ARCHITECTURES AND ALGORITHMS FOR DISTRIBUTED GENERATION CONTROL OF MICROGRIDS

BY

STANTON T. CADY

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Doctoral Committee:

Associate Professor Alejandro D. Domínguez-García, Chair
Adjunct Professor Christoforos N. Hadjicostis
Professor Peter W. Sauer
Professor Nitin H. Vaidya

ABSTRACT

Microgrids and the control challenges they pose have recently received significant attention in a wide array of research communities. While the potential to increase efficiency, reliability, and adaptability of the utility grid is a primary motivation for their development, microgrids can also be used to meet the growing electric power demands in numerous applications. Compared with large power systems, microgrids may rely on inertia-less generators such as photovoltaic arrays that are interfaced through an inverter. Although the lack of inertia and other microgrid characteristics pose control challenges, microgrids are amenable to new control paradigms, e.g., those that rely on distributed computations rather than a centralized processor.

We address the problem of distributed generation control in islanded ac microgrids with and without inertia. In the case of microgrids comprising heterogeneous generators, some of which have inertia, we propose a control architecture for frequency regulation and optimal dispatch designed to take advantage of microgrid-specific properties. For microgrids with no inertia, we propose a control architecture that is designed to drive the average frequency error to zero while ensuring that the frequency at every bus is equal and that the operating point that results is stable. In both cases, we also propose an implementation of each control architecture that relies on distributed algorithms that eliminate the need for a centralized processor with global information. For the architecture we propose for microgrids with inertia, we provide analytical and experimental results that verify the effectiveness of the proposed architecture, and illustrate the performance of the distributed algorithms on which it relies under a variety of scenarios. We verify the proposed control architecture for inertia-less microgrids by analytically showing that the resulting closed-loop system is stable; we also illustrate the features of the architecture using numerical simulations of three test cases applied to six- and 37-bus networks.

To my parents, Beth and Steve, and my sister, Stephanie.

ACKNOWLEDGMENTS

Throughout my time as a student at the University of Illinois at Urbana-Champaign—Fall 2005 to Spring 2016, over 10 years!—I was incredibly fortunate to receive tremendous support, direction, and inspiration from countless sources. Although the people that have helped me along the way are too numerous to list exhaustively, I am beyond grateful to each of them; the following is a subset of these people that I'd like to address specifically.

I would like to begin by expressing my sincere gratitude to my immediate family—my mom, dad, and sister—for everything they have done to enable me to achieve all that I have. In particular, I'd like to thank my parents for their enduring support, both financially and emotionally, which has allowed me to follow my dreams, even when those dreams haven't been particularly well-specified. I'd also like to thank my sister for supporting me in my pursuits—she has been invaluable as a responder to random, and randomly timed, questions, a person to whom I can vent frustrations, and, at times, a source of friendly competition (see, e.g., our ACT scores).

With equal sincerity, I'd like to thank my advisor, Professor Alejandro Domínguez-García, for his guidance throughout my time as a graduate student. I distinctly remember my first encounter with Professor Domínguez-García in the Fall semester of 2008, his first full semester at UIUC, when he gave a substitute lecture for the first course I took in the area of power and energy systems. At the time, I was thrown off by his accent, which was in stark contrast to the easy-to-understand vernacular of the professor for whom he was substituting, but, in working with him, I learned to understand him without issue, and in doing so, learned that the depth and breadth of his technical understanding are truly impressive. His expertise, coupled with his motivation and enthusiasm, which are seemingly inexhaustible, served as continuous inspiration in my own academic pursuits. Beyond his technical guidance, Professor Domínguez-García was a role model and a friend, and I

know that the impact he has had on my life will continue to shape me well into the future.

I would also like to thank the remainder of my doctoral committee: Professors Peter Sauer, Christoforos Hadjicostis, and Nitin Vaidya. Professor Sauer was influential throughout almost all of my academic career, including the aforementioned first course in power and energy systems; his kindness and ability to teach complex topics in a simple way were a big part of what encouraged me to continue with graduate school. I am very appreciative of the opportunity to work with Professor Hadjicostis throughout graduate school; his knowledge and intuition were incredibly useful in development of what became the topic of my thesis. Professor Vaidya's input on topics outside of my primary field of study proved to be invaluable in my research. In addition to my doctoral committee, I would also like to thank the following professors for their impact on my academic career: Philip Krein, Robert Pilawa-Podgurski, and Patrick Chapman.

During my time as a graduate student, I had the privilege of participating in several industrial internships. These opportunities provided me with excellent insight into some of the challenges faced outside of academia, and taught me skills that have proven to be invaluable. I would like to thank all of the mentors and people with whom I worked at each of these opportunities including: Mark Imbertson, Neilus O'Sullivan, and Udaya Ammu at Google; Mark Malhotra, Kevin Peterson, Timo Bruck, and Daniel Altin at Nest Labs; and Bruce Tsuchida, Pablo Ruiz, and Kai Van Horn at the Brattle Group.

My life as a graduate student was also shaped by the many friends and colleagues with whom I was lucky to build connections. Special thanks go to Sairaj Dhople, Charles Murray, and Jon Ehlmann with whom I had the opportunity to work on the 2009 Solar Decathlon home, and who were influential in my early academic pursuits. I've been fortunate to maintain connections with each of these three as we have continued on our own paths and hope to continue to do the same. I owe a great deal to my other academic siblings for their support over the years: Yu "Christine" Chen, Xichen Jiang, Jiangmeng Zhang, Brett Robbins, Justin Hughes, Kai Van Horn, and, for his numerous insights later in my doctoral work, Madi Zholbaryssov. Heartfelt thanks to my friend, roommate, and colleague, Matt Magill, for putting up with me without much complaint. Finally, equally sincere thanks go to the remaining friends I've made along the way: Enver Candan, Shamina Hossain-

McKenzie, Giang Chau Ngo, Raj Bhana, Joseph Silvers, Tim Duly, Trevor Hutchins, Gerald Nilles, Katherine Kim, and Dimitra Apostolopoulou.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

We begin this chapter by providing an overview of the work presented in this dissertation. Next, we compare the work presented herein to some existing works found in the literature. Finally, we outline the specific contributions of this dissertation and describe the organization of the remainder of the document.

1.1 Motivation and Overview of Problem

Recent technological advancements, particularly in communications, computation, and power electronics, have allowed microgrids to progress from a concept (see, e.g., [1, 2]) to a reality (see, e.g., [3, 4, 5]). Enabled by these progressions and motivated by initiatives such as the US-DoE SmartGrid [6], which seek to increase efficiency, reliability, and adaptability, microgrids have been proposed as a solution for providing electric power in a large array of applications. While the prototypical microgrid is envisioned as a land-based collection of loads and generators that have the capability to island from the utility grid, e.g., a neighborhood comprising homes with rooftop-mounted photovoltaic (PV) arrays, they have also been proposed for applications such as more electric aircraft [7, 8], ships with electric propulsion [9, 10], and telecommunications installations [11].

Although the nascency of microgrids precludes a strict definition, any collection of interconnected generators and loads that is capable of islanded operation is generally considered to be one. It follows, then, that there are no formal restrictions on the types of generators that may be present in a microgrid; however, like the prototypical example of a neighborhood comprising homes with rooftop-mounted photovoltaic (PV) arrays [2], microgrids may consist entirely of generators that are interfaced through power electronics.

Consequently, without the sizable spinning mass inherent in large-scale generators, this class of microgrid has little to no effective inertia. Moreover, irrespective of the presence of inertia, the power demands of loads in a microgrid can be large relative to the output capabilities of each generator.

Even though the properties mentioned above may not characterize all microgrids, e.g., dc microgrids (see, e.g., [12] and the references therein), in this dissertation, we restrict consideration to two classes of islanded ac microgrids for which they do: (i) those with heterogeneous generators, including those with generators with inertia and generators interfaced through power electronics, and (ii) those consisting entirely of generators interfaced through inverters. While the properties mentioned above, among others, complicate the problem of generation control for both classes, microgrids are unencumbered by the requirements and well-established concepts of their larger counterparts (see, e.g., [13]), making them amenable to new control paradigms. In particular, whereas generation control architectures for large power systems typically depend on a centrally located decision-maker, numerous distributed architectures have been proposed for microgrids (see, e.g., [14, 15]). Through a combination of computations performed by processors located at each bus and information exchanged between neighboring processors, these distributed architectures, which can meet the same objectives as their centralized counterparts, obviate the need for an entity with complete information about the number, type, or capabilities of the generation units in the system. Additionally, by eliminating the need for a centralized processor and a communication network connecting it to each generator, these distributed approaches can achieve higher system-level efficiency, reliability, and adaptability.

With respect to the first class of microgrids, i.e., those with inertia, their generation control objectives are similar to those in large ac power systems. More specifically, the generators therein must be controlled such that: (i) generation and demand are constantly balanced, (ii) frequency is regulated, and (iii) costs of generation are minimized; importantly, all of these goals must be met without violating any power output limits of the generation units. In a large power system, each of these objectives is achieved independently using a three-layered generation control architecture. The bottom layer operates nearly instantaneously and is responsible for balancing generation and demand. The primary function of the middle layer is to regulate frequency, i.e., ensure the system operates as close as possible to the nominal

frequency value, e.g., 60 Hz. In addition to regulating frequency, the middle layer is responsible for maintaining proper interchange power between control areas; however, in the context of microgrids, there is no notion of control areas. The function of the top layer is to optimally dispatch the generating units, i.e., ensure that the units are generating power in such a way that total operation costs are minimized [13]. Compared with the first distributed control architecture we propose in this dissertation, in a typical large power system, among the three previously mentioned control layers, only the bottom one operates using completely local information; the top two layers require a centralized decision maker to coordinate and control the generators.

In the context of the second class of microgrids, i.e., those with no inertia, a control architecture for frequency regulation must be designed to ensure that stable operation results and that the frequency at every bus in the system is equal to the desired reference value. While numerous control schemes for microgrids have been proposed in the literature, designing a frequency regulation architecture that achieves the aforementioned properties without the presence of inertia has received limited attention thus far. To the author's knowledge, no previous work has presented a scheme for frequency control in inertia-less ac microgrids for which stability and system-wide operation at the same frequency are guaranteed.

Next, we discuss some of the work that has been presented in the literature on control for microgrids including frequency regulation in inertia-less microgrids.

1.2 Literature Review

Although the purpose of the work presented herein is to develop distributed control architectures for frequency regulation and optimal dispatch in ac microgrids, a significant body of work exists in the literature on control architectures for microgrids in general, and distributed control architectures for large power systems. In the discussion that follows, we give an overview of some relevant recent works.

As mentioned above, the characteristics of microgrids pose a unique set of control challenges, some of which have been outlined in [2, 3]. Whereas the

primary focus of the control architectures presented herein is on distributed approaches, centralized approaches to microgrid control have been proposed in [16, 17, 18, 19, 20, 21]. In [14, 22], the authors propose distributed methods for primary control; comparatively, our focus is on the top two control layers, i.e., frequency regulation and optimal dispatch. Some work has been done to address the frequency regulation problem, with centralized approaches proposed in [18, 19], and, while [4, 15, 23, 24, 25, 26, 27] propose distributed solutions, all but [23] rely on undirected communication links. Although the approach in [23] also applies in directed communication networks, it is equivalent to a message passing protocol and requires an external signal to provide a control reference, which is not necessary in our work. Solving the optimal dispatch problem for microgrids has been addressed in a centralized manner in [16], with distributed solutions that rely on undirected information exchange proposed in [28, 29, 30, 31]; among these, only [29] accounts for limits on generator output. While [32] proposes a distributed model predictive control approach for the middle generation control layer, the focus is on maintaining the proper power interchange between areas in bulk power transmission networks. Finally, although not directly related to the work presented herein, it should also be noted that there is a related body of literature focusing on distributed approaches to reactive power control for voltage regulation (see, e.g., [33, 34, 35]).

Despite all the work referenced above on control architectures for ac microgrids in general, very little work has been done thus far on control for microgrids with no inertia. While [17, 18, 19] propose control architectures for frequency regulation in this class of microgrids, none provides analytical results guaranteeing both stable operation and system-wide operation at the desired frequency. The authors in [4, 25, 26] proposed control schemes for frequency regulation that they analytically showed to be stable; however, [25] relies on a linearized network model, [26] can only guarantee stability for certain initial conditions, and [4] does not guarantee that every bus oscillates at the same frequency. Finally, the authors in [15, 27] proposed a control architecture that they analytically showed to result in system-wide operation at the desired frequency, but could only guarantee local stability.

1.3 Contributions

This dissertation summarizes the main contributions of the work originally presented in [5, 36, 37, 38, 39, 40], which we organize into three parts. In Part I we introduce models to represent a microgrid and outline some distributed algorithms that will serve as primitives for implementing the distributed control architectures proposed in Parts II and III. In Part II, we propose a distributed control architecture for frequency regulation and optimal dispatch that is designed for islanded ac microgrids with heterogeneous generators, including those with inertia. In Part III, we propose a distributed control architecture for frequency regulation that is designed for islanded ac microgrids with no inertia, i.e., those comprising inverter-interfaced generators. In the discussion that follows, we provide an overview of the specific contributions in each part.

1.3.1 Contributions in Part I

We begin this part by introducing physical and cyber layer models to represent a microgrid and a communication network formed by a distributed architecture designed to control it. We then outline three distributed algorithms, two of which are established consensus-type algorithms; we propose a modification to one of these consensus algorithms that enables the computation of an approximation to the true asymptotic value to which it converges in finite time. The third algorithm enables the computation of generator output values that collectively meet the total load power demand without violating generation or line flow limits; although this algorithm is similar to the one presented in [37], we provide a proof of convergence for it in Chapter 3.

1.3.2 Contributions in Part II

The main contribution of the work in this part is the development and implementation of a distributed control architecture for islanded ac microgrids with inertia that replicates the functionality of the aforementioned top two control layers that are typical in large power systems, i.e., frequency regulation and optimal dispatch. Specifically, ac systems comprising synchronous generators and inverter-interfaced power supplies, collectively referred to as

distributed generation resources (DGRs), are considered, and we assume that each DGR is equipped with a local processor and a transceiver through which information can be exchanged, possibly unidirectionally, with neighboring DGRs. By relying on this directed communication network and on simple computations executed by the local processors, we provide two algorithms that allow the DGRs to determine their generation set-points in order to (i) regulate the system frequency and (ii) minimize total operational costs; like the control architectures from which ours is derived, both of these algorithms account for power output limits of the DGRs. Beyond introducing distributed alternatives to the top two generation control layers, a significant difference between our distributed control architecture and centralized counterparts used in bulk power systems is that ours is event triggered, i.e., generation set-points can be updated at non-fixed time instants, which eliminates unnecessary control efforts during periods in which the power demand remains relatively constant.

To demonstrate the effectiveness of the distributed generation control architecture presented in Part II, we tested it on a laboratory-grade microgrid. The electrical network of this microgrid comprises several small synchronous generators interconnected with resistive loads (inverter-interfaced power supplies are omitted due to lack of availability), whereas the cyber network for communication and control comprises Arduino microcontrollers outfitted with wireless transceivers. Using this microgrid, we experimentally verified the performance of the proposed control architecture under a variety of scenarios. Specifically, we provide results that illustrate the operation of the distributed frequency regulation and optimal dispatch functions following both an increase and a decrease in load. Additionally, we show how a DGR acting as spinning reserve can use the distributed frequency regulation function to independently determine if the collective power output limit of the online units is insufficient to balance the load, and act accordingly.

1.3.3 Contributions in Part III

In this part, we propose a distributed frequency regulation architecture designed for islanded inertia-less ac microgrids. As in [37], our proposed control architecture is designed to take advantage of the results in [41] by tracking

generator set-points that, for some initial load demand, are known to result in an operating point at which the phase angle difference between every bus is strictly smaller than $\frac{\pi}{2}$, i.e., so-called phase-cohesive operation results. (We formally define phase-cohesive operation and the properties that characterize it in Chapter 9.) Following one or more small perturbations to the power demanded by the loads, the architecture iteratively adjusts the generator set-points to drive the average frequency to some reference value while also ensuring that the operating point that results is phase cohesive. By regulating the average frequency around an operating point that is known to be phase cohesive, our controller ensures small-signal stability of the closed-loop system while also guaranteeing that the frequency at every bus is the same. To handle larger load perturbations, we also provide a method for triggering the recomputation of the generator set-points to be tracked based upon an estimate for the amount by which the system has deviated from the original phase-cohesive operating point. Using the distributed algorithms presented in Chapter 3, we also outline a distributed implementation of our proposed control architecture. These algorithms enable the acquisition of global information with which processors located at each bus can make decisions to collectively achieve the system-level objectives of our proposed control architecture. Finally, we provide analytical criteria for choosing gains that result in closed-loop stability and demonstrate the operation of our proposed architecture and its distributed implementation using numerical simulations of three case studies.

1.4 Document Organization

The remainder of this dissertation is organized as follows. In Chapter 2, we introduce a model to represent the physical layer of a microgrid, including dynamical models to represent the behavior of the generators and loads that comprise it. We further introduce a cyber layer model to represent the communication links between local bus processors that will be used to implement the distributed control architectures we will present in Parts I and II. In Chapter 3, we outline three distributed algorithms that will serve as primitives for distributively implementing the control architectures we will present in Parts I and II. Chapters 4 – 12 constitute Parts II and III, the or-

ganization of which we discuss next, and we provide some concluding remarks and ideas for future work in Chapter 13.

In Part II we propose a distributed generation control architecture for ac microgrids with heterogeneous generation resources, i.e., those with inertia and those interfaced through power electronics. This part begins with Chapter 4 in which we provide an overview of the problem to be solved and introduce some preliminary notions. In Chapter 5, we provide an overview of the desired control functions and outline specific control objectives; we also provide guidelines for choosing controller gains to ensure closed-loop stability. In Chapter 6, we describe the two algorithms that distributively implement the desired control functions. Chapter 7 describes a laboratory-grade microgrid that we developed for testing the performance of the proposed architecture. Experimental results obtained using the aforementioned laboratory-grade microgrid are presented in Chapter 8.

In Part III we propose a distributed architecture for frequency regulation in ac microgrids consisting entirely of generators interfaced through inverters, i.e., those with no inertia. We begin in Chapter 9 by providing an overview of the problem to be solved, introducing some assumptions to reduce the microgrid model, and providing a formal definition of phase cohesiveness. In Chapter 10, we introduce the control scheme for our proposed frequency regulation architecture for inertia-less microgrids and outline criteria for ensuring closed-loop stability. We outline a distributed implementation of this control architecture in Chapter 11. In order to validate our control architecture and its distributed implementation, we provide numerical simulation results for three test cases in Chapter 12.

Part I

Models and Primitives for Distributed Control

CHAPTER 2

MODELS

In this chapter, we outline a model to represent the physical network of a microgrid. The model uses notions from graph theory to represent the interconnections between buses in the system, the notation for which we outline first. We then introduce dynamical models to represent the behavior of generators and loads in a microgrid. Next, we outline some simplifying assumptions that reduce the model representing the dynamic behavior of the generators to a form that can be used to model the dynamics of inverter-interfaced generators. We also establish a second graph-theoretic model to describe the cyber layer which comprises processors located at each bus and communication links interconnecting them and is used to implement the distributed control architectures introduced later in the dissertation.

2.1 Physical Layer Model

In this section we outline a model that will be used to represent a microgrid throughout the remainder of this document. We begin by introducing an undirected graph that is used to represent electrical connections between buses. Then, we outline a model to represent the generator dynamics—the so-called structure preserving model—and a first-order dynamical model to represent the load behavior. Finally, we provide some simplifying assumptions that reduce the structure preserving model to one that is sufficient for representing the dynamics of an inverter-interfaced generator.

2.1.1 Structure Preserving Generator Model and Dynamic Load Model

For an n-bus microgrid, let $\mathcal{G}_p = (\mathcal{V}_p, \mathcal{E}_p)$ be an undirected simple graph representing the interconnections between buses. The vertex—or bus—set, $\mathcal{V}_p$, is defined to be $\mathcal{V}_p := \{1, 2, \ldots, n\} = \mathcal{V}_g \cup \mathcal{V}_\ell$, where $\mathcal{V}_g$ and $\mathcal{V}_\ell$ denote generator and load bus sets, respectively. Without loss of generality, we partition the bus set such that $\mathcal{V}_g := \{1, 2, \ldots, m\}$, $\mathcal{V}_\ell := \{m+1, m+2, \ldots, n\}$, and $\mathcal{V}_g \cap \mathcal{V}_\ell = \emptyset$, i.e., each bus has only a generator or load attached, but not both. The edge—or branch—set, $\mathcal{E}_p$, is defined to be $\mathcal{E}_p \subseteq \mathcal{V}_p \times \mathcal{V}_p$, where the edge $(i, j) \in \mathcal{E}_p$ if buses i and j are connected electrically. We denote the set of buses to which each bus i is connected by $\mathcal{N}_p(i) := \{j \in \mathcal{V}_p : (i, j) \in \mathcal{E}_p\}$, and denote the number of such buses by $\delta_p(i) = |\mathcal{N}_p(i)|$. Finally, we assume that no islands exist in the microgrid such that the graph $\mathcal{G}_p$ consists of a single connected component.

To maintain a general model of the microgrid, we represent the dynamics of each generator using the structure-preserving model with constant complex voltage behind reactance (see, e.g., [42, Section 7.9.2]), augmented to include governor dynamics. While this model is commonly used to describe the dynamics of synchronous generation units, in subsequent developments, we will introduce some simplifying assumptions to account for generators that, for the time scales under consideration in this work, have less complex dynamics, e.g., photovoltaic arrays that are interfaced through an inverter. Additionally, we assume that the loads exhibit non-constant dynamic behavior such that they can be represented using a first-order dynamical model.

At time $t \geq 0$, let $V_i(t)$ and $\theta_i(t)$ denote the voltage magnitude and angle of bus $i \in \mathcal{V}_p$, respectively. Furthermore, for bus $i \in \mathcal{V}_g$, let $\delta_i(t)$ denote the angle of its voltage behind reactance (or "internal voltage") as measured with respect to a synchronous reference rotating at the nominal system electrical frequency, ω_0; $\omega_i(t)$ denote its rotor electrical angular speed; $P_i^m(t)$ denote its prime mover mechanical power; and $u_i(t)$ denote its generation set-point, with lower and upper limits denoted by $\underline{u}_i$ and $\overline{u}_i$, respectively. Then, the

dynamics of generator $i \in \mathcal{V}_g$ can be represented by

$$\begin{aligned}
\frac{d\delta_i}{dt} =& \omega_i - \omega_0, && (2.1)\\
J_i\frac{d\omega_i}{dt} =& - D_i(\omega_i - \omega_0) + P_i^m \\
& - V_i \sum_{j=1}^{n} V_j \left[G_{(i,j)} \cos(\theta_i - \theta_j) + B_{(i,j)} \sin(\theta_i - \theta_j)\right], && (2.2)\\
\tau_i\frac{dP_i^m}{dt} =& - P_i^m - \frac{1}{R_i\omega_0}(\omega_i - \omega_0) + u_i, && (2.3)
\end{aligned}$$

where D_i [s/rad] is the generator damping coefficient, J_i [s^2/rad] is a scaled inertia constant, τ_i [s] is the generator governor time constant, R_i [pu] is the speed-droop characteristic slope of the generator, and $G_{(i,j)}$ and $B_{(i,j)}$, $(i,j) \in \mathcal{E}_p$, are conductance and susceptance between two electrically connected buses, respectively. Similarly, the dynamics of load $i \in \mathcal{V}_\ell$ can be represented by

$$H_i\frac{d\theta_i}{dt} = -\ell_i - V_i \sum_{j=1}^{n} V_j \left[G_{(i,j)} \cos(\theta_i - \theta_j) + B_{(i,j)} \sin(\theta_i - \theta_j)\right], \qquad (2.4)$$

where H_i [s/rad] is a time constant associated with the dynamics of load i and ℓ_i [pu] is the real power demanded by load i which is assumed to be constant.

It should be noted that we have omitted the reactive power models for synchronous generators and loads as they are unnecessary for the analysis presented herein. Also note that while not explicitly described, the model in (2.4) (and its analogous reactive power model) also includes constant-impedance loads as their effect can be captured by adding the appropriate terms to the diagonal entries of the network admittance matrix; similarly, constant-current loads can also be incorporated into the demand bus model (see, e.g., [42, Section 6.3] for the procedure).

2.1.2 Dynamical Generator Model Reduction

In the discussion that follows, we state a series of assumptions that collectively reduce the generator model in (2.1) – (2.3) to a form that can represent the dynamics of an inverter-interfaced generator. To account for microgrids

consisting of traditional generators and generators interfaced through inverters and for those exclusively comprising inverter-interfaced generators, we partition the generator bus set as $\mathcal{V}_g \subseteq \mathcal{V}_s \cup \mathcal{V}_i$, where $\mathcal{V}_s$ is the set of generators with inertia, and $\mathcal{V}_i$ is the set of generators with no inertia.

The first assumption we make is that the dynamics of the governor that regulates the power output of an inverter-interfaced generator is many orders of magnitude faster than the dynamical behavior of the other components in the system (see, e.g., [42, Appendix A] for a formalization of procedure for reducing dynamics with significant temporal separation). More specifically, we assume that the time constant in (2.3) is very small, i.e., for $i \in \mathcal{V}_i$, $\tau_i \to 0$. Thus, we take the left-hand side of (2.3) to be zero such that we can write the power output of the prime mover in terms of the generator set-point and the electrical angular speed, i.e.,

$$P_i^m = u_i - \frac{1}{R_i \omega_0}(\omega_i - \omega_0). \tag{2.5}$$

Then, by plugging (2.5) into (2.2), the generator model reduces to

$$\frac{d\delta_i}{dt} = \omega_i - \omega_0, \tag{2.6}$$

$$\begin{aligned} J_i \frac{d\omega_i}{dt} =& - \left(D_i + \frac{1}{R_i \omega_0} \right) (\omega_i - \omega_0) + u_i \\ & - V_i \sum_{j=1}^{n} V_j \left[G_{ij} \cos(\theta_i - \theta_j) + B_{ij} \sin(\theta_i - \theta_j) \right]. \end{aligned} \tag{2.7}$$

Finally, we assume that, unlike traditional generation units such as synchronous generators, those interfaced through an inverter have little to no inertia. More formally, we assume that $J_i \to 0$ for $i \in \mathcal{V}_i$; thus, the left-hand side of (2.7) is equal to zero and can be simplified as

$$\omega_i - \omega_0 = \frac{1}{D_i} \left(u_i - V_i \sum_{j=1}^{n} V_j \left[G_{ij} \cos(\theta_i - \theta_j) + B_{ij} \sin(\theta_i - \theta_j) \right] \right), \tag{2.8}$$

where $H_i := D_i + \frac{1}{R_i \omega_0}$. By plugging (2.8) into (2.1), we have

$$H_i \frac{d\delta_i}{dt} = u_i - V_i \sum_{j=1}^{n} V_j \left[G_{ij} \cos(\theta_i - \theta_j) + B_{ij} \sin(\theta_i - \theta_j) \right]. \tag{2.9}$$

2.2 Cyber Layer Model

In order to realize the distributed control architectures for microgrids that we propose later in Parts II and III, we assume that each generator and load in the system is outfitted with a local processor that is capable of performing simple computations such as addition and multiplication. We further assume that each local processor can communicate with the processors located at other nearby generators and loads. Although the transmission medium over which the processors can communicate does not affect the development and analysis herein (indeed, we assume that low-level communication issues do not propagate to the level with which we are concerned), we make the requirement, which we formalize later in this section, that the communication links collectively form a connected network. Furthermore, while a communication network with the same sparsity structure as the physical layer, i.e., one that conforms to the graph $\mathcal{G}_p$ defined in Section 2.1, would suffice, by abstracting the cyber layer from the physical layer, we allow for a very general communication modality; we establish a graph-theoretic model of this communication network next.

For the network formed by the communication links between local processors in the microgrid, we model the interconnections with a directed graph, which we denote by $\mathcal{G}_c = \{\mathcal{V}_c, \mathcal{E}_c\}$. The vertex—or node—set, $\mathcal{V}_c$, consists of processors, one for each bus in the physical layer, i.e., $\mathcal{V}_c := \{1, 2, \ldots, n\}$. Given that each load and generator is outfitted with a local processor, there is a one-to-one correspondence between nodes in the graphs modeling the physical and cyber layers, i.e., there exists a bijective function $h^c : \mathcal{V}_p \mapsto \mathcal{V}_c$. The communication graph edge set, $\mathcal{E}_c \subseteq \mathcal{V}_c \times \mathcal{V}_c$, comprises ordered tuples to represent the flow of information from one node to another; more specifically, the edge $\{i, j\} \in \mathcal{E}_c$ if node i can receive information from node j. By modeling the communication network using directed edges, we allow for the possibility of unidirectional transfers of information which may arise, for example, in situations where the transceiver hardware is inhomogeneous or the transceivers do not broadcast with uniform power.

In subsequent developments, we will rely on some elementary notions from graph theory; we introduce notation and recall the relevant concepts next. The set of nodes from which node i can receive information is referred to as the in-neighborhood of node i and is denoted by $\mathcal{N}_c^-(i) := \{j \in \mathcal{V}_c :$

$\{i, j\} \in \mathcal{E}_c\}$. Similarly, the set of nodes to which node i can send information is referred to as its out-neighborhood and is denoted by $\mathcal{N}_c^+(i) := \{j \in \mathcal{V}_c : \{j, i\} \in \mathcal{E}_c\}$. The number of nodes that can receive information from node i, referred to as its out-degree, is the cardinality of the out-neighborhood and is denoted by $\delta_c^+(i) := |\mathcal{N}_c^+(i)|$.

For a pair of nodes $i, j \in \mathcal{V}_c$, and for $\nu_0, \nu_1, \ldots, \nu_l \in \mathcal{V}_c$ and $e_0, e_1, \ldots, e_{l-1} \in \mathcal{E}_c$, we refer to the alternating sequence of nodes and edges

$$i \equiv \nu_0, e_0, \nu_1, e_1, \nu_2, \ldots, \nu_{l-1}, e_{l-1}, \nu_l \equiv j$$

as a directed path of length l between nodes i and j. Furthermore, we refer to the minimum distance from i to j, $i \neq j$, as the shortest-length directed path between the nodes, and denote its value by $d_c(i, j)$, with $d_c(i, j) = \infty$ if no path exists. The diameter of $\mathcal{G}_c$, which we denote by Δ_c, is defined to be the longest shortest path between any two nodes, i.e., $\Delta_c := \max_{i,j \in \mathcal{V}_c,\, i \neq j} d_c(i, j)$ [43]. [In general, for a graph $\mathcal{G}_x$, we denote its diameter by Δ_x.] Finally, for the algorithms that rely on the graph $\mathcal{G}_c$ introduced in Chapter 3, we require that the graph modeling the communication network be strongly connected, i.e., we require that the diameter of $\mathcal{G}_c$ be finite.

CHAPTER 3

DISTRIBUTED ALGORITHMS

In this chapter, we introduce three distributed algorithms that will be used as primitives for distributively implementing the frequency regulation architecture proposed in Parts II and III. The first two algorithms are consensus-type and allow for the computation of global information with which the individual nodes can make local control decisions. The third algorithm, which we refer to as the *feasible flow algorithm*, enables the computation of generator outputs and the resultant branch power flows that collectively meet the total power demanded by the loads in a microgrid without violating any branch flow limits.

For each of the algorithms, we index the iterations over which locally maintained values are updated by $k = 0, 1, 2, \ldots$. Additionally, we consider fixed communication networks and, although it is not a requirement, a broadcast model is used for all information exchanges, i.e., each node sends the same data to all nodes in its out-neighborhood. Finally, we assume that all transmitted information is successfully received by the intended recipient, i.e., the communication network is completely reliable; however, we also address the problem of packet loss in the context of one of the consensus algorithms by introducing a modification that reduces the negative effects of such temporary communication link failures.

3.1 Consensus-Type Algorithms

We begin this section by providing an overview of two consensus-type algorithms, both of which rely on a communication network that is described by the strongly connected directed graph model introduced in Section 2.2. The first enables the computation of the maximum (or minimum) of locally maintained nodal values in finite time; the second enables the nodes to

asymptotically determine a ratio of sums of values known locally by each of the individual nodes. Additionally, we outline two modified versions of the algorithm for computing ratios of sums: the first increases the robustness of the algorithm to temporary communication link failures; the second allows the nodes to compute an approximation to the true asymptotic value in finite time.

3.1.1 Max- and Min-Consensus Algorithms

Consider the vector $\eta := \Big[\eta_1, \eta_2, \ldots, \eta_n\Big]^{\mathrm{T}}$, where the value of η_i, $i \in \mathcal{V}_c$, is known only by node i, and suppose the nodes are interested in finding the maximum value among the η_i's, which we define to be $\overline{\eta} := \max_{i\in\mathcal{V}_c} \eta_i = \|\eta\|_\infty$. Let $\overline{\mu}_i[k]$ be an estimate for $\overline{\eta}$ maintained by node $i \in \mathcal{V}_c$ at iteration k. Then, as shown in [44], if the nodes initialize their estimates as $\overline{\mu}_i[0] = \eta_i$ and update them according to

$$\overline{\mu}_i[k+1] = \max_{j\in\mathcal{N}_c^-(i)\cup\{i\}} \overline{\mu}_j[k] \tag{3.1}$$

for each iteration k, it follows that, after a finite number of iterations bounded by the diameter of the graph, every node can obtain the value of $\overline{\eta}$, i.e., for some $k_m \leq \Delta_c$, $\overline{\mu}_i[k] = \overline{\eta}$ for $i \in \mathcal{V}_c$ and $k \geq k_m$.

If, in addition to iteratively updating the value of $\overline{\mu}_i[k]$, the nodes also maintain a second value, denoted by $\overline{\nu}_i[k]$, and update it at each k as

$$\overline{\nu}_i[k+1] = \operatorname*{argmax}_{\{\overline{\mu}_j \,:\, j\in\mathcal{N}_c^-(i)\cup\{i\}\}} \overline{\mu}_j[k], \tag{3.2}$$

it follows that $\overline{\nu}_i[k_m] = \operatorname*{argmax}_{\{\eta_j \,:\, j\in\mathcal{V}_c\}} \eta_j$ for $i \in \mathcal{V}_c$ and k_m as defined above, i.e., every node can obtain the index of the node that has the maximum η_i at the same iteration that $\overline{\eta}$ is acquired. (Note that in the case that two or more nodes have the maximizing value, e.g., $\eta_i = \eta_j = \overline{\mu}$, the value of $\overline{\nu}_l[k_m]$ can be taken to be the largest index.)

The formulation of the min-consensus algorithm is identical to the formulation of max consensus with the max operation in (3.1) and (3.2) replaced by the operation min. For completeness, we briefly introduce the min-consensus algorithm next.

Suppose the nodes are interested in finding the minimum value among the η_i's, which we define to be $\underline{\eta} := \min_{i \in \mathcal{V}_c} \eta_i$. Let $\underline{\mu}_i[k]$ be an estimate for $\underline{\eta}$ maintained by node $i \in \mathcal{V}_c$ at iteration k. As with max consensus, if the nodes initialize their estimates as $\underline{\mu}_i[0] = \eta_i$ and update them according to

$$\underline{\mu}_i[k+1] = \min_{j \in \mathcal{N}_c^-(i) \cup \{i\}} \underline{\mu}_j[k] \tag{3.3}$$

for each iteration k, it follows that, after a finite number of iterations bounded by the diameter of the graph, every node can obtain the value of $\underline{\eta}$, i.e., for some $k_m \leq \Delta_c$, $\underline{\mu}_i[k] = \underline{\eta}$ for $i \in \mathcal{V}_c$ and $k \geq k_m$.

3.1.2 Ratio-Consensus Algorithm

Consider a system comprising n processors and assume that the communication network describing the exchange of information between them can be described by the graph-theoretic model described in Section 2.2, i.e., the strongly connected graph $\mathcal{G}_c$. Each processor participating in the ratio-consensus algorithm maintains two values, y_i and z_i, referred to as internal states, which are (independently) updated at each iteration to be a linear combination of the previous internal states of all nodes in its in-neighborhood. Specifically, for each iteration $k \geq 0$, node i updates its internal states as

$$y_i[k+1] = \sum_{j \in \mathcal{N}_c^-(i) \cup \{i\}} \frac{1}{\delta_c^+(j)+1} y_j[k], \tag{3.4}$$

$$z_i[k+1] = \sum_{j \in \mathcal{N}_c^-(i) \cup \{i\}} \frac{1}{\delta_c^+(j)+1} z_j[k], \tag{3.5}$$

where $\delta_c^+(j)$ is the out-degree of processor $j \in \mathcal{N}_c^+(i) \cup \{i\}$. Assuming that $z_i[k] \neq 0, \ \forall k$, at each iteration, each processor $i \in \mathcal{V}_c$ computes

$$\gamma_i[k] = \frac{y_i[k]}{z_i[k]}. \tag{3.6}$$

The following lemma establishes that (3.6) converges to a constant that is equal $\forall i \in \mathcal{V}_c$ (see [45] for a proof).

Lemma 1 (Ratio-Consensus Algorithm)
Let $y_i[k], \ \forall i$, be the result of iteration (3.4) *for some $y_i[0], \ \forall i$, and $z_i[k], \ \forall i$,*

be the result of iteration (3.5) *for some* $z_i[0]$*, where* $z_i[0] > 0,\ \forall i$*; then, we have that*

$$\gamma = \lim_{k\to\infty} \gamma_i[k] = \lim_{k\to\infty} \frac{y_i[k]}{z_i[k]} = \frac{\sum_{l=1}^{n} y_l[0]}{\sum_{l=1}^{n} z_l[0]},\ \forall i \in \mathcal{V}_c. \tag{3.7}$$

□

The result in Lemma 1 implies that, through the exchange of information over the directed graph $\mathcal{G}_c$, ratio consensus allows the local processors to compute $\sum_{j=1}^{n} y_j[0]/\sum_{j=1}^{n} z_j[0]$, thus enabling them to acquire knowledge that is not directly obtainable from their in-neighbors. It should be noted that the number of iterations required to acquire this information is highly dependent on the structure of the graph modeling the communication network. Specifically, the convergence rate of (3.4) and (3.5) will depend on the second-largest in magnitude eigenvalue of the weight matrix $P = [P_{ij}]$, with $[P_{ij}] = \frac{1}{\delta_c^+(j)+1}$ when $\{i, j\} \in \mathcal{E}_c$ and $[P_{ij}] = 0$ otherwise.

3.1.3 Robust Ratio-Consensus Algorithm

We provide an overview of the so-called robust ratio-consensus algorithm, which was originally proposed in [46]. This algorithm is similar to the ratio-consensus algorithm presented in Section 3.1.2, but is modified to increase its robustness to temporary communication link failures that result from, for example, packet loss.

Rather than broadcast the latest state values as in (3.4) – (3.5), nodes participating in the robust ratio-consensus algorithm broadcast the sum of the weighted states up to and including the current iteration k. For the case when all links are available, i.e., no packets are lost, the weighted states for iteration k can be inferred from the information a processor receives from its in-neighbors. If a link is temporarily unavailable, however, the modification to the algorithm allows the receiving nodes to recover any lost information at the next successful iteration.

In order to account for the possibility that communication links may not be available at every iteration, it is necessary to slightly modify the graph-theoretic model describing the exchange of information between processors introduced in Section 2.2. To this end, we denote the graph representing

the network interconnecting the processors as $\mathcal{G}_c[k] = \{\mathcal{V}_c, \mathcal{E}_c[k]\}$, where $\mathcal{V}_c$ is independent of k as defined before, and $\mathcal{E}_c[k]$ is the set of edges where $(i,j) \in \mathcal{E}_c[k]$ if node i can receive information from node j at iteration k. We assume that $\mathcal{E}_c[k] \subseteq \mathcal{E}_c$, $\forall k \geq 0$, where $\mathcal{E}_c$ is as defined in Section 2.2, and describes the scenario in which all communication links are available.

As before, each node maintains two states, $y_i[k]$ and $z_i[k]$, that are updated at each iteration. Using the robust version of the algorithm, however, each node maintains two additional states, $\vartheta_i[k]$ and $\varrho_i[k]$, which are the values broadcasted to the out-neighbors of processor i at iteration k. The values of $\vartheta_i[k]$ and $\varrho_i[k]$ are the sum of $y_i[k]/(\delta_c^+(i)+1)$ and $z_i[k]/(\delta_c^+(i)+1)$ since the iterative process began, and thus, they are updated as follows:

$$\vartheta_i[k+1] = \vartheta_i[k] + \frac{1}{\delta_c^+(i)+1} y_i[k] = \sum_{t=0}^{k} \frac{1}{\delta_c^+(i)+1} y_i[t], \tag{3.8}$$

$$\varrho_i[k+1] = \varrho_i[k] + \frac{1}{\delta_c^+(i)+1} z_i[k] = \sum_{t=0}^{k} \frac{1}{\delta_c^+(i)+1} z_i[t], \tag{3.9}$$

with $\vartheta_i[0] = 0$ and $\varrho_i[0] = 0$. To account for the fact that the values received from in-neighbors are accumulated sums, each node i updates its states as

$$y_i[k+1] = \frac{1}{\delta_c^+(i)+1} y_i[k] + \sum_{j \in \mathcal{N}_c^-(i)} (\varepsilon_{\{i,j\}}[k+1] - \varepsilon_{\{i,j\}}[k]), \tag{3.10}$$

$$z_i[k+1] = \frac{1}{\delta_c^+(i)+1} z_i[k] + \sum_{j \in \mathcal{N}_c^-(i)} (\varphi_{\{i,j\}}[k+1] - \varphi_{\{i,j\}}[k]), \tag{3.11}$$

where the values of $\varepsilon_{\{i,j\}}[k+1]$ and $\varphi_{\{i,j\}}[k+1]$ depend on the successful receipt of a packet from node j during iteration k, and are given by

$$\varepsilon_{\{i,j\}}[k+1] = \begin{cases} \vartheta_j[k+1], & \text{if } \{i,j\} \in \mathcal{E}_c[k], \\ \varepsilon_{\{i,j\}}[k], & \text{if } \{i,j\} \notin \mathcal{E}_c[k], \end{cases} \tag{3.12}$$

$$\varphi_{\{i,j\}}[k+1] = \begin{cases} \varrho_j[k+1], & \text{if } \{i,j\} \in \mathcal{E}_c[k], \\ \varphi_{\{i,j\}}[k], & \text{if } \{i,j\} \notin \mathcal{E}_c[k]. \end{cases} \tag{3.13}$$

Example 1 (Robust Ratio Consensus Six-Node Simulation)
To demonstrate the efficacy of the robust ratio-consensus algorithm, we implement it using the hardware testbed introduced in Section 7.2 and Section 7.3.

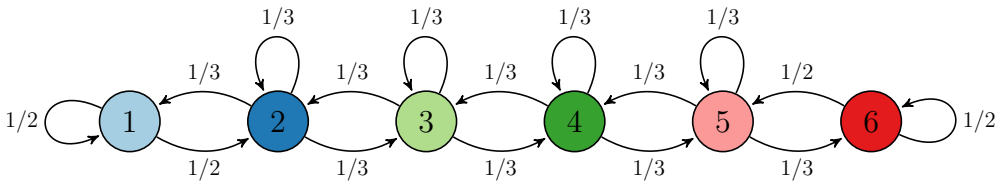


Figure 3.1: Graph of six-node network.

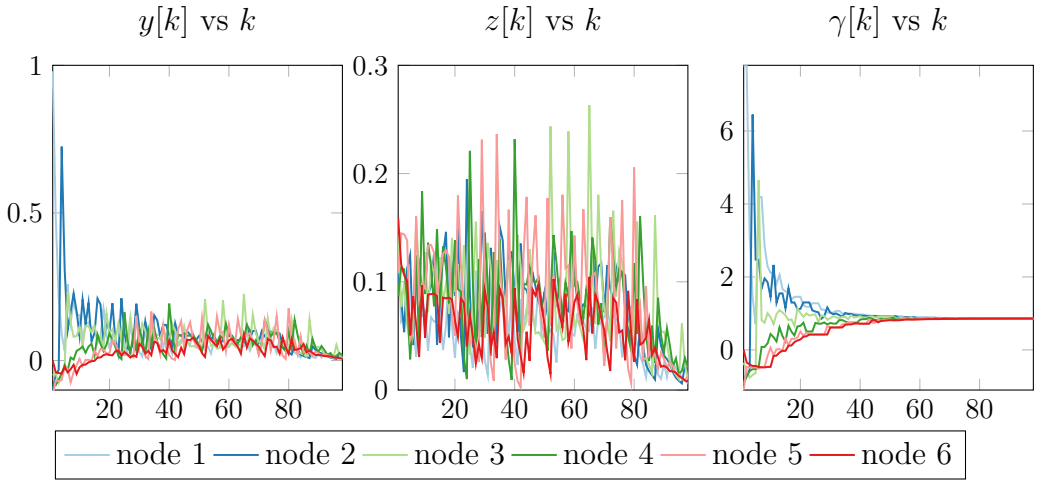


Figure 3.2: Evolution of states for robust constrained algorithm.

The six-node network represented by the graph in Fig. 3.1 is created using this hardware testbed and, in order to induce dropped packets, the iteration period is reduced to 40 *ms and no restrictions are placed on broadcast time. Initially, the values known by each node are given as*

$$y[0] = \Big[0.98, -0.1, -0.05, -0.08, -0.12, 0\Big]^{\mathrm{T}},$$
$$z[0] = \Big[0.126, 0.108, 0.143, 0.087, 0.109, 0.159\Big]^{\mathrm{T}}.$$

The evolution of the internal states, the $y_i[k]$'s and $z_i[k]$'s, and their ratio, the $\gamma_i[k]$'s, as computed by each of the processors in the hardware testbed are shown in Fig. 3.2. The plots of the internal states $y[k]$ and $z[k]$ show erratic behavior that does not appear to converge. Despite this, $\gamma_i[k]$ converges to a steady-state solution at which the ratio states of all the nodes are the same. ■

3.1.4 Finite-Time Approximate Ratio-Consensus Algorithm

In Section 3.1.2, we saw how the ratio consensus algorithm allows the nodes to compute the ratio of sums of locally maintained values. However, as (3.7) implies, the process by which the nodes obtain this value must extend over an infinite number of iterations, limiting its usefulness in practical control applications. One simple approach to acquire an approximation of the asymptotic value to which ratio consensus converges is to execute the algorithm for a finite number of iterations, predetermined so as to ensure that the value is within some accuracy bounds. While this is an adequate strategy, the number of iterations needed to converge to a sufficiently accurate solution depends on, among other things, the connectivity of the graph modeling the communication network (see, e.g., [47]). Given that it is difficult to distributively determine how many iterations are sufficient, this approach reduces the practicality of the control architecture for which the algorithm is a primitive. Thus, in the discussion that follows, we introduce an extension to the ratio-consensus algorithm that allows the nodes to compute an approximation to the asymptotic value obtained through ratio consensus to within a pre-specified bound in finite time.

Next, we define the approximate ratio consensus problem. Then, we introduce an algorithm for approximate ratio consensus that takes advantage of the max- and min-consensus algorithms outlined in Section 3.1.1. Finally, we demonstrate the algorithm using numerical simulations of two case studies.

Definition 1 (ϵ-Approximate Ratio Consensus)
Consider a network of nodes described by the directed graph outlined in Section 2.2, $\mathcal{G}_c = \{\mathcal{V}_c, \mathcal{E}_c\}$. Each node i maintains two states, denoted by $y_i[k]$ and $z_i[k]$, which are updated for $k = 0, 1, 2, \ldots, k_0 < \infty$ according to (3.4) *and* (3.5)*, respectively. At the end of this iterative process, the nodes have reached ϵ-approximate ratio consensus if*

$$\left|\frac{y_i[k_0]}{z_i[k_0]} - \gamma\right| < \epsilon, \ \forall i \in \mathcal{V}_c,$$

where γ is as defined in (3.7)*.*

Let ϵ be an error bound specified *a priori* and known by all nodes; then, the goal of the approximate ratio consensus problem is to distributively allow each node i to compute $\gamma_i[k_0]$ for $k_0 < \infty$ such that $|\gamma_i[k_0] - \gamma| < \epsilon$, $\forall i \in \mathcal{V}_c$.

Put differently, an algorithm for approximate ratio consensus is a distributed protocol that allows the nodes to reach ϵ-approximate ratio consensus.

Algorithm 1: Approximate Ratio-Consensus Algorithm

Input: $y_i[0], z_i[0], \epsilon$
Output: γ_i

begin
  set
    $\overline{\mu}_i[0] = \infty,\ \underline{\mu}_i[0] = -\infty,\ p_{ji} = \frac{1}{\delta_c^+(i)+1}$
  Let $k \geq 0$ index iterations
  repeat
    if $k \mod \Delta_c = 0$ **then**
      if $\overline{\mu}_i[k] - \underline{\mu}_i[k] < \epsilon$ **then**
        set
          $\sigma_i[k] = 1$
      set
        $\overline{\mu}_i[k] = \underline{\mu}_i[k] = y_i[k]/z_i[k]$
    broadcast to all $j \in \mathcal{N}_c^+(i)$
      $p_{ji}y_i[k], p_{ji}z_i[k], \overline{\mu}_i[k], \underline{\mu}_i[k]$
    receive from all $l \in \mathcal{N}_c^-(i)$
      $p_{il}y_l[k], p_{il}z_l[k], \overline{\mu}_l[k], \underline{\mu}_l[k]$
    compute
      $y_i[k+1] = \sum_{l \in \mathcal{N}_c^-(i)} p_{il}y_l[k]$
      $z_i[k+1] = \sum_{l \in \mathcal{N}_c^-(i)} p_{il}z_l[k]$
      $\overline{\mu}_i[k+1] = \max_{l \in \mathcal{N}_c^-(i)} \overline{\mu}_l[k]$
      $\underline{\mu}_i[k+1] = \min_{l \in \mathcal{N}_c^-(i)} \underline{\mu}_l[k]$
  until $\sigma_i[k] = 0$
  return γ_i

As mentioned previously, the algorithm we propose for approximate ratio consensus relies on the execution of min- and max-consensus algorithms; by executing min and max consensus in parallel, the nodes can simultaneously determine (an upper bound on) the iteration at which their ratios, $\{\gamma_i[k_0] \mid i \in \mathcal{V}\}$, are within ϵ of each other. Similar to ratio consensus, each node i participating in the algorithm we propose maintains two states, de-

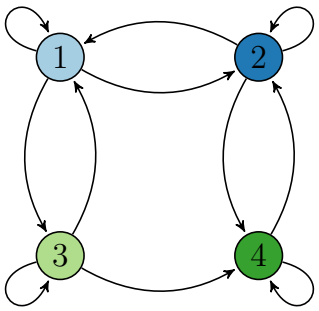


Figure 3.3: Graph of 4-node network.

noted by $y_i[k]$ and $z_i[k]$, which we refer to as *primary states*. In addition, each node i will maintain two *auxiliary states*, denoted by $\underline{\mu}_i[k]$ and $\overline{\mu}_i[k]$, which are updated using min- and max-consensus, respectively. As in [44], the auxiliary states, $\underline{\mu}_i[k]$ and $\overline{\mu}_i[k]$, will be periodically reinitialized.

To allow the nodes to determine when all the ratios in the set $\{\gamma_i \mid i \in \mathcal{V}_c\}$ are close to the asymptotic value, we use min- and a max-consensus, reset every Δ_c steps. More specifically, when $k \mod \Delta_c = 0$, the values of $\underline{\mu}_i[k]$ and $\overline{\mu}_i[k]$ are reinitialized to be $\underline{\mu}_i[k] = \overline{\mu}_i[k] = \gamma_i[k]$. Before reinitializing the values of $\underline{\mu}_i[k]$ and $\overline{\mu}_i[k]$, node i checks if the worst case error between the current maximum and minimum, given by $|\overline{\mu}_i[k] - \underline{\mu}_i[k]| = \overline{\mu}_i[k] - \underline{\mu}_i[k]$, is smaller than the desirable error bound, i.e., if $\overline{\mu}_i[k] - \underline{\mu}_i[k] < \epsilon$ for $k \mod \Delta_c = 0$; if that is the case, then the nodes stop iterating. The pseudocode for the algorithm above is provided in Algorithm 1.

Example 2 (Four-Node Approximate Ratio Consensus Simulation) *In [40] we used a 4-bus laboratory microgrid to demonstrate a distributed frequency regulation scheme based upon ratio consensus. In the experimental setup in [40], the exchange of information between the local processors is modeled by the graph in Fig. 3.3 and each instance of the ratio consensus algorithm is specified beforehand to be executed for 75 iterations. Furthermore, in the results in [40], it was shown that 3 serial instances of ratio consensus were necessary to achieve the control objectives therein; thus, in total 225 ratio consensus iterations were required.*

Using Algorithm 1, we simulated the same network with similar initial conditions to determine the number of iterations necessary to converge to a solution in which $|\gamma_i - \gamma| < 0.0001$*, i.e.,* $\epsilon = 0.0001$*. The plots in Figs. 3.4, 3.5, and 3.6 show the evolution of the primary ratio, min consensus, and max*

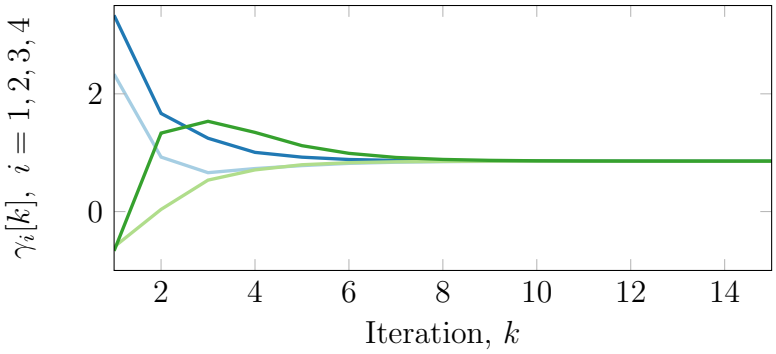


Figure 3.4: Primary ratio evolution for 4-node network.

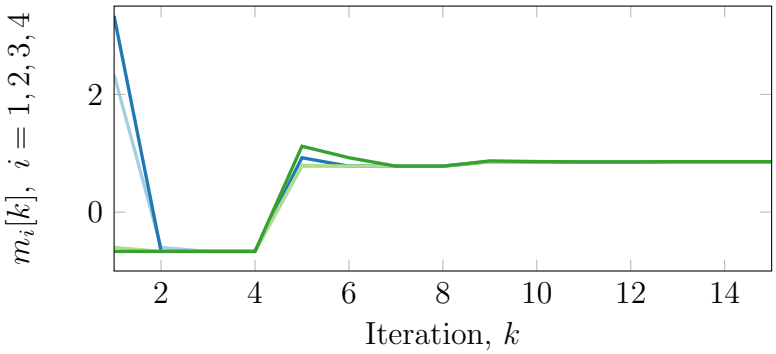


Figure 3.5: Evolution of min consensus for 4-node network.

consensus as computed by the nodes, respectively; as the figures illustrate, only 15 iterations are required before the nodes determine the maximum error is below the bound. Compared with the results in [40], the finite-time approximate consensus scheme requires only 48 iterations. ■

Example 3 (12-Node Approximate Ratio Consensus Simulation) *While ratio consensus, and by extension, the algorithm proposed in this section, can be used to compute more general quantities, i.e., ratios of sums of values known locally by each of the nodes, over a network of interconnected nodes, if properly initialized, it can also be used to compute the average of some values possessed by the nodes. More specifically, suppose each node i possesses a value π_i. To find the average of these values, it suffices to initialize the states in* (3.4) *and* (3.5) *to $y_i[0] = \pi_i$ and $z_i[0] = 1$, for all $i \in \mathcal{V}_c$, respectively. As implied by* (3.7)*, the value to which the computations performed by the nodes will asymptotically approach is $\gamma = \frac{\sum_{l=1}^{n} \pi_l}{\sum_{l=1}^{n} 1} = \frac{1}{n} \sum_{l=1}^{n} \pi_l$, i.e., the average of the values π_i. Using this approach, we compare our proposed algorithm with the ones in [44] and [48] by simulating it for the 12-node network*

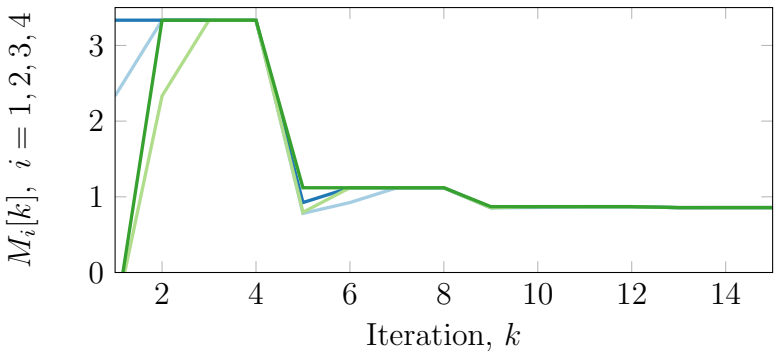


Figure 3.6: Evolution of max consensus for 4-node network.

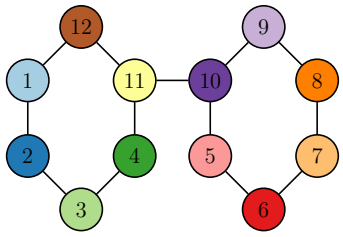


Figure 3.7: Graph of 12-node network (self loops omitted).

illustrated by the graph in Fig. 3.7; while our proposed algorithm works for networks with unidirectional communication links, the network we consider for this example is undirected.

As in the work to which we compare our proposed algorithm, we compute the average of the following values:

$$\pi = \Big[a, a, a, a, b, b, b, b, b, b, a, a\Big]^{\mathrm{T}},$$

where $a = 0.001$ *and* $b = 100$*; thus, the true average is* $\overline{\pi} = 50.0005$*. Similarly, the accuracy to which the average is to be computed is* $\epsilon = 0.0001$*. Given these initial conditions and bound, the plots in Figs. 3.8, 3.9, and 3.10 show the evolution of the ratio of primary states, min consensus, and max consensus as computed by the nodes, respectively. As these figures illustrate, the nodes converge rapidly, requiring only 263 iterations to determine that all the ratios of the primary states are within* ϵ *of the true average.*

To reach the same level of accuracy, the algorithms in [44] and [48] require 1059 and 1095 iterations, respectively; comparatively, our approach requires 24.8% and 24.0% fewer iterations, respectively. While this represents a significant reduction in the number of iterations required, nodes in our approach

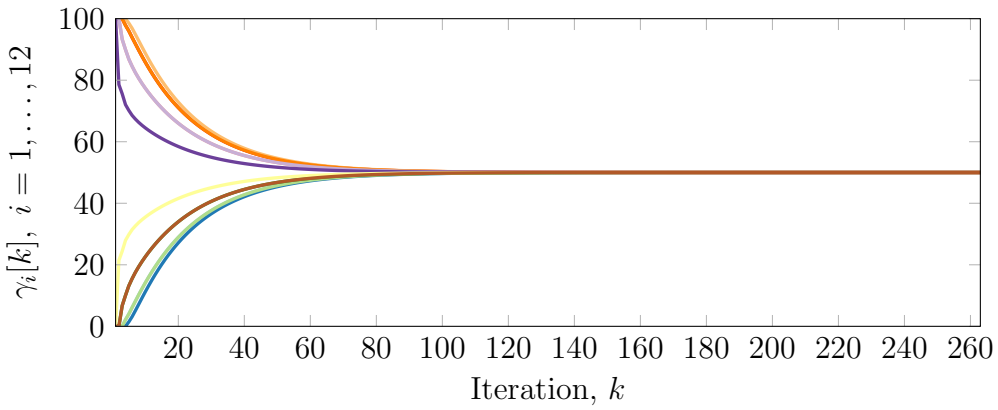


Figure 3.8: Primary ratio evolution for 12-node network.

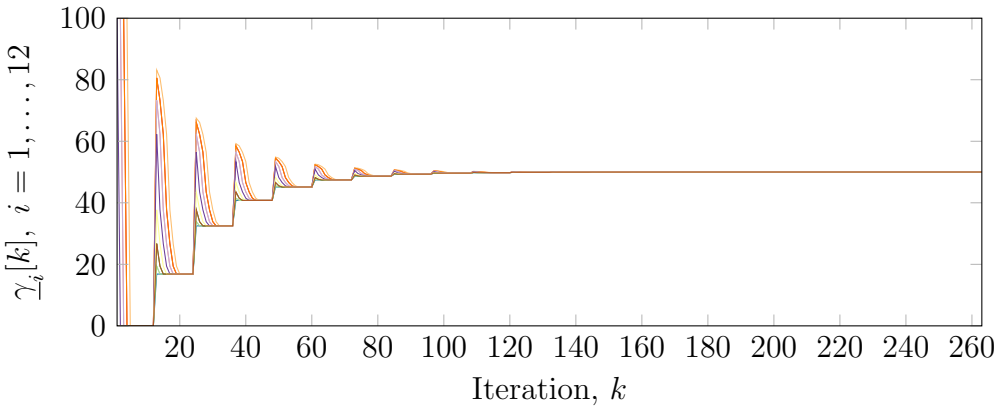


Figure 3.9: Evolution of min consensus for 12-node network.

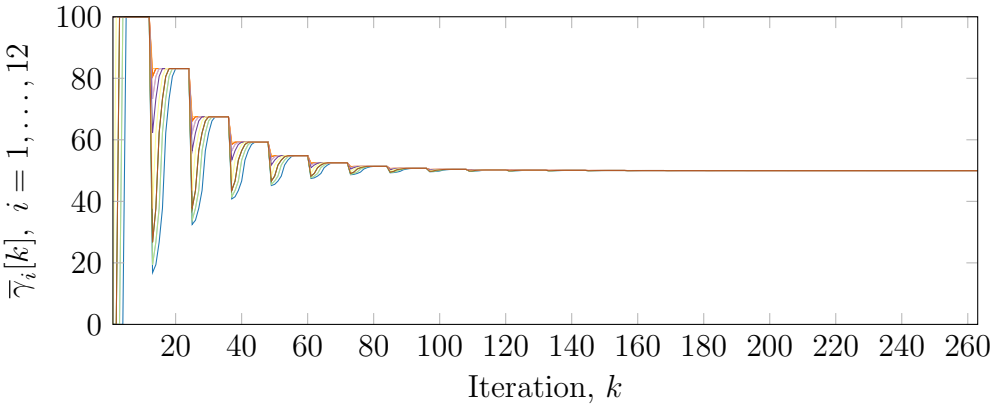


Figure 3.10: Evolution of max consensus for 12-node network.

must maintain four state variables compared to three and one in [44] and [48], respectively. Despite this additional overhead, the algorithm we propose can be used for more general computations and works in networks for which the graph representing the information exchange is directed and unbalanced.

Furthermore, the effects on the communication network would be minimal as all values exchanged between nodes in our approach can be included in a single packet of even the most simple protocols, e.g., IEEE 802.15.4. ■

3.2 Feasible Flow Algorithm

We begin the following discussion by defining a directed graph, which is based upon the undirected one in Section 2.1, and introducing other notation that will be used to model the branch power flows in a microgrid. Then, we introduce the problem of determining individual generator outputs that collectively balance the total power demanded by the loads in a microgrid without violating any generation or branch power flow limits. After formalizing the objectives of this problem, which we refer to as the *feasible flow problem*, we propose an algorithm for distributively solving it.

3.2.1 Directed Graph Network Flow Model and Notation

From the undirected graph modeling the microgrid network introduced in Section 2.1, consider a directed graph $\mathcal{G}_d = \{\mathcal{V}_d, \mathcal{E}_d\}$, where $\mathcal{V}_d = \mathcal{V}_p = \mathcal{V}_g \cup \mathcal{V}_\ell$, and the directed edge set, $\mathcal{E}_d$, consists of the natural and reverse orientation of the edges in $\mathcal{G}_p$, i.e., $\mathcal{E}_d := \{\{i,j\}, \{j,i\} : (i,j) \in \mathcal{E}_p\}$. We refer to the set of nodes from which directed edges originate that terminate at node $i \in \mathcal{V}_d$ as its in-neighborhood and define it to be $\mathcal{N}_d^-(i) := \{j : \{j,i\} \in \mathcal{E}_d\}$. Similarly, we refer to the set of nodes that terminate a directed edge originating from node $i \in \mathcal{V}_d$ as its out-neighborhood and define it to be $\mathcal{N}_d^+(i) := \{j : \{i,j\} \in \mathcal{E}_d\}$. We respectively denote the cardinality of the in- and out-neighborhood of node i by $\delta_d^-(i) = |\mathcal{N}_d^-(i)|$ and $\delta_d^+(i) = |\mathcal{N}_d^+(i)|$. As before, if we assume that the microgrid has no islands, i.e., the graph $\mathcal{G}_p$ consists of a single connected component, it follows that the directed graph $\mathcal{G}_d$ is strongly connected.

For each directed edge $\{i,j\} \in \mathcal{E}_d$, let $f_{\{i,j\}} \geq 0$ denote the nonnegative power flow from node i to node j, which is restricted to be within lower and upper limits, $\underline{f}_{\{i,j\}}$ and $\overline{f}_{\{i,j\}}$, respectively, i.e., $0 \leq \underline{f}_{\{i,j\}} \leq f_{\{i,j\}} \leq \overline{f}_{\{i,j\}}$. To ensure there are no source or sink nodes, we additionally restrict the flow limits such that at least one $\underline{f}_{\{i,j\}} > 0$ for $j \in \mathcal{N}_d^-(i)$ and at least one $\underline{f}_{\{i,j\}} > 0$ for $l \in \mathcal{N}_d^+(i)$. Given these flows, we define the total in- and out-flow at each

node i respectively as $f_i^- := \sum_{j \in \mathcal{N}_d^-(i)} f_{\{j,i\}}$ and $f_i^+ := \sum_{l \in \mathcal{N}_d^+(i)} f_{\{i,l\}}$. For each generator node $i \in \mathcal{V}_g$, let g_i denote its output, where $0 \leq \underline{g}_i \leq g_i \leq \overline{g}_i$ and $\underline{g}_i$ and $\overline{g}_i$ denote lower and upper limits, respectively. (Note that, while related, the set-point of generator $i \in \mathcal{V}_g$ and its respective limits, i.e., $u_i, \underline{u}_i, \overline{u}_i$, defined in Section 2.1 are not necessarily equivalent to $g_i, \underline{g}_i, \overline{g}_i$.) Similarly, for each load node $i \in \mathcal{V}_\ell$, let $\ell_i \geq 0$ denote its power demand. Finally, let

$$b_i := \begin{cases} f_i^- - f_i^+ + g_i, & i \in \mathcal{V}_g, \\ f_i^- - f_i^+ - \ell_i, & i \in \mathcal{V}_\ell, \end{cases} \tag{3.14}$$

be the *flow balance* at each node $i \in \mathcal{V}_d$, which is defined to be positive for the case in which the collective positive injections exceed the collective negative injections.

3.2.2 Feasible Flow Problem Statement

Based upon the definitions above, the *feasible flow problem* is summarized as follows. Suppose that the power demands at the load buses, i.e., the ℓ_i's, are known; then, the objective is to assign values to each of the flows $f_{\{i,j\}}$, $\{i,j\} \in \mathcal{E}_d$ and generator outputs g_i, $i \in \mathcal{V}_g$, such that the flow balance at each node is zero, the total generator outputs are balanced with the load power demands, and all flow assignments and generator outputs are within limits. More specifically, the objective is to obtain a set of flows $\{f_{\{i,j\}} : \{i,j\} \in \mathcal{E}_d\}$ and generator outputs $\{g_i : i \in \mathcal{V}_g\}$ such that:

F1. $\underline{f}_{\{i,j\}} \leq f_{\{i,j\}} \leq \overline{f}_{\{i,j\}}$ for $\{i,j\} \in \mathcal{E}_d$;

F2. $\underline{g}_i \leq g_i \leq \overline{g}_i$ for $i \in \mathcal{V}_g$; and

F3. $b_i = 0$ for $i \in \mathcal{V}_d$.

If we define $f := [f_e]$, $\underline{f} := [\underline{f}_e]$, $\overline{f} := [\overline{f}_e] \in \mathbb{R}^{|\mathcal{E}_d|}$ for $e \in \mathcal{E}_d$ and $g := [g_i], g := [\underline{g}_i], g := [\overline{g}_i] \in \mathbb{R}^{|\mathcal{V}_g|}$, for $i \in \mathcal{V}_g$, the feasible flow problem can be written in

matrix form as follows:

$$\begin{aligned} \text{find} \quad & f, g \\ \text{subject to} \quad & M_d f - \left[g^{\mathrm{T}}, -\ell^{\mathrm{T}}\right]^{\mathrm{T}} = 0_{|\mathcal{V}_d|}, \\ & \underline{f} \leq f \leq \overline{f}, \\ & \underline{g} \leq g \leq \overline{g}, \end{aligned} \tag{3.15}$$

where $0_{|\mathcal{V}_d|}$ is the $|\mathcal{V}_d|$-dimensional all zeros vector and $M_d \in \mathbb{R}^{|\mathcal{V}_d| \times |\mathcal{E}_d|}$ is the incidence matrix of $\mathcal{G}_d$, and is defined as $M_d := [m_{ie}^d]$ where

$$m_{ie}^d = \begin{cases} -1, & \text{if } i \text{ is the sink node of edge } e, \\ 1, & \text{if } i \text{ is the source node of edge } e, \\ 0, & \text{otherwise,} \end{cases} \tag{3.16}$$

for $e = \{i, j\} \in \mathcal{E}_d$.

Remark 1
For the case when $\left[g^{\mathrm{T}}, -\ell^{\mathrm{T}}\right]^{\mathrm{T}} = 0_{|\mathcal{V}_d|}$*, the feasible flow problem is equivalent to the network-theoretical problem of finding a flow assignment in the nullspace of* M_d*—the so-called circulation space—subject to limits (see, e.g., [49, 50]).* □

3.2.3 Feasible Flow Algorithm

In the discussion that follows, we introduce an algorithm that iteratively adjusts locally maintained estimates for flows and generator outputs such that the estimates asymptotically approach values that satisfy the feasible flow problem. Unlike the consensus-type algorithms presented in Section 3.1, the communication network on which this algorithm relies must conform to the graph modeling the physical microgrid network, $\mathcal{G}_p$; however, in Remark 3, we discuss one way that could be used to eliminate this topological requirement.

To support the distributed nature of our proposed algorithm, each local processor maintains an estimate for the value of the flows along edges connecting it to all of its in- and out-neighbors. More specifically, for each node $i \in \mathcal{V}_d$, the estimate maintained by i for the incoming flow from each $j \in \mathcal{N}_d^-(i)$ at iteration $k = 0, 1, 2, \ldots$ is denoted by $\hat{f}_{\{j,i\}}^i[k]$; similarly, node

i's estimate for the outgoing flow to $l \in \mathcal{N}_d^+(i)$ is denoted by $\hat{f}^i_{\{i,l\}}[k]$. Additionally, for each generator node $i \in \mathcal{V}_g$, we denote an estimate for its output at iteration k by $\hat{g}_i[k]$. Finally, based upon the flow and generator output estimates, the value of the flow balance at each node is computed at each iteration as

$$\hat{b}_i[k] = \begin{cases} \sum_{j\in\mathcal{N}_d^-(i)} \hat{f}^i_{\{j,i\}}[k] - \sum_{l\in\mathcal{N}_d^+(i)} \hat{f}^i_{\{i,l\}}[k] + \hat{g}_i[k], & i \in \mathcal{V}_g, \\ \sum_{j\in\mathcal{N}_d^-(i)} \hat{f}^i_{\{j,i\}}[k] - \sum_{l\in\mathcal{N}_d^+(i)} \hat{f}^i_{\{i,l\}}[k] - \ell_i, & i \in \mathcal{V}_\ell. \end{cases} \tag{3.17}$$

The algorithm we propose for distributively solving the feasible flow problem is given by the following procedure in which the flow and generator output estimates are initialized and iteratively updated using a three-step process.

[**Initialization**] Each node i initializes its incoming and outgoing flow estimates to be the average of its respective lower and upper limit, i.e., $\hat{f}^i_{\{j,i\}}[0] = \frac{1}{2}(\underline{f}_{\{j,i\}} + \overline{f}_{\{j,i\}})$, $j \in \mathcal{N}_d^-(i)$ and $\hat{f}^i_{\{i,l\}}[0] = \frac{1}{2}(\underline{f}_{\{i,l\}} + \overline{f}_{\{i,l\}})$, $l \in \mathcal{N}_d^+(i)$. Analogously, the estimate for the output of each generator node is initialized as $\hat{g}_i[0] = \frac{1}{2}(\underline{g}_i + \overline{g}_i)$, $i \in \mathcal{V}_g$.

[**Step 1**] Node i adjusts its estimate for each in- and out-flow in such a way that drives the flow balance estimate to zero, i.e., $\hat{b}_i[k] \to 0$ as $k \to \infty$. More specifically, node i adjusts its estimate for each in- and out-flow as

$$\tilde{f}^i_{\{j,i\}}[k+1] = \hat{f}^i_{\{j,i\}}[k] - \frac{\hat{b}_i[k]}{w_i},\ j \in \mathcal{N}_d^-(i), \tag{3.18}$$

$$\tilde{f}^i_{\{i,l\}}[k+1] = \hat{f}^i_{\{i,l\}}[k] + \frac{\hat{b}_i[k]}{w_i},\ l \in \mathcal{N}_d^+(i), \tag{3.19}$$

respectively, where $w_i := \delta_d^-(i) + \delta_d^+(i) + 1$. If node $i \in \mathcal{V}_g$, its output estimate is adjusted in a similar fashion to the in-flows, i.e.,

$$\hat{g}_i[k+1] = \hat{g}_i[k] - \frac{\hat{b}_i[k]}{w_i}. \tag{3.20}$$

[**Step 2**] Since each node updates its flow estimates during Step 1 independently, two neighboring nodes may have different estimates for the flow between them. Thus, by exchanging flow estimates with neighboring nodes,

each node updates its locally maintained estimates as

$$\hat{f}^i_{\{j,i\}}[k+1] = \frac{1}{2}\left(\tilde{f}^i_{\{j,i\}}[k+1] + \tilde{f}^j_{\{i,j\}}[k+1]\right),\ j \in \mathcal{N}_d^-(i), \tag{3.21}$$

$$\hat{f}^i_{\{i,l\}}[k+1] = \frac{1}{2}\left(\tilde{f}^i_{\{i,l\}}[k+1] + \tilde{f}^l_{\{l,i\}}[k+1]\right),\ l \in \mathcal{N}_d^+(i). \tag{3.22}$$

Note that while the generator outputs are analogous to in-flows, only each respective local processor maintains an estimate for its output, eliminating the need for an agreement step.

[**Step 3**] Finally, during the first two steps, each flow estimate may have been adjusted in such a way that violates limits. To ensure the flow assignment to which the nodes converge is feasible, any flow estimate exceeding its upper or lower bound is clamped to its respective limit, i.e.,

$$\hat{f}^i_{\{j,i\}}[k+1] = \begin{cases} \overline{f}_{\{j,i\}}, & \text{if } \hat{f}^i_{\{j,i\}}[k+1] > \overline{f}_{\{j,i\}}, \\ \underline{f}_{\{j,i\}}, & \text{if } \hat{f}^i_{\{j,i\}}[k+1] < \underline{f}_{\{j,i\}}, \end{cases} \quad j \in \mathcal{N}_d^-(i), \tag{3.23}$$

$$\hat{f}^i_{\{l,i\}}[k+1] = \begin{cases} \overline{f}_{\{l,i\}}, & \text{if } \hat{f}^i_{\{l,i\}}[k+1] > \overline{f}_{\{l,i\}}, \\ \underline{f}_{\{l,i\}}, & \text{if } \hat{f}^i_{\{l,i\}}[k+1] < \underline{f}_{\{l,i\}}, \end{cases} \quad l \in \mathcal{N}_d^-(i). \tag{3.24}$$

Similarly, the estimates for each generator output must be clamped to be within limits:

$$\hat{g}_i[k+1] = \begin{cases} \overline{g}_i, & \text{if } \hat{g}_i[k+1] > \overline{g}_i, \\ \underline{g}_i, & \text{if } \hat{g}_i[k+1] < \underline{g}_i. \end{cases} \tag{3.25}$$

Given that the nodes incident to any given edge clamp their flow estimates to the same limits during Step 3, it follows that at the beginning of each iteration, the estimates maintained by both nodes are equal, i.e., $\hat{f}^i_{\{j,i\}}[k] = \hat{f}^j_{\{i,j\}}[k]$, $\{i,j\} \in \mathcal{E}_d$; thus, we can combine Steps 1 and 2 into one operation in which

$$\hat{f}^i_{\{j,i\}}[k+1] = \hat{f}^i_{\{j,i\}}[k] + \frac{1}{2}\frac{\hat{b}_j[k]}{w_j} - \frac{1}{2}\frac{\hat{b}_i[k]}{w_i},\ j \in \mathcal{N}_d^-(i), \tag{3.26}$$

$$\hat{f}^i_{\{i,l\}}[k+1] = \hat{f}^i_{\{i,l\}}[k] + \frac{1}{2}\frac{\hat{b}_i[k]}{w_i} - \frac{1}{2}\frac{\hat{b}_l[k]}{w_l},\ l \in \mathcal{N}_d^+(i), \tag{3.27}$$

and, as before, each estimate is clamped to its limit according to (3.23) and

Algorithm 2: Distributed Feasible Flow Algorithm

Input: $\underline{f}_e, \overline{f}_e, e \in \mathcal{E}_d;\ \underline{g}_l, \overline{g}_l, l \in \mathcal{V}_g;\ \ell_j, j \in \mathcal{V}_\ell$
Output: $f^{i*}_{\{j,i\}}, f^{i*}_{\{i,l\}}, \{j,i\}, \{i,l\} \in \mathcal{E}_d;\ g^*_l, l \in \mathcal{V}_g$

Each node $i \in \mathcal{V}_d$ separately does the following:

begin

initialize

$\hat{f}^i_{\{j,i\}}[0] = \frac{1}{2}(\underline{f}_{\{j,i\}} + \overline{f}_{\{j,i\}}),\ j \in \mathcal{N}_d^-(i)$

$\hat{f}^i_{\{i,l\}}[0] = \frac{1}{2}(\underline{f}_{\{i,l\}} + \overline{f}_{\{i,l\}}),\ l \in \mathcal{N}_d^+(i)$

$\hat{g}_i[0] = \frac{1}{2}(\underline{g}_i + \overline{g}_i),\ i \in \mathcal{V}_g$

foreach *iteration,* $k = 0, 1, \ldots, k_f$ **do**

set

$\tilde{b}_i[k] = \hat{g}_i[k]$ if $i \in \mathcal{V}_g$, else $\tilde{b}_i[k] = -\ell_i$

compute

$\hat{b}_i[k] = \sum_{j \in \mathcal{N}_d^-(i)} \hat{f}^i_{\{j,i\}}[k] - \sum_{l \in \mathcal{N}_d^+(i)} \hat{f}^i_{\{i,l\}}[k] + \tilde{b}_i[k]$

transmit

$\hat{b}_i[k]/w_i$ to $j \in \mathcal{N}_d^-(i)$ and $l \in \mathcal{N}_d^+(i)$

receive

$\hat{b}_j[k]/w_j$ from $j \in \mathcal{N}_d^-(i)$

$\hat{b}_l[k]/w_l$ from $l \in \mathcal{N}_d^+(i)$

compute

$\hat{f}^i_{\{j,i\}}[k+1] = \hat{f}^i_{\{j,i\}}[k] + \frac{1}{2}\frac{\hat{b}_j[k]}{w_j} - \frac{1}{2}\frac{\hat{b}_i[k]}{w_i},\ j \in \mathcal{N}_d^-(i)$

$\hat{f}^i_{\{i,l\}}[k+1] = \hat{f}^i_{\{i,l\}}[k] + \frac{1}{2}\frac{\hat{b}_i[k]}{w_i} - \frac{1}{2}\frac{\hat{b}_l[k]}{w_l},\ l \in \mathcal{N}_d^+(i)$

$\hat{g}_i[k+1] = \hat{g}_i[k] - \frac{\hat{b}_i[k]}{w_i},\ i \in \mathcal{V}_g$

set

$$\hat{f}^i_{ji}[k+1] = \begin{cases} \underline{f}_{\{j,i\}}, \text{ if } \hat{f}_{\{j,i\}}[k+1] < \underline{f}_{\{j,i\}} \\ \overline{f}_{\{j,i\}}, \text{ if } \hat{f}_{\{j,i\}}[k+1] > \overline{f}_{\{j,i\}} \end{cases} j \in \mathcal{N}_d^-(i)$$

$$\hat{f}^i_{li}[k+1] = \begin{cases} \underline{f}_{\{l,i\}}, \text{ if } \hat{f}_{\{l,i\}}[k+1] < \underline{f}_{\{l,i\}} \\ \overline{f}_{\{l,i\}}, \text{ if } \hat{f}_{\{l,i\}}[k+1] > \overline{f}_{\{l,i\}} \end{cases} l \in \mathcal{N}_d^-(i)$$

$$\hat{g}_i[k+1] = \begin{cases} \underline{g}_i, \text{ if } \hat{g}_i[k+1] < \underline{g}_i \\ \overline{g}_i, \text{ if } \hat{g}_i[k+1] > \overline{g}_i \end{cases} i \in \mathcal{V}_g$$

For k_f sufficiently large, set:

$f^{i*}_{\{j,i\}} = \hat{f}^i_{\{j,i\}}[k_f],\ j \in \mathcal{N}_d^-(i),\ f^{i*}_{\{i,l\}} = \hat{f}^i_{\{i,l\}}[0],\ l \in \mathcal{N}_d^+(i);$

$g^*_l = \hat{g}_l[k_f],\ l \in \mathcal{V}_g$

(3.24).

The following proposition establishes the convergence of the algorithm to a solution that satisfies the feasible flow problem.

Proposition 1 (Convergence of the Feasible Flow Algorithm) *Suppose, for given load power demands, ℓ, a solution to the feasible flow problem specified by the objectives in F1 – F3 exists and let f_e^* for $e \in \mathcal{E}_d$ and g_l^* for $l \in \mathcal{V}_g$ denote the flows and generator outputs that satisfy it. Then, Algorithm 2 is guaranteed to yield f^* and g^* such that $M_d f^* - \left[g^*, -\ell\right]^{\mathrm{T}} = 0_{|\mathcal{V}_d|}$, and $\underline{f} \leq f^* \leq \overline{f}$ and $\underline{g} \leq g^* \leq \overline{g}$, where $f^* := [f_e^*]$ for $e \in \mathcal{E}_d$ and $g^* := [g_i^*]$ for $i \in \mathcal{V}_g$, i.e., as $k \to \infty$, $\hat{f}_e[k] \to f_e^*$, $e \in \mathcal{E}_d$; $\hat{g}_l[k] \to g_l^*$, $l \in \mathcal{V}_g$; and $\hat{b}_i[k] \to 0$ for $i \in \mathcal{V}_d$.*

Proof 1 (Proof for Proposition 1) *Consider the following optimization problem:*

$$\begin{array}{ll} \min\limits_{g,f} & b^{\mathrm{T}} W b \\ s.t. & \underline{f}_{\{i,j\}} \leq f_{\{i,j\}} \leq \overline{f}_{\{i,j\}}, \quad \{i,j\} \in \mathcal{E}_d \\ & \underline{g}_i \leq g_i \leq \overline{g}_i, \quad i \in \mathcal{V}_g, \end{array} \tag{3.28}$$

which is a quadratic programming (QP) problem, where W is some diagonal weight matrix. To solve QP, we define sets $\mathcal{D}_g = \{g : \underline{g}_j \leq g_j \leq \overline{g}_j,\ \forall j \in \mathcal{V}_g\}$ and $\mathcal{D}_f = \{f : \underline{f}_{\{i,j\}} \leq f_{\{i,j\}} \leq \overline{f}_{\{i,j\}}, \forall \{i,j\} \in \mathcal{E}_d\}$, and apply the gradient projection method, the iterations of which are given by:

$$g[k+1] = \left[g[k] - s \left(\frac{d b^{\mathrm{T}} W b}{du}[k] \right)^{\mathrm{T}} \right]_{\mathcal{D}_g}^{+}, \tag{3.29}$$

$$f[k+1] = \left[f[k] - s \left(\frac{d b^{\mathrm{T}} W b}{df}[k] \right)^{\mathrm{T}} \right]_{\mathcal{D}_f}^{+}, \tag{3.30}$$

where s is some constant stepsize chosen small enough as discussed in [51, Proposition 2.3.2], and $[\cdot]_{\mathcal{D}}^{+}$ denotes the projection onto the set $\mathcal{D}$. By [51, Proposition 2.3.1], (3.29) – (3.30) *converge to a stationary point that is a global minimum by [51, Proposition 2.1.2] since the cost function $b^{\mathrm{T}} W b$ is convex in g and f. The iterations in* (3.29) – (3.30) *can be simplified as*

follows:

$$g[k+1] = \left[g[k] - s \left(\frac{\partial b}{\partial g} \right)^{\mathrm{T}} W b[k] \right]^{+}_{\mathcal{D}_g}, \tag{3.31}$$

$$f[k+1] = \left[f[k] - s \left(\frac{\partial b}{\partial f} \right)^{\mathrm{T}} W b[k] \right]^{+}_{\mathcal{D}_f}. \tag{3.32}$$

The term $\frac{\partial b}{\partial g}$ *is a diagonal matrix with*

$$\left(\frac{\partial b}{\partial g} \right)_{ii} = 1, \quad \forall i \in \mathcal{V}_g. \tag{3.33}$$

Suppose $l \in \mathcal{N}_d^+(i)$*, then,*

$$\left(\frac{\partial b}{\partial f} \right)_{il} = \frac{\partial b_i}{\partial f_{\{i,l\}}} = -1, \tag{3.34}$$

and

$$\left(\frac{\partial b}{\partial f} \right)_{li} = \frac{\partial b_l}{\partial f_{\{i,l\}}} = 1. \tag{3.35}$$

We also have that

$$\left(\frac{\partial b}{\partial g} \right)_{ii} W_{ii} b_i[k] = W_{ii} b_i[k], \tag{3.36}$$

$$\left(\left(\frac{\partial b}{\partial f} \right)^{\mathrm{T}} \right)_{li} W_{ii} b_i[k] = \left(\frac{\partial b}{\partial f} \right)_{il} W_{ii} b_i[k] = -W_{ii} b_i[k], \tag{3.37}$$

and, for $l \neq i$*,*

$$\left(\left(\frac{\partial b}{\partial f} \right)^{\mathrm{T}} \right)_{il} W_{ll} b_j[k] = \left(\frac{\partial b}{\partial f} \right)_{li} W_{ll} b_l[k] = W_{ll} b_l[k]. \tag{3.38}$$

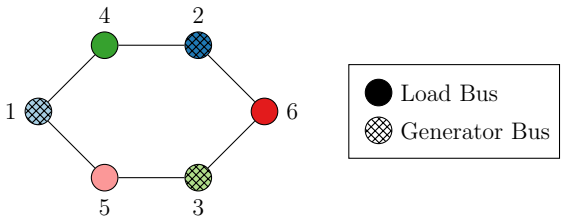


Figure 3.11: Graph-theoretic physical layer model, $\mathcal{V}_p$, for six-bus microgrid.

The iterations in (3.31) – (3.32) *can be written for each node* i *as follows:*

$$g_i[k+1] = [g_i[k] - sW_{ii}b_i[k])]_{\underline{g}_i}^{\overline{g}_i}, \tag{3.39}$$

$$f_{\{i,j\}}[k+1] = \left[f_{\{i,j\}}[k] + sW_{ii}b_i[k] - sW_{jj}b_j[k]\right]_{\underline{f}_{\{i,j\}}}^{\overline{f}_{\{i,j\}}}. \tag{3.40}$$

By choosing $W_{ii} = \frac{1}{\delta_d^-(i)+\delta_d^+(i)+1}$ *and* $s = 1/2$*, we obtain the updates used in the feasible flow algorithm.* □

A summary of the feasible flow algorithm including the above-described simplification is given in Algorithm 2. From this summary, we see that we can represent Algorithm 2 as a function that takes flow and generator output limits and load demands as inputs and yields flows and generator outputs that satisfy the feasible flow problem, i.e., the algorithm can be represented by

$$h^f : (\underline{f}, \overline{f}, \underline{g}, \overline{g}, \ell) \mapsto (f^*, g^*). \tag{3.41}$$

We illustrate the execution of Algorithm 2 in the following numerical simulation example.

Example 4 (Six-Node Feasible Flow Algorithm)
We demonstrate the operation of the algorithm we propose for solving the feasible flow problem using a six-node microgrid. A graph-theoretic model of the physical network, i.e., $\mathcal{G}_p$*, is illustrated in Fig. 3.11 whereas the mapping to the directed graph,* $\mathcal{G}_f$*, is illustrated in Fig. 3.12. The initial load values are given by* $\ell^0 = \left[\ell_4^0, \ell_5^0, \ell_6^0\right]^{\mathrm{T}} = \left[1.15, 1.25, 0.9\right]^{\mathrm{T}}$*, the generator limits are given by* $\underline{g} = \left[\underline{g}_1, \underline{g}_2, \underline{g}_3\right]^{\mathrm{T}} = \left[0.1, 0.15, 0.05\right]^{\mathrm{T}}$ *and* $\overline{g} = \left[\overline{g}_1, \overline{g}_2, \overline{g}_3\right]^{\mathrm{T}} =$

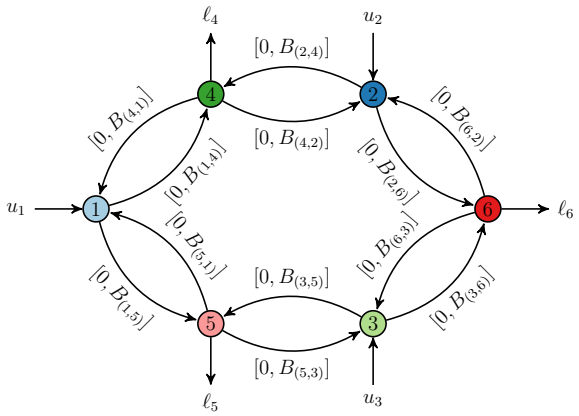


Figure 3.12: Graph-theoretic network flow model, $\mathcal{V}_d$, for six-bus microgrid.

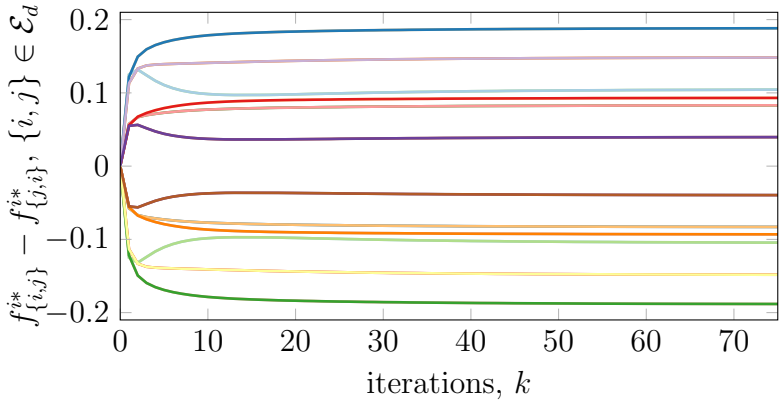


Figure 3.13: Evolution of flow estimates for six-node microgrid.

$\left[1.15, 2.65, 1.68\right]^{\mathrm{T}}$, *and the flow limits are given by* $\underline{f} = 0_{|\mathcal{E}_d|}$ *and*

$$\overline{f} = \left[\, B_{(4,1)}, B_{(1,4)}, B_{(1,5)}, B_{(5,1)}, B_{(2,4)}, B_{(4,2)}, B_{(3,5)}, B_{(5,3)}, B_{(2,6)}, B_{(6,2)}, B_{(3,6)}, B_{(6,3)} \,\right]^{\mathrm{T}},$$

where the susceptance values are given in Table B.3.

The evolution of the net out-flows, i.e., $\hat{f}^i_{\{i,j\}} - \hat{f}^i_{\{j,i\}}$, $\{i,j\} \in \mathcal{E}_d$, *for each node are plotted in Fig. 3.13 for the first 75 iterations of the algorithm's execution. Similarly, the evolution of the flow balances maintained by each node, i.e., the* $\hat{b}_i[k]$*'s, are shown in Fig. 3.14. From the figures, it can be seen that the algorithm converges quickly, with all of the flow balances quickly tending toward zero, and the net flows quickly approaching their final values.* ■

Remark 2

As with ratio consensus, the algorithm we propose to solve the feasible flow

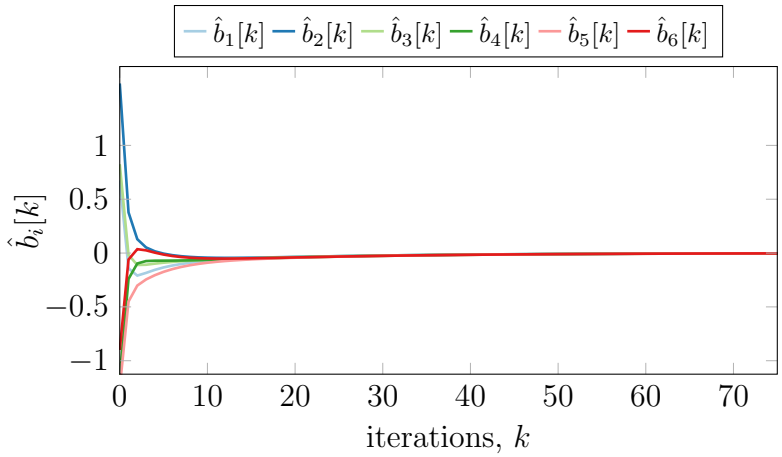


Figure 3.14: Evolution of nodal balances for six-node microgrid.

problem must extend over an infinite number of iterations. In reality, by stopping the iterative process after a finite number of iterations, which we denote by k_f, the worst-case error between the flow and generator output estimates and their true asymptotic values can be made arbitrarily small. Additionally, similar to the finite-time approach proposed in [36], the nodes can use max consensus, re-initialized every Δ_d iterations, to periodically compute the value of $\epsilon_b[\Delta_d k] := \max_{i\in\mathcal{V}_d} \hat{b}_i[\Delta_d k]$. Given that the values of the flow and generator output estimates depend on the flow balances, it may be possible to use this instance of max consensus to distributively determine the iteration at which all values of $\hat{f}^i_{\{i,j\}}[k]$, $\{i,j\} \in \mathcal{E}_d$ and $\hat{g}_i[k]$, $i \in \mathcal{V}_g$ are within some bound of their true asymptotic values. □

Remark 3

Although the communication graph modeling the exchange of information for Algorithm 2 must conform to the same topology as the underlying physical network, it is possible to use a slightly modified version of the ratio-consensus algorithm to eliminate this requirement. To achieve this, each local processor maintains and updates n ratios, one for each node in the network. More specifically, let $y^j_i[k]$ and $z^j_i[k]$ denote the numerator and denominator states corresponding to the ratio for node j maintained by node i. Suppose these states are initialized as $y^j_i[0] = \hat{b}_j[0]$ and $z^j_i[0] = w_j$ if $i = j$ and $y^j_i[0] = 0$

and $z_i^j[0] = 0$ *otherwise, and updated according to*

$$y_i^j[k+1] = \sum_{l \in \mathcal{N}_c^-(i) \cup \{i\}} \frac{1}{\delta_c^+(l)+1} y_l^j[k] + \Delta\hat{b}_j[k], \tag{3.42}$$

$$z_i^j[k+1] = \sum_{l \in \mathcal{N}_c^-(i) \cup \{i\}} \frac{1}{\delta_c^+(l)+1} z_l^j[k], \tag{3.43}$$

where

$$\Delta\hat{b}_j[k] = \begin{cases} \hat{b}_j[k] - \hat{b}_j[k-1], & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases} \tag{3.44}$$

Then, if $y_i^j[k]/z_i^j[k]$ *and* $y_i^l[k]/z_i^l[k]$ *are used by node* i *in place of* $\hat{b}_j[k]/w_j$ *and* $\hat{b}_l[k]/w_l$ *in* (3.26) *and* (3.27)*, respectively, it follows that the feasible flow algorithm can make use of the graph* $\mathcal{G}_c$ *for communication. Although we do not have a proof for this alternative to Algorithm 2, we have tested it using numerical simulations and have observed convergence for several cases.* □

Part II

Distributed Generation Control Architecture for Islanded AC Microgrids with Inertia

CHAPTER 4

INTRODUCTION AND PRELIMINARIES

We begin this chapter by providing a brief overview of the focus of the work presented in this part, and outline the organization of the remainder of this part. Then, we revisit the models presented in Chapter 2 and, finally, characterize the operating point at which every bus in a microgrid is synchronized.

4.1 Introduction

The focus of the work presented in this part is the development and implementation of a distributed control architecture for islanded microgrids that replicates the functionality of the frequency regulation and optimal dispatch control functions. In our setup, we consider ac systems comprised of synchronous generators and inverter-interfaced power supplies, collectively referred to as distributed generation resources (DGRs), and assume that each DGR is equipped with a local processor and a transceiver through which information can be exchanged, possibly unidirectionally, with neighboring DGRs. By relying on this directed communication network and on simple computations executed by the local processors, we provide two algorithms that allow the DGRs to determine their generation set-points in order to (i) regulate the system frequency and (ii) minimize total operational costs; like the control architectures from which ours is derived, both of these algorithms account for power output limits of the DGRs. Beyond introducing distributed alternatives to the top two generation control layers, a significant difference between our distributed control architecture and centralized counterparts used in bulk power systems is that ours is event triggered, i.e., generation set-points can be updated at non-fixed time instants, which eliminates unnecessary control efforts during periods in which the power demand remains relatively constant.

To demonstrate the effectiveness of our distributed generation control architecture, we tested it on a laboratory-grade microgrid. The electrical network of this microgrid comprises several small synchronous generators interconnected with resistive loads (inverter-interfaced power supplies are omitted due to lack of availability), whereas the cyber network for communication and control comprises Arduino microcontrollers outfitted with wireless transceivers. Using this microgrid, we experimentally verified the performance of the proposed control architecture under a variety of scenarios. Specifically, we provide results that illustrate the operation of the distributed frequency regulation and optimal dispatch functions following both an increase and a decrease in load. Additionally, we show how a DGR acting as spinning reserve can use the distributed frequency regulation function to independently determine if the collective power output limits of the online units is insufficient to balance the load, and act accordingly.

While the contributions outlined above describe the content discussed herein, they represent extensions of and improvements upon our preliminary work presented in [40, 39]. More specifically, we combine the frequency regulation and optimal dispatch algorithms proposed in the above papers into a complete distributed generation control architecture, outlining an event-triggered protocol which enables them to work together while accounting for microgrid-specific characteristics. Additionally, we provide a more thorough description of their operation and include some analysis on the frequency error dynamics of the closed-loop system. Finally, we build upon the experimental work presented in [40] to demonstrate both distributed generation control algorithms and the event-triggered protocol operating on a laboratory-grade microgrid.

The remainder of this part is organized as follows. In Section 4.2, we revisit the physical and cyber layer models introduced in Chapter 2 and characterize the operating point at which every bus in a microgrid with inertia is synchronized. In Chapter 5, we provide an overview of the desired control functions and outline specific control objectives. In Chapter 6, we describe the two algorithms that distributively implement the desired control functions. Chapter 7 describes a laboratory-grade microgrid that we developed for testing the performance of the proposed architecture. Experimental results obtained using the aforementioned microgrid are presented in Chapter 8.

4.2 Preliminaries

In this section, we briefly revisit the physical and cyber layer models introduced in Chapter 2, and provide some discussion on the interaction between the two layers. To account for a microgrid comprising generators with and without inertia, we slightly modify the notation introduced in Section 2.1.1 to include synchronous generators and inverter-interfaced generators. Additionally, we formally state an assumption that reduces the dynamical load model introduced in Chapter 2 to one that can represent constant power loads. Finally, we characterize the operating point at which every bus in the physical layer model is synchronized.

4.2.1 Physical and Cyber Layer Models

As in Chapter 2, we consider ac microgrids consisting of n buses. For the analysis presented in this part, we partition the generator buses, which we henceforth refer to as *DGR* buses, such that $|\mathcal{V}_s| =: p$ of them have synchronous generators attached to them, and $|\mathcal{V}_i| = m - p$ have inverter-interfaced power supplies attached to them. Without loss of generality, let $1, 2, \ldots, p$ index the DGR buses with synchronous generators; let $p+1, p+2, \ldots, m$ index the DGR buses with inverter-interfaced power supplies; and let $m+1, m+2, \ldots, n$ index the demand buses.

We represent the dynamics of each synchronous generator $i \in \mathcal{V}_s$ by the structure-preserving model in (2.1) – (2.3). For each DGR bus with an inverter-interfaced power supply, $i \in \mathcal{V}_i$, we represent the dynamics by the reduced generator model in (2.9). Unlike the microgrid model presented in Chapter 2, we assume that the demand buses can be modeled by constant power loads. More formally, from the load model in (2.4), we assume that $H_i \to 0$, $i \in \mathcal{V}_\ell$, such that the model representing the load buses is given by

$$0 = -P_i^d - V_i \sum_{k \in \mathcal{V}_p} V_k \left[G_{ik} \cos(\theta_i - \theta_k) + B_{ik} \sin(\theta_i - \theta_k) \right], \qquad (4.1)$$

where $P_i^d = \ell_i$ is the real power demand of load bus $i \in \mathcal{V}_\ell$

For the distributed control architecture proposed later in this part, we assume that each DGR is outfitted with a local processor that is capable of communicating with the local processors of other nearby DGRs, and that

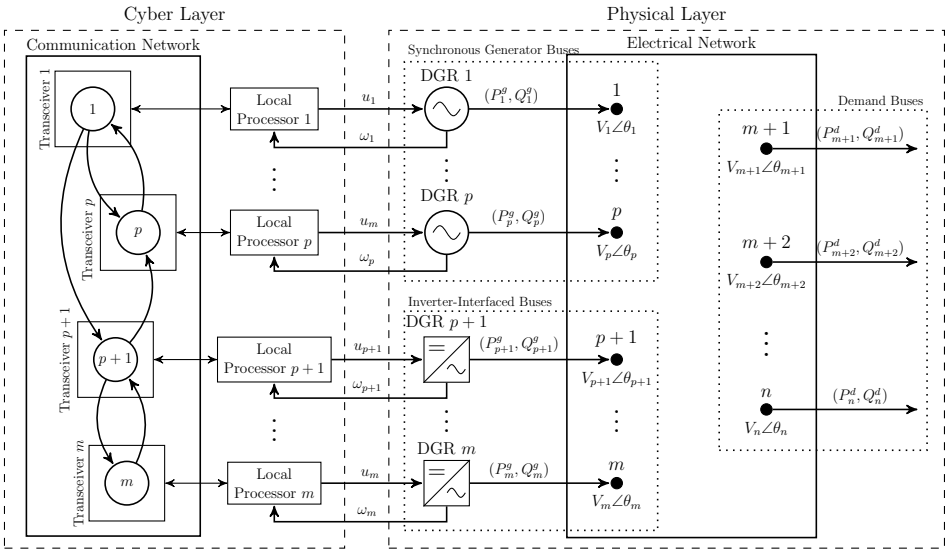


Figure 4.1: Distributed generation control architecture block diagram: on the right, the connectivity between DGRs and loads is not explicitly shown; on the left, the communication topology linking the different transceivers is not explicitly shown.

the network formed by these communication links can be modeled by the directed graph introduced in Section 2.2, i.e., $\mathcal{G}_c = \{\mathcal{V}_c, \mathcal{E}_c\}$. In this part, we only require that each DGR bus be equipped with a local processor; thus, $\mathcal{V}_c = \mathcal{V}_s \cup \mathcal{V}_i$ such that $\mathcal{V}_c \cap \mathcal{V}_\ell = \emptyset$. The block diagram in Fig. 4.1 describes the interactions between the physical layer used to model an ac microgrid and cyber layer used to model the distributed control architecture we propose later in this part. The components illustrated and the notation used in this diagram are introduced throughout the remainder of Part II.

4.2.2 DGR Synchronization

Let $u_i^r \in \mathbb{R}$ denote a constant set-point applied to DGR $i \in \mathcal{V}_g$, between time instants t_r and t_{r+1}, i.e., $u_i(t) = u_i^r$, $t_r \leq t < t_{r+1}$, and assume that the real power load at demand bus i, P_i^d, $i \in \mathcal{V}_\ell$, is constant. Under synchronous operation, all DGRs will operate at a constant common electrical frequency, ω^r, i.e., $\omega_i = \omega^r$, $i = \mathcal{V}_g$, which can be obtained as follows. In (2.2) – (2.3) and (2.9), set the left-hand side to zero; then, together with (4.1), and the

fact that $\omega_i = \omega^r,\ i \in \mathcal{V}_g$, we have

$$\begin{aligned}
0 =& - D_i(\omega^r - \omega_0) + P_i^m \\
& - V_i \sum_{k \in \mathcal{V}_p} V_k \left[G_{ik} \cos(\theta_i - \theta_k) + B_{ik} \sin(\theta_i - \theta_k) \right], \quad i \in \mathcal{V}_s, \qquad (4.2) \\
0 =& - P_i^m - \frac{1}{R_i \omega_0}(\omega^r - \omega_0) + u_i^r, \quad i \in \mathcal{V}_s, \qquad (4.3) \\
0 =& - H_i(\omega^r - \omega_0) + u_i^r \\
& - V_i \sum_{k \in \mathcal{V}_p} V_k \left[G_{ik} \cos(\theta_i - \theta_k) + B_{ik} \sin(\theta_i - \theta_k) \right], \quad i \in \mathcal{V}_i, \qquad (4.4) \\
0 =& -P_i^d - V_i \sum_{k \in \mathcal{V}_p} V_k \left[G_{ik} \cos(\theta_i - \theta_k) + B_{ik} \sin(\theta_i - \theta_k) \right], \quad i \in \mathcal{V}_\ell. \qquad (4.5)
\end{aligned}$$

Assume that between time instants t_r and t_{r+1} there exists $\theta_i,\ i \in \mathcal{V}_p$, and $V_i,\ i \in \mathcal{V}_p$, that satisfy (4.2) – (4.5) which we denote by θ_i^r, and V_i^r, respectively; this is essentially equivalent to the existence of a solution to the network power flow equations [42]. Then, by summing (4.2) – (4.5) over all i, it is easy to see that all the sine terms cancel out, which leads to

$$\omega_r = \omega_0 + \frac{\sum_{i \in \mathcal{V}_g} u_i^r + \sum_{i = \in \mathcal{V}_\ell} P_i^d - P_l^r}{\sum_{i \in \mathcal{V}_s}(D_i + 1/R_i\omega_0) + \sum_{i \in \mathcal{V}_i} H_i}, \qquad (4.6)$$

where $P_l^r = \sum_{i \in \mathcal{V}_p} \sum_{k \in \mathcal{V}_p} V_i^r V_k^r G_{ik} \cos(\theta_i^r - \theta_k^r)$. From (4.6) it follows that

$$\Delta\omega^r := \omega^r - \omega_0 = \frac{\sum_{i \in \mathcal{V}_g} u_i^r + \sum_{i = \in \mathcal{V}_\ell} P_i^d - P_l^r}{\sum_{i \in \mathcal{V}_s}(D_i + 1/R_i\omega_0) + \sum_{i \in \mathcal{V}_i} H_i}, \qquad (4.7)$$

which, in the remainder of this part, we will refer to as *frequency error*. As we will discuss later, one of the control objectives will be to adjust the u_i^r's in a distributed fashion so as to drive $\Delta\omega^r$ to zero.

Remark 4

It is important to note that we have not addressed questions of existence, uniqueness, and stability of synchronized solutions to (2.1) – (2.3) *and* (4.1) *as these are topics that have been well studied in the literature (see, e.g., [52, 53, 41] and the references therein), and their rigorous analysis is outside the scope of this work. However, the result in* (10.1) *is sufficient for the analysis in Section 6.3.* □

CHAPTER 5

CONTROL OBJECTIVES AND PROTOCOL

We begin this chapter by providing an overview of the control objectives to be met by our distributed architecture for generation control. These objectives are similar to those sought by centralized architectures for generation control used in bulk power systems, but they are achieved through a distributed computation over the communication network interconnecting the DGRs. This computation relies on two algorithms—their formal definition is deferred to Chapter 6—that enable the distributed implementation of the frequency regulation and optimal dispatch functions. We then specify a protocol that allows the DGRs to determine when to utilize each of the two aforementioned algorithms; this protocol is based on purely local measurements and as such is completely distributed.

5.1 Overview of Control Functions and Objectives

Despite being smaller and having lower ratings, the generation control objectives for islanded ac microgrids are similar to those for large power systems. Therefore, the control functions of our distributed architecture are derived from the functionality provided by the typical three-layer generation control architecture used in bulk power transmission networks. In the discussion that follows, we briefly summarize the control functions of each of these three layers, highlighting the aspects that are relevant to the realization of our distributed architecture.

5.1.1 Droop Control

The purpose of this control function is to instantaneously balance generation with demand through local actions taken by the speed governors of each

generation unit; in the synchronous generator model in (2.1) – (2.3), for example, the governor dynamic behavior is described by (2.3). Typically, the low-level control mechanism that provides this functionality, often referred to as *primary generation control* or *droop control*, is designed to allow instantaneous load changes to be divided proportionally with respect to generator ratings [54]. Regardless of power system size, droop control requires only local measurements and actuations and is inherently decentralized; thus, there is no need for a distributed alternative. As a result, in the remainder of this part, we omit further discussion on droop control other than to facilitate discussion of other control functions.

5.1.2 Frequency Regulation

While primary generation control provides a simple and effective way to compensate for changes in load, it does so at the expense of frequency deviations. If not eliminated, these variations may result in equipment damage or otherwise undesirable operation, e.g., induction motors and their connected loads rotating at speeds for which they are not designed. In large power systems, there is a control function, commonly referred to as *secondary generation control* or *automatic generation control* (AGC), that is responsible for returning the frequency to its nominal value, e.g., 60 Hz, following transients induced by load changes. In addition to regulating frequency, secondary generation control is also responsible for maintaining proper interchange power values between control areas. However, in the context of microgrids, there is no notion of control areas; thus, in our distributed architecture, the only objective of the secondary control function is *frequency regulation*.

To compensate for frequency deviations arising from changes in load and the subsequent response of droop control, the secondary generation control in large-scale power systems is commonly implemented using a closed-loop controller that adjusts the set-point of each generator based upon the integral of the frequency error (see, e.g., [54, 13]). The implementation of AGC is centralized with measurements and control signals telemetered to and from the generating units and the control center where the computer implementing the controller resides.

In Section 6.1, we will see that, when executed over several rounds and

properly initialized according to an estimate of the incremental demand for generation, the ratio-consensus algorithm can be used to replicate the functionality of the aforementioned centralized closed-loop controller for large power system AGC.

5.1.3 Optimal Dispatch

Although primary and secondary generation controls suffice to balance generation with demand and subsequently regulate frequency, they actuate without accounting for operational costs. Consequently, in a large power system, there is a third generation control function which seeks to determine the most economical way to allocate generation demand among all generators. That is, assuming the cost of each generator is a function of its power output, the objective of this tertiary control is to minimize the total generation cost subject to individual power output limits (see, e.g., [55]).

For large power systems, similar to secondary generation control, the optimal dispatch function relies on a decision-making algorithm executed on a computer that resides in a centralized location, e.g., a control center, with knowledge of the cost function and limits of each generator as well as the sum of the power output of all the generators. In Section 6.2, we will discuss an algorithm based on ratio consensus that is suitable for distributively implementing the optimal dispatch function.

5.2 Control Mode Protocol

Large power systems are vast, often containing thousands of loads that vary frequently; consequently, the actions of droop control result in near-constant frequency deviations. Furthermore, given that measurements and control signals must be telemetered to and from a central location, secondary and tertiary generation control in bulk power systems do not run continuously, relying instead on periodically triggered actuations performed approximately every two-to-five seconds and five minutes, respectively [13].

In the context of the class of microgrids considered in this work, and in order to motivate the control mode protocol proposed in this section, we make the following assumptions.

[**A1.**] There are fewer loads, each of which may be large relative to the total capacity of the DGRs.

[**A2.**] The collective inertia provided by certain types of DGRs (e.g., synchronous generators) may be small.

[**A3.**] Large frequency deviations may result following a load change.

Given assumptions [**A1**] – [**A3**], while minimizing cost is important, the main control objective in islanded microgrids is to regulate frequency [19]. As such, rather than operating on a fixed time interval, we adopt an event-triggered control architecture which only actuates when the frequency error exceeds a specified bound, and does not optimally dispatch the DGRs until the frequency is sufficiently close to the nominal value. In the discussion that follows, we specify the protocol utilized by the DGR local processors to determine the appropriate control mode, and introduce notation that will be used in subsequent developments.

Remark 5
While assumptions [**A1**] – [**A3**] *motivate our control mode protocol for microgrids, if they do not hold, scenarios may result in which the optimal dispatch algorithm is never executed. In particular, it is unlikely that these assumptions will be valid in, for example, bulk power transmission networks with a large number of frequently varying loads. Moreover, as noted in Section 3.1.2, the distributed algorithm on which our control relies may not scale well for systems with a large number of generation units.* □

To return the frequency to its nominal value following one or more load changes, and to subsequently dispatch the DGRs optimally, our architecture relies on the execution of two distributed algorithms (to be described in Chapter 6) over several rounds. Let r index the rounds over which the distributed algorithms are executed, and denote the generation set-point of DGR i at round r by u_i^r. Then, the relation between the generation set-point of DGR i between two consecutive rounds, r and $r+1$, is given by

$$u_i^{r+1} = u_i^r + \Delta u_i^r, \tag{5.1}$$

where Δu_i^r is the amount by which DGR i should adjust its output at round r. In subsequent developments, we assume that after the u_i^r's are set at each

round r, sufficient time elapses such that a new synchronized operating point is reached wherein $\omega_1^r = \omega_2^r = \cdots = \omega_m^r =: \omega^r$ (see analysis in Section 4.2.2) before each DGR computes Δu_i^r and adjusts its set-point according to (5.1). Furthermore, if a feasible solution exists at each round, i.e., $\sum_{i\in\mathcal{V}_g} \underline{P}_i \leq \sum_{i\in\mathcal{V}_g} u_i^r \leq \sum_{i\in\mathcal{V}_g} \overline{P}_i$, where $\underline{P}_i$ and $\overline{P}_i$ denote the lower and upper limits on the power output of DGR i, respectively, the algorithms developed in the next section will ensure that $\underline{P}_i \leq u_i^r \leq \overline{P}_i,\ i \in \mathcal{V}_g$.

Let $\epsilon > 0$ denote the maximum allowable electrical frequency[1] error, let ξ_i^r be an indicator for the optimal dispatch function for DGR i, i.e.,

$$\xi_i^r = \begin{cases} 0, & \text{if DGR } i \text{ optimally dispatched since last freq. reg.,} \\ 1, & \text{otherwise,} \end{cases} \tag{5.2}$$

and define $u^r = [u_1^r, \ldots, u_m^r]^{\mathrm{T}}$. Then, each DGR i determines the value of Δu_i^r at round r as follows:

$$\begin{aligned} \Delta u_i^r &= h_i(u_i^r, \psi^r) \\ &:= \begin{cases} h_i^f(u_i^r, \psi^r), & \text{if } |\omega^r - \omega_0| > \epsilon, \\ h_i^o(u_i^r, \psi^r), & \text{if } |\omega^r - \omega_0| \leq \epsilon \text{ and } \xi_i^r = 1, \\ 0, & \text{otherwise,} \end{cases} \end{aligned} \tag{5.3}$$

for some function $h_i^f : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ to be defined in Section 6.1, and some function $h_i^0 : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ to be defined in Section 6.2; and where the input ψ^r is given by

$$\psi^r = \begin{cases} \phi^f(\omega^r, u^r), & \text{if } |\omega^r - \omega_0| > \epsilon, \\ \phi^o(u^r), & \text{if } |\omega^r - \omega_0| \leq \epsilon, \end{cases} \tag{5.4}$$

for some functions $\phi^f : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ and $\phi^o : \mathbb{R}^n \mapsto \mathbb{R}$ to be defined in Section 6.1 and Section 6.2, respectively.

Given the protocol specified in (5.2) – (5.4), the timeline shown in Fig. 5.1 outlines the relative timescales over which the frequency regulation and optimal dispatch algorithms are executed. If we let s index the round during

[1]We assume that, aside from a proportionality constant, the electrical angular speed of DGR i, ω_i, is equivalent to the electrical frequency of the bus to which it is connected, and thus use the terms interchangeably.

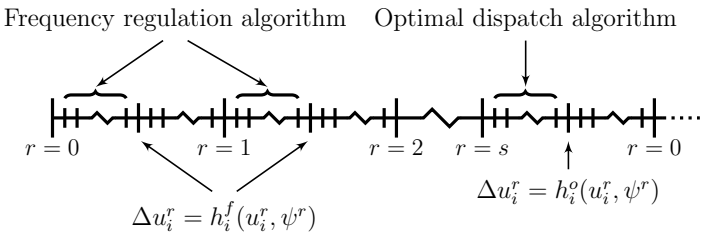


Figure 5.1: Timeline of distributed generation control algorithms.

which the DGRs are optimally dispatched, i.e., $r = s$ is the round for which $|\omega_i^r - \omega_0| \leq \epsilon$ and $\xi_i^r = 1$, the figure illustrates that, for $r < s$, the magnitude of the frequency error exceeds the specified bound and the frequency regulation algorithm is executed. In particular, at rounds $r = 0, 1, \ldots, s-1$, the DGRs adjust their generation set-points using $h_i^f(\cdot)$, the collective effect of which replicates the functionality of a discrete-time integral controller. At $r = s$, the magnitude of the frequency error has been reduced to within the specified bound and the DGR generation set-points are modified according to $h_i^o(\cdot)$, which yields the optimal generation allocation among the DGRs. Regardless of the algorithm used, after the set-points are adjusted, the system evolves according to (2.1) – (2.3) and (2.9).

From (5.3), we see that in order to determine the incremental set-point of each DGR, the *global* information captured in the value of ψ^r is needed; this value is determined by the output of the functions $\phi^f(\cdot)$ and $\phi^o(\cdot)$ in (5.4) for frequency regulation and optimal dispatch, respectively. Note also that, in order to evaluate these functions, it is necessary to have the set-points and electrical angular speeds of all the DGRs in the system. Additionally, as we will see in Chapter 6, $\phi^f(\cdot)$ and $\phi^o(\cdot)$ both depend on the minimum and maximum output power of all the DGRs in the system, while $\phi^o(\cdot)$ additionally requires the power output cost function parameters for each DGR. Although this implies that in order to evaluate (5.4), and in turn (5.3), it is necessary to pass all the above information to a centralized processor, in Chapter 6 we will explain how $\phi_i^f(\cdot)$ and $\phi_i^o(\cdot)$ can be evaluated using a distributed computation which relies on the ratio-consensus algorithm discussed in Section 3.1.2.

CHAPTER 6

CONTROL ALGORITHMS

In this chapter, we describe the two algorithms that serve to distributively compute functions $h_i^f(\cdot)$ and $h_i^o(\cdot)$ in (5.3), thus enabling the implementation of frequency regulation, which we outline in Section 6.1, and optimal dispatch without the need for a centralized processor, which we outline in Section 6.2. Additionally, we analyze the error dynamics of our proposed distributed control architecture during frequency regulation and after the generators have been optimally dispatched and provide criteria for choosing gains that stabilize the frequency.

6.1 Distributed Frequency Regulation

Consider a microgrid consisting of n DGRs outfitted with local processors, the interconnections of which can be represented by a strongly connected directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Assume that all the DGRs are synchronized to a common angular speed ω^r (not necessarily ω_0) determined by the values of the current DGR set-points, i.e., the u_i^r's (see analysis in Section 4.2.2). If we assume that each of the local processors is capable of obtaining a measurement of the speed of the DGR to which it is connected,[1] an estimate for the amount by which DGR i should adjust its generation set-point to drive the electrical frequency to the nominal value at round r is

$$\Delta\hat{u}_i^r = \kappa_i(\omega^r - \omega_0) =: \kappa_i \Delta\omega^r, \tag{6.1}$$

[1]While a speed measurement is required for primary generation control, we explicitly state this assumption to account for systems in which no droop controller is present, or the primary and secondary control functions are performed by separate processors and sensors.

where $\kappa_i < 0$ is some gain chosen by the local processor (in Section 6.3, we give some guidelines on how each DGR can choose this gain).

Given the common electrical frequency at which the DGRs operate, and with no power output constraints, a simple solution to the frequency regulation problem is for each DGR to adjust its set-point at each round r as in (6.1) until the frequency error $\Delta\omega^r$ vanishes. However, if constraints due to DGR power ratings or operational limitations are considered, this solution may not be feasible. Furthermore, it is well known that, if the DGRs do not operate at a common electrical frequency, applying independent control to each DGR may result in an unstable system (this is commonly referred to as isochronous governor control, see, e.g., [13, Section 9.5]). Thus, in order to address these problems, we use the ratio-consensus algorithm, with appropriately chosen initial conditions that account for DGR power output limits, to divide the estimated incremental generation demand among all DGRs.

Define $\Delta\underline{P}_i^r := \underline{P}_i - u_i^r$, and $\Delta\overline{P}_i^r := \overline{P}_i - u_i^r$ to be the incremental minimum and maximum power output of DGR i at round r, respectively. As mentioned previously, the characteristics of the loads and DGRs in islanded microgrids can lead to large frequency deviations; thus, we adopt an allocation scheme for the distributed frequency regulation algorithm whereby the incremental set-point of each DGR is adjusted proportionally to $\Delta\overline{P}_i^r - \Delta\underline{P}_i^r$ in order to eliminate the frequency error in as few rounds as possible. Furthermore, to ensure a feasible solution, we assume that, at each round r, the sum of the estimated incremental set-points lies within the collective incremental power output limits of the DGRs, i.e.,

$$\sum_{i\in\mathcal{V}_g} \Delta\underline{P}_i^r \leq \sum_{i\in\mathcal{V}_g} \Delta\hat{u}_i^r \leq \sum_{i\in\mathcal{V}_g} \Delta\overline{P}_i^r. \tag{6.2}$$

Then, the ratio consensus states in (3.4) and (3.5) are initialized to $y_i[0] = \Delta\hat{u}_i^r - \Delta\underline{P}_i^r$ and $z_i[0] = \Delta\overline{P}_i^r - \Delta\underline{P}_i^r$, $\forall i \in \mathcal{V}_c$, respectively. Given these initial

conditions, it follows from Lemma 1 that DGR i can asymptotically obtain

$$\begin{aligned}
\psi^r &= \lim_{k\to\infty} \gamma_i^r[k] = \lim_{k\to\infty} \frac{y_i^r[k]}{z_i^r[k]}, \\
&= \frac{\sum_{j\in\mathcal{V}_g} \kappa_j(\omega^r - \omega_0) - \sum_{j\in\mathcal{V}_g} \Delta\underline{P}_j^r}{\sum_{j\in\mathcal{V}_g} \left(\Delta\overline{P}_j^r - \Delta\underline{P}_j^r\right)}, \\
&= \frac{\sum_{j\in\mathcal{V}_g} \kappa_j(\omega^r - \omega_0) - \sum_{j\in\mathcal{V}_g} (\underline{P}_j - u_j^r)}{\sum_{j\in\mathcal{V}_g} \left(\overline{P}_j - \underline{P}_j\right)}, \\
&=: \phi^f(\omega^r, u^r),
\end{aligned} \tag{6.3}$$

which represents the magnitude of the sum of the estimated set-point adjustments as a percentage of the collective incremental power output limits of the DGRs.

After the ratio-consensus algorithm has converged and each DGR has obtained ψ^r, the incremental set-point of DGR i at round r is given as

$$\begin{aligned}
\Delta u_i^r &= \Delta\underline{P}_i^r + \psi^r \left(\Delta\overline{P}_i^r - \Delta\underline{P}_i^r\right), \\
&= \underline{P}_i - u_i^r + \psi^r \left(\overline{P}_i - \underline{P}_i\right), \\
&=: h_i^f \left(u_i^r, \psi^r\right),
\end{aligned} \tag{6.4}$$

and DGR i adjusts its reference command according to (5.1). Given this choice of initial conditions and the definition of $h_i^f(\cdot)$, we refer to the allocation scheme described in (6.1) – (6.4) as *fair splitting*.

In practice, the use of ratio consensus for obtaining ψ^r in (6.4) for frequency regulation will not extend over an infinite number of iterations but will only be executed for a finite number of iterations, k_f. A finite number of iterations implies a $\gamma_i[k_f]$ in (3.6) that is not necessarily the same for all nodes i, but can be bounded away from the true (limiting) value of ψ_r in (6.3) by a constant, ϵ_f. Moreover, the number of iterations, k_f, can be chosen, based upon the second-largest in magnitude eigenvalue of the weight matrix $P = [P_{ij}]$ (with $[P_{ij}] = \frac{1}{\delta_c^+(i)+1}$ when $\{i,j\} \in \mathcal{E}_c$ and $[P_{ij}] = 0$ otherwise) and the initial conditions of the ratio consensus algorithm, to make ϵ_f arbitrarily small. We omit a detailed discussion of this choice because it pertains to the convergence properties of the ratio consensus algorithm, which is not the focus of this work, and also because it was not an issue in our experiments. Next, we provide an example that illustrates the operation of the fair-splitting scheme

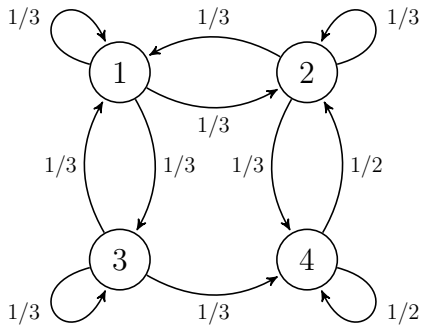


Figure 6.1: Graph of 4-node network.

and the choice of k_f.

Example 5 (Fair Splitting Feasible Solution)
Consider the 4-node network in Fig. 6.1, where each node represents the local processor of a DGR participating in the fair-splitting algorithm. Omitting the index of the round, let $\Delta\hat{u} = \Big[0.25, 0.4, -0.35, -0.2\Big]^{\mathrm{T}}$, *and let the incremental minimum and maximum power outputs of the DGRs be given, respectively, by*

$$\Delta\underline{P} = \Big[-0.1, -0.1, -0.2, -0.1\Big]^{\mathrm{T}},$$
$$\Delta\overline{P} = \Big[0.05, 0.05, 0.05, 0.05\Big]^{\mathrm{T}},$$

such that $\sum_{i=1}^{4} \Delta\underline{P}_i \leq \sum_{i=1}^{4} \Delta\hat{u}_i \leq \sum_{i=1}^{4} \Delta\overline{P}_i$. *Then, the initial values of* y *and* z *are set to*

$$y[0] = \Big[0.35, 0.5, -0.15, -0.1\Big]^{\mathrm{T}},$$
$$z[0] = \Big[0.15, 0.15, 0.25, 0.15\Big]^{\mathrm{T}},$$

respectively.

Given the above initial conditions, the evolution of $\psi_i[k]$, $i = 1, 2, 3, 4$, *over 25 iterations is shown in Fig. 6.2. From the figure, we see that after approximately 15 iterations, all nodes converge to a solution in which* $\psi = 0.86$. *Thus, the nodes determine the solution is feasible (see the discussion below) and compute the incremental generation set-point according to* (6.4), *and we have that* $\Delta u = \Big[0.029, 0.029, 0.015, 0.029\Big]^{\mathrm{T}}$. ■

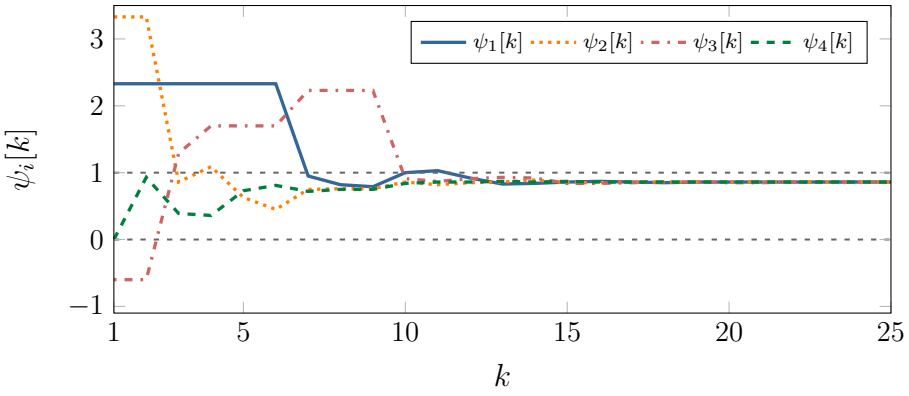


Figure 6.2: Fair-splitting algorithm evolution with feasible solution.

If we assign an objective function that takes value zero when the feasible set defined by the inequality constraints in (6.2) is nonempty or ∞ when the feasible set is empty, the proposed distributed approach to frequency regulation is equivalent to solving a feasibility problem (see, e.g., [56]). In particular, the value of ψ^r found using the fair-splitting procedure can be utilized by each DGR to independently determine if the sum of the incremental set-points is within the collective power output limits of the available DGRs. Specifically, at round r, if $\psi^r < 0$ or $\psi^r > 1$, each DGR knows that the collective bounds on power output have been exceeded; we illustrate this idea next.

Example 6 (Fair Splitting Infeasible Solution)
Consider the 4-node network in Fig. 6.1, where each node represents the local processor of a DGR participating in the fair-splitting algorithm. Omitting the index of the round, let $\Delta\hat{u} = \Big[0.3, 0.45, -0.2, -0.15,\Big]^{\mathrm{T}}$, *and let the incremental minimum and maximum power outputs of the nodes be given respectively as*

$$\Delta\underline{P} = \Big[-0.1, -0.05, -0.1, -0.05\Big]^{\mathrm{T}},$$
$$\Delta\overline{P} = \Big[0.05, 0.1, 0.1, 0.05\Big]^{\mathrm{T}},$$

such that $\sum_{i=1}^{4} \Delta\underline{P}_i = -0.3$ *and* $\sum_{i=1}^{4} \Delta\overline{P}_i = 0.3 < \sum_{i=1}^{4} \Delta\hat{u}_i = 0.4$. *Then,*

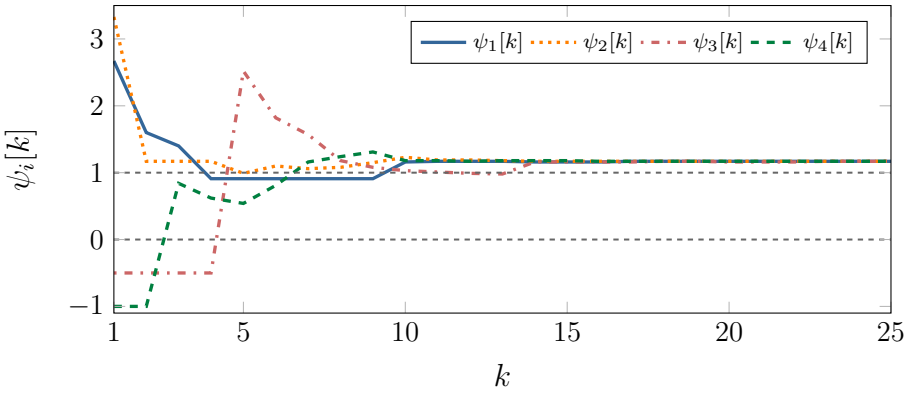


Figure 6.3: Fair-splitting algorithm evolution with infeasible solution.

we have that

$$y[0] = \Big[0.4, 0.5, -0.1, -0.1\Big]^{\mathrm{T}},$$
$$z[0] = \Big[0.15, 0.15, 0.2, 0.1\Big]^{\mathrm{T}}.$$

The evolution of $\psi_i[k]$, $i = 1, 2, 3, 4$, *over 25 iterations is shown in Fig. 6.3. From the figure, we see that after approximately 15 iterations, all nodes converge to a solution in which* $\psi = 1.17$. *Thus, the nodes determine the solution is infeasible (as they cannot adjust their output beyond their maximum power output). In Chapter 8, we show how the global information available through* ψ *can be used to, e.g., trigger reserve DGRs to come online.* ■

Remark 6

If a DGR i *does not participate in frequency regulation, it will set its* u_i *to some constant value* u_i^0*; however, this does not preclude DGR* i *from participating in the distributed algorithm for frequency regulation, i.e., the computation and communication aspects of the distributed protocol. To this end, the DGR only needs to set* $\underline{P}_i = \overline{P}_i = u_i$*; thus, it will initialize its* z_i *to* $z_i[0] = \overline{P}_i - \underline{P}_i = 0$*, and after the algorithm is executed, like all other DGRs, it will compute* $\Delta u_i^r = \Delta\underline{P}_i^r + \psi^r(\Delta\overline{P}_i^r - \Delta\underline{P}_i^r) = \underline{P}_i - u_i^r + \psi^r(\overline{P}_i - \underline{P}_i) = 0$ *such that* $u_i^{r+1} = u_i^r + \Delta u_i^r = u_i^r = u_i^0$. *Note that while Lemma 1 requires* $z_i[0] > 0$, $\forall i$, *in reality, only one* $z_i[0]$ *must be strictly positive while the others could be equal to zero. In practice, the nodes compute a close approximation to* ψ^r *after a sufficiently large number of iterations,* k_f. *Given that the graph is strongly connected,* $z_i[k_f]$ *is guaranteed to be bounded away from zero, which implies finite* ψ^r. □

Remark 7

The fair-splitting allocation scheme is not the only viable choice for distributively regulating frequency using the ratio-consensus algorithm. Other options include allocating incremental set-points based upon: (i) the response time of the DGRs such that the estimated incremental demand is divided proportionally to, for example, the ramp rates of the DGRs or the time constant of the governors; or (ii) the proximity of each DGR to its stability limit. We leave investigation of such alternatives to future work. □

6.2 Distributed Optimal Dispatch

Assuming a feasible solution exists, the execution of several rounds of the frequency regulation scheme described previously will make the absolute value of the frequency error, $\Delta\omega^r$, smaller than some bound, ϵ. Let $r = s$ be the round at which this event occurs; then, from (10.1) it follows that the u_i^s's must be such that

$$\sum_{i\in\mathcal{V}_g} u_i^s = \sum_{i=\in\mathcal{V}_\ell} P_i^d + \Delta\omega^s\big(\sum_{i\in\mathcal{V}_s}(D_i + 1/R_i\omega_0) + \sum_{i\in\mathcal{V}_i} H_i\big) := \chi.$$

As described previously, the fair-splitting procedure adjusts the incremental set-points of the DGRs proportionally with respect to incremental power output limits; thus, the u_i^s's might not necessarily be optimal, i.e., they might not minimize the overall cost of operation. Therefore, we would like to have a different generation allocation, $u_i^{s*},\ i \in \mathcal{V}_g$, among the DGRs while: (i) still satisfying their individual power output constraints, i.e., $\underline{P}_i \le u_i^{s*} \le \overline{P}_i,\ i \in \mathcal{V}_g$, and (ii) keeping their sum as before, i.e., $\sum_{i\in\mathcal{V}_g} u_i^{s*} = \sum_{i\in\mathcal{V}_g} u_i^s =: \chi$ (as we will show in Section 6.3, this will ensure that $\Delta\omega^r$ remains within ϵ). That is, assuming that the cost associated with each DGR is quadratic, we would like to find the global solution, which we denote by $u_i^{s*},\ i \in \mathcal{V}_g$, of the

following constrained optimization:

$$\begin{aligned} \underset{u_1,u_2,\ldots,u_n}{\text{minimize}} \quad & \sum_{i\in\mathcal{V}_g} \frac{(u_i-\alpha_i)^2}{2\beta_i} = \sum_{i\in\mathcal{V}_g} c_i(u_i) \\ \text{subject to} \quad & \sum_{i\in\mathcal{V}_g} u_i = \chi \\ & 0 \leq \underline{P}_i \leq u_i \leq \overline{P}_i,\ i \in \mathcal{V}_g, \end{aligned} \tag{6.5}$$

where $\alpha_i \leq 0$, $\beta_i > 0$; this problem is commonly referred to as *optimal dispatch* [13]. The constants α_i and β_i are a characteristic of the primary source of energy for DGR i; in the case of a synchronous generator DGR, if, e.g., the prime mover is a gas turbine, these constants are determined by the heat-rate curve (see, e.g., [55] for a detailed discussion). For an inverter-interfaced DGR, if, e.g., the primary energy source is an array of photovoltaic panels, α_i represents the maximum power point (MPP), while β_i penalizes deviations from the MPP.

6.2.1 A Centralized Solution to the Optimal Dispatch Problem

The problem of optimally dispatching DGRs at round s given in (6.5) is convex and has a separable structure [57, pp. 502-506]; thus, its solution can be found by solving the Lagrange dual (see, e.g., [57, 56]), which is given by

$$\text{maximize} \quad \lambda\chi + \sum_{i=1}^{m} f_i(\lambda), \tag{6.6}$$

where $f_i(\lambda),\ \forall i$, are

$$f_i(\lambda) = \begin{cases} \frac{(\underline{P}_i-\alpha_i)^2}{2\beta_i} - \lambda\underline{P}_i, & \lambda < \underline{\lambda}_i, \\ -\lambda(\alpha_i + \lambda\frac{\beta_i}{2}), & \underline{\lambda}_i \leq \lambda \leq \overline{\lambda}_i, \\ \frac{(\overline{P}_i-\alpha_i)^2}{2\beta_i} - \lambda\overline{P}_i, & \overline{\lambda}_i < \lambda, \end{cases} \tag{6.7}$$

with $\underline{\lambda}_i = \frac{\underline{P}_i-\alpha_i}{\beta_i}$ and $\overline{\lambda}_i = \frac{\overline{P}_i-\alpha_i}{\beta_i}$.

The Lagrange dual problem defined by (6.6) is convex and, in this case, provides the optimal solution to the primal problem. Furthermore, the cost

function in (6.7) is continuously differentiable, i.e., $f_i(\cdot) \in \mathcal{C}^1,\ \forall i$; thus, if the dual problem is feasible, the optimal solution λ^* must satisfy

$$\chi - \sum_{i=1}^{m} g_i(\lambda^*) = 0, \tag{6.8}$$

where $g_i(\lambda) = -\frac{df_i(\lambda)}{d\lambda}$, i.e.,

$$g_i(\lambda) = \begin{cases} \underline{P}_i, & \lambda < \underline{\lambda}_i, \\ \alpha_i + \lambda\beta_i, & \underline{\lambda}_i \leq \lambda \leq \overline{\lambda}_i, \\ \overline{P}_i, & \overline{\lambda}_i < \lambda. \end{cases} \tag{6.9}$$

Now, obtaining the value λ^* that satisfies (6.8) is equivalent to finding the intersection point of the functions $g(\lambda) := \sum_{i=1}^{m} g_i(\lambda)$ and $h(\lambda) := \chi$; this can be accomplished by evaluating the function $g(\cdot)$ for (at most) $2m$ values corresponding to the elements in $\mathcal{U} := \left\{\underline{\lambda}_1, \overline{\lambda}_2, \ldots, \underline{\lambda}_m, \overline{\lambda}_m\right\}$. To this end, define $\mathcal{U}^+ := \{\lambda \in \mathcal{U} : g(\lambda) \geq \chi\}$, and $\mathcal{U}^- := \{\lambda \in \mathcal{U} : g(\lambda) \leq \chi\}$; then, as proposed in [58], the value of λ^* is given by

$$\begin{aligned} \lambda^* &= \lambda^+ - \left(g(\lambda^+) - \chi\right) \frac{\lambda^+ - \lambda^-}{g(\lambda^+) - g(\lambda^-)}, \\ &=: \phi^o(u^s), \end{aligned} \tag{6.10}$$

where

$$\lambda^+ = \underset{\lambda \in \mathcal{U}^+}{\operatorname{argmin}} \, |g(\lambda) - \chi|, \tag{6.11}$$

$$\lambda^- = \underset{\lambda \in \mathcal{U}^-}{\operatorname{argmin}} \, |g(\lambda) - \chi|, \tag{6.12}$$

and the solution to the primal problem given in (6.5) is

$$u_i^{s*} = g_i(\lambda^*),\ i = 1, 2, \ldots, m. \tag{6.13}$$

From (6.10), it is easy to see that λ^* is obtained by a linear interpolation between the values of λ^- and λ^+; for this reason, we refer to the procedure outlined in (6.10) – (6.13) as the *interpolation method*.

6.2.2 A Distributed Implementation of the Interpolation Method

Although (6.13) provides the unique global solution to the optimal dispatch problem by solving its Lagrange dual, in order to determine $g(\lambda^+)$ and $g(\lambda^-)$, a centralized entity with knowledge of all the individual $g_i(\lambda)$'s and χ is required. If we rearrange (6.10), however, we will see that ratio consensus, together with a simple message-passing protocol, can be utilized to find λ^* without requiring a centralized decision maker or the need for each DGR to obtain all the individual functions $g_i(\cdot)$, the values they take for λ^+ or λ^-, or the value of χ.

First, note that (6.10) can be rewritten as

$$\lambda^* = \lambda^+ - \left(\frac{g(\lambda^+)}{\chi} - 1\right) \frac{\lambda^+ - \lambda^-}{\frac{g(\lambda^+)}{\chi} - \frac{g(\lambda^-)}{\chi}}, \tag{6.14}$$

where

$$\frac{g(\lambda^+)}{\chi} = \frac{\sum_{i=1}^m g_i(\lambda^+)}{\sum_{i=1}^m u_i^s}, \tag{6.15}$$

$$\frac{g(\lambda^-)}{\chi} = \frac{\sum_{i=1}^m g_i(\lambda^-)}{\sum_{i=1}^m u_i^s}, \tag{6.16}$$

which are ratios of sums of values that are known or can be computed by each local processor. As (6.15) and (6.16) imply, however, to find the optimal solution, each DGR must know λ^+ and λ^- *a priori*, that is, if each node knew which two $\lambda \in \mathcal{U}$ satisfied (6.11) and (6.12), ratio consensus could be used by the nodes to asymptotically obtain the ratios defined in (6.15) and (6.16), and consequently λ^*. From (6.14), we see that the criteria for determining λ^+ and λ^- are equivalent to finding $\lambda \in \mathcal{U}$ such that $g(\lambda)/\chi$ is closest to 1 from above and below, respectively; thus, if each DGR obtains $g(\lambda)/\chi$, $\forall \lambda \in \mathcal{U}$, it can determine which ratios correspond to $g(\lambda^+)/\chi$ and $g(\lambda^-)/\chi$. Furthermore, if each DGR also knows which $\lambda \in \mathcal{U}$ correspond to λ^+ and λ^-, λ^* can be computed and the optimal solution to the primal problem can be found using (6.13). Next, we provide a three-step procedure that enables the nodes to distributively perform the tasks above; the steps of this procedure are as follows.

[S1.] Without loss of generality, assume that initially, each node i only

knows its respective $\underline{\lambda}_i$ and $\overline{\lambda}_i$. Then, by broadcasting these values, all nodes in the out-neighborhood of i can obtain them, and in turn, broadcast them to their out-neighbors. Proceeding in this fashion, after a finite number of steps, bounded by the diameter of the graph representing the communication network, each node i will obtain every $\lambda \in \mathcal{U}$.

[S2.] Once each node has acquired every $\lambda \in \mathcal{U}$, ratio consensus is used with $2n$ numerator states and a single denominator state. Let $\underline{y}_i^{(j)}$ and $\overline{y}_i^{(j)}$, for $j = 1, \ldots, m$, and z_i, be the numerator and denominator states maintained by node i, respectively. Then, if the states are initialized such that $\underline{y}_i^{(j)}[0] = g_i(\underline{\lambda}_j)$, $\overline{y}_i^{(j)}[0] = g_i(\overline{\lambda}_j)$, for $j = 1, \ldots, n$, and $z_i[0] = u_i^s$, it follows from Lemma 1 that each node i can asymptotically obtain

$$\underline{\gamma}_i^{(j)} = \lim_{k\to\infty} \frac{\underline{y}_i^{(j)}[k]}{z_i[k]} = \frac{\sum_{l=1}^m g_l(\underline{\lambda}_j)}{\sum_{l=1}^m u_l^s} = \frac{g(\underline{\lambda}_j)}{\chi}, \tag{6.17}$$

$$\overline{\gamma}_i^{(j)} = \lim_{k\to\infty} \frac{\overline{y}_i^{(j)}[k]}{z_i[k]} = \frac{\sum_{l=1}^m g_l(\overline{\lambda}_j)}{\sum_{l=1}^m u_l^s} = \frac{g(\overline{\lambda}_j)}{\chi}, \tag{6.18}$$

for $j = 1, \ldots, m$, which correspond to $g(\lambda)/\chi$, $\forall \lambda \in \mathcal{U}$. Therefore, by determining which ratios are closest to 1 from above and below, and the corresponding λ^+ and λ^-, respectively, each DGR can compute λ^* using (6.14).

[S3.] After the values $\underline{\gamma}_i^{(j)}$ and $\overline{\gamma}_i^{(j)}$, $i, j = 1, \ldots, m$, have converged according to the ratio-consensus algorithm, and each node has determined λ^*, DGR i adjusts its reference command according to (5.1) where its incremental set-point at round $r = s$, with $\psi^s = \lambda^*$, is given by

$$\begin{aligned}
\Delta u_i^s &= g_i(\psi^s) - u_i^s, \\
&= \begin{cases} \underline{P}_i - u_i^s, & 0 \le \psi^s < \underline{\lambda}_i, \\ \alpha_i + \psi^s \beta_i - u_i^s, & \underline{\lambda}_i \le \psi^s \le \overline{\lambda}_i, \\ \overline{P}_i - u_i^s, & \overline{\lambda}_i < \psi^s. \end{cases} \\
&=: h_i^o(u_i^s, \psi^s).
\end{aligned} \tag{6.19}$$

Although the three-step procedure described above implies that each DGR must obtain every $\lambda \in \mathcal{U}$ (Step **S1**) before proceeding to run the ratio-consensus algorithm (Step **S2**), it is possible, with a slight modification, to execute the message-passing protocol and ratio consensus in parallel and still guarantee convergence of (6.17) and (6.18) (a formal proof can be found in

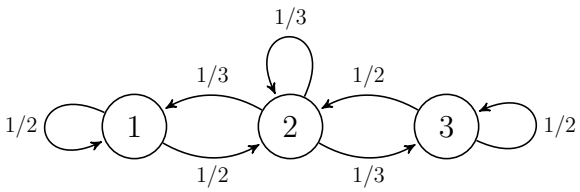


Figure 6.4: Graph of linear 3-node network.

[39]). In particular, each node i initializes z_i as before, but only node j initializes $\underline{y}_j^{(j)}[0] = g_j(\underline{\lambda}_j)$ and $\overline{y}_j^{(j)}[0] = g_j(\overline{\lambda}_j)$ since it is the only node that possesses $\underline{\lambda}_j$ and $\overline{\lambda}_j$ at $k = 0$. The remaining nodes initialize their numerator states corresponding to DGR j to zero, i.e., $\underline{y}_i^{(j)}[0] = 0$ and $\overline{y}_i^{(j)}[0] = 0$, for $i \neq j$, and, upon receiving $\underline{\lambda}_j$ and $\overline{\lambda}_j$, calculate $g_i(\underline{\lambda}_j)$ and $g_i(\overline{\lambda}_j)$, and add it to $\underline{y}_i^{(j)}$ and $\overline{y}_i^{(j)}$, respectively. The three-step procedure for distributively solving the optimal dispatch problem, including the modification to allow the message-passing protocol and ratio consensus to run in parallel, is described in Algorithm 3.

As was the case in the distributed algorithm for frequency regulation proposed in Section 6.1, the use of ratio consensus for obtaining $\underline{\gamma}_i^{(j)}$ and $\overline{\gamma}_i^{(j)}$ in (6.17) and (6.18) in practical optimal dispatch scenarios will only involve a finite number of iterations, k_o; this implies that the values $\underline{\gamma}_i^{(j)}$ and $\overline{\gamma}_i^{(j)}$ obtained at each node i will be bounded away from the true (limiting) values ($\frac{g(\underline{\lambda}_j)}{\chi}$ and $\frac{g(\overline{\lambda}_j)}{\chi}$, respectively) by a constant ϵ_o that can be made arbitrarily small by increasing k_o. The choice of ϵ_o affects the distance of the optimal solutions from the solution that will be chosen by the DGRs in a distributed fashion, but this distance can be made arbitrarily small by proper choice of k_o; we do not discuss the details of how to make this choice as this was not an issue in our experiments. The operation of Algorithm 3 is illustrated in the following example.

Example 7 (Optimal Dispatch)

Consider the graph in Fig. 6.4 which represents the exchange of information between DGR local processor in a 3-node network. We demonstrate the distributed optimal dispatch algorithm by showing the evolution of the $\underline{\gamma}$'s and $\overline{\gamma}$'s as computed by each of the local processors. Let $\alpha = \Big[-13/6, -3/2, -15/8\Big]$, $\beta = \Big[1/6, 1/2, 1/8\Big]$, $\underline{P} = \Big[10, 10, 20\Big]$, and $\overline{P} = \Big[100, 100, 120\Big]$. Also, omitting the index of the round, let $u = \Big[75, 75, 70\Big]$ such that $\chi = 220$. Given the

Algorithm 3: Distributed optimal dispatch

Each node $i,\ i = 1, \ldots, m$, **separately does the following:**

Input: $\alpha_i, \beta_i, \underline{P}_i, \overline{P}_i, u_i^s$

Output: Δu_i^s

begin

initialize states:

$$\underline{\lambda}_i = \frac{\underline{P}_i - \alpha_i}{\beta_i} \qquad \overline{\lambda}_i = \frac{\overline{P}_i - \alpha_i}{\beta_i}$$

$$\underline{y}_i^{(i)}[0] = g_i(\underline{\lambda}_i) \qquad \overline{y}_i^{(i)}[0] = g_i(\overline{\lambda}_i)$$

$$\underline{y}_i^{(j)}[0] = 0 \qquad \overline{y}_i^{(j)}[0] = 0,\ \forall j \neq i$$

$$z_i[0] = u_i^s$$

foreach *iteration,* $k = 0, 1, ..., k_o$ **do**

if $\underline{\lambda}_i$, $\overline{\lambda}_i$ *not broadcasted* **then**

broadcast $\underline{\lambda}_i$, $\overline{\lambda}_i$

if *received* $\underline{\lambda}_j$, $\overline{\lambda}_j$ **then**

$$\underline{y}_i^{(j)}[k] = \underline{y}_i^{(j)}[k] + g_i(\underline{\lambda}_j)$$
$$\overline{y}_i^{(j)}[k] = \overline{y}_i^{(j)}[k] + g_i(\overline{\lambda}_j)$$

update and broadcast states:

$$\underline{y}_i^{(j)}[k+1] = \sum_{l \in \mathcal{N}_i^-} \frac{1}{\delta_c^+(l) + 1} \underline{y}_l^{(j)}[k]$$

$$\overline{y}_i^{(j)}[k+1] = \sum_{l \in \mathcal{N}_i^-} \frac{1}{\delta_c^+(l) + 1} \overline{y}_l^{(j)}[k]$$

$$z_i[k+1] = \sum_{l \in \mathcal{N}_i^-} \frac{1}{\delta_c^+(l) + 1} z_l[k]$$

compute:

$$\underline{\gamma}_i^{(j)} = \frac{\underline{y}_i^{(j)}[k_o]}{z_i[k_o]} \qquad \overline{\gamma}_i^{(j)} = \frac{\overline{y}_i^{(j)}[k_o]}{z_i[k_o]}$$

determine $g(\lambda^+)/\chi$, $g(\lambda^-)/\chi$, λ^+, λ^-

compute $\psi^s = \lambda^*$ according to (6.14)

return $\Delta u_i^s = g_i(\psi^s) - u_i^s$

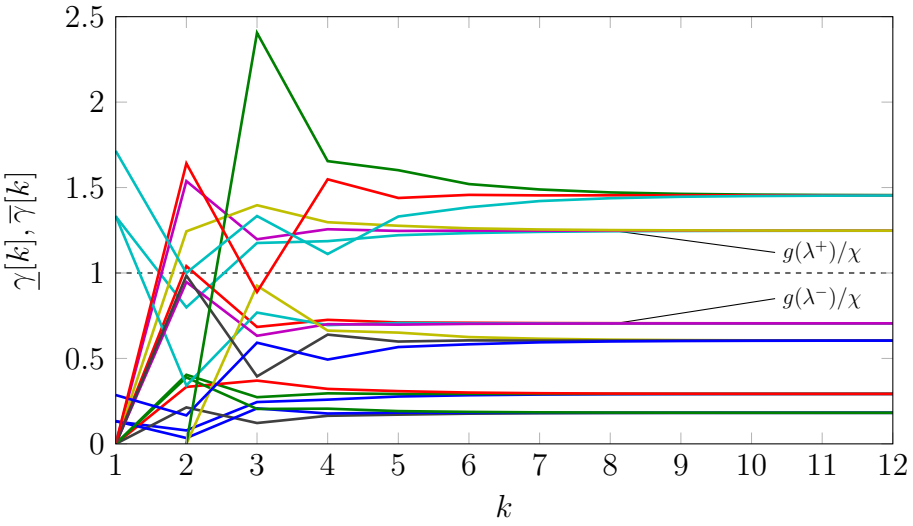


Figure 6.5: Example evolution of 6 instances of the ratio-consensus algorithm for determining the values of $g(\lambda^+)/\chi$ and $g(\lambda^-)/\chi$.

above values of α, β, $\underline{P}$, *and* $\overline{P}$, *and the definitions given in Section 6.2, we have that* $\underline{\lambda} = \begin{bmatrix}73, 23, 175\end{bmatrix}$ *and* $\overline{\lambda} = \begin{bmatrix}613, 203, 975\end{bmatrix}$.

The evolution of $\underline{\gamma}_i^{(j)}$ *and* $\overline{\gamma}_i^{(j)}$, *for* $i, j = 1, 2, 3$, *is shown in Fig. 6.5. From the figure, we see that all of the values converge after around 9 iterations, allowing the nodes to determine which ratios are closest from below and above to 1, i.e.,* $g(\lambda^-)/\chi$ *and* $g(\lambda^-)/\chi$, *the corresponding values of* λ^- *and* λ^+, *and the value of* λ^* *according to* (6.14). *After determining* $\psi = \lambda^* = 425.29$, *each node computes the amount by which it should adjust its generation set-point according to* (6.19) *and we have that* $\Delta u = \begin{bmatrix}-6.29, 25, -18.71\end{bmatrix}^{\mathrm{T}}$. *The plot shown in Fig. 6.6 illustrates the graphical interpretation of finding the optimal solution for this 3-node example.* ■

6.3 Choice of Controller Gains for Stabilizing Frequency

In this section, we will provide some guidelines for how to choose the κ_i's in (6.1) so that the distributed fair-splitting scheme for frequency regulation described in Section 6.1 decreases the frequency error, $\Delta\omega^r = \omega^r - \omega_0$. Additionally, we will show that if $|\Delta\omega^r| \leq \epsilon$ for some $r = s$, such that the distributed algorithm for optimal dispatch described in Section 6.2 is executed,

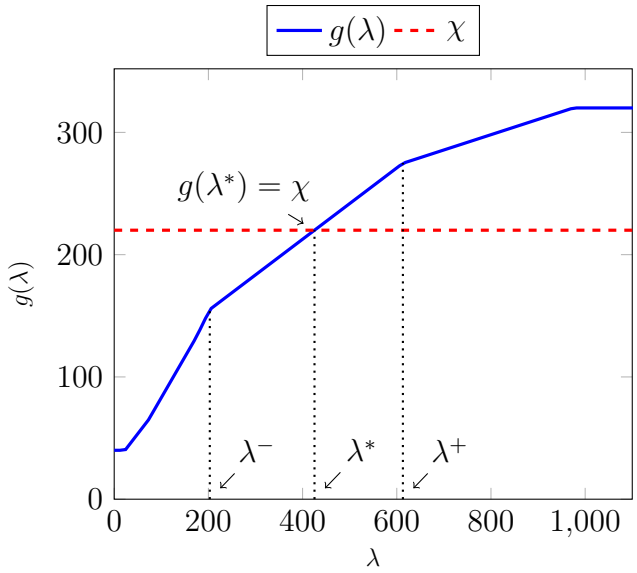


Figure 6.6: Graphical interpretation of optimal solution for $n = 3$

then $|\Delta\omega^{s+1}| \leq \epsilon$. To simplify the analysis in subsequent developments, we will make the following assumptions:

[A4.] The time constants associated with load changes are much slower than the time constants associated with the dynamics in (2.1) – (2.3) and (2.9), and the dynamics of the ratio-consensus algorithm.

[A5.] The error (with respect to the asymptotic solution) that results from executing ratio consensus a finite number of iterations is ignored. More specifically, the frequency regulation and optimal dispatch algorithms described in Section 6.1 and Section 6.2 are executed for a finite number of iterations, k_f and k_o, respectively, which causes inconsequential deviations from the asymptotic value defined in (6.3) and (6.19).

[A6.] Network losses are ignored, i.e., $P_l^r = 0$ in (10.1). While this simplifies the analysis, it does not preclude our distributed control approach from working on a lossy system. In fact, we show through experimental results in Chapter 8 that our approach can work in the presence of lossy interconnections.

6.3.1 Error Dynamics During Frequency Regulation

Together with assumptions [**A4**] – [**A6**] and the analysis in Section 4.2.2, it can be seen that, at iteration r, and for constant $u_i^r,\ i \in \mathcal{V}_g$, the common synchronous frequency at which the DGRs operate, ω^r, is given in (4.6), with the frequency error, $\Delta\omega^r$, defined in (10.1) (with $P_l^r = 0$). Therefore, since $u_i^{r+1} = u_i^r + \Delta u_i^r,\ i \in \mathcal{V}_g$, it follows that

$$\begin{aligned}
\Delta\omega^{r+1} =& \frac{\sum_{i\in\mathcal{V}_g} u_i^{r+1} - \sum_{i\in\mathcal{V}_\ell} P_i^d}{\sum_{i\in\mathcal{V}_s}(D_i + 1/R_i\omega_0) + \sum_{i\in\mathcal{V}_i} H_i},\\
=& \frac{\sum_{i\in\mathcal{V}_g}(u_i^r + \Delta u_i^r) - \sum_{i\in\mathcal{V}_\ell} P_i^d}{\sum_{i\in\mathcal{V}_s}(D_i + 1/R_i\omega_0) + \sum_{i\in\mathcal{V}_i} H_i},\\
=& \frac{\sum_{i\in\mathcal{V}_g} u_i^r - \sum_{i\in\mathcal{V}_\ell} P_i^d}{\sum_{i\in\mathcal{V}_s}(D_i + 1/R_i\omega_0) + \sum_{i\in\mathcal{V}_i} H_i}\\
&+ \frac{\sum_{i\in\mathcal{V}_g} \Delta u_i^r}{\sum_{i\in\mathcal{V}_s}(D_i + 1/R_i\omega_0) + \sum_{i\in\mathcal{V}_i} H_i},\\
=& \Delta\omega^r + \frac{\sum_{i\in\mathcal{V}_g} \Delta u_i^r}{\sum_{i\in\mathcal{V}_s}(D_i + 1/R_i\omega_0) + \sum_{i\in\mathcal{V}_i} H_i}.
\end{aligned} \tag{6.20}$$

Now, it follows from (6.3) and (6.4) that $\sum_{i\mathcal{V}_g} \Delta u_i^r = \sum_{i\in\mathcal{V}_g} \kappa_i(\omega^r - \omega_0) = \sum_{i\in\mathcal{V}_g} \kappa_i \Delta\omega^r$, and by plugging this expression into (6.20) and rearranging terms, we obtain

$$\Delta\omega^{r+1} = \left(1 + \frac{\sum_{i\in\mathcal{V}_g} \kappa_i}{\sum_{i\in\mathcal{V}_s}(D_i + 1/R_i\omega_0) + \sum_{i\in\mathcal{V}_i} H_i}\right)\Delta\omega^r. \tag{6.21}$$

Thus, if $\left|1 + \frac{\sum_{i\in\mathcal{V}_g} \kappa_i}{\sum_{i\in\mathcal{V}_s}(D_i+1/R_i\omega_0)+\sum_{i\in\mathcal{V}_i} H_i}\right| < 1$, then $\lim_{r\to\infty} \Delta\omega^r = 0$. Although there are many ways in which the κ_i's can be chosen such that $\Delta\omega^r$ goes to zero as r approaches infinity, a simple choice which requires only local information is to choose κ_i so as to satisfy

$$\begin{aligned}
-2(D_i + 1/R_i\omega_0) < \kappa_i < 0, \qquad & i \in \mathcal{V}_s,\\
-2H_i < \kappa_i < 0, \qquad & i \in \mathcal{V}_i.
\end{aligned}$$

6.3.2 Error Dynamics After Optimal Dispatch

As stated earlier, upon driving the frequency error to within the specified bound, i.e., $|\Delta\omega_r| < \epsilon$, the DGRs stop executing the distributed algorithm for frequency regulation and execute the distributed optimal dispatch algorithm described in Section 6.2. Let s denote the round at which the frequency error is driven below the error-tolerance, i.e., $|\Delta\omega^s| \leq \epsilon$. From the equality constraint in (6.5), it follows that the optimal dispatch solution $u_i^{s*},\ i = 1, 2, \ldots, n$, satisfies $\sum_{i\in\mathcal{V}_g} u_i^{s*} = \sum_{i\in\mathcal{V}_g} u_i^s$. But, from (5.1), (6.13), and (6.19), we also have that $u_i^{s+1} = u_i^s + \Delta u_i^s = u_i^{s*}$ such that $\sum_{i\in\mathcal{V}_g} \Delta u_i^s = 0$. Furthermore, given that

$$\Delta\omega^{s+1} = \Delta\omega^s + \frac{\sum_{i\in\mathcal{V}_g} \Delta u_i^s}{\sum_{i\in\mathcal{V}_s}(D_i + 1/R_i\omega_0) + \sum_{i\in\mathcal{V}_i} H_i}, \tag{6.22}$$

we have that $|\Delta\omega^{s+1}| = |\Delta\omega^s| \leq \epsilon$. Thus, after the frequency error is within the tolerance band, and the optimal dispatch algorithm is executed, the frequency error will remain within the tolerance band.

CHAPTER 7

LABORATORY TESTBED

We begin this chapter by describing the configuration and hardware specifications of the electrical network of a microgrid built in a laboratory to illustrate the effectiveness of the proposed distributed control architecture. Next, we provide the specifications for the hardware used to implement the microgrid's cyber network for communication, computation, and control. Then, we discuss the communication protocol created to exchange information for the distributed algorithms as well as a modification to ratio consensus which increases its robustness. Finally, we give an overview of a protocol used to synchronize the clocks of the nodes in the cyber network; this is important in order to ensure the correct execution of the control algorithms.

7.1 Physical Layer Hardware

To demonstrate the ability of the proposed distributed architecture to control a set of DGRs, we constructed the six-bus, 240 V, 3-phase power system shown in Fig. 7.1. The system comprises 3 Hampden Engineering synchronous machines, $\mathrm{G}_1, \mathrm{G}_2$, and G_3, (inverter-interfaced power supplies are omitted due to lack of availability) and 3 wye-connected resistive loads, labeled as $\mathrm{P}_1, \mathrm{P}_2$, and P_3. Each synchronous machine is connected at the shaft to a Kollmorgen Goldline brushless permanent magnet synchronous servomotor which serves as the prime mover. The synchronous machines have 3 pole-pairs; thus, operation at a mechanical speed of 1200 rpm results in an electrical frequency of 60 Hz, i.e., $f_i = 3\omega_i$, where f_i is the electrical frequency at the terminals of DGR i and ω_i has units rpm. The per-phase resistance of each load is adjusted by adding 500 Ω resistors in parallel. Each load can have up to 10 resistors in parallel per phase, yielding resistances in the range $500, 250, \ldots, 50\,\Omega$. Extra impedance is added via series inductors

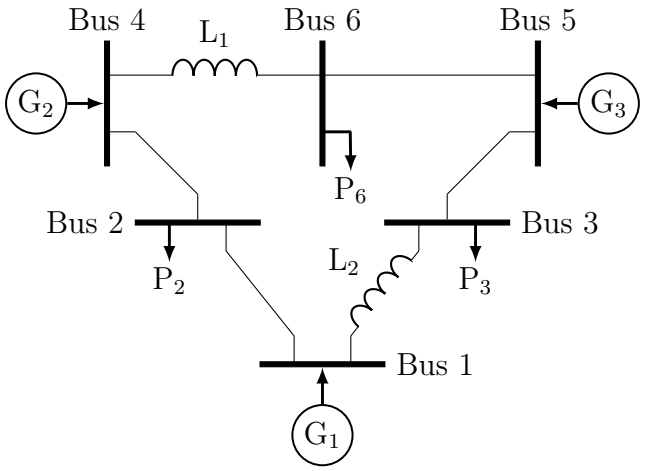


Figure 7.1: One-line diagram of experimental microgrid.

between bus 4 and bus 6 as well as between bus 1 and bus 3, as shown in Fig. 7.1. The per-phase inductances, the synchronous generator, and prime mover data are given in Appendix A.1.

7.2 Cyber Layer Hardware

The hardware chosen to create the cyber network for communication and computation is based upon the open-source electronics prototyping platform Arduino. While there are other suitable hardware choices, we chose Arduino for its simplicity and for the extensive software libraries and extension circuit boards, called shields, that are available.

Each node in the cyber network comprises an Arduino Mega 2560 microcontroller board, connected to a MaxStream XB24-DMCIT-250 revB XBee module via a SparkFun Electronics XBee shield. The Arduino Mega 2560 is based on the Atmel AVR ATmega2560, an 8-bit microcontroller with 256 kB of flash memory, a clock speed of 16 MHz, and four universal asynchronous receiver/transmitter (UART) ports. The XBee shield serves as an interface between the Arduino board and the XBee module while providing the requisite 3.3 V power supply via a voltage regulator. The XBee is an embedded RF module with a built-in chip antenna operating at 2.4 GHz that allows the nodes in the testbed to exchange packetized data wirelessly. Although our testbed utilizes wireless communication and is based upon XBee transceivers, other wireless technologies such as WiFi or Bluetooth, or wired communica-

tion would also be suitable.

While the XBee shield allows for plug-and-play operation between the Arduino and the XBee, it is configured to utilize the same UART port as the onboard USB-to-serial converter, preventing the Arduino from simultaneously communicating with a computer via the USB port and other nodes in the testbed via XBee. Thus, to allow for concurrent connections, we use a modified shield which routes the XBee UART to an alternative port on the Arduino. While this modification is useful for logging the evolution of the distributed algorithms, it is also necessary for enabling the Arduinos to control the prime movers connected to the synchronous generators in the electrical network.

7.3 Cyber Layer Software

Next, we discuss the communication protocol developed to exchange information among nodes in the cyber network, as well as a modification to the ratio-consensus algorithm that enables robust operation in the presence of unreliable communication links. Additionally, we provide an overview of the protocol used to synchronize the clocks of the cyber network nodes.

7.3.1 Communication Protocol

The communication protocol created to exchange information among the cyber network nodes can be described by the three abstraction layers illustrated in Fig. 7.2. The lowest layer, commonly referred to as medium access control (MAC), is implemented on the XBee modules and is based upon the ZigBee (IEEE 802.15.4) standard. The middle layer is a version of the xbee-arduino API [59], modified to account for the aforementioned alteration to the XBee shield and to enable incoming and outgoing packets to be time-stamped in order to increase the accuracy of the synchronization mechanism discussed later in this section. Each packet generated by the xbee-arduino API contains information about the sender and the intended recipient, allowing the Arduinos to send unicast messages as well as determine the source of broadcasted packets. Note that to enable use of the xbee-arduino API, the XBee modules are placed in API mode (AP=2 with escapes). The header

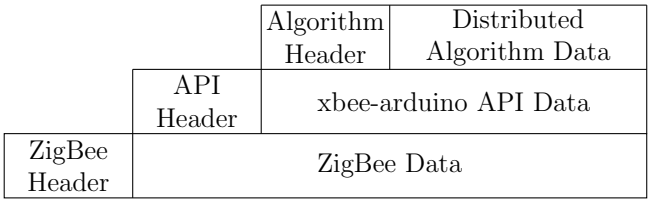


Figure 7.2: Communication protocol stack.

of the top layer contains information about which algorithm is being used, i.e., fair splitting or optimal dispatch, while the payload contains the actual information being exchanged.

In order to take advantage of the wireless medium used for communication among nodes, all of the packets used to exchange values for the distributed algorithms are broadcasted; that is, packets are not addressed to a particular node. Furthermore, to minimize network traffic, no acknowledgements are sent upon successful receipt of packets. To create a partially connected network despite the close proximity of the Arduinos during experimentation, each device is programmed to only accept packets received from the nodes in its in-neighborhood. In a more realistic setup, however, the testbed could be adapted to allow the availability of links between nodes to be determined dynamically based upon, for example, signal strength.

7.3.2 Ratio Consensus Implementation

Although the bottom layer of the protocol stack is designed to minimize packet collisions, it is still possible for packet loss to occur, particularly when attempting to reduce the time required to compute new generation commands using the distributed algorithms. To mitigate the effects of information lost due to temporarily unavailable communication links resulting from, e.g., neighboring nodes attempting to transmit simultaneously, we use the robust ratio-consensus algorithm outlined in Section 3.1.3. While the implementation details differ, it was shown in [46] that, under certain probabilistic assumptions on the link availability model, the asymptotic value of $\gamma_i[k]$ obtained with the robust ratio-consensus algorithm is identical to the original ratio-consensus algorithm described in Section 3.1.2 with probability one.

Upon examining (3.13), we see that in order to implement the robust

ratio-consensus algorithm, each node must store the most recent successfully received values, necessitating a unique identifier for each node in the in-neighborhood as well as a list of the identifiers of all possible in-neighbors given all available communication links. To identify the senders in our communication and computation testbed, we utilize the 64-bit hardware address associated with the XBee modules which is included in the xbee-arduino API headers. Furthermore, in our setup, the list of possible in-neighbors is programmed on the devices rather than generated at runtime.

Another important implementation detail that is evident from (3.13) is the fact that the previous successfully received values from each in-neighbor must be saved. In particular, each local processor i must store $\nu_{ij}[k]$ and $\eta_{ij}[k]$ for all $j \in \mathcal{N}_i^-$. While these values can be overwritten upon successfully receiving a packet from an in-neighbor, we see that an additional variable must be utilized for each numerator and denominator of ratio consensus, which, in the case of the distributed optimal dispatch algorithm, amounts to $2n + 1$ extra variables.

7.3.3 Clock Synchronization Protocol

Throughout the formulation of the ratio-consensus algorithm, we assumed that the local processors of all participating DGRs update the value of their state variables in unison; i.e., node i updates its state at iteration k at the same time node j updates its state, $\forall i, j, \in \mathcal{V}_c$. Without a common time reference and with no acknowledgements, however, it is possible for the local processors to update their states at different times which may result in convergence to an inaccurate solution. While robust ratio consensus reduces the sensitivity of the system to timing errors, it is still possible for the nodes to converge to the wrong solution. Thus, before the distributed control algorithms are executed, all nodes are synchronized to a common time reference. The synchronization mechanism used in our hardware testbed is based on the hierarchy referencing time synchronization (HRTS) protocol proposed in [60]. This protocol has low overhead requirements and is capable of synchronizing the clocks of several nodes to a reference using only three packets. A detailed description of the synchronization process is provided in [38].

7.3.4 Initialization Protocol

Following the synchronization of the clocks, one of the nodes must initiate the distributed algorithms to ensure that all nodes in the network begin execution at roughly the same time. To facilitate this, the first node that detects that the frequency error has exceeded the specified bound or that the DGRs should be optimally dispatched will broadcast a scheduling packet containing information about the algorithm used, the start time, the number of iterations, and the period of each iteration. Upon receiving the scheduling packet, all other nodes will rebroadcast it to allow the information to propagate throughout the network, then wait until the scheduled start time. If multiple scheduling packets are received, the one with the earliest start time is chosen.

CHAPTER 8

EXPERIMENTAL RESULTS

Using the laboratory testbed described in the previous chapter, we verify the effectiveness of our distributed generation control architecture under a variety of scenarios. We first present results demonstrating the distributed frequency regulation and optimal dispatch functions controlling the DGRs in the experimental microgrid following both an increase and a decrease in load. We then add a fourth DGR to the system which acts as a spinning reserve to illustrate how the local processor of each DGR can independently determine if the collective power output limits have been exceeded and act accordingly.

Throughout this chapter, all frequency is reported in hertz, and, to accommodate laboratory equipment features, the limits and set-points of the prime movers are given as torque rather than as power. To provide greater control while performing the experiments, instead of allowing a random DGR to initialize the algorithms as specified in Section 7.3.4, in the results that follow, we select a single DGR, referred to as the leader, which is responsible for estimating the total torque that needs to be added or removed at each round, i.e., without loss of generality let "1" index the leader, then $\Delta\hat{u}_1^r = \kappa_1(f^r - f_0)$ while $\Delta\hat{u}_i^r = 0$, $\forall i \neq 1$, $\forall r$. Furthermore, to ensure that ψ^r can be reliably computed to sufficient accuracy at each round, we chose to have the nodes execute 50 iterations with a period of 50 ms for the frequency regulation algorithm, and 100 iterations with a 300 ms period for the optimal dispatch algorithm.[1] Note that the period and number of iterations to be executed were chosen conservatively to allow sufficient time for the algorithms to converge and to reduce the response time of the system in case

[1] As implied by Lemma 1, convergence of the ratio-consensus algorithm is asymptotic. In practice, however, a finite number of iterations is sufficient to converge to a solution that is accurate to within the capabilities of the DGRs. For the experimental results presented in this chapter, the number of iterations necessary was determined *a priori* for each communication graph and was pre-programmed into the leader node.

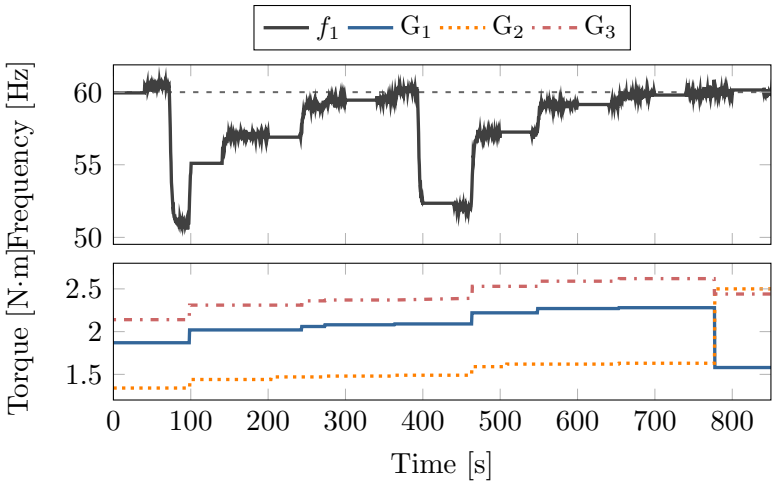


Figure 8.1: System response to load increase.

of a malfunction during experimentation; both the number of iterations and their period could be reduced given more development time, and by making use of algorithms such as the one proposed in [61].

In each of the scenarios we tested experimentally, the coefficients defining the DGR cost functions are $\alpha = \left[-13/6, -3/2, -15/8\right]^{\mathrm{T}}$ N·m and $\beta = \left[1/6, 1/2, 1/8\right]^{\mathrm{T}}$ N·m. Additionally, the desired nominal frequency is 60 Hz, i.e., $f_0 = 60$ Hz, and the error bound used by the leader to determine when to trigger the optimal dispatch algorithm as defined in (5.3) is $\epsilon = 0.4$ Hz. Finally, in all following scenarios, the loads are three-phase balanced while the field excitation of each of the DGRs is left constant, i.e., voltage is unregulated.

8.1 Load Increase

We begin by illustrating the system response to a load increase. In this experiment, the exchange of information between local DGR controllers is represented by the graph shown in Fig. 6.4, where nodes $1, 2, 3$ correspond to generators G_1, G_2, G_3 in Fig. 7.1, respectively. The minimum output torque of all DGRs is 0 N·m while the maximum output torques for DGRs G_1, G_2, and G_3 are 4, 2.5, and 3.5 N·m, respectively. Additionally, the gain used by node 1 (the leader node) to compute $\Delta\hat{u}_1^r$ according to (6.1) is chosen to be $\kappa_1 = 85$ μN·m·s.

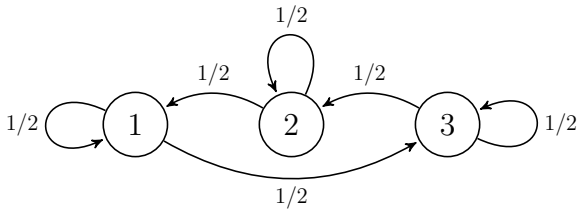


Figure 8.2: Graph of cyclic 3-node network.

The plot in Fig. 8.1 shows the torque set-points for each of the three generators as well as the frequency of DGR G_1 as the system responds to two load increases at times $t = 100$ s and $t = 400$ s, and a subsequent generator re-dispatch at time $t = 775$ s which is triggered to minimize the overall generation cost. For the first and second load changes, the total power drawn is increased from 313 W to 396 W, and from 396 W to 465 W by adjusting the resistive load from 500 Ω/phase to 250 Ω/phase at buses 3 and 2, respectively.

From Fig. 8.1 we see that, following the first load increase, the frequency measured at DGR G_1 returns to 60 Hz after four rounds of the fair-splitting algorithm. If we let $r = 0$ be the index of the round before the first load increase and subsequent execution of the frequency regulation algorithm, then the values of ψ^r as defined in (6.3) for $r = 1, 2, 3, 4$ are 0.58, 0.59, 0.59, and 0.60, respectively. Similar to the first load change, three rounds of the fair-splitting algorithm, corresponding to $r = 5, 6, 7$, return the frequency to the nominal value following the second load change. The values of ψ^r for the last three rounds of the frequency regulation algorithm are 0.63, 0.65, and 0.65, respectively.

After the two load changes and the execution of 7 rounds of the fair-splitting algorithm, the frequency is within the specified bound, i.e., $|f_1 - f_0| \leq \epsilon$, and, as described by (5.3), the optimal dispatch function is used to determine the amount by which the DGRs should adjust their output in order to minimize the overall cost. If we let $s = 8$ be the index of the round that the DGRs are re-dispatched optimally, then the local processors determine that $\psi^s = 276.36$ N·m, and, as Fig. 8.1 shows, the DGRs adjust their set-points according to (6.19) at time $t = 775$ s.

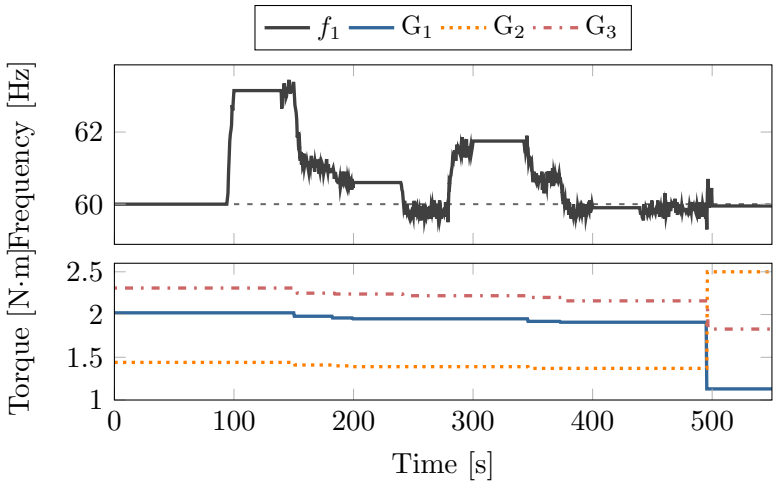


Figure 8.3: System response to load decrease.

8.2 Load Decrease

We now illustrate the system response to a load decrease. For this experiment, the cyclic graph in Fig. 8.2 represents the exchange of information among the DGR local processors. Similar to the previous experiment, DGR 1 is selected to be the leader with $\kappa_1 = 90$ μN·m·s, the minimum output torques of all DGRs are 0 N·m, and the maximum output torques for DGRs G_1, G_2, and G_3 are 4, 2.5, and 3.5 N·m, respectively.

The plot in Fig. 8.3 shows the torque set-points for each of the three generators as well as the frequency measured at DGR G_1 as the system responds to two load decreases at times $t = 100$ s and $t = 300$ s, and a subsequent generator re-dispatch at time $t = 500$ s, which is triggered to minimize the overall generation cost. For the first and second load changes, the total power drawn is decreased from 390 W to 365 W, and from 365 W to 352 W by adjusting the resistive load at bus 6 from 500 Ω/phase to 750 Ω/phase, and from 750 Ω/phase to 1000 Ω/phase, respectively.

Following the first load decrease, we see from Fig. 8.3 that the frequency of DGR G_1 returns to 60 Hz after three rounds of the fair-splitting algorithm. If we let $r = 0$ be the index of the round before the first load change and subsequent execution of the frequency regulation algorithm, then the values of ψ^r for $r = 1, 2, 3$ are 0.56, 0.56, and 0.56, respectively. After the second load change, two rounds of the fair-splitting algorithm, corresponding to $r = 4$ and 5, return the frequency to the nominal value. The values of ψ^r for

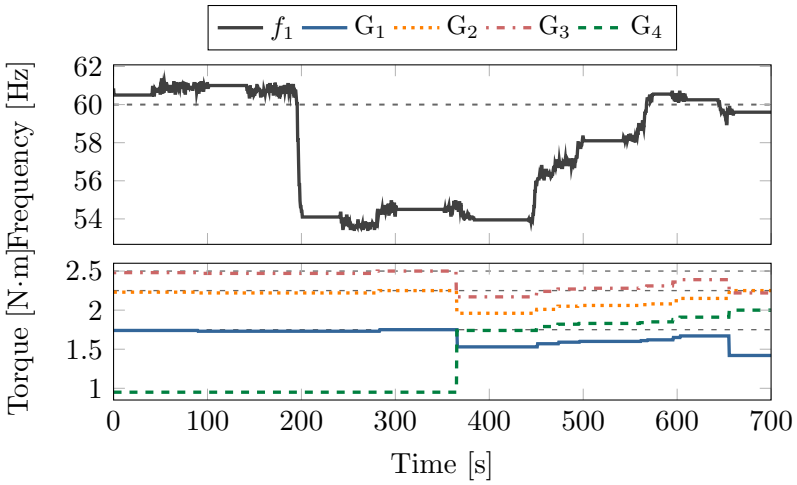


Figure 8.4: System response before and after spinning reserve switch-in.

the last two rounds of the frequency regulation algorithm are 0.55 and 0.54.

After the two load changes and the execution of 5 rounds of the fair-splitting algorithm, the frequency is within the specified bound and the generators are re-dispatched in order to minimize the overall cost. If we let $s = 6$ be the index of the round that the optimal dispatch function is executed, then the local processors determine that $\psi^s = 243.57$ N·m and the DGRs adjust their set-points according to (6.19) at time $t = 500$ s as shown in Fig. 8.3.

8.3 Load Increase with Spinning Reserve

As illustrated in Example 6, the value of ψ^r obtained through the distributed computation of the frequency regulation function allows each DGR local processor to independently determine if the collective power output limits have been exceeded. Next, we demonstrate how this information can be used by a DGR acting as a spinning reserve to determine when to come online.

In order to demonstrate the capability of each generator to independently determine when the demand for generation exceeds the collective output limits of the online DGRs, we connected an additional DGR to bus 2, which we refer to as DGR G_4. Initially, the additional DGR limits its output to the minimum torque required to operate at nominal speed, but participates in the distributed algorithm with a maximum torque of 0.95 N·m. The actual

maximum torque of the DGR G_4 is 2 N·m, while DGRs G_1, G_2, and G_3 can output up to 2.5, 2.25, and 1.75 N·m, respectively; the minimum torque output is 0.1 N·m for all DGRs. The exchange of information among the DGRs conforms to the graph in Fig. 6.1.

The plot in Fig. 8.4 shows the prime mover torque and the frequency of DGR G_1 as the electrical load is increased. At $t \approx 275$ s, the maximum output of DGRs G_1, G_2, and G_3 is reached, and the frequency error can no longer be driven to zero. At this point, by obtaining a value of ψ^r greater than one, DGR G_4 determines that the demand exceeds the collective limits and adjusts its maximum torque output to its actual value of 2 N·m. This occurs at $t \approx 360$ s, after which all DGRs are able to increase their output to drive the frequency to the nominal value of $f_0 = 60$ Hz. Once the reserve generator is switched in and the frequency is returned to the nominal value, the optimal dispatch function is executed and the DGRs adjust their set-points at time $t \approx 650$ s.

Part III

Distributed Frequency Regulation Architecture for Islanded AC Microgrids with No Inertia

CHAPTER 9

INTRODUCTION AND PRELIMINARIES

We begin this chapter by providing a brief overview of the focus of the work presented in this part, and outline the organization of the remainder of this part. Then, we revisit the models presented in Chapter 2 and introduce some assumptions that reduce the generator model to a more simple form that is sufficient to represent inverter interfaced generators. Finally, we formalize the notion of phase cohesiveness.

9.1 Introduction

In Section 1.1, we discussed the objectives that a control architecture for ac microgrids with no inertia should be designed to achieve. More specifically, we noted that, for inertia-less microgrids, a control architecture for frequency regulation must be designed to ensure that stable operation results and that the frequency at every bus in the system is equal to the desired reference value. We revisit these objectives next, and briefly discuss the notion of phase cohesiveness.

Although there is a lack of existing control architectures designed to ensure stable operation at a common reference frequency (see discussion in Section 1.2), the authors in [41] provide a condition for a broad class of coupled oscillators, which can model the generators and loads in an inertia-less microgrid, that is sufficient for meeting the so-called *phase-cohesiveness* requirement. In the context of power flows in a microgrid, satisfying this condition is equivalent to ensuring that the angle difference between every pair of connected buses in steady state will be strictly less than $\frac{\pi}{2}$. Moreover, if the condition for phase cohesiveness is met, the natural frequencies of the oscillators—equivalent to the power injected at each bus in a microgrid—and the coupling between them—equivalent to the admittances of the branches

interconnecting buses in a microgrid—are such that the system exhibits coherent behavior, i.e., the angle of every oscillator evolves at the same rate. Thus, when combined with a scheme for regulating the average frequency error, ensuring phase cohesiveness is sufficient for ensuring stability and system-wide operation at a common frequency. Next, we provide an overview of the approach we propose in this part for frequency regulation in ac microgrids with no inertia that makes use of the phase cohesiveness condition.

As in [37], our proposed control architecture is designed to take advantage of the results in [41] by tracking generator set-points that, for some initial load demand, are known to result in phase-cohesive operation. Following one or more small perturbations to the power demanded by the loads, the architecture iteratively adjusts the generator set-points to drive the average frequency to some reference value while also ensuring that the operating point that results is phase cohesive. By regulating the average frequency around an operating point that is known to be phase cohesive, our controller ensures small-signal stability of the closed-loop system while also guaranteeing that the frequency at every bus is the same. To handle larger load perturbations, we also provide a method for triggering the recomputation of the generator set-points to be tracked based upon an estimate for the amount by which the system has deviated from the original phase-cohesive operating point. Using three of the distributed algorithms introduced in Chapter 3, we also outline a distributed implementation of our proposed control architecture. These three algorithms enable the acquisition of global information with which processors located at each bus can make decisions to collectively achieve the system-level objectives of our proposed control architecture. Finally, we provide analytical criteria for choosing gains that result in closed-loop stability and demonstrate the operation of our proposed architecture and its distributed implementation using numerical simulations of three case studies

The remainder of this part is organized as follows. In Section 9.2, we revisit the physical and cyber layer models outlined in Chapter 2 and introduce some simplifications to the dynamic generator model; we also formalize of the notion of phase cohesiveness. In Chapter 10, we introduce the control scheme for our proposed frequency regulation architecture and outline criteria for ensuring closed-loop stability. In Chapter 11 we detail a distributed implementation of our proposed frequency regulation architecture that relies on the distributed algorithms introduced in Chapter 3. In order to vali-

date our control architecture and its distributed implementation, we provide simulation results for three test cases in Chapter 12.

9.2 Preliminaries

In this section, we briefly revisit the physical layer model introduced in Chapter 2, and outline some simplifications to the dynamics of the generator model. Then, we formally define the notion of phase cohesiveness and outline a sufficient condition for achieving it; we then pose the criterion for achieving phase cohesiveness and other control objectives as a feasibility problem.

9.2.1 Physical Layer Model Modifications

For the work presented in this part, we consider ac microgrids comprising loads and generators that can be represented by a first-order dynamical model; we further restrict consideration to systems in which all generation units are interfaced through power electronics, i.e., we consider systems comprising generators with no inertia. To facilitate the analytical discussion presented herein, we assume that the generators have no internal impedance, the power network is lossless, and the voltage magnitude at every bus is constant and unity. Although these assumptions may not hold under extreme loading conditions, the control architecture we propose in Chapter 10 is designed for operation under nominal conditions where they justifiably approximate network behavior (see, e.g., [41]). We formalize these assumptions next.

If we assume that the value of this reactance approaches zero, we have that $\delta_i \to \theta_i$ such that we can rewrite (2.9) as

$$H_i \frac{d\theta_i}{dt} = u_i - V_i \sum_{j=1}^{n} V_j \left[G_{ij} \cos(\theta_i - \theta_j) + B_{ij} \sin(\theta_i - \theta_j) \right]. \tag{9.1}$$

Additionally, if the spatial separation between buses in the microgrid is small, we can assume that the resistance of the lines interconnecting the buses is negligible, i.e., $G_{ij} \to 0$ for $(i, j) \in \mathcal{E}_p$. Finally, for the purposes of the control architecture proposed herein, we consider operation that does not deviate significantly from nominal conditions such that the voltage magnitude at each

bus is constant and unity, i.e., $V_i = 1$ pu for $i \in \mathcal{V}_p$ (for further justification of these final two assumptions, see, e.g., [41]).

Although the power demand of the load model in (2.4) is assumed to be constant, for the work in this part, we allow for small perturbations away from the initial power demanded by each load. To that end, let $\ell_i(t) := \ell_i^0 + \Delta\ell_i(t) \geq 0$ denote the real power demand at load bus $i \in \mathcal{V}_\ell$ where $\ell_i^0 \geq 0$ is the demand at $t = 0$, and $\Delta\ell_i(t)$ is a load perturbation that occurs at some $t > 0$. Given the assumptions outlined above, the dynamics of the generators and loads in an inertia-less microgrid can be written as

$$D_i \frac{d\theta_i(t)}{dt} = u_i(t) - \sum_{j \in \mathcal{N}_p(i)} B_{(i,j)} \sin(\theta_i(t) - \theta_j(t)), \qquad i \in \mathcal{V}_g, \quad (9.2)$$

$$D_i \frac{d\theta_i(t)}{dt} = -(\ell_i^0 + \Delta\ell_i(t)) - \sum_{j \in \mathcal{N}_p(i)} B_{(i,j)} \sin(\theta_i(t) - \theta_j(t)), \quad i \in \mathcal{V}_\ell, \quad (9.3)$$

where $D_i = H_i$ for $i \in \mathcal{V}_p$. For the case when $D_i = 1$, $i \in \mathcal{V}_p$, the reduced system in (9.2) – (9.3) is equivalent to the so-called Kuramoto model for coupled oscillators proposed in [62]. In the context of the Kuramoto coupled oscillator model, the u_i's and ℓ_i's correspond to the natural frequencies of the generator and load oscillators, respectively, while the $B_{(i,j)}$'s are the coupling coefficients between them.

Recall the graph representing the electrical connections between buses, $\mathcal{G}_p$, introduced in Section 2.1. If we arbitrarily assign a direction to each edge $e = (i, j) \in \mathcal{E}_p$, the oriented incidence matrix, denoted by $M_p \in \mathbb{R}^{n \times |\mathcal{E}_p|}$, is defined as $M_p := [m_{ie}^p]$ where

$$m_{ie}^p = \begin{cases} -1, & \text{if } i \text{ is the sink node of edge } e, \\ 1, & \text{if } i \text{ is the source node of edge } e, \\ 0, & \text{otherwise.} \end{cases} \quad (9.4)$$

Furthermore, we define the weighted Laplacian matrix of $\mathcal{G}_p$ as $L = M_p \text{diag}(\{B_{(i,j)} : (i, j) \in \mathcal{E}_p\}) M_p^{\mathrm{T}}$, where $\text{diag}(\{B_{(i,j)} : (i, j) \in \mathcal{E}_p\}) \in \mathbb{R}^{|\mathcal{E}_p| \times |\mathcal{E}_p|}$ is a diagonal matrix consisting of susceptances, $B_{(i,j)}$, $(i, j) \in \mathcal{E}_p$, between electrically connected buses i and j. Given that the Laplacian is singular (due to its zero eigenvalue), we will utilize its *Moore-Penrose pseudo inverse*, denoted by $L^\dagger$ (see, e.g., [63] for details on how to compute it).

9.2.2 Phase Cohesiveness Criterion

The control architecture we will propose in Chapter 10 is designed to adjust the output of the generators in a microgrid to track pre-determined set-points while simultaneously eliminating the frequency error that results from small perturbations in load. These pre-determined set-points are chosen such that, when applied, the system is stable around the resultant operating point, the average frequency equals some reference value, and the frequency at every bus is the same. In the discussion that follows, we formalize the properties that characterize such an operating point, and introduce a feasibility problem for choosing generator set-points, the solution of which meets these properties.

Given constant generator set-points, which we denote by u_i^*, $i \in \mathcal{V}_g$, and the initial load power demands ℓ_i^0, $i \in \mathcal{V}_\ell$, as described in Section 9.2.1, let $\theta^* = \left[\theta_1^*, \theta_2^*, \ldots, \theta_n^*\right]^{\mathrm{T}}$ denote a steady-state operating point of the system in (9.2) – (9.3). Then, for some reference frequency, which we denote by ω_0, we would like to determine the u_i^*'s such that, when applied to the generators, the operating point θ^* exists and is characterized by the following properties:

P1. the average frequency in the system equals the reference, i.e.,

$$\frac{\sum_{i \in \mathcal{V}_p} \dot{\theta}_i(t)}{n} = \omega_0;$$

P2. the frequency at every bus is equal, i.e.,

$$|\dot{\theta}_i(t) - \dot{\theta}_j(t)| = 0$$

for $i, j \in \mathcal{V}_p$; and

P3. the system is stable around the operating point θ^*, i.e.,

$$-\nabla_{\theta(t)} h(\theta(t))\Big|_{\theta=\theta^*} \preceq 0,$$

where $\theta := \left[\theta_1(t), \ldots, \theta_n(t)\right]^{\mathrm{T}}$ and $h(\theta(t)) = [h_e(\theta(t))]$ with $h_e(\theta(t)) := B_e \cos(\theta_i - \theta_j)$ for $e = (i, j) \in \mathcal{E}_p$.

While we will show in Chapter 10 that meeting Property P1 can be achieved by balancing generation and demand, i.e., choosing the u_i^*'s such

that $\sum_{i\in\mathcal{V}_g} u_i^* = \sum_{i\in\mathcal{V}_\ell} \ell_i^0$, ensuring Properties P2 and P3 are met is more difficult, especially when trying to do so in a distributed fashion. However, if generation and demand are balanced and the so-called *phase-cohesiveness* condition is met, i.e., $|\theta_i^* - \theta_j^*| \leq \phi$ for $(i,j) \in \mathcal{E}_p$ and $\phi \in [0, \pi/2)$ [41], it can be shown that the resultant operating point, θ^*, exists and is characterized by Properties P1 – P3. To that end, if we let $u^* = \left[u_1^*, u_2^*, \ldots, u_m^*\right]^{\mathrm{T}}$ and $\ell^0 = \left[\ell_{m+1}^0, \ell_{m+2}^0, \ldots, \ell_n^0\right]^{\mathrm{T}}$, then it follows from the results in [41] that, for a broad class of network topologies, if

$$\left\| M_p^{\mathrm{T}} L^{\dagger} \begin{bmatrix} u^* \\ -\ell^0 \end{bmatrix} \right\|_{\infty} < \sin(\phi), \tag{9.5}$$

for some angle $\phi \in [0, \pi/2)$, the resulting operating point, θ^*, exists and is phase cohesive. In subsequent developments, we refer to such u_i^*'s as *phase-cohesive set-points*; although this term is a slight abuse of nomenclature, we use it as short-hand to refer to set-points that result in phase-cohesive operation, subject to the imposed load demands. Despite being proven for several topologies, including acyclic networks and those comprising low-dimensional cycles, the condition in (9.5) is not sufficient for general networks. As a result, we restrict consideration to network topologies for which (9.5) holds (we refer the reader to [41] and supporting information for details).

Beyond finding phase-cohesive set-points that are balanced with demand, we must further restrict the problem to finding those set-points that lie within the individual bounds of the generators, i.e., $\underline{u}_i \leq u_i^* \leq \overline{u}_i$, $i \in \mathcal{V}_g$. Thus, the task of finding generator set-points that meet all of the above-described requirements can be summarized by the following feasibility problem:

$$\begin{aligned} \text{find} \quad & u \\ \text{subject to} \quad & \sum_{i\in\mathcal{V}_g} u_i = \sum_{i\in\mathcal{V}_\ell} \ell_i^0, \\ & \left\| M_p^{\mathrm{T}} L^{\dagger} \begin{bmatrix} u \\ -l^0 \end{bmatrix} \right\|_{\infty} < \sin(\phi), \\ & \underline{u}_i \leq u_i \leq \overline{u}_i, \, \forall i \in \mathcal{V}_g. \end{aligned} \tag{9.6}$$

CHAPTER 10

FREQUENCY REGULATION ARCHITECTURE

In this chapter, we propose a control architecture that is designed to regulate the frequency in an inertia-less ac microgrid. We begin by introducing relevant notation and by providing an overview of the architecture. Then, we formalize the control scheme and outline criteria for choosing gains that yield closed-loop stability. Finally, we describe a method for determining when to recompute set-points to be tracked so as to maintain phase cohesiveness.

10.1 Overview and Notation

From a system-level perspective, the scheme we propose for frequency regulation is similar to a discrete-time proportional integral controller. More specifically, after determining set-points that satisfy (9.6), which we denote by u_i^*, $i \in \mathcal{V}_g$, the set-point of each generator i is incrementally adjusted away from u_i^* over several discrete time intervals. Following one or more small perturbations in the power demanded by the loads, these incremental changes serve to drive the resulting frequency error in the system to zero. However, in general, these incremental adjustments move the system away from an operating point that is known to be phase cohesive. To counter this adverse effect, our control architecture includes a method for determining when to recompute the u_i^*'s based upon an estimate for the amount by which the system has deviated from the phase-cohesive operating point.

Without loss of generality, if we transform the microgrid model in (9.2) – (9.3) to a reference frame rotating at the reference frequency, ω_0, we see that an operating point at which the frequency of every bus is ω_0 is equivalent to one at which the derivative of the voltage angle at each bus is zero, i.e., $\frac{d\theta_i(t)}{dt} = 0$, $i \in \mathcal{V}_p$. Let $\Delta\overline{\omega}(t)$ denote the average frequency error in this reference frame, weighted by the time constants associated with the dynamics

of the generators and loads, and define it at time $t > 0$ to be

$$\Delta\overline{\omega}(t) := \frac{\sum_{i=1}^{n} D_i \frac{d\theta_i(t)}{dt}}{\sum_{i=1}^{n} D_i}. \tag{10.1}$$

Replacing the numerator in (10.1) with a summation of all of the equations in (9.2) and (9.3), it follows that the value of the average frequency error at time $t > 0$ is given by

$$\Delta\overline{\omega}(t) = \frac{1}{\sum_{i=1}^{n} D_i} \left(\sum_{i \in \mathcal{V}_g} u_i(t) - \sum_{i \in \mathcal{V}_\ell} \ell_i(t) \right). \tag{10.2}$$

As mentioned above, our frequency regulation architecture incrementally adjusts the generator set-points over several discrete time intervals. We refer to the intervals during which the controller is executed as *rounds* and index them by $r = 0, 1, 2, \ldots$. To simplify notation, we reset the round index every time the set-points to be tracked are computed, i.e., the u_i^*'s are always determined immediately prior to round $r = 0$. We denote the duration of the rounds by T_0 and define the time at the beginning of each round r to be $t_r := rT_0$. For our controller to eliminate the frequency error that results from changes in load, we assume that a sufficient number of rounds have elapsed between load perturbations. As a result, for the analysis of the control scheme we present next, we assume that the power demanded by the loads is constant, i.e., if the power demanded by load $i \in \mathcal{V}_\ell$ is perturbed by $\Delta\ell_i(t)$ at $t = t_0$, then $\ell_i(t) = \ell_i^0 + \Delta\ell_i(t_0)$ for $t_0 < t < r_0 T_0$ and r_0 sufficiently large.

Let $u_i[r] := u_i(t)$, $t_r \leq t < t_{r+1}$, be the set-point of generator $i \in \mathcal{V}_g$ during round r, and assume that it is adjusted at the beginning of the round and held constant for the remaining duration. Then, given the assumption that the load power demands are constant, it follows from (10.2) that the average frequency error during round r is

$$\Delta\overline{\omega}[r] = \overline{D} \left(\sum_{i \in \mathcal{V}_g} u_i[r] - \sum_{i \in \mathcal{V}_\ell} (\ell_i^0 + \Delta\ell_i) \right), \tag{10.3}$$

where $\overline{D} := \frac{1}{\sum_{i=1}^{n} D_i}$.

10.2 Control Scheme

From (10.3), if we assume that $u_i[r] = u_i^*$, for $i \in \mathcal{V}_g$ and $r = 0, 1, 2, \ldots$ i.e., the set-points of the generators are constant and satisfy (9.6), it is clear that, following one or more changes to the power demanded by the loads, the average frequency error at round r is given by

$$\begin{aligned}\Delta\overline{\omega}[r] &= \overline{D}\left(\sum_{i\in\mathcal{V}_g} u_i^* - \sum_{i\in\mathcal{V}_\ell} \ell_i^0 - \sum_{i\in\mathcal{V}_\ell} \Delta\ell_i\right), \\ &= -\overline{D}\sum_{i\in\mathcal{V}_\ell} \Delta\ell_i. \end{aligned} \tag{10.4}$$

That is, for constant generator set-points, the value of the average frequency error is equal to the additive inverse of the sum of the load perturbations, weighted by the sum of the generator and load time constants. We see from this relationship that, in order to drive the frequency error to zero, it is sufficient to adjust the $u_i[r]$'s such that

$$\lim_{r\to\infty} \sum_{i\in\mathcal{V}_g} u_i[r] = \sum_{i\in\mathcal{V}_\ell} \ell_i^0 + \Delta\ell_i. \tag{10.5}$$

Given previously determined phase-cohesive set-points, u_i^*, $i \in \mathcal{V}_g$, and in order to satisfy (10.5), our control architecture adjusts the set-point of generator i at round r according to

$$u_i[r] = u_i^* + \Delta u_i[r], \tag{10.6}$$

where $\Delta u_i[r]$ denotes the incremental amount by which $u_i[r]$ is adjusted away from u_i^*. We define the incremental set-point to be $\Delta u_i[r] := \alpha_i e_i[r]$, where the value of $e_i[r]$ is updated recursively as

$$e_i[r+1] = e_i[r] + \kappa_i \Delta\overline{\omega}[r], \tag{10.7}$$

for appropriately chosen gains α_i and κ_i for $i \in \mathcal{V}_g$, and with $e_i[0] = 0$, $i \in \mathcal{V}_g$. (Note that, in addition to resetting the round index, the value of e_i, $i \in \mathcal{V}_g$ must also be reset to zero when the u_i^*'s are recomputed.) Next, we outline conditions for choosing gains that ensure the closed-loop system is stable and that $\Delta\overline{\omega}[r] \to 0$ as $r \to \infty$.

10.3 Stability Analysis and Criteria for Choosing Gains

If we define $\alpha := \begin{bmatrix}\alpha_1, \dots, \alpha_m\end{bmatrix}^{\mathrm{T}}$, $\kappa := \begin{bmatrix}\kappa_1, \dots, \kappa_m\end{bmatrix}^{\mathrm{T}}$, and $e[r] := \begin{bmatrix}e_1[r], \dots, e_m[r]\end{bmatrix}^{\mathrm{T}}$, then, by stacking (10.3) with (10.7) for $i \in \mathcal{V}_g$, we can write the closed-loop system in matrix form as

$$\begin{bmatrix}\Delta\overline{\omega}[r+1] \\ e[r+1]\end{bmatrix} = \underbrace{\begin{bmatrix}\beta & \overline{D}\alpha^{\mathrm{T}} \\ \kappa & I_m\end{bmatrix}}_{:=\Phi} \begin{bmatrix}\Delta\overline{\omega}[r] \\ e[r]\end{bmatrix} + \begin{bmatrix}-\overline{D} \\ 0_m\end{bmatrix} \sum_{i\in\mathcal{V}_\ell} \Delta\ell_i, \tag{10.8}$$

where $\Phi \in \mathbb{R}^{(m+1)\times(m+1)}$, $\beta := \overline{D}\sum_{i\in\mathcal{V}_g}\alpha_i\kappa_i$, and I_m and 0_m represent the m-dimensional identity matrix and all-zeros vector, respectively.

To ensure that the closed-loop system in (10.8) is stable, the α_i's and κ_i's specified above must be chosen such that the spectral radius of Φ lies on the boundary of or within the unit circle. More specifically, for marginal stability, we must choose the α_i's and κ_i's such that $\rho(\Phi) \leq 1$, i.e., $|\lambda_j| < 1$ for $j = 1, 2, \dots, m+1$ where λ_j is the j^{th} eigenvalue of Φ. Given such gains, we can compute the asymptotic value that $\begin{bmatrix}\Delta\overline{\omega}[r], e[r]\end{bmatrix}^{\mathrm{T}}$ takes, which we denote by $\begin{bmatrix}\Delta\overline{\omega}^{ss}, e^{ss}\end{bmatrix}^{\mathrm{T}}$. As $r \to \infty$, we have that $\begin{bmatrix}\Delta\overline{\omega}[r], e[r]\end{bmatrix}^{\mathrm{T}} = \begin{bmatrix}\Delta\overline{\omega}[r+1], e[r+1]\end{bmatrix}^{\mathrm{T}} = \begin{bmatrix}\Delta\overline{\omega}^{ss}, e^{ss}\end{bmatrix}^{\mathrm{T}}$, and we can rewrite (10.8) as

$$\begin{bmatrix}1-\beta & -\overline{D}\alpha^{\mathrm{T}} \\ -\kappa & 0_{m\times m}\end{bmatrix} \begin{bmatrix}\Delta\overline{\omega}^{ss} \\ e^{ss}\end{bmatrix} = \begin{bmatrix}-\overline{D} \\ 0_m\end{bmatrix} \sum_{i\in\mathcal{V}_\ell} \Delta\ell_i. \tag{10.9}$$

The following proposition establishes the criteria for choosing gains such that the closed-loop system is marginally stable and $\Delta\overline{\omega}^{ss} = 0$.

Proposition 2 (Stability of Closed-Loop System)
Consider the system in (10.8)*. If α_i and κ_i for $i \in \mathcal{V}_g$ are chosen such that $-2 < \beta = \overline{D}\sum_{i\in\mathcal{V}_g}\alpha_i\kappa_i < 0$, then the system is marginally stable and the average frequency error asymptotically approaches zero, i.e., $\rho(\Phi) \leq 1$ and $\Delta\overline{\omega}[r] \to 0$ as $r \to \infty$. Additionally, the value of the average frequency error at any round r is given by*

$$\Delta\overline{\omega}[r] = \frac{\sum_{i\in\mathcal{V}_\ell}\Delta\ell_i}{\sum_{i\in\mathcal{V}_p} D_i}(1+\beta)^r = \overline{D}(1+\beta)^r \left(\sum_{i\in\mathcal{V}_\ell}\Delta\ell_i\right). \tag{10.10}$$

Proof 2 (Proof for Proposition 2)
In order to choose α_i's and κ_i's such that $\rho(\Phi) \leq 1$, where Φ is defined in (10.8), we consider the characteristic equation of Φ given by $p(\lambda) = \det(\Phi - \lambda I_{m+1})$, where I_{m+1} denotes the $(m+1)$-dimensional identity matrix. Expanding the expression for $p(\lambda)$, we have

$$p(\lambda) = \det\left(\begin{bmatrix} \beta-\lambda & \overline{D}\alpha_1 & \overline{D}\alpha_2 & \cdots & \overline{D}\alpha_m \\ \kappa_1 & 1-\lambda & 0 & \cdots & 0 \\ \kappa_2 & 0 & 1-\lambda & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ \kappa_m & 0 & \cdots & 0 & 1-\lambda \end{bmatrix}\right). \tag{10.11}$$

By taking advantage of the structure of $\Phi - \lambda I_{m+1}$ and using its cofactor expansion along row 1, $p(\lambda)$ becomes

$$p(\lambda) = (\beta-\lambda)(1-\lambda)^m + \overline{D}\sum_{j=1}^{m}(-1)^j\kappa_j \det(M_j^m), \tag{10.12}$$

where M_j^m is the matrix $\Phi - \lambda I_{m+1}$ with the first row and $(j+1)^{th}$ column deleted. Using cofactor expansion along the first column of M_j^m, it can be shown that

$$\det(M_j^m) = (-1)^{j+1}\alpha_j(1-\lambda)^{m-1}. \tag{10.13}$$

Thus, the characteristic equation becomes

$$\begin{aligned} p(\lambda) =& (\beta-\lambda)(1-\lambda)^m + \overline{D}(1-\lambda)^{m-1}\sum_{j=1}^{m}(-1)^j(-1)^{j+1}\alpha_j\kappa_j, \\ =& (1-\lambda)^{m-1}\lambda[\lambda-(\beta+1)], \end{aligned} \tag{10.14}$$

from which it follows that $\lambda_j = 1$, $j = 1, 2, \ldots, m-1$, $\lambda_m = 0$, and $\lambda_{m+1} = \beta + 1$.

From the analysis above, it is clear that by appropriately choosing gains such that $-2 \leq \beta \leq 0$, all of the eigenvalues of Φ will lie on the boundary of or within the unit circle. However, for marginal stability, given that $\lambda = 1$ has algebraic multiplicity $m-1$, we must show that Φ is nondefective, i.e., we must show that the geometric multiplicity of $\lambda = 1$ is $m-1$. If we

consider the characteristic equation in (10.11) *for* $\lambda = 1$*, we see that since the rightmost* m *columns of the matrix* $\Phi - I_{m+1}$ *are linearly dependent, the rank of* $\Phi - I_{m+1}$ *is 2. Given the relationship between the rank and nullity of a matrix, it is clear that the* $m-1$ *eigenvectors associated with* $\lambda = 1$ *are linearly independent, which implies that* $\lambda = 1$ *has geometric multiplicity* $m-1$*.*

Given that Φ *is nondefective, we can write its Jordan form as* $\Phi = UJU^{-1}$ *where* $J := diag(\{\lambda_j\})$ *and* U *is a nonsingular matrix composed of columns corresponding to the eigenvectors of* Φ*. It then follows that we can write the value of* $\left[\Delta\overline{\omega}[r], e[r]\right]^{\mathrm{T}}$ *in matrix form as*

$$\begin{bmatrix}\Delta\overline{\omega}[r]\\ e[r]\end{bmatrix} = UJ^rU^{-1}\begin{bmatrix}\Delta\overline{\omega}[0]\\ e[0]\end{bmatrix} + \left[\sum_{s=0}^{r-1} UJ^sU^{-1}\right]\begin{bmatrix}-1\\ 0_m\end{bmatrix}\frac{\sum_{i\in\mathcal{V}_\ell}\Delta\ell_i}{\sum_{i=1}^n D_i}. \quad (10.15)$$

Furthermore, it can be shown using (10.15) *that the value of the average frequency error given the controller in* (10.6) – (10.7) *at any round* r *is given by*

$$\Delta\overline{\omega}[r] = \frac{\sum_{i\in\mathcal{V}_\ell}\Delta\ell_i}{\sum_{i=1}^n D_i}(1+\beta)^r. \quad (10.16)$$

From (10.16)*, we see that choosing gains such that* $\beta = -2$ *will result in oscillatory behavior from the closed loop system; similarly, choosing gains such that* $\beta = 0$ *will lead to constant frequency error, i.e.,* $\Delta\overline{\omega}[r] \equiv \overline{D}\sum_{i\in\mathcal{V}_\ell}\Delta\ell_i$*,* $r = 0, 1, 2, \ldots$*. Thus, to ensure that* $\Delta\overline{\omega}[r] \to 0$ *as* $r \to \infty$*, we must enforce strict inequalities on the value of* β*, i.e., we restrict to choosing gains such that* $-2 < \beta < 0$*.* □

10.4 Estimating Deviation from Phase-Cohesive Operation

Recall the criterion for phase cohesiveness in (9.5), which was defined for generator set-points and load demands u^* and ℓ^0, respectively. More generally, for any set-points and load demands, $u = \left[u_1, \ldots, u_m\right]^{\mathrm{T}}$ and $\ell =$

$\left[\ell_{m+1}, \ldots, \ell_n\right]^{\mathrm{T}}$, respectively, we can define the function

$$h^p(u, \ell) := \left\| M_p^{\mathrm{T}} L^{\dagger} \begin{bmatrix} u \\ -\ell \end{bmatrix} \right\|_{\infty}, \tag{10.17}$$

where the operating point that results from the injections at the generators and loads is phase cohesive if the value that the function $h^p(\cdot)$ takes is strictly smaller than $\sin(\phi)$ for $\phi \in [0, \pi/2)$.

Given set-points found to result in phase-cohesive operation subject to the initial power demanded by the loads, u^* and ℓ^0, respectively, we define $\Delta u := \left[\Delta u_1, \ldots, u_m\right]^{\mathrm{T}}$ and $\Delta \ell := \left[\Delta \ell_{m+1}, \ldots, \ell_n\right]^{\mathrm{T}}$ to be vectors representing the amount by which the generators and loads deviate from the initial set-points and demands, respectively. Then, to ensure the system remains phase cohesive as it evolves away from the operating point for which the u_i^*'s were found, we must have $h^p(u^* + \Delta u, \ell^0 + \Delta \ell) < \sin(\phi)$. In the limit as $\phi \to \pi/2$, we have that $h^p(u^* + \Delta u, \ell^0 + \Delta \ell) \to 1$, which implies that the maximum amount $h^p(u^* + \Delta u, \ell^0 + \Delta \ell)$ can increase to ensure phase cohesiveness is given by $1 - h^p(u^*, \ell^0)$. In the discussion that follows, we describe a method for estimating the value that $h^p(\cdot)$ takes as our control architecture adjusts the generator set-points in response to changes in load.

By inspecting the update rule in (10.6) and the values on which the incremental set-point adjustment for each generator depends, we see that $\Delta u_i[r] \propto \sum_{j \in \mathcal{V}_\ell} \Delta \ell_j$. More specifically, if we substitute the value of $\Delta \overline{\omega}[r]$ from (10.10) into (10.7), we see that the value of $e_i[r]$ at any round r for $i \in \mathcal{V}_g$ is given by

$$\begin{aligned} e_i[r] &= e_i[0] + \kappa_i \sum_{s=0}^{r-1} \Delta \overline{\omega}[s], \\ &= e_i[0] + \kappa_i \overline{D} \left(\sum_{i \in \mathcal{V}_\ell} \Delta \ell_i \right) \left(\sum_{s=0}^{r-1} (1+\beta)^s \right), \end{aligned} \tag{10.18}$$

and, given that $e_i[0] = 0$, $i \in \mathcal{V}_g$, we have that the incremental set-point of generator i at round r is given by

$$\Delta u_i[r] = \alpha_i \kappa_i \overline{D} \left(\sum_{j \in \mathcal{V}_\ell} \Delta \ell_j \right) \left(\sum_{s=0}^{r-1} (1+\beta)^s \right). \tag{10.19}$$

Thus, given that the $\Delta u_i[r]$'s depend on the load perturbations, we can define a new function,

$$\begin{aligned} h^l(\ell^0 + \Delta\ell) &:= h^p(u^* + \Delta u(\Delta\ell), \ell^0 + \Delta\ell), \\ &= \left\| M_p^{\mathrm{T}} L^\dagger \begin{bmatrix} u^* + \Delta u(\Delta\ell) \\ -(\ell^0 + \Delta\ell) \end{bmatrix} \right\|_\infty, \end{aligned} \tag{10.20}$$

that takes the value of the load demands as input, and takes the same value as $h^p(\cdot)$ evaluated at $u = u^* + \Delta u$ and $\ell = \ell^0 + \Delta\ell$ for the closed-loop system. If we define the gradient of $h^l(\cdot)$ as

$$\nabla h^l(\ell) = \left[\tfrac{\partial h^l(\ell)}{\partial \ell_{m+1}}, \quad \cdots \quad , \tfrac{\partial h^l(\ell)}{\partial \ell_n} \right]^{\mathrm{T}}, \tag{10.21}$$

it follows that, for small perturbations in load around ℓ^0, we can estimate (10.20) as

$$\begin{aligned} h^l(\ell^0 + \Delta\ell) &\approx h^l(\ell^0) + \nabla h^l(\ell)^{\mathrm{T}} \Big|_{\ell=\ell^0} \Delta\ell, \\ &= h^l(\ell^0) + \sum_{i \in \mathcal{V}_\ell} \frac{\partial h^l(\ell)}{\partial \ell_i} \Big|_{\ell=\ell^0} \Delta\ell_i. \end{aligned} \tag{10.22}$$

By maintaining an estimate for the value of $\nabla h^l(\ell)^{\mathrm{T}} \Big|_{\ell=\ell^0} \Delta\ell$ as the amount of power demanded by the loads changes, the point at which phase-cohesive set-points should be recomputed can be determined. More specifically, a recomputation of the u_i^*'s should be done when

$$\xi(c) := \frac{\displaystyle\sum_{i \in \mathcal{V}_\ell} \frac{\partial h^l(\ell)}{\partial \ell_i} \Big|_{\ell=\ell^0} \Delta\ell_i}{c} \tag{10.23}$$

exceeds unity, where $0 < c < \left[1 - h^l(\ell^0)\right]$ is a constant that can be adjusted to trade off the frequency with which phase-cohesive set-points are computed and the desired phase cohesiveness margin.

CHAPTER 11

DISTRIBUTED IMPLEMENTATION

As described in Chapter 10, our proposed frequency regulation architecture requires global information to be implemented. In particular, the following values on which the architecture relies require system-level information to be computed: the average frequency error, $\Delta\overline{\omega}[r]$; the value of $\beta = \overline{D}\sum_{i\in\mathcal{V}_g}\alpha_i\kappa_i$, for ensuring closed loop stability; the phase-cohesive set-points, u_i^*, $i \in \mathcal{V}_g$; and the value of $\nabla h^l(\ell)^{\mathrm{T}}\Big|_{\ell=\ell^0}\Delta\ell$ to determine $\xi(c)$. In this chapter, we describe how the three algorithms outlined in Chapter 3—the max- and ratio-consensus and feasible flow algorithms—can collectively enable the distributed computation of all these values. Beyond eliminating the need for global information, the distributed implementation we propose does not rely on time-sensitive measurements of the frequency or phase angle; instead, only the injection at each bus is required.

We begin the following discussion by outlining how ratio consensus can be used to compute the value of $\Delta\overline{\omega}[r]$ at each round and to compute gains that ensure the closed loop system is stable. Then, we describe how the feasible flow algorithm can be used to compute the set-points that satisfy (9.6) and, when combined with the max- and ratio-consensus algorithms, can enable the distributed computation of an estimate for the value of $\xi(c)$ as defined in (10.23). Finally, we discuss the timeline over which each of the necessary values is distributively computed.

11.1 Computing the Average Frequency Error

By inspection of (10.3), we see that the definition of the average frequency error at each round r is a ratio of sums of values known by each local processor.

In particular, if we define

$$x_i[r] = \begin{cases} u_i[r], & \text{if } i \in \mathcal{V}_g, \\ -\ell_i, & \text{if } i \in \mathcal{V}_\ell, \end{cases} \tag{11.1}$$

it is clear that $\Delta\overline{\omega}[r] = \frac{\sum_{i\in\mathcal{V}_p} x_i[r]}{\sum_{i\in\mathcal{V}_p} D_i}$. Thus, if we use an instance of ratio consensus at each round r, with the states maintained by node i denoted by $y_i^r[k]$ and $z_i^r[k]$, and initialize the states as $y_i^r[0] = x_i[r]$ and $z_i^r[0] = D_i$, it follows that the nodes can asymptotically compute the average frequency error, i.e., $\lim_{k\to\infty} \frac{y_i^r[k]}{z_i^r[k]} = \Delta\overline{\omega}[r]$, $i \in \mathcal{V}_c$.

11.2 Computing Stable Gains

From the analysis in Section 10.3, it can be shown that, in the limit as $r \to \infty$, if the generator set-points are updated according to the control scheme in (10.6) – (10.7), the steady-state value of generator i's set-point is given by

$$u_i^{ss} = u_i^* + \frac{\alpha_i\kappa_i}{\sum_{l\in\mathcal{V}_g} \alpha_l\kappa_l} \sum_{j\in\mathcal{V}_\ell} \Delta\ell_j, \tag{11.2}$$

which implies that the product of gains $\alpha_i\kappa_i$ affects the amount by which generator i will adjust its set-point away from u_i^* in order to meet the total incremental demand for load. Furthermore, from (10.10), we see that if we choose the α_i's and κ_i's such that $\beta := \sum_{i\in\mathcal{V}_g} \alpha_i\kappa_i = -1$, it follows that the frequency error that results from one or more changes in load can be eliminated after one round of our proposed control architecture. Thus, to ensure that $\beta = -1$, we see that the summation $-\sum_{i\in\mathcal{V}_p} D_i$ must be divided among the generators, and that the specific choice of gains will dictate the proportion of the total incremental demand attributed to each generator.

Let $\Delta\underline{u}_i = \underline{u}_i - u_i^*$ and $\Delta\overline{u}_i = \overline{u}_i - u_i^*$ be the lower and upper bounds on the amount by which generator i can adjust its set-point away from u_i^* without violating its output limits and define $\gamma_g := \frac{-\sum_{i\in\mathcal{V}_p} D_i - \sum_{i\in\mathcal{V}_g} \Delta\underline{u}_i}{\sum_{i\in\mathcal{V}_g} \Delta\overline{u}_i - \Delta\underline{u}_i}$. Then, if we choose $\alpha_i\kappa_i = h_i^g(\gamma_g)$, where

$$\begin{aligned} h_i^g(\gamma_g) :=& \Delta\underline{u}_i + \gamma_g(\Delta\overline{u}_i - \Delta\underline{u}_i), \\ =& \underline{u}_i - u_i^* + \gamma_g(\overline{u}_i - \underline{u}_i), \end{aligned} \quad i \in \mathcal{V}_g, \tag{11.3}$$

it is easy to see that $\sum_{i\in\mathcal{V}_g}\alpha_i\kappa_i = -\sum_{i\in\mathcal{V}_p} D_i$ and that the summation $-\sum_{i\in\mathcal{V}_p} D_i$ is divided among the generators proportionally to their incremental set-point limits.

Similar to the so-called *fair splitting* allocation scheme in [5], we can use ratio consensus to compute the value of γ_g. More specifically, given the dependence of the value of γ_g on the u_i^*'s, we can use an instance of ratio consensus each time the phase-cohesive set-points are computed. We denote the states maintained by node i for each of these instances by $y_i^g[k]$ and $z_i^g[k]$, and, if we initialize them as

$$y_i^g[0] = \begin{cases} -D_i - \Delta\underline{u}_i, & \text{if } i \in \mathcal{V}_g, \\ -D_i, & \text{if } i \in \mathcal{V}_\ell, \end{cases} \tag{11.4}$$

and

$$z_i^g[0] = \begin{cases} \overline{u}_i - \underline{u}_i, & \text{if } i \in \mathcal{V}_g, \\ 0, & \text{if } i \in \mathcal{V}_\ell, \end{cases} \tag{11.5}$$

it follows from (3.7) that $\lim_{k\to\infty} \frac{y_i^g[k]}{z_i^g[k]} = \gamma_g$. By using this value as the argument of the $h_i^g(\cdot)$ function defined in (11.3), each node can compute the product of its gains as $\alpha_i\kappa_i = h_i^g(y_i^g[k_0]/z_i^g[k_0])$, for sufficiently large k_0. Since there are no constraints on the individual α_i's and κ_i's, we choose gains such that $\alpha_i = h_i^g(y_i^g[k_0]/z_i^g[k_0])$ and $\kappa_i = 1$. (Note that if we use the approximate ratio consensus algorithm described in Section 3.1.4 to compute γ_g, the approximation error that results will prevent the value of β from exactly equaling -1; thus, in reality, it may take more than one round to drive the frequency error to zero.)

Remark 8

Although using the function in (11.3) *to assign gains ensures that* $\beta = -1$*, it only guarantees that the incremental set-point of every generator is within its respective incremental limits for small enough collective load perturbations,* $\sum_{i\in\mathcal{V}_\ell}\Delta\ell_i$*. However, given that the control scheme in* (10.6) – (10.7) *is designed to regulate the frequency for small perturbations away from a pre-determined phase-cohesive operating point, we can assume that the load perturbations are sufficiently small such that this choice of gain ensures all incremental set-points are within limits for the operating range in which they*

are used.

An alternative function for choosing the gains that ensures $\beta = -1$ and guarantees all incremental set-points are within limits is:

$$\tilde{h}_i^g(\tilde{\gamma}_g, \tilde{\gamma}_d) := -\tilde{\gamma}_d \left(\Delta \underline{u}_i + \tilde{\gamma}_g(\Delta \overline{u}_i - \Delta \underline{u}_i)\right), \quad i \in \mathcal{V}_g, \tag{11.6}$$

where $\tilde{\gamma}_g := \frac{\sum_{i\in\mathcal{V}_\ell} \Delta\ell_i - \sum_{i\in\mathcal{V}_g} \Delta\underline{u}_i}{\sum_{i\in\mathcal{V}_g} \Delta\overline{u}_i - \Delta\underline{u}_i}$ and $\tilde{\gamma}_d := \frac{\sum_{i\in\mathcal{V}_p} D_i}{\sum_{j\in\mathcal{V}_\ell} \Delta\ell_j}$. If we assume that the total amount by which the load deviates is within the collective incremental capacity of the generators, i.e., $\sum_{i\in\mathcal{V}_g} \Delta\underline{u}_i \leq \sum_{i\in\mathcal{V}_\ell} \Delta\ell_i \leq \sum_{i\in\mathcal{V}_g} \Delta\overline{u}_i$, this choice of gains will divide the incremental demand among the generators while also ensuring $\beta = -1$. Although this alternative function has advantages compared to $h^g(\cdot)$, it requires that each load know the value of $\Delta\ell_i$ in order to compute $\tilde{\gamma}_d$. Additionally, while ratio consensus can be used to compute $\tilde{\gamma}_d$, doing so requires another operation before the incremental set-points can be determined. □

11.3 Computing Phase-Cohesive Set-Points

By inspection, we see that ensuring the phase cohesiveness criterion in (9.5) is satisfied is equivalent to ensuring that all network flows are within limits. More specifically, when the vector of injections at the generator and load buses is pre-multiplied by the matrix product $M_p^{\mathrm{T}} L^\dagger$, the elements of the resultant vector are the network power flows, normalized by the susceptances of the branches connecting each pair of buses, i.e., the value of each element is equal to $\sin(\theta_i(t) - \theta_j(t))$ for $(i,j) \in \mathcal{E}_p$. To ensure phase-cohesive operation results, every normalized flow in the network must be strictly less than $\sin(\phi) \to 1$ as $\phi \to \pi/2$, i.e., $M_p^{\mathrm{T}} L^\dagger \left[u, -\ell\right]^{\mathrm{T}} < 1_{|\mathcal{E}_p|}$, where $1_{|\mathcal{E}_p|}$ is the $|\mathcal{E}_p|$-dimensional all-ones vector. Thus, we see that the problem of choosing phase-cohesive set-points subject to the power injections at the load buses is analogous to choosing generator outputs such that no branch power flow exceeds its limit.

From the discussion above, it follows that, if a solution to the feasible flow problem exists for given load power demands, we can use Algorithm 2 to find the generator set-points that result in an operating point that satisfies the feasibility problem in (9.6), i.e., the operating point is phase-cohesive, the

total demand for load is balanced by the collective generator set-points, and all the generator set-points are within limits. In order to enforce appropriate limits for the feasible flow algorithm that ensure phase cohesiveness, we see that, as $\phi \to \pi/2$ and $\sin(\phi) \to 1$, the maximum branch power flow on any given line is equal to its susceptance. Thus, if we enforce lower and upper limits on the flow along each edge in the feasible flow problem equal to $\underline{f}_e = 0$ and $\overline{f}_e = B_{(i,j)}$, $e = \{i,j\}, \{j,i\} \in \mathcal{E}_d$, $(i,j) \in \mathcal{E}_p$, respectively, we can find set-points that satisfy (9.6) using Algorithm 2 as follows. Let $\underline{u} := \begin{bmatrix} \underline{u}_1, \ldots, \underline{u}_m \end{bmatrix}^{\mathrm{T}}$ and $\overline{u} := \begin{bmatrix} \overline{u}_1, \ldots, \overline{u}_m \end{bmatrix}^{\mathrm{T}}$ be vectors representing the lower and upper set-point limits of the generators and define $B := [B_{(i,j)} : \{i,j\}, \{j,i\} \in \mathcal{E}_d, (i,j) \in \mathcal{E}_p] \in \mathbb{R}^{|E_d|}$; then, using the functional representation of Algorithm 2, i.e., $h^f(\cdot)$, with parameters $(0_{|\mathcal{E}_d|}, B, \underline{u}, \overline{u}, \ell^0)$, where $0_{|\mathcal{E}_d|}$ is the $|\mathcal{E}_d|$-dimensional all zeros vector, the generator outputs that result satisfy (9.6), i.e., $u^* = g^*$.

11.4 Computing Phase Cohesiveness Margin

In Section 10.4, we showed that by monitoring the value of $\xi(c)$ as the amount of power demanded by the loads changes, the nodes can estimate when to recompute phase-cohesive generator set-points. Recall from (10.23) that the sensitivity of the function $h^l(\cdot)$ to small perturbations in load, evaluated at ℓ^0, $\nabla h^l(\ell)\Big|_{\ell=\ell^0}$, is required to compute $\xi(c)$. By noting that the output of the function $h^l(\cdot)$ is the maximum normalized branch flow subject to injections at the generators and loads, i.e., given set-points $u^* + \Delta u(\Delta\ell)$ and demands $\ell^0 + \Delta\ell$,

$$\left\| M_p^{\mathrm{T}} L^{\dagger} \begin{bmatrix} u^* + \Delta u(\Delta\ell) \\ -(\ell^0 + \Delta\ell) \end{bmatrix} \right\|_{\infty} = \max_{(i,j)\in\mathcal{E}_p} \sin(\theta_i(t) - \theta_j(t)) \tag{11.7}$$

we can use the following procedure, which combines the feasible flow algorithm and max consensus, to distributively compute $\nabla h^l(\ell)\Big|_{\ell=\ell^0}$.

From the functional representation of Algorithm 2 defined in (3.41), we see that, in addition to the generator outputs g_i^*, $i \in \mathcal{V}_g$, the set of flows $\{f_{\{i,j\}}^{i*}, f_{\{j,i\}}^{i*} : \{i,j\}, \{j,i\} \in \mathcal{E}_d\}$ is also computed. Thus, if we use the max-consensus algorithm, with the value initially known by node i given by $\eta_i =$

$\max_{j\in\mathcal{N}_p(i)} \frac{|f^{i*}_{\{i,j\}} - f^{i*}_{\{j,i\}}|}{B_{(i,j)}}$, it follows that, without measurements of $\theta_i(t), i \in \mathcal{V}_p$, as implied by (11.7), we can compute the value of

$$\left\| M_p^{\mathrm{T}} L^{\dagger} \begin{bmatrix} g^* \\ -\ell \end{bmatrix} \right\|_{\infty} = \max_{(i,j)\in\mathcal{E}_p} \frac{|f^{i*}_{\{i,j\}} - f^{i*}_{\{j,i\}}|}{B_{(i,j)}} = \max_{l\in\mathcal{V}_c} \eta_l,$$

subject to the parameters passed to the $h^f(\cdot)$ function.

If we approximate the gradient of $h^l(\cdot)$ by

$$\nabla h^l(\ell) \approx \left[\frac{\Delta h^l(\ell)}{\Delta \ell_{m+1}}, \quad \cdots \quad , \frac{\Delta h^l(\ell)}{\Delta \ell_n} \right]^{\mathrm{T}}, \tag{11.8}$$

we can compute it using $2|\mathcal{V}_\ell|$-instances of the feasible flow and max-consensus algorithms. More specifically, given that we are interested in determining the value of $\xi(c)$ for the closed-loop system as it evolves away from the operating point for which the u_i^*'s were computed, we can approximate the sensitivity of $h^l(\cdot)$ to small changes in demand at load $i \in \mathcal{V}_\ell$, i.e., the value of $\left.\frac{\Delta h^l(\ell)}{\Delta \ell_i}\right|_{\ell=\ell^0}$, subject to our proposed control architecture, as follows. Define $v_i \in \mathbb{R}^{|\mathcal{V}_p|}$ to be a vector with 1 in the i^{th} coordinate and 0's elsewhere; then, for each $i \in \mathcal{V}_\ell$, let $\ell(\epsilon v_i) := \ell^0 + \epsilon v_i$ be a vector of load demands with the i^{th} load perturbed by the value ϵ. From (11.2), let

$$u_j(\epsilon) := u_j^* + \epsilon \alpha_j \kappa_j / \sum_{l\in\mathcal{V}_g} \alpha_l \kappa_l \tag{11.9}$$

be the set-point of generator $j \in \mathcal{V}_g$ given one such load perturbation, subject to the control scheme in (10.6) – (10.7). Furthermore, let $f_e^*(\epsilon v_i)$, $e \in \mathcal{E}_d$ be the flows that result from Algorithm 2 with parameters $(0_{|\mathcal{E}_d|}, B, u(\epsilon), u(\epsilon), \ell(\epsilon v_i))$, where $u(\epsilon) := \Big[u_1(\epsilon), \ldots, u_m(\epsilon)\Big]^{\mathrm{T}}$. Then, we can use an instance of the max-consensus algorithm, where the value known by node j is

$$\eta_j^i(\epsilon) = \max_{l\in\mathcal{N}_p(j)} \frac{|f^{j*}_{\{j,l\}}(\epsilon v_i) - f^{j*}_{\{l,j\}}(\epsilon v_i)|}{B_{(j,l)}}, \tag{11.10}$$

to compute the value of

$$h^l(\ell(\epsilon v_i)) = \max_{(j,l)\in\mathcal{E}_p} \frac{|f^{j*}_{\{j,l\}}(\epsilon v_i) - f^{j*}_{\{l,j\}}(\epsilon v_i)|}{B_{(j,l)}} = \max_{j\in\mathcal{V}_c} \eta_j^i(\epsilon).$$

If we use the central difference approximation to estimate the value of $\left.\frac{\Delta h^l(\ell)}{\Delta \ell_i}\right|_{\ell=\ell^0}$, then it follows that

$$\left.\frac{\Delta h^l(\ell)}{\Delta \ell_i}\right|_{\ell=\ell^0} \approx \frac{h^l(\ell(\epsilon v_i)) - h^l(\ell(-\epsilon v_i))}{2\epsilon}, \tag{11.11}$$

where $h^l(\ell(-\epsilon v_i))$ is computed analogously to $h^l(\ell(\epsilon v_i))$ with $\ell(-\epsilon v_i) = \ell^0 - \epsilon v_i$ and $u_j(-\epsilon) = u_j^* - \epsilon\alpha_j\kappa_j / \sum_{l\in\mathcal{V}_g} \alpha_l\kappa_l$.

Given that the value of $\xi(c)$ depends on an appropriate choice of c, which depends on the value of $h^l(\ell)$ for $\ell = \ell^0$, we can use the process described above for computing the individual $h^l(\ell(\epsilon v_i))$'s to determine a value of c that is strictly less than $1 - h^l(\ell^0)$. By using max consensus to compute $h^l(\ell^0)$, which can be combined with a pre-determined local rule for choosing c, the node with the maximizing flow can also be determined, i.e., if $\eta_i = \max_{j\in\mathcal{N}_p(i)} \frac{|f^{i*}_{\{i,j\}} - f^{i*}_{\{j,i\}}|}{B_{(i,j)}}$, the node $l = \underset{\{\eta_j\,:\,j\in\mathcal{V}_c\}}{\text{argmax}}\ \eta_j$ is the one with the maximizing flow. Then, by combining the appropriately chosen value of c with the approximation to the sensitivity of the function $h^l(\cdot)$ found using the process described above, the nodes can compute the value of $\xi(c)$ as follows.

Let $y_i^p[k]$ and $z_i^p[k]$ denote the states maintained by node i, where $y_i^p[0] = \left.\frac{\Delta h^l(\ell)}{\Delta \ell_i}\right|_{\ell=\ell^0} \Delta\ell_i$ if $i \in \mathcal{V}_\ell$, and $y_i^p[0] = 0$ otherwise; and $z_i^p[0] = c$ if $i = \underset{\{\eta_j\,:\,j\in\mathcal{V}_c\}}{\text{argmax}}\ \eta_j$ and $z_i^p[0] = 0$ otherwise. Then, by updating the states according to (3.4) – (3.5), the nodes can asymptotically obtain

$$\lim_{k\to\infty} \frac{y_i^p[k]}{z_i^p[k]} = \frac{\displaystyle\sum_{i\in\mathcal{V}_\ell} \left.\frac{\Delta h^l(\ell)}{\Delta \ell_i}\right|_{\ell=\ell^0} \Delta\ell_i}{c}.$$

11.5 Timeline

In the preceding discussion, we showed how each of the values necessary to implement our proposed frequency regulation architecture can be acquired using a combination of distributed algorithms. Next, we provide an overview of the timeline over which our proposed distributed implementation operates, specifically discussing the order in which the algorithms must be executed to

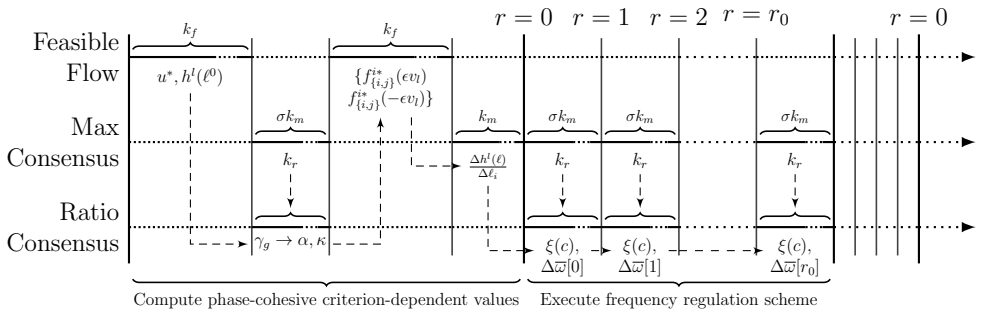


Figure 11.1: Timeline of distributed implementation of frequency regulation architecture.

properly compute the necessary values.

As discussed in Section 10.2, at each round of our proposed frequency regulation architecture, in addition to the weighted average frequency error, $\Delta\overline{\omega}[r]$, the output of each generator $i \in \mathcal{V}_g$ is adjusted according to its phase-cohesive set-point, u_i^*, and gains, α_i and κ_i, all of which depend on global information. Additionally, as discussed in Section 10.4, to ensure the system remains phase cohesive for large changes in load, the nodes monitor the value of $\xi(c)$, the definition of which depends on $\left.\frac{\Delta h^l(\ell)}{\Delta \ell_i}\right|_{\ell=\ell^0}$, $i \in \mathcal{V}_\ell$, as the system evolves. Given that these quantities—u_i^*, α_i, κ_i, $i \in \mathcal{V}_g$ and $\left.\frac{\Delta h^l(\ell)}{\Delta \ell_i}\right|_{\ell=\ell^0}$, $i \in \mathcal{V}_\ell$, which we collectively refer to as phase-cohesive criterion-dependent values—must be known to compute the generator set-points and $\xi(c)$, it is clear that they need to be determined before our proposed frequency regulation scheme can be executed. Moreover, from (11.3), we see that, in order to compute the $\left.\frac{\Delta h^l(\ell)}{\Delta \ell_i}\right|_{\ell=\ell^0}$'s, the controller gains, α_i and κ_i, $i \in \mathcal{V}_g$, must be known beforehand. Similarly, from (11.9), we see that the u_i^*'s must be known in order to compute the α_i's and κ_i's. Thus, it is clear that the phase cohesive criterion-dependent values must be computed in a specific order; we provide a detailed description of this order next.

Prior to operation and immediately following the round for which the value of $\xi(c)$ is found to exceed unity, i.e., before round $r = 0$, the nodes use the feasible flow algorithm to compute the phase-cohesive set-points, u^*. The generator nodes then use the u_i^*'s to determine their incremental limits, $\Delta\underline{u}_i$ and $\Delta\overline{u}_i$, $i \in \mathcal{V}_g$ and, together with the processors located at the load buses, use ratio consensus to determine the value of γ_g. With the value of γ_g,

or an approximation thereof, each generator processor then determines its controller gains, α_i and κ_i, according to the function $h_i^g(\cdot)$. Using $2|\mathcal{V}_\ell|$-instances of the feasible flow algorithm, the nodes then determine the sets of flows $\{f^{i*}_{\{i,j\}}(\epsilon v_l) : l \in \mathcal{V}_\ell, \{i,j\} \in \mathcal{E}_d\}$ and $\{f^{i*}_{\{i,j\}}(-\epsilon v_l) : l \in \mathcal{V}_\ell, \{i,j\} \in \mathcal{E}_d\}$, where $f^{i*}_{\{i,j\}}(\epsilon v_l)$ and $f^{i*}_{\{i,j\}}(-\epsilon v_l)$ are the flow assignments that result when load $l \in \mathcal{V}_\ell$ is perturbed by ϵ and $-\epsilon$, respectively, and the set-point of generator $j \in \mathcal{V}_g$ is lower- and upper-bounded to be $u(\epsilon)$ and $u(-\epsilon)$, respectively. From the sets of flows that result, the nodes can use max consensus to determine the values of $h^l(\ell(\epsilon v_l))$ and $h^l(\ell(-\epsilon v_l))$ for $l \in \mathcal{V}_\ell$, with which the load processors can compute an estimate of $\frac{\Delta h^l(\ell)}{\Delta \ell_l}\Big|_{\ell=\ell^0}$. After the phase-cohesive set-points, generator gains, and load sensitivity estimates are computed, the frequency regulation scheme can begin operation, with the nodes using separate instances of ratio consensus to compute the values of $\xi(c)$ and $\Delta\overline{\omega}[r]$ at each round. An overview of the order in which each value is computed is illustrated by the timeline in Fig. 11.1.

CHAPTER 12

SIMULATION RESULTS

In this chapter, we present numerical simulation results for three test cases in which our proposed control architecture and its distributed implementation are used. We begin this chapter by briefly describing software that was written to allow for numerical simulation of a microgrid and our proposed controller. Then, for each of the three cases, we demonstrate close-loop operation given a series of small perturbations to the loads and discuss several metrics that illustrate its effectiveness. We consider a six-bus ring network for the first two test cases; the network consists of three generators and three loads. For the third case, we utilize a tree network consisting of 37-buses, 15 of which are generators, and 22 of which are loads. In all cases, we make use of the finite-time ratio-consensus algorithm described in Section 3.1.4.

12.1 Numerical Simulation Software

Freely available software packages such as Power System Toolbox (PST) [64] allow for dynamic simulation of power systems. However, they are designed to account for myriad end-user scenarios which necessitates complexity that increases the difficulty of expanding upon or interfacing with them. Thus, in order to allow for the flexibility needed to simulate the microgrid model and proposed control architecture discussed herein, we developed a numerical simulator which we describe in brief next.

As written, the microgrid model in (2.1) – (2.4) is a differential-algebraic model. Even after the simplifications in Section 2.1.2 and Section 9.2 to reduce the generator model to a form that more closely resembles an inertia-less source, algebraic constraints will still be present if we wish to include heterogeneous generation resources. While the final model in (9.2) – (9.3) is purely dynamic, the simulation software written for this work was designed

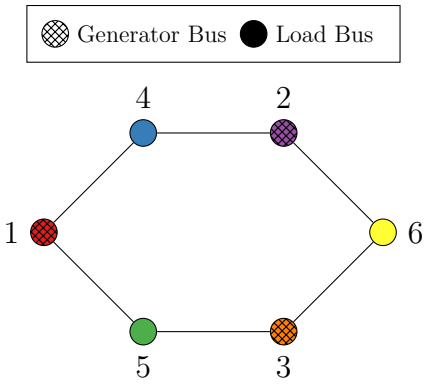


Figure 12.1: Graph-theoretic physical layer model, $\mathcal{V}_p$, for six-bus microgrid.

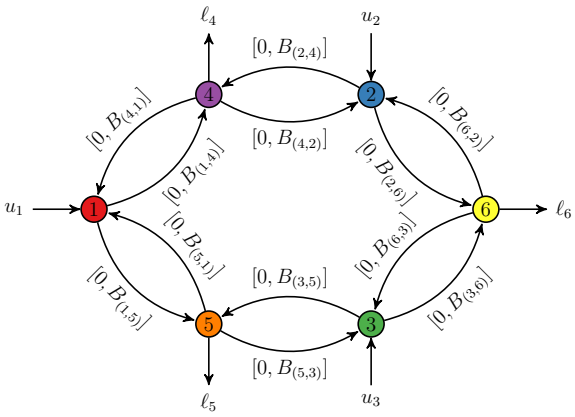
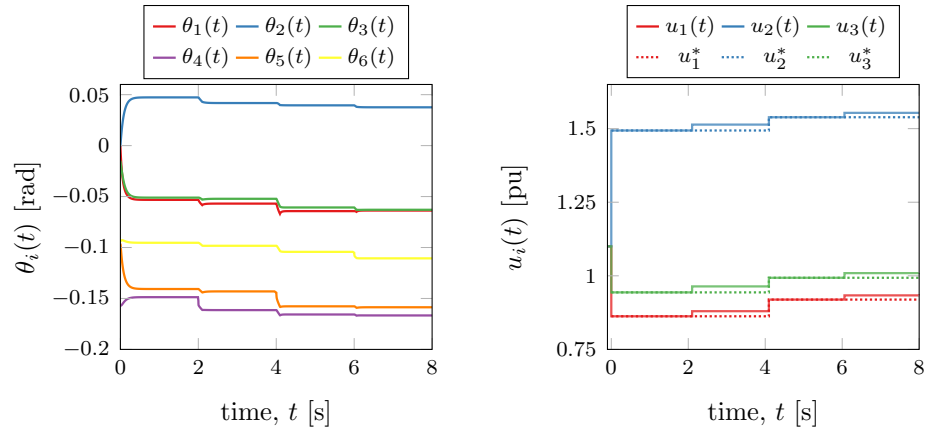


Figure 12.2: Graph-theoretic network flow model, $\mathcal{V}_d$, for six-bus microgrid.

more generally to allow for multiple generation types.

In order to simulate the differential-algebraic model representing the microgrid, we utilized the partitioned explicit method (see, e.g., [42]), combining the Newton-Rhapson method to solve the load flow algebraic equations with the fourth-order Runge-Kutta explicit method to perform the numerical integration. To allow for accurate simulation of the control architecture and its distributed implementation, we built upon the NetworkX [65] library to model the graph representing the exchange of information between nodes and used a fixed time step for the numerical methods so as to accurately account for the time needed for the distributed algorithms to execute.



(a) Bus voltage angles.

(b) Generator set-points.

Figure 12.3: Case I simulation results with loads ℓ_4, ℓ_5, and ℓ_6 perturbed at $t = 2$ s, $t = 4$ s, and $t = 6$ s, respectively.

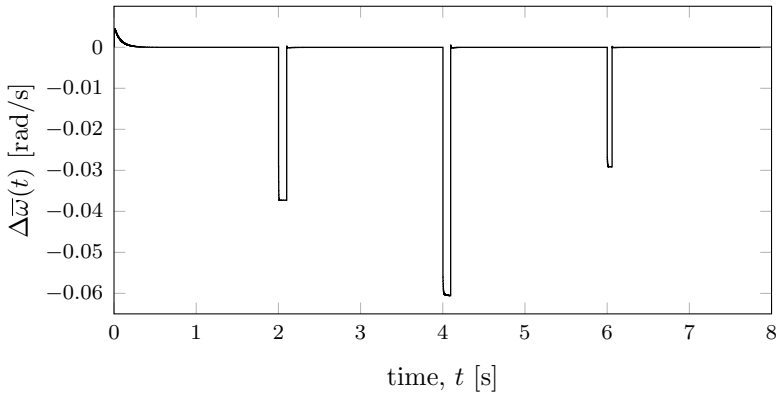


Figure 12.4: Weighted average frequency error for Case I.

12.2 Cases I and II: Six-Bus Ring Network

We consider a six-bus microgrid, the physical layer model of which is illustrated by the graph in Fig. 12.1. To provide an example of the mapping discussed in Section 3.2.1, an illustration for the directed graph used to model network flows for the feasible flow algorithm is shown in Fig. 12.2. Aside from the load perturbations, which are discussed in more depth later, the generator (Table B.1), load (Table B.2), simulation (Table B.4), and network (Table B.3) parameters are the same in Cases I and II. From the generator and load parameter tables, it can be seen that, for both cases, the total power initially demanded by the loads is $\sum_{i\in\mathcal{V}_\ell} \ell_i^0 = 3.3$ pu and that at time $t = 0^-$, prior to the computation of u_i^*'s, the generators equally share

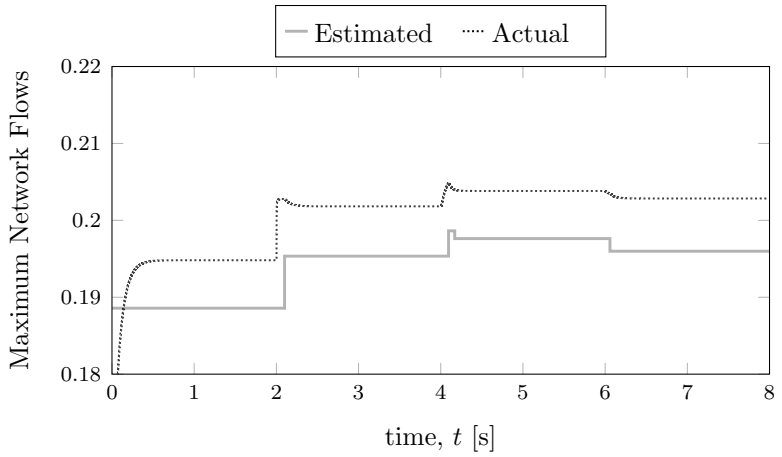


Figure 12.5: Estimated vs. actual maximum flow for Case I.

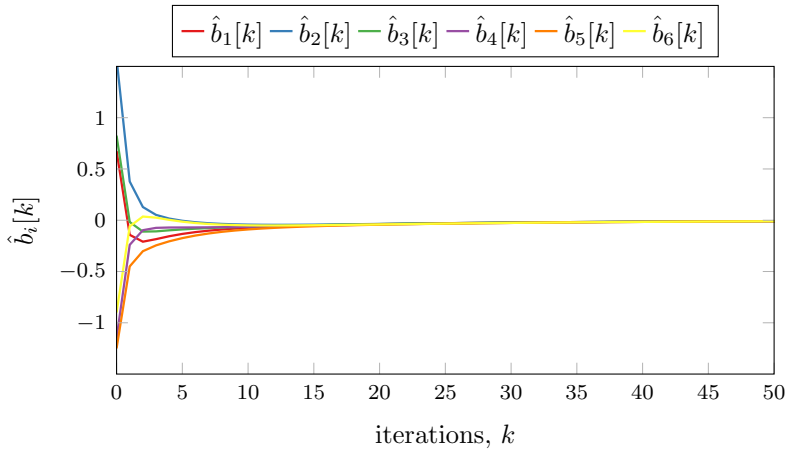
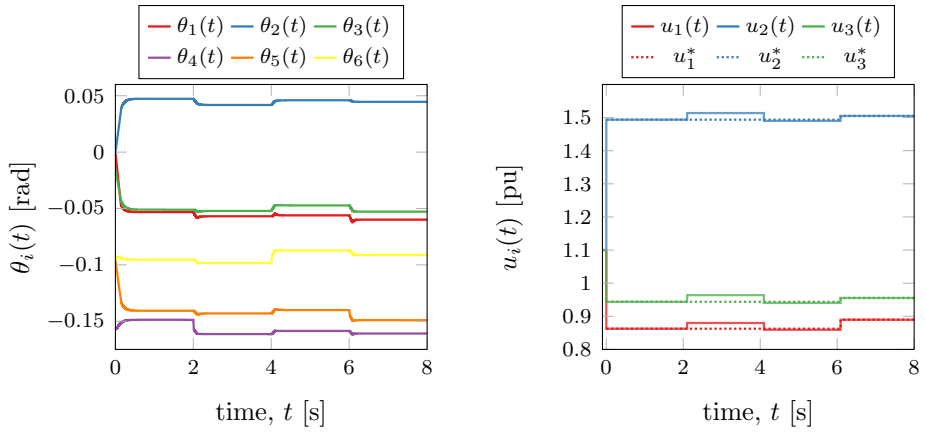


Figure 12.6: Nodal balance estimates as feasible flow algorithm evolves for initial computation of u^* for Case I.

the load such that $u_i[0^-] = 1.1$ pu, $i = 1, 2, 3$. Furthermore, the sensitivities of the function $h^l(\cdot)$ to perturbations in each load at the initial load values, ℓ^0, as computed at $t = 0$ are listed in Table 12.1. From the table, it can be seen that the sensitivity of the function $h^l(\cdot)$ to changes in load ℓ_6 is negative, i.e., a decrease in the power demanded by ℓ_6 will lead to an increase in the value of $h^l(\cdot)$.

12.2.1 Case I

For this case, we consider the closed-loop response of the six-bus microgrid subject to an increase in power demand by loads ℓ_4, ℓ_5, and ℓ_6 at time $t = 2$,



(a) Bus voltage angles.

(b) Generator set-points.

Figure 12.7: Case II simulation results with loads ℓ_4, ℓ_6, and ℓ_5 perturbed at $t = 2$ s, $t = 4$ s, and $t = 6$ s, respectively.

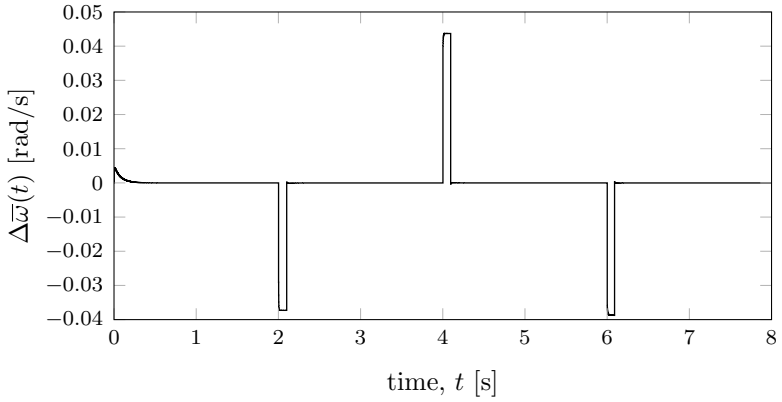


Figure 12.8: Weighted average frequency error for Case II.

$t = 4$, and $t = 6$, respectively. Each load demand is perturbed by an increase of at most 7.5% of its original value, with ℓ_4 perturbed by 5% such that $\Delta\ell_4 = 0.0575$ pu, ℓ_5 perturbed by 7.5% such that $\Delta\ell_5 = 0.09375$ pu, and ℓ_6 perturbed by 5% such that $\Delta\ell_6 = 0.045$ pu. Given these perturbations and the choice of $c = 0.01$, our proposed control architecture triggers a recomputation of the u_i^*'s at time $t \approx 4$ s, immediately after load ℓ_5 is perturbed. The response of the closed-loop system to the three perturbations and the recomputation of u^* is illustrated by the plots in Fig. 12.3 and 12.4. From Fig. 12.3a and Fig. 12.4, it can be seen that, following each load perturbation, the voltage angles stabilize and the weighted average frequency error that results from the changes in load is quickly eliminated. Additionally, from Fig. 12.3b, in which the generator set-points and their respective values

of u_i^* are illustrated, it can be seen that the generators increase their output according to the control scheme in (10.6) – (10.7) at times $t = 2$ s and $t = 6$ s and that new values of u^* are computed and applied following the perturbation to load ℓ_5 at $t = 4$ s. In addition to computing new u_i^*'s, the sensitivities of $h^l(\cdot)$ with respect to each load are also recomputed at $t = 4$ s, the values of which are provided in Table 12.2.

To demonstrate the effectiveness of our proposed method for estimating the value of the function $h^l(\cdot)$ as the loads are perturbed, the actual maximum normalized flow as computed at each time step, i.e., $\max_{(i,j)\in\mathcal{E}_p} \sin(\theta_i(t) - \theta_j(t))$, and its estimate, i.e., $h^l(\ell^0) + \nabla h^l(\ell)\Big|_{\ell=\ell^0}$, are shown in Fig. 12.5. From the figure, we see that the estimated maximum flow closely tracks the true value, with an error on the order of 0.001 throughout the simulation period. Additionally, we see that the estimate properly predicts that an increase in demand by load ℓ_6 will lead to a decrease in the maximum network flow. Finally, we illustrate the evolution of the nodal flow balances maintained by each node for the first 50 iterations of the feasible flow algorithm as it is used to compute the initial u_i^*'s in Fig. 12.6. From this figure, it can be seen that the flow balances maintained by all nodes quickly approach zero, with every node's estimate being within 0.001 of zero within the first 50 iterations.

12.2.2 Case II

For this case, we again consider the six-bus microgrid, but, in order to demonstrate the negative sensitivity of the function $h^l(\cdot)$ to changes in demand at load ℓ_6, we evaluate its closed-loop response to increases in demand at loads ℓ_4 and ℓ_5, and a decrease in demand at load ℓ_6. Specifically, the loads are perturbed as follows: ℓ_4 is increased by 5% at $t = 2$ s such that $\Delta\ell_4 = 0.0575$ pu, ℓ_6 is decreased by 7.5% at $t = 4$ s such that $\Delta\ell_6 = -0.0675$ pu, and ℓ_5 increased by 4.8% at $t = 6$ s such that $\Delta\ell_5 = 0.06$ pu. Given these perturba-

Table 12.1: Initial load sensitivities for Cases I and II.

i	4	5	6
ℓ_i^0	1.15	1.25	0.90
$\frac{\Delta h^l(\ell)}{\Delta \ell_i}\Big\vert_{\ell=\ell^0}$	0.1177	0.0351	−0.0342

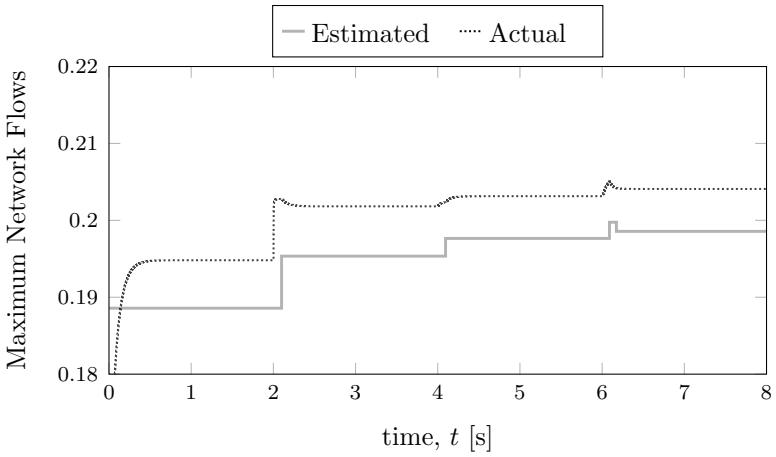


Figure 12.9: Estimated vs. actual maximum flow for Case II.

tions and the choice of $c = 0.01$, our proposed control architecture triggers a recomputation of the u_i^*'s (and the load sensitivities) at time $t \approx 6$ s, immediately after load ℓ_5 is perturbed.

The response of the closed-loop system to the three perturbations and the recomputation of u^* is illustrated by the plots in Fig. 12.7 and 12.8. As in Case I, Fig. 12.7a and Fig. 12.8, illustrate that, following each load perturbation, the voltage angles stabilize and the weighted average frequency error that results from the changes in load is quickly eliminated. Additionally, Fig. 12.7b illustrates the generator set-points and their respective u_i^*'s as the control architecture responds to the load perturbations. The figure illustrates that the third load perturbation, at time $t = 6$ s, increases the estimate for $h^l(\cdot)$ such that, given the value of c, a recomputation of the u_i^* is performed. Along with the recomputed generator set-points, the sensitivities of $h^l(\cdot)$ for each load are computed, the values of which are provided in Table 12.3. As before, the evolution of the estimated and actual maximum flows as computed throughout the simulation time are shown in Fig. 12.9. Similar to the

Table 12.2: Load sensitivities and perturbed demands for Case I after u^* is recomputed at $t = 4$ s.

i	4	5	6
$\ell_i^0 + \Delta\ell_i$	1.2363	1.3438	0.9000
$\frac{\Delta h^l(\ell)}{\Delta \ell_i}\Big\vert_{\ell=\ell^0+\Delta\ell}$	0.1153	0.0331	−0.0364

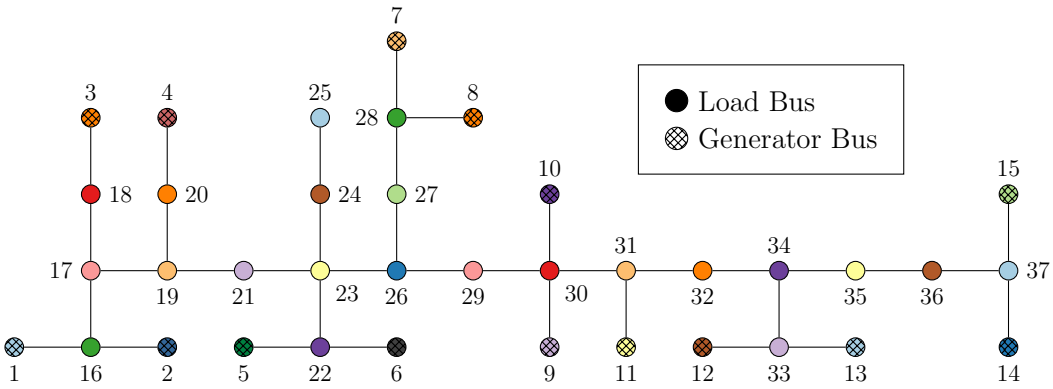


Figure 12.10: Graph-theoretic model of physical layer of 37-bus microgrid.

results from Case I, the figure shows that the estimate closely tracks the true maximum flow, with error on the order of 0.001.

12.3 Case III: 37-Bus Tree Network

For this case, we consider the closed loop response of the 37-bus tree network, the physical layer graph theoretic model for which is illustrated in Fig. 12.10. The parameters for the generators, the loads, the network, and the simulation are provided in Tables B.5, B.6, B.7, and B.8, respectively. From the generator and load parameter tables, it can be seen that, for both cases, the total power initially demanded by the loads is $\sum_{i\in\mathcal{V}_\ell} \ell_i^0 = 28.92$ pu and that at time $t = 0^-$, prior to the computation of u_i^*'s, the generators equally share the load such that $u_i[0^-] = 1.928$ pu, $i = 1, \ldots, 15$. Furthermore, the sensitivities of the function $h^l(\cdot)$ to perturbations in each load at the initial load values, ℓ^0, as computed at $t = 0$ are listed in Table 12.4.

To demonstrate the closed-loop response of our proposed frequency reg-

Table 12.3: Load sensitivities and perturbed demands for Case II after u^* is recomputed at $t = 6$ s.

i	4	5	6
$\ell_i^0 + \Delta\ell_i$	1.2075	1.3100	0.8325
$\frac{\Delta h^l(\ell)}{\Delta \ell_i}\Big\vert_{\ell=\ell^0+\Delta\ell}$	0.1163	0.0339	−0.0354

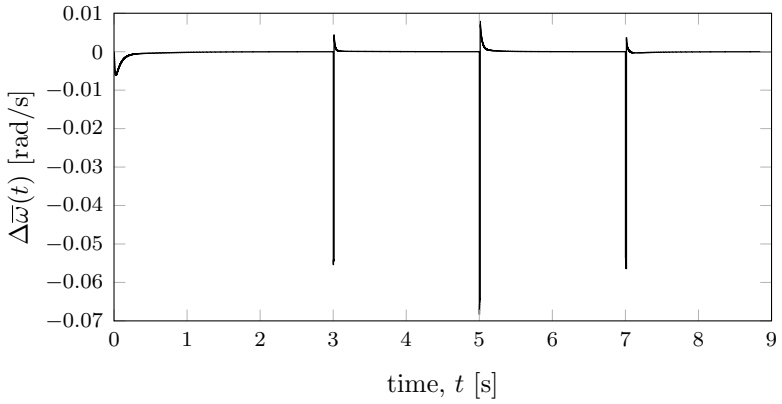


Figure 12.11: Weighted average frequency error for Case III.

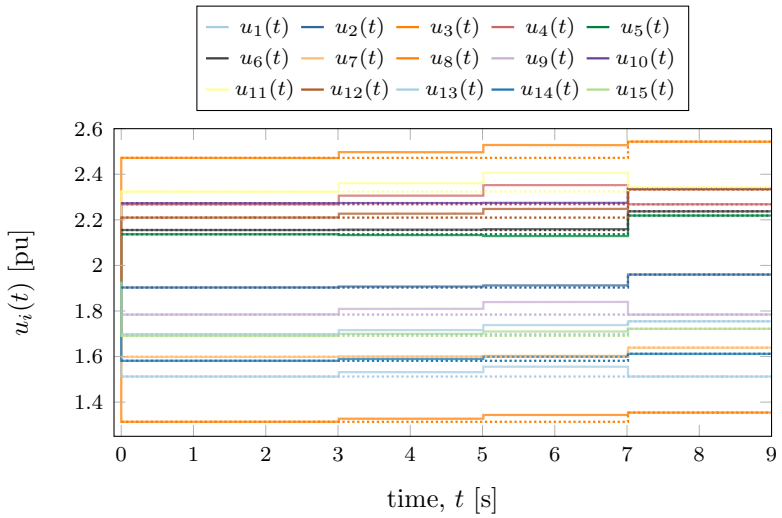


Figure 12.12: Generator set-points for Case III.

ulation architecture using the 37-bus network, we increase the amount of power demanded by loads ℓ_{20}, ℓ_{25}, and ℓ_{33} by 15% at $t = 3$ s, $t = 5$ s, and $t = 7$ s, respectively, such that $\Delta\ell_{20} = 0.2143$ pu, $\Delta\ell_{25} = 0.2624$ pu, and $\Delta\ell_{33} = 0.2195$ pu. Given these load perturbations and the choice of $c = 0.01$, a recomputation of the u_i^*'s and the sensitivities of $h^l(\cdot)$ to changes in load is performed shortly after load ℓ_{33} is perturbed at $t \approx 7$ s, with the new sensitivities listed in Table 12.4b. The response of the closed-loop system is illustrated by the plots in Fig. 12.11 and 12.12. Specifically, the evolution of the weighted average frequency error is shown in Fig. 12.11, from which it can be seen that the frequency error that results from the load perturbations is quickly eliminated. Additionally, the generator set-points and respective u_i^*'s (illustrated by the dotted traces) are shown in Fig. 12.12. From the

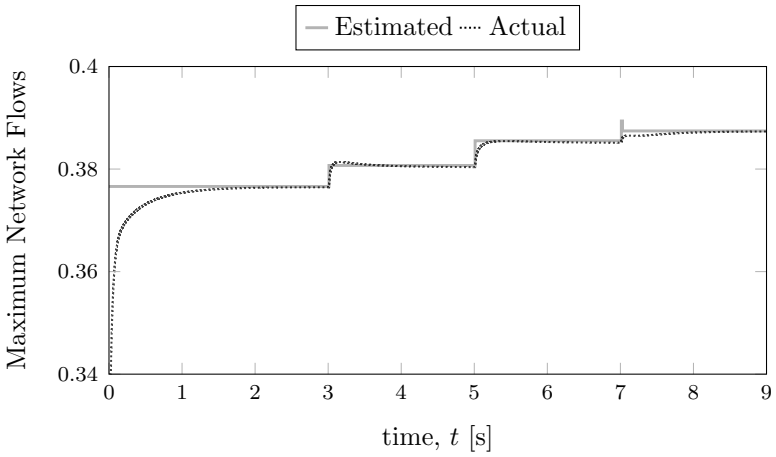


Figure 12.13: Estimated vs. actual maximum flow for 37-bus microgrid.

figure, it can be seen that the amount by which the generators deviate away from the originally computed u_i^*'s is relatively small, and that the new u_i^*'s are computed at $t \approx 7$ s after load ℓ_{33} is perturbed.

As in the six-bus cases, we illustrate the evolution of the estimated and computed values for the maximum flow throughout the simulation in Fig. 12.13. Similar to the smaller network cases, the estimated value of $h^l(\cdot)$ closely tracks the true computed value, with errors on the order of 0.0001 for the entire 9-second duration. Additionally, to demonstrate the feasible flow algorithm as used to compute the initial u_i^*'s for the larger network, the first 200 iterations of the evolution of the nodal flow estimates, i.e., the $\hat{b}_i[k]$'s, are shown in Fig. 12.14. While the larger network necessitates executing the algorithm for longer, the figure shows that the flow balances quickly tend toward zero, all to within an error 0.001 by the first 200 iterations.

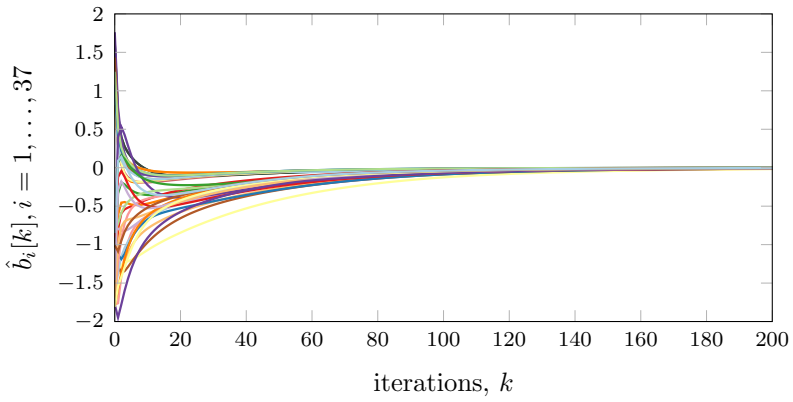


Figure 12.14: Nodal balance estimates as feasible flow algorithm evolves for initial computation of u^* for 37-bus microgrid.

Table 12.4: Load sensitivities for 37-bus microgrid.

(a) At initial load values, ℓ^0.

i	16	17	18	19	20	21	22	23	24	25	26
ℓ_i^0	1.5401	1.7809	1.7568	1.2087	1.4288	0.7858	0.9459	1.3387	0.9101	1.7496	0.8415
$\frac{\Delta h^1(\ell)}{\Delta \ell_i}\Big\vert_{\ell=\ell^0}$	0.0194	0.0193	0.0194	0.0191	0.0191	0.0187	0.0183	0.0184	0.0184	0.0184	0.0183

i	27	28	29	30	31	32	33	34	35	36	37
ℓ_i^0	0.8165	0.8020	1.6957	0.9086	1.6969	1.8038	1.4634	1.7792	1.8099	1.0085	0.8491
$\frac{\Delta h^1(\ell)}{\Delta \ell_i}\Big\vert_{\ell=\ell^0}$	0.0183	0.0183	0.0181	0.0180	0.0178	0.0177	0.0176	0.0176	0.0176	0.0175	0.0175

(b) At perturbed load values, $\ell^0 + \Delta\ell$.

i	16	17	18	19	20	21	22	23	24	25	26
$\ell_i^0 + \Delta\ell_i$	1.5401	2.0480	1.7568	1.2087	1.4288	0.7858	0.9459	1.3387	0.9101	2.0120	0.9677
$\frac{\Delta h^1(\ell)}{\Delta \ell_i}\Big\vert_{\ell=\ell^0+\Delta\ell}$	0.0214	0.0213	0.0214	0.0211	0.0211	0.0208	0.0203	0.0204	0.0204	0.0204	0.0203

i	27	28	29	30	31	32	33	34	35	36	37
$\ell_i^0 + \Delta\ell_i$	0.8165	0.8020	1.6957	0.9086	1.6969	1.8038	1.4634	1.7792	1.8099	1.0085	0.8491
$\frac{\Delta h^1(\ell)}{\Delta \ell_i}\Big\vert_{\ell=\ell^0+\Delta\ell}$	0.0203	0.0203	0.0202	0.0200	0.0199	0.0198	0.0197	0.0197	0.0197	0.0196	0.0196

CHAPTER 13

CONCLUDING REMARKS

In this chapter, we provide some concluding remarks about the work presented in each of the three parts of this dissertation. We close by providing some ideas for future work.

13.1 Part I

In Part I we outlined models to represent the physical and cyber layers of a microgrid and the interconnections between local processors needed to distributively implement the control architectures we proposed in Parts II and III. We also outlined three algorithms that were used as primitives for the distributed control architectures proposed herein. Additionally, we outlined some modifications to the so-called ratio-consensus algorithm that enables the computation of an approximation of the asymptotic value to which it converges in finite time. We also introduced an algorithm for distributively determining generator outputs and the resultant network flows such that the total load power demands are balanced and no line flow limits are exceeded.

13.2 Part II

In Part II, we proposed a distributed architecture for generation control in islanded ac microgrids. While microgrids are smaller and have lower ratings than, for example, bulk power transmission systems, the control objectives are similar; thus, the control functions of our architecture were derived from the three control functions provided by generation control architectures commonly adopted in large power systems. These functions are (i) droop control, (ii) frequency regulation, and (iii) optimal dispatch.

While droop control is completely decentralized, the implementation of the frequency regulation and optimal dispatch functions is typically centralized. However, by relying on local measurements, information obtained from neighboring generating units, and simple computations, we are able to achieve the same control objectives as those achieved by a centralized implementation. Compared to centralized ones, our distributed control approach can more easily adapt to changes, allowing the system to operate regardless of additions or removals of generating units.

A major component of this research was to experimentally verify the effectiveness of the proposed control architecture. To this end, we built an experimental microgrid comprised of several small synchronous generators and resistive loads all connected in a ring network. In this microgrid, the exchange of information among generating units was achieved via a wireless communications network which enabled the implementation of our distributed generation control algorithms for frequency regulation and optimal dispatch. We utilized this microgrid to verify the performance of these algorithms under numerous scenarios, including a case in which one of the generators, which was acting as spinning reserve, was able to come online after detecting that the collective power output limits were exceeded.

13.3 Part III

In Part III, we introduced a control architecture suitable for regulating the frequency in an islanded ac microgrid with no inertia. The approach we proposed is designed to regulate the average frequency subject to small load perturbations by adjusting the output of the generators in the system around set-points that are known to result in phase-cohesive operation. To handle larger perturbations in the loads, we also proposed a method that enables the computation of an estimate for the amount by which the system has deviated from the phase-cohesive operating point; the controller monitors this estimate as the system evolves to determine when the set-points to be tracked should be recomputed. We also proposed a distributed implementation of our proposed architecture that made use of the algorithm for determining phase-cohesive set-points and the network flows that result introduced in Part I. Finally, we demonstrated our proposed control architecture and its distributed

implementation using three case studies applied to two test systems.

13.4 Future Work

Although Parts II and III presented two architectures for distributed control of microgrids, each had its own tradeoffs, many of which resulted from efforts to simplify the problem to be solved. Given that the work in Part II was extended and improved upon in Part III, the following are some ways in which the latter architecture could be improved.

13.4.1 Improvements to Estimation of Deviation from Phase-Cohesive Operation

Recall the control architecture presented in Chapter 10, which was designed to ensure that the set-points of the generators tracked those that were known to be phase-cohesive. While this architecture was sufficient for ensuring phase-cohesive operation for small load perturbations, in Section 10.4 we introduced a method for estimating the amount by which the system deviated from the phase-cohesive operating point for larger changes in load. Next, we propose two possible alternatives for computing this estimate for future work.

As noted in Section 11.4, ensuring phase-cohesive operation is equivalent to ensuring that the power flow across each branch in a microgrid is within limits, specifically in such a way that ensures the phase angle difference across each branch is less than $\frac{\pi}{2}$. In Section 10.4, we proposed a method that was designed to ensure this for small perturbations in load by using an estimate for the deviation from phase cohesive operation based upon a first-order approximation of the function $h^l(\cdot)$. Extending this idea, which was designed to account for the fact that the deviation from a phase cohesive operating point depends on the amount by which the loads are perturbed and the distribution of the load perturbations, we can use so-called injection shift factors (ISFs), which are a common tool in power system analysis [66].

From the microgrid model in (9.2) – (9.3), let $\omega := \left[u^{\mathrm{T}}, -\ell^{\mathrm{T}}\right]^{\mathrm{T}}$ be a vector of power injections from each of the buses. Then, if we denote the power flow along branch $e = (i, j)$ by f_e, the following relationship maps changes in injections at the buses to changes in power flows along the branches using

linear sensitivities:

$$\begin{bmatrix} \Delta f_{e_1} \\ \vdots \\ \Delta f_{e_{|\mathcal{E}_p|}} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial f_{e_1}}{\partial \omega_1} & \cdots & \frac{\partial f_{e_1}}{\partial \omega_n} \\ \vdots & & \vdots \\ \frac{\partial f_{e_1}}{\partial \omega_1} & \cdots & \frac{\partial f_{e_1}}{\partial \omega_n} \end{bmatrix}}_{:=\Psi} \begin{bmatrix} \Delta \omega_1 \\ \vdots \\ \Delta \omega_n \end{bmatrix}, \tag{13.1}$$

where Ψ is referred to as the shift factor matrix. From this expression, we can write an estimate for the change in flow along branch $e \in \mathcal{E}_p$ given changes in injections $\begin{bmatrix} \Delta\omega_1, \ldots, \Delta\omega_n \end{bmatrix}^{\mathrm{T}}$ as

$$\Delta f_e = \frac{\partial f_e}{\partial \omega_1}\Delta\omega_1 + \cdots + \frac{\partial f_e}{\partial \omega_n}\Delta\omega_n, \tag{13.2}$$

If the values of the $\frac{\partial f_e}{\partial \omega_i}$'s were known, an estimate for the amount by which the flow along each branch will change given changes in injections at the buses could be estimated. With these branch flow change estimates, the amount of flow along each branch could be computed, and when they near their respective limits, new phase-cohesive set-points could be computed. Next we briefly discuss how the $\frac{\partial f_e}{\partial \omega_i}$'s could be computed using ratio consensus and the feasible flow algorithm.

By rearranging the terms in (13.2), we see that we can write the sensitivity $\frac{\partial f_e}{\partial \omega_i}$ as a ratio, i.e.,

$$\frac{\partial f_e}{\partial \omega_i} = \frac{\Delta f_e - \sum_{j=1, j\neq i}^{n} \frac{\partial f_e}{\partial \omega_j}\Delta\omega_j}{\Delta\omega_i}. \tag{13.3}$$

By using a method similar to the one discussed in (11.4), the nodes could compute an estimate for each Δf_e using the feasible flow algorithm, and then using $(n \times |\mathcal{E}_p|)$-instances of the ratio consensus algorithm, each element of the matrix Ψ could be computed.

Another option for determining when to recompute phase-cohesive set-points is to use the results from [41, Theorem 4]. More specifically, if we could impose limits on the amount by which the load demands can change, we could define a set of box constraints on the injections ω. Then, if we could explore all of the extremal points of the polytope whose vertices are defined by the box constraints, the results in [41, Theorem 4] state that, if

the system is phase cohesive at each of the extremal points, it will remain phase cohesive for any set of injections within limits. While the problem of exploring all of the extremal points is combinatorially prohibitive, it may be possible to take advantage of the structure of the problem using methods similar to, e.g., Simplex [67] to make computationally feasible.

13.4.2 Improvements to Feasible Flow Algorithm

As introduced in Section 3.2, the feasible flow algorithm, which is a big component of the architecture presented in Part III, must utilize a communication graph that conforms to the underlying electrical network and must be executed for an infinite number of iterations. To make this algorithm suitable for use in practical applications, both of these drawbacks should be accounted for by some future developments. It was noted in Remark 3 that it may be possible to use ratio consensus to decouple the communication for the feasible flow algorithm from the electrical network. While some early work was done to test such a modification, some additional work would need to be done to prove convergence, possibly relying on average consensus instead of ratio consensus. Finally, while it was noted in Remark 2 that it may be possible to use max- and min-consensus to bound the worst-case error in the flow balances as the algorithm evolves, some additional work would be needed to prove that such a scheme is sufficient.

APPENDIX A

PART I PARAMETERS

A.1 Physical Layer Hardware Data

Table A.1: Value of added inductances in 6-bus test power system.

Parameter	Inductance [mH]
$L_{1,A}$	2.041
$L_{1,B}$	1.905
$L_{1,C}$	1.961
$L_{2,A}$	4.175
$L_{2,B}$	4.162
$L_{2,C}$	4.059

Table A.2: Hampden Engineering Corporation Synchronous Machine.

Parameter	Value
Armature Voltage	133/230 RMS Volts
Armature Current	15.5/9 RMS Amps
Horsepower	2 Hp
Speed	1200 rpm
Frequency	60 Hz
Model	Syn-2

Table A.3: Kollmorgen Goldline Brushless Permanent Magnet Servomotor.

Parameter	Value
Stall Current (Continuous)	10.3 RMS Amps
Stall Current (Peak)	33.0 RMS Amps
Torque (Continuous)	6.44 N·m
Torque (Peak)	19.5 N·m
Rated L/L Voltage	230 RMS Volts
Torque (Continuous)	6.44 N·m
Maximum Speed	4900 rpm
Frequency	164 Hz
Model	B-206-C-21

APPENDIX B

PART II PARAMETERS

B.1 Six-Bus Microgrid Parameters

Table B.1: Six-bus microgrid generator set-point limits, initial set-points, and time constants.

i	$\underline{u}_i$	$\overline{u}_i$	$u_i[0^-]$	D_i
1	0.1	1.15	1.1	0.225
2	0.15	2.65	1.1	0.679
3	0.05	1.68	1.1	0.95

Table B.2: Six-bus microgrid load initial demands and time constants.

i	ℓ_i^0	D_i
4	1.15	0.0125
5	1.25	0.0679
6	0.9	0.0479

Table B.3: Six-bus microgrid network parameters.

B_{ij}	B_{11}	B_{22}	B_{33}	B_{44}	B_{55}	B_{66}
Susceptance	0.167	0.228	0.283	0.172	0.241	0.258

B_{ij}	B_{14}	B_{15}	B_{24}	B_{26}	B_{35}	B_{36}
Susceptance	-2.919	-6.685	-4.474	-4.375	-7.435	-6.274

Table B.4: Simulation parameters for Cases I and II.

Parameter	Value
Feasible flow iterations, k_0	200
Ratio consensus tolerance, ϵ	10^{-5}
Distributed algorithm iteration period	0.001 s
Constant for phase-cohesiveness margin, c	0.01
Numerical integration timestep	0.001 s

B.2 37-Bus Microgrid Parameters

Table B.5: 37-bus microgrid generator set-point limits, initial set-points, and time constants.

i	1	2	3	4	5
$\underline{u}_i$	0.1261	0.1014	0.224	0.2263	0.1228
$\overline{u}_i$	1.9623	2.6025	2.8415	2.2681	3.1077
$u_i[0]$	1.928	1.928	1.928	1.928	1.928
D_i	0.1073	0.1302	0.1169	0.124	0.1352

i	6	7	8	9	10
$\underline{u}_i$	0.3999	0.4151	0.1036	0.4821	0.4719
$\overline{u}_i$	2.8863	2.0725	1.5294	1.7841	3.0515
$u_i[0]$	1.928	1.928	1.928	1.928	1.928
D_i	0.1011	0.1009	0.1289	0.109	0.1378

i	11	12	13	14	15
$\underline{u}_i$	0.2419	0.1432	0.4439	0.1365	0.4672
$\overline{u}_i$	2.342	2.7223	1.5123	2.0293	2.0279
$u_i[0]$	1.928	1.928	1.928	1.928	1.928
D_i	0.1172	0.1388	0.1399	0.1473	0.1305

Table B.6: 37-bus microgrid load time constants.

i	16	17	18	19	20	21
D_i	0.0811	0.0898	0.0870	0.0515	0.0972	0.0950
i	21	22	23	24	25	26
D_i	0.0950	0.0735	0.0772	0.0507	0.0640	0.0883
i	27	28	29	30	31	32
D_i	0.0899	0.0809	0.0501	0.0605	0.0991	0.0645
i	32	33	34	35	36	37
D_i	0.0645	0.0770	0.0602	0.0845	0.0947	0.0681

Table B.7: Network parameters for 37-bus microgrid.

$B_{i,j}$	$B_{1,16}$	$B_{2,16}$	$B_{3,18}$	$B_{4,20}$	$B_{5,22}$	$B_{6,22}$
Susceptance	-7.612	-7.080	-6.564	-7.282	-6.384	-7.701
$B_{i,j}$	$B_{7,28}$	$B_{8,28}$	$B_{9,30}$	$B_{10,30}$	$B_{11,31}$	$B_{12,33}$
Susceptance	-6.394	-6.908	-7.009	-7.491	-7.343	-7.421
$B_{i,j}$	$B_{13,33}$	$B_{14,37}$	$B_{15,37}$	$B_{16,17}$	$B_{17,18}$	$B_{17,19}$
Susceptance	-6.465	-6.951	-7.717	-10.738	-10.75	-10.216
$B_{i,j}$	$B_{19,20}$	$B_{19,21}$	$B_{21,23}$	$B_{22,23}$	$B_{23,24}$	$B_{24,25}$
Susceptance	-10.375	-12.267	-10.404	-11.234	-10.948	-10.571
$B_{i,j}$	$B_{23,26}$	$B_{26,27}$	$B_{27,28}$	$B_{26,29}$	$B_{29,30}$	$B_{30,31}$
Susceptance	-11.368	-11.886	-11.06	-12.086	-10.116	-10.335
$B_{i,j}$	$B_{31,32}$	$B_{32,34}$	$B_{33,34}$	$B_{34,35}$	$B_{35,36}$	$B_{36,37}$
Susceptance	-12.162	-11.448	-11.013	-10.375	-11.928	-10.701

Table B.8: Simulation parameters for Case III.

Parameter	Value
Feasible flow iterations, k_0	600
Ratio consensus tolerance, ϵ	10^{-7}
Distributed algorithm iteration period	0.00001 s
Constant for phase-cohesiveness margin, c	0.01
Numerical integration timestep	0.001 s

REFERENCES

[1] G. Stavrakakis and G. Kariniotakis, "A general simulation algorithm for the accurate assessment of isolated diesel-wind turbines systems interaction. i. A general multimachine power system model," *Energy Conversion, IEEE Transactions on*, vol. 10, no. 3, pp. 577–583, Sep 1995.

[2] R. Lasseter et al., "Integration of distributed energy resources: The CERTS microgrid concept," Lawrence Berkeley National Laboratory, Tech. Rep. LBNL-50829, Apr. 2002.

[3] N. Hatziargyriou, H. Asano, R. Iravani, and C. Marnay, "Microgrids," *IEEE Power and Energy Magazine*, vol. 5, no. 4, pp. 78–94, July 2007.

[4] Q. Shafiee, J. Guerrero, and J. Vasquez, "Distributed secondary control for islanded microgrids – a novel approach," *IEEE Transactions on Power Electronics*, vol. 29, no. 2, pp. 1018–1031, Feb. 2014.

[5] S. T. Cady, A. D. Domínguez-García, and C. N. Hadjicostis, "A distributed generation control architecture for islanded ac microgrids," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 1717 – 1735, Jan. 2015.

[6] "U.S.DoE — smart grid," May 2015. [Online]. Available: http://energy.gov/oe/services/technology-development/smart-grid

[7] A. Emadi and M. Ehsani, "Aircraft power systems: technology, state of the art, and future trends," *IEEE Aerospace and Electronic Systems Magazine*, vol. 15, no. 1, pp. 28–32, Jan. 2000.

[8] J. Rosero, J. Ortega, E. Aldabas, and L. Romeral, "Moving towards a more electric aircraft," *IEEE Aerospace and Electronic Systems Magazine*, vol. 22, no. 3, pp. 3–9, Mar. 2007.

[9] K. Butler, N. Sarma, and V. Ragendra Prasad, "Network reconfiguration for service restoration in shipboard power distribution systems," *IEEE Transactions on Power Systems*, vol. 16, no. 4, pp. 653–661, Nov. 2001.

[10] A. Monti, D. Boroyevich, D. Cartes, R. Dougal, H. Ginn, G. Monnat, S. Pekarek, F. Ponci, E. Santi, S. Sudhoff, N. Schulz, W. Shutt, and F. Wang, "Ship power system control: a technology assessment," in *Proc. of IEEE Electric Ship Technologies Symposium*, 2005, pp. 292–297.

[11] T. Gruzs and J. Hall, "Ac, dc or hybrid power solutions for today's telecommunications facilities," in *Proc. of International Telecommunications Energy Conference*, 2000, pp. 361–368.

[12] J. Guerrero, J. Vasquez, J. Matas, L. de Vicuña, and M. Castilla, "Hierarchical control of droop-controlled ac and dc microgrids–a general approach toward standardization," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 1, pp. 158 – 172, Jan. 2011.

[13] A. Wood and B. Wollenberg, *Power Generation, Operation, and Control*. New York, NY: Wiley, 1996.

[14] M. Marwali, J.-W. Jung, and A. Keyhani, "Control of distributed generation systems - part II: Load sharing control," *IEEE Transactions on Power Electronics*, vol. 19, no. 6, pp. 1551–1561, Nov. 2004.

[15] J. W. Simpson-Porco, F. Dörfler, and F. Bullo, "Synchronization and power sharing for droop-controlled inverters in islanded microgrids," *Automatica*, vol. 49, no. 9, pp. 2603–2611, 2013.

[16] A. Tsikalakis and N. Hatziargyriou, "Centralized control for optimizing microgrids operation," *IEEE Transactions on Energy Conversion*, vol. 23, no. 1, pp. 241–248, Mar. 2008.

[17] J. Guerrero, J. Vasquez, J. Matas, M. Castilla, and L. de Vicuña, "Control strategy for flexible microgrid based on parallel line-interactive UPS systems," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 3, pp. 726–736, March 2009.

[18] A. Micallef, M. Apap, C. Spiteri-Staines, and J. Guerrero, "Secondary control for reactive power sharing in droop-controlled islanded microgrids," in *Proc. of IEEE International Symposium on Industrial Electronics*, 2012, pp. 1627–1633.

[19] J. Peças Lopes, C. Moreira, and A. Madureira, "Defining control strategies for microgrids islanded operation," *IEEE Transactions on Power Systems*, vol. 21, no. 2, pp. 916–924, May 2006.

[20] F. Katiraei and M. Iravani, "Power management strategies for a microgrid with multiple distributed generation units," *IEEE Transactions on Power Systems*, vol. 21, no. 4, pp. 1821–1831, Nov. 2006.

[21] N. Li, L. Chen, C. Zhao, and S. Low, "Connecting automatic generation control and economic dispatch from an optimization view," in *Proc. of American Control Conference*, 2014, pp. 735–740.

[22] G. Larsen, N. van Foreest, and J. Scherpen, "Distributed control of the power supply-demand balance," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 828–836, June 2013.

[23] A. Bidram, A. Davoudi, F. Lewis, and Z. Qu, "Secondary control of microgrids based on distributed cooperative control of multi-agent systems," *IET Generation, Transmission Distribution*, vol. 7, no. 8, pp. 822–831, Aug. 2013.

[24] M. Andreasson, H. Sandberg, D. Dimarogonas, and K. Johansson, "Distributed integral action: Stability analysis and frequency control of power systems," in *Proc. of IEEE Conference on Decision and Control*, 2012, pp. 2077–2083.

[25] A. Bidram, F. Lewis, A. Davoudi, and Z. Qu, "Frequency control of electric power microgrids using distributed cooperative control of multi-agent systems," in *Proc. of IEEE Conference on Cyber Technology in Automation, Control and Intelligent Systems (CYBER)*, May 2013, pp. 223–228.

[26] C.-Y. Chang and W. Zhang, "Distributed control of inverter-based lossy microgrids for power sharing and frequency regulation under voltage constraints," *arXiv preprint arXiv:1501.05890*, 2015.

[27] F. Dörfler, J. W. Simpson-Porco, and F. Bullo, "Breaking the hierarchy: Distributed control & economic optimality in microgrids," *IEEE Transactions on Control of Network Systems*, Jan. 2014, submitted. Available at http://arxiv.org/pdf/1401.1767v1.pdf.

[28] M. Arnold, R. R. Negenborn, G. Andersson, and B. D. Schutter, "Multi-area predictive control for combined electricity and natural gas systems," in *Proc. of European Control Conference*, 2009.

[29] A. Maknouninejad, W. Lin, H. G. Harno, Z. Qu, and M. A. Simaan, "Cooperative control for self-organizing microgrids and game strategies for optimal dispatch of distributed renewable generations," *Energy Systems*, pp. 23–60, March 2012.

[30] Y. Zhang, N. Gatsis, and G. Giannakis, "Robust distributed energy management for microgrids with renewables," in *Proc. of IEEE Conference on Smart Grid Communications*, 2012, pp. 510–515.

[31] M. Andreasson, D. Dimarogonas, K. Johansson, and H. Sandberg, "Distributed vs. centralized power systems frequency control," in *Proc. of European Control Conference*, 2013, pp. 3524–3529.

[32] A. Venkat, I. Hiskens, J. Rawlings, and S. Wright, "Distributed MPC strategies with application to power system automatic generation control," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 6, pp. 1192–1206, 2008.

[33] S. Bolognani and S. Zampieri, "Distributed control for optimal reactive power compensation in smart microgrids," in *Proc. of IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 6630–6635.

[34] E. Dall'Anese, H. Zhu, and G. Giannakis, "Distributed optimal power flow for smart microgrids," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1464–1475, Sept. 2013.

[35] B. Robbins, C. Hadjicostis, and A. Dominguez-Garcia, "A two-stage distributed architecture for voltage control in power distribution systems," *Power Systems, IEEE Transactions on*, vol. 28, no. 2, pp. 1470–1482, May 2013.

[36] S. T. Cady, A. D. Domínguez-García, and C. N. Hadjicostis, "Finite-time approximate consensus and its application to distributed frequency regulation in islanded ac microgrids," in *Proc. of Hawaii International Conference on System Sciences*, 2015.

[37] S. T. Cady, C. N. Hadjicostis, and A. D. Domínguez-García, "Distributed frequency control of inertia-less ac microgrids," in *Proc. of IEEE Conference on Decision and Control*, Dec. 2015.

[38] S. T. Cady, A. D. Domínguez-García, and C. N. Hadjicostis, "Robust implementation of distributed algorithms for control of distributed energy resources," in *Proc. of North American Power Symposium*, 2011, pp. 1–5.

[39] A. D. Domínguez-García, S. T. Cady, and C. N. Hadjicostis, "Decentralized optimal dispatch of distributed energy resources," in *Proc. of IEEE Conference on Decision and Control*, 2012, pp. 3688–3693.

[40] S. T. Cady and A. D. Dominguez-Garcia, "Distributed generation control of small-footprint power systems," in *Proc. of North American Power Symposium*, 2012, pp. 1–6.

[41] F. Dorfler, M. Chertkov, and F. Bullo, "Synchronization in complex oscillator networks and smart grids," *Proc. of the National Academy of Sciences*, vol. 110, no. 6, pp. 2005–2010, February 2013.

[42] P. Sauer and A. Pai, *Power System Dynamics and Stability*. Upper Saddle River, NJ: Prentice Hall, 1998.

[43] D. B. West, *Introduction to Graph Theory*, 2nd ed. Prentice Hall, 2001.

[44] V. Yadav and M. V. Salapaka, "Distributed protocol for determining when averaging consensus is reached," in *Communication, Control, and Computing (Allerton), 2007 45th Annual Allerton Conference on*, Oct. 2007, pp. 715–720.

[45] A. D. Domínguez-García and C. N. Hadjicostis, "Distributed algorithms for control of demand response and distributed energy resources," in *Proc. of IEEE Conference on Decision and Control*, 2011, pp. 27–32.

[46] A. D. Domínguez-García, C. N. Hadjicostis, and N. Vaidya, "Resilient networked control of distributed energy resources," *IEEE Journal on Selected Areas in Comm.*, vol. 30, no. 6, pp. 1137–1148, Jul. 2012.

[47] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, 2003.

[48] N. Manitara and C. N. Hadjicostis, "Distributed stopping in average consensus via event-triggered strategies," in *Allerton Conference on Communication, Control, and Computing*, Oct. 2013, pp. 1336–1343.

[49] L. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton, NJ: Princeton University Press, 1962.

[50] D. P. Bertsekas, P. A. Hosein, and P. Tseng, "Relaxation methods for network flow problems with convex arc costs," *SIAM Journal of Control and Optimization*, vol. 25, no. 5, pp. 1219–1243, 1987.

[51] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena Scientific, 1999.

[52] P. Varaiya, F. Wu, and R.-L. Chen, "Direct methods for transient stability analysis of power systems: Recent results," *Proc. of the IEEE*, vol. 73, no. 12, pp. 1703–1715, Dec. 1985.

[53] H.-D. Chiang, *Direct Methods for Stability Analysis of Electric Power Systems: Theoretical Foundation, BCU Methodologies, and Applications*. New York, NY: Wiley, 2011.

[54] A. S. Debs, *Modern Power Systems Control and Operation*. Boston, MA: Kluwer Academic Publishers, 1988.

[55] A. Bergen and V. Vittal, *Power System Analysis*. Upper Saddle River, NJ: Prentice Hall, 2000.

[56] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY: Cambridge University Press, 2004.

[57] D. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 2004.

[58] M. Madrigal and V. Quintana, "An analytical solution to the economic dispatch problem," *IEEE Power Engineering Review*, vol. 20, no. 9, pp. 52–55, Sep. 2000.

[59] A. Rapp, "xbee-arduino." [Online]. Available: http://code.google.com/p/xbee-arduino/

[60] H. Dai and R. Han, "TSync: a lightweight bidirectional time synchronization service for wireless sensor networks," *ACM SIGMOBILE Mobile Computing Communications Review*, vol. 8, pp. 125–139, Jan. 2004.

[61] N. Manitara and C. N. Hadjicostis, "Distributed stopping strategies for average consensus in digraphs," in *Proc. of IEEE International Symposium on Communications, Control, and Signal Processing*, May 2014.

[62] Y. Kuramoto, "Self-entrainment of a population of coupled non-linear oscillators," in *International Symposium on Mathematical Problems in Theoretical Physics*, ser. Lecture Notes in Physics, H. Araki, Ed. Springer Berlin Heidelberg, 1975, vol. 39, pp. 420–422.

[63] A. Ben-Israel and T. Greville, *Generalized Inverses: Theory and Applications*, ser. CMS Books in Mathematics. Springer, 2003.

[64] "Power system toolbox webpage," May 2015. [Online]. Available: http://www.eps.ee.kth.se/personal/vanfretti/pst/Power_System_Toolbox_Webpage_PST.html

[65] "Networkx," May 2015. [Online]. Available: http://networkx.github.io

[66] J. Peschon, D. S. Piercy, W. F. Tinney, and O. J. Tveit, "Sensitivity in power systems," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-87, no. 8, pp. 1687–1696, Aug 1968.

[67] G. Dantzig, *Linear Programming and Extensions*. Princeton University Press, 1998.