Iterative learning control in prosumer-based DC microgrids

Bachelor thesis

from

Paraskevi Vorropoulou

born on

04.05.1998 in Berlin, Germany

02. November 2020

Supervisor: Lia Strenge

Examiner: Prof. Dr.-Ing. Jörg Raisch
Control Systems Group
Institute of Energy and Automation Technology
Faculty IV - Electrical Engineering and Computer Science
Technische Universität Berlin

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den 27. Oktober 2020

.....

Paraskevi Vorropoulou

Zusammenfassung

Seit Ende des 20. Jahrhunderts ist der Begriff der Energiewende von großer Bedeutung. Darunter versteht man den Übergang von fossilen Energieträgern zur nachhaltigen Energieressourcen, die immer häufiger Anwendung finden in der Stromerzeugung. Immer beliebter wird unter anderem die dezentrale Energieversorgung, die verbrauchernahe Erzeugung elektrischer Energie, sodass der Strombedarf unmittelbar für den Verbraucher gedeckt ist. Dazu zählen Inselnetze beispielsweise, die prosumer-basierend sind, also produzierend und verbrauchend gleichzeitig.
Die zunehmende Nutzung von gleichstrom-basierend Ressourcen und Lasten ist auch ein Resultat der Energiewende, zum Beispiel die Photovoltaik und ist auch aus diesem Grund eine sehr relevante Lösung für die Elektrifizierung und verteilten Netzen im Niederspannungsbereich.
Lastprofile in Energienetzen haben aber oftmals periodische Komponenten. Diese können durch geeignete Regelung gelernt und allzu ganz kompensiert werden, sodass auch gleichzeitig Regelungsenergie eines Systems gespart wird. Geeignet dafür ist die iterativ lernende Regelung (ILR).
In dieser Arbeit wurde die iterative lernende Regelung in prosumer-basierenden Gleichstrominselnetzen angewandt, um die oben erwähnte Anforderung der Ausgleichung von periodischen Komponenten und des Energiesparens zu untersuchen. Dazu wird ein Modell für diese Art der Inselnetze vorgeschlagen und beschrieben mit geeigneten Gleichungen und Systemparametern als eine Ebene, und als weitere Kontrollebene wird die ILR damit verbunden, sodass ein Framework eingeführt wird. Dieses wird anschließend simuliert und dessen Leistung validiert.

Schlüsselwörter: Prosumer-basiertes Inselnetz, Gleichstrominselnetz iterativ lernende Regelung, Dezentrale Energieversorgung, ...

Abstract

Since the end of the 20th century, the concept of the energy transition has been of great importance. This is understood to mean the transition from fossil fuels to renewable energy resources, which are increasingly being used in power generation. Among other things, decentralized energy supply, the generation of electrical energy close to the consumer so that the electricity demand is covered directly for the consumer, is becoming increaslingly popular. This includes island networks, that are e.g. prosumer-based, i.e. producing and consuming simultaneously.
The increasing use of direct-current based resources and loads is also a result of the energy transition, and for this reason it is also a very relevant solution for electrification and distributed networks in the low-voltage range.
Load profiles in energy networks often have periodic components. These can be learned and compensated by appropriate controlling, so that the control energy of a system can be saved at the same time. The iterative learning control (ILC) is suitable for this purpose.
In this thesis the ILC is applied in prosumer-based direct current microgrids to investigate the above-mentioned requirement of balancing periodic components and energy saving. For this purpose, a model for this type of microgrids is proposed and described with suitable equations and system parameters as one level, and as a further control level the ILC is connected to it, so that a framework is introduced. This framework will then be simulated and its performance validated.

Keywords: Prosumer-based microgrids, direct current microgrids, iterative learning control, distributed energy generation, ...

Acknowledgements

First, I would like to express my deep gratitude to M.Sc. Lia Strenge for her outstanding supervision and the support during the process of this work; Dr.Chris Rackauckas for the helpful comments in Julia programming.
Furthermore many thanks for the love and energy from my parents, sisters, friends and my boyfriend.

Paraskevi Vorropoulou

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we introduce the background and motivation for this thesis in Section 1.1 and the related work in Section 1.2. Subsequently, a preview of all chapters in this thesis is given in Section 1.3.

1.1 Background and motivation

Microgrids have become one of the most popular research topics since the energy transition towards renewable energy and distributed generation. The centralized power system with fossil fuels is being transformed into a increasingly more decentralized power system including fluctuating renewable energy resources [6]. Due to the increased use of direct current based loads and distributed power generation (e.g. photovoltaic) direct current microgrids are a very relevant and popular solution in low-voltage distribution grids with a fair power-sharing for all consumers. To achieve this, power sharing of the prosumer nodes of a direct current microgrid and bounded voltage deviation from the reference voltage of the grid must be assumed [3] by applying a lower-layer control with these objectives.
However, the demand profiles in power grids contain periodic components. Therefore, the iterative learning control is applied as a higher-layer controller to learn the daily periodicity in order to compensate the periodic demand patterns and reduce the decentralized lower layer control energy, since it is assumed that planned day-ahead power infeed is cheaper than instantaneous control power [4, 6]. By learning the previous cycle, the iterative learning controller can optimize the performance of the system and reduce the error which is based on the previous cycle.
In this thesis, the overall network modeling of the direct current prosumer-based

microgrid in a differential-algebraic equation framework is firstly proposed with the lower-layer control energy design based on the previously mentioned control objectives. Furthermore, ILC framework will be proposed as a higher-layer controller, completing a hierarchically controlled system (to cope with different time scales (seconds to days)).

1.2 Related work

In the field of power systems, [7], as well as [8] and [9] define the main points of direct current microgrids. [5] offers us the concept of meshed DC microgrids, and [3] offers the concept of the droop controlled low-voltage DC microgrids. Both of them have the producer-consumer distinguishment and based on these equations the prosumer-model can be derived. Both sources by Lia Strenge provide differential-algebraic equations for the overall lower-layer microgrid satisfying the power sharing and bounded voltage deviation control objectives.
For the control methods of direct current microgrids, the multilayered schema is explained in [10] and [11] and [12] offer contributions on the two-layer control.
The concepts of iterative learning control are proposed by Moore in [13] and [14] with Fig. 2.5. Obviously this work is based on the previous work of [4], that provides an overall hierarchical control model for a prosumer-based alternating current microgrid with iterative learning control.

1.3 Outline

This work is divided as follows:

- **Chapter 1**: An introduction to the motivation, background and related work to this thesis
- **Chapter 2**: An overview of the theoretical preliminaries of the prosumer-based DC microgrids, power network modeling and the two-layer control with the ILC method for a better understanding of the following context
- **Chapter 3**: The proposal of the prosumer-based DC microgrid network model, based on the producer-consumer microgrid model, followed by the representation of the ILC design
- **Chapter 4**: The simulations of the overall model, a list of the results, the model validation and interpretation

- **Chapter 5**: Summary and future directions

Chapter 2

Theory and background information

Now, that the topic of the thesis was introduced and the question of the research was stated, there are necessary preliminaries that need to be mentioned and explained, in order have a better understanding fpr the following chapters.
Thus, we will clarify the basics of the *microgrids* and in particular the *(prosumer-based) DC microgrids* (2.1), *power network modeling* (2.2)and the *iterative learning control* method (2.3)

2.1 Basics of prosumer-based DC microgrids

In this section we are going to introduce the (prosumer-based DC) microgrid concept. Firstly, in order to understand the functionality of the microgrids and especially the DC microgrids, which are going to be introduced later, we should discuss, what *Distributed generation* is.

2.1.1 Distributed generation

Distributed generation is the electrical generation and storage performed by a variety of small, grid-connected or distribution system-connected devices referred to as distributed energy resources (*DER*), as defined in [15]. DER systems typically use renewable energy sources such as e.g. biogas, fuel cells, solar power or wind power [15]. Different from a transmission network, distributed generation basically generates electricity close to the consumer, which means that the efficiency of the current generation plant usually only covers the energy demand of current consumers that are directly next or in the closer environment.

Another difference between distributed and the central generation, as described in [16] is, that the distributed resources work with low voltages, with the central generators are provided through the higher voltages. Therefore, distributed generation can reduce electricity losses along transmission and distribution lines when connected to the electric utility's lower voltage distribution lines [16]. It can also provide lower-cost electricity and higher power reliability and security with fewer environmental consequences such in the common power generators. Distributed generation can be a single structure, such as a home or a place of residence, or it may be part of a *microgrid*, such as a large college campus or a industrial plant [15].

Since the operation area of a microgrid is now clarified, we will introduce in the next subsection with the microgrid and in particular the DC microgrid.

2.1.2 Microgrids

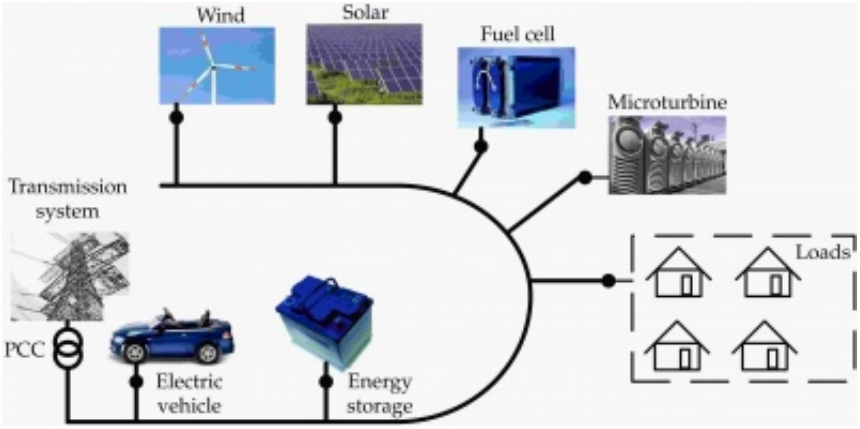

Figure 2.1: The microgrid concept

In this scope we are going to use the definition of a microgrid as in [7].

Definition 1. *from [7]*
"Microgrids comprise LV distribution systems with [...] DER (microturbines, fuel cells, PV, etc.) together with storage devices (flywheels, energy capacitors and batteries) and flexible loads. Such systems can be operated in a non-autonomous way, if interconnected to the grid, or in an autonomous way, if disconnected from the main grid. The operation of microsources in the network can provide distinct benefits to the overall system performance, if managed and coordinated efficiently."

By this definition we understand that a microgrid consists of a limited number of generation units, storage units and demand resources, all together located in a local distribution grid, as sketched in 2.1 [8]. Microgrids do not always have a synchronous connection to the main generation net which means that the tasks that are necessary for a constant and safe operation need to be provided, in cases of emergency [9]. They should though have the option of a synchronous connection to the main generation network. Such systems arise from the geographical circumstances that make a connection to the main network costly and almost impossible. Microgrids on the other hand serve to an increased security of supply, should a collaps of the main network occur [9].
An introduction on the alternating and direct current microgrids is given in the next section. Since the main focus of this thesis is on the direct current type of microgrids, the alternating current will be only mentioned summarized.

2.1.3 Direct current versus alternating current microgrids

Electric charge in *alternating current* (AC) changes direction periodically and is characterized by an amplitude and a frequency. The electric charge in *direct current* (DC) on the other hand only flows in one direction and remains constant the whole time. Back in time, DC could not be easily converted to high voltages due to the high transformation losses. Therefore, AC is the preferred type of current for power networks and used until today [17], but with the development of the DC-based power electronics for generating as well as load that allow efficient voltage transformation, DC is on the rise again and more often used in microgrids.
In [18], Table 4 in [25, p.399] a comparison between AC and DC microgrids is given. Since the line losses (the power that is lost in a conductor during transmission and distribution phase, as in [19]) in AC distribution are high, it leads to less efficiency and less power transmission. In DC distribution the power transmission is more efficient due to a lower line loss, no reactance in the line and a low line resistance itself. DC distribution does not obviously require frequency monitoring, since the frequency of DC is constantly zero, has no electromagnetic interference, like AC distribution does and is easier when it comes to the analysis, since only real numbers are involved.
Hence, DC distribution is chosen in this work due to higher efficiency and lower line losses. Finally, prosumer-based DC microgrid is defined in the next section.

2.1.4 Prosumer-based (DC) microgrids

In 2.1.1, it is mentioned that " [...] distributed generation basically generates electricity close to the consumer [...]". Obviously there is a distinction between generation and consumption for the energy transfer. The DC microgrid model, as introduced in 2.1.2 is also based on the separation of the as called *producers* and *consumers*. By definition:

- A distribution unit is called *producer*, if it feeds power into the grid.
- The distribution is called *consumer* when it draws power from the grid.

As in 2.1.1 producers "[...] typically (are) [...] renewable energy sources such as e.g. biogas, fuel cells, solar power or wind power that can feed power into the grid. Consumers are power loads such as homes or places of residence that draw the power from the grid for each purpose.
Spurred by technology development and regulatory frameworks, self-consumption of distributed renewable electricity generation has gained relevance in many power markets around the world [20]. But also in a daily usage, consumers can draw electricity at the same time and, for example, produce it via the solar photovoltaic panels on the roof and feed it into the grid.
For our research we need to understand the definition of a *prosumer*. The word is a portmanteau [21] , a linguistic blend of the words producer and consumer. Prosumers are units which produce energy and feed it into a distribution network, like in the solar photovoltaic panels for example [22]. They feed power into the grid in case that the produced energy is bigger than the energy demand. Prosumer draw the same power from the grid, when the energy demand exceeds the production of it.
Using this information in our DC microgrid model mentioned in section 2.1.3, the following definition 2 for prosumer-based DC microgrids can be given.

Definition 2. *A prosumer-based microgrid is a power network where every node of the grid can be a producer, a consumer or idle. In other works every node can feed power into the grid, draw power from the grid or not participate in the grid. Every node* $\alpha \in \mathcal{N}$ *can feed power* $P_\alpha \in \mathbb{R}_{\geq 0}$ *into the microgrid with* $P_\alpha \geq 0$*, receive power from the microgrid with* $P_\alpha \leq 0$ *or not participate in the microgrid.*

Since the prosumers are described as *nodes*, in the next section we are going to introduce the basics of power network modeling as in graph theory, so that we

have a better overview for the modeling of prosumer-based DC microgrids in Section 3

2.2 Basics of power network modeling

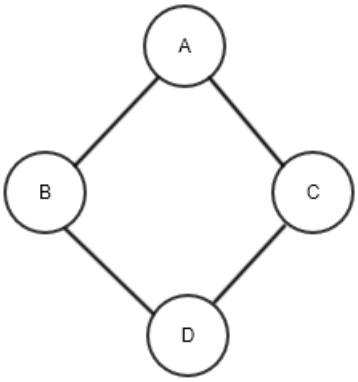

Figure 2.2: An example of an undirected graph

A simple power network can be represented as a graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} \in \mathbb{N}$ is the set of *vertices* or *nodes* and $\mathcal{E} \subseteq (\mathcal{V} \times \mathcal{V})$ the set of *edges* [23]. The nodes are connected via the edges and we consider the connection as *undirected*. In graph theory, a node is considered *incident* to an edge if the node is one of the two node the edge connects. An example of an undirected graph is shown in 2.2 [24].
As described in Definition 2, a node from a power network in the form of a graph is a representation of a prosumer, so either a producer, consumer or idle. An edge represents the power lines that connects the prosumers to each other. Current flows through the power lines, feeding or drawing electricity to or from the nodes. This means that the current flows in both direction, which is why the graph is also considered as *undirected*.
Let $N \in \mathbb{N}$ be the number of prosumers in a DC microgrid connected to each other. The prosumers are at grid nodes $j,k \in \{1,2,..,N\} =: \mathcal{N} \subset \mathbb{N}$. Then the number of the edges or *number of power lines* is defined as [3]:

$$m \leq \frac{N(N-1)}{2}$$

The index $h \in \{1,2,..,m\} =: \mathcal{M} \subset \mathbb{N}$ is then used for the power lines [3]. The connection jk is incident with the grid nodes j and k and starts from node j to node k. It contains the power line h.

To describe the relationship between between nodes and power, we use the *incidence matrix* $\boldsymbol{M}$ of a network graph with the dimension $m \times N$ with

$$\boldsymbol{M} = \begin{bmatrix} m_{1,1} & \dots & m_{1,N} \\ \vdots & \ddots & \vdots \\ m_{m,1} & \dots & m_{m,N} \end{bmatrix} \in \mathbb{R}^{m \times N} \tag{2.1}$$

where the entries for each connection are $m_{h,j} = 1$ and $m_{h,k} = -1$. All other entries $m_{h,\kappa} = 0$ are the non-incident nodes $\kappa \in \mathcal{N}$ to the power line h [3].
With the incidence matrix $\boldsymbol{M}$, the network topology is uniquely described. By using the incidence matrix, we can describe the Kirchhoff Laws in the network as constraints [3]. Since the topic of this thesis is to control the overall model of the prosumer-based direct current microgrid with the iterative learning control method, the next section introduces the two-layer control with the iterative learning control method.

2.3 The two-layer control with the iterative learning control method

In this section, we get to know the basics of the two-layer control with the *iterative learning control method*, by introducing in 2.3.1 the *standard control feedback loop*, which is the base for many control applications. We will later delve into the *low-layer-control* and the *high-layer-control*, that result the *two-layer-control method* (2.3.2), one specialization of which is the iterative learning control (2.3.3).

2.3.1 Basics of of the standard control feedback loop

In engineering, processes in form of a dynamical (time-dependent) system need to be controlled, in order to achieve control stability and an optimal behavior. Therefore, a *controller* is designed uniquely for the to-be-controlled system, to automatically regulate the output and keep it within the systems desired input value or "set point" [25]. If the system input changes for whatever reason, the output of the system must respond accordingly and change itself to reflect the new input value [25].
In most systems the controlling process, the controllers actions depend on the feedback from the process. In other words, we can control a process by adding an amount of the output to the measured error, so the actual output is compared with the desired and the error can be reduced or even eliminated. This type of control system is called *closed-loop control system* or *feedback control system* and

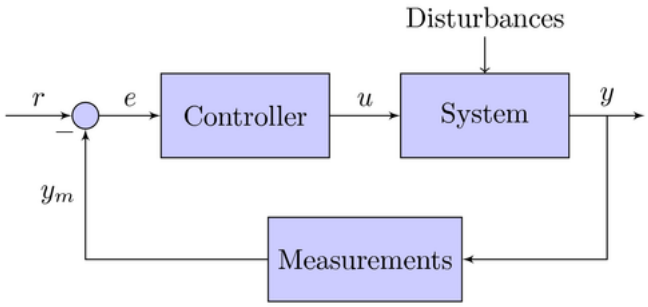

Figure 2.3: Closed-loop model in standard feedback control loop block representation [1], with the definitions as in [2, 3] r for the reference, e for the measured error, u for the system input, y for the system output and y_m for the measured system output

two-layer controllers belong to this family of controls.
The basics of the *two-layer control method*, which is going to be used in this research, are going to be explained in the next section 2.3.2.

2.3.2 Two-layer control

Microgrids control and management are, not as in the standard feedback control loop described in section 2.3.1, usually divided in multiobject tasks, covering mostly different time scales and physical levels [10]. This schema is divided hierarchically and every layer has a different task from e.g. frequency control to energy management. Two-layer architectures, which are used in our research as control architecture are e.g. used among others for economic optimization [11]. As described in [4], in microgrids, hierarchical control is typically divided in *primary*, *secondary* and *tertiary control*, also called energy management, referring to the same tasks. In particular it is said that "[...] *primary control is responsible for fast frequency stabilization and reacts in seconds, secondary control restores the frequency to its setpoint in terms of minutes and tertiary control refers to economic dispatch questions in the time scale of hours and days[...]*" [4]. The definition is referring to systems, where frequency is a control property. Since in direct current microgrids no frequency appears, primary, secondary and tertiary control, proposed by Meng and Ferrai Trecate in [10] and Luis Orlando Polanco Vasquez et al. in [12] are considered as follows:

1. *Primary control* performs the control of local power, voltage, and current. It normally follows the set-points given by upper level controllers and is communication-less power sharing among DERs [10].

2. *Secondary control* appears on top of primary control. It deals with issues in the system level, such as power quality regulation, microgrid synchronization with external grid for smooth reconnection [...] and so on [10].

3. *Tertiary control* is issued with optimization, management, and overall system regulations [10]. It considers the economical concerns [...] and manages the power flow between the microgrid and the main grid [12].

In decentralized hierarchical control, regulation takes place with only local information [12] and the control functions can be distributed into local controllers, or in our case *lower-layer controllers* and upper-level or *higher-layer controllers*. Primary and secondary control can be combined as the lower-layer control and tertiary control as the higher-layer control. The grid itself and the low-level controller form the *compound plant* Placing the higher-layer controller into the system can save lower-layer control energy [4].
In this research, the two-layer schema is used with an *iterative learning controller* as a higher-layer controller, proposed by Strenge et al. in [4]. The iterative learning method is going to be introduced in the next section.

2.3.3 Iterative learning control

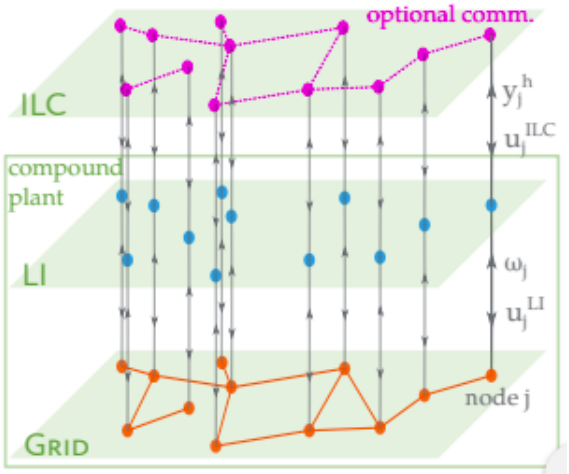

Figure 2.4: Control hierarchy with low-level control (LI) and iterative learning controller (ILC) [4]. Here, the grid current is alternating and the control achievement is the bounded frequency deviation (ω_j)

The *iterative learning control* (ILC) is a method that addresses the problem of transient response performance for systems that operate in a repetitive mode

[14]. Based on the errors observed during past operations, the transient response is improved by adjusting the input to the plant during future [14]. The basic

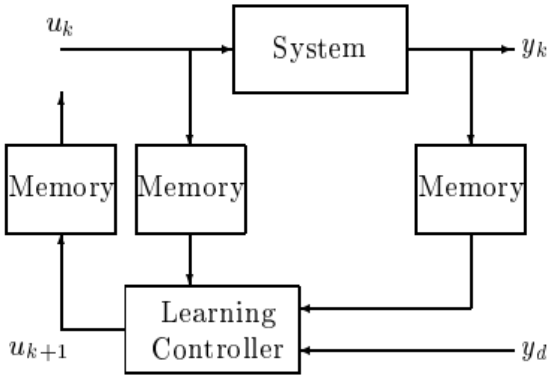

Figure 2.5: Iterative learning control configuration

idea of iterative learning control is illustrated in Figure 2.5 [13, p.427]. The ILC algorithm has the form, proposed by [4]

$$u^{c+1} = u^c - Ly^c \tag{2.2}$$

The power c is the cycle number. In this work c has the duration of one day. During the c-th repetition u^c is inputted, producing an output y^c. While the input and output is memorized, this output is observed as an error and based on this the next input u^{c+1} is modified. The new input is designed so that the new error is smaller than the previous input. By time shifting we get

$$u^c = u^{c-1} - Ly^{c-1} \tag{2.3}$$

with L a single scalar parameter $L = \kappa \in \mathbb{R}_{>0}$.
The proposed ILC approach is applied to learn a power infeed that compensates the periodic demand component and to save lower-layer-energy [4].
Now that the basics of prosumer-based direct current microgrids and the iterative learning control have been introduced, in the next chapter the network dynamics with the lower-layer and higher-layer controller can be modeled.

Chapter 3

Network modeling of the prosumer-based DC Microgrid and the iterative learning controller design

After having explained the research work and summarized the necessary theory preliminaries in the previous chapters, this chapter goes through the steps of modeling the prosumer-based direct current microgrid with iterative learning control. Section 3.1 starts off with the *low voltage direct current microgrid dynamics* (3.1.1), followed by the definition of the *prosumer-based direct current microgrid model* (2.1.4). The following section 2.3 derives the *structure and design of the iterative learning controller*, by defining the *control objectives* (3.2.1), the structure of the *higher-layer controller* (3.2.2) and the *learning dynamics* (**??**).

3.1 Network model of the prosumer-based DC microgrid with lower-layer control

In this section, we model the network of the prosumer-based DC microgrid and define the dynamics of it. The model is adapted by the *low voltage direct current microgrid* dynamics, proposed by Lia Strenge in her master thesis [3], where a distinction is made between producers and consumers. In this thesis, however, the model and equations are adapted to combine producer and consumer as prosumer. Despite everything it is necessary to be introduced to these dynamics, to have a better understanding for the modeling in Section 3.1.2

3.1.1 Network modeling of the producer-consumer DC microgrid

This Section serves as a base and defines the equations, that inspired the prosumer-based microgrid network dynamics and the lower-layer controller, respectively, with no influence of the ILC. We assume a constant demand for the consumer nodes and no higher-layer control input, to make the derivation of the equations clearer.
As explained above, the modeling of this network topology and the definitions and equations mentioned in this Section are proposed by Lia Strenge in "*Modeling and simulation of a droop controlled swarm type low voltage DC microgrid in a DAE framework - Master Thesis*" [3], as well as the work "*Stability of meshed DC microgrids using probabilistic analysis*" by L. Strenge, H. Kirchhoff, G.L. Ndow and F. Hellmann [5].
Recall from 2.2, that a microgrid is considered as an *undirected graph*. Then $N \in \mathbb{N}$ is the number of producers and consumers. The grid nodes $gn \in \{1,2,...,N\} =: \mathcal{N} \subset \mathbb{N}$ with $= \mathcal{N} = \mathcal{N}_P \cup \mathcal{N}_C$ represent the producers $n_P \in \mathcal{N}_P$ with index $j = 1,...,n_P$ and consumers $n_C \in \mathcal{N}_C$ with index $k = n_P+1,...,N$ and $n_P + n_C = N$, while the edges $pl \in \{1,2,...,m\} =: \mathcal{M} \subset \mathbb{N}$ with $m \leq N(N-1)/2$ and the index jk, represent the power lines connecting the producers with the consumers.
The network is modeled as an undirected connected graph with the edge-vertice-incidence matrix $\boldsymbol{M} \in \mathbb{R}^{m \times N}$, see (Eq. 2.1) in Section 2.2. Since the network is electric, the Kirchhoff laws must hold. The currents sum to zero at each node (Kirchhoff current law) and the voltages sum to zero in each mesh (kirchhoff voltage law) [3]. With these laws, we obtain the equations [3]:

$$v_{pl}(t) = \boldsymbol{M} v_{gn}(t) \tag{3.1}$$

$$i_{gn}(t) = \boldsymbol{M} i_{pl}(t) \tag{3.2}$$

where $t \in \mathbb{R}$ is the time in seconds, $\boldsymbol{M} \in \mathbb{R}^{m \times N}$ is the incidence matrix of the network; $v_{pl}(t) = (v_{pl,1}(t),...,v_{pl,m}(t))^T : \mathbb{R} \to \mathbb{R}^m$ is the vector of power line voltages in volt or the voltage difference between the connected grid nodes, respectively; $v_{gn}(t) = (v_1(t),...,v_N(t))^T : \mathbb{R} \to \mathbb{R}^N$ is the vector of grid node voltages in volt; $i_{gn}(t) = (i_1(t),...,i_N(t))^T : \mathbb{R} \to \mathbb{R}^N$ is the vector of the currents sum at the grid nodes in ampere and $i_{pl}(t) = (i_{pl,1}(t),...,i_{pl,m}(t))^T : \mathbb{R} \to \mathbb{R}^m$ is the vector of the power line currents in ampere [3].
Furthermore, we make the following assumptions:

i. Every node can feed electricity into the microgrid (producer with $i_{to,grid,j}(t) > 0, j = 1,...,n_P$), receive from the microgrid (consumer with $i_{to,grid,k}(t) < 0, k = n_P+1,...,N$) [5]

ii. At each node, there is no other information available other than local variables. Hence, the control is fully decentralized.

iii. The generation current at the producer nodes is known with [5]

$$i_{gen,j}(t) = K_{D,j}(V_{ref,j} - v_j(t)) \tag{3.3}$$

where $K_{D,j} > 0$ is the voltage droop coefficient and $V_{ref,j}$ the reference voltage at producer node $j \in \mathcal{N}_P$. Droop control (with its according droop coefficient mathimatically) is applied in a microgrid to stabilize the voltage drop [26].

iv. The current to the load at the consumer nodes is assumed to be known with [5]

$$i_{load,k}(t) = \frac{P_k(t)}{v_k(t)}, v_k(t) \neq 0 \tag{3.4}$$

where P_k is the load power drawn from the grid at consumer node $k \in \mathcal{N}_C$.

v. A power line in low voltage direct current applications behaves mostly inductive [3, p.28] with the context

$$L_{pl,jk}\frac{di_{pl,jk}(t)}{dt} = v_{pl,jk}(t) - R_{pl,jk}i_{pl,jk}(t) \tag{3.5}$$

where $L_{pl,jk}$ is the known inductance, $R_{pl,jk}$ the known resistance, $i_{pl,jk}(t)$ the power line current and $v_{pl,jk}(t)$ the power line voltage at power line $jk \in \mathcal{M}$. Since current flows in both directions, we further assume $L_{pl,jk} = L_{pl,kj}$, $R_{pl,jk} = R_{pl,kj}$, and $i_{pl,jk}(t) = i_{pl,kj}(t)$

vi. A producer or consumer can be electrically modeled by a current source, represented by the equation (Eq. 3.6) and a capacitor in parallel as in (Eq. 3.7) [3].

$$i_{gn}(t) = f(t,q) - i_{cap}(t) \tag{3.6}$$

$$i_{cap}(t) = \boldsymbol{C_{cap}}\frac{dv_{gn}(t)}{dt} \tag{3.7}$$

with $f : \mathbb{R} \times \{0,1\}^N \to \mathbb{R}^N$ as a source current vector in Ampere, $i_{cap}(t) = (i_{cap,1}(t), ..., i_{cap,2}(t))^T : \mathbb{R} \to \mathbb{R}^N_{>0}$ as a vector of capacitor currents in Ampere and $\boldsymbol{C_{cap}} = \text{diag}\,(C_{cap,1}, ..., C_{cap,N})^T \in \mathbb{R}^{N\times N}_{>0}$ as diagonal matrix of capacitances in Farad [3].

In assumption vi. the source current vector can be distinguished between the generation current from (Eq. 3.3) and the load current from (Eq. 3.4) with

$$f(t,q) = \begin{cases} K_{D,j}(V_{ref,j} - v_j(t)), & i_{togrid,i}(t) > 0 \\ \frac{P_k(t)}{v_k(t)}, & i_{togrid,i}(t) < 0 \end{cases} \tag{3.8}$$

with $i = 1, ..., n$ as the considered node and $v_k(t) \neq 0$. Furthermore we assume that the load power is constant, $P_k(t) = P, P \in \mathbb{R}$ Combining equations (Eq. 3.8), (Eq. 3.6) and (Eq. 3.7) and distinguishing between producer and consumer grid node voltage with

$$v_{gn}(t) = \begin{bmatrix} v_P(t) \\ v_C(t) \end{bmatrix} \tag{3.9}$$

producers listed first ($v_P(t) = (v_1(t), ..., v_{n_P}(t))^T$) and consumers listed last ($v_C(t) = (v_{n_P+1}(t), ..., v_n(t))^T$), we get the following equation [5]:

$$\boldsymbol{C_{cap}} \begin{bmatrix} \frac{dv_P(t)}{dt} \\ \frac{dv_C(t)}{dt} \end{bmatrix} = -i_{gn}(t) + \begin{bmatrix} \boldsymbol{K_D}(V_{ref,P} - v_P(t)) \\ P / v_C(t) \end{bmatrix} \tag{3.10}$$

$i_{gn}(t) = (i_{gn,1}(t), ..., i_{gn,N}(t))^T : \mathbb{R} \to \mathbb{R}^N$ is the vector of the currents sum at the grid nodes in ampere; $\boldsymbol{K_D} = \text{diag}(K_{D,j})$ is the diagonal droop coefficient matrix, j the producer index, as above; $V_{ref,P} = (V_{ref,1}, ..., V_{ref,n_P})^T$ the vector of the reference node voltages of the producers; $P = (P_1, ..., P_{n_c})^T$ is the load power vector of the consumers [5].
From equation (Eq. 3.5) in assumption v we get the equation:

$$\boldsymbol{L_{pl}} \frac{di_{pl}(t)}{dt} = v_{pl}(t) - \boldsymbol{R_{pl}} i_{pl}(t) \tag{3.11}$$

in vector representation, with $\boldsymbol{L_{pl}} = \text{diag}(L_{pl,jk})$ the diagonal power line inductance matrix and $\boldsymbol{R_{pl}} = \text{diag}(R_{pl,jk})$ the diagonal power line resistance matrix [5]. Recall, that $v_{pl}(t) = (v_{pl,1}(t), ..., v_{pl,m}(t))^T : \mathbb{R} \to \mathbb{R}^m$ is the vector of power line voltages in volt or the voltage difference between the connected grid nodes. With (Eq. 3.2) and (Eq. 3.1) we obtain the model equation system (Eq. 3.12) and (Eq. 3.13) in matrix vector form [5]

$$\boldsymbol{L_{pl}} \frac{di_{pl}(t)}{dt} = \boldsymbol{M} v_{gn}(t) - \boldsymbol{R_{pl}} i_{pl}(t) \tag{3.12}$$

$$\boldsymbol{C_{cap}} \begin{bmatrix} \frac{dv_P(t)}{dt} \\ \frac{dv_C(t)}{dt} \end{bmatrix} = -\boldsymbol{M} i_{pl}(t) + \begin{bmatrix} \boldsymbol{K_D}(V_{ref,P} - v_P(t)) \\ P / v_C(t) \end{bmatrix} \tag{3.13}$$

The overall producer-consumer direct current microgrid model is shown in Figure 3.1. Based on this system we can now define the network model of the prosumer-based structured microgrid in the next Section. Note that the load power is only in this Section considered as constant. Also, there is no higher-

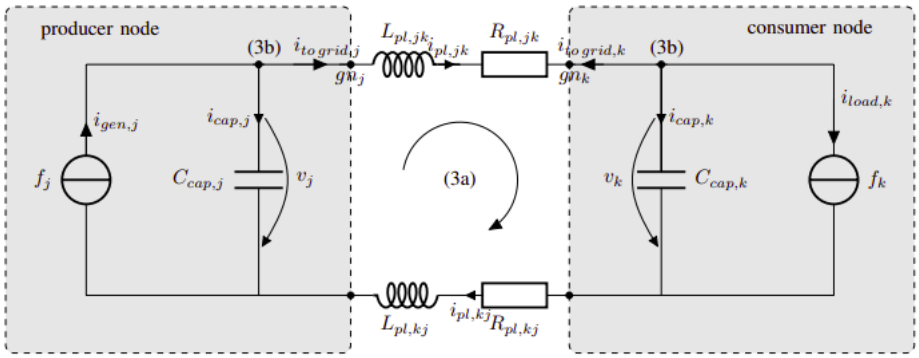

Figure 3.1: Model for the direct current microgrid. Here, simplified for $n = 2$ nodes and time argument (t). Index j for producers, index k for consumers. $i_{gen,j}(t) = f_j(t)$ are droop controlled sources (Eq. 3.3), $i_{load,k}(t) = f_k(t)$ are constant power load (Eq. 3.4) [5]

layer control input. For the purpose of the iterative learning control we must assume in the next Section that the power demand has a periodic and fluctuating character.

3.1.2 Network model of the prosumer-based DC microgrid with lower-layer control

Figure 3.2: Model for the proposed prosumer-based direct current microgrid. Here, simplified for $n = 2$ nodes.

Based on the network modeled in Section 3.1.1, the slightly different prosumer-based direct current model can be derived in this Section. Additionally the low-level control input is going to be used as well as the high-level control input, since the low-layer control structure must be considered now. It is also now going to be assumed, that the load profiles have periodic components in contrast to Section 3.1.1.
In Section 2.1.4, it is considered that a node in a prosumer-based dirrent current microgrid can feed power into the microgrid, receive power from it or not participate in the microgrid, see Definition 2. Therefore there is no differentiation anymore between producer and consumer; A node must be modeled as one unit combined by the two at first seperate units.
Recall Figure 3.1, where the model for the direct current, non-prosumer-based microgrid is illustrated. The seperate producer j node is connected via the power line jk with the consumer node k. For the prosumer-based model, this schema

can be adapted, but the power line jk connects prosumer j with prosumer k, respectively. Furthermore, $N \in \mathcal{N}$ is the number of prosumers. Therefore, the grid node voltages are now defined as
$v_g n(t) = (v_1(t), ..., v_n(t))^T$. Obviously, the node structure needs to be changed, so it can produce but also consume simultaneously.
A suggestion for this idea would be to combine the generation and load unit into one node. This means that a prosumer in the network is a *parallel connection* of the generation current source and the load current source. As in system (3.8) the generated and load current remain for a prosumer j in a microgrid and $t \in \mathbb{R}$ as follows:

$$i_{gen}(t) = K_{D,j}(V_{ref,j} - v_j(t)) \tag{3.14}$$

$$i_{load}(t) = \frac{P_j^d(t)}{v_j(t)} \tag{3.15}$$

For the control, the demand power $P_j(t)$ will be the uncontrolled net power demand $P_j^d(t) = P_j^f(t) + P_j^p(t)$ at node j which accounts for the actual demand or uncontrollable infeed from renewable sources [4]. It consists of a fluctuating part $P_j^f(t)$ and a and a periodic part $P_j^p(t)$ whose period is empirically known/estimated [4], $t \in \mathbb{R}$.
The model of the power line can be defined as in assumption v (3.1.1) in (Eq.3.5) with

$$L_{pl,jk}\frac{di_{pl,jk}(t)}{dt} = v_{pl,jk}(t) - R_{pl,jk}i_{pl,jk}(t)$$

The prosumers have the same behavior as explained in assumption vi in Section 3.1.1 and can be modeled by a current source and load. A current source for the higher-layer control input is added parallel to the generating current source. The ILC current is derived from the high-level control input u^{ILC} (Section 3.2.2)with:

$$i_{ILC,j}(t) = \frac{\boldsymbol{u}_j^{ILC}(t)}{v_j(t)}, v_j(t) \neq 0 \tag{3.16}$$

For this we need to consider Figure 3.2 and the Kirchhoff current law, which led to the following equation:

$$\begin{aligned} C_{cap,j}\frac{dv_j(t)}{dt} &= i_{ILC,j}(t) + i_{gen,j}(t) - i_j(t) - i_{load,j}(t) \\ C_{cap,j}\frac{dv_j(t)}{dt} &= \frac{\boldsymbol{u}_j^{ILC}(t)}{v_j(t)} + K_D(V_{ref,j} - v_j(t)) - i_j(t) - \frac{P_j^d(t)}{v_j(t)} \end{aligned} \tag{3.17}$$

(Eq. 3.17) can be derived using (Eq. 3.14) and (Eq. 3.15)
For the inherent ODE formulation of (3.17) and (Eq.3.5) , we obtain the closed-loop equations

$$\boldsymbol{L_{pl}}\frac{d\,i_{pl}(t)}{d\,t} = \boldsymbol{M}v_{gn}(t) - \boldsymbol{R_{pl}}i_{pl}(t) \tag{3.18a}$$

$$\boldsymbol{C_{cap}}\frac{d\,v_{gn}(t)}{d\,t} = \frac{\boldsymbol{u}^{ILC}(t) - P^d(t)}{v_{gn}(t)} + \boldsymbol{K_D}(V_{ref} - v_{gn}(t)) - \boldsymbol{M}i_{pl}(t), t \in \mathbb{R} \tag{3.18b}$$

with $P^d(t) = (P_1^d(t), ..., P_n^d(t))$ being the vector of the uncontrolled net power demand, consisting of a fluctuating part $P^f(t) = (P_1^f(t), ..., P_n^f(t))$ and a periodic part $P^p(t) = (P_1^p(t), ..., P_n^p(t))$.
To achieve bounded voltage deviation from the reference voltage of the grid in the lower layer of the control architecture, we use a first order controller that we refer to as the lower-layer control. The control law is given as:

$$\boldsymbol{u}^{LI} = \underbrace{\boldsymbol{K_D}(V_{ref} - v_{gn}(t))}_{i_{gen}(t)} \cdot v_{gn}(t) \tag{3.19}$$

3.2 Higher-layer controller structure and design

Recall that in this work the control consists of two layers, the low-level controller, see Section 3.1.2 and the high-level control. The low-level controller is responsible for the bounded voltage deviation and the power sharing.
In this Section the iterative learning controller is going to be, based on the provided information in Section 2.3.3 and the "*Iterative learning control in prosumer-based microgrids with hierarchical control*" paper [4] introduced and modeled. The *control objectives* are derived in Section 3.2.1, and based on the next Section 3.2.2 the *higher-layer controller* is designed.

3.2.1 Control objectives

The proposed overall control objectives are derived in [4] and [3] and combined.

(1) Bounded voltage deviation from the reference voltage of the grid, i.e., [3].

$$|v_{gn,j}(t) - V_{ref,j}| \leq e_V, \forall\, t \in \mathbb{R}, j \in \mathcal{N}, e_V \in \mathbb{R}_{>0} \tag{3.20}$$

(2) Power sharing of the prosumers [3]: The same amount of power is fed from all of the prosumers into the grid when the time deriatives are zero [3].

$$\lim_{t\to\infty} P_{gn,j}(t) = \lim_{t\to\infty} P_{gn,k}(t), j, k \in \mathcal{N} \tag{3.21}$$

(3) Low-level control energy small [4]:

$$\sum_{h=1}^{24} ||\mathbf{y}^{c,h}||_2 = \sum_{h=1}^{24} || \int_{t_h^c}^{t_{h+1}^c} \mathbf{u}^{LI}(\tau) d\tau ||_2 \ll \sum_{h=1}^{24} ||\mathbf{u}^{c,h}||_2, c \in \mathbb{N}_0, h = 1, ..., 24 \quad (3.22)$$

(1) is achieved through the low-level control regardless of the additional higher-layer control input as long as $|\sum_{j \in \mathcal{N}} u^{ILC} - P_j^d| \leq |\sum_{j \in \mathcal{N}} P_j^d|$ [4]. In the next Section the scheme of the higher-layer controller is designed. In this work, an iterative learning controller is used for this purpose, proposed by Lia Strenge et al. in [4]

3.2.2 Iterative learning controller

Recall (Eq. 2.3). The new input is designed so that the new error is smaller than the previous input. The proposed ILC approach and its parameters by [4] learns a power infeed that compensates the periodic demand component. With $c \in \mathbb{N}_0$ and $h = 1, ..., 24$, with c being the cycle with a duration of one day and h the hour, the following learning input is implemented:

$$\mathbf{u}^c = \mathbf{Q}(\mathbf{u}^{c-1} - \mathbf{L}\mathbf{y}^{c-1}), c > 0 \quad (3.23)$$

The daily input $\mathbf{u}^c$ is adjusted based on the error that is the low-level control energy of the previous day $\mathbf{y}^{c-1}$. The desired low-level control energy is zero. $\mathbf{L} \in \mathbb{R}^{24N \times 24N}$ is the learning matrix, which is simply chosen to be $\mathbf{L}(\kappa) = \kappa I$ with a single scalar parameter $\kappa \in \mathbb{R}_{>0}$ and $\mathbf{Q}$ is a butterworth low-pass filter with a relative cutoff frequency of $f_c = 1/6$ [4].
The proposed ILC approach can reduce the lower-layer control energy, since the required should be zero by learning the periodic demand [4].
The overall proposed nonlinear model with iterative learning control (3.18), (3.19) and (3.23) shall be validated, simulated and analyzed under certain assumptions in the next Chapter.

Chapter 4

Simulation-based analysis

In the earlier sections the preliminaries and model frame were introduced and expained. In this section the model validation and results of simulating will be analysed in Section 4.1 without the higher-layer control and in Section 4.2 with. The discussion of the results will take place in Section .
The code of the nonlinear model is written in Julia 1.4.0 and is available on request or on github. The simulations were performed using the DifferentialEquations.jl package, Rackauckas and Nie (2017), and the Rodas4p solver, Wanner and Hairer (1996).
For the model validation and the result representation we consider a fully connected microgrid of 4 nodes ($N = 4$, see notations at Section 3.1.2). Every node has a local low-level controller as well as a high-level controller, i.e., without additional communication [4]. The model parameters are summarized in Table 4.1.

Level	**Symbol**	**Value**	**SI Unit**	**Description**
	N	4	-	number of grid nodes in the network
	j	$1,2,3,4$	-	grid nodes in the network
	jk	$1,2,3,4,5,6$	-	power lines in the network
low-level	$C_{cap,j}$	0.01	F	given capacitance of the capacitor at the grid node
	$L_{pl,jk}$	$0.237 \cdot 10^{-4}$	H	given power line inductance
	$R_{pl,jk}$	$0.532 \cdot 10^{-1}$	Ω	given power line resistance
	$v_{gn,j}$	48	V	given reference voltage of the microgrid
	$K_{D,j}$	varies	$\frac{1}{\Omega}$	droop coefficient of the prosumer
high-level	κ	varies	$\frac{1}{h}$	learning parameter
	f_c	$\frac{1}{6}$	-	relative cutoff frequency of Butterworth lowpass filter Q

Table 4.1: Values of parameters

4.1 Model validation and results of the network without higher-layer control and periodic load profiles

To model the direct current prosumer-based microgrid network it is analyzed the compound plant, i.e. only the network model with lower-layer control at first. Figures 4.2 and 4.1 are the simulated currents per edge/power line and voltages per node. The voltage reference per node amounts $V_{ref,j} = 48\,\mathrm{V}$ and the node voltages oscillate in an interval $I = [47\,\mathrm{V}, 48\,\mathrm{V}]$, as seen in Figure 4.1.

Interpretation

Assuming that the difference of node voltage and reference voltage should not exceed a value of 2 V,so:

$$|v_{gn,j}(t) - V_{ref,j}| \leq 2\,\mathrm{V}, \forall\, t \in \mathbb{R}, j = 1,2,3,4$$

one can see that the control objective (1) of Section 3.2.1 is respected.
Control objective 2 states for the power sharing of the prosumers. When the time deriatives are zero, i.e. the highest and lowest values of the node powers, it can be seen in Figure 4.3 that every node feeds almost the same amount of power into the grid. The difference of the peaks of power for every node is max. 1 W. Section 4.2 refers to the validation of the network model with the iterative learning control.

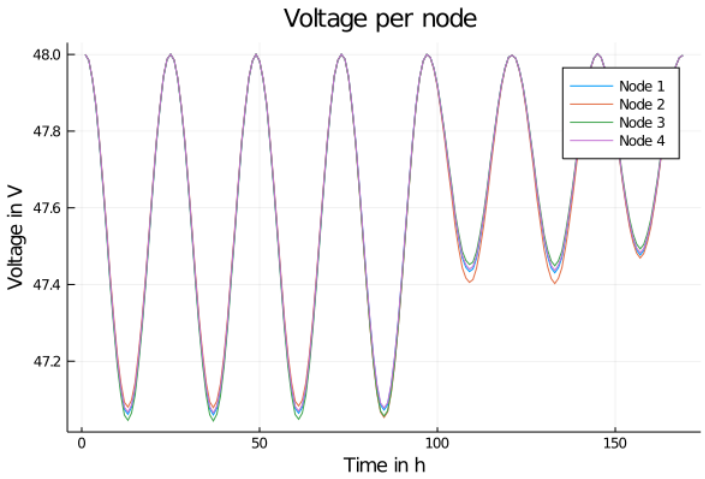

Figure 4.1: Voltage $v_{gn,j}(t)$ per node $j = 1,2,3,4$ with no higher-layer control for 7 days in hours

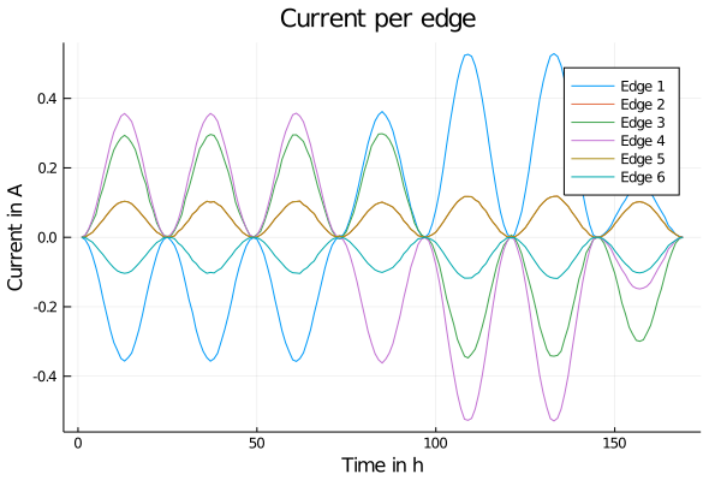

Figure 4.2: Current per edge with no higher-layer control for 7 days in hours

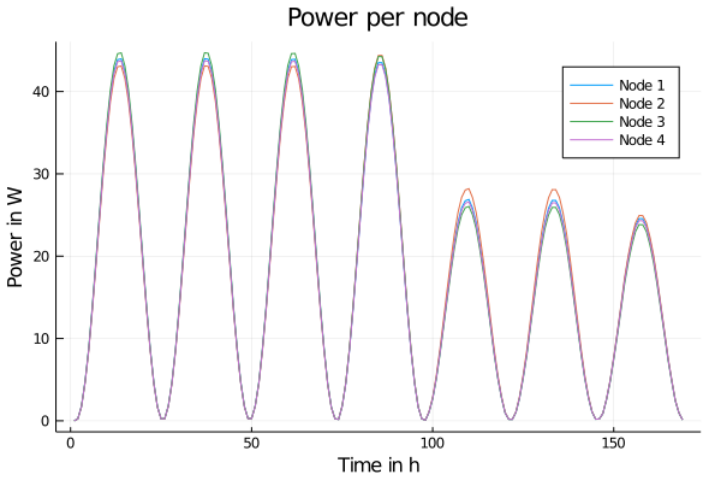

Figure 4.3: Low-level control energy with no higher-layer control for 7 days in hours

4.2 Model validation and results of the network with higher-layer control and periodic load profiles

For this section, we now consider the complete derived model of the prosumer-based direct current fully connected microgrid with the iterative learning control and without additional communication. The values of the parameters can be taken from Table 4.1.

In order to validate the proposed hierarchical controller and to discuss the behavior of it, the two control parameters, the learning parameter of the ILC κ and the droop coefficient of the lower-layer control $\boldsymbol{K_D}$ will be varied, so that different case distinctions arise. In particular, we will observe the following cases:

- **Case 1:** Validation of the network modeling with $K_{D,j} = 1$ and $\kappa = 1$ (4.2.1)
- **Case 2:** Variation of $\boldsymbol{K_D}$, $\kappa = 1$ (4.2.2)
- **Case 3:** Variation of κ, $K_{D,j} = 1$ (4.2.3)
- **Error dynamics for different learning gains**

4.2.1 Case 1: Validation of the network modeling with $K_{D,j} = 1$ and $\kappa = 1$

Case 1 serves the purpose to validate the overall nonlinear system model (3.18), (3.19) (3.23). The main results of case 1 are presented in the following.

Description

Figure 4.4 shows the injected power at the grid nodes. Recall that this is not the low-level control energy. Furthermore, it is the power that is calculated by the product of the voltages $v_{gn,t}(t)$ and the injected currents $i_{gn,t}(t)$ at the grid nodes.
Figure 4.5 shows the sum over all nodes of the demand P_j^d (transparent blue), the hourly integrated low-level control energy $y_j^{c,h}$ (red line) and the input from the ILC u_j^{ILC} (black line) for a learning scenario with daily step changes of the peak demand. The peak demand is stepping after four days and again after three or four more days and is different at each node (Figure 4.5, bottom).As soon as the peak demand steps the low-layer control energy increases on the same day and decreases again on the next day. E.g. a peak of the energy of approx. 50W appears on day 5.

Interpretation

Regarding the injected power of Figure 4.4, for every hour of the 7 days, the node energies are balanced and power sharing is achieved. This means that if at least one node produces an specific amount of energy, then the remaining node(s) consume this amount, the sum of all injected powers at every grid node is zero.

Regarding the two-layer control (Figure 4.5) it can be observed that the local ILC learns the local demand. It also steps after four days with a one-day delay (since on Day 1 the ILC does not have learned anything yet). By observing the low-level control energy it can also be determined, that power sharing in the network is achieved through the low-layer control input u_j^{LI} from the first day. The sum of the lower-layer control energy $y^{c,h,s} = \sum_j y_j^{c,h}$ is distributed on all 4 nodes proportionally for every $K_{D,j} \in \mathbb{R}_{>0}$ (homogenous scenario). The following relationship (for every amount of prosumer-nodes) arises:

$$N \cdot K_{D,j} \cdot y_j^{c,h} = y^{c,h,s}, N \in \mathcal{N} \subseteq \mathbb{N}, j = 1,2,...,N, K_{D,j} \in \mathbb{R}_{>0} \tag{4.1}$$

$$4 \cdot y_j^{c,h} = y^{c,h,s}, N = 4, K_{D,j} = 1, j = 1,2,3,4$$

Furthermore, the ILC learns based on the synchronized state for the whole network reducing successfully the lower-layer control energy at all nodes. The results of the summed quantities show that, after each demand peak step the ILC learns to compensate the new demand and thereby decreases the low-level control energy to less than 10% of its original value within two days.
The fast increase of low-level control energy during the fifth day occurs, since the ILC has not learned yet the new peak demand and therefore reacts by not being able to keep the low-level energy small.
Case 1 serves as the "*default*" case for the purpose of validating the network modeling. However, both the low-level and higher-layer controller must be tested in further cases to clarify the conclusions. E.g. in the next case the influence of the control parameter $K_{D,j}$ will be analyzed to confirm the proposed control objectives of power sharing and bounded voltage (3.2.1).

4.2.2 Case 2: Variation of K_D and $\kappa = 1$

Case 2 serves the purpose of analyzing the learning behavior of the hierarchical control for different droop coefficients $K_{D,j}$, $j = 1,2,3,4$ and the influence of this coefficient on the power sharing. For this, K will be set to a vector of different values in order to examine how the variation of droop coefficients affects (quantitatively) the low-level energy.

Description

Note that the coefficient determines the voltage drop at the output, in our case the difference $V_{ref,j} - v_{gn,j}(t)$, $j = 1,2,3,4$.Two different *heterogeneous* scenario have been selected for this case, where different droop coefficients are applied for every node. *Homogeneous* scenarios include the same coefficients on every

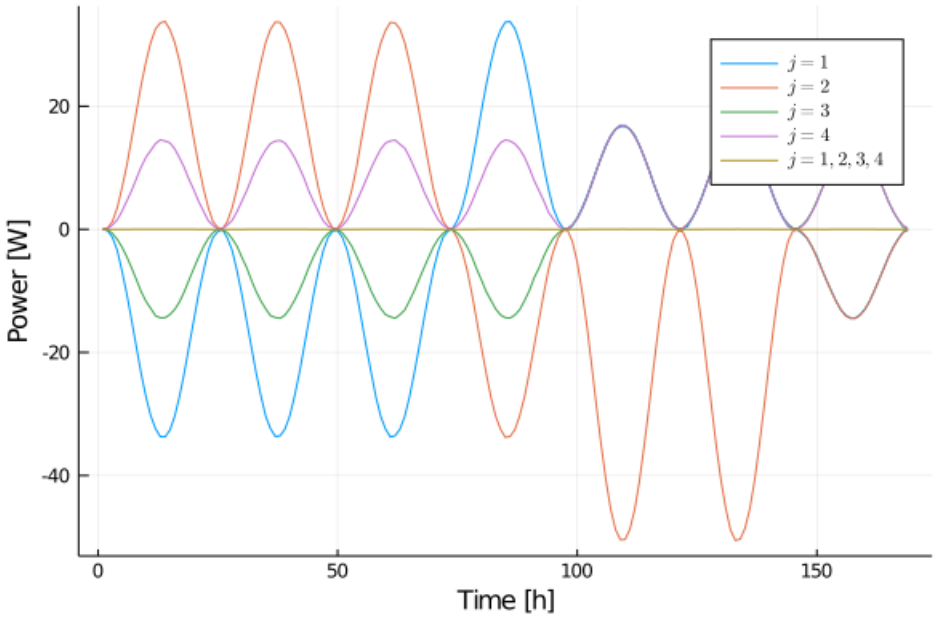

Figure 4.4: Injected power at every node $j = 1,2,3,4$ and the sum (yellow line)

node. The following droop coefficients $\boldsymbol{K_D} = (K_{D,1}, K_{D,2}, K_{D,3}, K_{D,4})^T$ were used:

$$\boldsymbol{K_{D,a}} = \begin{pmatrix} 0.1 \\ 1 \\ 2 \\ 5 \end{pmatrix} \Omega^{-1}, \boldsymbol{K_{D,b}} = \begin{pmatrix} 1 \\ 1 \\ 0.1 \\ 1 \end{pmatrix} \Omega^{-1}$$

Four different values have been assigned for every node in $\boldsymbol{K_{D,a}}$ and one different value in $\boldsymbol{K_{D,b}}$. Figures 4.6 and 4.7 show the demand P_j^d (transparent blue), the hourly integrated low-level control energy $y_j^{c,h}$ (red line) and the input from the ILC u_j^{ILC} (black line) and the sums $\sum_j P_j^d$, $\sum_j y_j^{c,h}$ and $\sum_j u_j^{ILC}$ for learning scenarios with daily step changes of the peak demand for the droop coefficient $\boldsymbol{K_{D,a}}$ and $\boldsymbol{K_{D,b}}$, respectively, $j = 1,2,3,4$.

$\boldsymbol{K_{D,a}}$: Changing the droop coefficient for every node obviously changes also the value of the daily low-level control energy. E.g. in node 1, the droop coefficient $K_{D,a,1} = 0.1\,\Omega^{-1}$ was applied. Since this parameter is also an immediate part of u^{LI} (Eq. 3.19) the hourly integrated energy is reduced by almost $\frac{1}{10}$ compared to the energy on node 2 with $K_{D,a,2} = 1\,\Omega^{-1}$. However, the energy on node 2 is smaller than the energies in Case 1 ($K_{D,j} = 1\,\mathrm{h}^{-1}$). For node 4, the droop coefficient $K_{D,a,4} = 5\,\Omega^{-1}$ was applied. The integrated energy in this node is almost

5 times bigger compared to node 2. Increased, i.e. decreased lower-layer control energy leads to equivalently increased, i.e. decreased ILC power, which can also be observed in Figure 4.6. The sum of powers on each node is observed to be equivalent compared to Case 1 ($K_{D,j} = 1\,\mathrm{h}^{-1}$).

$\boldsymbol{K_{D,b}}$**:** In this case, it is observed that that nodes $j = 1$, $j = 2$ and $j = 4$, where the droop coefficient $1\,\mathrm{h}^{-1}$ is applied show the same amount of low-level control energy. The energies are bigger than the Case 1 ($K_{D,j} = 1\,\mathrm{h}^{-1}$) energies with the same droop coefficient. For node $j = 3$ with $K_{D,b,3} = 0.1\,\mathrm{h}^{-1}$ the energy is reduced by almost $\frac{1}{10}$ compared to the other nodes. The sum of powers on each node is equivalent compared to Case 1 ($K_{D,j} = 1\,\mathrm{h}^{-1}$).

Interpretation

The higher the droop coefficient is, the smaller is the difference between the two voltages. A higher droop coefficient also means a faster reaction of the currents, resulting increasing currents. The low-level control energy per node increases for increasing $K_{D,j}$ and decreases for decreasing $K_{D,j}$, respectively.
The sum of the low-layer control energy at every node of the nonlinear model $y^{c,h,s} = \sum_j y_j^{c,h}$ with the parameters of Table 4.1 is for every $K_{D,j} \in \mathbb{R}_{>0}$ equal. Changing the droop coefficient in a heterogenous scenario will only change the distribution of the integrated energies on every node and therefore the value of integrated energies per node. A higher droop coefficient at a node will increase the lower-layer control energy there and reduce it on the rest of the nodes. Likewise the reduction of the droop coefficient at a node will increase the energy of the other nodes. The sum of powers at every node is distributed on all nodes proportionally, depending on the coefficient. For every heterogenous scenario the following relationship arises:

$$\sum_{j}^{N} K_{D,j} \cdot y_j^{c,h} = y^{c,h,s}, N \in \mathcal{N} \subseteq \mathbb{N}, j = 1,2,...,N, K_{D,j} \in \mathbb{R}_{>0} \tag{4.2}$$

$$0.1 \cdot y_1^{c,h} + y_2^{c,h} + 2 \cdot y_3^{c,h} + 5 \cdot y_4^{c,h} = y^{c,h,s}, j = 1,2,3,4, N = 4, \boldsymbol{K_D} = \boldsymbol{K_{D,a}}$$

$$y_1^{y,h} + y_2^{c,h} + 0.1 \cdot y_3^{c,h} + y_4^{c,h} = y^{c,h,s}, j = 1,2,3,4, N = 4, \boldsymbol{K_D} = \boldsymbol{K_{D,b}}$$

Fair power sharing is achieved for all prosumers. This is necessary in order to assure the same rights for each user regardless of the position in the microgrid. In other words, when a prosumer of the grid feeds a greater amount of power into the grid, the other prosumer will feed a equivalently smaller amount of power intro the grid, to keep the total energy balance of the microgrid.
As also in Case 1 ($K_{D,j} = 1\,\mathrm{h}^{-1}$), the peak demand steps after four days and again after 3 days and is different at each node (Figures 4.6 and 4.7,bottom). The local

ILC learns to compensate the new local demand after each demand peak step, stepping with the same frequency as the local demand and the low-level control energy decreases also after the day after the ILC has learned the new periodic demand.
The next case varies the learning parameter κ for a constant droop coefficient $K_{D,j} = 1\,\Omega^{-1}$ to analyze the learning behavior of the iterative learning controller.

4.2.3 Case 3: Variation of κ and $K_{D,j} = 1\,\Omega^{-1}$

Case 3 serves the purpose of analyzing the learning behavior of the hierarchical control for different learning parameter $\kappa \in]0,2[\text{hour}^{-1}$.
Figure 4.8 presents the sum over all nodes of the demand $\sum_j P_j^d$, the low-level control energy $\sum_j y_j^{c,h}$ and the input from the ILC $\sum_j u_j^{ILC}$ for a learning scenario with artificial step changes of the peak demand for a variety of different learning parameter κ.

Observations

$\kappa \in]0,1[\text{h}^{-1}$: For $\kappa \in]0,1[\text{h}^{-1}$ the ILC power regulates slowly during the second days and learns the first peak of demand. During the fifth day, for $\kappa = 0.25\,\text{h}^{-1}$ the ILC has reached almost 70% of the demand and in the next days it regulates to the next amplitude. For $\kappa = 0.5\,\text{h}^{-1}$ and $\kappa = 0.75\,\text{h}^{-1}$ the learning improves, while the ILC power approaches during the fifth day the peak demand. However, the response for $\kappa \in]0,1[\text{h}^{-1}$ is slower than for $\kappa = 1\,\text{h}^{-1}$, where the ILC has reached the amount of the demand power during the second day.
In addition the low-level control energy decreases for $\kappa \in]0,1[\text{h}^{-1}$, with a bigger κ having a faster decrease as result. E.g. the sum of low-level control energy over all nodes decreases faster for $\kappa = 0.75\,\text{h}^{-1}$ than for $\kappa = 0.5\,\text{h}^{-1}$. The negative peak of the energy that is observed for $\kappa = 1\,\text{h}^{-1}$ in Figure 4.5 grows with an increasing κ.

$\kappa \in]1,2[\text{h}^{-1}$: For κ in $]1,2[\text{h}^{-1}$ the ILC shows a quick regulation during the second day and learns the first peak of demand. The more κ increases the bigger the amount of the power in the second day gets and the more the learning behavior of the ILC downgrades, as it shows a variety of different amplitudes. E.g. in Figure 4.8 for $\kappa = 1.5\,\text{h}^{-1}$ the demand amplitude is 180 during the four first days and the ILC shows an amplitude of 250 in the second and an amplitude of approx. 130 in the third day For $\kappa = 1\,\text{h}^{-1}$ the ILC learn the demand by adapting the amplitudes of 180 immediately after the first day.

The lower-layer control energy over all nodes decreases in $\mathbb{R}_{>0}$ and increases in $\mathbb{R}_{<0}$ after the first day for all $\kappa =]1,2[\mathrm{h}^{-1}$, while the negative peak during the fifth day ($\kappa = 1\,\mathrm{h}^{-1}$) decreases.

Interpretation

If one compares the single representations from Figure 4.8 with each other, as well as with the representation in Figure 4.5, one quickly notices that in the interval $\kappa \in]0,1[\mathrm{h}^{-1}$ the learning behavior of the iterative learning controller improves for increasing κ and deteriorates for increasing κ in $]1,2[\mathrm{h}^{-1}$. For $\kappa = 1\,\mathrm{h}^{-1}$ the learning scenario is the most efficient. The controller reacts sensitive to the change of the demand amplitudes, so that the integrated energy increases again, until the controller has learned the demand. In the next Section it will be studied how the error convergence in the iteration domain depends on the choice of the scalar learning parameter.

4.2.4 Error dynamics for different learning gains

We study how the error convergence in the iteration domain depends on the choice of the scalar learning parameter κ. Again, the nonlinear model with the two-layer control is used with modified demand pattern [4]. The periodic peak demand is between 0.6 and 0.9 W/W at the different nodes and the fluctuating component varies randomly from day to day within $[-0.7, 0.7]$W/W. The smaller amplitudes of the demand in this case ensure that the increased low-level energy on day 5 from the upper sections are again minimised.
Different values of $\kappa = [0,2]\mathrm{h}^{-1}$ are considered. For each value, the error norm, i.e. a measure of the overall low-level control energy is calculated for each of the 20 first days [4]. Figure 4.9 shows the results of the study. For $\kappa \in [0,1]$ the error norm converges faster for increasing κ. The error norm is constant for $0\,\mathrm{h}^{-1}$ (no learning) over the cycles. For $\kappa \in [1,2]$ the error norm converges slower for increasing κ, i.e the error norm is not converging for $\kappa = 2\,\mathrm{h}^{-1}$. The convergence of the error norm is therefore fastest for $\kappa = 1\,\mathrm{h}^{-1}$. This observation confirms the statement from Case 3, that for $\kappa = 1\,\mathrm{h}^{-1}$ the learning scenario is the optimal.

4.3 Discussion

Taking all results from Sections 4.1 and 4.2 for the prosumer-based direct current microgrid with iterative learning control, it can be said that:

1. The suggested prosumer-based direct current microgrid framework (Eq. 3.18) with lower-layer control (Eq. 3.19) achieves voltage droop and power sharing with the corresponding droop coefficient $\boldsymbol{K_D}$. Depending on the individual purposes, e.g. if a specific prosumer should produce/consume a bigger amount of power, the coefficient, as well as the compound plant parameters can be varied. For an equal distribution of power inbetween the nodes, choosing for the droop coefficient $K_{D,j} = 1\,\Omega^{-1}$ will complete this purpose.

2. The suggested iterative learning controller (Eq. 3.23) not only learned the local demand and contributed with the low-level controller by ensuring the power sharing. Instead, it saved the low-level control energy, showing some sensitivity, whenever the peak demand changed.

3. The low-level control energy was decreased for almost every learning parameter κ. By calculating the error convergence for the low-level control energy, it was showed that the optimal learning behavior is achieved at $\kappa = 1\,\mathrm{h}^{-1}$.

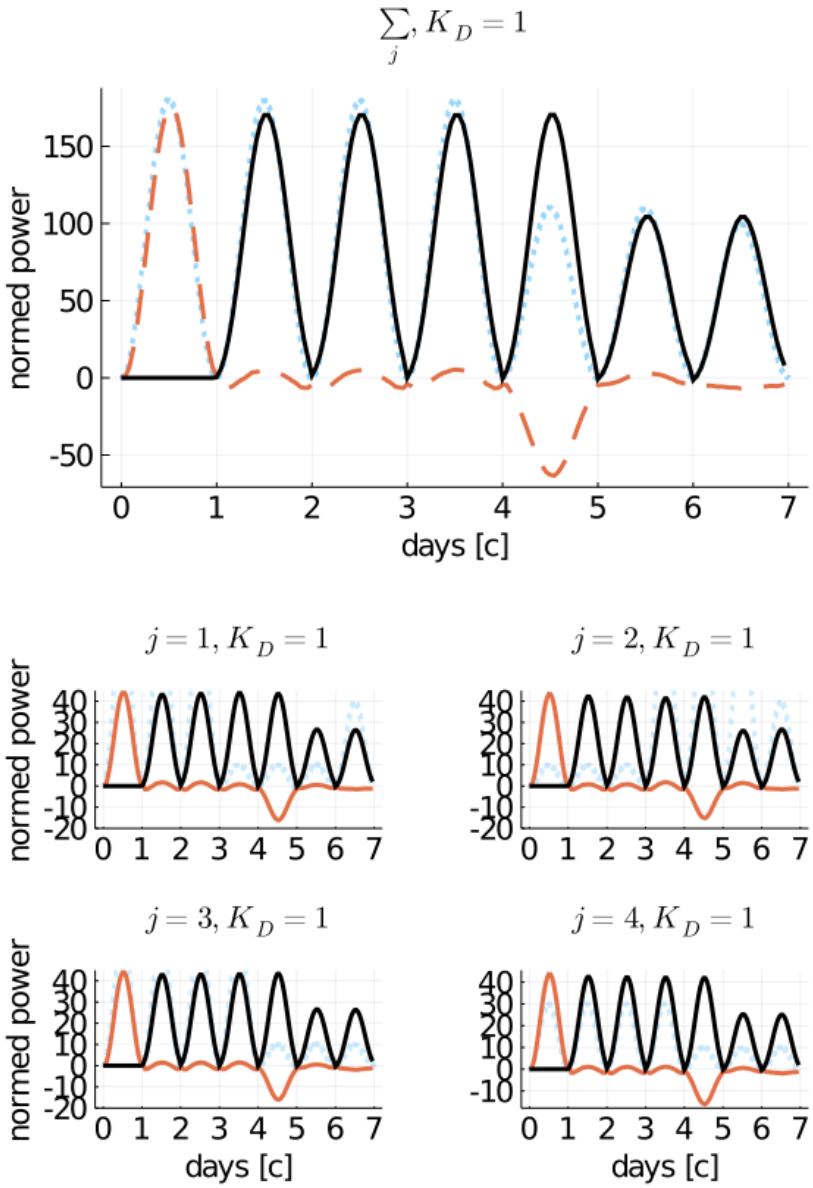

Figure 4.5: Top: sum for all nodes $j = 1,2,3,4$, $\kappa = 1\,\mathrm{h}^{-1}$ and $K_{D,j} = 1\,\Omega^{-1}$; transparent blue: $\sum_j P_j^d$, dashed red: $\sum_j y_j^{c,h}$, solid black: $\sum_j u_j^{ILC}$; Bottom: seperately for nodes $j = 1,2,3,4$, $\kappa = 1\,\mathrm{h}^{-1}$, $K_{D,j} = 1\,\Omega^{-1}$; transparent blue: P_j^d, solid red: $y_j^{c,h}$, solid black: u_j^{ILC}

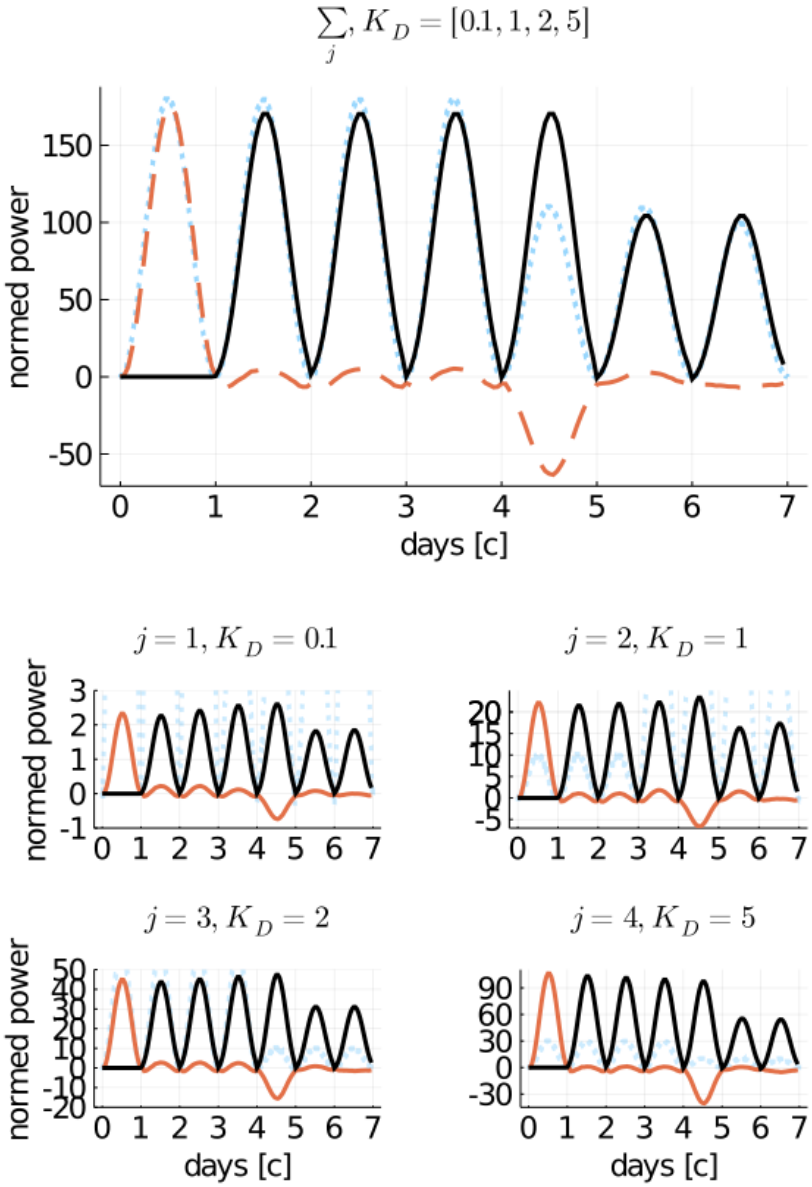

Figure 4.6: Top: sum for all nodes $j = 1,2,3,4$, $\kappa = 1\,\mathrm{h}^{-1}$ and $\boldsymbol{K_{D,a}} = (0.1,1,2,5)^T\Omega^{-1}$; transparent blue: $\sum_j P_j^d$, dashed red: $\sum_j y_j^{c,h}$, solid black: $\sum_j u_j^{ILC}$; Bottom: seperately for nodes $j = 1,2,3,4$, $\kappa = 1\,\mathrm{h}^{-1}$, $\boldsymbol{K_{D,a}} = (0.1,1,2,5)^T\Omega^{-1}$; transparent blue: P_j^d, solid red: $y_j^{c,h}$, solid black: u_j^{ILC}

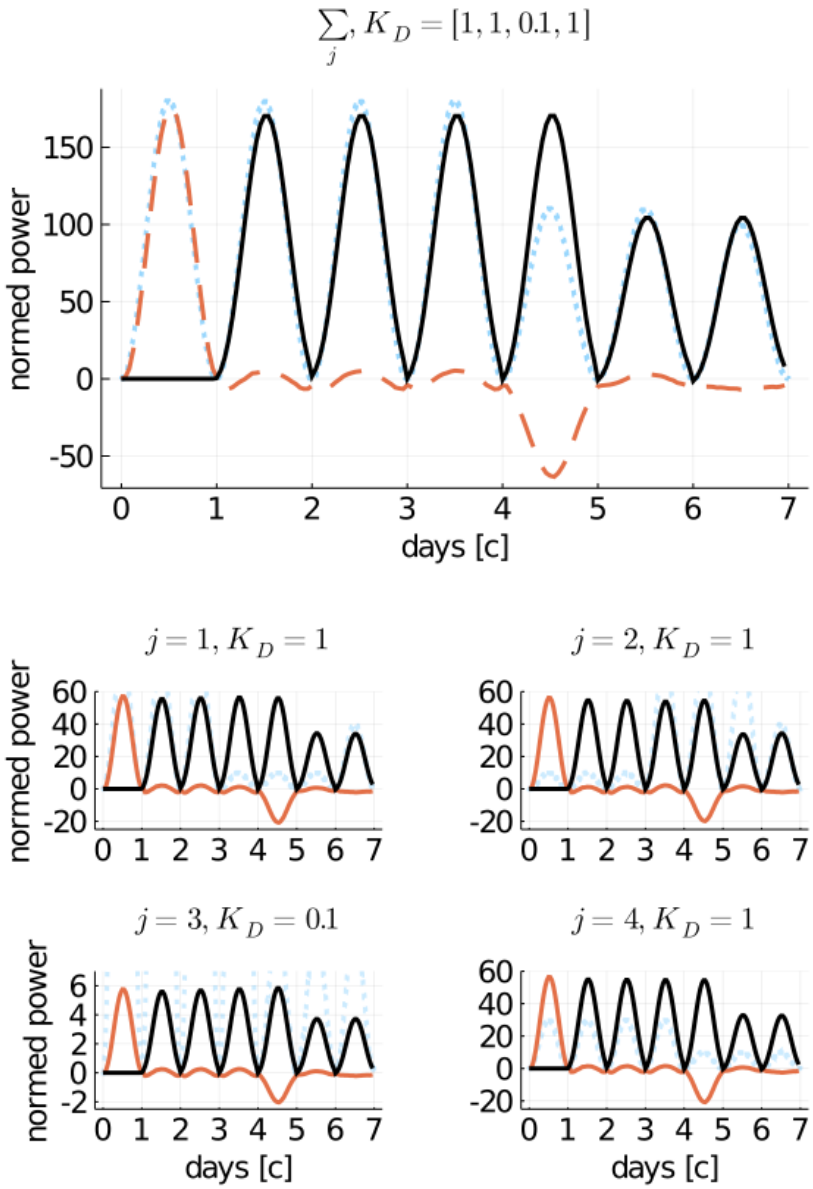

Figure 4.7: Top: sum for all nodes $j = 1,2,3,4$, $\kappa = 1\,\mathrm{h}^{-1}$ and $\boldsymbol{K_{D,b}} = (1,1,0.1,1)^T\Omega^{-1}$; transparent blue: $\sum_j P_j^d$, dashed red: $\sum_j y_j^{c,h}$, solid black: $\sum_j u_j^{ILC}$; Bottom: seperately for nodes $j = 1,2,3,4$, $\kappa = 1\,\mathrm{h}^{-1}$, $\boldsymbol{K_{D,b}} = (1,1,0.1,1)^T\Omega^{-1}$; transparent blue: P_j^d, solid red: $y_j^{c,h}$, solid black: u_j^{ILC}

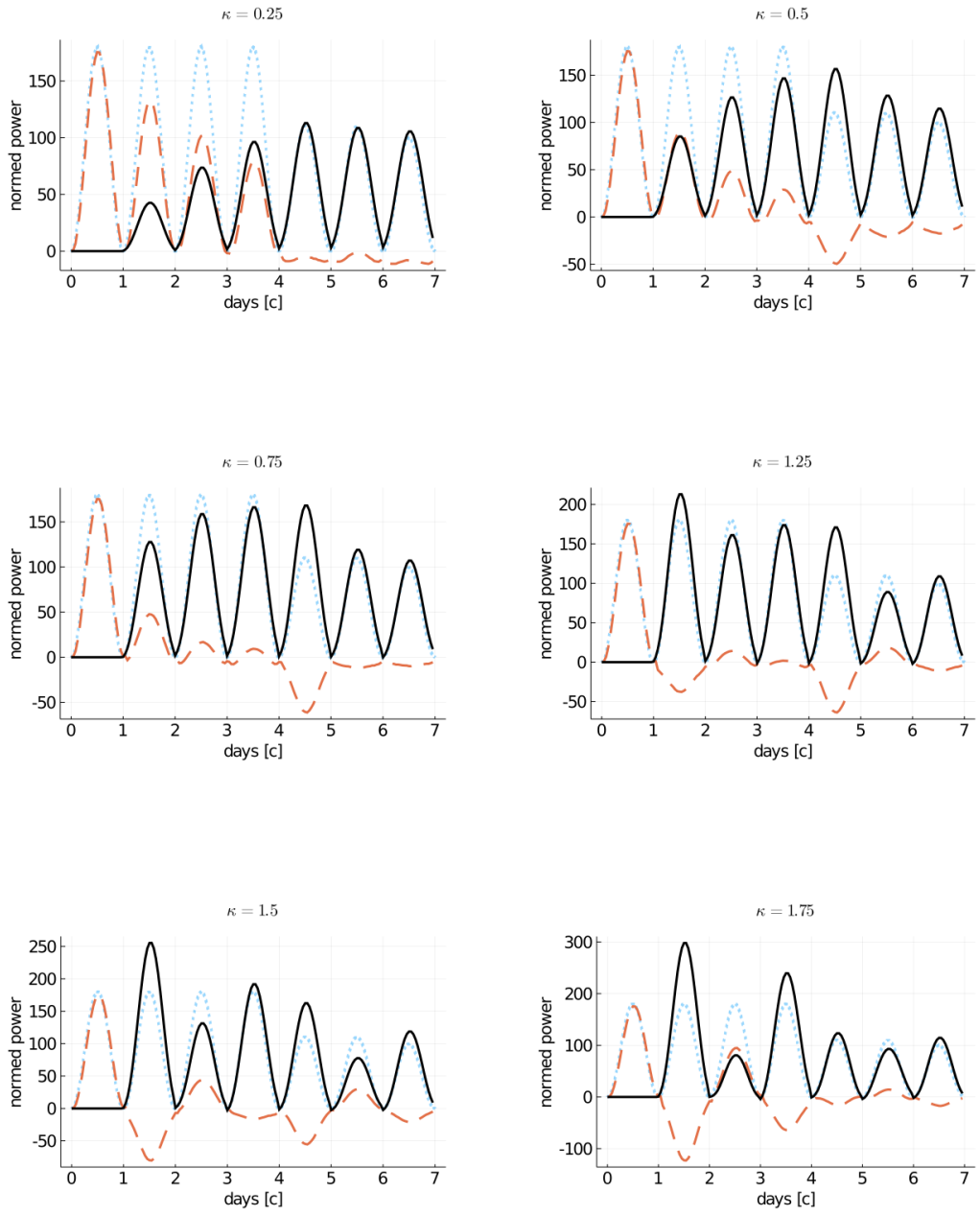

Figure 4.8: sum for all nodes $j = 1,2,3,4$ with $K_{D,j} = 1\Omega^{-1}$; Top left: $\kappa = 0.25\,\mathrm{h}^{-1}$; Top right: $\kappa = 0.5\,\mathrm{h}^{-1}$; Center left: $\kappa = 0.75\,\mathrm{h}^{-1}$; Center right: $\kappa = 1.25\,\mathrm{h}^{-1}$; Bottom left: $\kappa = 1.5\,\mathrm{h}^{-1}$; Bottom right: $\kappa = 1.75\,\mathrm{h}^{-1}$; transparent blue: $\sum_j P_j^d$, dashed red: $\sum_j y_j^{c,h}$, solid black: $\sum_j u_j^{ILC}$;

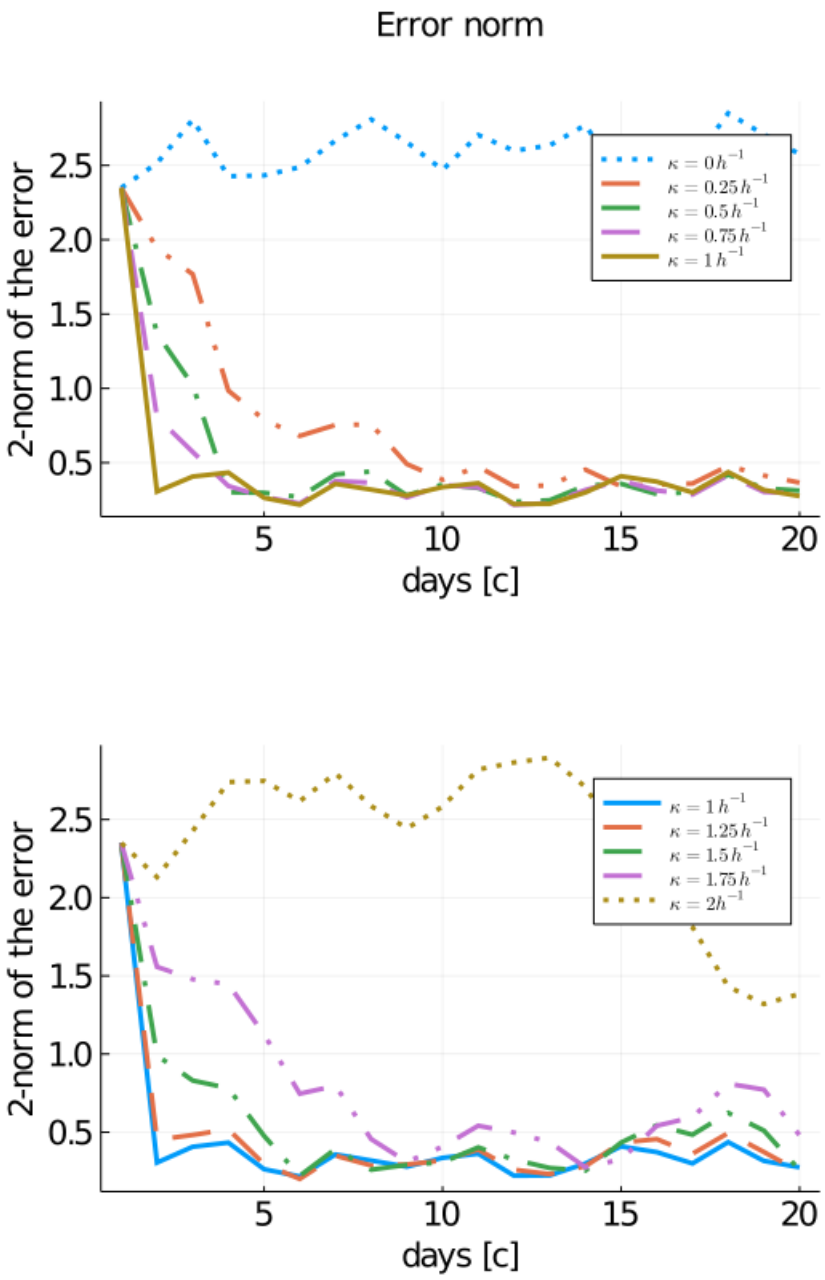

Figure 4.9: Error norm ($||\boldsymbol{e}^c||_2$) over the days for different learning gains with the nonlinear simulation

Chapter 5

Conclusions

The thesis and results presented in the previous sections will be summarized in Section 5.1. Further, a number of recommendations for future work will be provided in Section 5.2.

5.1 Summary of the thesis

We propose a framework for a hierarchical control in a prosumer-based direct current microgrid with suitable parameters. The prosumer-based network modeling is derived from the already existing producer-consumer scheme and the Kirchhoff Laws (Section 2.1.4). For the low-level control, a controller with the main focus on power sharing and voltage droop was applied (Section 3.2.1). By choosing the appropriate control coefficient, power sharing was achieved, as seen in the simulation in Section 4.1 and 4.2. For the higher-layer control the predefined iterative learning controller was applied, a controller that learns the periodic demand pattern in order to save as much low-level control energy (Section 3.2) and by choosing the appropriate learning parameter, the learning scenario can achieved different velocities, while power sharing and voltage droop is guaranteed.

5.2 Future Work

In the future, the analysis of the iterative learning control in a prosumer-based direct current microgrid by implementing a lifted system for asymptotic stability and monotonic convergence should be developed. Furthermore, viewing the learning scenario with real demand data and comparing it to the already existing

prosumer-based alternating current microgrid from [4] are tasks that could be done.

Bibliography

[1] "K.M. Fauske. Example: Control system principles. 2006." www.texample.net/tikz/examples/control-system-principles/, (02.10.2020).

[2] "Orzetto. Feedback loop with descriptions. Wikipedia," https://en.wikipedia.org/wiki/Control_theory#/media/File:Feedback_loop_with_descriptions.svg, (02.10.2020).

[3] L. Strenge, "Modeling and simulation of a droop controlled swarm type low voltage dc microgrid in a dae framework - master thesis," *A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Scientific Computing*, 2015.

[4] R. B. P. S. F. H. J. K. J. R. T. S. Lia Strenge, Xiaohan Jing, "Iterative learning control in prosumer-based microgrids with hierarchical control," *Technische Universität Berlin, Control Systems Group, Potsdam Institute for Climate Impact Research year = 2019, timestamp = Accessed: 18.05.2020.*

[5] G. N. F. H. Lia Strenge, H. Kirchhoff, "Stability of meshed dc microgrids using probabilistic analysis," *2017 IEEE Second International Conference on DC Microgrids (ICDCM)*, 2017.

[6] X. Jing, "Iterative learning control in hierarchically controlled power systems," *Master thesis - Technische Universität Berlin*, 2020.

[7] N. Hatziargyriou *et al.*, "Microgrids-large scale integration of microgeneration to low voltage grids," *Conference: CIGRE 2006, 41st Session Conference, Paris, France*, 2006.

[8] "Microgrid concepts - Christian A. Hans, Johannes Schiffer, Truong Duc Trung, Jörg Raisch," https://www.mpi-magdeburg.mpg.de/1378198/Control-Concepts-for-Microgrids, (21.05.2020).

[9] C. Schwaegerl and L. Tao, "The microgrids concept," *Wiley Online Library*, 2017.

[10] D. F. Lexuan Meng, Giancarlo Ferrari Trecate, "Review on control of dc microgrids and multiple microgrid clusters," *IEEE JOURNAL OF EMERGING AND SELECTED TOPICS IN POWER ELECTRONICS, VOL. 5, NO. 3*, 2017.

[11] X. Zhang, "Hierarchical and multilayer control structures based on mpc for large-scale systems," *POLITECNICO DI MILANO DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA DOCTORAL PROGRAMME IN INFORMATION TECHNOLOGY*, 2015.

[12] L. O. P. Vasquez *et al.*, "Optimal energy management within a microgrid: A comparative study," *energies, MDPI*, 2018.

[13] K. L. Moore, "Chapter 9 - iterative learning control," *CSM EGES 504/604A Colloquium Golden, CO*, 2006.

[14] A. Moore, Chen, "An introduction to iterative learning control theory," *CSM EGES 504/604A Colloquium Golden, CO*, 2006.

[15] "Distributed generation - Wikipedia, the free encyclopedia," https://en.wikipedia.org/wiki/Distributed_generation, (07.09.2020).

[16] "Distributed Generation of Electricity and its Environmental Impacts - US EPA United States Environmental Protection Agency," https://www.epa.gov/energy/distributed-generation-electricity-and-its-environmental-impacts, (07.09.2020).

[17] "Mikeius - translation: The current war: Nikola tesla - what the fact?! 31," https://www.youtube.com/watch?v=aTowR4acOl4, (18.09.2020).

[18] J. Justo *et al.*, "Ac-microgrids versus dc-microgrids with distributed energy resources: A review," *ELSEVIER, Renewable and Sustainable Energy Reviews, 24:387–405,*, 2009.

[19] "What is electric line losses? - Padarabinda Samal," https://www.quora.com/What-is-electric-line-losses, (20.09.2020).

[20] A. Z. Wolf-Peter Schill and F. Kunz, "Prosumage of solar electricity: Pros, cons, and the system perspective," *Deutsches Institut für Wirtschaftsforschung*, 2017.

[21] "Portmanteau - Wikipedia, the free encyclopedia," https://en.wikipedia.org/wiki/Portmanteau, (21.09.2020).

[22] “Was ist ein "Prosumer"? - Bundesminesterium für Wirtschaft und Energie,” https://www.bmwi-energiewende.de/EWD/Redaktion/Newsletter/2016/06/Meldung/direkt-erklaert.html, (13.06.2020).

[23] J. R. T. S. Johannes Schiffer, Darina Goldin, “Synchronization of droop-controlled microgrids with distributed rotational and electronic generation,” *52nd IEEE Conference on Decision and Control December 10-13, 2013. Florence, Italy*, 2013.

[24] “Directed and undirected graphs,” https://subscription.packtpub.com/book/application_development/9781788995573/8/ch08lvl1sec62/directed-and-undirected-graphs, (02.10.2020).

[25] “Open-loop-system - Electronics Tutorials,” https://www.electronics-tutorials.ws/systems/open-loop-system.html, (01.10.2020).

[26] L. Fusheng and Z. Fengquan, “Composition and classification of the microgrid- 2.4.2.3 droop control,” *Microgrid Technology and Engineering Application*, 2016.

[27] S. Anand and B. G. Fernandes, “Reduced-order model and stability analysis of low-voltage dc microgrid,” *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 60, NO. 11, NOVEMBER 2013*, 2013.

Appendix

This chapter contains the sample code of the simulation framework utilized in this thesis in Section A and the sample code for the error norm calculation in Section B.

A Simulation code

```
begin
using JLD2, FileIO, GraphIO, CSV, DataFrames
using ForwardDiff
using Distributed
using LightGraphs # create network topologies
using LinearAlgebra
using DifferentialEquations
using GraphPlot
using Random
using NetworkDynamics
using Plots
using Parameters
using ToeplitzMatrices
using DSP
using LaTeXStrings
using Distributions
using StatsBase
using Roots
using Interpolations
Random.seed!(42)
end

begin
dir = @__DIR__
```

```
N = 4 #Number of nodes
N_half = Int(N/2)
num_days = 7
l_day = 3600*24 # DemCurve.l_day
l_hour = 3600 # DemCurve.l_hour
l_minute = 60
end

struct demand_amp_var
demand
end

function (dav::demand_amp_var)(t)
index = Int(floor(t / (24*3600)))
dav.demand[index + 1,:]
end

begin
graph = random_regular_graph(iseven(3N) ? N : (N-1), 3)
end

@with_kw mutable struct LeakyIntegratorPars
K
R
L_inv
C_inv
v_ref
n_prod
n_cons
end

@with_kw mutable struct ILCPars
kappa
mismatch_yesterday
daily_background_power
current_background_power
ilc_nodes
ilc_covers
Q
end
```

```
@with_kw mutable struct incidences
inc_i
inc_v
end

@with_kw mutable struct UlMoparss
N::Int
ll::LeakyIntegratorPars
hl::ILCPars
inc::incidences
periodic_infeed
periodic_demand
fluctuating_infeed
residual_demand
incidence

function UlMoparss(N::Int,
ll::LeakyIntegratorPars,
hl:: ILCPars,
inc::incidences,
periodic_infeed,
periodic_demand,
fluctuating_infeed,
residual_demand)
new(N, ll,
hl,
inc,
periodic_infeed,
periodic_demand,
fluctuating_infeed,
residual_demand,
incidence_matrix(graph,oriented=true))
end
end

function set_parameters(N, kappa, Q)
low_layer_control = LeakyIntegratorPars(K = 1., R = 0.0532, L_inv =
v_ref = 48. ,  n_prod = num_prod, n_cons = N.-num_prod)
control_incidences = incidences(inc_i = zeros(N), inc_v = zeros(Int
higher_layer_control = ILCPars(kappa = kappa, mismatch_yesterday=ze
periodic_infeed = t -> zeros(N)
```

```
peak_demand = rand(N)
periodic_demand = t -> zeros(N)
fluctuating_infeed = t -> zeros(N)
residual_demand = t -> zeros(N)

return UlMoparss(N,low_layer_control,
higher_layer_control,
control_incidences,
periodic_infeed,
periodic_demand,
fluctuating_infeed,
residual_demand)
end

begin
current_filter = 1:Int(1.5N)
voltage_filter = Int(1.5N)+1:Int(2.5N)
energy_filter = Int(2.5N)+1:Int(3.5N)#3N+1:4N
end

function prosumerToymodel!(du, u, p, t)
n_lines = Int(1.5*p.N)

#state variables
i = u[1:n_lines]
v = u[(n_lines+1):Int(2.5*p.N)]
di = @view du[1:n_lines]
dv = @view du[(n_lines+1):Int(2.5*p.N)]
control_power_integrator = @view du[Int(2.5*p.N)+1:Int(3.5*p.N)]

periodic_power =  p.periodic_demand(t) .+ p.periodic_infeed(t) #det
fluctuating_power =   p.residual_demand(t) .+ p.fluctuating_infeed(
Pd = periodic_power + fluctuating_power # Demand

i_ILC =  p.hl.current_background_power./v # power ILC in form of a

p.inc.inc_v = p.incidence' * v
p.inc.inc_i = p.incidence * i
i_gen = p.ll.K .* (p.ll.v_ref .- v) # generated current
i_load = Pd./(v.+1) # load current
```

```
#differential equations
@. di = p.ll.L_inv .*(-(p.ll.R.*i) .+ p.inc.inc_v)
@. dv = p.ll.C_inv.*(i_ILC.+i_gen.- p.inc.inc_i .- i_load)
@. control_power_integrator = i_gen.* v          #power LI

return nothing
end

@doc """
HourlyUpdate()
Store the integrated control power in memory.
See also [`(hu::HourlyUpdate)`](@ref).
"""
struct HourlyUpdate
integrated_control_power_history
HourlyUpdate() = new([])
end

@doc """
HourlyUpdate(integrator)
PeriodicCallback function acting on the `integrator` that is called
"""
function (hu::HourlyUpdate)(integrator)
hour = mod(round(Int, integrator.t/3600.), 24) + 1
last_hour = mod(hour-2, 24) + 1
power_idx = Int(2.5*integrator.p.N)+1:Int(3.5*integrator.p.N) # pow

#power calculation y^c
integrator.p.hl.mismatch_yesterday[last_hour,:] .= 1/3600 .* integr
integrator.u[power_idx] .= 0.

integrator.p.hl.current_background_power .= integrator.p.hl.daily_b

nothing
end

function DailyUpdate_X(integrator)
#ilc
integrator.p.hl.daily_background_power =  integrator.p.hl.Q * (inte
nothing
end
```

```
demand_amp1 = demand_amp_var(repeat([80 80 80 10 10 10 40 40 40 40
demand_amp2 = demand_amp_var(repeat([10 10 10 80 80 80 40 40 40 40
demand_amp3 = demand_amp_var(repeat([60 60 60 60 10 10 10 40 40 40
demand_amp4 = demand_amp_var(repeat([30 30 30 30 10 10 10 80 80 80
demand_amp = t->vcat(demand_amp1(t), demand_amp2(t), demand_amp3(t)


periodic_demand =  t-> demand_amp(t) .* sin(t*pi/(24*3600))^2
samples = 24*4
inter = interpolate([.2 * randn(N) for i in 1:(num_days * samples +
residual_demand = t -> inter(1. + t / (24*3600) * samples)

##################  set higher_layer_control ##################

kappa = 1.
vc1 = 1:N # ilc_nodes (here: without communication)
cover1 = Dict([v => [] for v in vc1])# ilc_cover
u = [zeros(1000,1);1;zeros(1000,1)];
fc = 1/6;
a = digitalfilter(Lowpass(fc),Butterworth(2));
Q1 = filtfilt(a,u);# Markov Parameter
Q = Toeplitz(Q1[1001:1001+24-1],Q1[1001:1001+24-1]);

################### set parameters ############################
begin
param = set_parameters(N, kappa, Q)
param.periodic_demand = periodic_demand
param.residual_demand = residual_demand
param.hl.daily_background_power .= 0
param.hl.current_background_power .= 0
param.hl.mismatch_yesterday .= 0.
end
######################## solving ################################
begin
fp = [0. 0. 0. 0. 0. 0. 48. 48. 48. 48. 0. 0. 0. 0.] #initial condi
factor = 0.
ic = factor .* ones(14)
tspan = (0. , num_days * 1_day) # 1 Tag
tspan2 = (0., 10.0)
#tspan3 = (0., 200.)
```

```
ode = ODEProblem(prosumerToymodel!, fp, tspan, param,
callback=CallbackSet(PeriodicCallback(HourlyUpdate(), 1_hour),
PeriodicCallback(DailyUpdate_X, 1_day)))
end
sol = solve(ode, Rodas4())


##########################################################################
######################### ENERGIES ########################################
hourly_energy = zeros(24*num_days+1,N)

for i=1:24*num_days+1
for j = 1:N
hourly_energy[i,j] = sol((i-1)*3600)[energy_filter[j]]./3600 #the h
end
end

ILC_power = zeros(num_days+2,24,N)
for j = 1:N
ILC_power[2,:,j] = Q*(zeros(24,1) +  kappa*hourly_energy[1:24,j]) #
end
norm_energy_d = zeros(num_days,N)
for j = 1:N
norm_energy_d[1,j] = norm(hourly_energy[1:24,j])
end

for i=2:num_days
for j = 1:N
ILC_power[i+1,:,j] = Q*(ILC_power[i,:,j] +  kappa*hourly_energy[(i-
norm_energy_d[i,j] = norm(hourly_energy[(i-1)*24+1:i*24,j])
end
end

ILC_power_agg = [norm(mean(ILC_power,dims=3)[d,:]) for d in 1:num_d
ILC_power_hourly_mean = vcat(mean(ILC_power,dims=3)[:,:,1]'...)
ILC_power_hourly_mean_node1 = vcat(ILC_power[:,:,1]'...)
ILC_power_hourly = [norm(reshape(ILC_power,(num_days+2)*24,N)[h,:])
ILC_power_hourly_node1 = [norm(reshape(ILC_power,(num_days+2)*24,N)
dd = t->((-periodic_demand(t) .- residual_demand(t)))
load_amp = [first(maximum(dd(t))) for t in 1:3600*24:3600*24*num_da
```

```
norm_hourly_energy = [norm(hourly_energy[h,:]) for h in 1:24*num_da

################################### PLOTTING ########################


node = 1
p1 = plot()
ILC_power_hourly_mean_node = vcat(ILC_power[:,:,node]'...)
plot!(0:num_days*l_day, t -> -dd(t)[node], alpha=0.2, label = latex
plot!(1:3600:24*num_days*3600,hourly_energy[1:num_days*24,node], la
plot!(1:3600:num_days*24*3600,  ILC_power_hourly_mean_node[1:num_da
xtickfontsize=14,
legendfontsize=10, linewidth=3, yaxis=("normed power",font(14)),leg
title!(latexstring("j = $(node), K_D = 1"))
ylims!(-25,60)
savefig("$dir/plots/kappa_1/K_variance/K=[0.1,0.1,1,0.1]/DC_prosume


node = 2
p2 = plot()
ILC_power_hourly_mean_node = vcat(ILC_power[:,:,node]'...)
plot!(0:num_days*l_day, t -> -dd(t)[node], alpha=0.2, label = latex
plot!(1:3600:24*num_days*3600,hourly_energy[1:num_days*24,node], la
plot!(1:3600:num_days*24*3600,  ILC_power_hourly_mean_node[1:num_da
xtickfontsize=14,
legendfontsize=10, linewidth=3,legend=false, lc =:black, margin=5Pl
#savefig("$dir/plots/kappa_1/DC_prosumer_demand_seconds_node_$(node
title!(latexstring("j = $(node), K_D = 1"))
ylims!(-25,60)
savefig("$dir/plots/kappa_1/K=1/DC_prosumer_demand_seconds_$(node)_

node = 3
p3 = plot()
ILC_power_hourly_mean_node = vcat(ILC_power[:,:,node]'...)
plot!(0:num_days*l_day, t -> -dd(t)[node], alpha=0.2, label = latex
plot!(1:3600:24*num_days*3600,hourly_energy[1:num_days*24,node], la
plot!(1:3600:num_days*24*3600,  ILC_power_hourly_mean_node[1:num_da
xtickfontsize=14,
legendfontsize=10, linewidth=3, xaxis = ("days [c]",font(14)),yaxis
#savefig("$dir/plots/kappa_1/DC_prosumer_demand_seconds_node_$(node
title!(latexstring("j = $(node), K_D = 0.1"))
```

```
ylims!(-2.5,7)
savefig("$dir/plots/kappa_1/K=1/DC_prosumer_demand_seconds_$(node)_

node = 4
p4 = plot()
ILC_power_hourly_mean_node = vcat(ILC_power[:,:,node]'...)
plot!(0:num_days*1_day, t -> -dd(t)[node], alpha=0.2, label = latex
plot!(1:3600:24*num_days*3600,hourly_energy[1:num_days*24,node], la
plot!(1:3600:num_days*24*3600,  ILC_power_hourly_mean_node[1:num_da
xtickfontsize=14,
legendfontsize=10, linewidth=3, xaxis = ("days [c]",font(14)),legen
#savefig("$dir/plots/kappa_1/DC_prosumer_demand_seconds_node_$(node
title!(latexstring("j = $(node), K_D = 1"))
savefig("$dir/plots/kappa_1/K=1/DC_prosumer_demand_seconds_$(node)_

# SUM
psum = plot()
ILC_power_hourly_mean_sum = (vcat(ILC_power[:,:,1]'...) .+ vcat(ILC
plot!(0:num_days*1_day, t -> -(dd(t)[1] .+ dd(t)[2] .+ dd(t)[3] .+
plot!(1:3600:24*num_days*3600,(hourly_energy[1:num_days*24,1] + hou
plot!(1:3600:num_days*24*3600,  ILC_power_hourly_mean_sum[1:num_day
xtickfontsize=14,
legendfontsize=10, linewidth=3, xaxis = ("days [c]",font(14)),yaxis
title!(latexstring("\\sum_j, K_D = [1,1,0.1,1]"))
#title!(latexstring("K_D = 1, \\kappa = 1"))
#savefig("$dir/plots/kappa_1/kappa_1_DC_prosumer_demand_seconds_sum
#title!(latexstring("\\kappa = 2"))
#savefig("$dir/plots/kappa_1/Powers_K_[0.1_1_0.1_1]_node_sum_hetero
savefig("$dir/plots/manual_calc_variation_kappa/kappa_1/K_variance/

p_dif_nodes = plot(p1,p2,p3,p4, legend=false)
#savefig("$dir/plots/kappa_1/K_variance/K_[0.1_1_0.1_1]_seperate.pn
savefig("$dir/plots/manual_calc_variation_kappa/kappa_1/K_variance/


##########################################################################
##################### HOURLY ENERGY CURRENT AND VOLTAGE PLOTTING ##

cur = plot(sol, vars = current_filter, title = "Current per edge ",
xlabel!("Time in s")
ylabel!("Current in A")
```

```
savefig("$dir/plots/DC_prosumer_current_per_edge.png")

volt = plot(sol, vars = voltage_filter,title = "Voltage per node ")
xlabel!("Time in s")
ylabel!("Voltage in V")
savefig("$dir/plots/DC_prosumer_voltage_per_node.png")


ener = plot(sol, vars = energy_filter, title = "Energy per node", l
xlabel!("Time in s")
ylabel!("Power in W")
savefig("$dir/plots/DC_prosumer_constant_power_no_ILC_voltage_per_n


plot(hourly_energy,title = "hourly energy", label = ["Node 1" "Node
plot!(hourly_energy[:,1] .+ hourly_energy[:,2].+hourly_energy[:,3]

xlabel!("Time in h")
ylabel!("Power in W")
savefig("$dir/plots/kappa_1/energy_bilance.png")
savefig("$dir/plots/kappa_1/K_variance/K_[0.1_1_0.1_1]_hourly_energ
savefig("$dir/plots/kappa_1/K_variance/K_[0.1_1_2_5]_hourly_energy.
hourly_current = zeros(24*num_days+1,Int(1.5*N))

for i=1:24*num_days+1
for j = 1:Int(1.5*N)
hourly_current[i,j] = sol((i-1)*3600)[current_filter[j]] # weil das
auch durch 3600 geteilt wird
end
end
plot(hourly_current,title = "Current per edge ", label = ["Edge 1"
xlabel!("Time in h")
ylabel!("Current in A")
savefig("$dir/plots/DC_prosumer_no_ILC_current_per_edge.png")

hourly_voltage = zeros(24*num_days+1,N)

for i=1:24*num_days+1
for j = 1:N
hourly_voltage[i,j] = sol((i-1)*3600)[voltage_filter[j]] # weil das
auch durch 3600 geteilt wird
```

```
end
end
plot(hourly_voltage[:,1],title = "Voltage per node ", label=latexst
plot!(hourly_voltage[:,2],title = "Voltage per node ", label=latexs
plot!(hourly_voltage[:,3],title = "Voltage per node ", label=latexs
plot!(hourly_voltage[:,4],title = "Voltage per node ", label=latexs

xlabel!("Time in h")
ylabel!("Voltage in V")
savefig("$dir/plots/DC_prosumer_no_ILC_voltage_per_node.png")
```

B Error norm Calculation

```
using JLD2, FileIO, GraphIO, CSV, DataFrames
using Distributed
using Interpolations

_calc = false
_slurm = false

if _calc
using ClusterManagers
if length(ARGS) > 0
N_tasks = parse(Int, ARGS[1])
else
N_tasks = 1
end
N_worker = N_tasks
if _slurm
addprocs(SlurmManager(N_worker))
else
addprocs(N_worker)
end
println()
println(nprocs(), " processes")
println(length(workers()), " workers")
else
using Plots
end

# here comes the broadcast
```

```
# https://docs.julialang.org/en/v1/stdlib/Distributed/index.html#Di
begin
calc = $_calc # if false, only plotting
end

begin
dir = @__DIR__
include("$dir/src/system_structs.jl")
include("$dir/src/network_dynamics.jl")
end

begin
using DifferentialEquations
using Distributions
using LightGraphs
using LinearAlgebra
using Random
using StatsBase
using Statistics
using Parameters
using DSP
using ToeplitzMatrices
Random.seed!(42)
end




begin
dir = @__DIR__
N = 4 #Number of nodes
batch_size = 1
num_prod = 2 # producer nodes
nom_cons = N - num_prod
N_half = Int(N/2)
num_days = 20
l_day = 3600*24 # DemCurve.l_day
l_hour = 3600 # DemCurve.l_hour
l_minute = 60

end
```

```
begin
current_filter = 1:Int(1.5N)
voltage_filter = Int(1.5N)+1:Int(2.5N)
energy_filter = Int(2.5N)+1:Int(3.5N)#3N+1:4N
end

begin
graph = random_regular_graph(iseven(3N) ? N : (N-1), 3)

end
#_graph_lst = []
#for i in 1:1
#       push!(_graph_lst, random_regular_graph(iseven(3N) ? N : (N-
#end
#graph_lst = $_graph_lst

@with_kw mutable struct LeakyIntegratorPars
K
R
L_inv
C_inv
v_ref
n_prod
n_cons
end

@with_kw mutable struct ILCPars
kappa
mismatch_yesterday
daily_background_power
current_background_power
ilc_nodes
ilc_covers
Q
end
@with_kw mutable struct incidences
inc_i
inc_v
end

@with_kw mutable struct UlMoparss
```

```
N::Int
ll::LeakyIntegratorPars
hl::ILCPars
inc::incidences
periodic_infeed
periodic_demand
fluctuating_infeed
residual_demand
incidence

function UlMoparss(N::Int,
ll::LeakyIntegratorPars,
hl:: ILCPars,
inc::incidences,
periodic_infeed,
periodic_demand,
fluctuating_infeed,
residual_demand)
new(N, ll,
hl,
inc,
periodic_infeed,
periodic_demand,
fluctuating_infeed,
residual_demand,
incidence_matrix(graph,oriented=true))
end
end

function set_parameters(N, kappa, Q)
low_layer_control = LeakyIntegratorPars(K = [0.1, 1., 0.1, 1.] , R
v_ref = 48. ,  n_prod = num_prod, n_cons = N.-num_prod)
control_incidences = incidences(inc_i = zeros(N), inc_v = zeros(Int
higher_layer_control = ILCPars(kappa = kappa, mismatch_yesterday=ze
periodic_infeed = t -> zeros(N)
peak_demand = rand(N)
periodic_demand = t -> zeros(N)
fluctuating_infeed = t -> zeros(N)
residual_demand = t -> zeros(N)

return UlMoparss(N,low_layer_control,
```

```
higher_layer_control,
control_incidences,
periodic_infeed,
periodic_demand,
fluctuating_infeed,
residual_demand)
end
##############################################
function prosumerToymodel!(du, u, p, t)

n_lines = Int(1.5*p.N)

#state variables
i = u[1:n_lines]
v = u[(n_lines+1):Int(2.5*p.N)]

di = @view du[1:n_lines]
dv = @view du[(n_lines+1):Int(2.5*p.N)]
control_power_integrator = @view du[Int(2.5*p.N)+1:Int(3.5*p.N)]

periodic_power =  p.periodic_demand(t) .+ p.periodic_infeed(t) #det
fluctuating_power =   p.residual_demand(t) .+ p.fluctuating_infeed(
Pd = periodic_power + fluctuating_power

i_ILC =  p.hl.current_background_power./v

p.inc.inc_v = p.incidence' * v
p.inc.inc_i = p.incidence * i

i_gen = p.ll.K .* (p.ll.v_ref .- v)
i_load = Pd./(v.+1)

@. di = p.ll.L_inv .*(-(p.ll.R.*i) .+ p.inc.inc_v)
@. dv = p.ll.C_inv.*(i_ILC.+i_gen.- p.inc.inc_i .- i_load)

@. control_power_integrator =  i_gen.* v        #Power LI


return nothing
end
@doc """
```

```
HourlyUpdate()
Store the integrated control power in memory.
See also [`(hu::HourlyUpdate)`](@ref).
"""
struct HourlyUpdate
integrated_control_power_history
HourlyUpdate() = new([])
end



@doc """
HourlyUpdate(integrator)
PeriodicCallback function acting on the `integrator` that is called
"""
function (hu::HourlyUpdate)(integrator)
hour = mod(round(Int, integrator.t/3600.), 24) + 1
last_hour = mod(hour-2, 24) + 1
power_idx = Int(2.5*integrator.p.N)+1:Int(3.5*integrator.p.N) # pow

#power calculation y^c
integrator.p.hl.mismatch_yesterday[last_hour,:] .= 1/3600 .* integr
integrator.u[power_idx] .= 0.

integrator.p.hl.current_background_power .= integrator.p.hl.daily_b

nothing
end



function DailyUpdate_X(integrator)
#ilc
integrator.p.hl.daily_background_power =  integrator.p.hl.Q * (inte
nothing
end
# Monte Carlo functions

get_run(i, batch_size) = mod(i, batch_size)==0 ? batch_size : mod(i
get_batch(i, batch_size) = 1 + (i - 1) % batch_size
```

```
function prob_func_ic(prob, i, repeat, batch_size, kappa_lst, num_d
println("sim ", i)
run = get_run(i, batch_size)
batch = get_batch(i, batch_size)

prob.p.hl.daily_background_power .= 0.
prob.p.hl.current_background_power .= 0.
prob.p.hl.mismatch_yesterday .= 0.

prob.p.hl.kappa = kappa_lst[batch]

#prob.p.coupling = 800. .* diagm(0=>ones(ne(prob.p.graph)))

hourly_update = HourlyUpdate()

ODEProblem(prosumerToymodel!, prob.u0, prob.tspan, prob.p,
callback=CallbackSet(PeriodicCallback(hourly_update, 3600),
PeriodicCallback(DailyUpdate_X, 3600*24)))
end



function observer_ic(sol, i, energy_filter, num_days,N) # what shou
# sol.prob.callback.discrete_callbacks[1].affect!.f.integrated_cont

hourly_energy = zeros(24*num_days,N)
for i=1:24*num_days
for j = 1:N
hourly_energy[i,j] = sol(i*3600)[energy_filter[j]]./3600
end
end

ILC_power = zeros(num_days,24,N)
norm_energy_d = zeros(num_days,N)
for j = 1:N
norm_energy_d[1,j] = norm(hourly_energy[1:24,j])
end

for i=2:num_days
for j = 1:N
```

```
ILC_power[i,:,j] = sol.prob.p.hl.Q*(ILC_power[i-1,:,j] +
sol.prob.p.hl.kappa*hourly_energy[(i-1)*24+1:i*24,j])
end
for j = 1:N
norm_energy_d[i,j] = norm(hourly_energy[(i-1)*24+1:i*24,j])
end
end

((sol.prob.p.hl.kappa, hourly_energy, norm_energy_d), false)
end


##############################################
# this should only run on one process
##############################################

struct demand_amp_var
demand
end


function (dav::demand_amp_var)(t)
index = Int(floor(t / (24*3600)))
dav.demand[index + 1,:]
end



demand_amp1 = demand_amp_var(60 .+ rand(num_days+1,Int(N/4)).* 40.)
demand_amp2 = demand_amp_var(70 .+ rand(num_days+1,Int(N/4)).* 30.)
demand_amp3 = demand_amp_var(80 .+ rand(num_days+1,Int(N/4)).* 20.)
demand_amp4 = demand_amp_var(90 .+ rand(num_days+1,Int(N/4)).* 10.)
demand_amp = t->vcat(demand_amp1(t), demand_amp2(t),demand_amp3(t),



periodic_demand =  t-> demand_amp(t)./100 .* sin(t*pi/(24*3600))^2
samples = 24*4
inter = interpolate([.2 * randn(N) for i in 1:(num_days * samples +
residual_demand = t -> inter(1. + t / (24*3600) * samples) # 1. + i
```

```
##########################################
#             SIM                        #
##########################################

##################  set higher_layer_control ##################

kappa = 1.
vc1 = 1:N # ilc_nodes (here: without communication)
cover1 = Dict([v => [] for v in vc1])# ilc_cover
u = [zeros(1000,1);1;zeros(1000,1)];
fc = 1/6;
a = digitalfilter(Lowpass(fc),Butterworth(2));
Q1 = filtfilt(a,u);# Markov Parameter
Q = Toeplitz(Q1[1001:1001+24-1],Q1[1001:1001+24-1]);


# kappa_lst = (0:0.01:2) ./ l_hour
begin
kappa_lst = (0:.25:2)
kappa = kappa_lst[1]
num_monte = batch_size*length(kappa_lst)
end

################### set parameters ############################
begin
param = set_parameters(N, kappa, Q)
param.periodic_demand = periodic_demand
param.residual_demand = residual_demand
param.hl.daily_background_power .= 0
param.hl.current_background_power .= 0
param.hl.mismatch_yesterday .= 0.
end

begin
fp = [0. 0. 0. 0. 0. 0. 48. 48. 48. 48. 0. 0. 0. 0.] #initial condi
factor = 0.
ic = factor .* ones(14)
tspan = (0. , num_days * l_day) # 1 Tag
tspan2 = (0., 10.0)
#tspan3 = (0., 200.)
ode = ODEProblem(prosumerToymodel!, fp, tspan, param,
```

```
callback=CallbackSet(PeriodicCallback(HourlyUpdate(), 1_hour),
PeriodicCallback(DailyUpdate_X, 1_day)))
end
sol = solve(ode, Rodas4())

#################################### mein code kommt bis hier #####
monte_prob = EnsembleProblem(
ode,
output_func = (sol, i) -> observer_ic(sol, i, energy_filter,num_days
prob_func = (prob,i,repeat) -> prob_func_ic(prob,i,repeat, batch_siz
#       reduction = (u, data, I) -> experiments.reduction_ic(u, data
u_init = [])

res = solve(monte_prob,
Rodas4P(),
trajectories=num_monte,
batch_size=batch_size)

kappa = [p[1] for p in res.u]
hourly_energy = [p[2] for p in res.u]
norm_energy_d = [p[3] for p in res.u]
######################################## ab hier unveraendert #######
using LaTeXStrings
plot(mean(norm_energy_d[1],dims=2),legend=:topright, label = L"\kapp
xtickfontsize=14, linestyle=:dot, margin=8Plots.mm,
legendfontsize=8, linewidth=3,xaxis=("days [c]",font(14)), yaxis =
ylims=(0,1e6)
plot!(mean(norm_energy_d[2],dims=2), label= L"\kappa = 0.25\, h^{-1}
plot!(mean(norm_energy_d[3],dims=2), label= L"\kappa = 0.5\, h^{-1}
plot!(mean(norm_energy_d[4],dims=2),label=  L"\kappa = 0.75\, h^{-1}
plot!(mean(norm_energy_d[5],dims=2), label= L"\kappa = 1\, h^{-1}",
title!("Error norm")
savefig("$dir/plots/variation_kappa_leq_1_hetero.png")

using LaTeXStrings
plot(mean(norm_energy_d[5],dims=2),legend=:topright, label = L"\kapp
xtickfontsize=14, linestyle =:solid, margin=8Plots.mm,left_margin=1
legendfontsize=8, linewidth=3,xaxis=("days [c]",font(14)), yaxis=("
# ylims=(0,1e6)
plot!(mean(norm_energy_d[6],dims=2),label=  L"\kappa = 1.25\, h^{-1}
plot!(mean(norm_energy_d[7],dims=2),label=  L"\kappa = 1.5\, h^{-1}
```

```
plot!(mean(norm_energy_d[8],dims=2),label=  L"\kappa = 1.75\, h^{-1}
plot!(mean(norm_energy_d[9],dims=2), label= L"\kappa = 2 h^{-1}", l
#title!("Error norm")
savefig("$dir/plots/variation_kappa_geq_1_hetero.png")
```