . . . 1

1-2 Research objective . . . 2

1-3 Outline . . . 2

2 Electricity markets **3**

2-1 Current electricity market . . . 3

2-1-1 Forward market . . . 4

2-1-2 Day-ahead market . . . 4

2-1-3 Intraday market . . . 4

2-1-4 Balancing market . . . 5

2-2 Future electricity market . . . 5

2-2-1 Smart grid . . . 5

2-2-2 Market organization . . . 6

3 Mathematical background **9**

3-1 Game theory . . . 9

3-1-1 Noncooperative game theory . . . 9

3-1-2 Cooperative game theory . . . 11

3-2 Generalized Nash equilibrium problems and variational inequalities . . . 12

3-3 Stochastic generalized Nash equilibrium problems . . . 14

4 Problem formulation **17**

4-1 Notation . . . 17

4-2 Electricity market model . . . 17

4-2-1 Cost functions and constraints . . . 18

4-3 Economic dispatch problem . . . 21

5 Distributed algorithms **23**
5-1 Main algorithm derivation . . . 23
5-1-1 Consensus constraint . . . 25
5-2 Preconditioned forward-backward operator splitting . . . 26
5-3 Forward-backward-forward operator splitting . . . 27
5-4 Forward-backward-half-forward operator splitting . . . 28
5-5 Stochastic algorithms . . . 29

6 Simulation and results **33**
6-1 Demand data . . . 33
6-2 Main results . . . 34
6-2-1 Total load vs. dispatchable generation and import from main grid . . . 34
6-2-2 Stopping criteria . . . 36
6-3 Incidence matrix vs. Laplacian matrix . . . 38
6-4 Scalability . . . 39
6-5 Stochastic demand . . . 40
6-6 Storage constraints . . . 43
6-6-1 Hard constraint . . . 43
6-6-2 Soft constraint . . . 44
6-6-3 Forecast . . . 46

7 Discussion **49**

8 Conclusion and recommendations **51**
8-1 Summary and conclusions . . . 51
8-1-1 Main results . . . 51
8-1-2 Scalability . . . 52
8-1-3 Stochastics . . . 52
8-1-4 Storage constraint . . . 52
8-2 Recommendations . . . 52

A Simulation values **55**
A-1 Values cost functions . . . 55
A-2 Step sizes . . . 55

Bibliography **57**

Glossary **61**
List of Acronyms . . . 61
List of Symbols . . . 62

List of Figures

2-1 Overview of the current power grid [7] 4

2-2 Overview of smart grid paradigm [12] 6

6-1 Unscaled load profiles from the OpenEI database 34

6-2 Total power imported from the main grid and produced by dispatchable generation units vs. total load in a 10-agent network using the pFB algorithm 35

6-3 Total power imported from the main grid and produced by dispatchable generation units vs. total load in a 10-agent network using the FBF algorithm 35

6-4 Total power imported from the main grid and produced by dispatchable generation units vs. total load in a 10-agent network using the FBHF algorithm 36

6-5 Primal residual for each scenario for all three algorithms 37

6-6 Dual residual for each scenario for all three algorithms 37

6-7 Disagreement on dual variable λ for each scenario for all three algorithms 38

6-8 Number of iterations and elapsed cpu time vs. the level of connectivity of the communication graph $\mathcal{G}$ (achieved on Matlab R2020a with a 1.8 GHz Intel Core i7 and 16 GB DDR4 RAM) 39

6-9 Number of iterations and elapsed cpu time vs. the amount of agents in the network (achieved on Matlab R2020a with a 1.8 GHz Intel Core i7 and 16 GB DDR4 RAM) 40

6-10 Primal residual for each scenario for all three algorithms for the stochastic case . 41

6-11 Dual residual for each scenario for all three algorithms for the stochastic case . . 42

6-12 Disagreement on dual variable λ for each scenario for all three algorithms for the stochastic case 42

6-13 Primal and dual residual of the deterministic case vs. the stochastic case for the FBF algorithm with four agents with storage 43

6-14 Primal and dual residual and disagreement on the dual variable λ when using a uniform distribution for all three algorithms for the scenario where all agents have storage units 44

6-15 Demand profile used for the simulations 45

6-16 Total cost and average state of charge per day when using a hard and soft constraint compared to using no constraint 45
6-17 Overview of using a prediction horizon of 2 days and control horizon of 1 day . . 46
6-18 Comparison between a prediction horizon of 1 and 2 days for all three algorithms 47

A-1 Primal and dual residual and disagreement on λ when using $\delta_i = 0.1$ instead of $\delta = 0.05,\ \forall i \in \mathcal{N}$ for the FBF algorithm when none of the agents have storage units 56

List of Tables

6-1 Total cost per algorithm for each scenario 36
6-2 Total cost when using a hard and soft constraint compared to using no constraint for the preconditioned forward-backward (pFB) algorithm 45
6-3 Total cost when using a prediction horizon of two days compared to a prediction horizon of one day 47

Chapter 1

Introduction

1-1 Motivation

Environmental concerns over the past years have asked for a change in the way our energy is produced. The past five years were the hottest years ever recorded and average sea levels have been rising at a rate roughly twice as fast as the long-term trend since recording began in 1880 [1]. It has become ever more evident that a paradigm shift from conventional energy resources, such as coal and natural gas, to renewable energy resources, such as solar and wind power, is necessary. Not only do renewable energy resources cause less pollution, they also reduce the burden on the main grid by producing a proportion of the demand locally [2].

In the EU, buildings are responsible for approximately 40% of energy consumption and 36% of greenhouse gas emissions [3], so a lot of improvement can be made in this sector. Large scale deployment of renewable energy resources can be realized in a distributive manner with the increase in small scale renewable energy generators, such as residential solar panels. These distributed energy resources (DERs) invite consumers to become prosumers: people who consume and produce energy. However, renewable energy resources are dependent on the weather which makes them volatile. Because supply and demand should always be exactly matched in an electricity network, the volatile nature of renewable energy resources poses a challenge. Besides the volatility of renewable energy production, another problem is the growing load on the grid. This is caused by the increase in electric appliances, such as electric vehicles (EVs) and electric heating and cooling systems in buildings.

With an increasing amount of DERs, along with advancements in information and communication technologies, the need for a central entity to regulate the electricity market decreases. A power network that contains intelligent nodes that can operate, communicate, and interact autonomously to efficiently deliver power and electricity to their consumers is called a smart grid [4]. In a smart grid, prosumers could have the possibility to trade their lack or excess of energy directly with each other and to schedule the demand of appliances to reduce the peaks in energy demand. Another promising technology to ensure a stable operation of the grid is the use of storage devices. With these new technologies gaining ground, the question arises

how to safely and efficiently distribute and use power in a smart grid. In the literature about future power grids different methods have been proposed how the grid should be managed. However, an efficient and reliable algorithm to allocate the resources in a smart grid has yet to be found. For a survey on methods found in literature the reader is referred to [5].

1-2 Research objective

Since the agents in a power network are assumed to be self-interested parties, game theory is a widely used framework to obtain a market equilibrium. An equilibrium state in a noncooperative game is called a Nash equilibrium. A special kind of noncooperative game is one where the agents have some shared resources, which is called a generalized (noncooperative) game. In a generalized game the strategy set of each agent is dependent on the decisions of the other agents, which means the shared constraint couples the decisions of the agents. Decentralized algorithms for Nash equilibrium seeking in generalized noncooperative networked games have gained high interest recently. These newly developed algorithms provide the means for efficient and robust energy management mechanisms [6]. However, they have not yet been tested in an energy market setting. This work aims at using recently developed distributed algorithms for generalized games on a resource allocation problem to obtain a stable operation of the grid and minimize the cost of power usage, while respecting the shared constraints.

1-3 Outline

First, Chapter 2 gives a short introduction on how the electricity market of today is organised and how the future grid is envisioned. Chapter 3 gives the mathematical background from which the algorithms in this work are derived. First, the basic idea of game theory is explained, after which the more specific case of a generalized noncooperative networked game with coupling constraints is considered. Then, in Chapter 4 the electricity market model on which the algorithms will be tested is derived. Chapter 5 gives the derivation of the proposed algorithms and in Chapter 6 these algorithms are used in simulations on the market model derived in Chapter 4. Finally, Chapter 7 gives the discussion and Chapter 8 concludes this work and gives some recommendations for future work.

Chapter 2

Electricity markets

This chapter gives a short introduction on electricity markets. First, the operation of the current electricity market will be explained in Section 2-1. Then, in Section 2-2 the electricity market of the future will be discussed.

2-1 Current electricity market

The electricity system consists of two parts: a physical infrastructure and an organised electricity market where electricity is traded [7]. The physical infrastructure consists of generators and a transport system. The transport system is usually divided in transmission lines for long distance transport, and a distribution network for electricity delivery to consumers. The electricity market consists of electricity suppliers, consumers, transmission system operators (TSOs), distribution system operators (DSOs) and regulators. The TSO is responsible for the transportation of electricity over long-distance power lines, while the DSO is responsible for the distribution of electricity to the customers [7]. The DSO also installs electricity meters at customers and communicates the consumption to the electricity providers. Unlike other commodities, such as coal, electricity cannot (yet) be stored at large scale. This means that supply and demand of electricity should always be balanced. Balancing of the grid is the responsibility of the TSO and balance responsible parties. In Figure 2-1 a basic overview of the current power grid is shown.

In general, electricity markets are divided in four different types, depending on the time period before delivery [8]. These are the forward market, the day-ahead market, the intraday market and the balancing market. Beside these four markets, there also exist long-term contracts, which can cover up to 20 years or more [7]. In what follows a brief description of each market will be given. How these markets are organised in detail differs from country to country, but the next four sections will give a general idea of how these markets work.

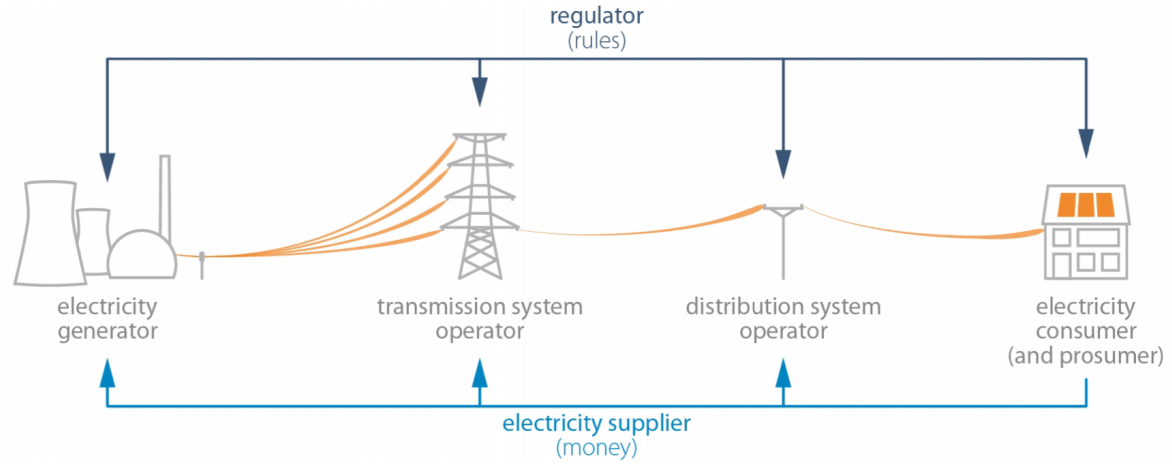


Figure 2-1: Overview of the current power grid [7]

2-1-1 Forward market

On the forward market contracts are traded to buy electricity for days, weeks or years in advance. With these contracts a certain amount of power is guaranteed to be available in the future for the buyer. These types of contracts are generally for large amounts of electricity and cover most of the electricity that is traded [8].

2-1-2 Day-ahead market

On the day-ahead market, contracts are traded for a time period of 24 hours ahead. These are power contracts for physical delivery at each hour for the next day's 24 hours. At this point in time a more accurate forecast of the demand profile of the next day is available, so the amount of power that is bought for the next day is a more accurate representation of the power that is actually going to be used. The day-ahead market works with an auction mechanism, where suppliers place bids on their supply and buyers place bids to purchase electrical energy. Bids to supply power are ranked in ascending order and bids to buy power are ranked in descending order. The intersection of supply and demand determines the marginal price for the entire region covered by the electricity pool [9].

2-1-3 Intraday market

On the intraday market, the expected demand for which energy is bought on the forward and day-ahead market is further refined due to better forecasts. On this market power products can be continuously traded in hourly intervals as well as freely definable blocks up to 5 minutes prior to delivery [10]. Instead of the auction mechanism used in the day-ahead market where the clearing price is the price for all transactions, the intraday market uses a pay-as-bid process. This means that the same product can have different prices depending on the time the trade was made. With an increase in volatile energy resources, such as wind and solar power, the intraday market is becoming increasingly important [9].

2-1-4 Balancing market

A mismatch between supply and demand causes the frequency of the grid to change, where excess supply causes an increase in frequency, and excess demand causes a decrease in frequency. In Europe the standard frequency of the grid is 50 Hertz. A change in this frequency can cause damage to assets or even power outages. Because it is impossible to exactly predict how much power is needed at each point in the future, a fourth market is needed, which is called the balancing market. Because balancing is used to keep the frequency within a narrow bound, it is also called frequency control. To ensure a balanced operation of the grid in this last stage, flexible generators are used to meet the peaks in demand in real-time. This short term balancing of supply and demand is done in three stages: primary reserves, secondary reserves and tertiary reserves. The difference between these three reserves is the time period within which they are activated. If there is a mismatch between supply and demand, the primary reserves can be activated within seconds. If the mismatch still exists, secondary reserves can be used to get the supply to meet the demand again. This secondary control can be activated for one, up to several minutes. The tertiary reserves are used if there still exists a deviation from the nominal value and can be activated for several minutes up to several hours.

2-2 Future electricity market

In the current power system there are only a few large power supplying companies. With an increase in renewable energy sources, such as wind farms, and distributed energy generation, such as residential solar panels, the volatility of energy supply increases. Also, with an increasing amount of demanding electric appliances, such as electric vehicles (EVs) and electrical heating and cooling systems, the demand of electricity increases. This increasing volatility and demand ask for more intelligent solutions in order to ensure a safe and reliable operation of the grid. One way of achieving this is to liberalize the electricity markets and use a bottom-up approach which allows for a more proactive role for consumers.

2-2-1 Smart grid

Allowing consumers to have a more active role in the power system asks for more intelligent solutions in the grid. A grid that has intelligent nodes is often referred to as a smart grid. In [4] a smart grid is defined as a power network composed of intelligent nodes that can operate, communicate and interact autonomously to efficiently deliver power and electricity to their consumers. Another concept that is often seen in power grid literature is that of a microgrid. According to [11] a microgrid is a single, controllable, independent power system that consists of distributed energy resources (DERs), loads, energy storage, and control devices. In a microgrid DERs and storage devices are directly connected to the users. A microgrid can be connected to the main grid or operate in an islanded mode. A microgrid with smart grid characteristics allows for an autonomous and efficient power network. Figure 2-2 shows an illustration of the smart grid paradigm.

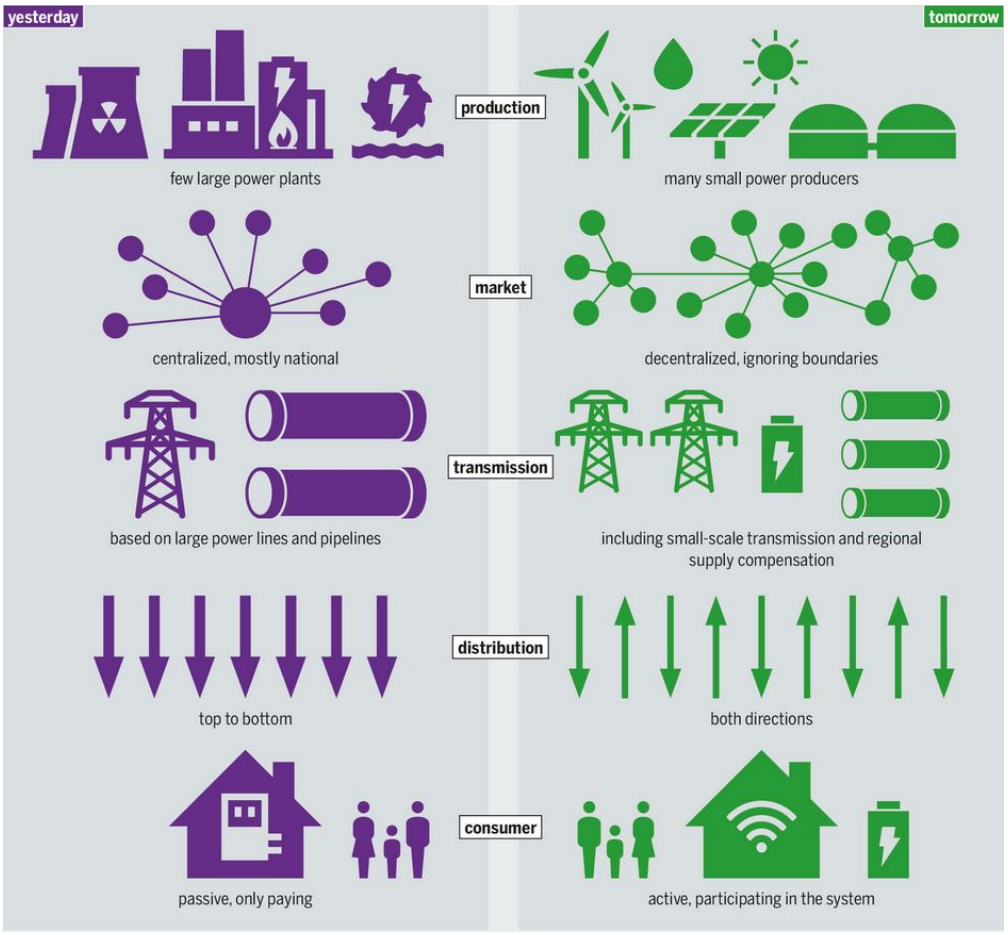


Figure 2-2: Overview of smart grid paradigm [12]

In the literature about future electricity markets there are two main methods to provide a stable and reliable operation of a microgrid: demand management and energy storage. Demand management involves scheduling the use of appliances in order to reduce peaks in the total energy demand and to reduce cost. However, as not all appliances can be shifted to a time with little demand (e.g. because a consumer wants to use that appliance at a certain time or the appliance requires a continuous operation), shifting demand does not solve the problem of the intermittent nature of renewable energy resources completely. Another way to reduce peaks in the demand is the use of storage devices. Distributed energy storage in the form of batteries is still rather expensive, but as technology advances, the cost of these storage units will decrease and the economical services they can provide will increase [13].

2-2-2 Market organization

Besides the physical infrastructure and assets in a microgrid, the grid also needs an organized market where power can be traded. The degree of decentralization in these markets is an aspect where the papers in literature differ. Some papers consider a fully decentralized market structure which allows full autonomy of the agents in the system. This means that all prosumers that participate in the network can trade electricity directly with each other and

the main grid, without the need of a central entity. Also, the relevant control signals needed for the optimization process are shared directly between the agents. Other papers consider a distributed market structure where a central coordinator is in charge of the trading process. The preference goes to a full peer-to-peer (P2P) market, because it lacks the need of a central entity and allows for more autonomy of the agents. To achieve a full P2P market a distributed algorithm is necessary, as opposed to semi-decentralized algorithms were a central node is necessary to share all relevant control signals.

Chapter 3

Mathematical background

This chapter introduces the mathematical concepts that are used in the rest of this thesis. First, the basic idea of game theory is explained in Section 3-1. Game theory consists of two main branches, which are discussed in Section 3-1-1 and Section 3-1-2. Then, in Section 3-2 the specific case of a noncooperative game is discussed where the agents in the game have some shared resources. The theory presented in this section is what forms the basis for the algorithms presented in the rest of this work. Finally, where in Section 3-2 it is assumed the problem is deterministic, Section 3-3 discusses the case when there are uncertainties in the system.

3-1 Game theory

According to [14] game theory can be defined as: "The study of mathematical conflicts and cooperation between rational intelligent decision-makers." It provides a framework to analyze situations in which two or more agents have to make decisions that influence each others welfare [14]. Because of the conflicting interests of the agents that perform P2P trading or demand management, game theory provides a very effective tool for the decision-making processes. There are two main branches within game theory: noncooperative game theory and cooperative game theory. These concepts are explained in more detail in Section 3-1-1 and Section 3-1-2, respectively.

3-1-1 Noncooperative game theory

Noncooperative game theory can be used to analyze the decision making process of independent players with partially or totally conflicting interests. It captures the distributed decision-making process that allows the players to optimize their objective functions coupled through the actions of the involved players, without any coordination or communication [4]. Noncooperative does not necessarily mean that the players do not cooperate; it rather means that cooperation must be self-enforcing and cannot be the result of any communication or

coordination of strategic choices among the players [15]. There are two types of noncooperative games: static games and dynamic games. A static game is a game in which time or information does not affect the choices of the players. It can be seen as a one-shot process in which players only make a decision once [15]. Even when the decisions of the players are taken at different points in time, without them having any information about the choices of the others, the game is considered static. A dynamic game is a game in which players have some form of information about each others' choices. In such a game the agents can act more than once and time has a central role in the decision making [15]. With some additional information, dynamic games can be formulated as static games. A popular representation for describing a noncooperative static or dynamic game is the notion of a strategic (or normal) form. Following [15, Definition 3.1] a strategic game is defined as $G = (\mathcal{N}, (\mathcal{S}_i)_{i\in\mathcal{N}}, (u_i)_{i\in\mathcal{N}})$, where:

- $\mathcal{N}$ is a finite set of players, i.e., $\mathcal{N} = \{1, \ldots, N\}$.
- $\mathcal{S}_i$ is the set of available strategies for player i.
- $u_i : \mathcal{S} \to \mathbb{R}$ is the utility (payoff) function for player i, with $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_i \times \cdots \times \mathcal{S}_N$ (Cartesian product of the strategy sets).

In such a game, each player wants to choose an action $s_i \in \mathcal{S}_i$ that optimizes its utility function $u_i(s_i, \boldsymbol{s}_{-i})$. This utility function does not only depend on the strategy of that player, s_i, but also on the strategies of all other players than player i, denoted by $\boldsymbol{s}_{-i}$. Note that in a static game, since no information is available, the notion of action and strategy mean the same thing. In a dynamic game however, one has to distinguish between the two. A strategy can be seen as a mapping from the information available to the player to the action set of this player [15]. There are games in which the strategy choices are made in a deterministic matter, which are called pure strategies, and games in which the strategies follow a probability distribution over the action sets, which are called mixed strategies.

The most accepted solution concept for noncooperative games is the so-called Nash equilibrium. A Nash equilibrium is a state of a noncooperative game where no player can improve its utility by changing its own strategy, if the other players maintain their current strategy [15]. When dealing with pure strategies, the Nash equilibrium can be defined as in [15]: A pure-strategy Nash equilibrium of a noncooperative game $G = (\mathcal{N}, (\mathcal{S}_i)_{i\in\mathcal{N}}, (u_i)_{i\in\mathcal{N}})$ is a strategy profile $s^* \in \mathcal{S}$ such that $\forall i \in \mathcal{N}$ the following holds:

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, \boldsymbol{s}_{-i}^*), \quad \forall s_i \in \mathcal{S}_i. \tag{3-1}$$

The main advantage of this solution concept is that it characterizes a stable state of a noncooperative game, which can often be reached by the players in a distributive manner and with little coordination [4]. However, there are also some drawbacks. A noncooperative game can have multiple Nash equilibria or none at all. Selecting a desirable Nash equilibrium from multiple solutions is a challenging topic. Also, a Nash equilibrium is not necessarily the best outcome in terms of payoff. For examples of noncooperative games and Nash equilibria the reader is referred to [15, Chapter 3].

3-1-2 Cooperative game theory

While noncooperative game theory studies the behaviour of competing players that cannot coordinate their decisions or communicate with one another, cooperative game theory provides analytical tools to study the behaviour of players when they cooperate [15]. Cooperative game theory has two main branches: bargaining theory and coalitional games. Bargaining theory studies the bargaining process among players to determine the terms of the cooperation. Coalitional games describe the formation of cooperating groups, called coalitions, which can strengthen the position of the players in a game. Coalitional games prove to be a very powerful tool for designing fair, robust, practical and efficient cooperation strategies in communication networks. A coalitional game is defined by the pair $(\mathcal{N}, v)$, where $\mathcal{N}$ is the set of players in a coalitional game and v is a mapping that determines the payoff that each player receives, called the coalition value [15]. Any coalition $S \subseteq \mathcal{N}$ represents an agreement between the players in S to act as a single group. Coalitional games are grouped into three classes: canonical coalitional games, coalition-formation games and coalitional graph games.

In a canonical coalitional game, the coalition is formed by all the players of the game. The main question is how to stabilize the coalition and how to distribute the gains of the coalition among the players in a fair manner. This can be done by using the Shapley value. The Shapley value is the expected marginal contribution of player i to the grand coalition, when the players join the coalition in a random order [15]. Following the notation in [15], given a canonical game $(\mathcal{N}, v)$, the Shapley value $\phi(v)$ assigns the payoff $\phi_i(v)$ for every player i given by

$$\phi_i(v) = \sum_{S \subseteq \mathcal{N} \setminus \{i\}} \frac{|S|!(N - |S| - 1)!}{N!} [v(S \cup \{i\}) - v(S)]. \tag{3-2}$$

This way, the total surplus generated by the grand coalition is distributed fairly among the agents.

In a coalition-formation game, the main question is how the coalitions must be formed and what the cost for cooperation is. Coalition-formation games can be divided into two groups: static coalition-formation games and dynamic coalition-formation games. In a static coalition-formation game, the structure of the coalition is imposed by an external factor and the objective is to study the properties of this structure, like stability. The main objective of dynamic coalition-formation games is to analyze the formation of coalitions through the players' interactions and to study how the system reacts to environmental changes, for example a change in the number of players [16]. Solving coalition-formation games, the dynamic variants in specific, is more difficult and application specific than solving canonical games.

A coalitional game where players' interactions are governed by a communication graph structure is referred to as a coalitional graph game. The main question here is how to stabilize the grand coalition or form a network structure taking the communication graph into account [15]. For a more elaborate treatment of this topic, the reader is referred to [15, Chapter 7].

3-2 Generalized Nash equilibrium problems and variational inequalities

With a standard Nash equilibrium problem (NEP) as described in Section 3-1-1, only the payoff of each agent depends on the decisions of the other agents. However, when the players have some shared limited resources, as is the case in a power grid, not only the payoff depends on the decisions of the other players, but also the feasible set of each player depends on the strategies of the other players. The problem is then called a generalized Nash equilibrium problem (GNEP). In many practical applications, ranging from economics to engineering, these coupling constraints have to be considered to obtain a realistic model [17]. In a GNEP, each agent seeks to minimize its own cost function under not only local feasibility constraints, but also some joint feasibility constraints. In what follows it will be explained in more detail what a generalized game looks like and how they can be solved.

We consider a set of noncooperative agents $\mathcal{I} = \{1, \dots, N\}$, each of which has to choose its strategy $x_i \in \mathbb{R}^{n_i}$ from its local decision set $\Omega_i \subseteq \mathbb{R}^{n_i}$. The cost function of each agent depends on the decision of agent i itself, x_i, and the decisions of the other agents, $\boldsymbol{x}_{-i}$:

$$J_i(x_i, \boldsymbol{x}_{-i}) := f_i(x_i, \boldsymbol{x}_{-i}) + g_i(x_i). \tag{3-3}$$

The part $g_i(x_i)$ denotes a local cost and can be non smooth. For this example we consider the local cost to be a local constraint via an indicator function, i.e., $g_i(x_i) = I_{\Omega_i}(x_i)$. Furthermore, because we consider a game with shared limited resources, the coupling between the agents not only arises from the cost function, but also via their feasible decision sets. In this example the specific case is considered where the game has affine coupling constraints, i.e., $A\boldsymbol{x} \leq b$. Each player i aims to minimize its objective function, given the strategies $\boldsymbol{x}_{-i}$ of the other players:

$$\forall i \in \mathcal{I} : \begin{cases} \min_{x_i \in \Omega_i} & J_i(x_i, \boldsymbol{x}_{-i}) \\ \text{s.t.} & A_i x_i \leq b - \sum_{j \neq i}^{N} A_j x_j. \end{cases} \tag{3-4}$$

The feasible set of each player i, which is dependent on the decisions of the other players, $\boldsymbol{x}_{-i}$, is denoted by

$$\mathcal{X}_i(\boldsymbol{x}_{-i}) := \left\{ y_i \in \Omega_i \mid A_i y_i \leq b - \sum_{j \neq i}^{N} A_j x_j \right\}. \tag{3-5}$$

The collective feasible set can be written as

$$\boldsymbol{\mathcal{X}} = \{ \boldsymbol{y} \in \boldsymbol{\Omega} \mid A\boldsymbol{y} - b \leq 0_m \}. \tag{3-6}$$

It is assumed that for each $i \in \mathcal{I}$ and $\boldsymbol{x}_{-i} \in \mathcal{X}_{-i}$ the function $f_i(\cdot, x_{-i})$ is convex and continuously differentiable and the set $\boldsymbol{\mathcal{X}}$ satisfies Slater's constraint qualification [18]. A generalized Nash equilibrium (GNE) is a collective strategy $\boldsymbol{x}^* \in \boldsymbol{\mathcal{X}}$ such that for all $i \in \mathcal{I}$

$$J_i(x_i^*, \boldsymbol{x}_{-i}^*) \leq \inf\{ J_i(y, \boldsymbol{x}_{-i}^*) | y \in \mathcal{X}_i(\boldsymbol{x}_{-i}^*) \}. \tag{3-7}$$

However, GNEPs are generally very hard to solve. According to [19] it is reasonable to state that the only GNEPs for which solution procedures exist, are the ones that can be reduced to

a variational inequality (VI), for which a vast amount of theory is available. The VI defined by $\boldsymbol{\mathcal{X}}$ and F, is the problem of finding a vector $\boldsymbol{x}^* \in \boldsymbol{\mathcal{X}}$ such that

$$F(\boldsymbol{x}^*)^T(\boldsymbol{x} - \boldsymbol{x}^*) \geq 0, \quad \forall \boldsymbol{x} \in \boldsymbol{\mathcal{X}}, \tag{3-8}$$

where $\boldsymbol{\mathcal{X}} \subset \mathbb{R}^n$ is a closed convex set and $F : \boldsymbol{\mathcal{X}} \to \mathbb{R}^n$ is a continuous function. The VI$(\boldsymbol{\mathcal{X}}, F)$ is equivalent to finding a vector $\boldsymbol{x}^* \in \boldsymbol{\mathcal{X}}$ that satisfies the minimum principle

$$\nabla f(\boldsymbol{x}^*)^T(\boldsymbol{x} - \boldsymbol{x}^*) \geq 0, \quad \forall \boldsymbol{x} \in \boldsymbol{\mathcal{X}}, \tag{3-9}$$

where $F = \nabla f$. If we now define $\boldsymbol{\mathcal{X}}$ as in (3-6) and F as

$$F(\boldsymbol{x}) \equiv \begin{pmatrix} \nabla_{x_1} J_1(x_1, \boldsymbol{x}_{-1}) \\ \vdots \\ \nabla_{x_N} J_N(x_N, \boldsymbol{x}_{-N}) \end{pmatrix}, \tag{3-10}$$

where for each $i \in \mathcal{I}$, $J_i(x_i, \boldsymbol{x}_{-i})$ is defined as in (3-3), every solution of the VI$(\boldsymbol{\mathcal{X}}, F)$ is a solution of the GNEP (3-4) [19]. However, in the reduction from a GNEP to a VI not all solutions are preserved. It can even happen that the GNEP has solutions, but the corresponding VI has none. In order for the solution set of the VI to be nonempty and compact, the set $\boldsymbol{\mathcal{X}}$ must be compact and convex and the function $F(\boldsymbol{x})$ must be continuous [20].

To come to a solution procedure, we start by stating the Lagrangian of the GNEP, which is given by

$$\mathcal{L}_i(\boldsymbol{x}, \lambda_i) := J_i(x_i, \boldsymbol{x}_i) + I_{\Omega_i}(x_i) + \lambda_i^\top (A\boldsymbol{x} - b), \tag{3-11}$$

where $\lambda_i \in \mathbb{R}^m_{\geq 0}$ is the Lagrangian dual variable. The Karush-Kuhn-Tucker (KKT) conditions for the GNEP are

$$\begin{cases} 0 \in \nabla_{x_i} J_i(x_i^*, \boldsymbol{x}_{-i}^*) + \mathrm{N}_{\Omega_i}(x_i^*) + A_i^\top \lambda_i^* \\ 0 \in \mathrm{N}_{\mathbb{R}^m_{\geq 0}}(\lambda_i^*) - (A\boldsymbol{x}^* - b). \end{cases} \tag{3-12}$$

In terms of the VI, the collective decision $\boldsymbol{x}^*$ is a solution of the VI$(\boldsymbol{\mathcal{X}}, F)$ if and only if

$$\boldsymbol{x}^* \in \arg\min_{\boldsymbol{y} \in \boldsymbol{\mathcal{X}}} (\boldsymbol{y} - \boldsymbol{x}^*)^\top F(\boldsymbol{x}^*), \tag{3-13}$$

which gives rise to the following KKT conditions:

$$\begin{cases} 0 \in \nabla_{x_i} J_i(x_i^*, \boldsymbol{x}_{-i}^*) + \mathrm{N}_{\Omega_i}(x_i^*) + A_i^\top \lambda^* \\ 0 \in \mathrm{N}_{\mathbb{R}^m_{\geq 0}}(\lambda^*) - (A\boldsymbol{x}^* - b). \end{cases} \tag{3-14}$$

The solution $\boldsymbol{x}$ of the VI$(\boldsymbol{\mathcal{X}}, F)$ at which the KKT conditions (3-14) hold is a solution of the GNEP at which the KKT conditions (3-12) hold with $\lambda_1 = \lambda_2 = \ldots = \lambda_N$[19]. This is called the variational GNE (v-GNE). Conversely, the solution $\boldsymbol{x}$ of the GNEP at which the KKT conditions (3-12) hold with $\lambda_1 = \lambda_2 = \ldots = \lambda_N$, is a solution of the VI$(\boldsymbol{\mathcal{X}}, F)$ [19]. In other words, the agents have to reach consensus on the dual variable λ to come to the variational solution. This variational solution also reflects a notion of fairness among the agents [21].

The KKT conditions in (3-14) can be written in a more compact form as the monotone inclusion

$$0 \in \mathcal{T}(\boldsymbol{x}, \boldsymbol{\lambda}) := \begin{bmatrix} F(\boldsymbol{x}) + \mathrm{N}_{\boldsymbol{\Omega}}(\boldsymbol{x}) + A^\top \lambda \\ \mathrm{N}_{\mathbb{R}^m_{\geq 0}}(\lambda) - (A\boldsymbol{x} - b) \end{bmatrix}. \tag{3-15}$$

Finding a solution of the VI($\boldsymbol{\mathcal{X}}, F$) (and hence of the original variational GNEP (v-GNEP)) now comes down to finding the zeros of the operator $\mathcal{T}$. To solve this monotone inclusion, different operator splitting schemes can be used. One of the simplest operator splitting schemes is the forward-backward (FB) scheme [22]. The operator $\mathcal{T}$ is split in a part $\mathcal{A}$ and $\mathcal{B}$, for which it holds that $\mathcal{T} = \mathcal{A} + \mathcal{B}$. The zeros of the mapping $\mathcal{A} + \mathcal{B}$ correspond to the fixed point of a specific operator depending on both $\mathcal{A}$ and $\mathcal{B}$. The solution can then be found by using a fixed point iteration on this operator. Besides the FB splitting, different operator splitting methods exist which all have their benefits and drawbacks, e.g. the forward-backward-forward and forward-backward-half-forward scheme [23].

3-3 Stochastic generalized Nash equilibrium problems

The NEP and GNEP discussed in the previous section were derived for the deterministic case when there is no uncertainty in the system. However, in real applications one may encounter uncertainties, for example uncertainty in the demand in an electricity market. When the cost functions of the NEP contain expected value functions, the problem is called a stochastic NEP (SNEP). When the system also has shared constraints, a SNEP becomes a stochastic GNEP (SGNEP). If the random variable is known, a SGNEP could be solved in the same way as a deterministic GNEP by solving the corresponding VI. However, because the pseudo-gradient is usually not accessible due to excessive computations, the solution of the stochastic VI (SVI) relies on samples of the random variables [18]. There are two main methodologies which are used for this matter: sample average approximation (SAA) and stochastic approximation (SA). In the SAA approach, the expected value is replaced with the average over an infinite number of samples of the random variable [24]. In the SA approach, each agent uses only one realization of the random variable. This method is computationally less expensive, but usually requires stronger assumptions on the mappings involved [18]. As an alternative, the SA scheme can be used with variance reduction. This scheme considers the average over an increasing amount of samples, which is possible when there is a lot of data available. This method is computationally more expensive than using only one realization, but usually requires less strong assumptions on the mappings involved [24].

Using the same setup as in the previous section, a set of agents $\mathcal{I} = \{1, \ldots, N\}$ with strategies $x_i \in \Omega_i \subseteq \mathbb{R}^{n_i}$ is considered. The cost function of each agent i depends on the decision of agent i itself, $\boldsymbol{x}_i$, the decisions of the other agents, x_{-i}, and in the stochastic case also on the random variable $\xi : \Xi \to \mathbb{R}^d$ through which the uncertainty is expressed:

$$\mathbb{J}_i(x_i, \boldsymbol{x}_{-i}) := \mathbb{E}_\xi[f_i(x_i, \boldsymbol{x}_{-i}, \xi(\omega))] + g_i(x_i), \tag{3-16}$$

where $(\Xi, \mathcal{F}, \mathbb{P})$ denotes the probability space. The local cost $g_i(x_i)$ is again taken to be the local constraint via the indicator function, $g_i(x_i) = I_{\Omega_i}(x_i)$, and we also look at the specific case where the game has affine coupling constraints $A\boldsymbol{x} \leq b$. The feasible set of each agent

$i \in \mathcal{I}$ is denoted by (3-5) and the collective feasible set by (3-6). It is assumed there is no uncertainty in the constraints and, as in the deterministic case, that for each $i \in \mathcal{I}$ and $\boldsymbol{x}_{-i} \in \mathcal{X}_{-i}$ the cost function is convex and continuously differentiable and the set $\boldsymbol{\mathcal{X}}$ satisfies Slater's constraint qualification [18].

Given the decisions of the other agents $\boldsymbol{x}_{-i}$, each player aims to to solve the local optimization problem:

$$\forall i \in \mathcal{I} : \begin{cases} \min_{x_i \in \Omega_i} & \mathbb{J}(x_i, \boldsymbol{x}_{-i}) \\ \text{s.t.} & A_i x_i \leq b - \sum_{j \neq i}^{N} A_j x_j. \end{cases} \tag{3-17}$$

The aim of the algorithm is to compute the stochastic GNE, that is, a collective strategy $\boldsymbol{x}^* \in \mathcal{X}$ such that for all $i \in \mathcal{I}$

$$\mathbb{J}_i(x_i^*, \boldsymbol{x}_{-i}^*) \leq \inf\{\mathbb{J}_i(y, \boldsymbol{x}_{-i}^*) | y \in \mathcal{X}_i(\boldsymbol{x}_{-i}^*)\}. \tag{3-18}$$

As in the previous section, the solution we are interested in, is the solution of the associated SVI($\mathcal{X}, \mathbb{F}$)

$$\langle \mathbb{F}(\boldsymbol{x}^*), \boldsymbol{x} - \boldsymbol{x}^* \rangle \geq 0, \quad \forall x \in \mathcal{X}, \tag{3-19}$$

where $\mathbb{F}(\boldsymbol{x})$ is the (pseudo) gradient mapping

$$\mathbb{F}(\boldsymbol{x}) = \text{col}((\mathbb{E}[\nabla_{x_i} f_i(x_i, \boldsymbol{x}_{-i}, \xi)])_{i \in \mathcal{I}}). \tag{3-20}$$

The Lagrangian of the SGNEP is

$$\mathcal{L}_i(\boldsymbol{x}, \lambda_i) := \mathbb{E}_\xi[f_i(x_i, \boldsymbol{x}_{-i}, \xi)] + I_{\Omega_i}(x_i) + \lambda_i^\top (A\boldsymbol{x} - b), \tag{3-21}$$

where $\lambda_i \in \mathbb{R}^m_{\geq 0}$ is the Lagrangian dual variable. The KKT conditions for the SGNEP are

$$\begin{cases} 0 \in \mathbb{E}[\nabla_{x_i} f_i(x_i^*, \boldsymbol{x}_{-i}^*, \xi)] + \mathrm{N}_{\Omega_i}(x_i^*) + A_i^\top \lambda_i^* \\ 0 \in N_{\mathbb{R}^m_{\geq 0}}(\lambda_i^*) - (A\boldsymbol{x}^* - b). \end{cases} \tag{3-22}$$

In terms of the associated SVI, the collective decision $\boldsymbol{x}^*$ is a solution of the SVI($\mathcal{X}, \mathbb{F}$) if and only if

$$\boldsymbol{x}^* \in \arg\min_{\boldsymbol{y} \in \boldsymbol{\mathcal{X}}} (\boldsymbol{y} - \boldsymbol{x}^*)^\top \mathbb{F}(\boldsymbol{x}^*), \tag{3-23}$$

which gives the following KKT conditions:

$$\forall i \in \mathcal{I} : \begin{cases} 0 \in \mathbb{E}[\nabla_{x_i} f_i(x_i^*, \boldsymbol{x}_{-i}^*, \xi)] + \mathrm{N}_{\Omega_i}(x_i^*) + A_i^\top \lambda^* \\ 0 \in N_{\mathbb{R}^m_{\geq 0}}(\lambda^*) - (A\boldsymbol{x}^* - b). \end{cases} \tag{3-24}$$

Again, the agents have to reach consensus on the dual variable λ to come to the variational solution of the game. As in the deterministic case, the KKT conditions (3-24) can be rewritten as a monotone inclusion on which different operator splitting schemes can be applied.

Chapter 4

Problem formulation

This chapter presents the electricity market model for which the algorithms in Chapter 5 will be derived. First, Section 4-1 gives some notations that are used in this chapter and Chapter 5. Then, Section 4-2 describes the market setup, after which Section 4-3 will give the final optimization problem which will be used in Chapter 5.

4-1 Notation

$\mathbf{0}_N$ and $\mathbf{1}_N$ denote a vector consisting of N zeros or ones, respectively. Given $x_1, \ldots, x_N \in \mathbb{R}^n$, $\boldsymbol{x} := \text{col}(x_1, \ldots, x_N) = [x_1^\top, \ldots, x_N^\top]^\top$. The resolvent of a maximally monotone operator A is denoted by $J_A = (\text{Id} + A)^{-1}$. The mapping $I_S : \mathbb{R}^n \to \{0, \infty\}$ denotes the indicator function for the set $S \subseteq \mathbb{R}^n$, i.e., $I_S(x) = 0$ if $x \in S$, $I_S(x) = \infty$ otherwise. The set-valued operator $\text{N}_S : \mathbb{R}^n \to \mathbb{R}^n$ denotes the normal cone operator for the set $S \subseteq \mathbb{R}^n$ and is defined as $\text{N}_S(x) = \partial I_S(x) = \begin{cases} \emptyset, & x \notin S \\ \{w \in \mathbb{R}^n | w^\top (z - x)\} \leq 0, \ \forall z \in S, & x \in S \end{cases}$.

4-2 Electricity market model

The electricity market considered in this work is a full peer-to-peer (P2P) market, derived from the electricity market model presented in [6]. A day-ahead P2P electricity market with prosumers in a microgrid is modeled as an economic dispatch problem. The agents (prosumers) in the system can trade electricity with the main grid as well as with (some of the) other agents autonomously and they might have dispatchable generation capabilities and/or storage units. All relevant control signals needed for the optimization process are shared directly between the agents without the need of a central coordinator.

The total set of agents is denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$. They communicate through the communication graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where the vertices $\mathcal{N}$ represent the agents and the edges $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ represent the communication links between the them, with $|\mathcal{E}| = E$. The set of

prosumers with which prosumer i can trade power are called the neighbors of agent i and is denoted by $\mathcal{N}_i = \{j|(i,j) \in \mathcal{E}\}$. Hence, agent i can trade energy with agent j only if the edge (i,j) lies in the set of edges $\mathcal{E}$. Furthermore, the graph $\mathcal{G}$ is undirected, which means that if for agents i and j $(i,j) \in \mathcal{E}$, we also have that $(j,i) \in \mathcal{E}$. Also, the graph is assumed to be connected, i.e., there always exists a path between any two prosumers.

At each time instant h, each agent $i \in \mathcal{N}$ has a certain demand, denoted by $p^d_{i,h}$. This demand is defined as the difference between the aggregate load at that time instant and the total power produced by non-dispatchable generation units (such as a PV system) owned by that agent, where a positive $p^d_{i,h}$ means that the aggregate load at time instant h is greater than the non-dispatchable generation. To meet this demand $p^d_{i,h}$, agent i has the following set of decision variables at each time instant h:

$$x_{i,h} = \text{col}\left(p^{dg}_{i,h},\ p^{st}_{i,h},\ p^{mg}_{i,h},\ \{p^{tr}_{(i,j),h}\}_{j\in\mathcal{N}_i}\right) \in \mathbb{R}^{n_i}, \tag{4-1}$$

where $p^{dg}_{i,h} \in \mathbb{R}_{\geq 0}$ denotes the power produced by dispatchable generation units, $p^{st}_{i,h} \in \mathbb{R}$ denotes the power delivered to/by the storage unit, $p^{mg}_{i,h} \in \mathbb{R}$ denotes the power traded with the main grid and $p^{tr}_{(i,j),h} \in \mathbb{R}$ denotes the power traded with neighbor $j \in \mathcal{N}_i$. Note that a positive value for $p^{st}_{i,h}$ means power is used from the storage unit and a negative value means power is delivered to the storage unit. For $p^{mg}_{i,h}$ and $p^{tr}_{(i,j),h}$ a positive value means buying power and a negative value means selling power.

The goal of each agent is to minimize the cost of power usage, while certain constraints must be satisfied. In what follows the cost functions and constraints for each of the decision variables will be discussed.

4-2-1 Cost functions and constraints

Power balance

As stated before, $p^d_{i,h}$ denotes the difference in the aggregate load and the total generation by non-dispatchable generation units for agent i at time instant h. Hence, $p^d_{i,h}$ indicates the actual power demand of that agent at that time instant. This power must be obtained by dispatchable generation, using the storage unit, trading with the main grid and/or trading with neighbors. Therefore, these decision variables combined must equal the demand at each time instant:

$$\mathbf{1}^\top_{n_i} x_{i,h} = p^d_{i,h}. \tag{4-2}$$

Dispatchable generation

Residential dispatchable generation units are for example micro-combined heat and power (CHP) units [25]. A micro-CHP unit can simultaneously produce heat and electricity in individual homes and is considered an important technology for the future power grid [26].

The group of agents that own a dispatchable generation unit is denoted by $\mathcal{N}^{dg} \subseteq \mathcal{N}$. Since the goal of the optimization problem is to minimize the cost of power usage, also the power produced by dispatchable generation units must be minimized. A widely adopted cost function

for dispatchable generation is a quadratic convex function [27], [28]. The cost function reads as

$$f_{i,h}^{dg} = q_i^{dg}(p_{i,h}^{dg})^2 + c_i^{dg} p_{i,h}^{dg}, \tag{4-3}$$

where $q_i^{dg} > 0$ and c_i^{dg} are constants.

The amount of power that can be produced by dispatchable generation by agent i is bounded by a minimum amount $\underline{p}_i^{dg} > 0$ and a maximum amount $\overline{p}_i^{dg} \geq 0$. The constraint on the decision variable p_i^{dg} now reads as

$$\begin{aligned} \underline{p}_i^{dg} \leq p_{i,h}^{dg} \leq \overline{p}_i^{dg}, \quad \text{if } i \in \mathcal{N}^{dg} \\ p_{i,h}^{dg} = 0, \quad \text{otherwise.} \end{aligned} \tag{4-4}$$

Storage unit

Storage units are also an important part of the future grid. To deal with the volatility of renewable energy resources and increased load in the grid, they can provide a range of services to ensure a reliable and efficient operation, e.g. providing frequency response, black-start capabilities and reserve capacity and supporting self-consumption for residences with rooftop photovoltaic (PV) systems [13]. As of now, using batteries for energy storage is still rather expensive, but as technology advances, the cost of these storage units will decrease and the economical services they can provide will increase [13].

The group of agents that own a storage unit is denoted by $\mathcal{N}^{dg} \subseteq \mathcal{N}$. Because the maximum capacity and efficiency of a battery deteriorate with the use of the battery [29], another goal of the optimization is to minimize the use of the storage unit. The cost function $f_{i,h}^{st}$ for using the storage unit is defined by a quadratic cost:

$$f_{i,h}^{st} = q_i^{st}(p_{i,h}^{st})^2 + c_i^{st} p_{i,h}^{st}, \tag{4-5}$$

where $q_i^{st} \geq 0$ and c_i^{st} are constants.

There are a multiple methods to estimate the state of charge (SOC) of a lithium-ion battery [30]. Because accurate SOC estimation is quite a complex matter, see e.g., [31] and [30], and it is not the goal of this work to have a model of the battery that is as realistic as possible, a simple integrator is used to estimate the SOC. Using the SOC, denoted by $s_{i,h} > 0$, the constraints on the decision variable $p_{i,h}^{st}$ are given by

$$\begin{cases} s_{i,h+1} & = a_i s_{i,h} + b_i p_{i,h}^{st}, \\ & \underline{s}_i \leq s_{i,h+1} \leq \overline{s}_i, \quad \text{if } i \in \mathcal{N}^{st} \\ & -p_i^{ch} \leq p_{i,h}^{st} \leq p_i^{dh}, \\ & p_{i,h}^{st} = 0, \quad \text{otherwise,} \end{cases} \tag{4-6}$$

where $a_i \in (0, 1]$ is the efficiency of storage, $b_i = -\frac{T_s}{e_{cap,i}}$, where T_s denotes the sampling time and $e_{cap,i}$ the maximum capacity of storage, $\underline{s}_i, \overline{s}_i \in [0, 1]$ denote the minimum and maximum SOC of the storage unit of agent i, respectively, and $p_i^{ch} \geq 0$ and $p_i^{dh} \geq 0$ denote the maximum charging and discharging power.

Trades with main grid

Each agent i can trade power with the main grid. At each time instant h, the price of electricity from the main grid is dependent on the total electricity demand at that time instant. A typical function for the price is of quadratic nature [27]:

$$c_h^{mg}(\hat{p}_h^{mg}) = q_h^{mg}(\hat{p}_h^{mg})^2, \tag{4-7}$$

where $q_h^{mg} > 0$ is a constant and $\hat{p}_h^{mg}$ denotes the total load on the main grid:

$$\hat{p}_h^{mg} = \sum_{i \in \mathcal{N}} p_{i,h}^{mg}. \tag{4-8}$$

The cost function for trades with the main grid of agent i at time instant h is defined as the price at that time instant times the relative amount of power agent i trades with the main grid:

$$f_{i,h}^{mg}\left(p_{i,h}^{mg}, \hat{p}_h^{mg}\right) = c_h^{mg}(\hat{p}_h^{mg}) \frac{p_{i,h}^{mg}}{\sum_{j \in \mathcal{N}} p_{j,h}^{mg}} = q_h^{mg} \left(\sum_{j \in \mathcal{N}} p_{j,h}^{mg} \right) p_{i,h}^{mg}. \tag{4-9}$$

To prevent congestion of the grid and to ensure continuous operation of the generators that supply the main grid, the amount of electricity that can be traded between the main grid and the agents is bounded by a lower and upper bound $\overline{p}^{mg} > \underline{p}^{mg} \geq 0$:

$$\underline{p}^{mg} \leq \sum_{i \in \mathcal{N}} p_{i,h}^{mg} \leq \overline{p}^{mg}. \tag{4-10}$$

This constraint on the aggregate load couples the decisions of the agents.

P2P trading

As stated before, the group of agents with which agent i can trade are called the neighbors of agent i and is denoted by $\mathcal{N}_i$. The cost function of P2P trades is defined by a linear function as

$$f_{i,h}^{tr}\left(\{p_{(i,j),h}^{tr}\}_{j \in \mathcal{N}_i}\right) = \sum_{j \in \mathcal{N}_i} c_{(i,j)}^{tr} p_{(i,j),h}^{tr}, \tag{4-11}$$

where $c_{(i,j)}^{tr} > 0$ is the cost of the electricity trade on which agent i and j have agreed.

The amount of power agent i and j can trade is bounded by a maximum value $\overline{p}_{(i,j)}^{tr}$:

$$-\overline{p}_{(i,j)}^{tr} \leq p_{(i,j),h}^{tr} \leq \overline{p}_{(i,j)}^{tr}, \quad \forall j \in \mathcal{N}_i. \tag{4-12}$$

The last constraint dictates that at each time step the amount of power agent i trades with agent j is the same amount as agent j trades with agent i, but with opposite sign:

$$p_{(i,j),h}^{tr} + p_{(j,i),h}^{tr} = 0, \quad \forall j \in \mathcal{N}_i. \tag{4-13}$$

Hence, the amount of power agent i sells to agent j is the same amount agent j buys from agent i at each time instant. Constraint (4-13) is called the reciprocity constraint and also couples the decisions of agents i and j.

4-3 Economic dispatch problem

Following [6], a more compact notation is introduced for some of the constraints and decision variables. First, define the set that contains only the local constraints of agent i:

$$x_{i,h} \in \Omega_{i,h}, \tag{4-14}$$

where $\Omega_{i,h}$ is a set such that the constraints (4-2), (4-4), (4-6) and (4-12) hold. Furthermore, define the stacked vector x_i, which contains the decision variables of agent i over all time steps:

$$x_i = \text{col}(\{x_{i,h}\}_{h\in\mathcal{H}}), \quad \text{where } \mathcal{H} = \{1, 2, \dots, H\}, \tag{4-15}$$

where H is the amount of time instants. Just as in [6], two matrices are used in order to manipulate the selection of the different decision variables from the vector x_i, in particular the trades with the main grid and trades with neighboring agents. For all $i \in \mathcal{N}$ we define

$$S_i^{mg} := I_H \otimes a_{n_i,3}^\top, \tag{4-16}$$

$$S_{(i,j)}^{tr} := I_H \otimes a_{n_i,r(i,j)}^\top, \quad \forall j \in \mathcal{N}_i, \tag{4-17}$$

where $a_{n_i,l} \in \mathbb{R}^{n_i}$ is a column vector of dimension n_i which contains all zeros except for the $l-$th entry being 1. In (4-17) $r(i,j)$ denotes the index of the power traded with neighbor j. Hence, S_i^{mg} and $S_{(i,j)}^{tr}$ select the decision variables associated with trading with the main grid and neighbor j, respectively:

$$\begin{aligned} S_i^{mg} &= \text{col}(\{p_{i,h}^{mg}\}_{h\in\mathcal{H}}) =: p_i^{mg}, \\ S_{(i,j)}^{tr} &= \text{col}(\{p_{(i,j),h}^{tr}\}_{h\in\mathcal{H}}) =: p_{(i,j)}^{tr}, \quad \forall j \in \mathcal{N}_i. \end{aligned}$$

Now that all the cost functions and constraints are defined, the economic dispatch problem can be formulated. As stated before, the goal of each agent is to minimize the cost of power usage, while meeting the constraints. This means that each agent i searches for a strategy x_i^* that minimizes its local objective function, while being subject to all the constraints. The cumulative local objective function for agent i is the summation of the cost functions (4-3), (4-5), (4-11) and (4-9) over the time horizon $\mathcal{H}$:

$$J_i(x_i, \hat{p}^{mg}) = \sum_{h\in\mathcal{H}} \left(f_{i,h}^{dg}(p_{i,h}^{dg}) + f_{i,h}^{st}(p_{i,h}^{st}) + f_{i,h}^{mg}(p_{i,h}^{mg}, \hat{p}_h^{mg}) + f_{i,h}^{tr}(p_{i,h}^{tr}), \right) \tag{4-18}$$

where $\hat{p}^{mg} = \text{col}(\hat{p}_1^{mg}, \dots, \hat{p}_H^{mg})$, with $\hat{p}_h^{mg}$ as defined in (4-8). Now, the optimization problem of agent i can be written as

$$x_i^* \in \begin{cases} \text{argmin}_{x_i} & J_i(x_i, \sum_{j\in\mathcal{N}} S_j^{mg} x_j) & \text{(4-19a)} \\ \text{s.t.} & x_i \in \Omega_i := \prod_{h\in\mathcal{H}} \Omega_{i,h}, & \text{(4-19b)} \\ & S_{(i,j)}^{tr} x_i + S_{(j,i)}^{tr} x_j = \mathbf{0}, \ \forall j \in \mathcal{N}_i, & \text{(4-19c)} \\ & \underline{p}^{mg} \mathbf{1}_H \le \sum_{j\in\mathcal{N}} S_j^{mg} x_j \le \bar{p}^{mg} \mathbf{1}_H, & \text{(4-19d)} \end{cases}$$

Following [6], to make the computations easier, the constraints in (4-19) are written in a more compact form. Let us start by recasting the reciprocity constraints (4-19c) by introducing the incidence-selection matrix R, whose $(l,i)-$block is

$$[R]_{l,i} := \begin{cases} S^{tr}_{(i,j)}, & \text{if } e_l = (i,j), \text{ for some } j \neq i, \\ \mathbf{0}_{(H\times Hn_i)}, & \text{otherwise,} \end{cases} \tag{4-20}$$

where $S^{tr}_{(i,j)}$ is defined in (4-17). Using this matrix R, the reciprocity constraint (4-19c) can be written compactly as

$$R\boldsymbol{x} = \mathbf{0}_{EH}. \tag{4-21}$$

The grid constraint (4-19d) can be recast to a more compact form as

$$A\boldsymbol{x} - b \leq \mathbf{0}_{2H}, \tag{4-22}$$

where

$$A = \begin{bmatrix} S \\ -S \end{bmatrix}, \tag{4-23}$$

$$S = \begin{bmatrix} S^{mg}_1, \ldots, S^{mg}_N \end{bmatrix} \tag{4-24}$$

and

$$b = \begin{bmatrix} \bar{p}^{mg}\mathbf{1}_H \\ -\underline{p}^{mg}\mathbf{1}_H \end{bmatrix}. \tag{4-25}$$

The cost function (4-18) can be split in a part that contains the cost of only the local decision variables of each agent and a part that also contains the decision variables of the other agents, which couples the decisions of all agents. The local part of the cost function is given by

$$J_{i,lc}(x_i) = \sum_{h\in\mathcal{H}} \left(f^{dg}_{i,h}(p^{dg}_{i,h}) + f^{st}_{i,h}(p^{st}_{i,h}) + f^{tr}_{i,h}(p^{tr}_{i,h}) \right) \tag{4-26}$$

and the coupling part by

$$J_{i,cp}(x_i, S\boldsymbol{x}) = \left(Q \sum_{j\in\mathcal{N}} S^{mg}_j x_j \right)^{\top} S^{mg}_i x_i, \tag{4-27}$$

where $Q := \text{diag}(q^{mg}_1, \ldots, q^{mg}_H)$.

Using the compact notation for the reciprocity and grid constraints, the optimization problem (4-19) can be written in standard form:

$$\forall i \in \mathcal{N} : \begin{cases} \text{argmin}_{x_i} & J_{i,lc}(x_i) + J_{i,cp}(x_i, S\boldsymbol{x}) & \text{(4-28a)} \\ \text{s.t.} & x_i \in \Omega_i, & \text{(4-28b)} \\ & R\boldsymbol{x} = \mathbf{0}_{EH}, & \text{(4-28c)} \\ & A\boldsymbol{x} - b \leq \mathbf{0}_{2H}, & \text{(4-28d)} \end{cases}$$

This optimization problem is the starting point for the algorithms derived in Chapter 5.

Chapter 5

Distributed algorithms

Now the optimization problem is defined in standard form, the main algorithms to solve the economic dispatch problem given in (4-19) can be derived. As stated in Chapter 3, we are looking for the variational solution of the generalized Nash equilibrium problem (GNEP) given in (4-28). Because we want a full peer-to-peer (P2P) market, where there is no need for a central coordinator, the goal of the algorithms is to compute the variational generalized Nash equilibrium (v-GNE) in a distributive manner. In order to arrive at the algorithms, first, Section 5-1 will give the main derivation towards the monotone operator $\mathcal{T}$, which will be used for the different operator splitting schemes. Section 5-2 will derive a preconditioned forward-backward (pFB) scheme, Section 5-3 a forward-backward-forward (FBF) scheme and Section 5-4 a forward-backward-half-forward (FBHF) scheme. Note that it is not the intention of this work to provide theoretical guarantees of convergence of the proposed algorithms, but to test them on an energy market model. Therefore, no proofs of convergence will be given in this chapter. For a more theoretical derivation of the algorithms with proofs, see e.g. [32] and [23].

5-1 Main algorithm derivation

First, define the feasible set of each agent i, which consists of the local feasible set Ω_i and the shared constraints:

$$\mathcal{X}_i(\boldsymbol{x}_{-i}) = \{y_i \in \Omega_i \mid R_i y_i = \mathbf{0}_{EH},\ A_i y_i \leq b - \sum_{j \neq i}^{N} A_j x_j\}, \tag{5-1}$$

where H is the number of time steps and E is the number of edges of the communication graph $\mathcal{G}$ (defined in Section 4-2). The collective feasible set can now be written as

$$\boldsymbol{\mathcal{X}} = \{\boldsymbol{y} \in \boldsymbol{\Omega} \mid R\boldsymbol{y} = \mathbf{0}_{EH},\ A\boldsymbol{y} - b \leq \mathbf{0}_{2H}\}, \tag{5-2}$$

where $\boldsymbol{\Omega} = \prod_{i\in\mathcal{N}} \Omega_i$, R is defined in (4-21), A in (4-23) and b in (4-25). A collective strategy $\boldsymbol{x}^* = \text{col}(x_1^*, \ldots, x_N^*) \in \boldsymbol{\mathcal{X}}$ is a generalized Nash equilibrium (GNE) for the game given in (4-28) if, for all $i \in \mathcal{N}$, it holds that

$$J_i(x_i^*, S\boldsymbol{x}^*) \leq \inf\{J_i(y, S\boldsymbol{x}^*) \mid y \in \mathcal{X}_i\}, \tag{5-3}$$

where S is defined in (4-24). To find the GNE, we first define the Lagrangian of the game in (4-28) for each agent i:

$$\mathcal{L}_i(\boldsymbol{x}, \mu_i, \lambda_i) = J_i(x_i, S\boldsymbol{x}) + I_{\Omega_i}(x_i) + \mu_i^\top R\boldsymbol{x} + \lambda_i^\top (A\boldsymbol{x} - b), \tag{5-4}$$

where $I_{\Omega_i}(x_i)$ is the indicator function for the set of local constraints Ω_i, μ_i is the dual variable associated with the reciprocity constraint and λ_i is the dual variable associated with the grid constraint. The Karush-Kuhn-Tucker (KKT) conditions associated with this Lagrangian read as

$$\begin{cases} \mathbf{0}_{n_i} \in \nabla_{x_i} J_i(x_i^*, S\boldsymbol{x}^*) + \mathrm{N}_{\Omega_i}(x_i^*) + R_i^\top \mu_i^* + A_i^\top \lambda_i^* \\ \mathbf{0}_{EH} \in R\boldsymbol{x}^* \\ \mathbf{0}_{2H} \in \mathrm{N}_{\mathbb{R}^{2H}_{\geq 0}}(\lambda_i^*) - (A\boldsymbol{x}^* - b). \end{cases} \tag{5-5}$$

As stated in Chapter 3, the GNEPs we are interested in are those that can be passed to a corresponding variational inequality (VI), i.e., the problem of finding a vector $\boldsymbol{x}^* \in \boldsymbol{\mathcal{X}}$, such that

$$F(\boldsymbol{x}^*)^T(\boldsymbol{x} - \boldsymbol{x}^*) \geq 0, \quad \forall \boldsymbol{x} \in \boldsymbol{\mathcal{X}}. \tag{5-6}$$

The solutions that are preserved when passing the GNEP to a VI are those for which all players have the same Lagrangian multipliers for the grid and reciprocity constraints, i.e., $\lambda_1 = \lambda_2 = \ldots = \lambda_N$ and $\mu_1 = \mu_2 = \ldots = \mu_N$, respectively [19]. Besides the social fairness of this solution, a v-GNE is also computationally attractive, because a VI can be solved by operator splitting techniques. Since the collective strategy $\boldsymbol{x}^*$ is a solution of the VI($\boldsymbol{\mathcal{X}}, F$) if and only if

$$\boldsymbol{x}^* \in \arg\min_{\boldsymbol{y}\in\boldsymbol{\mathcal{X}}} (\boldsymbol{y} - \boldsymbol{x}^*)^\top F(\boldsymbol{x}^*), \tag{5-7}$$

the KKT conditions of the VI read as

$$\begin{cases} \mathbf{0}_{n_i} \in \nabla_{x_i} J_i(x_i^*, S\boldsymbol{x}^*) + \mathrm{N}_{\Omega_i}(x_i^*) + R_i^\top \mu^* + A_i^\top \lambda^* \\ \mathbf{0}_{EH} \in R\boldsymbol{x}^* \\ \mathbf{0}_{2H} \in \mathrm{N}_{\mathbb{R}^{2H}_{\geq 0}}(\lambda^*) - (A\boldsymbol{x}^* - b). \end{cases} \tag{5-8}$$

These KKT conditions can be written in a more compact form as the following monotone operator:

$$\mathcal{T}(\boldsymbol{x}, \mu, \boldsymbol{\lambda}) : \begin{bmatrix} \boldsymbol{x} \\ \mu \\ \boldsymbol{\lambda} \end{bmatrix} \mapsto \begin{bmatrix} F(\boldsymbol{x}) + \mathrm{N}_{\boldsymbol{\Omega}}(\boldsymbol{x}) + R^\top \mu + \overline{A}^\top \boldsymbol{\lambda} \\ -R\boldsymbol{x} \\ \mathrm{N}_{\mathbb{R}^{2HN}_{\geq 0}}(\boldsymbol{\lambda}) - (\overline{A}\boldsymbol{x} - \overline{b}) \end{bmatrix}, \tag{5-9}$$

where

$$F(\boldsymbol{x}) = \text{col}((\nabla_{x_i} J_i(x_i, S\boldsymbol{x}))_{i\in\mathcal{N}}), \tag{5-10}$$

$\overline{A} = \text{blkdiag}(A_1, \ldots, A_N)$ and $\overline{b} = \frac{1}{N}\text{col}(b, \ldots, b)$, where b is defined in (4-25).

The pFB and the FBHF algorithms need the pseudo-gradient mapping $F(\boldsymbol{x})$ to be strongly monotone to guarantee convergence, while the FBF only needs plain monotonicity of the involved operators [23]. Because the pseudo-gradient mapping $F(\boldsymbol{x})$ is only (maximally) monotone [33], the pFB and the FBHF algorithms are not guaranteed to converge. However, the results in Chapter 6 will show that the pFB and FBHF also converge to a solution.

5-1-1 Consensus constraint

To impose consensus on the dual variable $\boldsymbol{\lambda}$, the monotone operator $\mathcal{T}$ can be extended by an auxiliary variable. This can be done in two ways: using the incidence matrix $V \in \mathbb{R}^{E\times N}$ or the Laplacian matrix $L \in \mathbb{R}^{N\times N}$ of the communication graph $\mathcal{G}$ (defined in Section 4-2). The incidence matrix V is defined such that $V^\top V = L$.

Laplacian matrix

In [32] the authors proposed to the use the Laplacian constraint $\overline{L}\boldsymbol{\lambda} = \mathbf{0}_{2HN}$ to force consensus on the dual variable λ, i.e., $\lambda_i = \lambda_j,\ \forall j \in \mathcal{N}_i$, where $\overline{L} = L \otimes I_{2H} \in \mathbb{R}^{2HN\times 2HN}$. The operator $\mathcal{T}$ is expanded by introducing the auxiliary variable $\boldsymbol{z} \in \mathbb{R}^{2HN}$. The extended operator $\mathcal{T}_e$ reads as

$$\mathcal{T}_e(\boldsymbol{x},\mu,\boldsymbol{z},\boldsymbol{\lambda}) : \begin{bmatrix}\boldsymbol{x}\\ \mu\\ \boldsymbol{z}\\ \boldsymbol{\lambda}\end{bmatrix} \mapsto \begin{bmatrix} F(\boldsymbol{x}) + \mathrm{N}_{\boldsymbol{\Omega}}(\boldsymbol{x}) + R^\top\mu + \overline{A}^\top\boldsymbol{\lambda} \\ -R\boldsymbol{x} \\ \overline{L}\boldsymbol{\lambda} \\ \mathrm{N}_{\mathbb{R}^{2HN}_{\geq 0}}(\boldsymbol{\lambda}) - (\overline{A}\boldsymbol{x} - \overline{b}) - \overline{L}^\top\boldsymbol{z} + \overline{L}\boldsymbol{\lambda}\end{bmatrix}. \tag{5-11}$$

Incidence matrix

In [33], to impose consensus on the dual variable λ, the consensus constraint $\overline{V}\boldsymbol{\lambda} = \mathbf{0}_{2HN}$ is added, where $\overline{V} = V \otimes I_{2H} \in \mathbb{R}^{2HE\times 2HN}$. The operator $\mathcal{T}$ is expanded with the auxiliary variable $\boldsymbol{z} \in \mathbb{R}^{2HN}$ and defined as follows:

$$\mathcal{T}_e(\boldsymbol{x},\mu,\boldsymbol{z},\boldsymbol{\lambda}) : \begin{bmatrix}\boldsymbol{x}\\ \mu\\ \boldsymbol{z}\\ \boldsymbol{\lambda}\end{bmatrix} \mapsto \begin{bmatrix} F(\boldsymbol{x}) + \mathrm{N}_{\boldsymbol{\Omega}}(\boldsymbol{x}) + R^\top\mu + \overline{A}^\top\boldsymbol{\lambda} \\ -R\boldsymbol{x} \\ -\overline{V}\boldsymbol{\lambda} \\ \mathrm{N}_{\mathbb{R}^{2HN}_{\geq 0}}(\boldsymbol{\lambda}) - (\overline{A}\boldsymbol{x} - \overline{b}) + \overline{V}^\top\boldsymbol{z}\end{bmatrix}. \tag{5-12}$$

The original GNE seeking problem given in (4-19) has now become the problem of finding the zeros of the operator $\mathcal{T}_e$ in (5-11) or (5-12).

The incidence matrix uses the edges of the communication graph $\mathcal{G}$, where the Laplacian matrix uses the nodes of $\mathcal{G}$. When there are a lot of edges the Laplacian matrix may become faster. However, the Laplacian matrix uses an extra communication step. Because of the extra communication step needed when using the Laplacian matrix, the algorithms are derived using

the incidence matrix. Section 6-3 will show a comparison with the algorithms that are derived using the Laplacian matrix.

5-2 Preconditioned forward-backward operator splitting

Using the extended operator $\mathcal{T}_e$ given in (5-12) we can derive the three algorithms. We start by splitting the operator $\mathcal{T}_e$ in three parts, $\overline{\mathcal{A}}$, $\overline{\mathcal{B}}$ and $\overline{\mathcal{C}}$, in the following way:

$$\begin{aligned}
\overline{\mathcal{A}}: \begin{bmatrix} \boldsymbol{x} \\ \mu \\ \boldsymbol{z} \\ \boldsymbol{\lambda} \end{bmatrix} &\mapsto \begin{bmatrix} F(\boldsymbol{x}) \\ 0 \\ 0 \\ \overline{b} \end{bmatrix} \\
\overline{\mathcal{B}}: \begin{bmatrix} \boldsymbol{x} \\ \mu \\ \boldsymbol{z} \\ \boldsymbol{\lambda} \end{bmatrix} &\mapsto \begin{bmatrix} 0 & R^\top & 0 & \overline{A}^\top \\ -R^\top & 0 & 0 & 0 \\ 0 & 0 & 0 & -\overline{V} \\ -\overline{A} & 0 & \overline{V}^\top & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \mu \\ \boldsymbol{z} \\ \boldsymbol{\lambda} \end{bmatrix} \\
\overline{\mathcal{C}}: \begin{bmatrix} \boldsymbol{x} \\ \mu \\ \boldsymbol{z} \\ \boldsymbol{\lambda} \end{bmatrix} &\mapsto \begin{bmatrix} \mathrm{N}_{\boldsymbol{\Omega}}(\boldsymbol{x}) \\ 0 \\ 0 \\ \mathrm{N}_{\mathbb{R}^{2HN}_{\geq 0}}(\boldsymbol{\lambda}) \end{bmatrix}
\end{aligned} \tag{5-13}$$

The zeros of $\overline{\mathcal{A}}+\overline{\mathcal{B}}+\overline{\mathcal{C}}$ correspond to the zeros of the monotone operator $\mathcal{T}_e$ in (5-12). As stated before, the zeros of $\mathcal{T}_e$ correspond to the solution of the GNE given in (4-19) [18]. Because the forward-backward (FB) scheme cannot be applied directly to the GNEP, a preconditioning matrix is necessary [32]. The preconditioning matrix is given by:

$$\Phi_{FB} = \begin{bmatrix} \overline{\alpha}^{-1} & -R^\top & 0 & -\overline{A}^\top \\ -R & \overline{\beta}^{-1} & 0 & 0 \\ 0 & 0 & \overline{\gamma}^{-1} & \overline{V} \\ -\overline{A} & 0 & \overline{V}^\top & \overline{\delta}^{-1} \end{bmatrix}, \tag{5-14}$$

where the step sizes on the diagonal are defined as follows:

$$\overline{\alpha} = \mathrm{blkdiag}(\{\overline{\alpha_i}\}_{i\in\mathcal{N}}), \tag{5-15}$$

where the $\overline{\alpha}_i$'s are defined as $\overline{\alpha} = \mathrm{blkdiag}(\overline{\alpha}_{i,1}, \ldots, \overline{\alpha}_{i,H})$, where $\alpha_{i,h} = \mathrm{diag}\left(\alpha^{dg}_{i,h}, \alpha^{st}_{i,h}, \alpha^{mg}_{i,h}, \{\alpha^{tr}_{(i,j),h}\}_{j\in\mathcal{N}_i}\right)$,

$$\overline{\beta} = \beta I_{EH}, \tag{5-16}$$

$$\overline{\gamma} = \gamma I_{2HN}, \tag{5-17}$$

and

$$\overline{\delta} = \delta I_{2HN}. \tag{5-18}$$

If we denote the state variable of the distributed algorithm by $\omega = \text{col}(\boldsymbol{x}, \mu, \boldsymbol{z}, \boldsymbol{\lambda})$, the pFB algorithm can be written as a fixed-point iteration of the form $\omega^{k+1} = T_{FB}\omega^k$ where $T_{FB} = J_{\Phi^{-1}(\overline{\mathcal{B}}+\overline{\mathcal{C}})}(\text{Id} - \Phi_{FB}^{-1}\overline{\mathcal{A}})$, i.e.:

$$\omega^{k+1} = (\text{Id} + \Phi_{FB}^{-1}(\overline{\mathcal{B}} + \overline{\mathcal{C}}))^{-1} \circ (\text{Id} - \Phi_{FB}^{-1}\overline{\mathcal{A}})\omega^k. \tag{5-19}$$

Using a change of coordinates $w_i^k = [\overline{V}^\top z^k]_i$, we arrive at Algorithm 1, where the matrices S_i^{mg} and $S_{(i,j)}^{tr}$ are defined in (4-16) and (4-17), respectively. For all $i \in \mathcal{N}$, the algorithm works as follows:

- **Strategy update**: Agent i updates its strategy x_i. In order to do this it needs to calculate the gradient of the cost function and project the solution onto its local constraint set.
- **Communication**:
  - Agent i sends its most recent trading strategy $S_{(i,j)}^{tr}x_i^{k+1}$ with agent j to the corresponding trading partner $j \in \mathcal{N}_i$, $\forall j \in \mathcal{N}_i$.
  - Agent i sends its local copy of λ_i to all neighbors in $\mathcal{N}_i$.
- **Dual variable update (reciprocity constraint)**: Agent i updates its local copy of the dual variable $\mu_{(i,j)}$.
- **Auxiliary variable update**: Agent i updates its auxiliary variable w_i.
- **Dual variable update (grid constraint)**: Agent i updates its local copy of the dual variable λ_i.

The advantage of the pFB over the FBF algorithm is that it only needs one evaluation of the pseudo-gradient mapping per iterative step. The disadvantage is that it needs strong monotonicity of the pseudo-gradient mapping to guarantee convergence, which is not always satisfied. Compared to the FBHF algorithm, the pFB converges under the same conditions.

5-3 Forward-backward-forward operator splitting

Where the pFB generates one sequence, the FBF scheme generates two sequences $(\boldsymbol{u}^k, \boldsymbol{v}^k)_{k\geq 0}$. First, define the operator $\overline{\mathcal{D}} = \overline{\mathcal{A}} + \overline{\mathcal{B}}$, where $\overline{\mathcal{A}}$ and $\overline{\mathcal{B}}$ are defined in (5-13). Now, the two sequences are defined as:

$$\begin{aligned} \boldsymbol{u}^k &= (\text{Id} + \Psi^{-1}\overline{\mathcal{C}})^{-1} \circ (\boldsymbol{v}^k - \Psi^{-1}\overline{\mathcal{D}}\boldsymbol{v}^k) \\ \boldsymbol{v}^{k+1} &= \boldsymbol{u}^k + \Psi^{-1}(\overline{\mathcal{D}}\boldsymbol{v}^k - \overline{\mathcal{D}}\boldsymbol{u}^k). \end{aligned} \tag{5-20}$$

The matrix Ψ is a block-diagonal matrix that contains the step sizes:

$$\Psi = \begin{bmatrix} \overline{\alpha}^{-1} & 0 & 0 & 0 \\ 0 & \overline{\beta}^{-1} & 0 & 0 \\ 0 & 0 & \overline{\gamma}^{-1} & 0 \\ 0 & 0 & 0 & \overline{\delta}^{-1} \end{bmatrix}, \tag{5-21}$$

Algorithm 1 Preconditioned Forward-Backward

Initialization: $\forall i \in \mathcal{N}$, set:
Initial conditions: $x_i^0 \in \Omega_i$, $\mu_{(i,j)}^0 = \mathbf{0}$, $\forall j \in \mathcal{N}_i$, $w_i^0 = \mathbf{0}$ and $\lambda_i^0 \in \mathbb{R}^{2H}_{\geq 0}$

Iterate until convergence:
$\forall i \in \mathcal{N}$:

(1) Receives $S_j^{mg} x_j^k$, $\forall j \in \mathcal{N}$, then updates

$$x_i^{k+1} = \text{proj}_{\Omega_i}\left\{ x_i^k - \overline{\alpha}_i \left(\nabla_{x_i} J_i(x_i^k, S\boldsymbol{x}^k) + \sum_{j \in \mathcal{N}_j} \left((\mu_{(i,j)}^k)^\top S_{(i,j)}^{tr} \right) + (\lambda_i^k)^\top \begin{bmatrix} S_i^{mg} \\ -S_i^{mg} \end{bmatrix} \right) \right\}$$

(2) Receives $S_{(j,i)}^{tr} x_j^{k+1}$ and λ_j^k, $\forall j \in \mathcal{N}_i$, then updates:

$$\begin{aligned}
\mu_{(i,j)}^{k+1} &= \mu_{(i,j)}^k - \beta_{(i,j)} \left(S_{(i,j)}^{tr} x_i^k + S_{(j,i)}^{tr} x_j^k - 2S_{(i,j)}^{tr} x_i^{k+1} - 2S_{(j,i)}^{tr} x_j^{k+1} \right), \quad \forall j \in \mathcal{N}_i \\
w_i^{k+1} &= w_i^k + \gamma_i \sum_{j \in \mathcal{N}_i} (\lambda_i^k - \lambda_j^k) \\
\lambda_i^{k+1} &= \text{proj}_{\mathbb{R}^{2H}_{\geq 0}} \left\{ \lambda_i^k + \delta_i \left(\begin{bmatrix} S_i^{mg} \\ -S_i^{mg} \end{bmatrix} (2x_i^{k+1} - x_i^k) - \begin{bmatrix} \frac{\bar{p}^{mg}}{N} \mathbf{1}_H \\ -\frac{\bar{p}^{mg}}{N} \mathbf{1}_H \end{bmatrix} - 2w_i^{k+1} + w_i^k \right) \right\}
\end{aligned}$$

where the step sizes are defined in (5-15), (5-16), (5-17) and (5-18). The FBF algorithm that follows from these two sequences is given in Algorithm 2.

The theoretical advantage of the FBF over the pFB and FBHF is that it only needs monotonicity of the involved operators instead of strong monotonicity [23]. However, this comes at the cost of an extra communication round, because the pseudo-gradient has to be evaluated twice each iterative step.

5-4 Forward-backward-half-forward operator splitting

Just like the FBF, the FBHF operator splitting also generates two sequences, but uses a different splitting. The two sequences for the FBHF are defined as:

$$\begin{aligned}
\boldsymbol{u}^k &= (\text{Id} + \Psi^{-1}\overline{\mathcal{C}})^{-1} \circ (\boldsymbol{v}^k - \Psi^{-1}(\overline{\mathcal{A}} + \overline{\mathcal{B}})\boldsymbol{v}^k) \\
\boldsymbol{v}^{k+1} &= \boldsymbol{u}^k + \Psi^{-1}(\overline{\mathcal{B}}\boldsymbol{v}^k - \overline{\mathcal{B}}\boldsymbol{u}^k).
\end{aligned} \tag{5-22}$$

The matrix Ψ in (5-22) is defined as in (5-21). The FBHF algorithm is given in Algorithm 3.

The advantage of the FBHF compared to the FBF is that it needs only one evaluation of the pseudo-gradient mapping, so the agents only need to communicate once each iterative step. However, it needs the pseudo-gradient to be strongly monotone to provide a theoretical guarantee of convergence [23], just like the pFB algorithm.

Algorithm 2 Distributed Forward-Backward-Forward

Initialization: $\forall i \in \mathcal{N}$, set:
Initial conditions: $x_i^0 \in \Omega_i$, $\mu_{(i,j)}^0 = \mathbf{0}$, $\forall j \in \mathcal{N}_i$, $w_i^0 = \mathbf{0}$ and $\lambda_i^0 \in \mathbb{R}^{2H}_{\geq 0}$

Iterate until convergence:
$\forall i \in \mathcal{N}$:

(1) Receives $S_j^{mg} x_j^k$, $\forall j \in \mathcal{N}$ and $S_{(j,i)}^{tr} x_j^k$ and λ_j^k, $\forall j \in \mathcal{N}_i$, then updates:

$$\begin{aligned}
\tilde{x}_i^k &= \text{proj}_{\Omega_i}\left\{ x_i^k - \overline{\alpha}_i \left(\nabla_{x_i} J_i(x_i^k, S\boldsymbol{x}^k) + \sum_{j \in \mathcal{N}_i} \left((\mu_{(i,j)}^k)^\top S_{(i,j)}^{tr} \right) + (\lambda_i^k)^\top \begin{bmatrix} S_i^{mg} \\ -S_i^{mg} \end{bmatrix} \right) \right\} \\
\tilde{\mu}_{(i,j)}^k &= \mu_{(i,j)}^k + \beta_{(i,j)} \left(S_{(i,j)}^{tr} x_i^k + S_{(j,i)}^{tr} x_j^k \right), \quad \forall j \in \mathcal{N}_i \\
\tilde{w}_i^k &= w_i^k + \gamma_i \sum_{j \in \mathcal{N}_i} (\lambda_i^k - \lambda_j^k) \\
\tilde{\lambda}_i^k &= \text{proj}_{\mathbb{R}^{2H}_{\geq 0}} \left\{ \lambda_i^k + \delta_i \left(\begin{bmatrix} S_i^{mg} \\ -S_i^{mg} \end{bmatrix} x_i^k - \begin{bmatrix} \frac{\bar{p}^{mg}}{N} \mathbf{1}_H \\ -\frac{\bar{p}^{mg}}{N} \mathbf{1}_H \end{bmatrix} - w_i^k \right) \right\}
\end{aligned}$$

(2) Receives $S_j^{mg} \tilde{x}_j^k$, $\forall j \in \mathcal{N}$ and $S_j^{tr} \tilde{x}_j^k$ and $\tilde{\lambda}_i^k$, $\forall j \in \mathcal{N}_i$, then updates:

$$\begin{aligned}
x_i^{k+1} &= \tilde{x}_i^k + \overline{\alpha}_i \Bigg(\nabla_{x_i} J_i(x_i^k, S\boldsymbol{x}^k) - \nabla_{x_i} J_i(\tilde{x}_i^k, S\tilde{\boldsymbol{x}}^k) + \sum_{j \in \mathcal{N}_i} \left((\mu_{(i,j)}^k)^\top S_{(i,j)}^{tr} \right) - \sum_{j \in \mathcal{N}_i} \left((\tilde{\mu}_{(i,j)}^k)^\top S_{(i,j)}^{tr} \right) \\
&\quad + (\lambda_i^k - \tilde{\lambda}_i^k)^\top \begin{bmatrix} S_i^{mg} \\ -S_i^{mg} \end{bmatrix} \Bigg) \\
\mu_{(i,j)}^{k+1} &= \tilde{\mu}_{(i,j)}^k - \beta_{(i,j)} \left(S_{(i,j)}^{tr} x_i^k + S_{(j,i)}^{tr} x_j^k - S_{(i,j)}^{tr} \tilde{x}_i^k - S_{(j,i)}^{tr} \tilde{x}_j^k \right), \quad \forall j \in \mathcal{N}_i \\
w_i^{k+1} &= \tilde{w}_i^k - \gamma_i \sum_{j \in \mathcal{N}_i} \left((\lambda_i^k - \lambda_j^k) - (\tilde{\lambda}_i^k - \tilde{\lambda}_j^k) \right) \\
\lambda_i^{k+1} &= \tilde{\lambda}_i^k + \delta_i \left(\begin{bmatrix} S_i^{mg} \\ -S_i^{mg} \end{bmatrix} (x_i^k - \tilde{x}_i^k) - w_i^k + \tilde{w}_i^k \right)
\end{aligned}$$

5-5 Stochastic algorithms

In the previous sections, the algorithms are derived for the case where the total demand is assumed to be exactly known for each agent. However, in reality the day-ahead demand can only be estimated. Therefore, this section shows the derivation of the stochastic case until we arrive at the monotone inclusion $\mathcal{T}_e$. The derivations for the pFB, FBF and FBHF algorithms can be done in the same way as shown in Section 5-2, Section 5-3 and Section 5-4, respectively. Because there is a lot of data available, the stochastic approximation (SA) scheme with variance reduction is used. Also, because of the increasing number of samples, the SA scheme with variance reduction needs less strong assumptions on the mappings involved than the SA with only one realization of the cost function [24].

Algorithm 3 Distributed Forward-Backward-Half-Forward

Initialization: $\forall i \in \mathcal{N}$, set:
Initial conditions: $x_i^0 \in \Omega_i$, $\mu_{(i,j)}^0 = \mathbf{0}$, $\forall j \in \mathcal{N}_i$, $w_i^0 = \mathbf{0}$ and $\lambda_i^0 \in \mathbb{R}_{\geq 0}^{2H}$

Iterate until convergence:
$\forall i \in \mathcal{N}$:

(1) Receives $S_j^{mg} x_j^k$, $\forall j \in \mathcal{N}$ and $S_j^{tr} x_j^k$ and λ_j^k, $\forall j \in \mathcal{N}_i$, then updates:

$$\begin{aligned}
\tilde{x}_i^k &= \text{proj}_{\Omega_i}\left\{ x_i^k - \overline{\alpha}_i \left(\nabla_{x_i} J_i(x_i^k, S\boldsymbol{x}^k) + \sum_{j \in \mathcal{N}_i} \left((\mu_{(i,j)}^k)^\top S_{(i,j)}^{tr} \right) + (\lambda_i^k)^\top \begin{bmatrix} S_i^{mg} \\ -S_i^{mg} \end{bmatrix} \right) \right\} \\
\tilde{\mu}_{(i,j)}^k &= \mu_{(i,j)}^k + \beta_{(i,j)} \left(S_{(i,j)}^{tr} x_i^k + S_{(j,i)}^{tr} x_j^k \right), \quad \forall j \in \mathcal{N}_i \\
\tilde{w}_i^k &= w_i^k + \gamma_i \sum_{j \in \mathcal{N}_i} (\lambda_i^k - \lambda_j^k) \\
\tilde{\lambda}_i^k &= \text{proj}_{\mathbb{R}_{\geq 0}^{2H}} \left\{ \lambda_i^k + \delta_i \left(\begin{bmatrix} S_i^{mg} \\ -S_i^{mg} \end{bmatrix} x_i^k - \begin{bmatrix} \frac{\bar{p}^{mg}}{N} \mathbf{1}_H \\ -\frac{\bar{p}^{mg}}{N} \mathbf{1}_H \end{bmatrix} - w_i^k \right) \right\}
\end{aligned}$$

(2) Receives $S_j^{tr} \tilde{x}_j^k$ and $\tilde{\lambda}_j^k$, $\forall j \in \mathcal{N}_i$, then updates:

$$\begin{aligned}
x_i^{k+1} &= \tilde{x}_i^k + \overline{\alpha}_i \left(\sum_{j \in \mathcal{N}_i} \left((\mu_{(i,j)}^k - \tilde{\mu}_{(i,j)}^k)^\top S_{(i,j)}^{tr} \right) + (\lambda_i^k - \tilde{\lambda}_i^k)^\top \begin{bmatrix} S_i^{mg} \\ -S_i^{mg} \end{bmatrix} \right) \\
\mu_{(i,j)}^{k+1} &= \tilde{\mu}_{(i,j)}^k - \beta_{(i,j)} \left(S_{(i,j)}^{tr} x_i^k + S_{(j,i)}^{tr} x_j^k - S_{(i,j)}^{tr} \tilde{x}_i^k - S_{(j,i)}^{tr} \tilde{x}_j^k \right), \quad \forall j \in \mathcal{N}_i \\
w_i^{k+1} &= \tilde{w}_i^k - \gamma_i \sum_{j \in \mathcal{N}_i} \left((\lambda_i^k - \lambda_j^k) - (\tilde{\lambda}_i^k - \tilde{\lambda}_j^k) \right) \\
\lambda_i^{k+1} &= \tilde{\lambda}_i^k - \delta_i \left(\begin{bmatrix} S_i^{mg} \\ -S_i^{mg} \end{bmatrix} (x_i^k - \tilde{x}_i^k) - w_i^k + \tilde{w}_i^k \right)
\end{aligned}$$

The derivation is the same as shown in Section 5-1, but now each cost function J_i is an expected value function $\mathbb{J}_i$, which can be defined as

$$\mathbb{J}_i(x_i, S\boldsymbol{x}) = \mathbb{E}[J_i(x_i, S\boldsymbol{x}, \xi)], \tag{5-23}$$

where ξ is the random variable. The stochastic GNE (SGNE) can be defined as

$$\mathbb{J}_i(x_i, S\boldsymbol{x}) \leq \inf\{\mathbb{J}_i(y, S\boldsymbol{x}) \mid y \in \mathcal{X}_i\}, \tag{5-24}$$

where $\mathcal{X}_i$ is defined in (5-1). As in the deterministic case, we seek a stochastic v-GNE (v-SGNE) by studying the associated stochastic VI (SVI), which reads as

$$\mathbb{F}(\boldsymbol{x}^*)^T (\boldsymbol{x} - \boldsymbol{x}^*) \geq 0, \quad \forall \boldsymbol{x} \in \boldsymbol{\mathcal{X}}, \tag{5-25}$$

where

$$\mathbb{F}(\boldsymbol{x}) \equiv \begin{pmatrix} \nabla_{x_1}\mathbb{J}_1(x_1, S\boldsymbol{x}) \\ \vdots \\ \nabla_{x_N}\mathbb{J}_N(x_N, S\boldsymbol{x}) \end{pmatrix}, \tag{5-26}$$

and $\boldsymbol{\mathcal{X}}$ as defined in (5-2). To recast the stochastic GNEP (SGNEP) into a monotone inclusion, first, define the Lagrangian of the of the SGNEP for each agent i:

$$\mathcal{L}_i(\boldsymbol{x}, \mu_i, \lambda_i) = \mathbb{J}_i(x_i, S\boldsymbol{x}) + I_{\Omega_i}(x_i) + \mu_i^\top R\boldsymbol{x} + \lambda_i^\top(A\boldsymbol{x} - b). \tag{5-27}$$

The vector $\boldsymbol{x}^*$ is a SGNE if and only if the following KKT conditions are satisfied:

$$\begin{cases} \mathbf{0}_{n_i} \in \mathbb{E}[\nabla_{x_i} J_i(x_i^*, S\boldsymbol{x}^*, \xi)] + \mathrm{N}_{\Omega_i}(x_i^*) + R_i^\top \mu_i^* + A_i^\top \lambda_i^* \\ \mathbf{0}_{EH} \in R\boldsymbol{x}^* \\ \mathbf{0}_{2H} \in \mathrm{N}_{\mathbb{R}^{2H}_{\geq 0}}(\lambda_i^*) - (A\boldsymbol{x}^* - b). \end{cases} \tag{5-28}$$

Since the collective strategy $\boldsymbol{x}^*$ is a solution of the VI($\boldsymbol{\mathcal{X}}, \mathbb{F}$) if and only if

$$\boldsymbol{x}^* \in \arg\min_{\boldsymbol{y} \in \boldsymbol{\mathcal{X}}} (\boldsymbol{y} - \boldsymbol{x}^*)^\top \mathbb{F}(\boldsymbol{x}^*), \tag{5-29}$$

the KKT conditions of the SVI read as

$$\begin{cases} \mathbf{0}_{n_i} \in \mathbb{E}[\nabla_{x_i} J_i(x_i^*, S\boldsymbol{x}^*, \xi)] + \mathrm{N}_{\Omega_i}(x_i^*) + R_i^\top \mu^* + A_i^\top \lambda^* \\ \mathbf{0}_{EH} \in R\boldsymbol{x}^* \\ \mathbf{0}_{2H} \in \mathrm{N}_{\mathbb{R}^{2H}_{\geq 0}}(\lambda^*) - (A\boldsymbol{x}^* - b). \end{cases} \tag{5-30}$$

As in the deterministic case, we can rewrite the KKT conditions in (5-30) in a more compact form as

$$\mathcal{T}(\boldsymbol{x}, \mu, \boldsymbol{\lambda}) : \begin{bmatrix} \boldsymbol{x} \\ \mu \\ \boldsymbol{\lambda} \end{bmatrix} \mapsto \begin{bmatrix} \mathbb{F}(x) + \mathrm{N}_{\boldsymbol{\Omega}}(\boldsymbol{x}) + R^\top \mu + \overline{A}^\top \boldsymbol{\lambda} \\ -R\boldsymbol{x} \\ \mathrm{N}_{\mathbb{R}^{2HN}_{\geq 0}}(\boldsymbol{\lambda}) - (\overline{A}\boldsymbol{x} - \overline{b}) \end{bmatrix}, \tag{5-31}$$

with $\mathbb{F}$ as defined in (5-26). We assume that each agent has access to a pool of samples of the random variable and is able to compute an approximation of $\mathbb{E}[\nabla_{x_i} J_i(x_i^*, S\boldsymbol{x}^*, \xi)]$ [18].

The v-SGNE we seek is the equilibrium for which the agents reach consensus on the dual variables λ_i. To impose this consensus on the dual variables the operator $\mathcal{T}$ is extended by an auxiliary variable z:

$$\mathcal{T}_e(\boldsymbol{x}, \mu, \boldsymbol{z}, \boldsymbol{\lambda}) : \begin{bmatrix} \boldsymbol{x} \\ \mu \\ \boldsymbol{z} \\ \boldsymbol{\lambda} \end{bmatrix} \mapsto \begin{bmatrix} \mathbb{F}(x) + \mathrm{N}_{\boldsymbol{\Omega}}(\boldsymbol{x}) + R^\top \mu + \overline{A}^\top \boldsymbol{\lambda} \\ -R\boldsymbol{x} \\ -\overline{V}\boldsymbol{\lambda} \\ \mathrm{N}_{\mathbb{R}^{2HN}_{\geq 0}}(\boldsymbol{\lambda}) - (\overline{A}\boldsymbol{x} - \overline{b}) + \overline{V}^\top \boldsymbol{z} \end{bmatrix}. \tag{5-32}$$

The derivation of the stochastic version of the pFB, FBF and FBHF is done in the same way as shown in Section 5-2, Section 5-3 and Section 5-4, respectively.

Chapter 6

Simulation and results

This chapter discusses the simulations that were performed to test the algorithms derived in the previous chapter. First, Section 6-1 presents the energy data that is used in the simulations. As stated in the introduction, the main goal of the algorithms is to achieve a more stable operation of the grid while minimizing the cost of power usage. Section 6-2 shows these main results of the algorithms. As described in Section 5-1-1, either the Laplacian matrix or the incidence matrix of the communication graph $\mathcal{G}$ can be used to force consensus on the dual variable λ. Section 6-3 compares the performance of these two matrices. Next, Section 6-4 shows how the algorithms behave for different amounts of agents. As described in Section 5-5, in a realistic model the exact day-ahead demand cannot be known and has to be modeled as a random variable. Section 6-5 shows the performance of the algorithms using this uncertainty in the demand. Finally, Section 6-6 discusses a method to make more optimal use of the storage units.

6-1 Demand data

To simulate a realistic scenario of a neighborhood, different types of agents are taken into account. In the simulations performed in the next sections, six types of customers are considered. There are three different types of residential customers and three types of commercial customers. The three residential customers are based on their total power usage each day and are divided in low-, medium- and high-scale users. For the commercial customers, a primary school, a small office and a small retail store are used.

The data is taken from the OpenEI platform [34]. The primary school, office and retail load profiles have been scaled down for the simulations because of the limited total number of agents. For the simulations one office, one primary school and one retail store is used. The rest of the agents are randomly taken from the low-, medium- and high-scale residential profiles. Only the large-scale residential customers, the primary school, the office building and retail store have dispatchable generation units. The larger the energy profile, the lower

the cost of dispatchable generation. Figure 6-1 shows what the different load profiles look like.

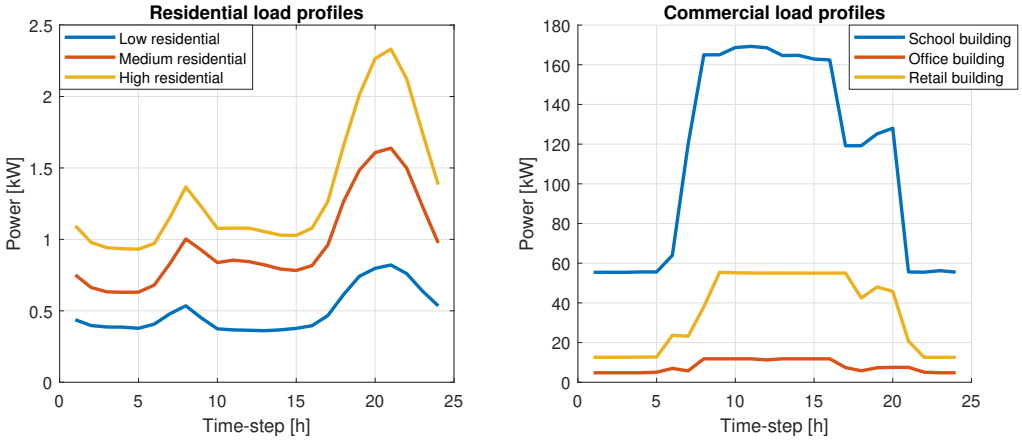


Figure 6-1: Unscaled load profiles from the OpenEI database

6-2 Main results

Section 6-2-1 presents the main results of the three algorithms derived in Chapter 5, which is to obtain a stable operation of the grid, while minimizing the cost of power usage. Section 6-2-2 shows the performance of the algorithms in terms of convergence of the primal and dual residuals associated with the reciprocity constraint. As stated in Section 5-1, where the forward-backward-forward (FBF) only needs plain monotonicity of the pseudo-gradient mapping $F(\boldsymbol{x})$ (defined in (5-10)) to guarantee convergence, the preconditioned forward-backward (pFB) and forward-backward-half-forward (FBHF) algorithms need strong monotonicity of $F(\boldsymbol{x})$ to guarantee convergence, which is not satisfied in this case. However, the results in Section 6-2-2 show that also these two algorithms converge towards a solution. The simulations were performed with Matlab R2020a using the Yalmip toolbox [35] with the quadprog solver. In this section, all simulations were performed for a 10-agent network.

6-2-1 Total load vs. dispatchable generation and import from main grid

The main goal of the algorithms is to obtain a stable operation of the grid while minimizing the cost of power usage. Figure 6-2, Figure 6-3 and Figure 6-4 show the total power imported from the main grid, total power produced by dispatchable generation and the total load at each time step for the pFB, FBF and FBHF algorithm, respectively. For the scenario where none of the agents have storage units, the demand has to be met by importing from the main grid and by dispatchable generation, if present. When four of the agents have storage units the load on the main grid gets flat and peaks in the demand are accounted for by the storage units. When all agents have storage the load on the main grid stays flat, but gets lower because more power can be accounted for by the storage units.

Besides a stable operation of the grid, we also wanted to achieve a reduction in cost. Table 6-1 shows the total cost for each scenario. It can be observed that by using storage units to reduce

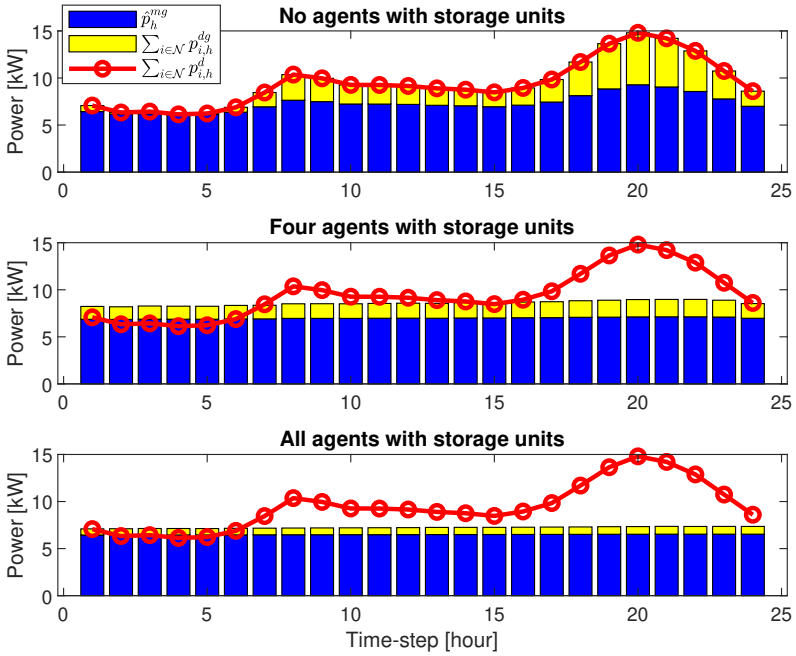


Figure 6-2: Total power imported from the main grid and produced by dispatchable generation units vs. total load in a 10-agent network using the pFB algorithm

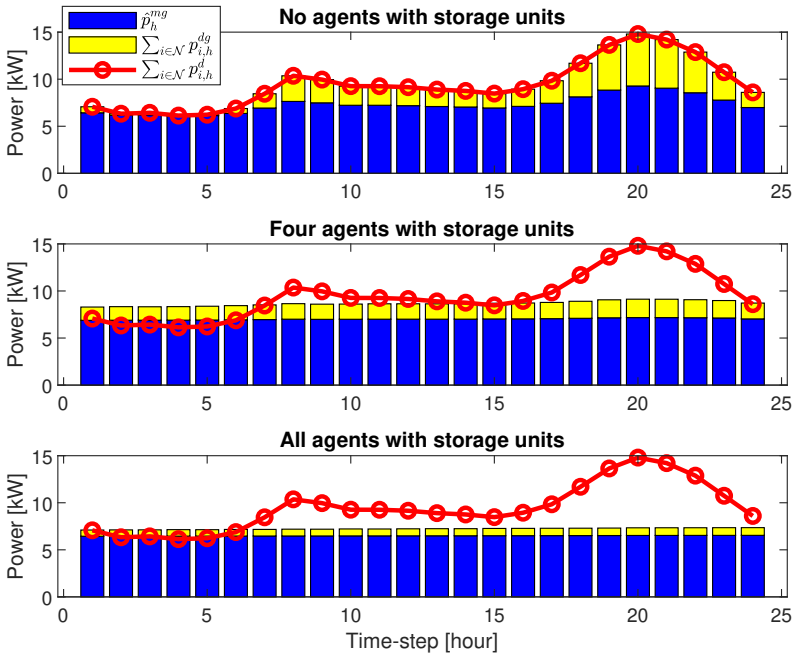


Figure 6-3: Total power imported from the main grid and produced by dispatchable generation units vs. total load in a 10-agent network using the FBF algorithm

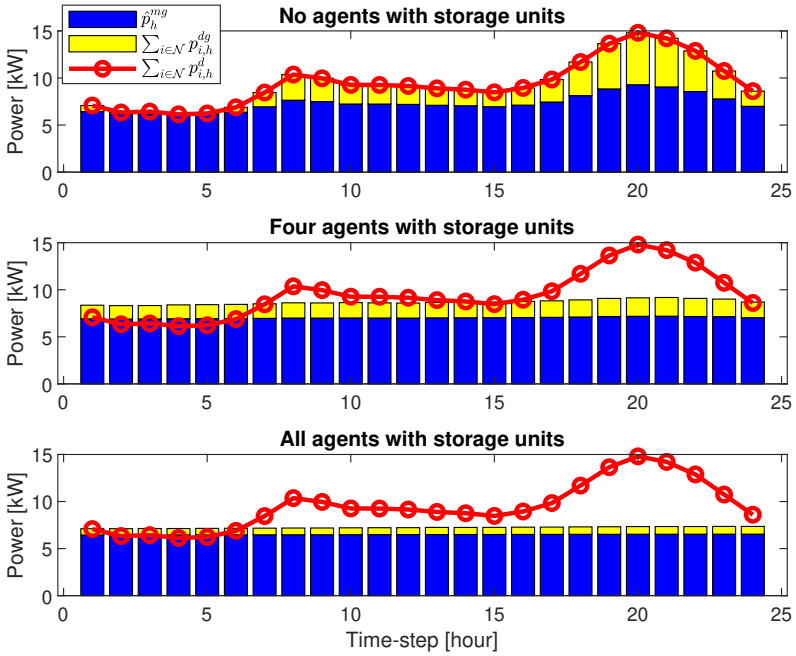


Figure 6-4: Total power imported from the main grid and produced by dispatchable generation units vs. total load in a 10-agent network using the FBHF algorithm

the load on the main grid in peak hours also the cost is reduced.

Scenario	No agents storage	4 agents storage	10 agents storage
pFB	100%	-6.63%	-12.29%
FBF	100%	-6.19%	-12.30%
FBHF	100%	-6.11%	-12.29%

Table 6-1: Total cost per algorithm for each scenario

6-2-2 Stopping criteria

The stopping criteria used in the simulations are the primal and dual residuals associated with the reciprocity constraint, i.e., $||\text{col}(\{S^{tr}_{(i,j)}u^k_i + S^{tr}_{(j,i)}u^k_j\}_{\forall j\in\mathcal{N}_i, i\in\mathcal{N}})||_2 \leq 0.001$ and $||\text{col}(\{u^{k+1}_i - u^k_i\}_{i\in\mathcal{N}})||_2 \leq 0.01$, respectively. Figure 6-5 shows the convergence of the primal residual for the three algorithms for each of the three scenario's. As expected, the primal residuals all converge towards the stopping value. Figure 6-6 shows the convergence of the dual residual for the three algorithms for each of the three scenario's. Also the dual residuals all converge towards the stopping value. As stated in Chapter 3, the solution we are looking for is the variational solution of the game, i.e., the solution where $\lambda_1 = \ldots = \lambda_N$, $\forall i \in \mathcal{N}$. Figure 6-7 shows the disagreement on this dual variable λ for the three algorithms for each of the three scenarios. The disagreement is given by $||(L \otimes I_{2H})\lambda^k)||$, where L is the Laplacian matrix of the communication graph $\mathcal{G}$, I_{2H} is a $(2H \times 2H)$ identity matrix and H denotes the number

of time steps. It can be observed that this value gets smaller over time, which means the individual λ_i's are converging towards the same value.

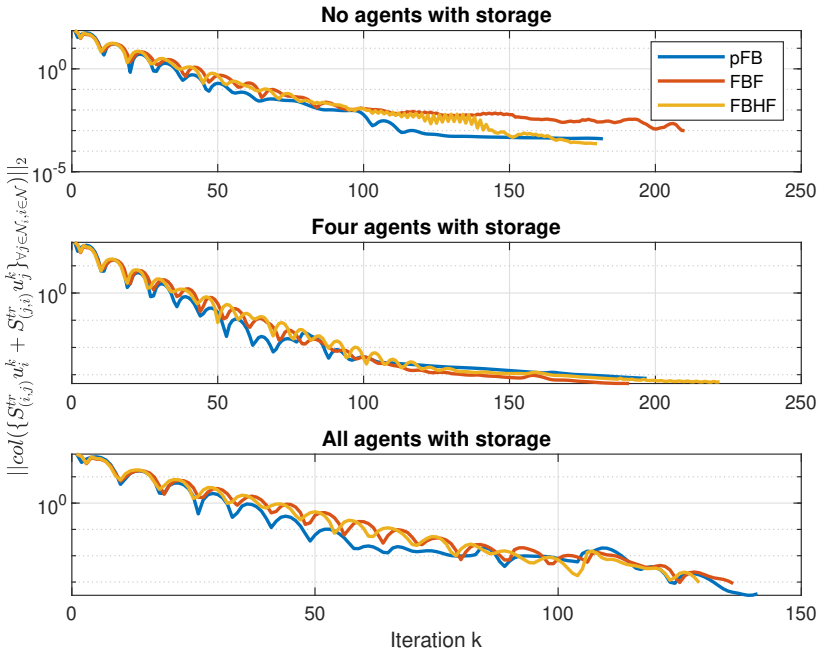


Figure 6-5: Primal residual for each scenario for all three algorithms

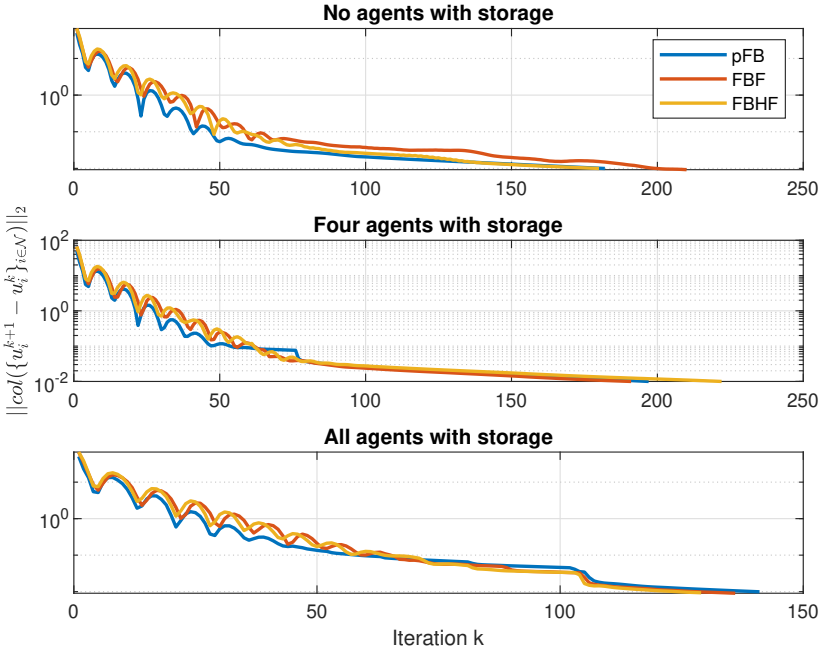


Figure 6-6: Dual residual for each scenario for all three algorithms

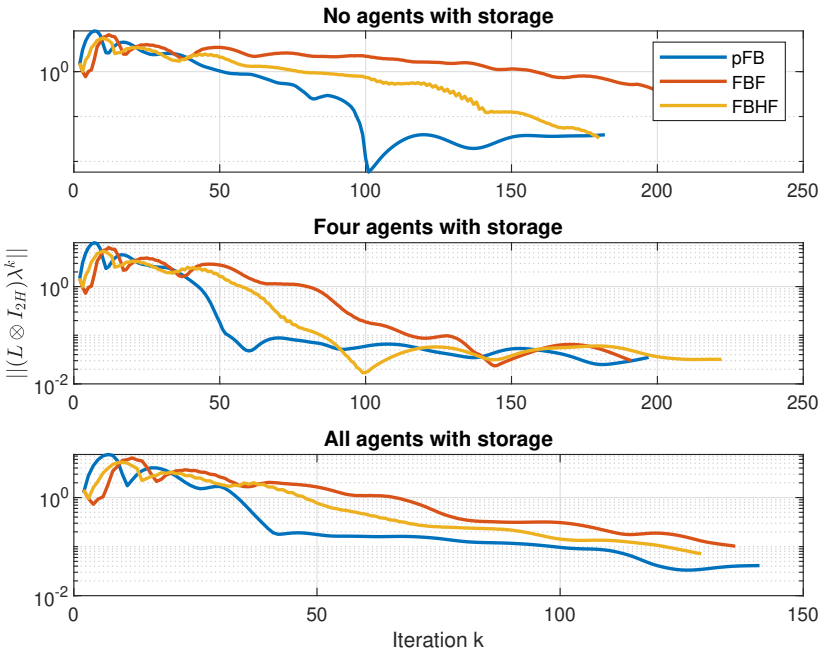


Figure 6-7: Disagreement on dual variable λ for each scenario for all three algorithms

6-3 Incidence matrix vs. Laplacian matrix

As described in Section 5-1-1, two different matrices can be used to force consensus on the dual variable λ: the Laplacian matrix L or the incidence matrix V of the communication graph $\mathcal{G}$. In all other simulations in this chapter the incidence matrix V was used because it needs one communication step less per iteration. To show the difference in simulation between the two matrices, Figure 6-8 shows the number of iterations and the elapsed CPU time for both versions of the algorithms for different levels of connectivity for a 10-agent network. The level of connectivity is defined as the number of communication links between the agents divided by the total possible number of communication links. Four different levels of connectivity are tested, which are 0.222, 0.5, 0.75 and 1. A connectivity level of 0.222 is a ring graph in a 10-agent network, a connectivity level of 1 means a fully connected graph where each agent has a connection with every other agent. In the legend of Figure 6-8 the algorithms derived with the incidence matrix have their regular name and the algorithms derived with the Laplacian matrix have a subscript L added. The communication links were created randomly each simulation. The results in Figure 6-8 show the average results over three simulations. It can observed that for a 10-agent network, using a ring graph results in more iterations, where the other three levels of graph connectivity show roughly the same amount. Although the amount of iterations does not differ significantly between a connectivity level of 0.5 0.75 and 1, the average CPU time does increase for each algorithm for a connectivity of 0.75 and 1 compared to a connectivity of 0.5. Between the two different versions of each algorithm no real trend is visible for the different levels of connectivity. Except when using a ring graph, the performance between each two versions is similar. Comparing all algorithms, both versions

of the FBHF algorithm are slightly faster than the other algorithms, and the two versions of the pFB are the slowest, except when using a ring graph.

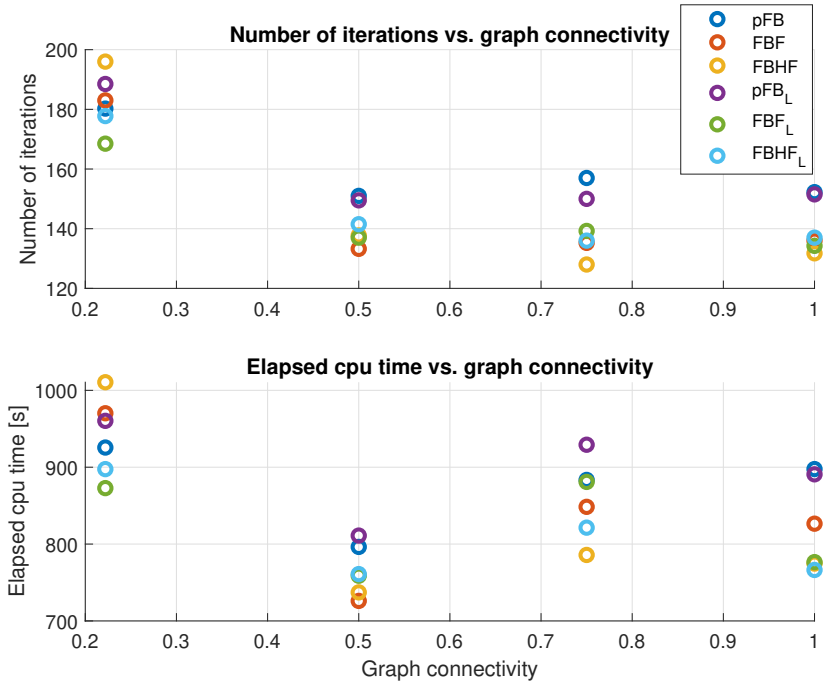


Figure 6-8: Number of iterations and elapsed cpu time vs. the level of connectivity of the communication graph $\mathcal{G}$ (achieved on Matlab R2020a with a 1.8 GHz Intel Core i7 and 16 GB DDR4 RAM)

6-4 Scalability

To test how the algorithms behave with an increasing number of agents, simulations were done using a 10-, 20-, and 30-agent network, shown in Figure 6-9. For all simulations the graph connectivity was taken to be 0.5, and the same stopping criteria were used as in Section 6-2, i.e., $||\text{col}(\{S^{tr}_{(i,j)}u^k_i + S^{tr}_{(j,i)}u^k_j\}_{\forall j\in\mathcal{N}_i, i\in\mathcal{N}})||_2 \leq 0.001$ and $||\text{col}(\{u^{k+1}_i - u^k_i\}_{i\in\mathcal{N}})||_2 \leq 0.01$. The results in Figure 6-9 represent the average of three simulations. It can be observed that both the amount of iterations and the elapsed CPU time increase as the number of agents increases. This is to be expected, since the strategy vector gets larger and there are more agents that have to communicate with each other. In particular, where the amount of iterations not even doubles from a 10- to a 30-agent network, the elapsed cpu time is roughly 20 times as high. Also noticeable is that for the 30-agent network, the amount of iterations and cpu time of the FBF increase more than that of the pFB and FBHF. This is probably due to the fact that the FBF needs to evaluate the gradient twice each iterative step, instead of once.

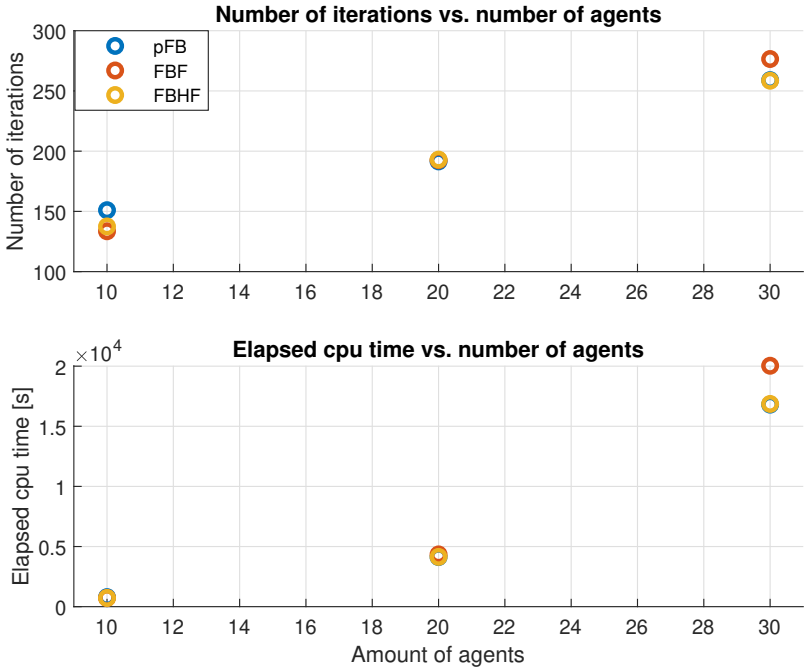


Figure 6-9: Number of iterations and elapsed cpu time vs. the amount of agents in the network (achieved on Matlab R2020a with a 1.8 GHz Intel Core i7 and 16 GB DDR4 RAM)

6-5 Stochastic demand

In the simulations done until now, it was assumed all agents have exact knowledge of the total demand profile for the day ahead. However, in reality the demand for the day ahead cannot be exactly known, but has to be estimated. Therefore this section discusses the case where the total demand for the day ahead is modeled as a random variable. When using this stochastic demand, the algorithms are not guaranteed to converge anymore. This section shows that in the simulations the algorithms still converge towards a solution.

For all three algorithms the cost function is approximated using the stochastic approximation (SA) scheme with variance reduction, because it is assumed there is a lot of data available (see Section 5-5). Each agent has access to an increasing number S_k of samples of the random variable ξ and is able to compute an approximation of $\mathbb{F}(\boldsymbol{x})$ [18]:

$$\begin{aligned}\hat{F}(\boldsymbol{x},\boldsymbol{\xi}) &= F_{SA}(\boldsymbol{x},\boldsymbol{\xi}) \\ &= \operatorname{col}\left(\frac{1}{S_k}\sum_{t=1}^{S_k}\nabla_{x_1}J_1(x_1,S\boldsymbol{x},\xi_1^{(t)}),\ldots,\frac{1}{S_k}\sum_{t=1}^{S_k}\nabla_{x_N}J_N(x_N,S\boldsymbol{x},\xi_N^{(t)})\right),\end{aligned} \tag{6-1}$$

where $\boldsymbol{\xi} = \operatorname{col}(\bar{\xi}_1,\ldots,\bar{\xi}_N)$, $\forall i \in \mathcal{N}$, $\bar{\xi}_i = \operatorname{col}(\xi_i^{(t)},\ldots,\xi_i^{(S_k)})$ and $\boldsymbol{\xi}$ is an i.i.d. sequence of random variables drawn from a normal distribution.

Because in the stochastic case the residuals do not converge as well as in the deterministic case, the stopping criteria are relaxed to $||\operatorname{col}(\{S^{tr}_{(i,j)}u_i^k + S^{tr}_{(j,i)}u_j^k\}_{\forall j\in\mathcal{N}_i, i\in\mathcal{N}})||_2 \leq 0.01$ and

$||\text{col}(\{u_i^{k+1} - u_i^k\}_{i\in\mathcal{N}})||_2 \leq 0.1$. The simulations were carried out using a 10-agent network and a graph connectivity of 0.5. Figure 6-10 and Figure 6-11 show the convergence of the primal and dual residuals, respectively. Figure 6-12 shows the disagreement on the dual variable λ. It can be observed that the two residuals and the disagreement on λ still converge in this stochastic case, although at a slower rate than in the deterministic case. Also, they keep oscillating, which is to be expected because each agent uses a slightly different approximation of the demand. Also noticeable is that the FBF algorithm needs significantly more iterations to converge to the stopping criteria than the pFB and the FBHF algorithms for all three scenario's. In the deterministic case in Section 6-2-2 there was not such a difference.

To make the comparison more clear with the deterministic case, Figure 6-13 shows a plot of the convergence of the primal and dual residual of the FBF algorithm for the scenario where four agents have storage units for both the deterministic and stochastic case. As one can see, the stochastic version of the algorithm converges at a slower rate than the deterministic one.

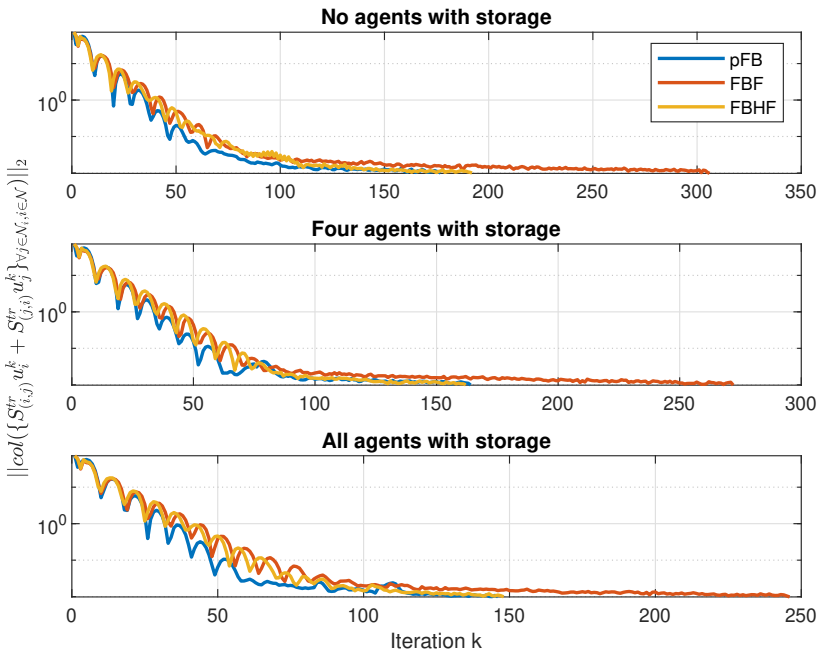


Figure 6-10: Primal residual for each scenario for all three algorithms for the stochastic case

Instead of using a normal distribution for the stochastic demand, the algorithms were also tested when the numbers were drawn from a uniform distribution. The results are shown in Figure 6-14. Because the results for the three scenario's are similar, only the results for one of the three scenario's is shown when using a uniform distribution. It can be observed that when using a uniform distribution, the algorithms do not reach the stopping criterion for the primal residual used for the normal distribution. This is to be expected, because when drawing from a uniform distribution the approximations of the different agents differ more from each other.

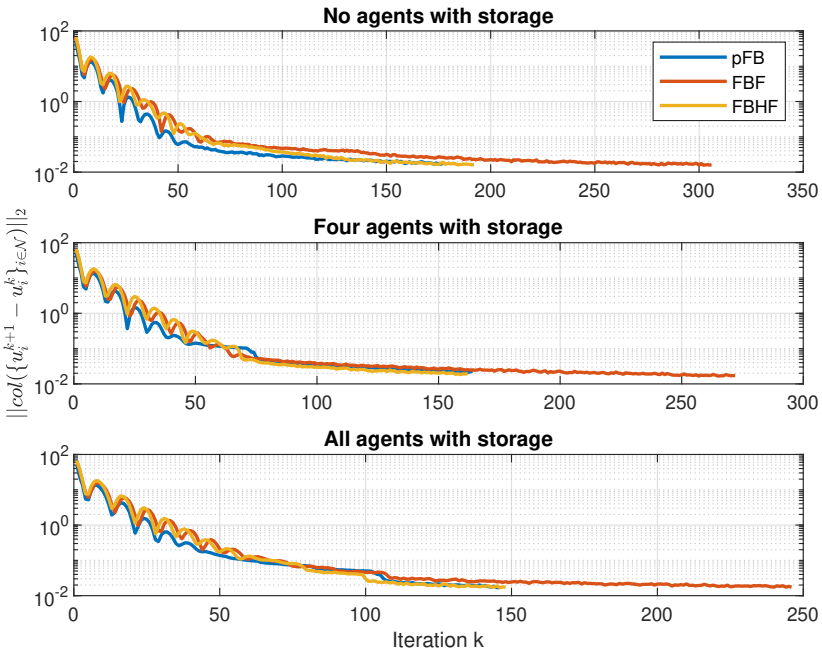


Figure 6-11: Dual residual for each scenario for all three algorithms for the stochastic case

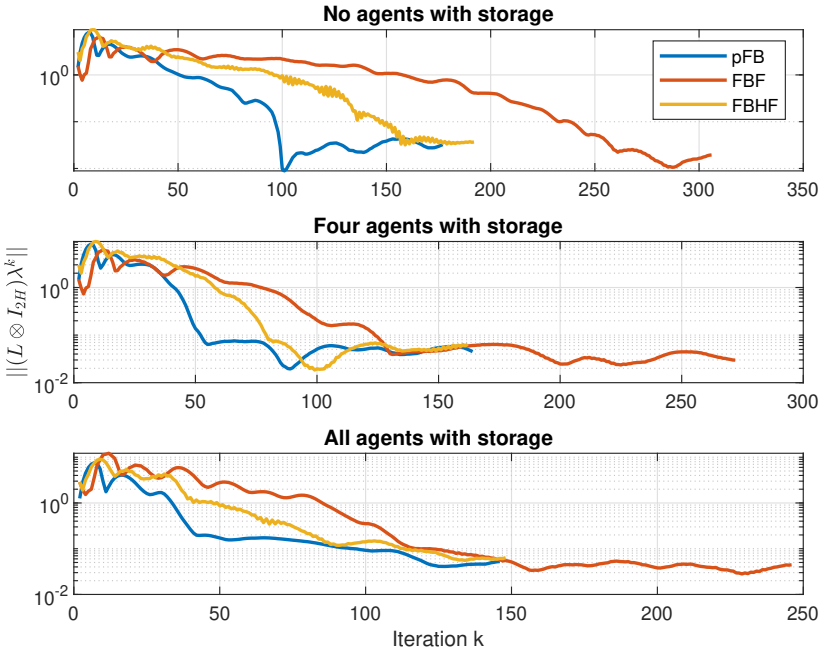


Figure 6-12: Disagreement on dual variable λ for each scenario for all three algorithms for the stochastic case

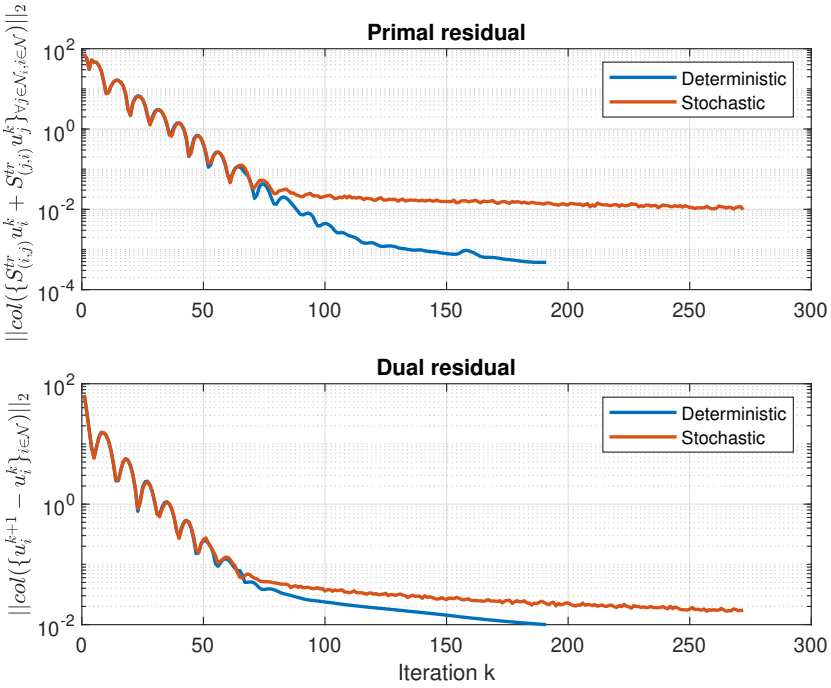


Figure 6-13: Primal and dual residual of the deterministic case vs. the stochastic case for the FBF algorithm with four agents with storage

6-6 Storage constraints

The simulations until now were performed for a single day, which means it might happen that the best strategy is to use all of the available battery capacity on that day, leaving no capacity for the start of the next day. However, one could imagine a situation in which some days have a higher total demand, which means it could be beneficial to save some battery capacity on a certain day for later days. To make more optimal use of the battery, an extra constraint can be imposed. First, Section 6-6-1 and Section 6-6-2 discuss two methods that were tested, but did not result in a better performance of the algorithm. For the sake of completeness, they are added in this section. Section 6-6-3 discusses a method that did result in a reduction of the cost.

6-6-1 Hard constraint

In order to discourage too much usage of the storage unit in a day, a constraint can be imposed that enforces the final state of charge (SOC) to be the same as the initial SOC:

$$s_{i,H} = s_{i,0}, \tag{6-2}$$

where H denotes the last time step of a day. However, this constraint is also enforced on days with higher demand. Since the goal was to save battery power for days with higher demand, this constraint is too restrictive. A better way would be to have a soft constraint, which allows for more flexibility.

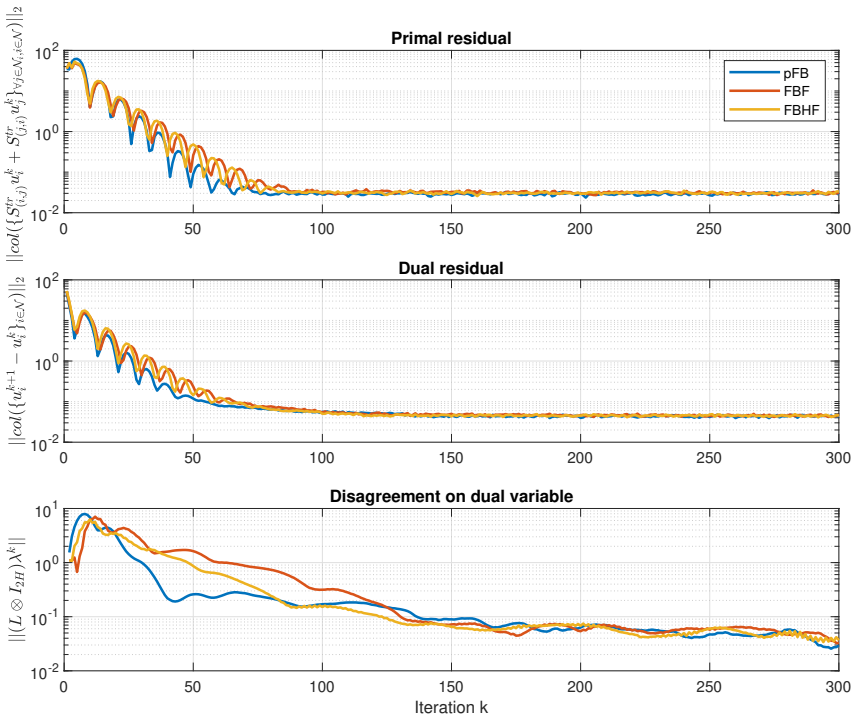


Figure 6-14: Primal and dual residual and disagreement on the dual variable λ when using a uniform distribution for all three algorithms for the scenario where all agents have storage units

6-6-2 Soft constraint

The soft constraint penalizes a deviation from the initial SOC at the end of the day. This is added as an extra term in the cost function:

$$f_i^{st} = \sum_{h \in \mathcal{H}} \left(q_i^{st} (p_{i,h}^{st})^2 + c_i^{st} p_{i,h}^{st} \right) + d^{st}(s_{i,0} - s_{i,H}), \tag{6-3}$$

where s_i, h is the SOC at time step h as defined in (4-6) and $d^{st} \in \mathbb{R}$ is a penalty factor. This way, the more the SOC at the end of the day deviates from the initial SOC, the higher the penalty gets. Another element is added to this penalty that also penalizes long term use of the storage unit. This is done by increasing the penalty factor by a certain amount each day the SOC does not reach the initial value of 0.5. The more days an agent has less than the initial SOC at the end of the day, the higher the penalty gets.

Both the hard and soft constraint were tested with the pFB algorithm on a data set that contains the demand profiles of 8 consecutive days, where day 3 and 4 have a higher total demand than the other days, see Figure 6-15. On the first day, all the agents start with a SOC of 0.5 and each consecutive day they start with the SOC with which they ended on the day before. This way the strategy is determined over all 8 days. Table 6-2 shows the cost of both the hard and soft constraint compared to using no extra constraint. As expected, the soft constraint performs better than the hard constraint, but both methods perform worse than having no extra constraint. Figure 6-16 shows the total cost and average SOC at the

end of each day. It can be observed that the hard constraint results in a higher cost every day and the soft constraint only results in a lower cost on day 3.

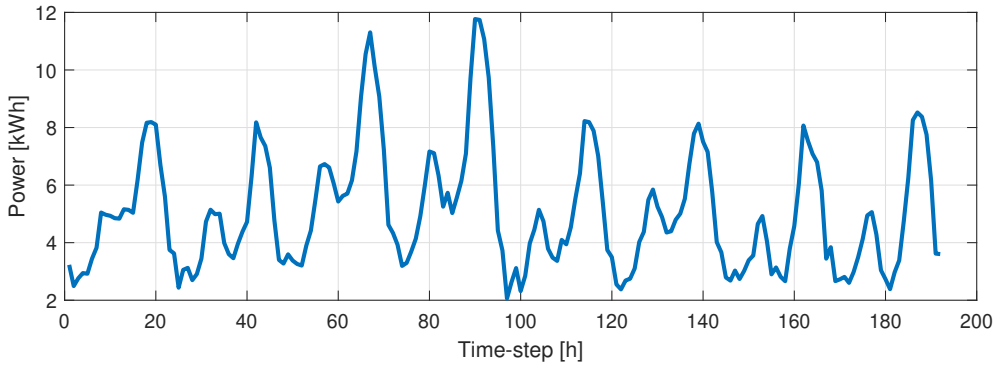


Figure 6-15: Demand profile used for the simulations

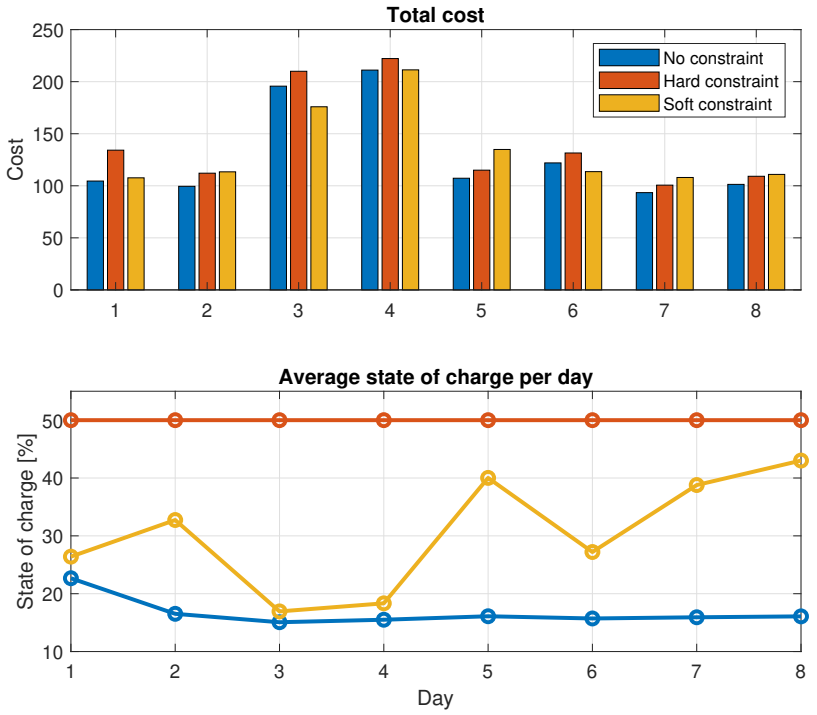


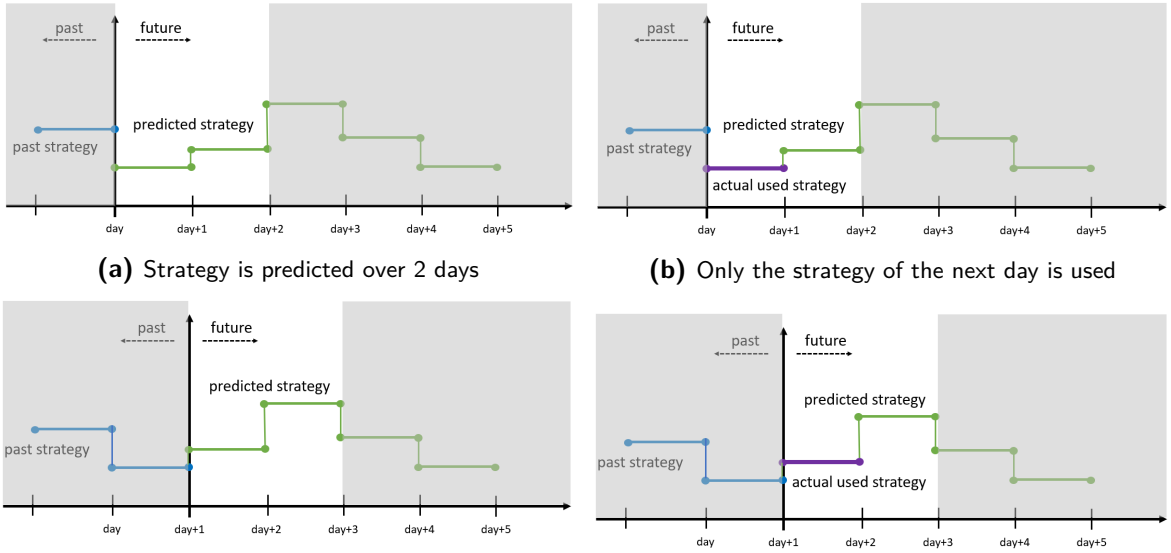
Figure 6-16: Total cost and average state of charge per day when using a hard and soft constraint compared to using no constraint

No constraint	Hard constraint	Soft constraint
100%	+6.91%	+1.78%

Table 6-2: Total cost when using a hard and soft constraint compared to using no constraint for the pFB algorithm

6-6-3 Forecast

A better way to account for days with more demand is to use the forecast of more than one day ahead when calculating the optimal strategy for a certain day, just like in model predictive control (MPC). In the simulations in Section 6-2 to Section 6-5, a single day was simulated where only the (expected) demand of that particular day was used when determining the strategy. To account for days with higher demand, instead of only looking at the expected demand of the day for which the strategy has to be determined, each agent also uses the expected demand of days in the future to determine its strategy. This way the algorithm optimizes a strategy over multiple days, but only implements the strategy of the upcoming day. Figure 6-17 shows an overview of this methodology with a prediction horizon of 2 days and a control horizon of 1 day. In Figure 6-17a the strategy is predicted over a period of 2 days. From this strategy, only the strategy of the upcoming day is used, which can be seen in Figure 6-17b. After that day, the algorithm moves one day forward and the same process is repeated , shown in Figure 6-17c and Figure 6-17c. In this example a control horizon of 1 day is used, which consists of 24 time steps in the simulations. The control horizon could also be reduced so less time steps are actually used from each strategy. However, this way a new strategy has to be calculated more often which would increase simulation time.



(a) Strategy is predicted over 2 days

(b) Only the strategy of the next day is used

(c) Horizon is moved one day forward, again predicting the strategy over 2 days

(d) Again only the strategy of the next day is used

Figure 6-17: Overview of using a prediction horizon of 2 days and control horizon of 1 day

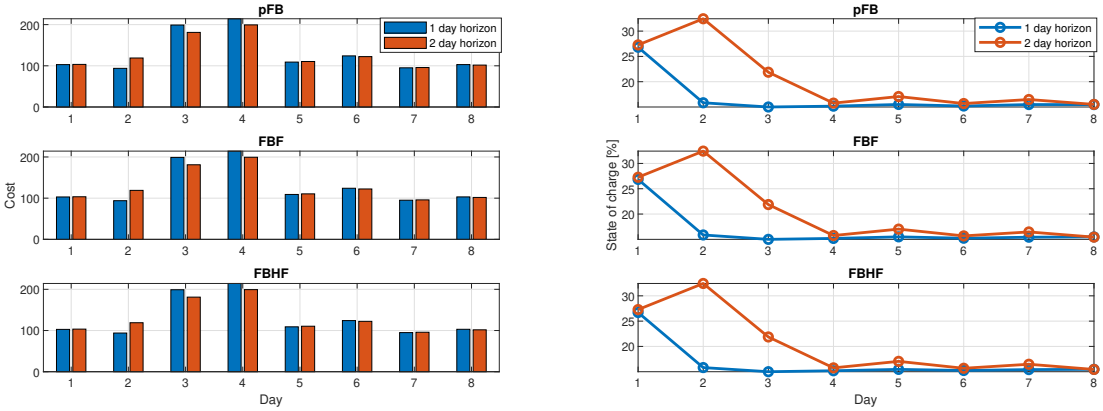
This method was tested with a prediction horizon of 2 days, with a control horizon of 1 day, and compared with a prediction horizon of only 1 day, also with a control horizon of 1 day. Again, the demand profile shown in Figure 6-15 is used. By looking more than one day ahead when determining the strategy of the next day, battery capacity can be saved for when energy is more expensive, i.e., when the total demand is higher. As explained in Section 6-5, in reality the total demand at each time step in the future can only be estimated, so for these simulations the total demand is again modeled as a random variable, where the numbers are drawn from a normal distribution.

Table 6-3 shows the total cost when using a prediction horizon of 2 days compared to a prediction horizon of 1 day. Although the reduction is slight, using a 2 day prediction horizon results in a lower cost.

Prediction horizon	1 day	2 days
pFB	100%	-0.77%
FBF	100%	-0.77%
FBHF	100%	-0.78%

Table 6-3: Total cost when using a prediction horizon of two days compared to a prediction horizon of one day

Figure 6-18a shows the total cost of each day for the two different cases. It can be observed that using a 2 day prediction horizon increases the cost on the second day and reduces the cost on day 3 and 4, where the total demand is higher. This is to be expected since on the second day battery capacity is saved, which is used on day 3 and 4. This can be seen in Figure 6-18b, which shows the average SOC over all agents at the end of each day. As expected, when using a 2 day prediction horizon, the average SOC over all agents is higher after day 2 and 3. Although not as visible as for day 2 and 3, on day 5 and 7 also a small amount of battery power is saved to use on day 6 and 8, respectively.



(a) Total cost over all agents for a prediction horizon of 1 and 2 days for all three algorithms

(b) Average SOC of all agents per day for a prediction horizon of 1 and 2 days for all three algorithms

Figure 6-18: Comparison between a prediction horizon of 1 and 2 days for all three algorithms

Chapter 7

Discussion

Three distributed algorithms were proposed to obtain a safe and efficient operation of the future grid while respecting the imposed constraints as well as minimizing the cost of power usage. They were tested in a simulation on an electricity market model where it was shown that the algorithms are capable of achieving this goal in a distributive manner. However, although the cost functions in this model are based on real life examples, e.g., a quadratic cost function for producing electricity, and real electricity demand data was used, all coefficients for the cost functions and constraints are artificial. To get a more meaningful result in terms of cost reduction, it would be useful to test the algorithms with more realistic values for the coefficients.

In the literature about energy management and peer-to-peer (P2P) trading, each paper considers a slightly different aspect of the energy market and slightly different goals they want to achieve [27], [36], [37]. Because there is no single test platform where each algorithm and method can be tested, it is hard to make a meaningful comparison to existing literature in terms of performance and cost reduction. Although it is hard to make a comparison in terms of performance, one thing that is important to note is that there are very few papers that consider the shared constraints necessary for a realistic model, which is an advantage of the algorithms used in this thesis. Also, the proposed algorithms are capable of handling uncertainties.

The step sizes used for all simulations were found by trial and error. However, the performance of the algorithms is very dependent on these step sizes. For example, changing the value of the step size δ_i to 0.1 for all agents instead of 0.05 makes the algorithms more than twice as slow. The forward-backward-forward (FBF) does not even converge for this value of δ_i in the scenario where none of the agents have storage (see Section A-2).

In Section 6-4 the results were shown for an increasing number of agents. It can be observed that the amount of time needed for each iteration increases significantly for an increasing amount of agents. Because the goal of this thesis was to prove the algorithms work and not to be programmed as efficiently as possible for a large amount of agents, this problem might be partly due to the way it was programmed rather than a problem of the actual algorithms

themselves. It should be noted that these simulations were done on a single computer, where the optimization steps for each agent were done consecutively. In a real application all agents can do their own optimization at the same time. This would reduce the simulation time significantly.

Chapter 8

Conclusion and recommendations

8-1 Summary and conclusions

The increase in renewable distributed energy resources and the increase in electric appliances pose a problem to the electric grid. Because supply and demand should always be balanced and power outages because of grid congestion must be prevented, energy management algorithms are being developed to ensure a safe and efficient operation of the future grid.

8-1-1 Main results

Because of the selfish nature of the agents in a power grid, game theory was used to model the electricity market. To prevent congestion of the grid, a constraint was added to limit the total amount of power that all agents combined can buy from the main grid at each time step. Together with a reciprocity constraint for the peer-to-peer (P2P) trades, these constraints couple the decisions of the agents, which make the game a so called generalized Nash equilibrium problem (GNEP). To solve this generalized game, it was passed to a variational inequality (VI). On its turn, this VI was recast as a monotone inclusion. This monotone inclusion can be solved by a fixed-point iteration using different operator splitting schemes. Because we wanted a full P2P market without the need of a central coordinator, we looked for distributed algorithms in which each agent only knows its local cost function and local feasible set and the dual variables are shared directly between the agents. To obtain a fully distributed algorithm, three recently developed operator splitting schemes, i.e., preconditioned forward-backward (pFB), forward-backward-forward (FBF) and forward-backward-half-forward (FBHF), were derived for the electricity market model and tested in a simulation. Although the pFB and FBHF algorithms are not guaranteed to converge with the used cost function, the simulations show that all three algorithms are capable of ensuring a stable and safe operation of the grid while reducing the cost.

8-1-2 Scalability

The algorithms were also tested for different amounts of agents, where it was shown that the time each iteration takes increases significantly when the number of agents increases. The amount of iterations to reach the stopping criteria also increased, but where the amount of iterations was only twice as high when going from a 10- to a 30-agent network, the elapsed CPU time was more than 20 times as high. It was found that the FBF algorithm needs more iterations and CPU time with an increasing number of agents than the pFB and the FBHF algorithms.

8-1-3 Stochastics

In Section 6-2 the simulations were performed using a deterministic demand profile for the day ahead. Since the day ahead demand profile cannot be exactly known in reality, the algorithms were also tested using a stochastic demand profile. To approximate the cost function, the stochastic approximation (SA) method with variance reduction was used where the numbers where drawn from a normal distribution. Although the algorithms are not guaranteed to converge using this stochastic demand, it was shown that all three of them still converge towards a solution.

8-1-4 Storage constraint

All simulations up until Section 6-6 were done for a single day ahead. As it may occur that some days have a higher total demand than others, it might be beneficial to take the expected demand of extra days in the future into account when determining a strategy. The algorithms were tested in a simulation that involves multiple days. It was found that by using the forecast of an extra day ahead when calculating the strategy of the next day, the cost over all days was decreased by 0.8%.

8-2 Recommendations

This section discusses the recommendations that arose from working on this thesis. Most of these recommendations follow from the limitations discussed in Chapter 7.

The electricity market setup in this thesis was used to prove the algorithms are capable of achieving the goal stated in Section 1-2. However, the coefficients for the cost functions and the constraints are artificial. To get a more meaningful result, it would be useful to test the algorithms in a more realistic energy market setting.

Because the algorithms are so sensitive to the step sizes, it would be interesting to run an optimization of the step sizes to obtain optimal values. This way the algorithms could potentially become even faster.

Because the goal of this thesis was to prove the algorithms work, they were not programmed as efficiently as possible. In Section 6-4 it was shown that the algorithms become significantly slower when the number of agents increases. It would be interesting to see how much improvement can be achieved by programming more efficiently.

In the simulations in this thesis a time step of one hour was used. In future work one could run the algorithms with smaller time steps so they can also be used for the intraday and balancing market, as described in Section 2-1-3 and Section 2-1-4.

In general it would be helpful to have a test platform to get a more meaningful comparison between different energy management algorithms and methods.

Appendix A

Simulation values

This appendix shows the values that were used for the cost functions (Section A-1) and for the step sizes (Section A-2). Section A-2 also shows the influence of the step sizes on the performance of the algorithms.

A-1 Values cost functions

For the cost functions, the following values were used: $q_h^{mg} = \frac{1.5}{N}$, $\forall h \in \mathcal{H}$, $q_i^{st} = 0$ and $c_i^{st} = 1$, $\forall i \in \mathcal{N}^{st}$, $c_{(i,j)}^{tr} = 1$, $\forall j \in \mathcal{N}_i$ and $i \in \mathcal{N}$ and $c_i^{dg} = 1$, $\forall i \in \mathcal{N}^{dg}$. Furthermore, we have that $q_i^{dg} = \frac{1}{3}$ for the large-scale residential customers and $q_i^{dg} = \frac{1}{4}$ for the school, office and retail buildings.

A-2 Step sizes

The step sizes used in the simulations were found by trial and error. The following values were used:

- $\alpha_{i,h}^{dg} = 0.1,\ \forall i \in \mathcal{N}$
- $\alpha_{i,h}^{st} = 0.1,\ \forall i \in \mathcal{N}$
- $\alpha_{i,h}^{mg} = 0.1,\ \forall i \in \mathcal{N}$
- $\alpha_{(i,j),h}^{tr} = 2.5,\ \forall i \in \mathcal{N},\ \forall j \in \mathcal{N}_i$
- $\beta_{ij} = \beta = 0.1,\ \forall i \in \mathcal{N},\ \forall j \in \mathcal{N}_i$
- $\gamma_i = \gamma = 0.1,\ \forall i \in \mathcal{N}$
- $\delta_i = \delta = 0.05,\ \forall i \in \mathcal{N}$

The algorithms are very sensitive to the step sizes. For example, using $\delta_i = 0.1$ instead of $\delta_i = 0.05,\ \forall i \in \mathcal{N}$, makes the algorithms much slower, needing more iterations to reach the stopping criteria, or not converging at all. For example, the forward-backward-half-forward (FBHF) needed approximately 180 iterations for the scenario where none of the agents have storage units to converge to the given stopping criteria when using $\delta_i = 0.05,\ \forall i \in \mathcal{N}$, and 440 iterations when using $\delta_i = 0.1,\ \forall i \in \mathcal{N}$. Even more so, the forward-backward-forward (FBF) algorithm does not converges at all anymore using this value for the scenario where none of the agents have storage units. The results for this specific case are shown in Figure A-1. The residuals appear to converge at first, but from around 75 iterations they start to diverge. The disagreement on the dual variable λ diverges from the start. It is clear from these results that the algorithms show great dependency on the chosen step sizes.

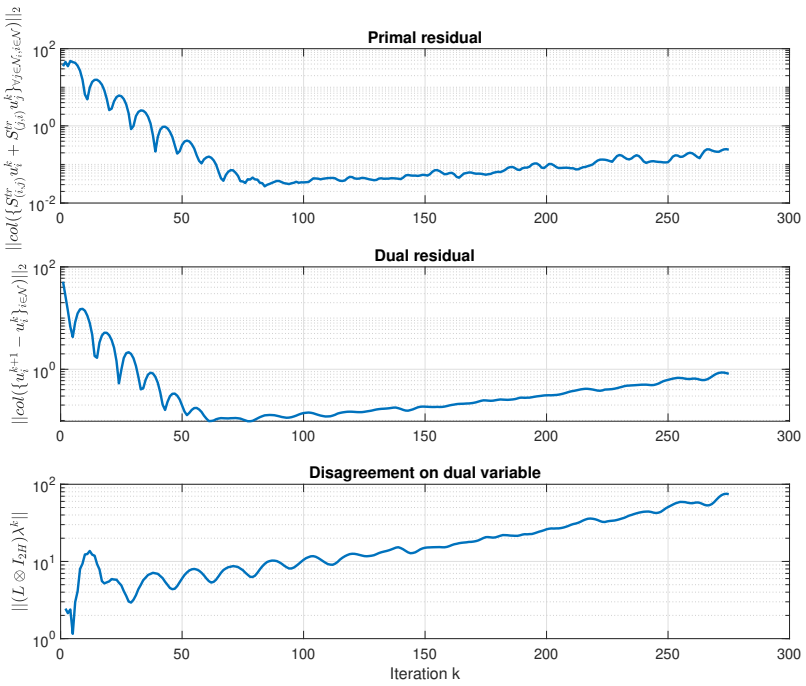


Figure A-1: Primal and dual residual and disagreement on λ when using $\delta_i = 0.1$ instead of $\delta = 0.05,\ \forall i \in \mathcal{N}$ for the FBF algorithm when none of the agents have storage units

Bibliography

[1] Center on Global Energy Policy, "Energy transition fact sheet," 2019.

[2] J. Abdella and K. Shuaib, "Peer to peer distributed energy trading in smart grids: A survey," *Energies*, vol. 11, no. 6, 2018.

[3] European Commision, "Energy performance of buildings directive," 2019.

[4] W. Saad, Z. Han, H. V. Poor, and T. Başar, "Game-theoretic methods for the smart grid: An overview of microgrid systems, demand-side management, and smart grid communications," *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 86–105, 2012.

[5] J. A. Logeman, "Demand Management and Peer- to-peer Trading in the Future Smart Grid," 2020.

[6] G. Belgioioso, W. Ananduta, S. Grammatico, and C. Ocampo-martinez, "Energy Management and Peer-to-peer Trading in Future Smart Grids : A Distributed Game-Theoretic Approach," vol. 675318, no. 675318, 2020.

[7] G. Erbach, "Understanding electricity markets in the EU," *European Parlimentary Research Service*, no. November, p. 10, 2016.

[8] F. Nieuwenhout and A. Brand, "The impact of wind power on day-ahead electricity prices in the Netherlands," *2011 8th International Conference on the European Energy Market, EEM 11*, pp. 226–230, 2011.

[9] P. Mäntysaari, *EU Electricity Trade Law*.

[10] Frontier Economics, "METIS Technical Note T4: Overview of European Electricity Markets," *METIS Technical notes*, no. February, p. 67, 2016.

[11] L. Fusheng, L. Ruisheng, and Z. Fengquan, "Overview of microgrid," *Microgrid Technology and Engineering Application*, pp. 1–10, 2016.

[12] "Staying big or getting smaller."

[13] IRENA, "Costs and Markets to 2030," *Electricity Storage and Renewables: Costs and Markets to 2030, https://www.irena.org/publications/2017/Oct/Electricity-storage-and-renewables-costs-and-markets*, no. October, 2017.

[14] R. B. Meyerson, *Game Theory: The Analysis of Conflict*. Harvard University Press, 1997.

[15] Z. Han, D. Niyato, W. Saad, T. Başar, and A. Hjørungnes, *Game theory in wireless and communication networks: Theory, models, and applications*, vol. 9780521196. 2011.

[16] W. Tushar, C. Yuen, H. Mohsenian-Rad, T. Saha, H. V. Poor, and K. L. Wood, "Transforming energy networks via peer-to-peer energy trading: The potential of game-theoretic approaches," *IEEE Signal Processing Magazine*, vol. 35, no. 4, pp. 90–111, 2018.

[17] A. A. Kulkarni and U. V. Shanbhag, "On the variational equilibrium as a refinement of the generalized Nash equilibrium," *Automatica*, vol. 48, no. 1, pp. 45–55, 2012.

[18] B. Franci and S. Grammatico, "Distributed Forward-Backward algorithms for stochastic generalized Nash equilibrium seeking," no. 802348, 2019.

[19] F. Facchinei, A. Fischer, and V. Piccialli, "On generalized Nash games and variational inequalities," *Operations Research Letters*, vol. 35, no. 2, pp. 159–164, 2007.

[20] G. Scutari, D. P. Palomar, F. Facchinei, and J.-S. Pang, "Convex Optimization, Game Theory, and Variational Inequality Theory," no. May, pp. 35–49, 2010.

[21] F. Facchinei and C. Kanzow, "Generalized Nash equilibrium problems," *Annals of Operations Research*, vol. 175, no. 1, pp. 177–211, 2010.

[22] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2011.

[23] B. Franci and S. Grammatico, "Distributed forward-backward (half) forward algorithms for generalized Nash equilibrium seeking," 2019.

[24] B. Franci and S. Grammatico, "A distributed forward-backward algorithm for stochastic generalized Nash equilibrium seeking," *European Control Conference 2020, ECC 2020*, vol. XX, no. XX, pp. 1117–1122, 2020.

[25] N. M. Isa, C. W. Tan, and A. H. Yatim, "A comprehensive review of cogeneration system in a microgrid: A perspective from architecture and operating system," *Renewable and Sustainable Energy Reviews*, vol. 81, no. June 2017, pp. 2236–2263, 2018.

[26] D. P. Duffy, "CHP and Microgrids - A Profitable Pair," 2018.

[27] A. H. Mohsenian-Rad, V. W. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, "Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid," *IEEE Transactions on Smart Grid*, vol. 1, no. 3, pp. 320–331, 2010.

[28] I. Atzeni, L. G. Ordóñez, G. Scutari, D. P. Palomar, and J. R. Fonollosa, "Demand-side management via distributed energy generation and storage optimization," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 866–876, 2013.

[29] B. N. Popov, "Cycle Life Modeling of Lithium-Ion Batteries," no. January 2004, 2015.

[30] J. M. Id, G. Luo, M. Ricco, and M. Swierczynski, "Overview of Lithium-Ion Battery Modeling Methods for State-of-Charge Estimation in Electrical Vehicles," 2018.

[31] M. Murnane, "A Closer Look at State of Charge (SOC) and State of Health (SOH) Estimation Techniques for Batteries,"

[32] P. Yi and L. Pavel, "An operator splitting approach for distributed generalized Nash equilibria computation," *Automatica*, vol. 102, pp. 111–121, 2019.

[33] G. Belgioioso and S. Grammatico, "A distributed proximal-point algorithm for nash equilibrium seeking in generalized potential games with linearly coupled cost functions," *2019 18th European Control Conference, ECC 2019*, vol. 0, pp. 3390–3395, 2019.

[34] "Data OpenEI."

[35] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," *Proceedings of the IEEE International Symposium on Computer-Aided Control System Design*, pp. 284–289, 2004.

[36] B. G. Kim, S. Ren, M. Van Der Schaar, and J. W. Lee, "Bidirectional energy trading and residential load scheduling with electric vehicles in the smart grid," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 7, pp. 1219–1234, 2013.

[37] F. Moret and P. Pinson, "Energy Collectives: a Community and Fairness based Approach to Future Electricity Markets," *IEEE Transactions on Power Systems*, pp. 1–11, 2018.

Glossary

List of Acronyms

P2P	peer-to-peer
DER	distributed energy resource
EV	electric vehicle
TSO	transmission system operator
DSO	distribution system operator
PV	photovoltaic
NEP	Nash equilibrium problem
GNEP	generalized Nash equilibrium problem
GNE	generalized Nash equilibrium
v-GNE	variational GNE
v-GNEP	variational GNEP
SNEP	stochastic NEP
SGNEP	stochastic GNEP
SGNE	stochastic GNE stochastic GNEPs
v-SGNE	stochastic v-GNE
SOC	state of charge
CHP	combined heat and power
FB	forward-backward
pFB	preconditioned forward-backward
FBF	forward-backward-forward
FBHF	forward-backward-half-forward
SA	stochastic approximation

SAA	sample average approximation
KKT	Karush-Kuhn-Tucker
VI	variational inequality
SVI	stochastic VI
MPC	model predictive control