Energy Storage for Frequency Regulation on the Electric Grid

by

Olivia Leitermann

S.B., Massachusetts Institute of Tech.(2005)

S.M., Massachusetts Institute of Tech.(2008)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

Massachusetts Institute of Technology

June 2012

Author_____

Department of Electrical Engineering and Computer Science
23 May 2012

Certified by_____

James L. Kirtley, Jr.
Professor
Thesis Supervisor

Accepted by_____

Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Energy Storage for Frequency Regulation on the Electric Grid

by

Olivia Leitermann

Submitted to the Department of Electrical Engineering and Computer Science
on 23 May 2012, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

ANCILLARY services such as frequency regulation are required for reliable operation of the electric grid. Currently, the same traditional thermal generators that supply bulk power also perform nearly all frequency regulation. Instead, using high power energy storage resources to provide frequency regulation can allow traditional thermal generators to operate more smoothly. However, using energy storage alone for frequency regulation would require an unreasonably large energy storage capacity. Duration curves for energy capacity and instantaneous ramp rate are used to evaluate the requirements and benefits of using energy storage for a component of frequency regulation. Filtering is used to separate the portion of a frequency regulation control signal suitable for provision by an energy storage unit from the portion suitable for provision by traditional thermal generating resources. Not all frequency regulation signals are equally amenable to the filtering approach used here. Data from two U.S. control areas are used to demonstrate the techniques and the results are compared.

Thesis Supervisor: James L. Kirtley, Jr.
Title: Professor

Acknowledgements

I would like to acknowledge the sponsors of this research, the MIT-Portugal Program in collaboration with the Fundação para a Ciência e a Tecnologia (FCT) of Portugal, and the Masdar Institute of Science and Technology, Abu Dhabi.

In addition to financial support, I have received an enormous amount of help and guidance from many people. My supervisor, Prof. James L. Kirtly, Jr., has steered me from the early stages of problem formulation to the clarification and careful presentation of ideas in this thesis. He has kept me on the right track while forcing me to discover the hard problems for myself. My thesis committee members Prof. Steven B. Leeb and Prof. Aleksandar Stankovic̀ (of Tufts) have also been immensely helpful, by pointing out the weak areas in my work and offering suggestions to improve them.

The wonderfully helpful control area engineers that I have worked with both supplied me with the data that makes this thesis interesting and were quick to offer context and information about that data. Bob Wolfinger and Scott Baker assisted me with the PJM data, and the several people from the unnamed control area were equally accommodating.

Many other professors and colleagues at various times answered questions, helped me with unfamiliar concepts, listened to presentations, and offered advice. Prof. George C. Verghese is a master of signal processing and helped me interpret the odd differences between my two data sets, as well as offering general knowledge from his former life as a power engineer. Dr. Faisal Kashif also helped lead me through the wilds of signal processing and filter design. Prof. Ignacio J. Pérez-Arriaga (of Comillas University, Madrid and MIT), and Prof. Marija D. Ilić (of Carnegie-Mellon), and Stephen R. Connors have each repeatedly supplied some excellent ideas and power systems context. My fellow MIT students, including Jiankang Wang, Young Jin Kim, and the members of the MIT Electricity Student Research Group, as well as Elizabeth Ward and Somak Chatterjee, have been an enormous resource for ferreting out data, explaining unfamiliar concepts, brainstorming, and critiquing presentations (and also are a lot of fun).

In my long journey through MIT, all of the professors that I have encountered as teachers and as advisors have been brilliant and excited about sharing their understanding. I have learned from them nearly everything I know about engineering and lots of ways to think about problems. My friends and classmates have always been around to encourage me to work, support me through hard times, and help me goof off.

Sabrina has been there for me since I started this work, and has kept me sane along the way. I don't think I could have finished this thesis without her.

My parents instilled in me a love of learning and encouraged my curiosity. There was never anything I needed that they did not try to provide. They have made me the person I am today.

Thank you all.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

THE ELECTRIC power grid is of fundamental importance in society today. Electric power is an enabling technology for a large fraction of everyday tasks and sophisticated processes. It provides a clean, safe, versatile power source for a huge variety of applications.

The electric power system depends for its operation on the balance between generation and load. This balance must be constantly maintained. One piece of this balance is frequency regulation (or load frequency control, LFC), which is the rapid following of load by generators. This task is called frequency regulation because the frequency of the system indicates the load balance, as do the observed power flows. Frequency regulation follows the load on a time scale from seconds to minutes, after generator governors have responded but before load-following capacity is brought online to follow the slow load changes from hour to hour.

Currently, traditional thermal generators perform the majority of the frequency regulation on the system. However this is not a task at which they perform well. Frequency regulation is also a duty which increases wear on thermal generators and also decreases efficiency. Thermal generators are designed primarily to deliver bulk energy on the electric grid, but they are not well suited to the provision of frequency regulation.

By contrast, energy storage technologies have many characteristics which make them well suited to the provision of frequency regulation. Frequency regulation is primarily a power service, with changes in output power being required over relatively short times. The frequency regulation signal also reverses frequently, limiting the amount of total energy delivery which is required for its provision. Additionally, many energy storage technologies have relatively large power capacity compared to their energy capacity. This means that they cannot economically be used for diurnal cycling or energy arbitrage, but the small energy capacity compared to power capability is sufficient to provide frequency regulation.

However, moderately-sized energy storage alone is not capable of providing frequency regulation. The regulation signal, as it is created now in most control areas, does not display zero average power. This means that a resource which is following the regulation signal would need to deliver a small amount of energy over the course of the hours or days it is assigned to regulate. Additionally, even if the average power component is removed from

regulation signals, large amounts of energy delivery are generally required to follow the signals exactly. This means that a very large energy storage unit would be required, which very slowly filled and emptied over the hours. This is not a particularly good use of fast energy storage resources, and it represents a large amount of expensive storage.

A solution to this difficulty is to use a hybrid combination of energy storage and traditional thermal generation to provide frequency regulation. Energy storage takes care of the fast fluctuating component of the regulation signal, while thermal generation is instructed to more slowly follow the general trend of the signal, and to enable the energy storage to operate without delivering too much energy [65]. This hybrid scheme is similar to strategies employed in other fields, including fuel cell operation [33, 35], power electronics [101], and in the operating reserves on the electric grid [78].

A hybrid system with both energy storage and traditional generation assets jointly delivering frequency regulation provides advantages by allowing the thermal generation to operate at a steadier power level, and hence decreasing wear on the thermal assets due to unit ramping and increasing thermal unit efficiency (see Section 3.3). Linear filters are used to divide the frequency regulation signal into a low frequency component for thermal generators and a high frequency component for energy storage. However, not all signals are equally suitable for division in this manner. Power signals which contain a relatively large amount of fast fluctuations will deliver more benefit from such a treatment than signals with a more smooth aspect. Because the frequency regulation signal depends on many aspects of system operation, regulation signals exist which exhibit both of these aspects. The characteristics of a particular signal of interest must be examined to determine the relative suitability of a hybrid energy storage frequency regulation approach.

1.1 Frequency Regulation

Frequency regulation is a component of operating reserves for good control of the electric grid. In response to step changes in load, generator inertia immediately works to restore the balance between generation and load, and generator governors change load based on frequency to bring the system to a new equilibrium point. In order to restore the frequency to its scheduled value, and to maintain tieline power flows between control areas at scheduled values, frequency regulation is performed, consisting of small increases or decreases in power output by participating generators [43]. In general, frequency regulation is provided by a subset of the system's thermal or hydraulic power plants that are connected to the automatic generation control (AGC) signal created by the system operator. This signal is usually updated every 2-10 seconds, and either indicates the new requested power output for the

generator or, for some older plants, indicates whether the power setpoint should be raised or lowered [2, Chapter 11] [56, 60].

In large electric power interconnections such as in the United States and Europe, the frequency performance of an area is generally regulated. The Eastern Interconnection of the US is managed by several control areas working together to maintain grid operations, including the balance between generation and load. Each control area is responsible for balancing its generation against its load, and in so doing for maintaining the interconnection frequency at its scheduled value (near 60 Hz) and maintaining power flows across tielines between areas at their scheduled values. This system ensures that each control area is responsible for its own load, and does not impose power fluctuations on its neighbors. This is usually measured by a quantity known as Area Control Error (ACE), which is the sum of the current tieline power flow error and the scaled current frequency error [76].[1] The frequency error is scaled by the frequency bias setting. This setting is determined for each area based on the generator governor frequency response and load characteristics [77]. Thus ACE is designed to approximately represent the deficit generation in the area as a function of time.

In the US, the control of the load-generation balance is mandated by the North American Electric Reliability Council (NERC) Control Performance Standards 1 and 2 (CPS1 and CPS2). These CPS requirements mandate the minimum control level for ACE in each control area so that areas are taking care of their own load changes. CPS1 requires that the ACE for a control area be in the direction to bring the frequency back towards its scheduled value at least a certain fraction of the time. This is helpful because if the interconnection frequency is low, it is valuable to have some control areas overgenerating to bring the frequency back toward its scheduled value, even if that means that they are exporting unscheduled power on tielines. CPS2 limits the average value of ACE over each 10-minute period for each area, to ensure that the total load-generation imbalance for the area does not tend to be too large in magnitude [76].The CPS limits replace older and largely similar standards known as A1 and A2 [17]. The degree of frequency control which is sufficient from a technical, rather than regulatory, standpoint has been a subject of debate for some time [88, 51, 5, 74, 94, 19, 17, 36, 97, 48, 37, 1, 42].

Each control area produces an AGC signal based on its ACE values, to regulate those values. In general, frequency regulation is performed by a subset of the various power plants in a control area which have the capability to respond to an AGC signal, with each plant dedicating a small portion of its power capacity to AGC [85]. Generators engaged in frequency regulation require a data connection to the system operator in order to respond

[1] For the purposes of frequency regulation, frequency is usually considered to be uniform over a control area, or even over the interconnection. Although local phase angle rates of change can differ, this is averaged out to some degree over the course of several seconds.

appropriately to the AGC signal. Usually the amount of generation power capacity engaged in AGC is approximately 1–2% of the total load for the control area [68].

There has been some interest in the literature in the provision of frequency regulation by renewable resources other than hydropower. These explorations seem to be primarily in the early stages, e.g. [72, 30, 54]. The primary factor limiting the use of wind and solar resources for frequency regulation seems to be the necessity of reducing power output below its maximum, either to provide space to increase power for regulation up, or when providing regulation down. The energy which goes intentionally un-captured reduces the total compensation available without an attending reduction in fuel or maintenance costs. An exception is the use of either inherent stored energy in the system (such as wind turbine inertia or thermal storage in a concentrated solar-thermal plant) or supplemental energy storage (as when storage is added to the output of a photovoltaic solar plant for other reasons) to provide regulation service. In the case of frequency regulation using stored energy from a renewable generator, the techniques described in this thesis would apply.

1.2 Related Work

For some time, it has been recognized that fast load fluctuations are difficult for traditional thermal generators to follow. The usual approach to this case has been to separate out the fastest fluctuations and declare those too difficult to regulate, and then to perform regulation for the remainder of the signal [22, 88]. However, if too much of the signal is removed, then control performance can suffer.

An approach which is more closely related to this work is presented in [85], where several thermal-dominated utilities and a hydro-dominated utility agreed to transfer regulation to the hydro-dominated utility to reduce the regulating burden on the thermal generating units. This effectively meant that the hydro-dominated utility assumed regulating control of all the fast fluctuations and the thermal-dominated requesting utilities continued to follow slower load trends, so that not too much power was transferred. Batteries are enlisted to perform the fast regulation in [90] in a similar manner, with the participating thermal generation programmed with a deadband large enough to prevent its following all of the fast fluctuations. The Israeli grid operates as a relatively small island system (for political reasons) and performs load-generation balancing slightly differently, but [57] evaluates the simulated inclusion of a battery storage system to perform primary frequency response. It was found that the battery discharged quickly if operated in open loop, and a high-pass filter was added to its controlling signal in order to maintain its state of charge. PJM, a large US control area, is also planning to implement a divided regulation signal with separate

markets for fast and slow regulation resources [82]. The amount of energy storage required for frequency regulation is briefly addressed in [68].

Although frequency regulation is a relatively fast-acting resource, it is not intended for contingency response and does not dominate the instantaneous power system response to load steps. This makes the system frequency over time in response to a load step a flawed measure of frequency regulation performance [51]. However, some work with using energy storage for frequency response or regulation has focused on the response of the system frequency to load steps [75, 95].

1.3 Thesis Objectives and Contributions

The objectives of this thesis are to address several main questions. The first question is of what use lower-energy, higher-power energy storage can be on the electric grid. This work finds that energy storage can be valuable for frequency regulation. In this application, a further important question is how much energy storage is required for frequency regulation in a given control area. This amount will vary among areas, and is dependent on the signal characteristics of the control area frequency regulation signal. If energy storage is to be applied to frequency regulation, how should it be dispatched to greatest advantage? And what advantages does energy storage offer for grid operation to justify its use? This thesis addresses these questions and presents tools and a framework to find the answers.

The body of this thesis is organized as follows. Chapter 2 introduces the two data sets used in this thesis, and develops the graphical tools referred to as ramp rate duration curves and energy duration curves which describe the use of non-traditional resources for frequency regulation. Chapter 4 then describes the characteristics of some relevant energy storage technologies and notes some of the cases where these technologies have been integrated into the electric grid, for frequency regulation and for other purposes. The characteristics of thermal generators which are relevant to frequency regulation are described in Chapter 3, along with the limitations of traditional thermal generation technology for performing frequency regulation. The main technique of dividing the regulation load between an energy storage unit and a traditional generator is introduced in Chapter 5, and the benefits of this technique are described. Chapter 6 discusses the importance of the characteristics of the frequency regulation signal for this approach, and the aspects of the signal that make it more or less suitable for the filtering approach of Chapter 5. Finally, Chapter 7 concludes the thesis. Taken as a whole, this work demonstrates mechanisms for determining the amount energy storage which is useful for frequency regulation, discusses how that storage should be dispatched, and diagnoses the circumstances under which storage becomes most useful.

Chapter 2

Load Characteristics and Duration Curves

2.1 Introduction

THIS chapter introduces the graphical tools that are developed in this thesis and the signals which are used to illustrate them. Ramp rate duration curves describe the ramping behavior required by a signal and energy duration curves describe the energy which must be delivered to follow the signal. These tools explain the benefits of dividing the regulation burden which are described in Chapter 5.

2.2 Fast Regulation

Load frequency control (LFC) or frequency regulation is the change in output power of a subset of generators on the grid in order to follow unpredictable load fluctuations, based on fed-back metrics of system balance: frequency, and tieline flows. The electric power load varies on several time scales, from weekly and seasonal cycles to daily load patterns down to changes over the course of less than a minute. The daily, weekly, and seasonal cycles may be predicted with good accuracy. In particular, day-ahead hourly load predictions are usually accurate to a few percent or less [12, 13]. Generation is scheduled to follow this predicted bulk load. By contrast, intra-hour load fluctuations are much more difficult to predict and exhibit a more random characteristic. The fast load fluctuations of under a minute to a few minutes are followed by generation under feedback control, based on the system frequency and tieline power flows. This fast load following is referred to as LFC or frequency regulation, and is performed by generators under automatic generation control (AGC).

As discussed in Chapter 1 and throughout this thesis, certain types of energy storage technologies are well-suited to providing frequency regulation. Energy storage technologies such as flywheels, fast batteries, and others (see Chapter 4) can respond very quickly and reliably

to control signals for changing their power output, and the rate of change of power output does not generally contribute to the wear on the devices. However, many of these technologies are rather expensive compared to traditional thermal generation resources and so the size of storage unit that might be required to provide all frequency regulation for a control area is likely to be prohibitive in cost. This chapter explores two available regulation data sets in the context of the use of energy storage for frequency regulation.

The use of energy storage units for LFC has been limited by the concern that the storage units will unexpectedly be completely filled or emptied and hence be made unavailable for regulation. The graphical tools suggested here, called energy-duration curves and ramp-rate-duration curves, seek to manage and inform the dispatch of energy storage for LFC. Using these tools on a representative data set, it is easy to see the net energy storage and ramp rates that LFC requires. Storage unit unavailability due to empty or full storage capacity may be predicted based on historical data and the system may be designed to avoid or mitigate any such outages. Different methods for dividing the power signal between energy storage and traditional thermal assets may then be easily compared using the curves.

Some additional analytical tools are required to facilitate the use of energy storage units for LFC. Because of the mixture of time scales in this problem, time-series graphics of LFC power requirements offer little insight. For thermal units, the load-duration curve is an especially useful tool for examining use patterns [4]. However, although the power output distributions of energy storage or thermal generating units may still be of concern, load-duration curves provide no information on ramp rates or required net energy delivery, critical characteristics for thermal and energy storage units, respectively. When using an energy storage unit in concert with thermal units, two related metrics become important. The first we shall call the ramp-rate-duration curve, and the second the energy-duration curve [65].

2.3 Data Sets

This thesis uses two data sets to illustrate the techniques described. Both data sets record the frequency regulation requirement for a U.S. control area. The structural differences between the two data sets lead to differences in resulting performance which are explored in Chapter 6.

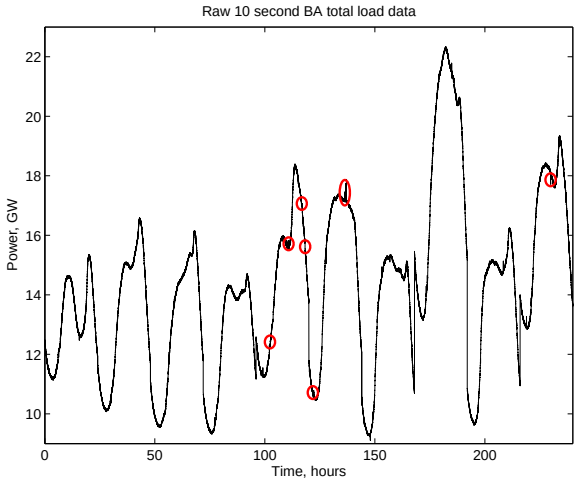

Figure 2.1: Data set A, total generated power in a balancing area on 10 non-consecutive days. Data points are sampled every 10 seconds. Some areas of suspicious data are indicated, where the signal abruptly changes by tens or hundreds of megawatts for a short time, then returns to near its original value. Balancing area engineers indicated these changes are more likely to be due to telemetry errors than to abrupt changes in power generation.

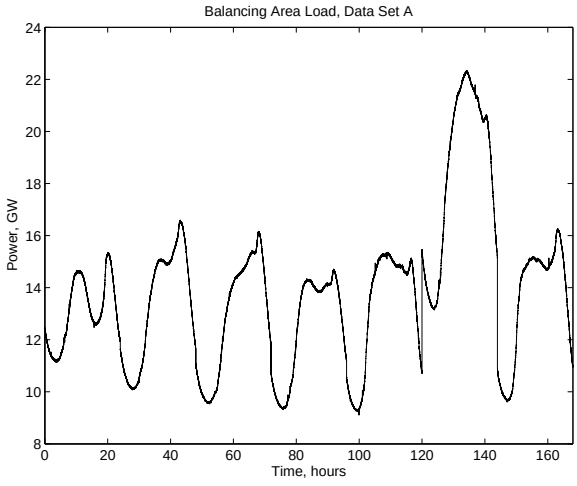

Figure 2.2: Data set A, total generated power in a balancing area on 7 of 10 non-consecutive days. Data points are sampled every 10 seconds. Note that discontinuities between days do not represent actual step changes in generation and each day is processed individually in figures that follow.

2.3.1 Data Set A

To demonstrate this concept, this paper uses a data set from a United States power control area[1] that includes total control area load sampled at 10 second intervals, referred to as data set A in this work. The data set runs for 10 non-consecutive days[2] representing different load conditions and times of year, and is shown in Fig. 2.1. On 3 of the 10 days, anomalies of a few minutes each may be seen where the power output signal changes by tens or hundreds of megawatts for a short time, and then returns to near its original value. These areas have been indicated in the figure. The balancing area engineers who provided this data indicated that such anomalies are probably due to data collection errors rather than to actual step changes in generation. For this reason it was decided that these anomalies ought not to be included in the analysis. Fast recorded changes in load which do not correspond to actual load behavior would tend to indicate more necessity for fast resources like energy storage units than is actually required by the real loads. In order for the analysis to be conservative, such anomalous data should not be considered. For simplicity, the days with major anomalies were removed from the data set, and the remaining 7 days were processed.[3] The data from these 7 days is shown in Fig. 2.2.

The generation data of Fig. 2.2 still includes short anomalies of only one or two points each. While these are difficult to see in the figure, they appear as narrow spikes. Again, these are likely to be due to data collection errors and are not believed to be representative of actual balancing area load, hence should be removed to yield a conservative analysis. In order to remove these points, a 5-point (50-second) median filter was used; i.e., each point was assigned a value equal to the median of the point itself and the two points preceding and following.[4] The median-filtered data is pictured in Fig. 2.3. The difference between the raw data and the median filtered data is pictured in Fig. 2.4. A block diagram showing the filtering strategy and the relationships among the figures is included as Fig. 2.6. Some high-frequency fluctuations have been removed from the data, but the remaining median filtered signal is still believed to be representative. A comparison of the effects of the 5-point median filter with a 3-point (30-second) and a 7-point(70-second) median filter is shown in Fig. 2.5. As can be seen in the figure, the 3-point median filter leaves some anomalous data

[1] In the US, regional authorities such as independent system operators, vertically-integrated utilities, or governmental authorities are responsible for operation of the electric grid within a region, including the balancing of generation and load.

[2] This graph includes data from Sundays 23 Mar. 08, 28 Sep. 08, and 30 Aug. 09; Saturdays 11 Apr. 09, and 2 Jan. 10; Monday 16 Jun. 08, Tuesday 26 May 09, Wednesday 29 Jul. 09, Thursday 1 Oct. 09, and Friday 22 Feb. 08. Data was obtained from balancing area engineers, but it was requested that the balancing area not be named.

[3] This graph includes data only from Sundays 23 Mar. 08, 28 Sep. 08, and 30 Aug. 09; Saturday 11 Apr. 09; Tuesday 26 May 09, Wednesday 29 Jul. 09, and Thursday 1 Oct. 09.

[4] All data processing is done on each day individually to avoid artifacts due to the discontinuity between days.

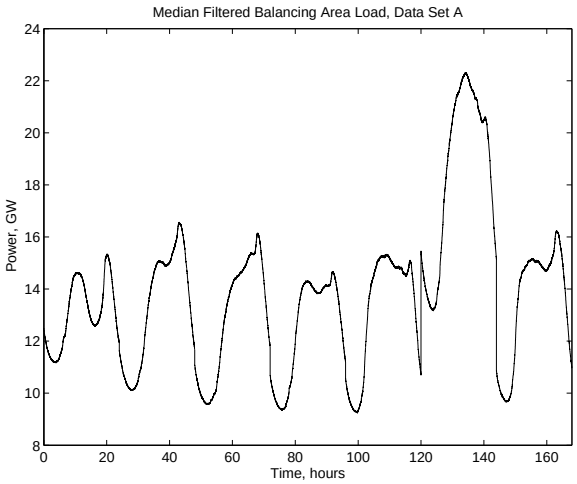

Figure 2.3: Data set A, total load power in a balancing area on 7 non-consecutive days. Data points are sampled every 10 seconds. A 5-point (50-second) median filter was used to remove spurious single data points. Compare to Fig. 2.2.

points that the 5-point filter removes, and the 7-point median filter has a similar effect to the 5-point filter. The analysis following in this chapter uses primarily the 5-point median filtered data, but does offer some comparisons between the raw and the median filtered data from the 7 days.

These data include the total balancing area generation with its predictable daily variations. In general, daily load may be predicted within a few percent or less [12, 13]. Because such

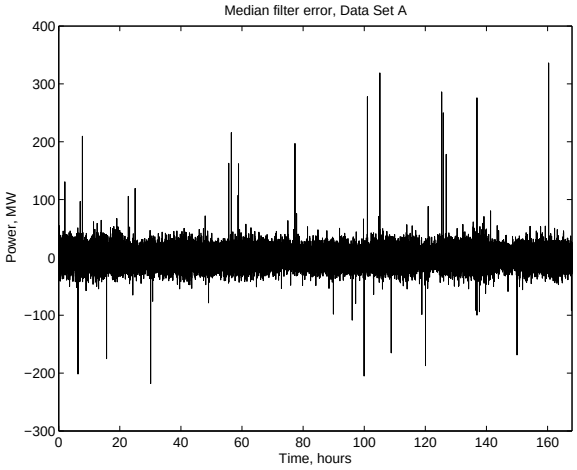

Figure 2.4: Power removed from the data set A as a result of the 5 point median filter; that is, the difference between the raw signal of Fig. 2.2 and the median filtered signal of Fig. 2.3.

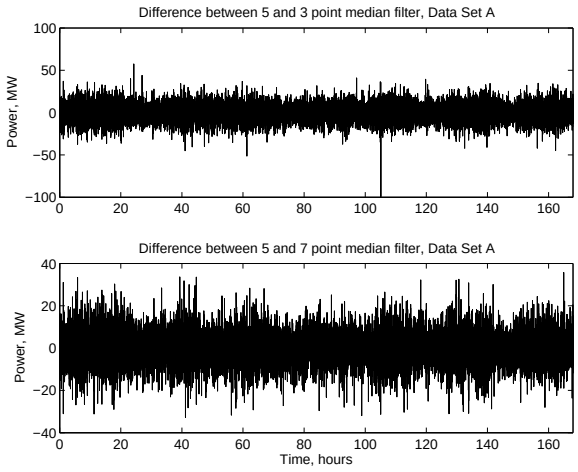

Figure 2.5: Difference between filtered versions of data set A using a 5-point median filter (Fig. 2.3) and using 3-point or 7-point median filter. Note that the 5-point median filter removes some anomalous points which the 3-point filter does not remove, whereas the 7-point filter results in nearly the same changes as the 5-point filter.

predictions were not available, and to avoid the influence of sharp transitions from hour to hour for bulk generation schedules, a non-causal (predictive) filter was used to generate a value for the "bulk load" around which the fluctuations may be seen. This non-causal filter was a 90-minute sliding window averaging filter, with the window centered around the point to be calculated. The setup of this filter is also included in Fig. 2.6. This averaging filter produced the slowly moving bulk load of Fig. 2.7. The technique of working with the load fluctuations only, with the bulk load portion of the signal removed, more closely mimics standard practice in frequency regulation, which is to manage the area control error (ACE) rather than to follow the total area load ([76], see Section 1.1).

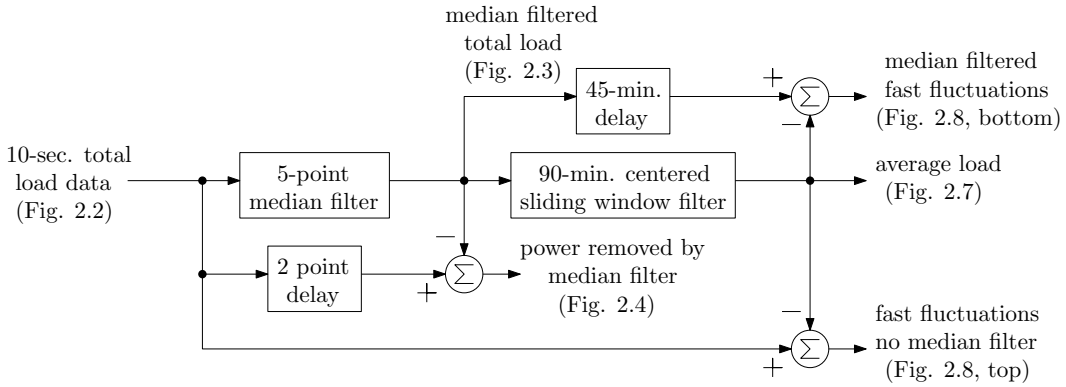

Figure 2.6: Block diagram illustrating the preliminary manipulations of data set A. The median filter removes anomalous points, and the predictive centered sliding window filter separates the bulk load from the rapid fluctuations.

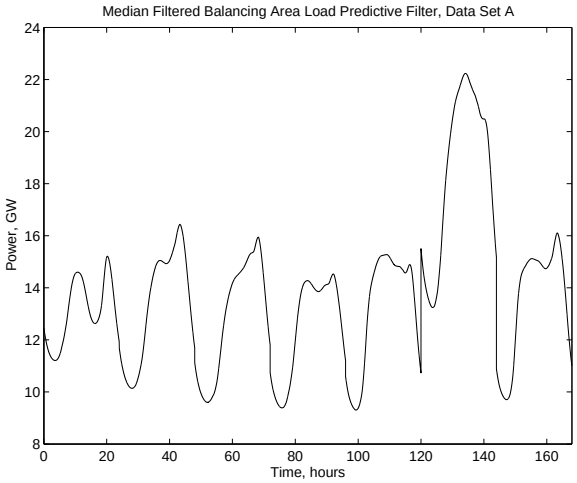

Figure 2.7: Average power component in data set A of Fig. 2.3 obtained by using a non-causal (predictive) sliding window filter centered around an interval of 90 minutes. This is used as a proxy for the area load prediction, and is subtracted from the load of Fig. 2.3 to yield the rapid fluctuations of greatest interest here.

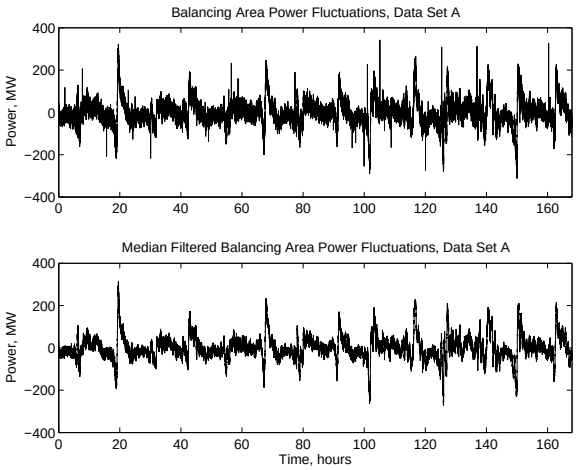

Figure 2.8: Fluctuation component of the balancing area load of data set A, obtained by subtracting the average power of Fig. 2.7 from either the raw or the median-filtered data of Fig. 2.2 or 2.3. In this figure, the effect of the median filter at removing single anomalous points and slightly decreasing the total variability becomes evident.

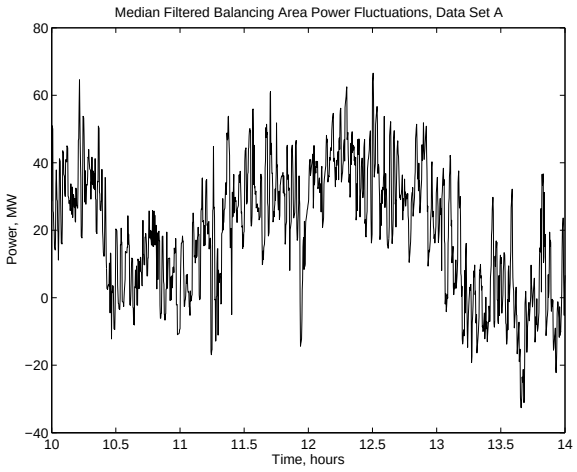

Figure 2.9: A portion of the fluctuation component of the balancing area median filtered load of data set A, obtained by subtracting the average power of Fig. 2.7 from the median-filtered data of Fig. 2.3. The same plot as Fig. 2.8 (bottom) on a different scale, to make the shape of the data more visible.

The remaining fluctuations, found by subtracting from each point of the original data set the corresponding average power point, may be seen in Fig. 2.8 for both the median-filtered and the raw data. In this format, the effect of the median filter becomes more obvious, as the data spikes are largely removed and the total variation is somewhat smaller. A portion of the median filtered data is included as Fig. 2.9 to show the behavior of the signal on a tractable time scale. Note that these fluctuations include both negative and positive power values, which corresponds to the fact that these fluctuations take place around a bulk power generation setpoint. This use of positive and negative power terminology is consistent with the conventions of LFC [56].

Figure 2.8 also makes clear the need for the additional analytical tools which are discussed in this chapter. While some basic information about the signal is available from the graph, such as the approximate average value and the size of most of the variations, it is difficult to understand the signals because of the mix of timescales involved.

2.3.2 Data Set B

Another publicly available data set, here referred to as data set B, is the regulation data from the PJM Regional Transmission Organization (RTO) in the eastern U.S. The PJM balancing area has made regulation control data sampled every 4 seconds available at its

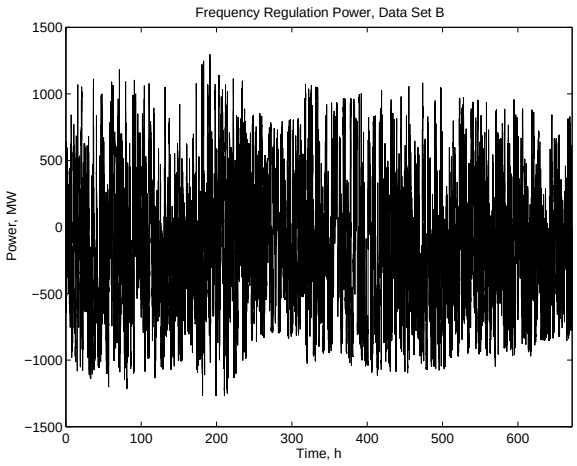

Figure 2.10: Data set B: raw 4 second PJM regulation data from 4 non-consecutive weeks: 1–7 July 2010; 1–7 September 2010; 1–7 January 2011; and 1–7 March 2011.

website[5] for four one-week periods: 1–7 July 2010; 1–7 September 2010; 1–7 January 2011; and 1–7 March 2011. This is the historical control signal sent to the generators performing frequency regulation during those weeks. The raw regulation data is presented in Fig. 2.10. A portion of the data is shown in Fig. 2.11 to show the behavior of the signal on a more tractable time scale. This is a regulation-only data set so it need not be separated into bulk energy and fluctuations like data set A was (as described in the previous subsection). Data set B also does not seem to contain anomalous points like those noted in data set A above, possibly because it is the output of a control system rather than inputs recorded from various generators, so median filtering was not considered necessary for data set B. For comparison, the total hourly load for the time period included in data set B is plotted in Fig. 2.12.[6]

[5] http://www.pjm.com/markets-and-operations/ancillary-services/mkt-based-regulation.aspx hosts Excel files under the heading "RTO Regulation Signal Historical Raise/Lower Percentages" which contain 4 second samples of total control area regulation signal, available as of 25 Jan 2012. Data set B is taken from the column labeled "RTO RegA" in the spreadsheet file, corresponding to the traditional regulation dispatch signal for the PJM control area, as confirmed by personal communications with Scott Baker of PJM.

[6] The hourly bulk load data was included in the monthly regulation `csv` files at `http://www.pjm.com/markets-and-operations/market-settlements/preliminary-billing-reports/pjm-reg-data.aspx` as of 25 January 2012. The column labeled "Total PJM RT Load (MWh)" was used to generate Fig. 2.12.

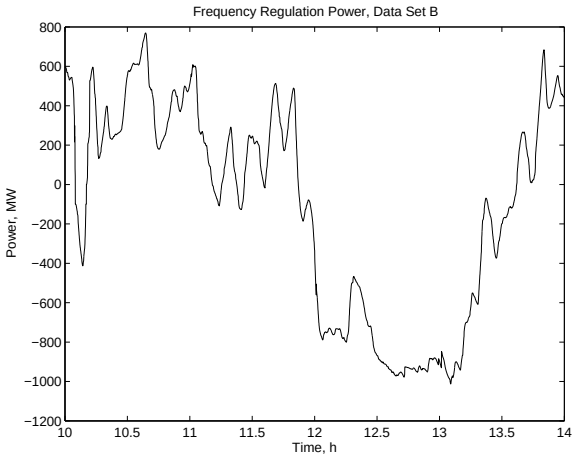

Figure 2.11: Data set B: a portion of raw 4 second PJM regulation data from 1 July 2010. The same plot as Fig. 2.10 on a different scale, to make the shape of the data more visible.

2.4 Power Duration Curves

One way to more easily interpret a graph like Fig. 2.8 is by using a power-duration curve. This is analogous to a load-duration curve used for economic dispatch or reliability studies [4], but here we are still considering only the load fluctuations like those in Fig. 2.8.

A power-duration curve used here is a graph of the fraction of time that a single or collective resource is acting at a given power level or lower. The opposite convention from regular load-

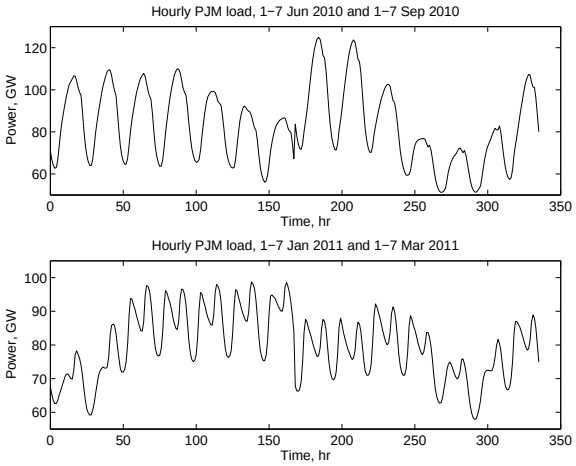

Figure 2.12: PJM total hourly load for the four weeks during which data set B, shown in Fig. 2.10, is available: 1–7 July 2010; 1–7 September 2010; 1–7 January 2011; and 1–7 March 2011. This bulk load data may be compared to the bulk load of Fig. 2.7.

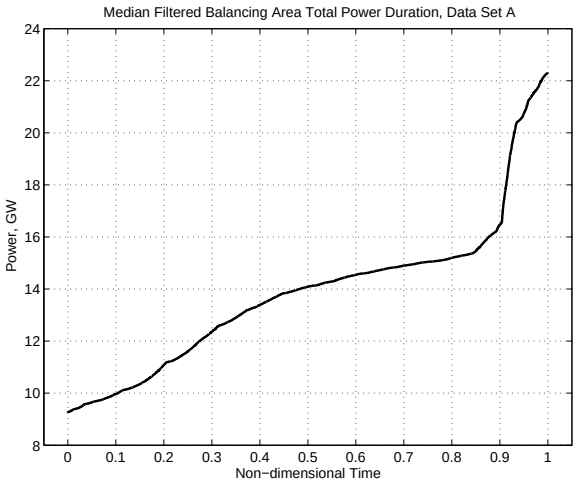

Figure 2.13: Total load power duration curve for the 10-second median-filtered balancing area data of Fig. 2.3. This graph is dominated by the bulk energy delivered, and the 7 days covered by the data are probably not representative of the annual bulk load.

duration curves is used to highlight the fact that this includes very short-term variations, and not the hourly or average dispatch of conventional load-duration curves. Figure 2.13 is a power-duration curve for data set A of Fig. 2.3 and Fig. 2.14 is a power-duration curve for only the median-filtered fluctuations, corresponding to Fig. 2.8 (bottom). Similarly, Fig. 2.15 is the cumulative power duration curve for the PJM regulation data of Fig. 2.10. The curves indicate the fraction of time for which the total load power is the indicated value or smaller. While the total power duration curve of Fig. 2.13 is dominated by the bulk energy component, in Fig. 2.14 the distribution of the fluctuations may be easily examined.

This graphical way of interpreting the data is useful for quickly determining the required (power) capacity for regulation, but it does not offer information concerning the cycles in which it might be used. For example, both slow, 30-minute cycles and fast 3-minute cycles could produce the same power-duration curve. For this reason, the ramp-rate-duration curves described in the next section were developed.

2.5 Ramp Rate Duration Curves

For traditional thermal and hydroelectric generators, there are limits on the rate of change of power output, or power ramp rate (see Section 3.1). A ramp-rate-duration curve displays the ramping capability required by a signal in much the same way as a power-duration

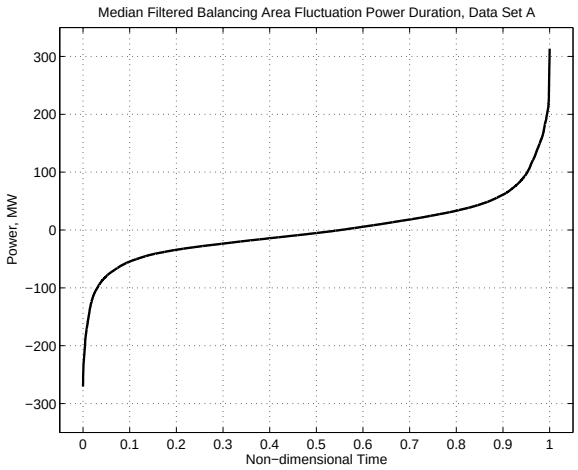

Figure 2.14: Fluctuation power-duration curve for the 10-second median-filtered balancing area data of data set A, whose time series is included in Fig. 2.8 (bottom). This graph more clearly shows the power capacity required to account for the fast fluctuations of demand. Note that because this is a fluctuation-only signal which is delivered on top of the bulk power generation, negative power values indicate operation below the bulk power setpoint rather than a negative power output.

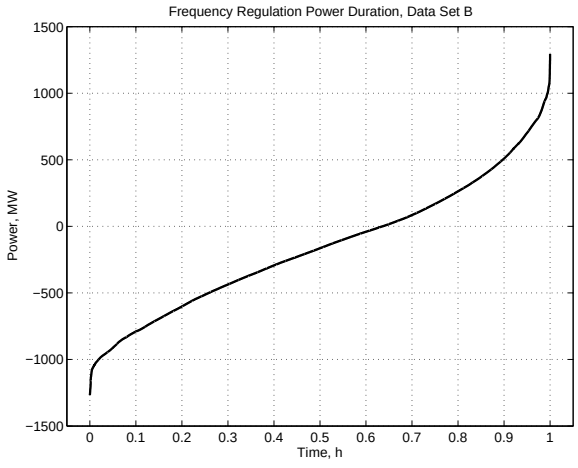

Figure 2.15: Cumulative power duration curve for data set B, 4 non-consecutive weeks of 4 second PJM regulation data of Fig. 2.10. Note that because this is a fluctuation-only signal which is delivered on top of the bulk power generation, negative power values indicate operation below the bulk power setpoint rather than a negative power output.

curve displays the use of power capacity. It is a visual representation of the fraction of time that a certain total ramp rate is required of a generating system.

A ramp rate duration curve can be constructed by first determining the ramp rate by taking the derivative (or finite differences) of the dispatched power curve. Note that this is not necessarily a long-term ramp rate sustained for an extended period of time, as might be seen during startup, shutdown, or load pickup, but is an instantaneous (10 second or 4 second) ramp rate from one data point to the next. Because the instantaneous ramp rate is produced by a numerical differentiation operation, it is relatively susceptible to noise generated by errors in measurement. When the data points have errors, the error in the instantaneous ramp rate is amplified. However, to the extent that the data points represent the actual or required instantaneous total power output of a set of generators, the numerical ramp rate represents the actual or required combined instantaneous ramp rates of those generators.

The ramp rate duration curve is then created by tallying the fraction of time the ramp rate is at or below a certain level, for example by sorting the ramp rate curve data. A related approach was used for wind time-series data in [99] and for the binned single-sample changes in power for frequency regulation in [68].

The ramp rate time series curve for data set A set, both raw and processed through the median filter, is shown in Fig. 2.16, but is difficult to interpret. While some difference in the size of variations is evident between the raw and the median-filtered, data, any further structure in the graphs is not apparent at these time scales. The larger size of variations in the original as compared to the median filtered version of the ramp rate is probably because the median filter suppresses fast, single-point changes in the power level which are otherwise reflected in the ramp rate. For comparison, a similar graph is provided as Fig. 2.17 for the data after processing by a 3-point or a 7-point median filter, rather than the 5-point median filter used in general. While data treated with the 3-point filter include some isolated points with a very high ramp rate that do not occur in the data after treatment with the 5-point filter, the ramp rates of the data after the application of the 5-point and the 7-point filter look largely similar.

The ramp rate time series for the PJM regulation data is included as Fig. 2.20.

The ramp rate duration curve corresponding to Fig. 2.16 (bottom) is in Fig. 2.18, and ramp rate duration curves reflecting the two alternate median filters included in Fig. 2.17 are pictured in Fig. 2.19. Likewise Fig. 2.21 is a similar curve for data set B of Fig. 2.20. Figure 2.22 is a closer view of the ramp rate duration curve of Fig. 2.21 to make the characteristics in the middle of the range more visible.

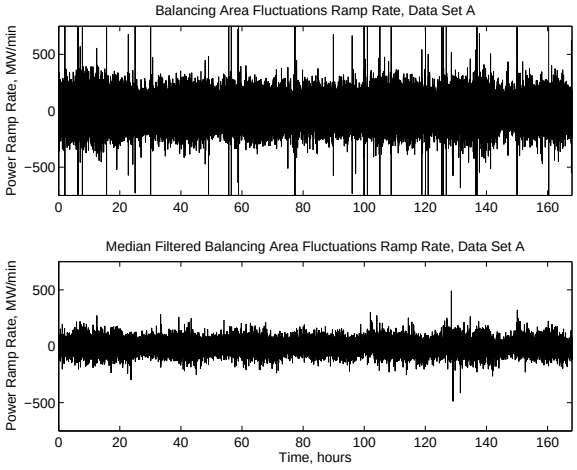

Figure 2.16: Instantaneous ramp rate for raw and median filtered versions of data set A of Fig. 2.8, obtained by taking first differences of the data points. Note that the median filter leads to less spiking in the ramp rates and lower ramp rates overall. The bulk energy has been removed, as it contributes very little to ramping.

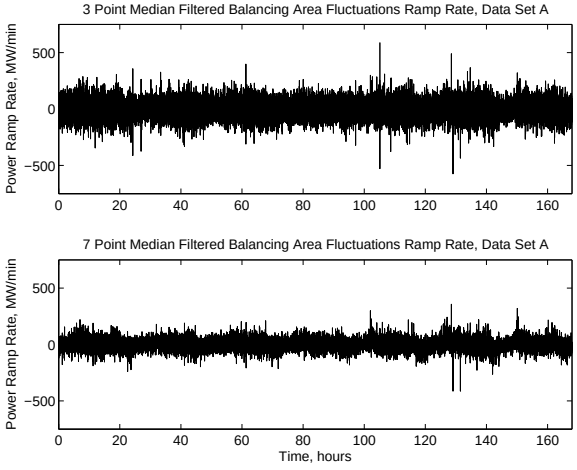

Figure 2.17: Instantaneous ramp rate for data set A of Fig.2.8 (top) after treatment with a 3-point and with a 7-point median filter. The 3-point median filtered data includes a number of locations with unusually large ramp rates, while the ramp rates of the 7-point filtered data are more similar to those of the 5-point filtered data.

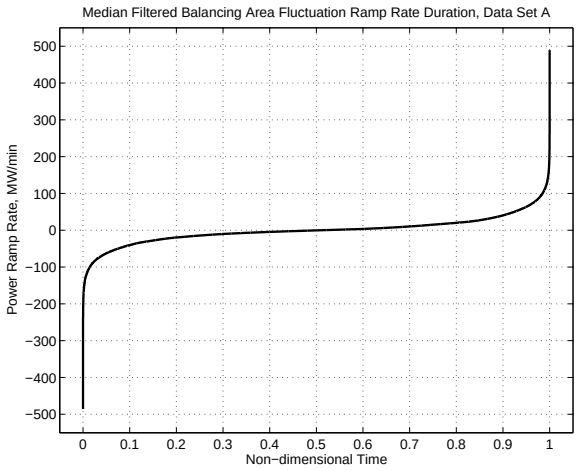

Figure 2.18: Ramp rate duration curve for data set A power fluctuations after application of 5-point median filter (corresponding to Fig. 2.8, bottom).

The ramp rate duration curves clearly indicate the potentially large ramp rate requirement of the regulation signals, compared to the capabilities of thermal units [51].Although some of the larger ramp rates indicated by these plots may still be due to data collection errors, data set A indicates an average absolute ramp rate of over 24 MW/min, with a substantial fraction of time spent at 40 MW/min or more, provided by a total generating capacity of 10-20 GW. Data set B indicates an average absolute ramp rate of 69 MW/min for a total generating capacity of 60-120 GW, and the signals indicate a requirement in excess of 100 MW/min a substantial fraction of time.

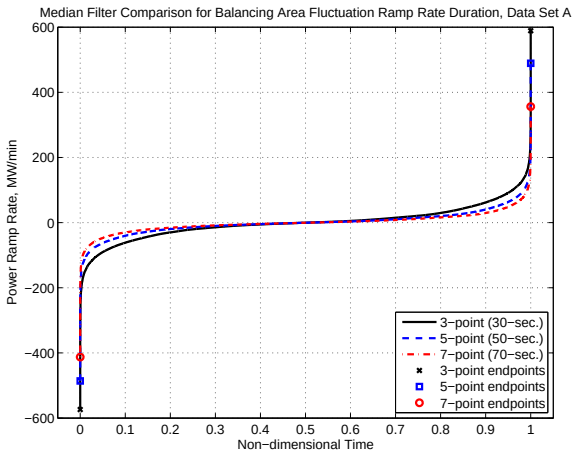

Figure 2.19: Ramp rate duration curve for data set A power fluctuations using the three median filters which have been considered (corresponding to Fig. 2.17).

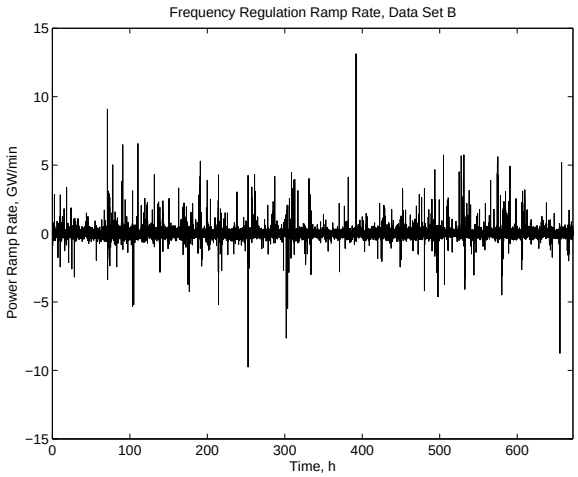

Figure 2.20: Instantaneous ramp rate for data set B of Fig. 2.10.

The reason for using a ramp rate duration curve stems from a performance difference between traditional thermal assets and energy storage units, discussed in more detail in Chapters 3 and 4. While in general energy storage units are able to ramp from one power level to another very quickly, ramping with thermal units is slow and is more expensive than steady state operation [27, 68, 63]. From the ramp rate duration curve of an LFC signal sent to a thermal unit, both the maximum ramp rate to be required of the unit and the fraction of time the unit is ramping at any given rate are clear.

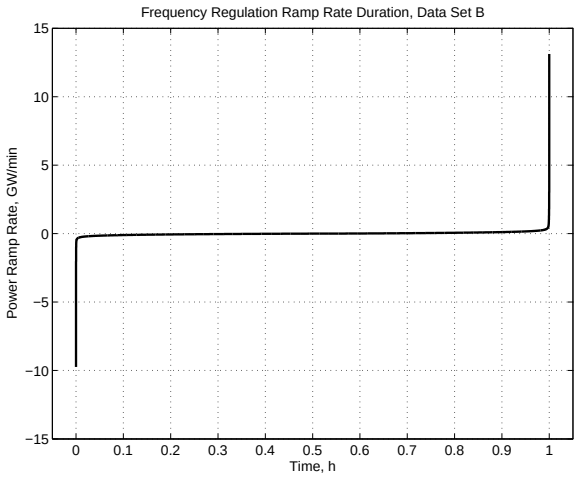

Figure 2.21: Ramp rate duration curves for data set B of Fig. 2.20.

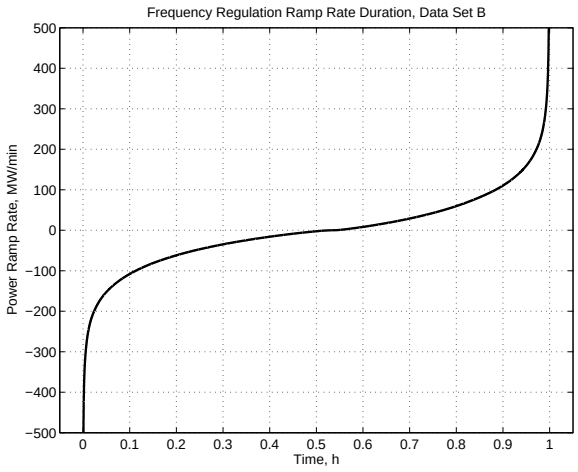

Figure 2.22: Ramp rate duration curves for data set B data of Fig. 2.20. Identical to Fig. 2.21 except for axis scaling; the curve goes off the scale at the extreme ends of the range.

2.6 Energy Duration Curves

The energy duration curve is similar to the ramp rate duration curve, but tallies net energy required at each instant. This can be done by integrating or accumulating the power curve over time. An energy duration curve is primarily of interest for storage resources that are constrained to inject zero average power.[7] It may also be useful for assessing the dispatch of other energy-constrained resources such as hydraulic units with reservoirs or solar-thermal units with thermal storage by indicating their state of charge distribution. Because a nonzero average power value will lead to a ramp in energy upon integration, the energy-duration curve for a fossil-fueled thermal unit is of less interest. The energy represented by the fluctuations of data set A of Fig. 2.8 (bottom) is pictured in Fig. 2.23. The energy required to conform to the data set B regulation signal of Fig. 2.10 is shown in Fig. 2.25. Note that the absolute energy values in the energy duration curves are of less interest than the total change, because the absolute energy levels are dependent only on the initial state of charge of the storage unit. For this reason, the data have all been shifted vertically to set the low energy point for each plot to zero, and the shapes of the graphs should be compared rather than their vertical positions. Each separate time period (day or week) starts at the same energy value in these plots.

The energy-duration curve is created by tallying the percent of time that the net energy requirement is at or below a certain level. (For example, this can be done by sorting the

[7] In fact, energy storage devices will generally draw a small amount of average power to compensate for losses during energy conversion and storage.

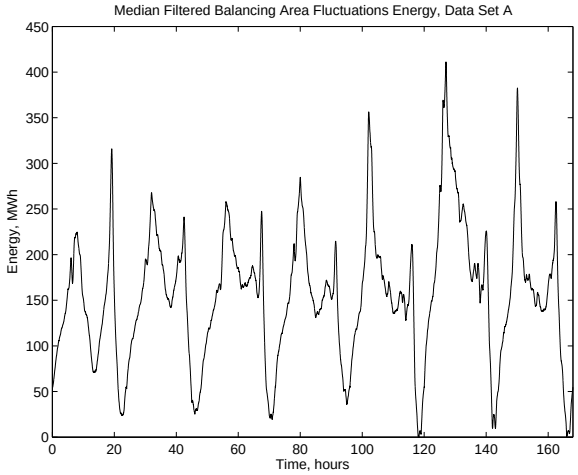

Figure 2.23: Energy represented by median filtered data set A of Fig. 2.8 (bottom), obtained by taking accumulations of power data. Because the absolute energy levels represent only initial conditions and are less important than the changes in energy, the graph has been shifted to have a minimum at zero energy. Each non-consecutive day starts at the same initial condition.

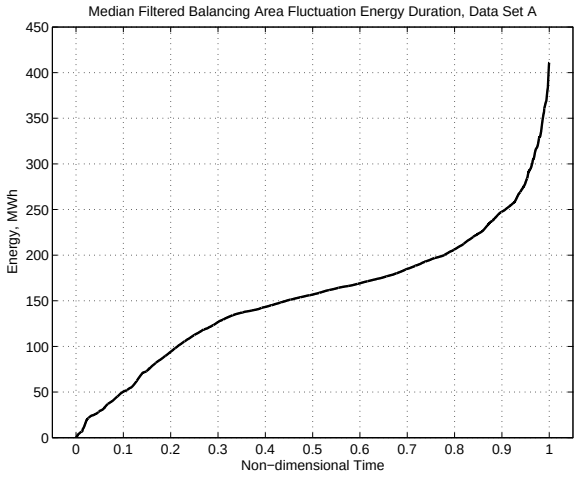

Figure 2.24: Energy duration curve of median filtered data set A fluctuations, derived from Fig. 2.23. Again, the absolute energy levels have been shifted to produce a minimum at zero energy, and each non-consecutive day starts at the same initial condition.

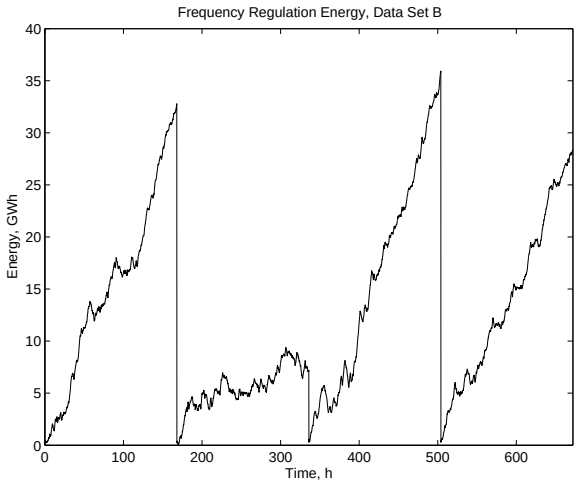

Figure 2.25: Energy delivery required as indicated by data set B of Fig. 2.10, obtained by taking accumulations of power data. Because the absolute energy levels represent only initial conditions and are less important than the changes in energy, the graph has been shifted to have a minimum at zero energy. The 4 non-consecutive one week intervals each start at the same initial condition.

data points.) Figure 2.24 shows the energy duration curve corresponding to data set A of Fig. 2.23. The energy duration curve corresponding to the data set B regulation signals of Fig. 2.25 may be found in Fig. 2.26.

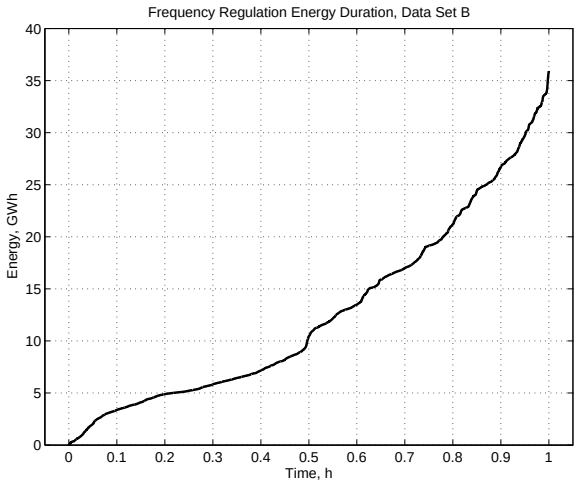

Figure 2.26: Energy duration curves of data set B from Fig. 2.25. Again, the absolute energy levels have been shifted to produce a minimum at zero energy and each non-consecutive week begins at the same energy level.

In this example, it is clear from Figs. 2.23 and 2.26 that the regulation signal and the fluctuations produced by subtracting out the slow predicted average power still indicate a very large amount of energy. The total amount of energy required to perfectly follow the data set A fluctuation signal is over 400 MWh. The corresponding amount of energy required to perfectly follow the data set B regulation signal is over 35 GWh. These values represent impractically enormous storage units for most technologies (see Chapter 4). This indicates the necessity of the technique described in Chapter 5, of separating the load power fluctuations into a component to be provided by energy storage and a component to be provided by more traditional generation assets.

2.7 Summary

This chapter describes how the power, ramp rate, and energy duration curves can facilitate the evaluation of regulation data signals. Ramp rate duration curves illustrate the fast changing nature of the signals, which can be difficult for traditional thermal power plants to follow, as described in Chapter 3. Energy duration curves make clear that the raw regulation signals are not suited to provision entirely by energy storage units because of the prohibitively large storage capacity that the raw signals require, and power duration curves demonstrate the total capacity required on regulation duty and the frequency with which that capacity is called upon.

The two available data sets described in Section 2.3 will be used throughout this thesis to illustrate the strategies which may be used to provide regulation services with a mix of storage and traditional thermal generation assets. Data set A has the advantage of representing actual generation data, so that it does not indicate more fluctuation than can be reasonably required for good control. Data set B has a faster 4 second sampling rate that can capture somewhat more of the interesting dynamics in this regime; however, because it is a regulation requirement signal rather than a generated power signal, it is possible that it indicates more ramping than is actually needed. In general, data set A processed with the 5-point median filter will be used in the work that follows alongside data set B.

The ramp rate, energy, and power duration curves described in this chapter are useful tools for evaluating the combined thermal and storage dispatch strategies for LFC provision which are examined in Chapter 5.

Chapter 3

Pertinent Characteristics of Thermal Generators

3.1 Thermal Generators and Frequency Regulation

JUST as the overall generation mix in most areas is dominated by traditional generation resources such as fossil-fueled steam and combustion turbines along with hydroturbines [105], the provision of frequency regulation is primarily provided by these traditional resources as well [56]. In general, frequency regulation is performed by thermal or hydraulic power plants connected to the automatic generation control (AGC) signal generated by the system operator. This signal is generally updated every 2-10 seconds, and either indicates the new requested power output for the generator or, for some older plants, indicates whether the power setpoint should be raised or lowered. The signal is produced by the system operator for the balancing area, and generally represents a filtered version of the Area Control Error (ACE), as described in Section 1.1.

In general, frequency regulation is performed by a subset of the various power plants in a control area which have the capability to respond to an AGC signal. The fastest-responding traditional generation resources tend to be hydroturbines, which are often used for frequency regulation when they are available (and auxiliary constraints on water flow allow; see Section 4.2)[11]. Combustion turbines and slower-responding combined cycle and steam turbines make up the remainder of the power capacity used to perform frequency regulation. At this time, unconventional resources like energy storage, coordinated demand resources, and alternative generation are involved in frequency regulation only in isolated and experimental situations, although their impact in this sector is growing (see Section 1.1 and Chapter 4).

Power plants performing frequency regulation generally have only a fraction of their capacity committed to AGC. It is suggested in [42] that 3–5% of the capacity of a generator on AGC is committed to frequency regulation. According to the analysis of regulation bids for a summer afternoon hour in California described in [68], the (gas-fired) steam turbines bid

an average of 9% of their capacity into the regulation market, while the combined cycle and hydraulic units bid an average of 16% of their capacity.

3.1.1 Changing Demand for Regulation Capability

Environmental concerns and limitations in the supply of fossil fuels are causing pressure on the electric power sector to incorporate increasing amounts of nontraditional generation resources, primarily wind turbines, with some solar thermal and solar photovoltaic installations also entering the generation mix. These resources are generally intermittent and variable, exhibiting frequent unpredictable changes in power output. This variable generation is likely to contribute to the regulation requirement of a control area. In [69], the authors predict a modest increase in both regulation capacity and regulation ramping requirements as a result of increased wind penetration. The review of several wind integration studies in [81] also concludes that there is a moderate increase in regulation requirement with the incorporation of wind generation. Reference [108] reaches a similar conclusion.

In addition to any increase in regulation requirement as a result of intermittent renewable generation, since renewable generation often does not itself participate in frequency regulation, a larger share of the increased frequency regulation requirement must be provided by the fewer required traditional resources on a system when an appreciable portion of variable renewable generation is installed in a control area.

Furthermore, the bulk load distribution in the United States is shifting overall, with a larger fraction of load demand occurring only a few hours a year, and a smaller fraction of load which is always drawn ("baseload") [53]. These factors combined complicate the problem of scheduling generators for AGC, because the generators which are providing frequency regulation must be running at a substantial fraction of their load. If available on-line resources are unable to provide the required capacity for frequency regulation, other units must be allowed to run at higher cost in order to provide that capacity.

3.2 Limitations of Thermal Generators for Frequency Regulation

Traditional thermal generators respond much more slowly to control signals than many storage technologies. Hydroturbines can also sometimes respond very quickly especially when compared to thermal generators and as such their use is favored for frequency regulation. However, the amount of new hydropower which can be built in the US is extremely limited

(see Section 4.2). This means that the degree to which hydropower can provide the rapidly changing power required for frequency regulation is limited largely to existing installations. The response of hydroturbines to power control measures is itself not ideal, as the dynamics of the water can lead to an initial decrease in power when an increase is requested, because of water flow dynamics in the penstocks described by right half-plane zeros [112]. A battery energy storage system was added to the grid in Metlakatla, Alaska partially because the existing hydro generating units "lack the speed of response required to follow the loading fluctuations" from a large sawmill load [73].

There has been an ongoing discussion on the "right amount" of control to provide for many years. For the most part, this thesis intentionally sidesteps this question of the fidelity with which frequency ought to be controlled. As this Chapter indicates, frequency control can be expensive, but the lack of sufficient control also has negative implications for power quality and system security as well as the relations among neighboring control areas. In the U.S., minimum standards for control are set by a collaboration of industry and regulatory groups (see Section 1.1). This section presents evidence from the literature that existing thermal generation has difficulty providing sufficient control, or is not fast enough.

The speed of response of thermal power plants to requested changes varies among unit types, but in general such changes occur over the course of a few minutes. Reference [55] includes examples of both good and poor generator response to AGC. In [51], the authors discuss in general terms the limitations in response of existing generation. Generation rate constraints on the order of 3% of capacity per minute are named. It is suggested in [84] that the pending generator response should be fed forward to calculations of future control action to help prevent over-control. A generic description of the response of steam powered units to load changes is available in [47], which also suggests that the maximum setpoint rate of change is less than ±9% per minute. Rate limits of 4–10 MW/min were found on the units examined in [25]. Some design choices intended to decrease certain types of thermal stress associated with load setpoint changes, such as variable-pressure operation and turbine-follows-boiler control, also decrease the responsiveness of the generator to control changes [62, Chapter 7] (see Section 3.3).

In the course of developing models for the response of power plants to both governor action and LFC, the authors in [50] estimate that the average delay (in addition to first-order time constants) between LFC command and power response is generally in the range of 10–80 seconds, with the slower values occurring in coal units and units using sliding pressure control. Similarly, a 140 MW unit is modeled in [39]. The response of the system to throttle valve motion causing a –6 MW reduction in output power takes about 50 seconds, and the response to a boiler step which increases the throttle pressure and raises the power output

8 MW takes about 5 minutes. For an overview of some of the most common models used for thermal generators in stability and similar studies, see Appendix A.

Of the 120 units connected to the CAISO AGC system as of [68], the ramping capability of each unit ranged from 3 MW/min or less to more than 150 MW/min, although the large majority of units displayed ramping capability of less than 40 MW/min. In general, the (gas-fired) steam turbines and combined cycle units could ramp at about 2% of capacity per minute, hydraulic units could provide about 30% of capacity per minute (or more) and combustion turbines could ramp at about 20% of capacity per minute.

The improvement of AGC control performance in the participating utilities was a stated goal of the transfer of regulation responsibility from several control areas to a nearby hydro-dominated utility in [85]. In some control areas, loads are unusually variable, and this can make it difficult for the system operator to procure enough control for the area, as measured by minimum control regulations. This is particularly common in areas with large amounts of high-powered industrial load (such electric furnaces, mills, or mining equipment). For example, in [42] the difficulty that one control area has in following the rapidly-changing load from electric furnaces is described. Reference [22] concerns a control area with a large and rapidly changing industrial load component. The system operator has difficulty keeping up with ACE and with control criteria[1] because of the "non-conforming" industrial load. The solution as described attempts to separate that portion of load which can be followed by AGC from that which cannot. It was found that when the slow generators on AGC were subject to rapidly changing control requests, the resulting control performance deteriorated, rather than improved. The description of the improved control strategy in [88] notes that there are classes of disturbances that ought not to be followed, because of the increased expense of control yielding results only marginally improved, if at all.

The effectiveness of various speeds of regulation is explored in [68]. The benchmark control system used in that work predicts violations of control performance standards (see Section 1.1) based on simple persistence forecasts of load variation and requests the minimum control action necessary to avoid those violations. This benchmark, referred to as "optimal" control, assumes perfect responsiveness and no limits on the capacity, energy, or ramp rate of the controlling generators. However, this formulation of optimal control places an emphasis on fast resources and uses very limited prediction. Using this metric, it is calculated that ideal resources are about 1.7 times more effective at regulation than average hydroturbines, about 2.7 times more effective than average combustion turbines, and about 28 or more times more effective than average gas-fired steam and combined cycle plants. A storage resource with 15 minute duration is, by this metric, nearly as efficient as

[1] At the time of this reference, the older NERC control criteria known as A1 and A2 were still in force.

the ideal resource (1.4 times better than average hydropower, 2.2 times better than average combustion turbines, and 23 times better than average steam or combined cycle plants).

3.3 Penalties for Thermal Generation Incurred by Frequency Regulation

There are several mechanisms for operation of thermal generation for frequency regulation can lead to increased operating expense. One is that some efficiency penalty from transient operation is likely. A second increased expense is due to additional wear from transient operation which results in higher maintenance costs and more frequent repair. A third difficulty is that generation must be operating at part load in order to have room to increase power output in response to an AGC signal, and part load is usually not the most efficient operating point. Additionally, the necessity of part-loading generators performing frequency regulation can require out-of-order dispatch, which raises the overall system operating cost.

3.3.1 Ramping Efficiency

For thermal generators, continual power ramping like that required for frequency regulation is not as efficient as operation at a steady power level. Several mechanisms leading to an efficiency penalty for ramping are mentioned in the literature, including entropy generation at valves; inadequately controlled fuel-air mixture in the boiler; and human error. It is important to recognize that any efficiency penalty is applied to the entire power output of the plant, not only to the portion of the output used to perform frequency regulation. This means that even a small decrease in efficiency can be substantial compared to the capacity engaged in frequency regulation.

In [60], it is argued that reduced plant efficiencies amounting to 1% or more may be expected when operating under AGC. This is consistent with the statement in [10, p. 23] that a 1–2% drop in efficiency was found with frequent reversals of an AGC signal in an ABB simulation, compared to constant power operation. A figure plotting the efficiency loss of a steam generator as a function of the degree of throttling is offered in [29] and reproduced as Fig. 3.1. The figure indicates a similar 0.5–1.5% change in efficiency as the degree of throttling increases, and the figure is offered during a discussion of frequency response and regulation rather than as regards slower load following or other large power setpoint changes.

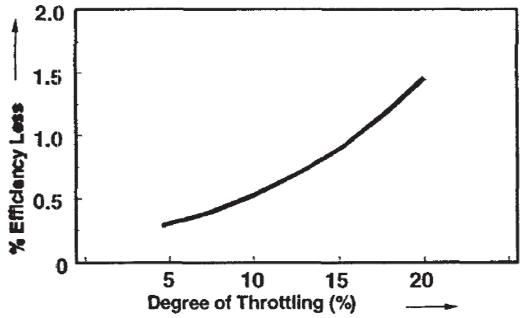

Figure 6 : Efficiency Losses due to Steam Throttling

Figure 3.1: The expected change in efficiency as the degree of throttling is increased in a steam plant. Reproduced from [29].

As a very rough estimate of the monetary costs of an efficiency drop due to frequency regulation, consider a scenario where a 500 MW generator performs frequency regulation using 6% of its capacity, or 30 MW. If the generator has a fuel cost of about \$30/MWh-electric, a thermal-to-electric efficiency of about 50%, experiences an efficiency reduction of 1% due to operation under AGC rather than at a constant power output, and operates at an energy setpoint of 485 MW to allow room for its power output to increase and decrease by ±15 MW, then the efficiency penalty will increase the total input fuel costs by about \$0.6/MWh-electric,[2] for a total cost increase of nearly \$300 for the generator output over the course of an hour. If this increase of \$300 is divided over the 30 MW capacity performing regulation, it corresponds to a capacity cost of frequency regulation of about \$10/MW for the hour resulting only from the additional cost of the power fluctuations. While this example does not provide an exact estimate, it does indicate the large cost imposed by a relatively small drop in plant efficiency due to frequency regulation.

By contrast, in [26], the authors build a physically-based model for a steam power plant, including dynamics of the boiler, the turbine, the feedwater cycle, and some auxiliary equipment. A careful accounting is made of all the energy flows in the system to determine what, if any, efficiency penalty there is for fast, small cycles. The authors conclude that there is a calculated but exceedingly small efficiency penalty for these cycles. Unfortunately, there is no empirical data to which the physical modeling may be compared,[3] rendering the resulting efficiency numbers less convincing.

[2]The change in fuel cost of power is the difference between the before and after values of the fuel price per Btu divided by the thermal-to-electric efficiency, or $\frac{\Delta\$}{P_{out}} = \frac{\$/P_{in}}{49\%} - \frac{\$/P_{in}}{50\%}$, where $\frac{\$/P_{in}}{50\%}$ is equal to the fuel cost of \$30/MWh-electric.

[3]The authors mention a planned study of a physical plant, but no evidence of this later study was found.

While [100] attempted to evaluate the costs of AGC using a statistical study of very granular operating costs, they did not compare these costs with the operation of similar plants under constant conditions. Instead, the successive power and ramp rate points were treated individually as snapshots, and regressions were performed to evaluate the effect of ramping. The results showed only a minimal efficiency penalty resulting from AGC. However, if the decrease in efficiency due to AGC results from transients in the process dynamics, one would not expect to find such a decrease by comparing different instantaneous heat rate values and their power and ramp rate values. The dynamics mean that the successive points cannot be analyzed separately. As a result, this work does not seem to make a strong argument against the existence of an efficiency penalty for plants performing AGC.

Other sources also hint at the expected efficiency penalties associated with the generator ramping necessary for load frequency control. For example, [97] estimates that reducing generation "swings" by 50% could result in a fuel savings of $1 million per year.

In [110], tests on generators under manual control designed to mimic AGC found that a ±10 MW variation in power output for three different units caused an increase in fuel costs of up to 3.8% (and an average increase of 1.4%), although the authors caution that any error introduced by test equipment may be large compared to the reported effects. Tests of transferring AGC control from thermal-dominated utilities to a nearby hydro-dominated utility were reported in [85], and indicated fuel savings of a fraction of a percent were likely, such as 0.35% for one 535 MW unit regulating versus flat load and 0.36% for another participating utility.

3.3.2 Wear Effects

When thermal power plants perform AGC, their setpoint must change rapidly, and this can cause additional wear in a system which is designed for constant-power operation, as most power plants are. Wear effects can include valve damage from frequent operation, thermal or mechanical stresses in components, damage from insufficiently controlled steam or boiler chemistry, and an increased risk of damage caused by human error due to transient operation.

The majority of the literature relating to power plant wear due to transient operation focuses on daily cycling and load following. This is a mode of operation where plants are ramped from a low load point (perhaps 10–25% of rated power) to full load over a large portion of their operating range, in a time of tens of minutes to hours, and induces different stresses from AGC [61, 96]. There is a consensus in the literature that deep cycles cause appreciable wear. How this analysis relates to the shallower but more rapid cycles of AGC is less clear.

One possible wear effect associated with ramping is the thermal gradient across the rotor, as explored in [20]. It is noted that the temperature of the rotor grows or decays exponentially depending on the interior and exterior temperatures. If this thermal gradient exceeds an allowed value, then some rotor damage is expected. If the predicted thermal state is controlled and used to dynamically limit ramp rate, the overall wear can be reduced or a faster allowed ramp rate allowed under some circumstances, compared to a static ramp rate limit. An approximated temperature change of 2°F/MW is used, which may also be applicable for the smaller cycles considered here. If there is a temperature change when the power level is changed, then there is the possibility of thermal gradients resulting in thermally-induced wear. This thermal gradient damage calculation is used along with manufacturer data in [109] to perform an economic dispatch considering the ramping cost associated with generator setpoint changes. This is in contrast to the usual strategy of performing economic dispatch constrained by fixed ramp rate limits. Again, the focus is exclusively on the load-following ramp events preceded and followed by steady state operation.

In [63, 64], increases in unit operating costs due to cycling are described, including shortened component life due to creep-fatigue interaction, increased forced outage rates, and increased likelihood of operator error during transient operation. It is asserted that these costs are frequently underestimated by plant owners and operators. The importance of adequately controlling plant chemistry, especially during transient operation, is also mentioned. The work presents the correlation between empirical operating profiles, outage rates, and operating costs and describes a method to calculate cycling costs for a particular unit. While it does not address shallow power changes that may be expected in AGC operation, it is reasonable to suspect that some of the same damage mechanisms, such as creep-fatigue interaction and operator error, may be present in that operating mode. Related work catalogs damage mechanisms and includes additional data on numbers of plant starts versus operation and maintenance costs to examine the relation between cycling duty and cost [96].

A similar approach, taking large databases of power plant maintenance costs and comparing correlations between cycling duty and costs, is used in [23]. A pronounced increased cost was found with increased cycling, but modeling efforts to capture all such effects were hampered by the inconsistency and incompleteness of the available data sets.

A report which focuses on the effects of cycling on power plant turbines and generators in particular is [62]. In addition to detailed examination of the expected failure modes, historical data from many power plants over 20 years is compared, to evaluate the substantial increase in the likelihood of failure of particular components when subjected to cycling

operation. While the main focus of the work is start-stop cycles, frequency response and regulation is briefly mentioned as well.

There are a few sources which specifically address the component wear associated with frequency regulation. In [60], the mechanisms of component wear in valves and turbines are described, and a methodology is developed for determining regulating costs based on plant data. Reference [51] also repeatedly alludes to increased costs associated with wear and tear on generators for excessive or unnecessary control, although they do not list the particulars of this wear.

In the literature concerning AGC, an expectaction of increased wear and tear or maintenance costs which are attributable to LFC is frequently stated. For example, [97] reported that by a reduction in the generation swings for LGC, "savings in maintenance costs are also expected but no amount was estimated." Reference [110] expresses a similar sentiment. In [85], reduced wear and tear on generators as a result of the sharing of the regulation burden with a neighboring hydro-dominated utility was expected, but the value of these benefits was not calculated.

3.3.3 Part-load Effects

In order for a generator to provide frequency regulation, it must generally be operating in the middle of its range, with enough stable generating capacity above and below its setpoint to allow an increase or decrease in power generation in response to the control signal. This can lead to two types of negative effects: first, that the part load efficiency is frequently lower than the optimal operating point efficiency (generally near full load); and second, that generators which are able to provide regulation must be dispatched to this part-load point, potentially out of the normal dispatch order.

Units engaged in frequency regulation often have about 10% of their capacity performing regulation, meaning they need to be operating at least 5% away from their minimum or maximum load point. The efficiency of generators at part load is generally lower than the efficiency at the preferred operating point, usually at or near full load [60].

In both the open electricity markets and vertically-integrated control areas, power plants are dispatched in order of their variable costs from least expensive to most expensive. In a market, this dispatch order is based on the market bids; in a vertically-integrated system the order is based on calculated marginal cost [2, Chapter 11]. However, in order to provide frequency regulation, the power plants in the area must sometimes be dispatched out of order. Either a more expensive unit must be run at part load (where it would be off

otherwise) because it is needed to provide frequency regulation, or a less expensive unit must be run at part load (where it would be at full capacity otherwise) in order to provide head room for frequency regulation [42]. In both cases, bulk energy is being provided by a more expensive resource, despite the fact that a less expensive resource is otherwise available, in order to provide adequate control. These problems can be magnified on small or isolated systems [73].

In order to mitigate these problems, it would be necessary to commit less thermal generating capacity to frequency regulation while maintaining adequate control. However, the techniques described in Chapters 5 and 6 may be of limited use to reduce the traditional generation capacity required for frequency regulation, because the frequency regulation signals are dominated in power level by their slower components.

3.4 Performance Metrics

Since this thesis focuses on improvement of thermal power plant operations by the introduction of energy storage to perform a portion of frequency regulation, it is important to choose a metric by which to measure the benefits to be gained from the addition of energy storage. One clear benefit would be the reduction of total power capacity required for frequency regulation. However, because the regulation signals used here have a power range requirement which is dominated by very slow fluctuations, the techniques described in Chapter 5 do not provide a reduction in the thermal regulation power capacity requirement. Thermal (and hydraulic) power capacity required for frequency regulation remains an important consideration for future work.

The second metric, which is more relevant for this work, concerns the ramping of traditional generators. While it seems that low-capacity energy storage units may not be able to decrease the total power requirement for traditional generation performing frequency regulation, it is possible to substantially decrease the amount and speed of power output changes requested of those traditional units. One way to compare ramp rates is to consider the fraction of time that a generator spends ramping at a given rate, as with the ramp rate duration curves of Section 2.5.

A single figure of merit related to ramping performance is desirable, in order to compare the performance of different dispatch schemes, as in Section 5.2. For a particular unit, the ideal metric would take into account the effects of ramping at different rates, and would estimate the regimes where ramping was cheap or free (e.g. very low ramp rates); the regimes where ramping was expensive (e.g. high but permissible ramp rates); and where operation was not

possible or permitted. Such a metric would be the best way to compare the costs associated with different operating profiles. If the techniques in this thesis were to be applied to reduce ramp rates for a particular thermal generator, an analysis of this type would be valuable.

There are practical limitations with an ideal metric. One issue that the literature illustrates is that even for a given unit, the estimation of the costs associated with ramping is difficult and imprecise. Furthermore, the effects of ramping vary between units, depending on turbine type, fuel type, design, age, and operating mode. Very little information on unit ramping costs is publicly available, and variation among units makes a single metric used for all units problematic in any case. However, some way to compare the total ramping effect is required.

A common metric for the amount of control used in some AGC papers is the total change in power requested, both up and down, either as accumulated setpoint changes or total number of control pulses (for older systems). Work using this total includes [19, 84, 1]. A Texas control area with non-conforming loads describes in [97] their experience with an improved AGC algorithm, and uses both the total change in power as well as the number of requested changes of direction, or pulse reversals. Both total power and response speed are considered in the FERC regulation instructing market operators to compensate resources for performance in the regulation markets [32].

Reference [85] describes the transfer of regulation duty from several utilities dominated by thermal generators to a nearby hydro-dominated utility. In evaluating the benefits from this transfer, changes in total control power requests, requests for change in direction, and actual generation change and reversals were tracked. Additionally, "[i]t was hypothesized that even though the number of significant reversals of direction were reduced, the higher response rate when moving due to step changes in the setpoint may have lessened the fuel savings." This suggests that the total power metric used may not capture the most important effects of the change in regulation requirements in the participating utilities, and a metric which tracked the ramp rates of the participating units may be preferable.

A simple ramping metric of mean absolute ramp rate, $\left\langle \left| \frac{dP}{dt} \right| \right\rangle$ is used for most of the work in this thesis. Mean absolute ramp rate assumes that ramping up and ramping down are approximately equivalent, and indicates that a 2% per minute ramp rate is twice as costly as a 1% per minute ramp rate, and so on. This is essentially the same as the mean ramp rate of the controlled variable used as a performance metric in [88]. Mean absolute ramp rate is also used in [43] to compare the required generation characteristics on different time scales.

Another approach is to weight high ramp rates more heavily, for example with a root mean square (RMS) metric, $\sqrt{\left\langle \left(\frac{dP}{dt}\right)^2 \right\rangle}$. This classic measure of variation penalizes high ramp rates much more heavily than low ramp rates, which implies the hypothesis that low ramp rates are inexpensive and high ramp rates are very expensive. For example, a 2% per minute ramp rate is four times as costly as a 1% per minute ramp rate. The RMS ramp rate metric is also sometimes used in the work that follows, for comparison.

Both mean absolute ramp rate and RMS ramp rate are meaningful and helpful metrics to compare regulation profiles. While neither is an ideal measure of the cost of ramping for any given unit, both give a reasonable comparison of the intensity of ramping.

3.5 Conclusions

This chapter points to some ways that thermal power plant operation could be improved by removing some of the regulating burden from those power plants. The approach taken in this thesis is reduce the ramp rate of thermal generators engaged in frequency regulation, and thereby to avoid efficiency and wear penalties associated with excessive ramping. While the reduction of the total power capacity available for regulation by thermal generators would also be desirable, it was found that the techniques in this thesis do not provide this additional benefit. Future work which controlled the power levels of nontraditional resources such as demand response over longer time scales might be of help in reducing the power capacity required for regulation. For the remainder of this thesis, the primary performance metric for the improvement of operation of thermal generators performing frequency regulation is the required ramp rate.

Chapter 4

Characteristics of Storage Units

4.1 Introduction

ENERGY storage has played an important role in the electric grid since its inception [24] and has been considered or installed for many uses. Newer energy storage technologies have not yet been integrated into the electric grid in large amounts. Part of the reason that the capacity of grid-connected energy storage is small is because of the high cost of many technologies compared to bulk electricity generation. This has led to only very limited use of energy storage for bulk energy arbitrage. By contrast, high power, lower energy services like frequency regulation may be more readily delivered by energy storage. The rapidly controllable power from energy storage devices can be valuable for frequency regulation, as is discussed in this thesis. Energy storage is also being installed for applications such as transmission or distribution upgrade deferral, system stabilization, voltage and VAr control, and synchronized reserves, as well as for backup power, peak shaving, and renewables smoothing [38, 87].

There are some traits widely shared by many energy storage technologies, which make them reasonable to examine as a group. First, energy storage technologies have some limited-energy characteristic. While some storage technologies also convert energy from external sources, there is a substantial portion of energy which is taken directly from the grid to a storage medium and then delivered back to the grid on demand. The identity of the storage medium is relevant for these applications primarily in how it limits the power capacity, energy capacity, efficiency, and lifetime of the storage device. For this reason, energy storage units are treated interchangeably for most of this thesis. However, because of the differences in performance characteristics among storage units, these characteristics are briefly examined in this chapter.

Of course, no medium can store energy and return it when needed completely without losses. The efficiency of storage units varies from about 60% round trip to the high 90% range, depending on the technology and application. There are several major regimes for losses in energy storage. Some loss is usually present in the transfer of energy from electrical potential to the storage medium and back, including frictional and thermodynamic losses

in mechanical systems, and resistive and reaction losses in chemical systems. Other losses are associated with electrical power conversion and delivery, such as losses in an inverter or rectifier or in grid-interfacing transformers and lines. Energy storage units themselves often exhibit some form of self-discharge, such as chemical migration in a battery or electrochemical capacitor. Additionally, many systems include auxiliary equipment that must be run during charge, discharge, or standby. Such auxiliaries include fans and cooling equipment, circulating or vacuum pumps, active components such as magnetic bearings, and monitoring and control equipment. Most energy storage technologies experience considerable losses in at least two of these regimes.

The rate of change of power output or intake of which most energy storage technologies are capable compares favorably with that of most thermal generators. Even hydropower, which shares many characteristics with traditional thermal generation, is widely considered to be fast compared to most thermal generation (see also Section 3.1). Other technologies may be controlled even faster, in some cases being limited in their rate of change of power output primarily by the design of the power conversion equipment rather than by the storage technology [27].

There are many energy storage technologies which are either currently in use in grid-connected applications or which have been studied for such uses. Hydropower, including hydraulic installations with pumped storage capacity, are in widespread use. Two compressed air energy storage (CAES) installations provide grid services, and other similar plants have been considered. Smaller additions of batteries and flywheels have been or are grid-connected in various locations. Other storage technologies have seen some limited grid integration as well. The following sections describe some of the relevant characteristics of each of these technologies for frequency regulation. Table 4.1 includes some estimates of the efficiency, costs, and cycle life of several storage technologies (see also [83]).

4.2 Hydropower

Hydropower includes many different types of hydraulically-powered units. While all fundamentally convert the gravitational potential of water to kinetic and then to electrical energy, some types of units resemble energy storage more closely than others. Hydroturbines, which draw kinetic energy from moving water, may be installed between two reservoirs in a penstock, or in the natural stream of a river ("run-of-river").[1]

[1] While the extraction energy from tidal flows has been considered [21], this is not common and does not provide an energy-storage or limited-energy characteristic.

Table 4.1: Storage technology lifetime cost, efficiency and cycle life, reproduced from [92, Table 4]

Technology	Power Cost $/kW	Energy Cost $/kWh	Round-trip Efficiency % ac-ac	Cycles
Advanced lead-acid	400	330	80	2 000
Sodium-sulfur	350	350	75	3 000
Carbon-enhanced lead-acid	400	330	75	20 000
Zinc-bromide	400	400	70	3 000
Vanadium redox	400	600	65	5 000
Lithium-ion	400	600	85	4 000
CAES	700	5	70	25 000
Pumped hydro	1 200	75	85	25 000
Flywheels	600	1 600	95	25 000
Electrochemical capacitors	500	10 000	95	25 000

The hydraulic turbines which are most similar to energy storage are pumped storage units without a river inflow to the reservoir. These large-capacity units essentially form a pure energy storage system. There are also reservoir hydraulic units which primarily use a river inflow to the upper reservoir, but which also have pumping capability. These are limited energy resources in that the total energy output is limited, but the energy may be delivered at flexible times based on the power rating of the units. The pumping capacity of hydraulic plants is also similar to energy storage in its similarity to a negative load while charging. Hydraulic plants which consist of high and low reservoirs fed by a river inflow which do not have pumping capability are also considered limited-energy resources, for although they never behave as negative loads, they can flexibly deliver a limited amount of energy for short times according to the turbine power ratings. Run-of-river plants and reservoir-based tidal plants do not generally have this characteristic, as they are constrained to run when water is flowing, and any curtailed power is lost.

Because of the large volumes of water involved and the importance of water flows to the human and natural activity of the surrounding area, strict limits are frequently placed on the water flows through hydroturbines, whether pumping capability is available or not. The water which is used for power generation must also be managed suitably for municipal and irrigation water supplies, for fish and aquatic ecosystems, for flooding prevention, and for recreation or navigation. Some limits on water flows are seasonal, maintaining reservoir levels throughout the watershed; some are daily or weekly discharge limits which allow the unit operator flexibility to deliver power when most valuable over the course of a day or week, but not to perform arbitrage over longer times; and some flow and rate of change of flow limits are so severe as to require operation of a hydroturbine as a baseload generator [11].

While hydropower storage shares some of the characteristics of other energy storage technologies, it also has some important differences. First, because of the large volumes of water which can be stored in the reservoirs, the energy storage capacity of hydraulic storage is much larger compared to its power rating than that of other storage technologies. That is, if left to run from full to empty state of charge at full power, it takes a much longer discharge time than most other storage technologies, with discharge times of hours or days. This makes it economically feasible to provide energy arbitrage with pumped hydro plants. Another important difference is that hydroturbines, while fast compared to steam turbines, are still subject to slower inertial effects and mechanical time constants, making them much slower to respond than most other storage technologies, many of which are electronically interfaced [112]. For example, the system operators in Metlakatla, Alaska found that the available hydroturbines "lack the speed of response required to follow the loading fluctuations" from a large and quickly changing sawmill load [73]. (See Section 3.1 for additional discussion of the capabilities of hydraulic turbines.)

There are two main reasons that the amount of new hydraulic capacity that can be installed is limited, particularly in areas like the US and Europe where energy extraction is heavily developed. First and most importantly, the best sites for hydropower have already been developed, and what remains are more marginal sites. Second, the environmental impact of building dams for reservoir-based hydropower is usually substantial and permitting requirements remain extensive. There will continue to be some limited ongoing hydropower development consisting of adding power conversion equipment to existing unpowered dams; of refitting existing hydropower installations for larger power or pumping capacity; and of occasional new hydropower projects. While hydropower plays an important role in frequency regulation, its capacity to contribute to regulation is not likely to grow in the near future [14].

4.3 Compressed Air Energy Storage

Another energy storage technology with a large energy capacity and that is largely based on traditional generation technology is compressed air energy storage (CAES). This technique uses a vessel, usually an underground cavern, to store compressed air for use with a combustion turbine. Half or more of the mechanical power output during normal turbine operation is required to operate the compressor which feeds compressed air into the turbine. In CAES, the compressor runs on off-peak electric power to charge the reservoir with compressed air, which can be fed back into the turbine with the compressor off to extract energy from the reservoir. The additional mechanical power produced by the turbine and not being drawn by the compressor is converted by the generator. For CAES plants using

underground caverns as reservoirs, a large amount of energy may be stored, resulting in a discharge time of tens of hours at peak power. This makes the plants compatible with peak-shaving or energy arbitrage [27].

There are two CAES plants currently in operation. The first was a 290 MW plant in Huntorf, Germany which began operation in 1978. It is used for peak shaving, spinning reserve, and VAr support. Round trip efficiency for the stored energy is approximately 85%, although fuel is required for energy extraction. The plant has a four-hour energy capacity at full power, and the salt cavern which is used to store the compressed air has demonstrated very low air leakage and good stability [16]. The second operating CAES plant is a 110 MW installation in McIntosh, Alabama completed in 1991, with 26 hours of storage. Estimated round-trip efficiency is 75%, and the plant has been used for peak shaving, energy arbitrage, and spinning reserve [27].

CAES is different from other energy storage technologies in that, for existing systems, the delivery of stored energy is only possible when the combustion turbine is operating. Pneumatic motor-generator sets which charge and discharge a reservoir without an attached combustion turbine have been investigated, but are still in the technology development stages [49]. This restriction of running the gas turbine does not affect its use for peak-shaving, although it does place limits on economic use for frequency regulation.

CAES plants use a combustion turbine for energy conversion, which means that the limits on changes in generator power are similar for CAES and for combustion turbines. However, because the load of the compressor is independent of the turbine output power, the compressor power can be changed relatively rapidly, providing some faster control. This improves the response speed of CAES plants, although their rate of change of power output is still slower than that of electronically-interfaced energy storage technologies [27]. Some researchers have considered hybridizing CAES plants with faster energy storage (such as electrochemical capacitors) to provide a faster power response for the collective plant [66].

The least expensive way to store air for CAES is in geological features. Suitable features include salt caverns, former limestone mines, and aquifers in porous rock. Salt caverns in particular have a long history of use to store compressed natural gas (as fuel) and the technology to build and use them is mature. While the requirement for geological features does limit the siting of potential CAES plants, favorable geologies are common and occur over most of the US [27].

Table 4.2: Battery efficiency and cycle life, compiled from [27] and [44] (lithium ion only).

Chemistry	Est. Efficiency (% dc-dc)	Approx. Cycle Life (100% DOD)
Lead-acid	75–85	10^2
Sodium-sulfur	85–90	10^3
Vanadium redox	80	10^4
Nickel	60–85	$1 \cdot 10^3+$
Zinc Bromide	70–75	10^3
Polysulfide Bromide	65–70	N/A
Lithium Ion	80–95	10^5

4.4 Batteries

Batteries are the next most commonly used grid-connected energy storage technology after pumped hydro. A battery can be made out of many different chemistries, and battery varieties all have different characteristics. A battery can be a sealed system or one where the electrolyte (and the associated energy capacity) can be pumped through the cells and stored separately. Some batteries run at room temperature and some must run at elevated temperatures. For many batteries, the total energy that can be removed depends on the discharge rate, and such batteries are rated for more energy when longer discharge times are allowed. Some estimates for dc-dc efficiency and cycle life for several battery chemistries are listed in Table 4.2.

4.4.1 Lead-Acid

Lead-acid batteries are one of the oldest and the most common. This also makes them relatively inexpensive, as does the fact that lead is a relatively common and inexpensive metal. Lead-acid batteries are, however, more limited in their ability to cycle than some alternative chemistries. [27, Chapter 6] [107, 15].

There have been a number of grid-connected lead-acid battery installations over the years, many of which have provided frequency regulation, sometimes among other services. In 1986, West Berlin installed a lead-acid battery system to provide regulation and synchronized reserve for the city's small, islanded grid. The system provided 17 MW and 14 MWh, and it worked as planned until 1993 when the city's electric grid was reconnected to that of the rest of (reunified) Germany. Following reunification, the batteries continued to provide spinning reserve until 1995 [15, 59, 107, 91].

Another small island system which used lead-acid batteries for regulation (as well as synchronized reserve) was Puerto Rico, which installed a 21 MW, 14 MWh system in 1993. The system reduced the need for new generation capacity and improved reliability [104, 79].

Metlakatla, Alaska is a town with an islanded electric grid and sufficient capacity of hydrogeneration to cover the town's load. The local utility also owns a diesel generator which is expensive to run, particularly because of the high cost of transporting fuel to the remote area. The diesel generator was frequently operated primarily to provide spinning reserve and frequency regulation because the hydraulic turbines could not respond quickly enough to a large and variable sawmill load. The lead-acid battery system installed in 1997 has a 1 MW continuous power rating and has an energy capacity of about 1.3 MWh [73, 79].

At least two multipurpose lead-acid battery demonstration projects have also been installed, and both performed frequency regulation among other tasks. In 1986 a 1 MW, 4 MWh facility was installed by the Kansai Power Company in Tatsurni, Japan. In 1988 a 10 MW, 40 MWh system was installed by Southern California Edison in Chino, CA [79].

Design studies for frequency regulation using lead-acid battery storage systemsi include [57]. In addition to applications which provide frequency regulation, lead-acid batteries have been used for many other grid-connected projects as well. Common applications are peak shaving to reduce demand charges, providing load levelling for isolated systems, providing backup power, and performing voltage regulation [107, 79, 15].

4.4.2 Flow Batteries

Most of the common battery types are designed as modules of self-contained cells. While sometimes electrolye is agitated or otherwise conditioned, or water is added to replace that which is lost to off-gassing, the electrolyte remains within the cell. There is a class of batteries known as flow batteries, however, for which the electrolyte is cycled through the cells and is stored in tanks. The battery energy is in large part stored in the charged electrolyte, meaning that the energy capacity of the battery is decoupled from its power rating, and is limited primarily by the electrolyte tank size. This is an advantage for applications requiring longer-term storage.

There are several chemistries which are used to make flow batteries, including zinc bromide, polysulfide bromide, and vanadium redox. The power capacity of the batteries is determined by the cell size and construction, and the energy capacity is determined by the volume of electrolyte available. The electrolyte is pumped through the cells and the storage tank during charging or discharging. This provides efficient cooling to the cells, but the power

required by the pumps does reduce the system efficiency somewhat. Self-discharge also arises from chemical migration in the cells, with self-discharge losses in the range of 1% energy per hour typical. In standby, the pumps can often be shut off to reduce power demand, and cells can sometimes be drained to reduce self-discharge. The lifetime for many cells is generally limited by calendar life rather than by cycle life, making these batteries well-suited to heavy cycling duty [27, Chapters 9–11].

A number of test and demonstration systems of flow batteries of all three chemistries have been installed, primarily for applications such as peak shaving, equipment upgrade deferral, and uninterruptible power supplies [27]. A design study for frequency regulation using vanadium redox flow batteries is reported in [90].

4.4.3 Other Chemistries

There are other battery chemistries which have been used for grid-interfaced energy storage, and more which are currently under development. Nickel batteries are an older technology which have occasionally been considered for grid integration [27, Chapter 7].

The Golden Valley Electric Association (GVEA) is a utility cooperative near Fairbanks, Alaska. The utility has only a weak tie to the south and expensive local coal-fired and oil-fired generation. In order to minimize costs, GVEA operates with low reserves and has historically used load-shedding to meet reserve requirements. In 2003, GVEA installed a nickel-cadmium battery system with a 27 MW, 6.7 MWh capacity to provide primarily synchronized reserve. The system also has other capabilities, including frequency regulation [87] [27, Chapter 7].

Lithium batteries have tended to be more expensive than other battery types, but development continues and some newer lithium battery technologies have been used for grid integration [106].

Sodium-sulfur batteries are a high-temperature battery technology which have been integrated into the Japanese electric grid since the 1980s. This commercial technology typically displays a duration of about 8 hours, with a pulsed over-rating power typically allowed for a few minutes. DC-DC efficiency of 85–90% is typical, with standby losses of about 1% of capacity per hundred hours, largely due to heat losses. A number of commercial and test installations have been installed in Japan and elsewhere, although these primarily have been used for load leveling and uninterruptable power supplies [27, Chapter 8].

Other battery chemistries under development include liquid metal batteries [8, 9]. These are intended be inexpensive and to offer a large energy capacity compared to other storage technologies, and so may be more suitable for bulk energy applications than for frequency regulation.

4.5 Flywheels

The flywheel is a mechanical energy storage technology which has been used for stabilizing spinning motion since long before its use in electric grids. For grid-interfaced applications, the spinning mass is connected to an electric machine to alternately draw power from the grid and re-inject it. Generally grid-connected flywheels are designed with variable speed drives and electronic interfaces, and the response speed to changes in power commands is very fast. Because the storage medium (the flywheel itself) and the power equipment (the electric machine) are designed separately, the power and energy capacities of a system are flexible with respect to one another [27].

Flywheels have been used for frequency regulation at a 20 MW plant in Stephenstown, NY [31] as well as smaller test installations [71]. An application with similar cycling behavior is part of the New York subway system, for managing the acceleration and regenerative braking energy of trains. Other common flywheel applications include UPS systems, power quality, and bridging power [27].

4.6 Other Technologies

Other technologies have been suggested or attempted for use in grid-connected applications, and as new energy storage technologies are developed some of these will also prove suitable.

One such energy storage technology which has not found widespread application is magnetic energy storage. In magnetic energy storage, a current-carrying coil with high inductance is used to store energy, which can then be quickly delivered to and from the grid by controlled inverter/rectifier sets. This leads to a power rating which can be extremely high, limited primarily by the product of the voltage and current ratings of the coil. A superconducting coil is usually suggested in order to reduce the standby losses of the storage. The primary losses associated with a superconducting magnetic energy storage (SMES) unit are refrigeration power, loss in the power electronics which must be in the superconducting loop, and conversion losses. For a superconducting coil, the energy which may be stored is generally

limited by the current rating of the coil and by the electromagnetic forces produced in and around the coil. Normal loss magnetic energy storage (NLMES) has been suggested, but is not widely considered a valuable energy storage option because of the continual resistive losses in the coil.

Design studies for SMES systems for grid-connected applications include a 5.5 GWh reference design by Los Alamos National Laboratory [41] and a 1.2 GWh design [3]. Authors who have posited the use of SMES for frequency regulation applications include [75, 95]. More commonly, SMES has been used in applications where a large amount of power is required over a few seconds, such as stability and bridging power, with storage units capable of a few megawatts for a few seconds [27].

Another technology which stores electrical energy is capacitors, where the energy is stored in the electric field resulting from charge separation. Electrochemical capacitors,[2] which use the same capacitance which is created by a battery, are most commonly suggested for grid-connected energy storage because of their potentially high capacitance, on the order of kilofarads. Like SMES, capacitors are capable of very rapid discharge with power rating limited primarily by heating from equivalent series resistance, but the amount of energy available is quite small, about a few kilowatt-hours or less. This high power and low energy characteristic has generally been applied for bridging power, stabilization, and to absorb large power transients [27].

A technique that is not exactly a storage technology but shares important characteristics is the use of demand response for frequency regulation. In this technique, intermittent loads which require a given amount energy over a longer period but which can be operated flexibly within a time window are controlled to draw power at such times as to provide frequency regulation. Rather than being limited-energy resources, these are limited-load resources. This technique is most easily implemented with large loads like water system pumps, since they need to be individually controlled in order to respond to the frequency regulation signal [28]. A similar situation arises when storage is available in other end use loads, such as thermal storage for building heating and cooling systems.

[2] Trade names for electrochemical capacitors include "supercapacitors," "ultracapacitors," and "hypercapacitors."

Chapter 5

Dividing Regulation Burden and Regulation Dispatch

5.1 Introduction

LOAD frequency control (LFC, also called frequency regulation), is a task that may require both rapidly fluctuating power and substantial amounts of energy delivered or sunk over minutes to hours.

The approach described in this work is to split the LFC signal between two dissimilar sets of assets: nimble but lower-capacity energy storage units, and slower traditional thermal generators. The goal is to decrease the fuel and maintenance requirements of the thermal generators and to enable the integration of variable generation resources that can increase LFC requirements [81]. Further, this approach may enable better provision of LFC through the use of storage to track fast fluctuations without increased cost. This chapter discusses a broad strategy for incorporating energy storage in the dispatch of LFC capacity. Chapter 3 discusses the relevant aspects of the thermal generation in more detail, and Chapter 4 investigates the characteristics of some appropriate storage technologies.

The approach taken here is to assume that thermal power plants can follow a signal precisely and accurately at some expense, and to examine changes in that signal in order to reduce the expense. The existing literature on the expense associated with fast changes of thermal plant output power level is examined in Chapter 3. The two classes of costs from frequency regulation are expected to be associated with fast ramping of thermal plants and with total capacity on regulation duty.

This approach sidesteps questions of the value of a given degree of control performance and how that may be measured. As discussed in Chapters 2 and 6, the two data sets available for this work are slightly different in that data set A is the total delivered power for the control area and data set B is the control signal for the generators performing frequency regulation. For the analysis in this chapter, only data set A is used to demonstrate the techniques described. Because data set A consists of the actual delivered power in the control area, it

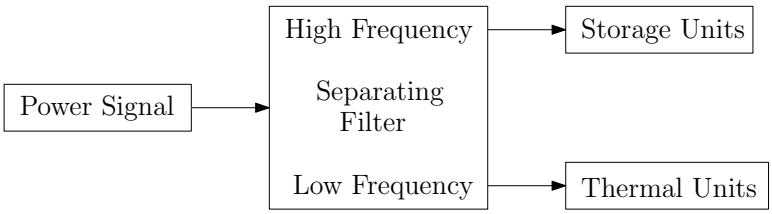

Figure 5.1: Block diagram of scheme to partition load frequency control signal between thermal generators and energy storage units.

is assumed both that the control performance is adequate and that the existing traditional generation resources are capable of following this signal. For a comprehensive look at the performance of data set B under the techniques described in this chapter, see Chapter 6.

The alternative to making assumptions about the completeness and correctness of the system response to regulation is to model the available generation and the expected loads and evaluate the frequency performance. This may present a more complete argument but requires extensive knowledge of the physical system of interest to be credible, and also would encourage comparison and evaluation of control performance requirements to determine what sort of regulation is good enough in a particular context. By contrast, the strategy of focusing on limiting costly operations is more flexible among systems and does not require the same amount of system data to produce conclusions.

Note that the language of LFC generally refers to positive and negative regulation power for all assets, not only for energy storage devices. This positive or negative power is the offset from a unit operating point. For an energy storage unit, this operating point is approximately zero (or slightly negative, to compensate for losses). For a traditional fossil-fueled power plant, this setpoint might be 0.9 per unit, and regulating power might be 0.1 per unit, so that regulating at -0.1 per unit corresponds to 0.8 per unit power overall, and regulating at +0.1 per unit corresponds to 1.0 per unit power overall. [56]

The main approach taken in this work is to divide the burden of LFC between fast energy storage units and slower traditional thermal generators [43]. In this way, the energy storage can assume the fastest-cycling portion of the required LFC and allow thermal generators to be operated at steadier conditions. This approach is illustrated in Fig. 5.1. Each portion of the system operates at a preferred point, with nimble energy storage closely following moment-to-moment fluctuations and thermal generation providing slower bulk energy. The composite system could be referred to as a "virtual power plant" because the storage unit and the thermal unit operate in concert to deliver the total requested power, although they need not be co-located. Performance of frequency regulation under the current system, however, does depend on a data connection to the system operator, making it less likely that many small distributed units would be an economical way to provide frequency regulation.

This chapter describes some possible strategies for design and operation of a composite system consisting of a fast, small-capacity energy storage unit and a large bulk thermal unit. While the benefits of this strategy can be seen without a dynamic model for a thermal power plant as is done in this chapter, if thermal unit model were replaced with a more realistic, slower-acting unit, the fidelity with which the combined system follows the control signal would become more evident. [68]

5.2 Simple Filtering

To illustrate the utility of the energy- and ramp-rate-duration curves introduced in Chapter 2, consider the task of partitioning the load power signal of Fig. 2.8 between fast-acting, limited-energy storage units and slower, limited-power thermal generators. The goal of the partition is to limit the total required energy storage and the maximum and average ramp rate of the thermal units while adequately responding to the entire signal [65]. The ultimate aim is to produce a method to partition the power requirement in real-time. Hence, only causal candidate filters are investigated. Filters like the non-causal (predictive) sliding-window filter of Section 2.3 may not be used for this task. Note that the predicted load (from historical and weather data) may still be omitted or subtracted out from a power signal, as was done in Section 2.3, because the mechanisms for bulk energy production and load following are separate and not considered here.

To demonstrate the use of this technique, the effects of a class of simple open-loop separating filters will be explored. All filters of interest seek to partition the signal between the thermal and energy storage assets to more effectively take advantage of the strengths of each unit type. One possible separating filter type is a high pass filter and its complement, as illustrated in Fig. 5.2. The Chebyshev type I high-pass filter [80, Section 7.2] was selected for its good attenuation of low frequencies and its fast transition band. A filter of order 3 was found to offer a good compromise between fast roll-off and the increased delay produced by additional poles.

A range of cutoff frequencies was selected corresponding to periods from 3 minutes to 60 minutes. The frequency responses of the filters which are evaluated here are pictured in Fig. 5.3. A built-in bilinear transform in MATLAB was used to convert the analog Chebyshev prototype filters to digital IIR filters [80, Section 7.3]. It can be seen in the figures that follow that filters with these cutoff frequencies cover the range of interesting filter behavior in this case.

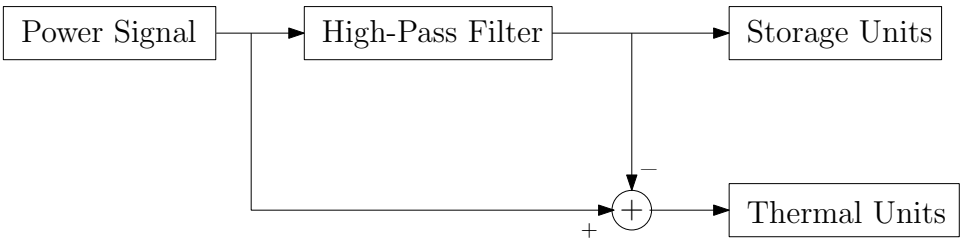

Figure 5.2: Block diagram of one type of separating filter (as introduced in Fig. 5.1) to partition load frequency control signal between thermal generators and energy storage units.

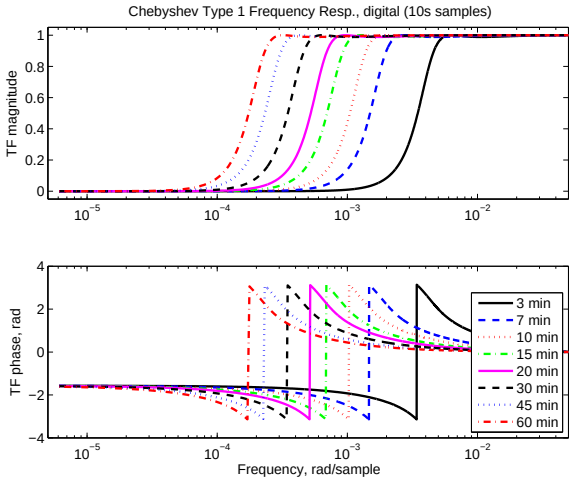

Figure 5.3: Frequency characteristics of Chebyshev type I high-pass filters [80, Section 7.2] evaluated in this section. All filters have order 3.

Data set A set from Section 2.3 was filtered in this manner, to divide the high frequency portion of the signal—suitable for an energy storage device—from the low frequency portion of the signal—suitable for thermal units. The time series of the low and high frequency portions of data set A for one of the filters is shown in Fig. 5.4, although, as discussed in Chapter 2, it is difficult to interpret such a plot. A portion of the high and low frequency components is also shown in Fig. 5.5, to show the detail of the signals. The plots do make clear that the high frequency component of the signal consists primarily of fast, even fluctuations while the low frequency component of the signal includes larger, slow positive and negative excursions.

As covered in Section 2.4, an easier way to examine the power distributions of the separate portions of the data signals is to use so-called power duration curves to display the power

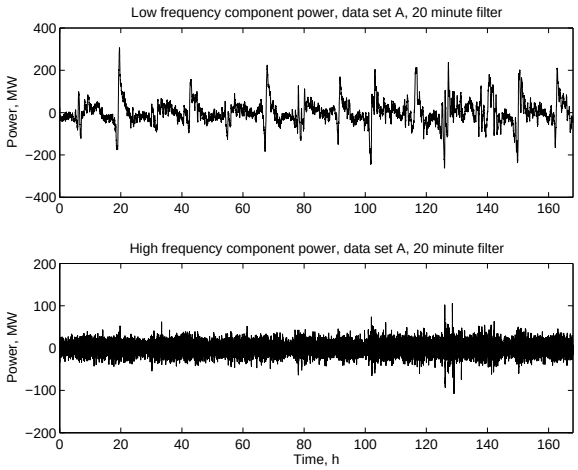

Figure 5.4: Example of high frequency and low frequency portions of data set A, separated using Chebyshev type 1 high pass filter with 20 minute cutoff.

requirements of both signal portions. The power duration curves for the high frequency portions of data set A, suitable for an energy storage unit, are shown in Fig. 5.6. The corresponding power duration curves for the low frequency portions of the data, suitable for a slow-moving thermal power plant, are included as Fig. 5.7. The storage unit does little to alter the overall power requirement on the thermal units, probably because this power requirement is dominated by slow fluctuations that entail large amounts of energy.

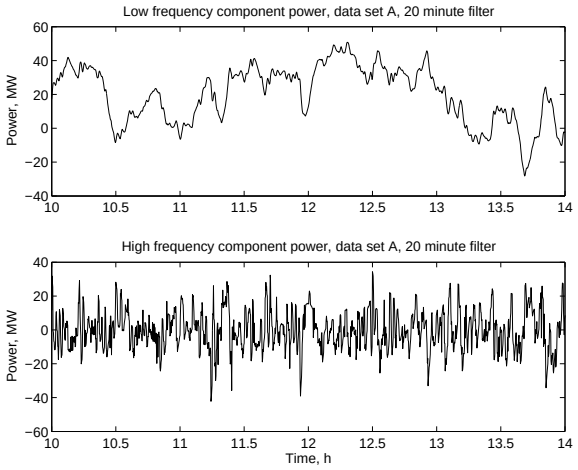

Figure 5.5: A portion of high frequency and low frequency components of data set A, separated using Chebyshev type 1 high pass filter with 20 minute cutoff. Similar to Fig. 5.4 but zoomed in to show detail of signals.

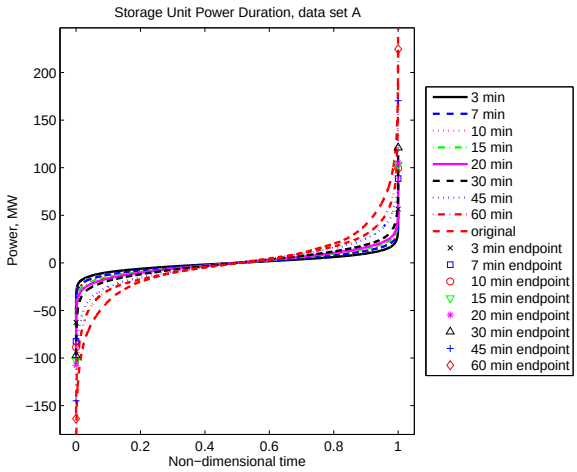

Figure 5.6: Power-duration curve of high frequency portion of data set A using Chebyshev high-pass filters of Fig. 5.3. Corresponds to Fig. 5.4 (bottom).

The benefit to the system from this type of filtering comes from the reduction of ramping required of the thermal units. This reduction is most clear from a ramp rate duration curve as introduced in Section 2.5. The ramp rate duration curve of the low frequency portions of data set A are included as Fig. 5.8. From this plot the reduction in thermal unit ramping as a result of the filtering becomes apparent. This ramping is instead performed by the energy storage units on the system, for which the ramp rate duration curve is included as Fig. 5.9.

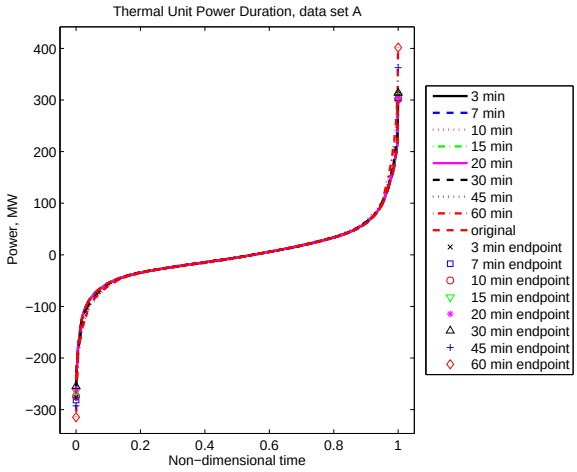

Figure 5.7: Power-duration curve of low frequency portion of load power signal using Chebyshev high-pass filters of Fig. 5.3. Corresponds to Fig. 5.4 (top).

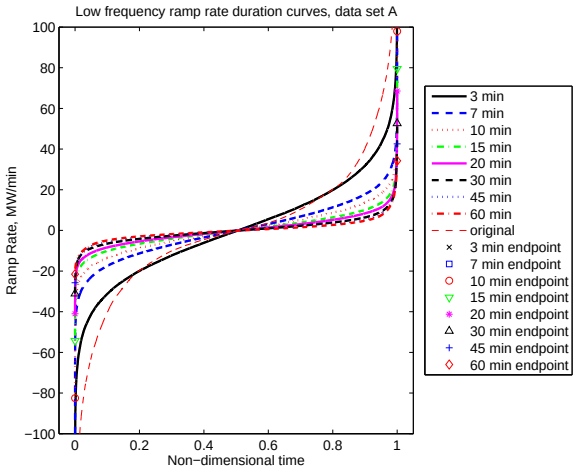

Figure 5.8: Ramp-rate-duration curve of low frequency portion of data set A, obtained by subtracting from the original signal the output of Chebyshev high-pass filters of Fig. 5.3. By comparing the filtered and the raw data it may be seen that both the maximum and the mean ramp rates required of the thermal units have been substantially reduced.

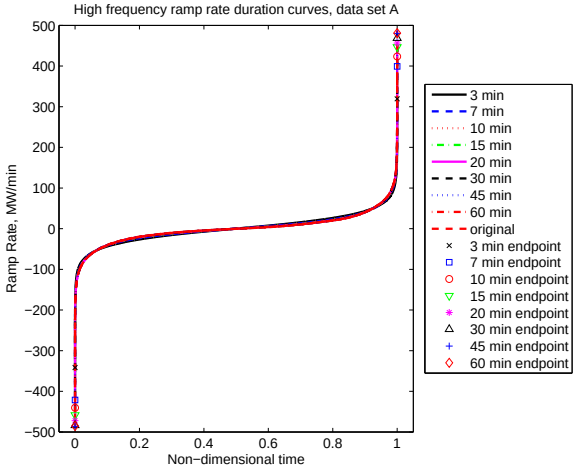

Figure 5.9: Ramp-rate-duration curve of high frequency portion of data set A, obtained by subtracting from the original signal the output of Chebyshev high-pass filters of Fig. 5.3. The storage unit is performing most of the ramping required of the total system.

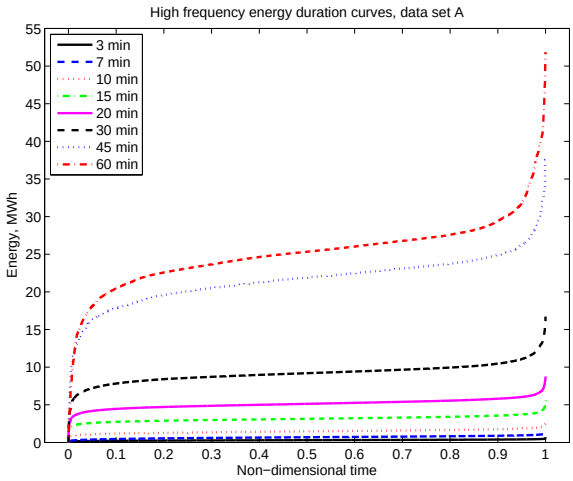

Figure 5.10: Energy duration curve for high frequency portion of filtered data set A, obtained by subtracting from the original signal the output of Chebyshev high-pass filters of Fig. 5.3.

The amount of energy storage which is required in order to achieve this reduction in ramping is a key consideration, and may be readily evaluated by the use of the energy duration curves introduced in Section 2.6. Energy duration curves for data set A are included as Fig. 5.10. Note that the values of the energy delivered depend on the initial condition of the storage unit, and it is the relative change in energy over the whole data set which indicates the fundamental behavior of the system, and so for these plots the energy levels have been shifted so that the lowest energy is placed at zero.

The filters require total energy storage of a moderate size. It is also of note that in all cases, the storage units may be spending the majority of the time at a middle state of charge with only occasional excursions to a much lower or much higher state of charge. This may not represent the best use of the storage resources depending on storage type, since it means that a large amount of energy capacity is being purchased but rarely used. However, this could also be an indication for a system in closed-loop feedback, possibly alongside a more sophisticated filtering algorithm. Such a closed-loop system could be designed to eliminate those rare occasions when the storage unit must either fill completely or empty completely, and so to obtain a similar improvement in thermal unit ramping from a smaller storage unit.

Another way to evaluate dispatch strategies directly in one graph is to compare a measure of energy storage with a measure of total ramping. For energy storage, one good metric is the total energy required, which is closely related to system cost. This is the maximum net energy required from the storage unit over the period covered by the data sets (which is the maximum point for each curve in Fig. 5.10). Several metrics could be used for system

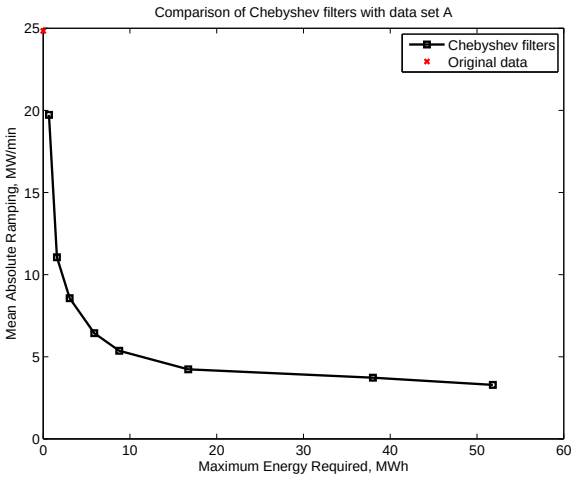

Figure 5.11: Comparison of Chebyshev filters of Fig. 5.3 according to maximum energy storage requirement and mean absolute ramp rate for data set A.

ramping. A plant-specific metric could compare the total cost of plant ramping at different rates. While this could be valuable for evaluating the effects on a particular unit or for a group of similar plants, the limited available data on cost of ramping and the variety of units performing frequency regulation makes such a metric impractical here. One simple metric which captures the overall ramping characteristic is mean absolute ramp rate, which considers ramping up and ramping down to be equivalent, and considers ramping at 2 MW/min to be twice as harmful as ramping at 1 MW/min, and so on.

Plotting the energy storage versus mean absolute ramp rate pairs for each filter gives a visual interpretation of the trade-off between energy storage cost and benefit to the thermal generators. This plot is presented as Fig. 5.11, with a comparison among the Chebyshev filters of Fig. 5.3. The ramping requirement of the raw (total) data is also plotted for reference. An alternative metric for ramping intensity is to use the root mean square (RMS) ramp rate for the low frequency portion of the signal. The RMS ramping emphasizes the impact of times with a high ramp rate, which might be more relevant for thermal power plants which can supply slow changes in output power with little penalty but are sensitive to larger ramp rates.[1] A plot similar to Fig. 5.11 which uses RMS ramp rate rather than mean absolute ramp rate as a metric is included as Fig. 5.12. While the ramping values are different for the two cases, the overall shape is the same.

[1]For a further discussion of ramping in thermal generators, see Chapter 3.

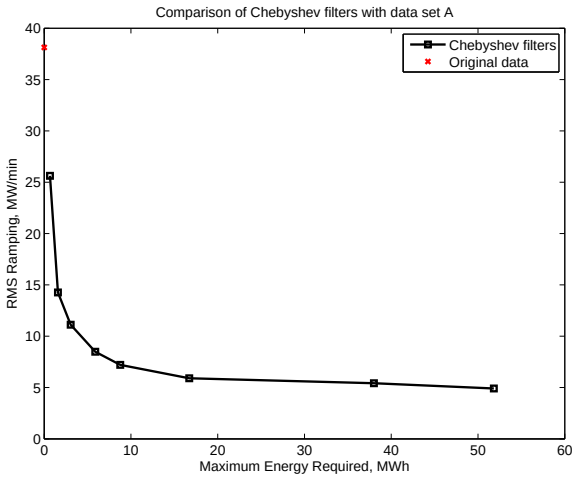

Figure 5.12: Comparison of Chebyshev filters of Fig. 5.3 according to maximum energy storage requirement and RMS ramp rate for data set A.

5.3 Closed Loop Filtering

A disadvantage to a straight feed-forward filter like that described in Section 5.2 is that it does not take into account any information from the energy storage device. This means that any small offset in the average power delivered, for example as a result of losses, could cause the energy storage device to be completely emptied or filled. In this case, the device would be unable to perform its assigned frequency regulation properly. A solution to this difficulty is to incorporate feedback of the energy storage unit's state of charge, and to adjust the power levels of the storage device in order to maintain the device at its preferred charge point. This closed loop control of the state of charge of the device protects the storage unit from emptying or filling completely, even in the presence of storage system losses, or if the separating filter does not supply a power signal to the storage system which is precisely zero average. A 50% preferred state of charge is used here, which gives the largest available bidirectional energy capacity, but this preferred point could also take into account the operational constraints of a particular storage technology, such as a battery state of charge which minimizes degradation. A preferred operating point could also be chosen to facilitate the use of an energy storage device for another purpose in addition to frequency regulation. As will be discussed in Subsection 5.3.2, the signal for the storage unit may also be adjusted to account for the expected losses.

The system takes as an input signal the total desired power for the control area. In the simulations reported in this Section, data set A introduced in Section 2.3 is used as an input of requested control power to evaluate the performance of the filtering with state of charge

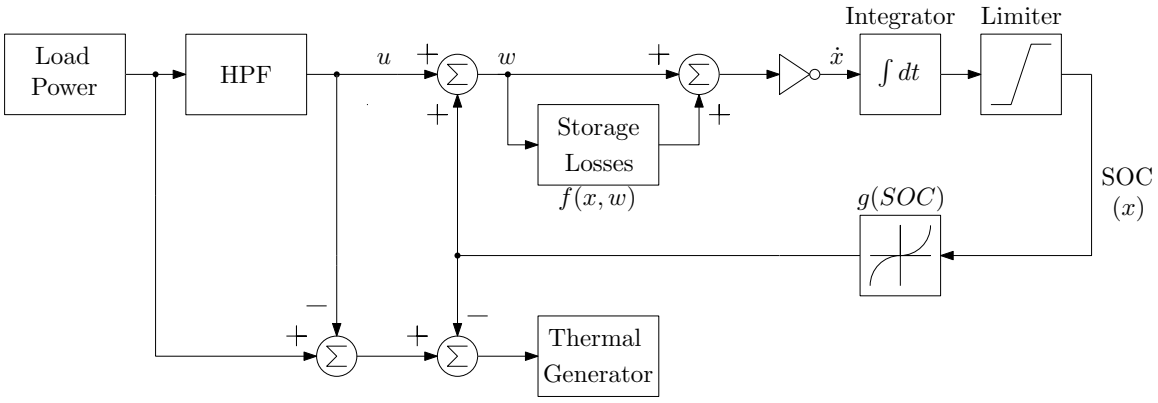

Figure 5.13: Block-diagram overview of the virtual power plant system.

feedback. This input to the closed-loop filtering system comes from the system operator response to the frequency and tieline power flow performance of the balancing area.

Using the same type of filters as in Section 5.2, the total requested power input is divided into a fraction (the high-frequency component) to be sent to the energy storage and a fraction (the low-frequency component) to be sent to the thermal generation. The overall scheme is pictured in Fig. 5.13. The main division between these portions of the signal is provided by the high-pass filter as before, but these signals are added to a nonlinear pure state feedback. The feedback function is discussed in Subsection 5.3.3. In addition to responding to the required regulation signals, the energy storage unit draws or injects a small amount of power to bring its state of charge back toward its preferred operating point. This balancing power is subtracted from the power delivered by the thermal power plant performing regulation so that the total power delivered is unaffected. The portions of power delivered by the energy storage unit and by a sufficiently fast thermal generator are complimentary, in that the system is designed so that they will always sum to the total power of the input signal.

This separation of the task of energy balancing in the storage unit and signal filtering in feed-forward has some advantages. First, the simple dynamics of the closed-loop system facilitate the demonstration of stability. Introducing additional dynamics into the storage feedback path would require more sophisticated techniques to demonstrate its stability (see Subsection 5.3.3). The incorporation of a high-pass filter into the feedback system would necessitate complex and careful design for stability. Further, the limit on the bandwidth of the signal which is being fed to the storage system allow the storage design to perform well as long as it handles the fast portion of the signal, and does not require tailored performance across the entire signal frequency band. This is related to the minor-loop control technique in classical feedback [67]. Finally, the use of the feed-forward filters enables easy comparison to the open-loop case.

The filters used for this separation are a subset of the filters used in Section 5.2. From Fig. 5.33 it can be seen that the Chebyshev 10 minute, 15 minute, 20 minute, and 30 minute filters (order 3, type I high-pass) capture much of the filtering space with good reduction in ramping provided by a small amount of energy storage, and these filters were used as the basis for simulations in this section. Just as in the previous discussion, this division of the power signal into low frequency and high frequency components could also be performed by other classes of filters.

The signal going to the thermal generator consists of the required input power minus the power provided by the energy storage unit. In the simulation, this is implemented by subtracting the output of the high-pass filter from the input power, then adjusting the signal by the fed-back balancing power.

5.3.1 Storage Modeling

The storage unit is modeled as an analog integrator with saturation. The state of charge is limited on a normalized scale between -1 and 1. The total size of the storage for each filter is set equal to 120% of the storage requirement found in the open-loop filtering cases for the same filter, illustrated in Fig. 5.10. The values are also listed in Table 5.1. As illustrated in Fig. 5.13, the adjusted power signal from the high-pass filter is further adjusted by the loss model of the storage unit, described in Subsection 5.3.2, before undergoing integration. The signal is also inverted to switch from power output to energy input. The normalized state of charge is finally fed back for the loss calculation and the balancing signal. This model only operates properly if the state of charge does not reach a completely full or completely empty state. If the state of charge reaches the rails, it generates a flag, and storage sizes were selected to avoid this case.

5.3.2 Storage Unit Loss Modeling

While a detailed loss model of a specific energy storage system is outside the scope of this work, a realistic loss model was desired for validation. Several loss mechanisms were modeled for the energy storage unit. A graphical representation of these losses may be found in Fig. 5.14. Uncertainty in the storage loss model was added by way of random factors which change periodically (every 15 seconds) over the course of the simulation and influence the total efficiency. This randomness is intended to account for unmodeled factors like system temperature or higher-order effects. The losses are modeled as one-way efficiency and a self-discharge power. The efficiency is the sum of a base loss fraction, a term proportional to the power level (resulting in loss proportional to the power flow squared), a term proportional

Table 5.1: Storage unit size and self-discharge power for closed-loop filtering simulations using data set A.

Filter Cutoff	Storage size (MWh)	Self-discharge power (kW)
10 minute	4	40
15 min	8	80
20 min	11	110
30 min	21	210

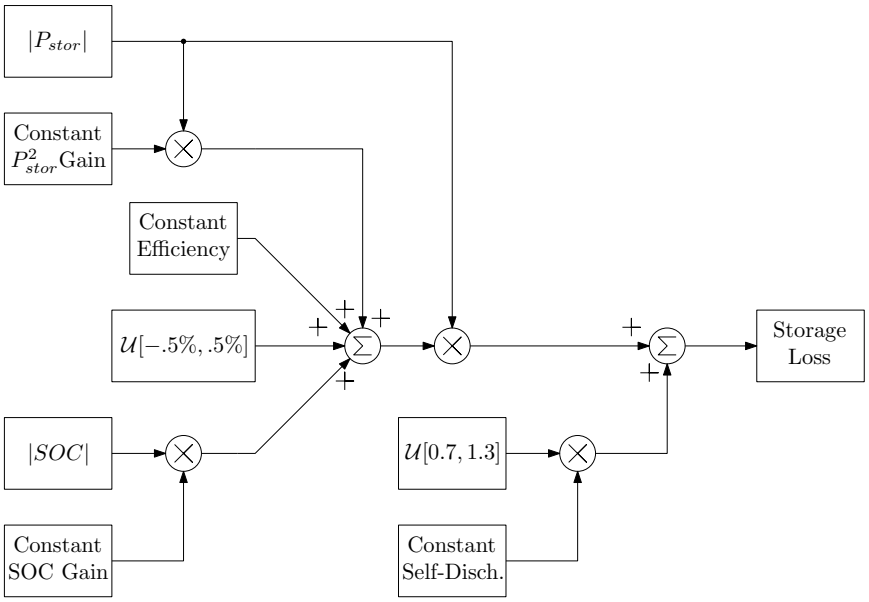

Figure 5.14: Block-diagram overview of the full model of storage losses.

to the absolute value of the state of charge, and a variable term with a uniform distribution. The calculated efficiency is then multiplied by the absolute value of the power flow to result in an efficiency-based loss. The self-discharge power is a fixed term dependent on the storage size equal to 10 kW/MWh, scaled by a variable fraction with a uniform distribution, as listed in Table 5.2. Because of this storage-size-dependent loss term, the overall efficiency in the smaller storage units is somewhat higher than in the larger storage units. The choice of loss constants was aimed at generating a ballpark round-trip efficiency of about 90%, which is near the high end of the efficiency ranges for several storage technologies [27]. For further discussion of storage technologies and efficiencies, see Chapter 4.

5.3.3 Feedback Design and Stability

The feedback was designed to force the energy storage back toward its half-full equilibrium position, where maximum bidirectional power and energy capability is available. When the

Loss type	Formula	Storage model value	Approximate model value
Base loss fraction	$k_p \cdot \lvert P \rvert$	2.6%	3.0%
Squared loss	$k_{sq} \cdot P^2$	$7.5 \cdot 10^{-4}$MW	$3.75 \cdot 10^{-4}$MW
SOC loss fraction	$k_{SOC} \cdot \lvert SOC \rvert \cdot \lvert P \rvert$	0.02	–
Variable loss fraction	$v_{eff} \cdot \lvert P \rvert$	$\mathcal{U}[-2\%, 2\%]$	–
Constant self-discharge	$k_{SD} \cdot W_{MAX}$	10 kW/MWh	10 kW/MWh
Variable self-discharge coefficient	$v_{SD} \cdot P_{SD}$	$\mathcal{U}[0.7, 1.3]$	1

Table 5.2: Parameters and descriptions for storage loss modeling and approximations for loss feed-forward.

system is close to the center of its capacity, the feedback gain is small and the system is only lightly pushed toward equilibrium. By contrast, when the system approaches full or empty, the feedback driving the system toward the center is much stronger. This general shape is available in many functional forms, but a simple cubic function of the normalized state of charge was used here, $g(x) = k_g \cdot x^3$, as shown in Fig. 5.15. The gain was adjusted so that stability would be ensured under all power conditions outside of a 30% to 70% state of charge band. Stability inside of that band is of less import, both because the feedback gain of the system is quite small in that interval and because that band represents a good operational range. For less extreme power demands, the guaranteed stability band extends closer to the 50% preferred state of charge.

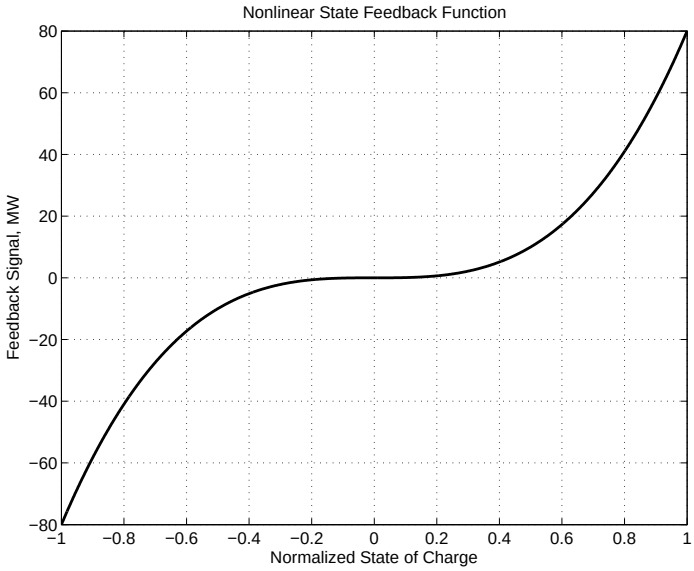

Figure 5.15: Plot of the nonlinear state feedback function used in these simulations, $g(x) = k_g \cdot x^3$, with $k_g = 80$ MW, where x is the normalized state of charge, on $[-1, 1]$.

Because a closed-loop control system has been introduced, the stability of the system is a concern. Instability in a storage feedback control loop would force the storage device to fill or empty completely and lose its ability to draw or inject power. Stability may be easily demonstrated in this system because it is first-order, with only one dynamic element (the integrator in the storage model). For any first-order system, stability is ensured if the derivative of the state variable is always of the correct sign to push the system back toward equilibrium. For first-order systems only, this is a sufficient condition for stability [98, p. 286]. For this system, we only wish to guarantee stability in the ranges of $SOC < 30\%$ or $SOC > 70\%$. When the state of charge x is normalized to $[-1, 1]$, $SOC < 30\%$ corresponds to $x \in [-1, -0.4]$ and $SOC > 70\%$ corresponds to $x \in [0.4, 1]$.

A state model of the system is helpful for formulating the question of system stability. The state model of the system may be represented as:

$$\begin{aligned} & \dot{x} = -u - g(x) - f(x, w) \qquad (5.1) \\ \text{where} \quad & x = \text{normalized state of charge, on } [-1, 1], \\ & u = \text{input power signal, MW}, \\ & w = u + g(x) \end{aligned}$$

That is, the rate of charge or discharge of the storage unit is equal to the opposite of its net power output, which is itself equal to the sum of the requested high-frequency power, the feedback power, and the loss power. The storage loss model is represented by $f(x, w)$, and the nonlinear feedback function is represented by $g(x)$. Stability of the feedback system requires that the feedback term have the opposite sign from x at all points of interest; i.e., $g(x)+f(x, w)$ must have the same sign as x. Note that $f(x, w)$ represents power losses in the storage device, and is therefore always positive. The feedback function $g(x) = k_g \cdot x^3$ always has the same sign as x. This means that for $x > 0$, stability is always ensured. For $x < 0$, the losses tend to cancel the feedback function, so stability requires that $|g(x)| > |f(x, w)|$, meaning that the feedback function dominates the losses. Hence, to demonstrate stability, we need to show that the feedback function is large enough in the region of interest, i.e., $x \in [-1, -0.4]$ and power request input at an allowed level.

As is presented in Subsection 5.3.2, the equation for the loss model is:

$$f(x, w) = k_{sq} \cdot w^2 + (k_p + k_{SOC} \cdot |x| + v_{eff}) \cdot |w| + P_{SD} \cdot v_{SD} \qquad (5.2)$$

In order to ensure that $g(x) + f(x, w) < 0$ for $x \in [-1, -0.4]$ and allowed values of drawn power w, $g(x)$ is separated into two parts, $h(x)$ and $m(x)$. Each part of $g(x)$ dominates one of the terms of the loss function, so that their sum will always dominate the complete loss function, as follows:

$$
\begin{aligned}
g(x) &= h(x) + m(x) && (5.3)\\
|h(x)| &> k_{SOC} \cdot |x| \cdot |w| && (5.4)\\
|m(x)| &> k_{sq} \cdot w^2 + (k_p + v_{eff}) \cdot |w| + P_{SD} \cdot v_{SD} && (5.5)\\
& && (5.6)
\end{aligned}
$$

The two component functions are chosen of the same form as $g(x)$, with $h(x) = k_{g1} \cdot x^3$ and $m(x) = k_{g2} \cdot x^3$, so that their sum is a function with the correct form. The minimum values of k_{g1} and k_{g2} may then be determined based on the requirements of the individual inequalities. The portion of $f(x, w)$ which is a function of x is linear in x, so the cubic $|h(x)|$ will be greater than the loss component over the whole interval if it is larger at $x = -0.4$. The remainder of the loss function is independent of x, so that again, the cubic $|m(x)|$ will be greater than the loss component if it is larger at $x = -0.4$. The numerical values for the loss coefficients are listed in Table 5.2. The loss function is monotonically increasing as the storage power $|w|$ increases, meaning that to ensure stability across the whole range, the inequalities must hold at the maximum power level. From Fig. 5.6, it may be seen that the required power from the energy storage unit is nearly always 50 MW or less in open loop operation, so this value is used for the maximum of $|w|$. The variable loss terms v_{eff} and v_{SD} are set to their maximum values, 2% and 1.3, respectively, and the largest self-discharge power is chosen.[2] From these, bounds on the two function coefficients may be calculated, as $k_{g1} > 6.25$ and $k_{g2} > 69.5$, yielding $k_g = k_{g1} + k_{g2} > 75.8$. Based on this result, the value $k_g = 80$ MW was selected, making the feedback function $g(x) = 80x^3$. This choice guarantees stability for $x \in [-1, -0.4]$ and $|w| \leq 50$ MW. If the power level is lower, the system will also be stable closer to the 50% state of charge point at $x = 0$. Similarly, as x decreases toward -1, stability is assured for an increasing power range, e.g. up to 190 MW at $x = -0.8$.

[2] A lower self-discharge power in fact has a small effect on the minimum feedback coefficient, reducing it in this case from 75.8 MW (for 210 kW self-discharge) to 72.3 MW (for 40 kW self-discharge).

5.3.4 Simulation Tools

A Simulink model was built of the system including the filter, the storage, the losses, and the thermal plant, as described in this document. The Simulink model was configured as an input-output block, with the outputs including most variables of interest. The simulation can then be run with any properly-configured load data set, and the power delivered by both the thermal unit and the storage unit can then be used for other calculations. The storage model, including the loss modeling, is not physically-based for any particular system, but is simply an implementation of the equations as described in this paper. The current "perfect" thermal generator model is a simple unity gain that delivers any requested power level. As discussed at the beginning of this Chapter, this corresponds to an assumption that the thermal generation operates quickly enough to adequately respond to the signal. The results of the filtering scheme can be compared to the original to determine the changes in operating cost between the two schemes. It is of note that the system displays a large but well-damped transient spike at the beginning of the simulations. This was eliminated from the calculations by allowing the system to initialize with a constant input equal to the first input power value for some time before the data stream begins. This spike is not believed to indicate a problem because this is a system which is always operated in steady-state and rarely turned on, and the turn-on behavior may be adjusted with specialized functions active only at start-up.

5.3.5 Results

Results are presented for the simulations using data set A introduced in Section 2.3. The load is assumed constant between data samples in a zero-order hold configuration. For nonconsecutive days, each day's simulation is initialized and run separately, and the results are only concatenated after all simulations have run. Figure 5.16 shows the total power and the power delivered by the thermal unit. The corresponding power out of the storage unit is plotted as Fig. 5.17, with the storage state of charge in Fig. 5.18.

The output of the simulation is evaluated with the use of duration curves as described in Chapter 2 and in the same manner as Section 5.2. The distribution of power delivered by the storage unit is in Fig. 5.19. Likewise the power distribution delivered by the thermal plant is included as Fig. 5.20. The decrease in thermal unit ramping which can be achieved as a result of this filtering is shown in Figs. 5.21 and 5.22, which also demonstrate that the majority of the ramping to follow the signal is being performed by the relatively small storage units. The statistics of the use of the storage units are shown as energy duration curves in Fig. 5.23. The control of the storage unit keeps it centered at the middle of its

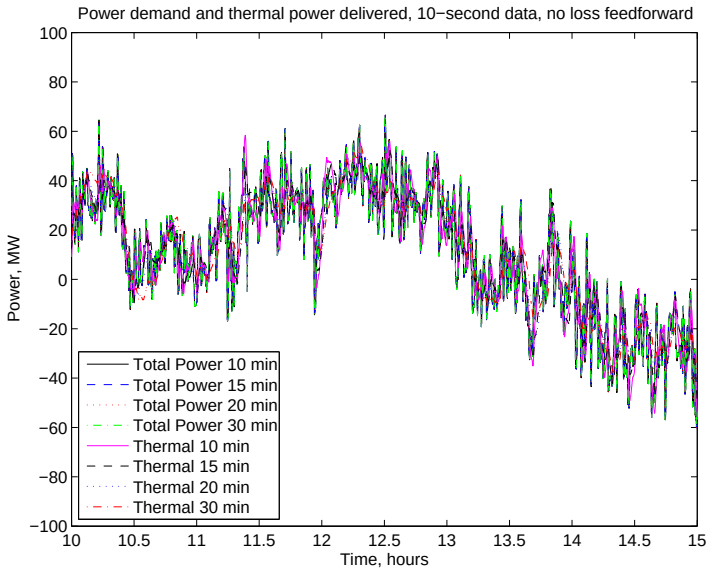

Figure 5.16: Time series plot of the power delivered by the thermal unit and the total power delivered by the virtual power plant for a portion of data set A. The total output is equal to the input signal, which is the power demand for a balancing area over seven nonconsecutive days (each day is processed separately).

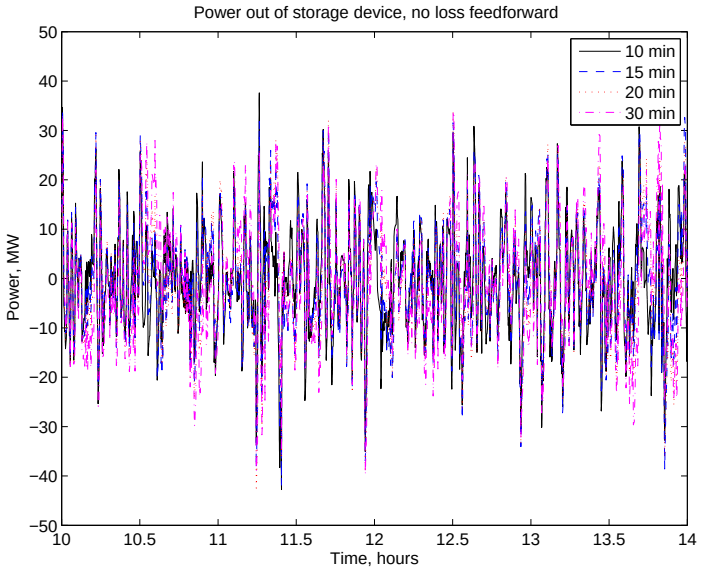

Figure 5.17: Time series plot of the power delivered by the energy storage unit over the simulation period for a portion of data set A.

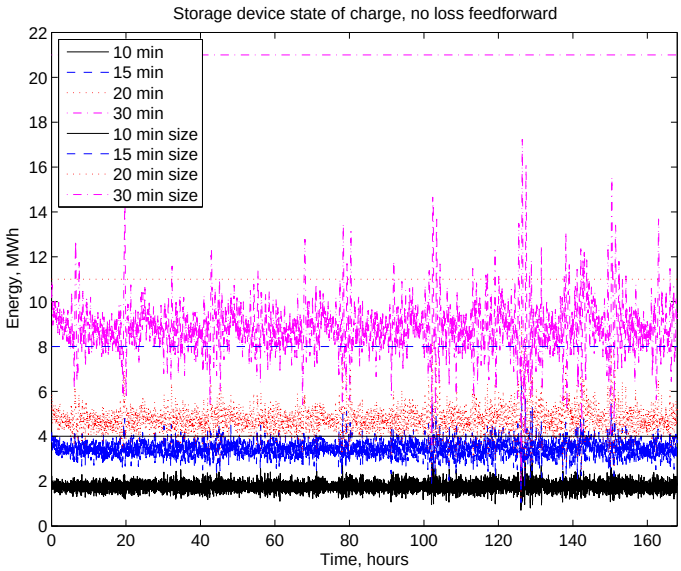

Figure 5.18: Time series plot of the storage state of charge over the simulation period for data set A. None of the storage units either fill completely or empty completely over the course of the simulation: see capacities in Table 5.1. In contrast to Fig. 5.10, these energy values represent simulated states of charge rather than net energy absorbed or delivered, and so have not been shifted vertically.

energy range, providing maximum energy and power capacity in both directions most of the time.

It is of note that the duration curves still have long "tails," where a large amount of power, energy, or ramping capacity is required for a very small fraction of the simulation period. This is likely to be most problematic in the storage power and energy capacity ratings, where the additional rarely-used capability can lead to unnecessary expense. This may also be a problem for some storage technologies and not for others, if frequent shallow cycles and occasional deep cycles in charge are allowed (see Chapter 4).

5.3.6 Adding Approximate Loss in Feed-Forward

As can be seen in Figs. 5.18, while the feedback does help the storage unit to remain in the middle of is charge range, it does tend to empty more quickly than it fills because of the losses in the storage model, which lead to an unbalanced system. One way to prevent this problem is by accounting for the losses in a feed-forward loss approximation. In this scheme, illustrated in Fig. 5.24, the expected losses in the storage device are computed based on the output of the high pass filter, and these are subtracted from the power signal

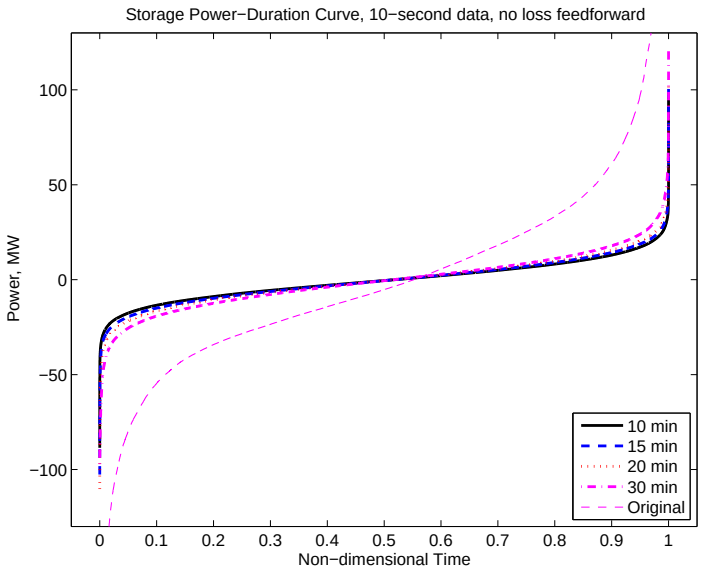

Figure 5.19: Power duration curve of the output of the energy storage unit for data set A, compared to the total system power. This graph represents the fraction of time that the storage unit is delivering an amount of power less than or equal to the curve amount.

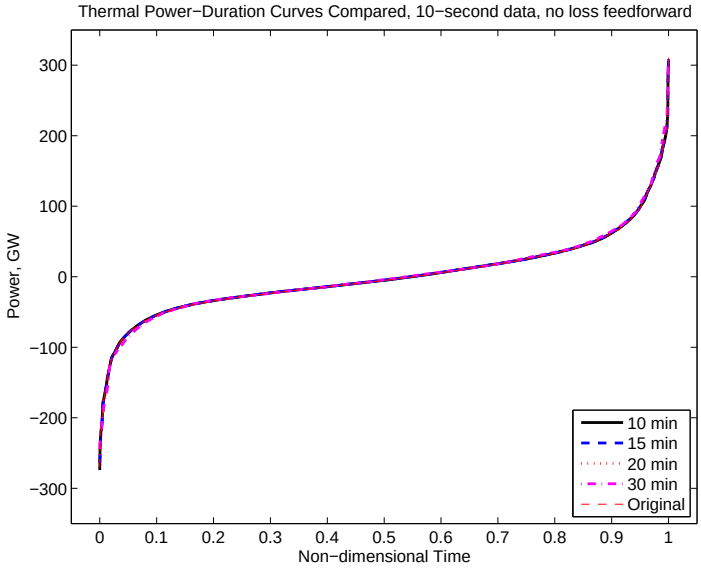

Figure 5.20: Power duration curve of the thermal unit and the total virtual power plant output compared to the total system power for data set A. This graph represents the fraction of time that the thermal unit and the virtual power plant are each delivering an amount of power less than or equal to the curve amount. The thermal unit provides the majority of the energy in the system.

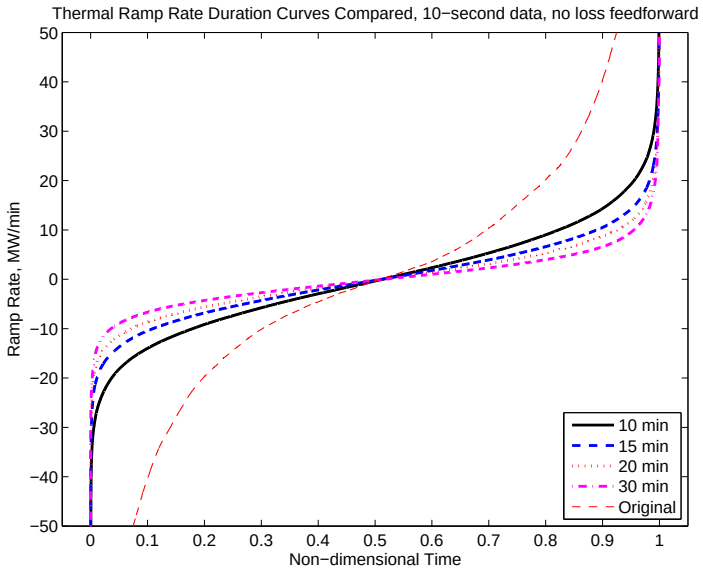

Figure 5.21: Ramp rate duration curves for the thermal unit for data set A. These may be compared to the ramp rate indicated by the original signal. This graph represents the fraction of time that each element had an instantaneous rate of change of less than or equal to the curve amount. The tails of the graph extend somewhat past the edge of the plot; the maximum values are approximately ±55 to ±1500 MW/min.

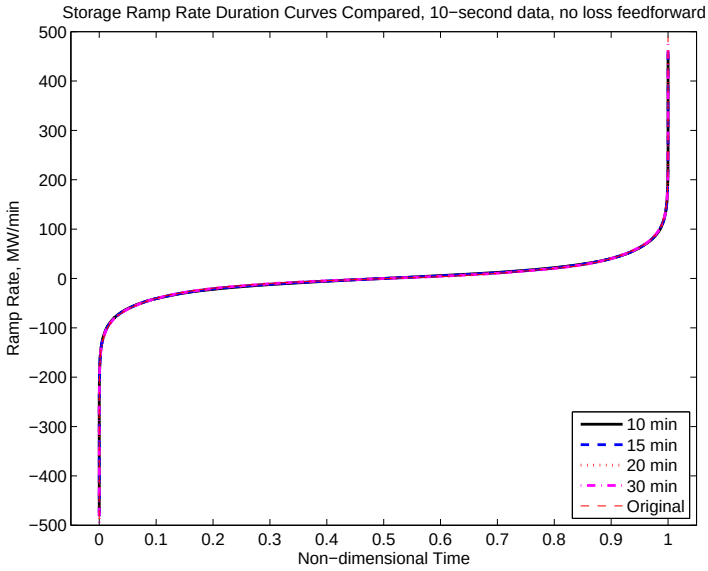

Figure 5.22: Ramp rate duration curves for the storage unit for data set A. These may be compared to the ramp rate indicated by the original signal. Nearly all of the required ramping capability is provided by the storage units.

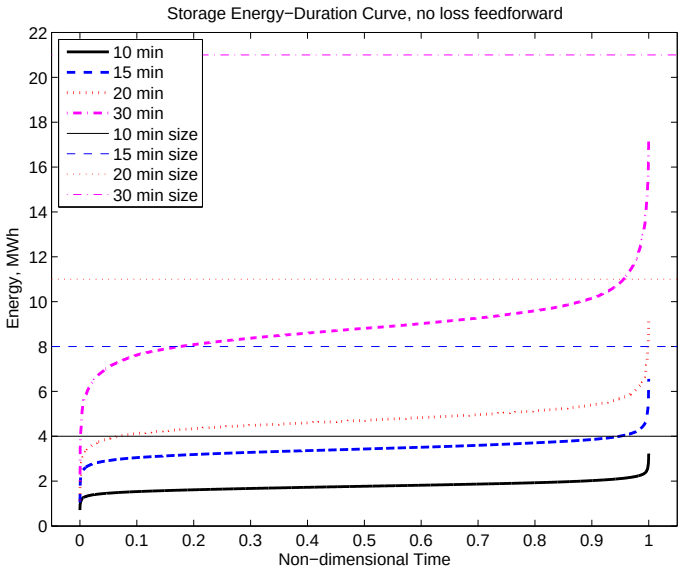

Figure 5.23: Energy duration curve of the storage unit for data set A. The graph represents the fraction of time that the storage unit spent at a state of charge less than or equal to the curve value. The energy capacities of the storage units for each simulation are listed in Table 5.1. The storage units never completely fill or empty over the simulation period, and spend the large majority of their time in the middle of their charge ranges.

for the energy storage unit while being added to the signal for the thermal plant. If the expected loss calculation is a moderately good model of the loss in the storage unit, then this fed-forward loss correction will help keep the storage unit from tending to empty.

The loss model used here in the feed-forward part of the system is intended to be a good, largely unbiased estimate of the actual storage unit losses, but to be missing some terms which cause it to match the actual storage losses only approximately. This is intended

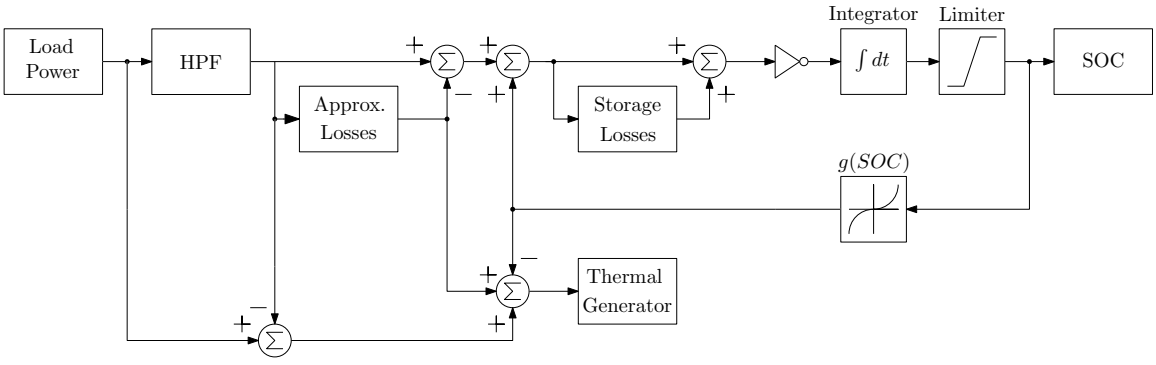

Figure 5.24: Block-diagram overview of the virtual power plant system, with loss modeling and approximate loss feed-forward included.

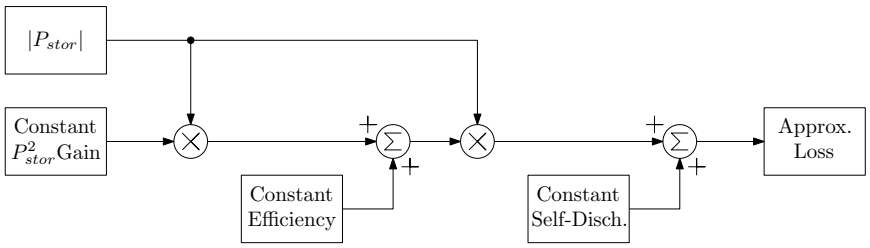

Figure 5.25: Block-diagram overview of the approximated model of storage losses.

to mimic the imperfect loss modeling expected for a real energy storage unit. The loss model, illustrated in Fig. 5.25, includes efficiency with a constant efficiency term and a term proportional to the power level (squared power term), as well as a constant self-discharge power. The values of the constants for the squared power term and the constant self-discharge term are the same as for the model of the real system. The value of the base loss fraction is equal to the base value in the storage model plus an offset for the term proportional to state of charge corresponding to a 40% or 60% storage state of charge. The coefficients of all the terms in the storage loss model and approximate loss model may also be found in Table 5.2.

The system with the inclusion of feed-forward approximate loss should be more stable than the system with loss represented but not predicted. The stability of the system was calculated in Subsection 5.3.3 by comparing the magnitude of the feedback function to the magnitude of the loss function. The difference between the approximate feed-forward loss and the full storage loss model will always be less than the worst case full storage loss, because some of the terms precisely cancel. This means that a smaller feedback is required to ensure stability, and the range over which stability is guaranteed is expanded.[3]

The performance of the system with feed-forward approximate loss included is compared to that of the system with feedback only for the data set used here. The difference between the state of charge of the system with the feed-forward approximate loss and the system without over the course of a day for the 20 minute filter is shown in Fig. 5.26. While the two scenarios start out at the same point, the system with loss feed-forward settles to a higher state of charge than that without, by about a megawatt-hour. This is a substantial amount compared to the 11 MWh storage unit in this simulation. The mean difference in the instantaneous state of charge between the cases with and without loss feed-forward may be calculated, using only the last 20 hours of each day, during which the unit seems to have settled to steady-state. These mean differences are listed in Table 5.3. The difference in behavior with loss feed-forward may also be seen in the power delivered by the storage and

[3]If desired, the feedback coefficient could also be reduced for this configuration, so long as the sign of the overall feedback remains correct under all conditions.

Filter Cutoff	Mean SOC difference	SOC fractional difference
10 min	0.24 MWh	6.1%
15 min	0.56 MWh	7.0%
20 min	0.79 MWh	7.2%
30 min	1.60 MWh	7.6%

Table 5.3: Difference in mean state of charge between case with loss feed-forward and without loss feed-forward over last 20 hours of all 7 days for each filter, as well as normalized fraction of total capacity represented by the mean difference.

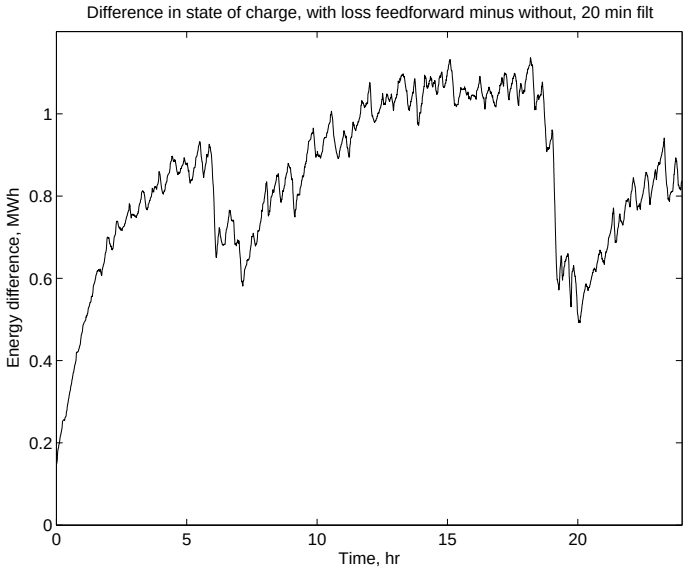

Figure 5.26: Difference between state of charge with and without approximate loss feed-forward for 20 minute filter over the first day of data set A.

thermal devices in each case, as in Fig. 5.27 (for a single day for the 20 minute filter case). Once the state of charge has settled, the average power delivered by the storage units and the thermal units is the same in the case with loss feed-forward as in the case without loss feed-forward.

The overall performance of the system with fed-forward approximate loss is also represented by duration curves in Figs. 5.28 through 5.32, which are analogues to the curves in Figs. 5.19 through 5.23. The inclusion of loss feed-forward allows the storage unit to operate closer to its intended middle state of charge, because the additional loss power drawn out of the storage has been accounted for.

The plot of Fig. 5.11 can be updated to reflect the results of the closed-loop filtering simulation, both with and without the approximate loss feed-forward. For these plots, the total energy storage is computed as the difference between the maximum and the minimum

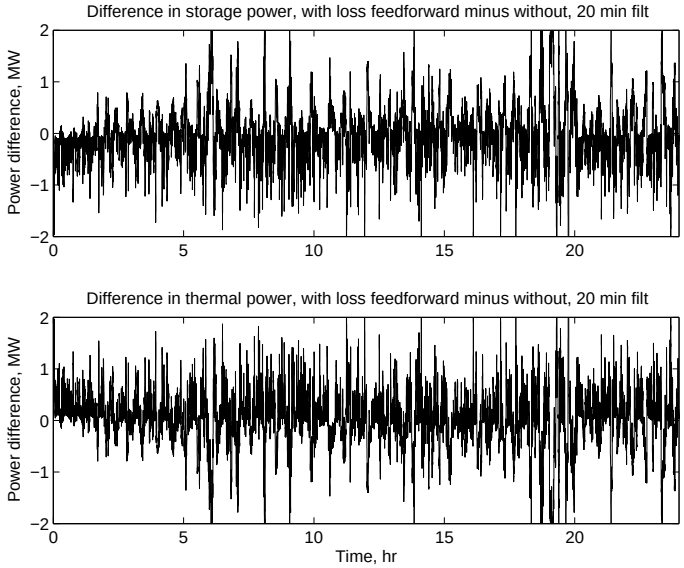

Figure 5.27: Difference between power delivered by storage unit with and without approximate loss feed-forward for 20 minute filter over the first day of data set A.

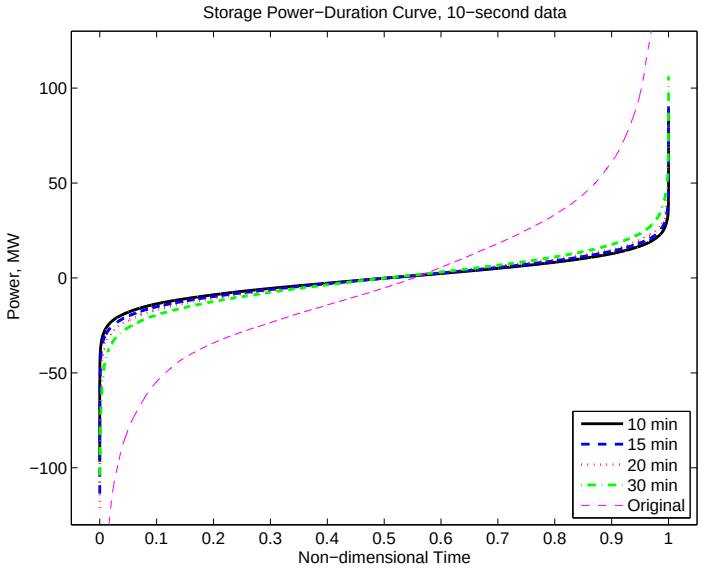

Figure 5.28: Power duration curve of the output of the energy storage unit for data set A with approximate loss included in feed-forward, as well as the total system power. Compare to Fig. 5.19.

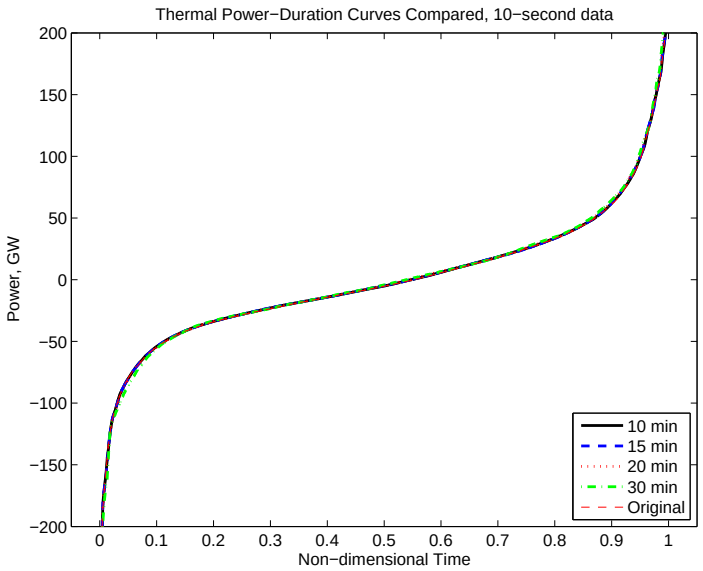

Figure 5.29: Power duration curve of the thermal unit and the total virtual power plant output for data set A with the approximate loss included in feed-forward, compared to the total system power. The thermal unit provides the majority of the energy in the system. Compare to Fig. 5.20.

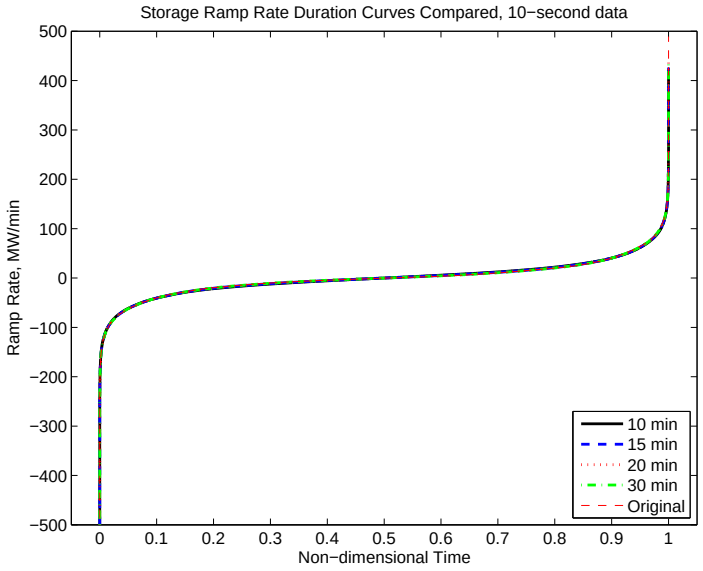

Figure 5.30: Ramp rate duration curves for the storage unit for data set A with the approximate loss included in feed-forward. These may be compared to the ramp rate indicated by the original signal. Similar to Fig. 5.22.

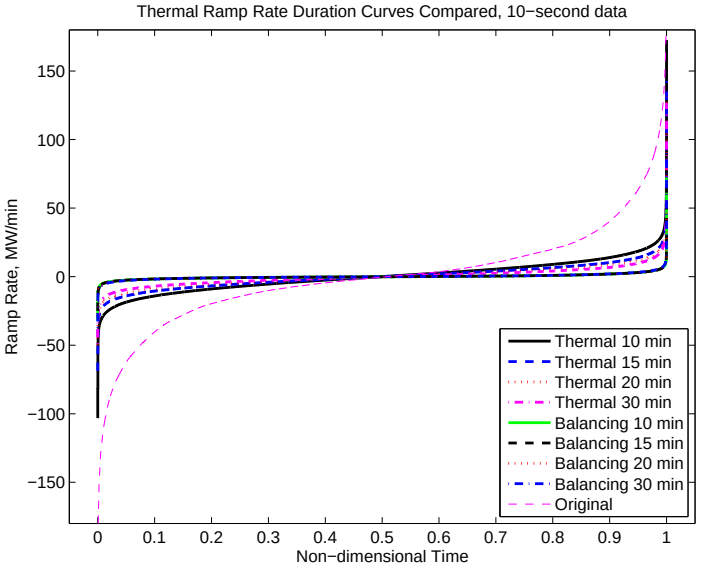

Figure 5.31: Ramp rate duration curves for the thermal unit for data set A with the approximate loss included in feed-forward. These may be compared to the ramp rate indicated by the original signal, and to the balancing signal required of the thermal unit to account for energy losses and to drive the storage toward its half-full state. Compare to Fig. 5.21.

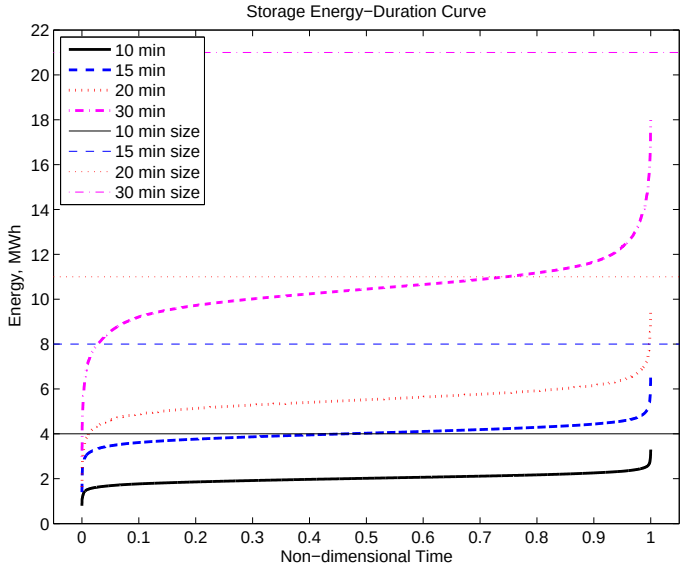

Figure 5.32: Energy duration curve of the storage unit for data set A with the approximate loss included in feed-forward. The energy capacities of the storage units for each simulation are listed in Table 5.1. The storage units never completely fill or empty over the simulation period, and spend the large majority of their time in the middle of their charge ranges. Compare to Fig. 5.23.

state of charge, in accordance with the conventions of the previous figures, rather than the allowed capacity (as was listed in Table 5.1). This graph is presented as Fig. 5.33.

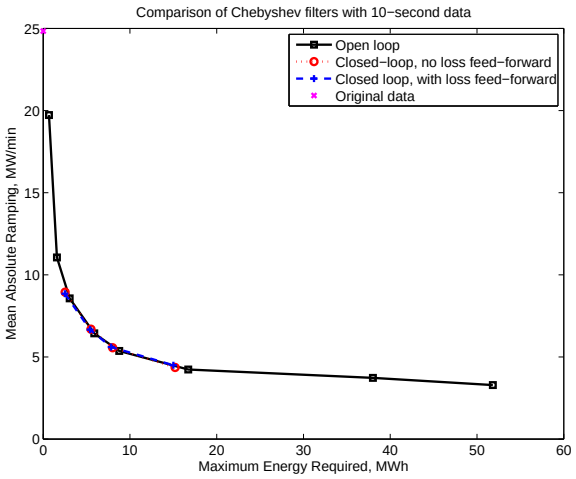

Figure 5.33: Comparison of different filters in terms of average absolute ramp rate versus maximum energy storage requirement for data set A. The filters of Fig. 5.11 are included as well as the closed-loop filters of this Section.

5.4 Discussion

In order to reduce the ramping required of traditional thermal power plants performing frequency regulation, the regulation burden is divided between these plants and an energy storage unit of a small size and moderate power capacity. The thermal power plants continue to deliver most of the energy associated with frequency regulation, but respond only to the slower-moving portion of the signal and hence are required to ramp at slower rates and less frequently. The energy storage units respond to the fastest fluctuations, and are able to do so with a limited amount of energy available. The ramp rate duration curves introduced in Chapter 2 are used to evaluate the benefits in terms of reduced ramping, and the energy duration curves (also introduced in Chapter 2) help to visualize the state of charge performance of the energy storage unit. More familiar power duration curves also indicate the power requirements of both types of assets.

In order to compare the performance of a dispatch strategy for energy storage and traditional thermal power plants acting in concert, the requirements for each type of unit can be distilled into a single metric, and plotted together to indicate the relative merits of various strategies. The most relevant metric for the decrease in thermal plant ramp rate provided as a result of the addition of energy storage is a time-average ramp rate, with mean absolute ramping used

here. For energy storage units, the maximum energy requirement provides a good metric. Both of these single metrics are related to the cost associated with capital investments and system operations (primarily component wear for ramping and storage system cost for energy).

In a real system, a more sophisticated strategy than the simple open-loop filtering of Section 5.2 is required. For example, the energy storage unit should not be allowed to empty completely due to power conversion losses; some feedback control of the storage state of charge is required. A simple closed-loop model of signal separation and control with an energy storage unit was discussed in Section 5.3, and its performance was similar to that of the open-loop system with the same filters.

Overall, this chapter demonstrated how a significant reduction in thermal plant ramp rate is possible with the incorporation of an energy storage unit to cover the fast fluctuations in frequency regulation. Data set A was used here, and certain characteristics of data set A are required for the good performance of this strategy. These favorable characteristics are not present in all frequency regulation signals, and this leads to variable performance of the signal separation scheme. Data set B lacks some of the favorable qualities of data set A, degrading the effectiveness of similar filters to those used in this chapter. These important differences in its signal structure are explored in detail in Chapter 6.

Chapter 6

Importance of Signal Characteristics

AS WAS demonstrated in Chapter 5, one way to partition the required regulation power for a control area is to pass the regulation signal through a filter which separates the high and low frequency components of the signal, and send the high frequency component to the energy storage unit while sending only the low frequency component to the thermal generators on regulation duty. The benefits of this technique in terms of reduced ramping of the thermal generators using only a moderate amount of energy storage were indicated.

However, data set A (used for the analysis of Chapter 5) possesses some characteristics which make it particularly suitable to this treatment. Data set B (also introduced in Section 2.3) does not share all of these characteristics. This leads to a smaller reduction in thermal unit ramping when the signal is divided into high-frequency and low-frequency portions. The differences in data structure which lead to this performance difference are described in this chapter.

6.1 Differences in the Time and Frequency Domains

The relevant differences between data sets A and B may be explored in both the time domain and the frequency domain. A first step is to examine the signals up close. When a whole day, a whole week or more of a regulation signal versus time is plotted in a single graph, it is difficult to see any features clearly, but a plot of a few hours of a signal is more revealing. Looking at the two signals in Fig. 6.1, data set A seems to have far more rapid, small fluctuations on top of the general trends as compared to the original data set B; it is "fuzzier." This is a first hint that perhaps the frequency content of the waveforms may differ in important ways. One may imagine that if data set A is "fuzzier," it contains relatively more high frequency component, making it easier for a filter to remove the "fuzziness" and thus reduce the overall ramping in the signal substantially. Data set B, sampled at 4 seconds, by contrast might contain relatively less high frequency component, meaning that any filter must be much more specialized to successfully smooth the signal and reduce the overall ramp rate.

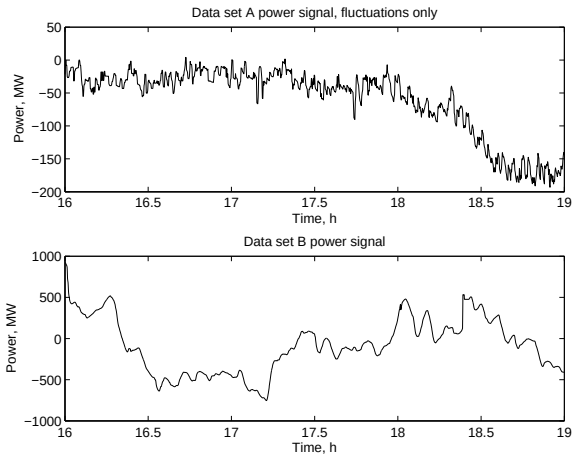

Figure 6.1: A three-hour portion of each of data sets A and B, for visual comparison. Note the qualitative difference in "fuzziness" between the two signals, possibly indicating a difference in frequency content.

Next the frequency domain behavior of the two signals may be examined. The power spectral density (PSD), or squared transform magnitude, is used in this case because the signals are stochastic in nature, and the standard Fourier transform does not converge in a statistical sense with the length of a data set [52, Chapter 6]. Welch's method of finding the power spectral density is used here, which takes windowed overlapping segments of the original signal, performs the Fourier transform, and averages across the result. A Hamming window is used to provide a compromise between high frequency resolution and reducing leakage through side lobes [102, Chapter 2]. The PSD for each day of data set A and for each week of data set B are in Figs. 6.2 and 6.3. Since the PSD curves for the different periods of each signal are very similar, these are again averaged to produce the plots of Fig. 6.4. From this plot, it seems that the magnitude of the transform of data set A falls off much less quickly with increasing frequency, meaning that it may be more dominated by its high frequency component.

Another way to compare the frequency domain behavior of the two data sets is by integrating the PSD curves to get the portion of the signal power[1] in each frequency band. This cumulative power is then normalized to the total signal power, yielding the fraction of the total signal energy at or below each frequency. These curves, generated by a numerical integration of Fig. 6.4, are included as Fig. 6.5. Examining this figure, it would seem that data set B does have a larger portion of its signal power in the very low frequency portion of the spectrum, compared to data set A. While the results observed in the frequency domain

[1]Note the distinction between these signals, which are in physical units of power (MW), and the convention of signal power as the square of the signal, in this case in units of generated power squared per frequency (MW^2/Hz).

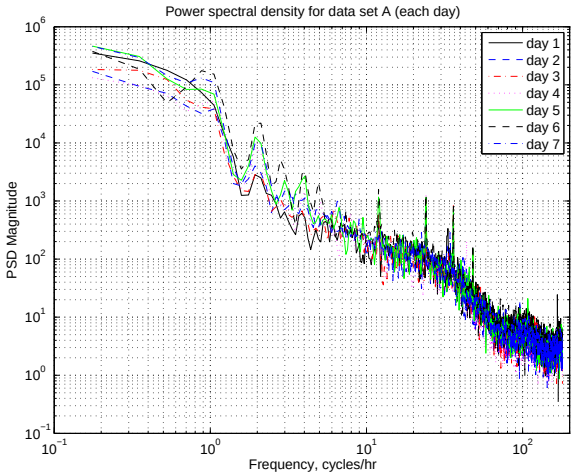

Figure 6.2: Power spectral density of data set A, calculated using Welch's method, using eight overlapping segments resulting in a somewhat larger minimum frequency than a plain Fourier transform. The spectrum of each day has been taken separately. All days exhibit similar shapes. In Fig. 6.4, all days are averaged to reduce variance.

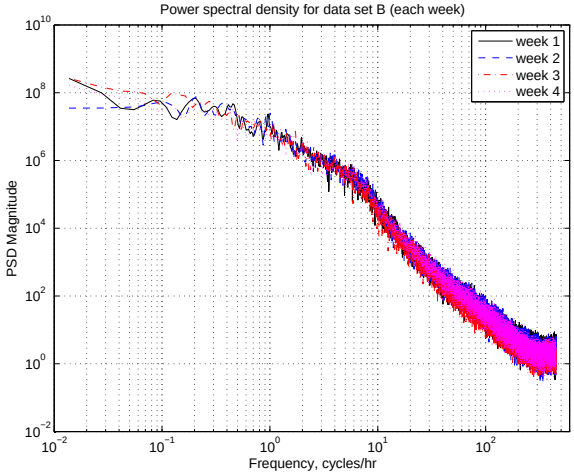

Figure 6.3: Power spectral density of data set B, calculated using Welch's method, using eight overlapping segments resulting in a somewhat larger minimum frequency than a plain Fourier transform. The spectrum of each week has been taken separately, and all weeks demonstrate good grouping, enabling the use of averaging across weeks to reduce variance.

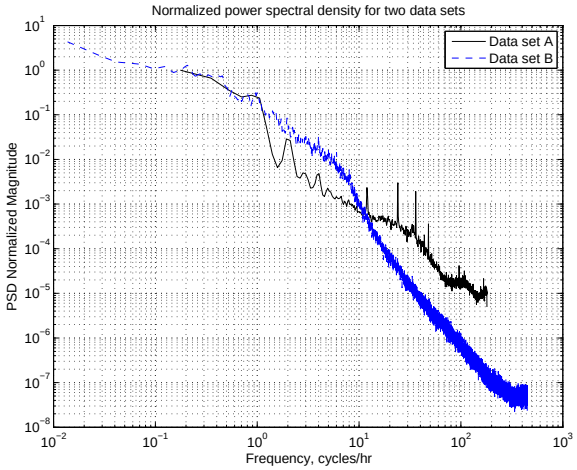

Figure 6.4: Normalized mean power spectral density of data sets A and B, with all time periods of Figs. 6.2 and 6.3 averaged for each data set. For easy comparison, the spectra have been normalized to have unit value at the lowest frequency observed in both data sets.

are not unequivocal, they do seem to be consistent with the difference in the signal shapes which is evident visually in Fig. 6.1.

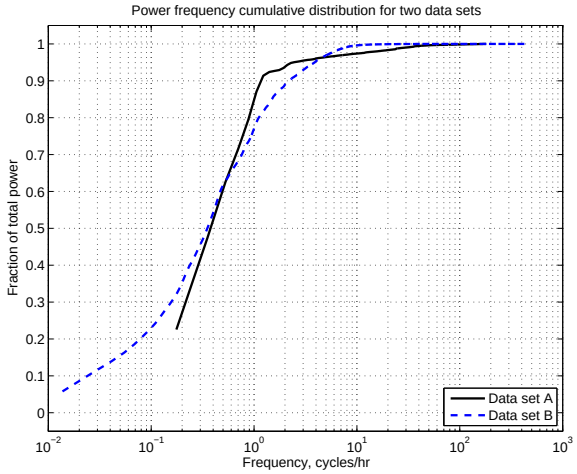

Figure 6.5: Power spectral distribution, i.e., the integrated power spectral density normalized to unit signal power, for data sets A and B. The curves show the fraction of the total signal power at a frequency less than or equal to the indicated value. Data set B seems to contain relatively more power at low frequencies than data set A.

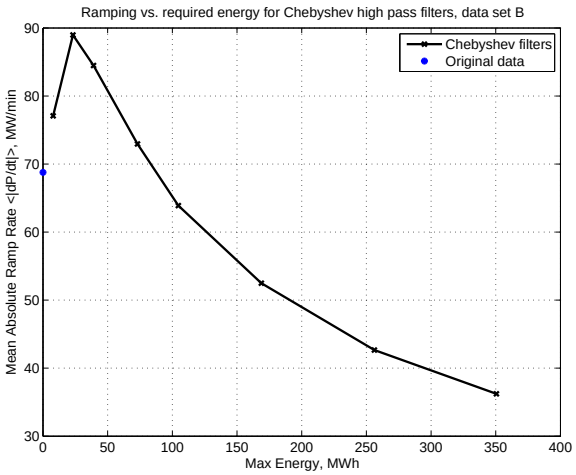

Figure 6.6: Mean absolute ramp rate of low frequency portion versus required energy for high frequency portion for data set B using the same family of Chebyshev type 1 high pass order 3 filters as in Fig. 5.11, converted for 4 second sampling rate. Filter cutoff frequencies correspond to 3 minutes, 7 minutes, 10 minutes, 15 minutes, 20 minutes, 30 minutes, 45 minutes, and 60 minutes, from upper left to lower right.

6.2 Importance of Signal Characteristics to Filtering Performance

In the previous section, a first indication is given that data sets A and B possess different structure. When data set B is passed through similar filters to those in Chapter 5, the low frequency portion ends up with a mean absolute ramp rate which is sometimes lower and sometimes *higher* than that of the original data, where we would expect the mean absolute ramp rate of the low frequency portion to always be lower than that of the original data, based on our experience with data set A in Chapter 5. Figure 6.6 illustrates the problem, as for higher filter cutoff frequencies the mean absolute ramp rate of the low frequency portion of the signal is higher than the ramping of the original signal. For lower filter cutoff frequencies, the ramping has been reduced a moderate amount, but at the cost of a much larger energy storage requirement. This is a very different result from that with data set A (for example see Fig. 5.11), and indicates a potential challenge in using the approach of Chapter 5. A similar plot which uses the root mean square (RMS) ramp rate rather than the mean absolute ramp rate is included as Fig. 6.7. The RMS ramp rate of the filtered portion is never larger than that of the unfiltered portion, but neither does the ramp rate show any substantial reduction as a result of the filtering for several of the filters.

The duration curves for data set B, in the format introduced in Chapter 2 and using the same filters as in Fig. 6.6, are included as Figs. 6.8–6.11. Again, Fig. 6.8 shows that for

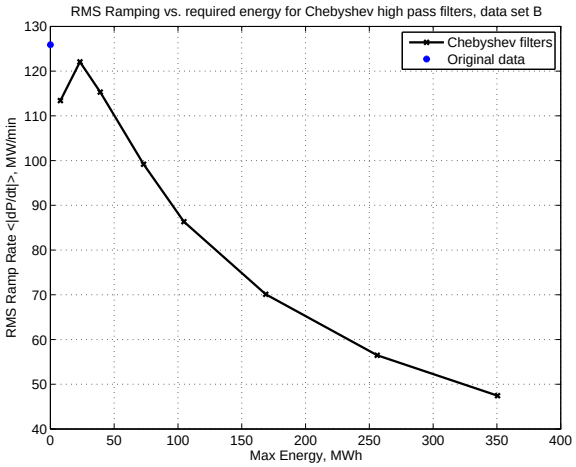

Figure 6.7: RMS ramp rate of low frequency portion versus required energy for high frequency portion for data set B using the same family of Chebyshev type 1 high pass order 3 filters as in Fig. 5.11, converted for 4 second sampling rate. Filter cutoff frequencies correspond to 3 minutes, 7 minutes, 10 minutes, 15 minutes, 20 minutes, 30 minutes, 45 minutes, and 60 minutes, from upper left to lower right. While the all filters reduce the RMS ramp rate compared to the unfiltered signal, that reduction is minimal for some filters and limited for all filters.

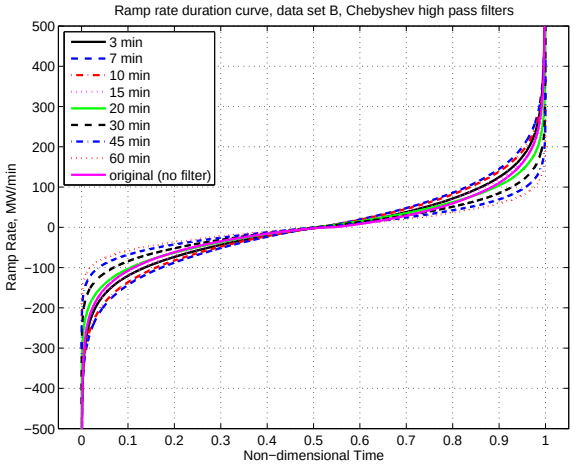

Figure 6.8: Ramp rate duration curve for low frequency portion of data set B obtained using Chebyshev type 1 high pass order 3 filters with a range of cutoff frequencies corresponding to 3 to 60 minutes. Note that ramping is consistently higher for the low frequency portion for some of the filters than for the original data.

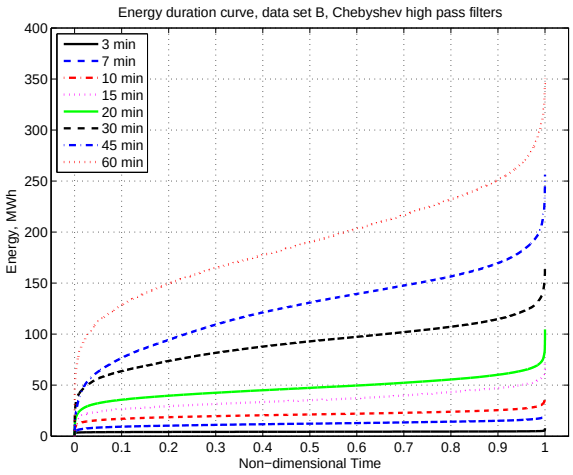

Figure 6.9: Energy duration curve for high frequency portion of data set B obtained using Chebyshev type 1 high pass order 3 filters with a range of cutoff frequencies corresponding to 3 to 60 minutes.

certain filters, the low frequency portion of the signal has a uniformly larger ramp rate than the original. By normalizing both the ramp rate and the energy storage to the mean bulk power of the control area over the recorded intervals, the two data sets may be compared

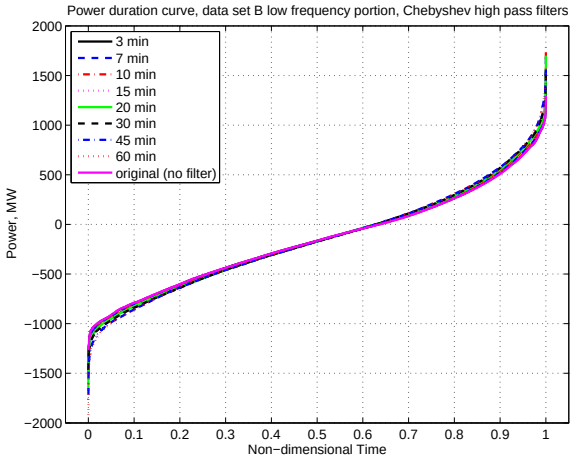

Figure 6.10: Power duration curves for low frequency portion of data set B filtered using Chebyshev type 1 high pass order 3 filters with a range of cutoff frequencies corresponding to 3 to 60 minutes.

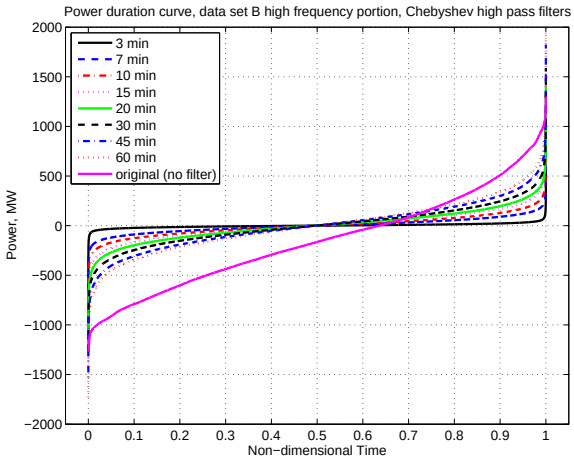

Figure 6.11: Power duration curves for high frequency portion of data set B filtered using Chebyshev type 1 high pass order 3 filters with a range of cutoff frequencies corresponding to 3 to 60 minutes.

on approximately equal terms.[2] Both data sets A and B are plotted together in terms of normalized ramp rate versus normalized energy requirement in Fig. 6.12.

6.3 Origin of Differences

There are a few reasons we might expect data set B to be different from data set A, the behavior of which was investigated in Chapter 5. One is that it is from a substantially larger control area, as is clear from comparing Figs. 2.2 and 2.12. Control area size could also affect the degree of control that is required for good performance, based on control performance standards [76] and other considerations, as discussed in Section 1.1. There are a larger number of varying loads in a larger control area, and so there may be more averaging of these loads. A larger control area with more generation resources contains more physical inertia in the rotating machines. The degree of interconnection with neighboring areas is a consideration, since the area control error (ACE) calculation used to determine compliance with CPS 1 and CPS 2 incorporates both an inadvertent interarea interchange term and a frequency term.

[2]There is no guarantee that either the ramping characteristics or the energy storage capacity required to cover fast fluctuations will scale linearly with control area size. If a load power signal were the superposition of truly uncorrelated variables, a scaling with the square root of size might be expected. However, a linear normalization provides a good approximate comparison.

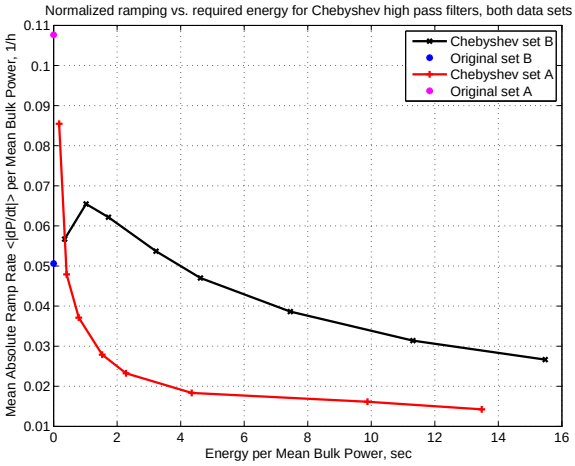

Figure 6.12: Mean absolute ramp rate of low frequency portion versus required energy for high frequency portion for data sets A and B using the same family of Chebyshev type 1 high pass order 3 filters. A superposition of Figs. 6.6 and 5.11, with all variables normalized to control area mean bulk power over the data interval. Filter cutoff frequencies correspond to 3 minutes, 7 minutes, 10 minutes, 15 minutes, 20 minutes, 30 minutes, 45 minutes, and 60 minutes, from upper left to lower right.

A second reason we might expect the data sets to be different is that they are measured slightly differently. A high level illustration of the origin of these two signals is included as Fig. 6.13. Data set B is a control signal for the generators performing frequency regulation whereas data set A includes the total power generation (and hence load) of the control area. The control signal is thus dominated by the dynamics of the balancing area's controller logic, and the total generation power signal is dominated by the collective dynamics of the generators performing frequency regulation.

It is expected that both the regulation controller and the power plant dynamics will be generally low pass in nature, in a describing function sense. However, both are likely to incorporate nonlinearities (such as deadbands, rate limiters, and others) and the thermal

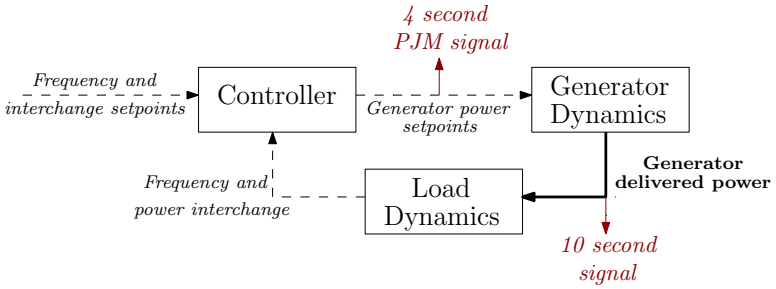

Figure 6.13: A very high level schematic representation of frequency control on the electric grid, indicating the difference in sensing location between the two signals.

generators in particular are likely to exhibit state and time dependencies which may make their responses less smooth. Hydraulic turbines also generally include dynamics best modeled by right half-plane zeros, which produce contrary power step response. For example, see the generator step responses in [114, 112] and the data in [25]. Further, the generator dynamics block in Fig. 6.13 is not one generating plant but a group of plants within a control area. These plants may each be following a slightly different signal, each with different fidelity. This summing of many responses, with some errors in coordination, could create a signal which has different characteristics from the single control signal like that of data set B. These distinctions become especially important when making determinations about the control performance which is required.

A further reason we might expect the two signals to differ is that the control philosophies of the two control areas could be different. Many different control strategies could potentially be used to comply control standards, as they are quite generic. Even two control areas of similar size and composition might use different internal standards of control to achieve compliance, resulting in tighter or looser system control. Such differences in philosophy could affect the power quality to customers, the requirements on the participating generators, and the inadvertent power interchange with neighboring control areas, among other things. The benefits of tighter versus looser control have been under discussion in the literature for some time; for example [51, 74, 94, 19, 17, 36, 97, 48, 37, 1, 42]. An evaluation of the relative merits of such control philosophies is outside the scope of this work, but it is a potentially important consideration driving the sort of generation control being considered here.

As noted in Section 2.3, data set A was treated with a median filter prior to further analysis, to eliminate anomalous single data points. Because data set B did not exhibit similar single point spikes, a median filter was not used. It is not expected that this median filter would cause the difference in behavior under filtering which is described here between the data sets, but for completeness, data set B was processed with two median filters which are analogs to the 5 point, 50 second filter used with data set A. The first is a 5 point, 20 second median filter and the second is a 13 point, 52 second median filter. The median-filtered data set B was then passed through the same Chebyshev high pass filters as in Fig. 6.6. The mean absolute ramp rate of the low frequency portions of the median filtered data are plotted against the maximum energy required in the high frequency portions in Fig. 6.14. It is clear from the figure that the median filter makes very little difference in the behavior of data set B processed by the Chebyshev filters, with the results from the median filtered and unfiltered data sets nearly identical.

Also as discussed in Section 2.3, data set A included the bulk load of the control area in addition to the fast fluctuations of interest. In order to remove the bulk load and produce

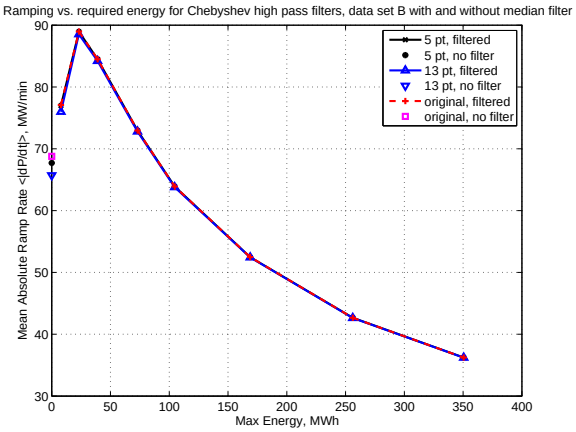

Figure 6.14: Mean absolute ramp rate for low frequency portion versus required energy for high frequency portion of data set B, after processing by a 5 point (20 second) median filter; after processing by a 13 point (52 second) median filter; and without any median filtering. Filter cutoff frequencies correspond to 3 minutes, 7 minutes, 10 minutes, 15 minutes, 20 minutes, 30 minutes, 45 minutes, and 60 minutes, from upper left to lower right.

a suitable frequency regulation signal, a predictive filter was used. Data set B does not include any bulk load component, as it is the frequency regulation command signal to the area generators under automatic control, and hence no such predictive filter was necessary. However, data set B is not precisely zero-average, and so a 90-minute non-causal (predictive) averaging filter was also tested to demonstrate that such a filter did not lead to the resulting differences in behavior. The original and average power time series of data set B is shown in Fig. 6.15. The power duration curve of the average power, indicating the fraction of time that the average power deviates from zero by a given amount, is included as Fig. 6.16. The filtering performance on data set B after the 90-minute average power has been removed is compared to the original version of data set B in Fig. 6.17, which shows the mean absolute ramp rate versus maximum energy storage requirement for both cases. There is a small difference for the slowest filters, but particularly for faster filters, the filter performance is the same with or without the predictive filter.

Finally, a very clear difference between the two data sets is that data set B is sampled at 4 second intervals, more than twice the 10 second sampling rate of data set A. Whether this difference in sampling rate leads to the differences in performance that are evident is an important consideration. One way to check this hypothesis is to resample data set B from 4 seconds to 10 seconds, assuming it to be band-limited below the Nyquist frequency corresponding to 4 seconds.[3] The same calculations are then performed with data set B

[3]While we do not have a strong assurance that the original signal is band limited in this fashion, the analysis of this work in general also assumes this to be the case, so no inconsistency is introduced by this assumption here.

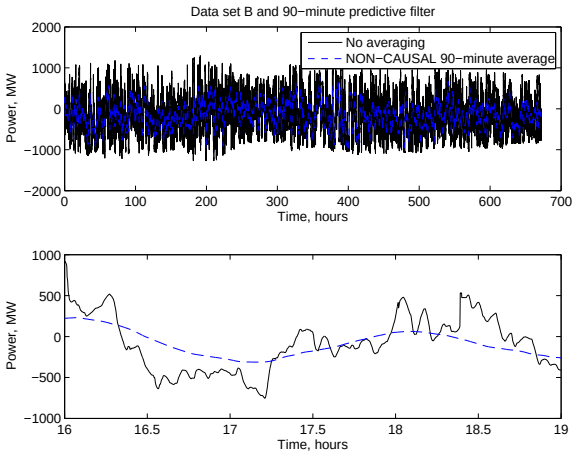

Figure 6.15: Average 90-minute power of data set B using a predictive (non-causal) filter similar to that used for data set A in Subsection 2.3.1, compared to the complete signal. Both the full data set and a portion are shown.

resampled to 10 seconds as with data set A, also sampled at 10 seconds, using the same filters, and performance can be directly compared. Figure 6.18 shows the behavior of data set B resampled to 10 seconds using the same filters as used with data set A. The performance of data set B resampled to 10 seconds under this class of filters is almost identical to the performance of data set B with the original 4 second sampling rate. The undesirable increase in mean absolute ramp rate of the low frequency portion of the signal when compared to the total signal of data set B (resampled to 10 seconds) is still evident. This indicates that the differences in sampling frequency do not cause the different behavior of the two data sets.

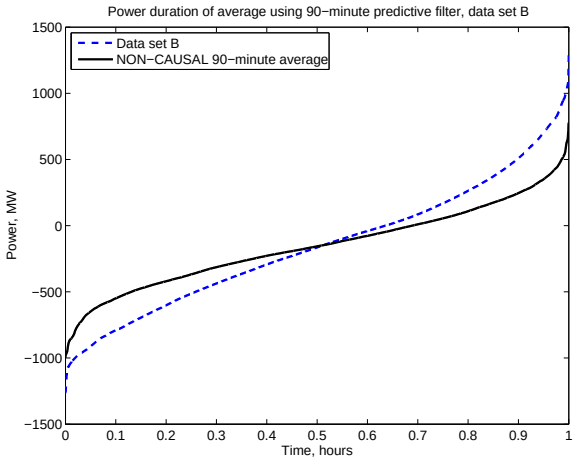

Figure 6.16: Power duration curve of the 90-minute predictive (non-causal) average of data set B, compared to the power represented by the complete signal.

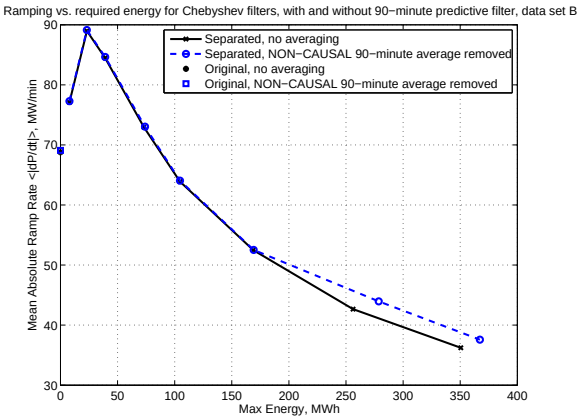

Figure 6.17: Mean absolute ramp rate for low frequency portion versus required energy for high frequency portion of data set B, before and after removal of 90-minute average power using a non-causal (predictive) sliding window filter similar to that used for data set A in Subsection 2.3.1. Filter cutoff frequencies correspond to 3 minutes, 7 minutes, 10 minutes, 15 minutes, 20 minutes, 30 minutes, 45 minutes, and 60 minutes, from upper left to lower right. The filter performance is very similar with and without the use of the averaging filter.

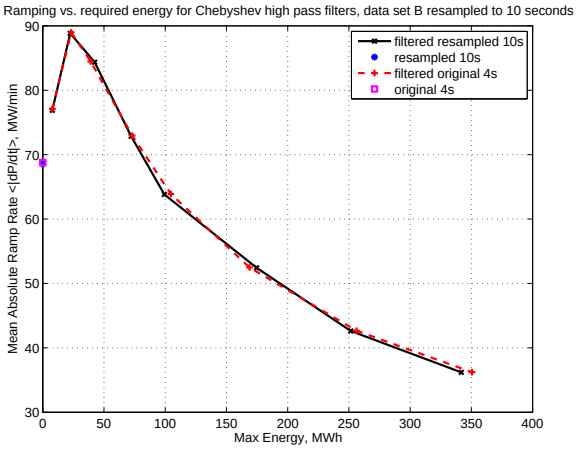

Figure 6.18: Mean absolute ramp rate for low frequency portion versus required energy for high frequency portion of data set B resampled to 10 seconds obtained using same Chebyshev type 1 high pass order 3 filters as in Fig. 5.11. Compare to Fig. 6.6, with original data set B and similar filters designed for 4 second sampling rate. Filter cutoff frequencies correspond to 3 minutes, 7 minutes, 10 minutes, 15 minutes, 20 minutes, 30 minutes, 45 minutes, and 60 minutes, from upper left to lower right.

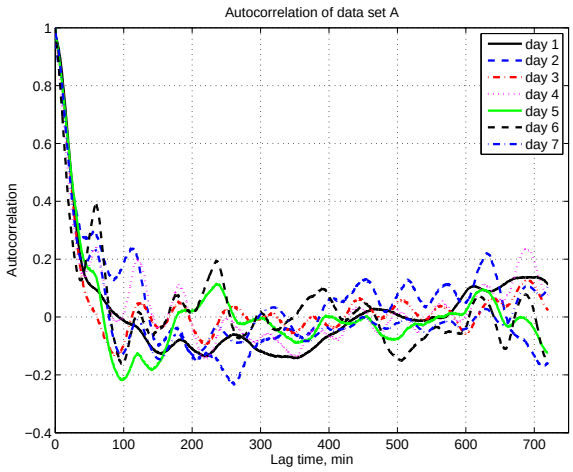

Figure 6.19: Autocorrelation for each day of data set A. Note that the values are somewhat grouped together for the different days.

6.4 Alternative Descriptions of Structural Differences

A further good descriptor of stochastic signals is autocorrelation, i.e., how closely data points are (linearly) related to the point n lags back, normalized to the signal variance [7, Section 2.1]. For a series x_t, the autocovariance at lag k, c_k, and the autocorrelation at lag k, r_k, are as follows:

$$r_k = \frac{c_k}{c_0} \tag{6.1}$$

$$c_k = \frac{1}{N}\sum_{t=1}^{N-k}(x_t - \bar{x})(x_{t+k} - \bar{x}) \tag{6.2}$$

$$k = 1, 2, 3, \ldots$$

The autocorrelation for each separate day of data set A and for each week of data set B was computed, and they are plotted together in Figs. 6.19 and 6.20. For each data set, the autocorrelations from the separate days or weeks are averaged, and are plotted together in Fig. 6.21. While it is clear that there are differences in autocorrelation between the two data sets, the meaning of those differences is much less clear.

Another statistical metric less commonly used in electrical engineering is the partial autocorrelation [7, Section 3.2]. Partial autocorrelation is the linear relation between a point and the point n lags back, with the effect of the intervening points removed. For example,

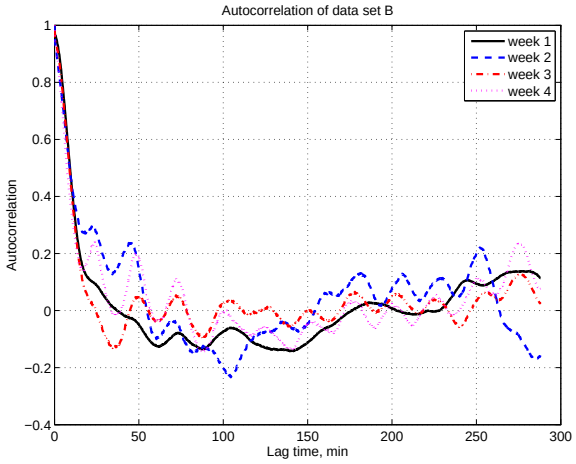

Figure 6.20: Autocorrelation for each week of data set B. Note that the values are somewhat grouped together for the different weeks.

if a stochastic sequence is formed by taking the weighted sum of the previous point with a random shock, the sequence will have nonzero autocorrelation out to many shocks, as the effect of the first point is seen in the second, the effect of the second, and hence also the first, is seen in the third, and so on. Partial autocorrelation seeks to remove this effect, and separate only the direct effect from the previous points from their effects carried through the intervening points.

The autocorrelation and partial autocorrelation are important indicators for a class of Box-Jenkins models known as autoregressive (AR) models [7, Section 3.2]. An autocorrelation

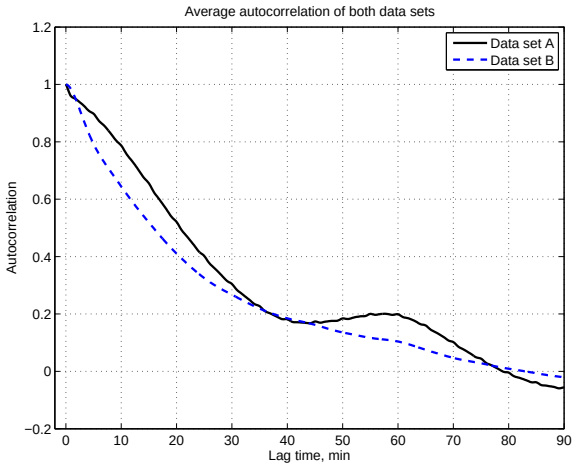

Figure 6.21: Mean autocorrelation for data sets A and B. While the curves are somewhat different, the differences in shape do not offer intuition as to the differences between the underlying signals.

model of order p (AR(p)) is a stochastic model where the next point is a weighted sum of the p previous points and a random (independent identically distributed, IID) "shock" or fluctuation.

$$\begin{aligned} y_t &= y_{t-1}\phi_1 + y_{t-2}\phi_2 + \ldots + y_{t-p}\phi_p + a_t \\ a_t &\sim \text{IID} \end{aligned} \tag{6.3}$$

The values of ϕ_i determine the characteristics of the model. An AR model may also be thought of as a digital filter with an infinite impulse response (IIR) and with tap weights ϕ_i, operating on an IID noise sequence a_t. In this context, the partial autocorrelation is closely related to the autoregressive model corresponding to a time series data set. In fact, the partial autocorrelation of lag k may also be understood as the kth coefficient ϕ_k of the best fit AR(k) model for a time series data set, in the least squares sense[4]. From this definition, it is clear that for data generated from an AR(p) model, the partial autocorrelation will be nonzero only for lags $k \leq p$. This rapid cutoff of partial autocorrelation is commonly used for model identification in time series analysis, where it indicates a potential type (AR) and order (p) for a model with a good fit (see e.g. [6, Chapter 9]).

With this in mind, the partial autocorrelation was computed for each day and week of the two data sets, separately. These are plotted in Figs. 6.22 and 6.23 for data sets A and B, respectively. Since the partial autocorrelations from the different periods for each data set are grouped together well, the partial autocorrelations were averaged across all the days (for data set A) and weeks (for data set B) and are plotted together in Fig. 6.24. The graph of Fig. 6.24 shows the pronounced difference evident between the two data sets. While the partial autocorrelation of data set A remains largely positive and slowly falls from near 1 to near 0, the partial autocorrelation takes a significant negative excursion at lag 2 before returning to near zero. These partial autocorrelations indicate that the two data sets have substantially different structure, although the partial autocorrelation does not provide the sort of intuitive understanding which is available from a frequency domain investigation. Since the autocorrelation and partial autocorrelation structure of each of these data sets is consistent with an AR model of moderate order, these models will be constructed and compared in Section 6.7 below.

[4]Plan to include the brief derivation that shows this, probably as appendix.

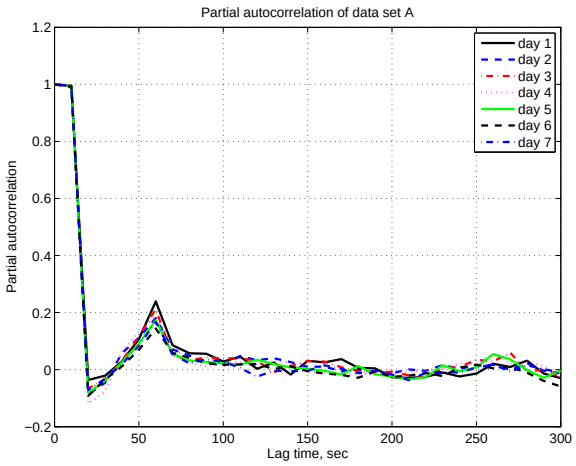

Figure 6.22: Partial autocorrelation for each day of data set A. Note that the values group together very closely for the different days.

6.5 Comparing a Structured Signal

At first glance, it is difficult to understand why any reasonably well-behaved low-pass filtering operation would lead to an *increase* in ramp rates, as opposed to a decrease or no change. This is partially due to the fact that the ramping metric used here is nonlinear, treating negative and positive ramping as equivalent. The lowest absolute ramp rate is sought, which is not quite the same as looking for the low frequency component of the signal. For example, consider a trapezoidal wave like that shown in Fig. 6.25. It has a unit amplitude and period, is symmetric, and the flat portion at the maximum and minimum

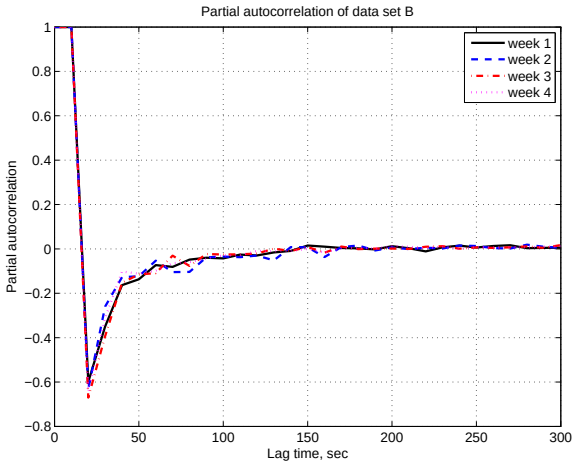

Figure 6.23: Partial autocorrelation for each week of data set B. Note that the values group together very closely for the different weeks.

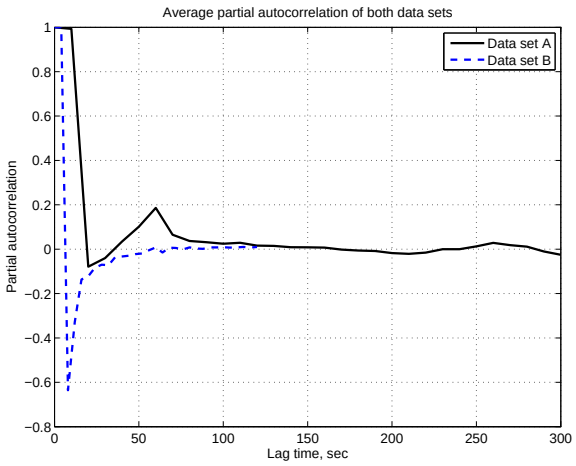

Figure 6.24: Mean partial autocorrelation for data sets A and B. The curves have quite different shapes, indicating a substantial difference in data structure.

values has length a, where $a < 0.5$. Its mean absolute ramp rate is constant at 2 independent of the value of a, because the shorter the time spent ramping, the steeper the ramp must be during that interval. For this waveform, the Fourier series is as follows [115, p. 52]:

$$f(t) = \frac{4}{\pi^2(1-2a)} \cdot \sum_{n=1,3,5,\ldots}^{\infty} \frac{\cos(n\pi a)\cos(2\pi nt)}{n^2} \tag{6.4}$$

Partial sums of the Fourier series mimic the behavior of perfect "brick wall" filters with unity transmission in the passband and zero transmission in the stopband. This is the equivalent of performing a filtering operation by taking weighted sums of the frequency components. The full trapezoidal wave (with $a = 0.25$) is pictured in Fig. 6.25 along with its second partial sum. The second partial sum is clearly smoother in the sense that it has no sharp transitions. However, it is not clear from the figure which curve will have lower mean absolute ramp rate; in fact, the original waveform has a lower mean absolute ramp rate than the second partial sum.

For this simple function, the mean absolute ramp rate of the partial sums may be calculated analytically as a function of the order of the partial sum and the pulse width a.

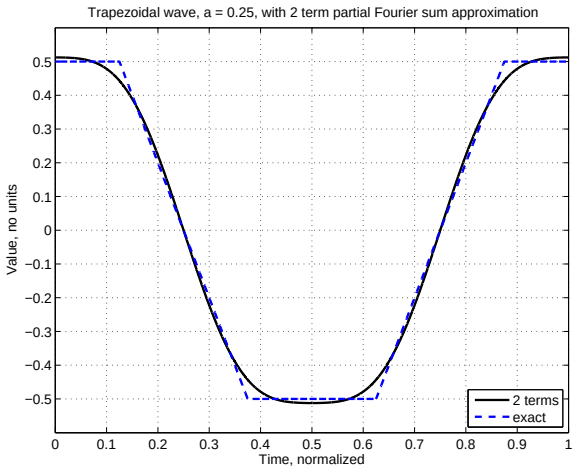

Figure 6.25: Unit trapezoidal wave with $a = 0.25$ along with the second partial sum of the Fourier series over one period.

$$\frac{df}{dt} = \frac{8}{\pi(1-2a)} \cdot \sum_{n=1,3,5,\ldots}^{k} \frac{\cos(n\pi a)\sin(2\pi nt)}{n} \tag{6.5}$$

$$\left\langle \left| \frac{df_k}{dt} \right| \right\rangle = \frac{8}{\pi(1-2a)} \cdot \int_0^1 \left| \sum_{n=1,3,5,\ldots}^{k} \frac{\cos(n\pi a)\sin(2\pi nt)}{n} \right| \mathrm{d}t \tag{6.6}$$

For example, this may be calculated for the first partial sum as:

$$\left| \frac{df_1}{dt} \right| = \frac{8}{\pi(1-2a)} \cdot |\cos(\pi a)\sin(2\pi t)| \tag{6.7}$$

$$\left\langle \left| \frac{df_1}{dt} \right| \right\rangle = \frac{16\cos(\pi a)}{\pi^2(1-2a)} \tag{6.8}$$

The evaluation of the averaging integral is not as neat for the higher order partial sums as for the first, but the principle is the same and the calculation is not difficult and may be approximated numerically with ease. Figure 6.26 plots the mean absolute ramp rate of the various partial sums against the pulse width a. The symmetric case of $a = 0.25$ illustrated in Fig. 6.25 is in the center; the left extreme $a = 0$ corresponds to a triangle wave and the right extreme $a \to 0.5$ approaches a square wave. The constant mean absolute ramp rate of 2 for all the waveforms is also plotted.

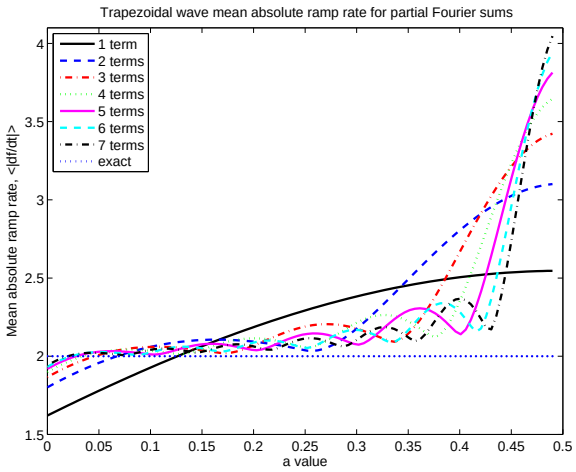

Figure 6.26: The mean absolute ramp rate of the first through seventh partial sums of the unit trapezoidal wave with for a range of pulse widths a. For certain wave shapes and certain partial sums, the mean absolute ramp rate of the partial sum is *larger* than that of the trapezoidal wave itself.

For comparison, the RMS ramp rate for both the original signal and the partial sums may also be calculated and compared. For the original signal, the RMS ramp rate is equal to $\frac{2}{\sqrt{1-2a}}$. For the Fourier series, this may be calculated as:

$$\left\langle \left(\frac{df_k}{dt} \right)^2 \right\rangle = \frac{64}{\pi^2(1-2a)^2} \int_0^1 \left(\sum_{n\,\text{odd}}^{k} \frac{1}{n} \cos(n\pi a) \sin(2\pi n t) \right)^2 \text{dt} \tag{6.9}$$

$$\left\langle \left(\frac{df_k}{dt} \right)^2 \right\rangle = \frac{32}{\pi^2(1-2a)^2} \cdot \sum_{n\,\text{odd}}^{k} \frac{1}{n^2} \cos^2(n\pi a) \tag{6.10}$$

$$\left(\frac{df_k}{dt} \right)_{RMS} = \frac{4\sqrt{2}}{\pi(1-2a)} \cdot \sqrt{\sum_{n\,\text{odd}}^{k} \frac{1}{n^2} \cos^2(n\pi a)} \tag{6.11}$$

The RMS ramp rate for both the partial sums and the original signal is pictured in Fig. 6.27. It can be seen that the RMS ramp rate of the partial sum is always smaller than that of the original trapezoidal wave, although for some wave shapes and partial sums the two values are very similar. While one interpretation of this fact would be that the RMS ramp rate is a more robust indicator of ramping degree, the most correct value would be based on the physical performance parameters of a thermal generating plant (se the discussion of thermal power plants in Chapter 3).

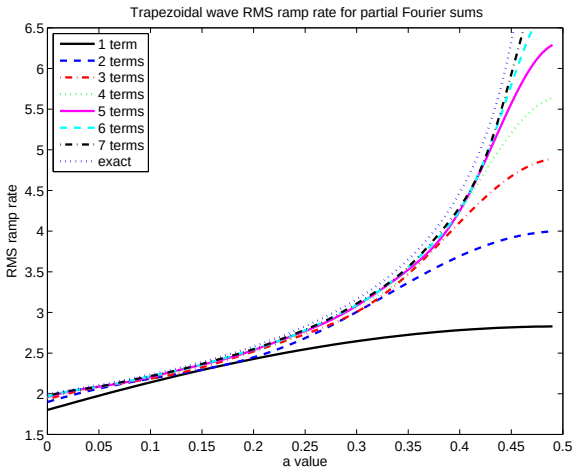

Figure 6.27: The RMS ramp rate of the first through seventh partial sums of the unit trapezoidal wave with for a range of pulse widths a. The RMS ramp rate of the partial sum is always smaller than that of the trapezoidal wave itself, although for certain wave shapes and partial sums, the two values are very similar.

It is clear that depending on the shape of the trapezoid wave and the number of terms in the partial sum (i.e., the "cutoff frequency" of an equivalent filter), the mean absolute ramp rate of the low frequency portion could be higher than, lower than, or equal to that of the original signal. The mechanism for the increase in ramping is that the system is constantly ramping and does not benefit from the flat portions of the curve, while also potentially including a higher peak ramp rate than that in the original waveform. In this sense, the trapezoid wave can behave as a pathological case for the treatment being investigated here of low pass filtering to reduce mean absolute ramp rate—an odd thing for such a simple waveform. Furthermore, it is easy to imagine how a waveform similar in shape to a trapezoid wave could arise from less structured fluctuations in a system with a ramp rate limitation and a deadband.

This example makes clear a drawback of this approach to signal separation into fast and slow components by the use of linear filters. The metric being used is a nonlinear one chosen for physical reasons, and it overlaps with but is not equivalent to a division of a signal into low and high frequency bands with linear filters. The low frequency portion of a signal is not necessarily that which minimizes mean absolute ramp rate.

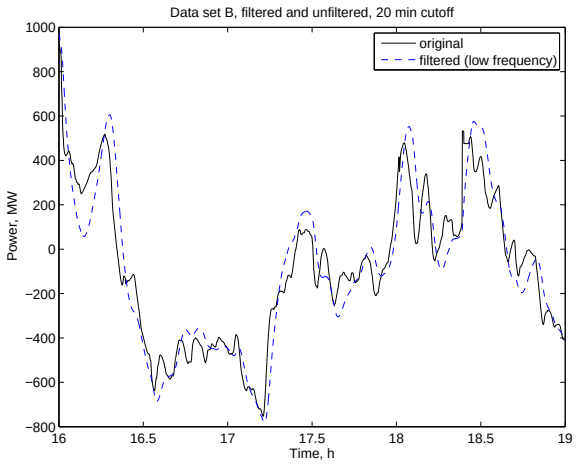

Figure 6.28: A portion of data set B and its low frequency component obtained by use of a Chebyshev type 1 high pass order 3 filter with a 20 minute cutoff. Note the overshoot in the low frequency filtered signal.

6.6 Other Filters

Having discovered this discrepancy in behavior between the two data sets, it is important to investigate whether the undesirable increase in ramping in the low frequency portion of data set B is, in whole or in part, due to the filter class that was selected in Chapter 5. As will be demonstrated in this section, a specially designed filter could likely provide improved ramping for a decreased energy storage requirement as compared to the Chebyshev filters, but it may still require a far larger energy storage unit compared to the ramping improvement it offers.

While investigating alternative filters to use with data set B, it is important to remember that the filter design and energy storage selection is dependent on all the aspects of a control area, including size and type of load, control strategy, and interconnection with other areas. Information about the frequency regulation signal sampled at several locations in the closed loop system, as well as the dynamics of the control logic and the traditional generators, is probably required to produce a good filter design for a given system. In this case, where only two data sets from different locations and times are available, the conclusions to be drawn about filter design are limited to the response of data set B to various classes of filters. This serves to illustrate some of the difficulties which might be encountered when integrating energy storage for frequency regulation, as well as demonstrating that the large difference in results between data sets A and B with the Chebyshev high pass filters described in Sections 5.2 and 6.2 is not solely due to an idiosyncrasy of this filter type.

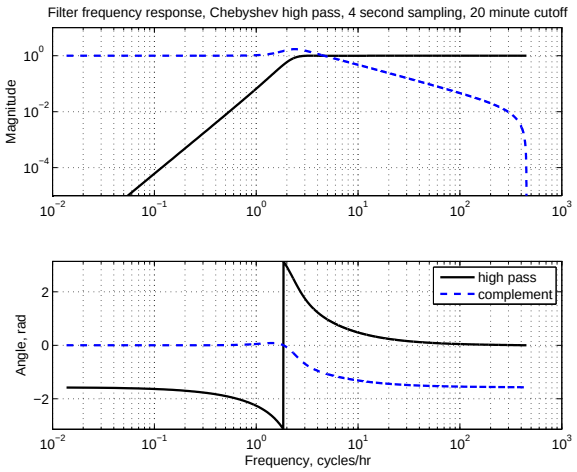

Figure 6.29: Magnitude and phase of the Chebyshev type 1 high pass order 3 filter with a 20 minute cutoff and its complement, both used with data set B. The high pass part has little frequency peaking but the low pass portion exhibits substantial frequency peaking.

First we examine the class of Chebyshev filters that was used in Chapter 5 and Section 6.2 to split the signal into low and high frequency components. As discussed in Section 5.2, this is a class of Chebyshev type I high pass order 3 filters that are characterized by equiripple magnitude in the passband and monotonic (increasing) magnitude in the stopband [80, Section 7.2]. The low frequency portion of the signal was produced by taking the complement of the filter, i.e. subtracting it from one. The frequency response of this type of filter and its complement are shown in Fig. 6.29 for the 20 minute cutoff frequency. While the magnitude response of the high pass filter has a smooth transition near the cutoff frequency, the complementary filter exhibits some magnitude peaking in that region. This magnitude peaking in the low pass portion of the filter could be associated with poor time domain behavior. The step response of the high pass filter is shown in Fig. 6.30. The step response shows a substantial amount of overshoot (about 30%), which could be contributing to poor performance of the filter in the time domain.

One approach is to examine other classes of analog-derived IIR filters. The low pass Chebyshev filter of the same type and order could be used in an attempt to improve the performance of the low frequency portion of the filter specifically, while maintaining the relatively sharp magnitude rolloff of the Chebyshev filter class. The frequency response of the low-pass version of the Chebyshev type I filter of order 3 and its complement are shown in Fig. 6.31. The step response of this filter is in Fig. 6.32. It has a reduced overshoot compared to the high pass filter (closer to 10%).

Another filter option is the Butterworth filter [80, Section 7.2], which is designed for a magnitude which is maximally flat and monotonically decreasing with frequency. Both

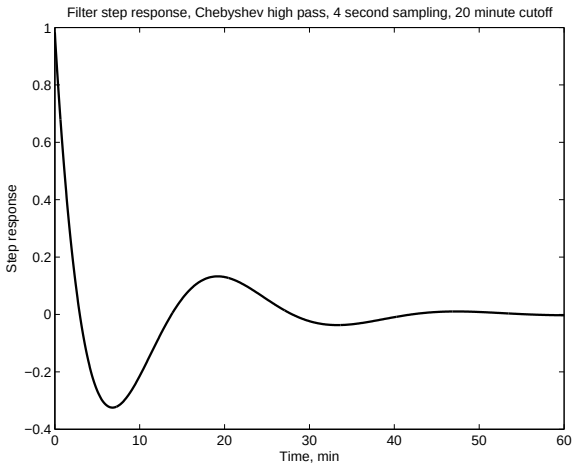

Figure 6.30: Step response of the Chebyshev type 1 high pass order 3 filter with a 20 minute cutoff. There is substantial overshoot in the step response (about 30%).

high-pass and low-pass versions of the Butterworth filter may be designed. These have been converted to discrete-time filters with the use of the same bilinear transform. The frequency responses of the high-pass and low-pass versions, respectively, are shown in Figs. 6.33 and 6.34. The step response of the high pass filter is shown in Fig. 6.35 and the step response of the low pass filter is in Fig. 6.36. The overshoot is about 30% for the high pass filter and less than 10% for the low pass filter.

Of the common classes of analog filters, another which stands out for its potential utility in this case is the Bessel filter [80, Section 7.2], which is designed for a maximally flat group

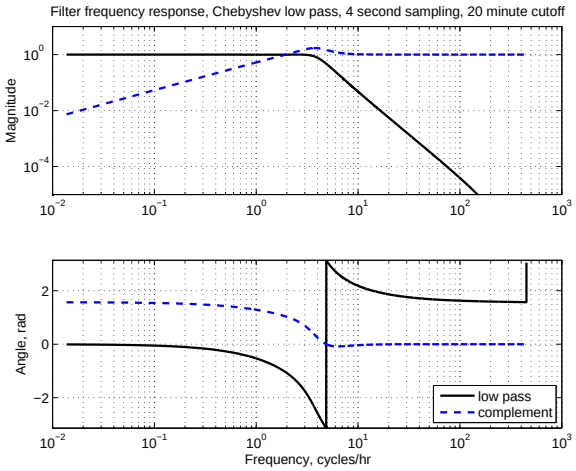

Figure 6.31: Magnitude and phase of the Chebyshev type 1 low pass order 3 filter with a 20 minute cutoff and its complement, both used with data set B.

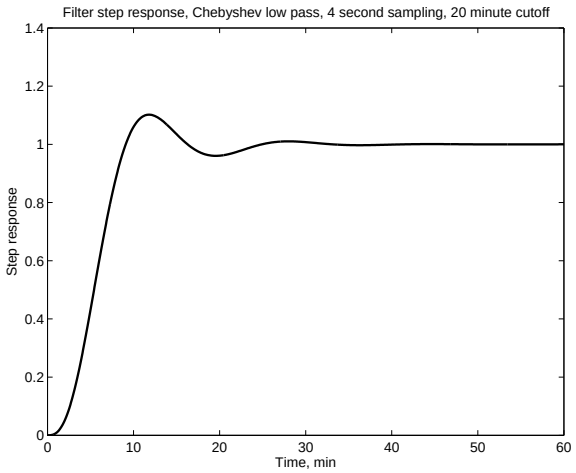

Figure 6.32: Step response of the Chebyshev type 1 low pass order 3 filter with a 20 minute cutoff, used with data set B. The overshoot (about 10%) is reduced compared to the high pass version.

delay, rather than for magnitude characteristics in the frequency domain. This leads to a smooth step response. Because the Bessel filter is designed for phase characteristics, it cannot be converted to a high pass filter using standard high pass frequency transforms; its complement is its only meaningful high pass counterpart. The invariant impulse response method was used to convert the analog prototype Bessel filter to a digital version [80, Section 7.3] Bessel filters of order 3 were used for this investigation. An example magnitude response for the filter with a 20 minute cutoff is shown in Fig. 6.37, and its step response

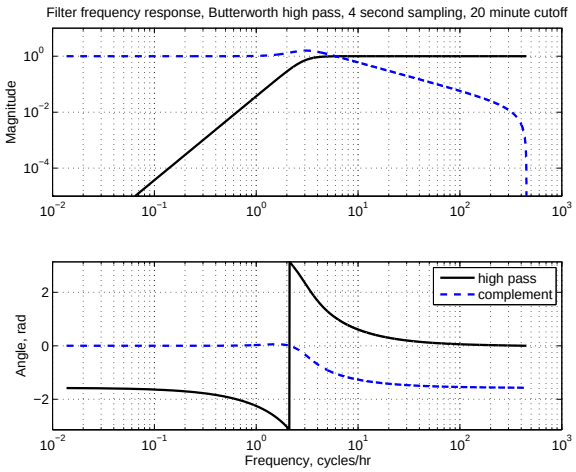

Figure 6.33: Magnitude and phase of the Butterworth high pass order 3 filter with a 20 minute cutoff and its complement, both used with data set B. The high pass part has little frequency peaking but the low pass portion exhibits substantial frequency peaking.

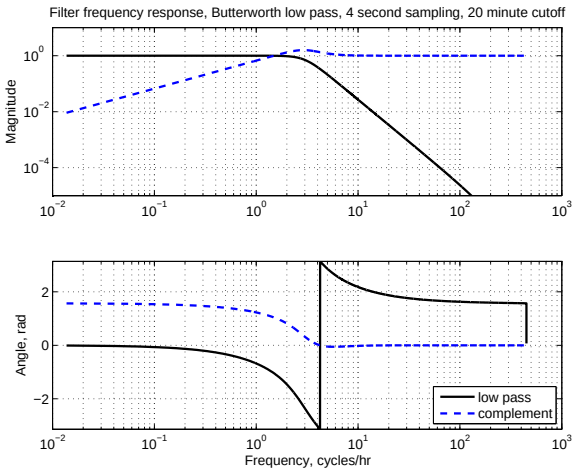

Figure 6.34: Magnitude and phase of the Butterworth low pass order 3 filter with a 20 minute cutoff and its complement, both used with data set B.

is shown in Fig. 6.38. While these filters exhibit slower magnitude roll-off in the frequency domain, the step response demonstrates their superior time domain characteristics, with very little overshoot (less than 1%).

One difficulty that all of the low pass analog-derived filters exhibit is increased effective delay compared to the high pass filters. Figure 6.39 shows a portion of data set B along with its low pass component calculated using the low pass Chebyshev filter and the low pass Bessel filter, both with a 20 minute cutoff. While both filters do a reasonably good job of smoothing the fluctuations present in the signal, both produce an output which is

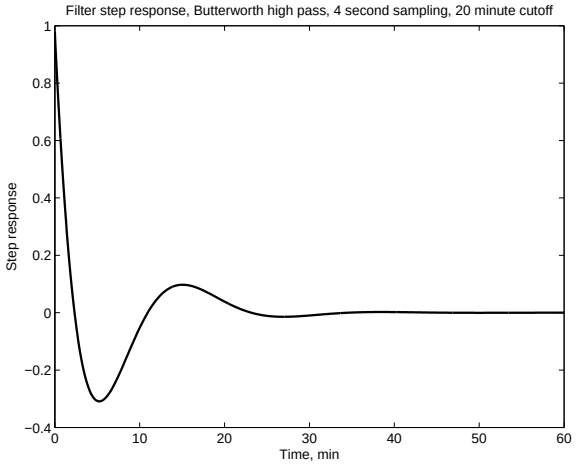

Figure 6.35: Step response of the Butterworth high pass order 3 filter with a 20 minute cutoff. There is substantial overshoot in the step response (about 30%).

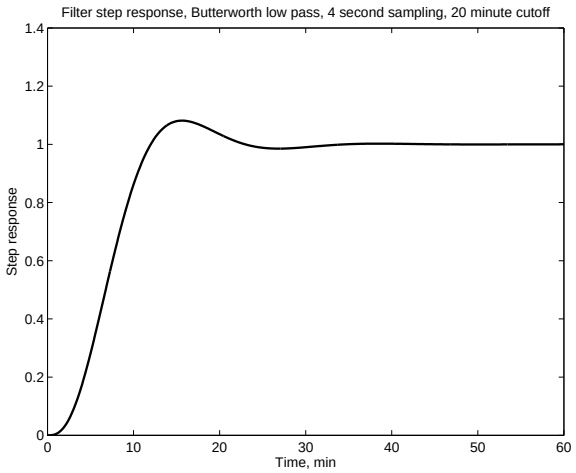

Figure 6.36: Step response of the Butterworth low pass order 3 filter with a 20 minute cutoff. The overshoot (less than 10%) is reduced compared to the high pass version.

substantially delayed, in a visual sense, from the original signal. Because these are nonlinear phase filters, there is no single delay value, but the delay appears to be very approximately 3 minutes for this type of signal in this case. This compares to a delay closer to 90 seconds in the case of the Chebyshev high-pass filter of Fig. 6.28. The result of this delay is an increased energy storage requirement, because the time lag of the low frequency portion leads to a large area between the original signal and its "slow" portion being sent to the thermal generators, and this area represents energy which must be delivered by the storage device.

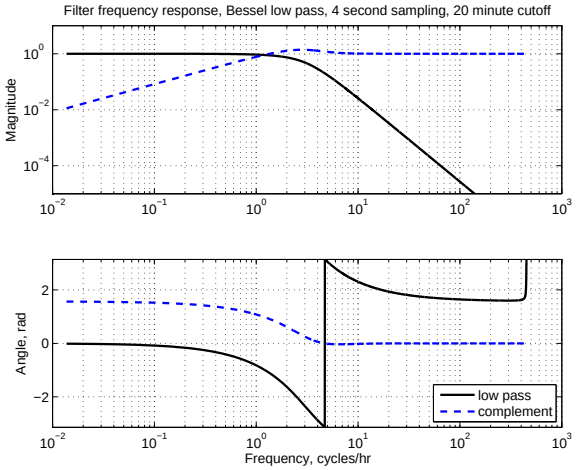

Figure 6.37: Magnitude and phase of the Bessel order 3 low pass filter with a 20 minute cutoff and its complement, both used with data set B. Bessel filters have a slower roll-off but a smoother step response.

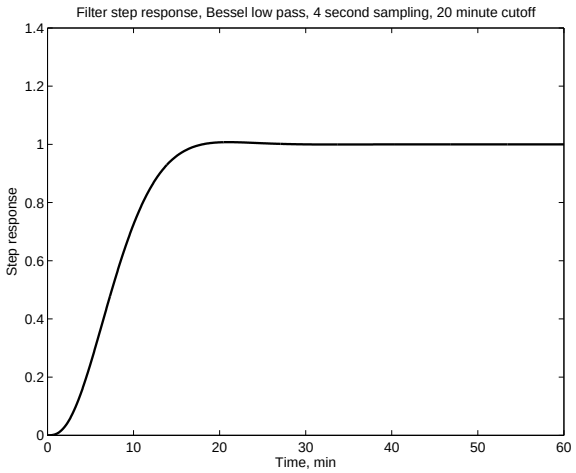

Figure 6.38: Step response of the Bessel order 3 low pass filter with a 20 minute cutoff, used with data set B. Note the very limited overshoot (less than 1%).

To illustrate the performance of these classes of filters, versions with a range of cutoff frequencies are used to process data set B, and the results are compared in terms of the mean absolute ramp rate of the low frequency portion and the maximum energy requirement of the high frequency portion. This graph is included as Fig. 6.40, which is similar to Fig. 6.6 with the addition of the curves from the two new filter classes. The low pass versions of the Chebyshev, Butterworth, and Bessel filters exhibit somewhat lower ramping than the high pass Chebyshev and Butterworth, at the cost of a large energy requirement. Overall, the

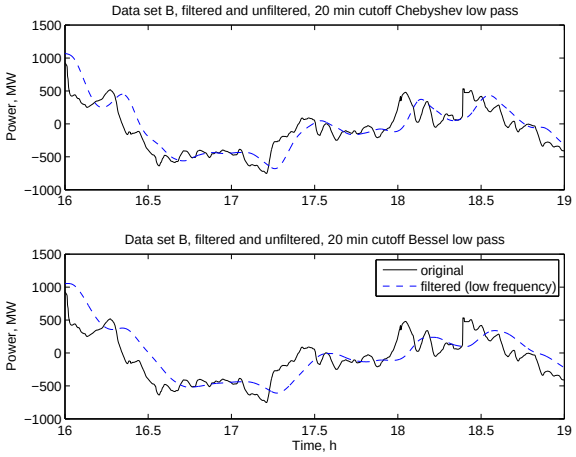

Figure 6.39: The power signal of a portion of data set B, with the low frequency portion as extracted by the low pass Chebyshev type 1 filter (top) and the low pass Bessel filter (bottom). Note the evident delay (very approximately 3 minutes) between the signal and the low frequency component for both filters.

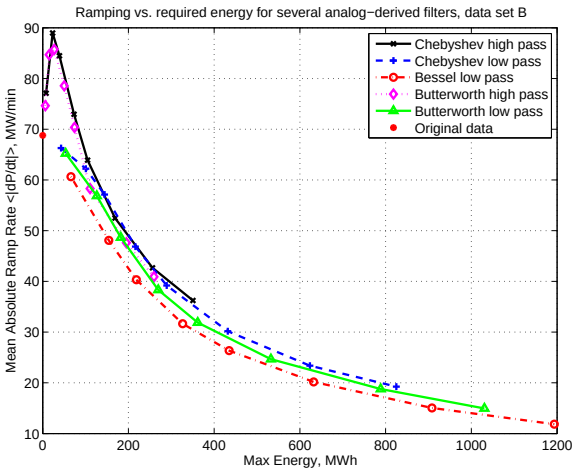

Figure 6.40: Mean absolute ramp rate vs. maximum energy storage requirement for several classes of IIR filters, all of order 3: the Chebyshev type 1 high pass (compare Fig. 6.6); the Chebyshev type 1 low pass; the Butterworth high pass; the Butterworth low pass; and the Bessel low pass. Filter cutoff frequencies in all cases correspond to 3 minutes, 7 minutes, 10 minutes, 15 minutes, 20 minutes, 30 minutes, 45 minutes, and 60 minutes, from upper left to lower right.

performance of these filters does not compare favorably with the impressive results gained with the Chebyshev high pass filter and data set A in Chapter 5.

The last class of filters that will be examined here are classical linear-phase finite impulse response (FIR) filters which approximate unity transmission in the passband and zero transmission in the stopband. MATLAB's `fir1` function was used to produce FIR filters of specified order, using a Hamming window [80, Section 3.2]. The primary advantages here of using FIR filters over analog-derived IIR filters are the linear phase characteristic and the potential for a sharp frequency cutoff with low stopband transmission. The linear phase characteristic means that the signal is not distorted by the filter and results in a smooth step response without overshoot. The sharp frequency cutoff and low stopband transmission ensure that the signal is stripped of its high-frequency components.

A disadvantage of FIR filters is their relatively large delay. Because of the linear phase characteristic, this class of filters exhibits no dispersion and has a constant delay over all frequencies, equal to $(k+1)/2$ samples for a filter of order k. In this case it makes sense to compare the delayed filtered signal to the original signal after it is delayed by the same amount, because this delay can be defined and is not a function of frequency. This delay could present a serious problem for using this type of filter in a real time operation like frequency regulation. If the delay is not too large compared to the dynamics of other parts of the power system, it may be possible to accept a few samples delay in the filter. If the

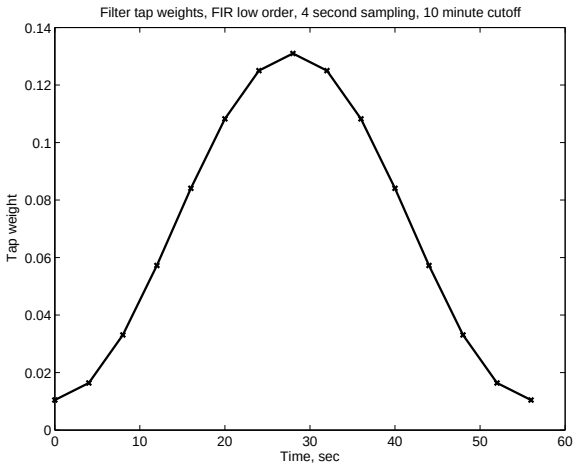

Figure 6.41: Tap weights of FIR filter with order 15 and a cutoff frequency corresponding to 10 minutes.

filter order is high and the delay is large, or if the stability margins of the closed-loop power system are small and sensitive to any delay, the filter will not be suitable for real-time use in frequency regulation. However, an FIR filter of high order may still be useful as a basis for comparison to determine signal properties and as a benchmark for filter performance.

Two FIR filters were investigated here, to provide two different benchmarks. The first is a filter which uses the largest order which still could be consistent with use in a real-time frequency regulation context. A delay of 30 seconds was selected as the longest delay which might be tolerable in this context. A 15 sample filter corresponds to a delay of just over 30 seconds (8 samples).[5] The second FIR filter design was chosen with an order high enough to deliver reasonably close approximations to the ideal cutoff characteristics up to a cutoff frequency corresponding to a 30 minute period. This led to a filter of order 901, which produces a delay of about 30 minutes (451 samples). The goal of this filter is to demonstrate the potential for signal separation with a very good filter. While the delay associated with this filter is far larger than could be tolerated in a frequency regulation system, this filter is useful as a benchmark for good filter design. A very good causal filter design may approach the performance of this long-delay (predictive) filter, but it is unlikely to exceed its performance. The tap weights for the two filter designs for the 10 minute cutoff case are included in Figs. 6.41 and 6.42. The frequency responses of these filters are also included in Figs. 6.43 and 6.44.

[5]Odd orders were chosen to obtain a delay of an integer number of samples.

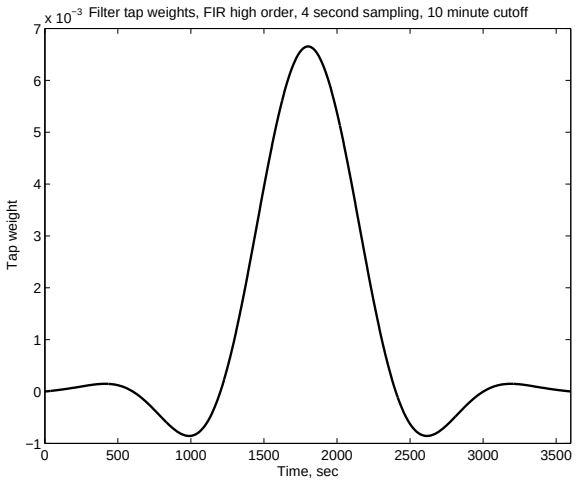

Figure 6.42: Tap weight of FIR filter with order 901 and a cutoff frequency corresponding to 10 minutes.

These FIR filters were used on data set B. Figure 6.45 shows visually the performance of the FIR filter of each order designed for a 10 minute cutoff frequency.[6] The original signal has been delayed to match the delay of the filter. The mean absolute ramp rate of the low frequency portion separated by this class of filters is plotted against the required energy

[6]Note that while MATLAB's design equations allow for the design of a filter of order 15 with a low normalized cutoff frequency such as 0.002, or 30 minutes with a 4 second sampling frequency, the resulting filter does not have a frequency characteristic which approximates this ideal because of the short sampling window as compared to the desired frequency components.

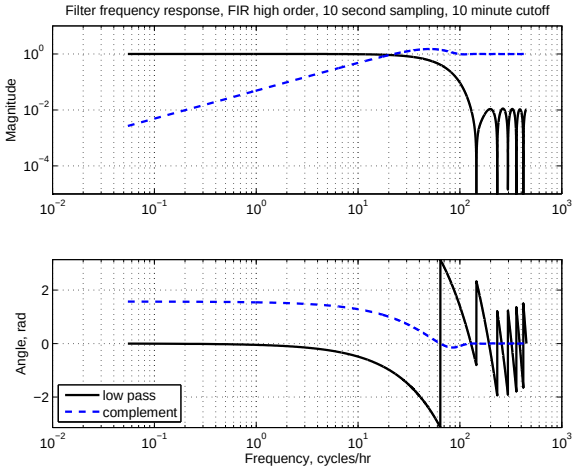

Figure 6.43: Magnitude and phase of tested order 15 FIR filter with design cutoff corresponding to 10 minutes. The effective cutoff frequency is substantially higher because of the low filter order.

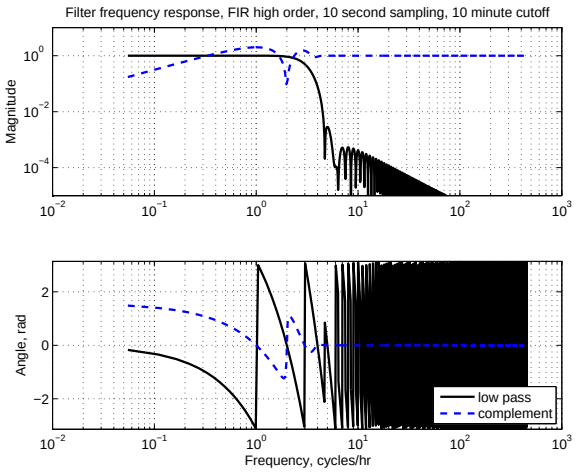

Figure 6.44: Magnitude and phase of tested FIR filter with order 901 and cutoff corresponding to 10 minutes.

storage in Fig. 6.46, along with the original Chebyshev high pass filters, for reference. Because of their low order, the short FIR filters have a limited effect on the ramp rate, and make very little use of the energy storage unit. The long filters provide a reasonable reduction in ramping for an amount of energy storage which is not prohibitively large, but again, this long filter incorporates a 30 minute delay which is not acceptable for real time operation, and so is useful only as a benchmark.

Finally, we leave further filter design as outside the scope of this work. Again, while a sophisticated filter design for a given system is likely to provide better performance than the rather arbitrarily selected Chebyshev filter, what is of greater interest here is the origin and

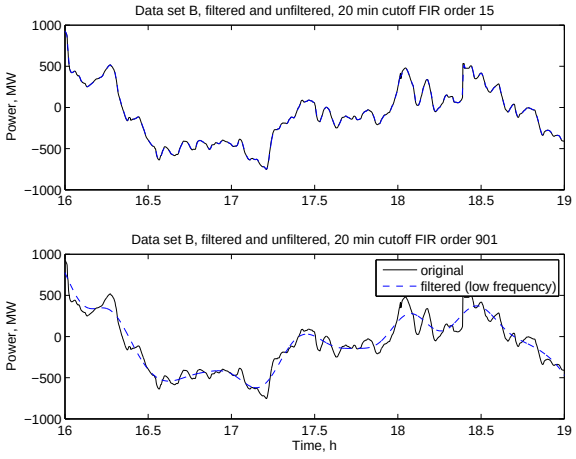

Figure 6.45: The power signal of a portion of data set B, with the low frequency portion as extracted by the FIR order 15 and order 901 filters, with a 10 minute cutoff frequency.

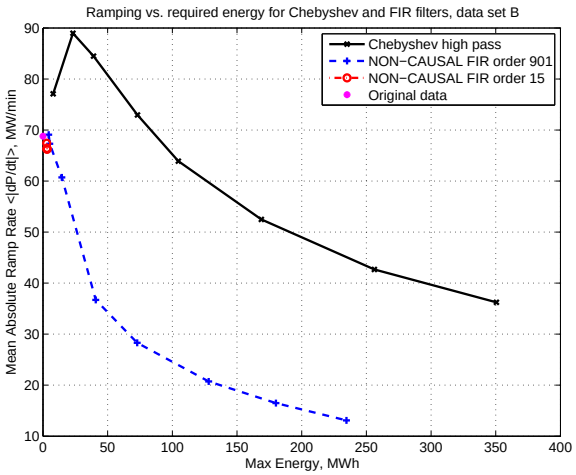

Figure 6.46: Mean absolute ramp rate vs. maximum energy storage requirement for the FIR low pass filters of Figs. 6.41 and 6.42 and the Chebyshev type 1 high pass order 3 for comparison (compare Fig. 6.6). The cutoff frequency is increasing from upper left to lower right. All filters include cutoff frequencies corresponding to 3 minutes, 7 minutes, 10 minutes, 15 minutes, 20 minutes, and 30 minutes. The FIR filters also include 20 second and 1 minute cutoffs, and the Chebyshev filters also include 45 minute and 60 minute cutoffs.

description of the differences between the two data sets. If some sort of separation technique is to be employed in diverse environments for using a combination of energy storage devices and traditional thermal power plants to provide frequency regulation in concert, an understanding of the signal properties which may lead to preferred or unacceptable behavior is more important than finding an improved filter design for a particular case.

6.7 Synthetic Data

One way to investigate whether the characteristics that have been described so far can fully explain the behavioral differences between the two data sets is to produce synthetic data based on the modeling suggested by those characteristics, and assess whether that synthetic data exhibits similar behavior. With this in mind, this section describes the production and evaluation of synthetic data based on the Box-Jenkins autoregressive (AR) models also mentioned in Section 6.4 [7, Section 3.2]. An autoregressive model is chosen as opposed to a moving average or mixed model because the partial autocorrelation falls off to near zero much faster than the autocorrelation does [6, Chapter 9].

The best fit AR models for a variety of model orders were calculated for each of the data sets, using MATLAB's `ar` function and its least-squares forward-backward approach [70,

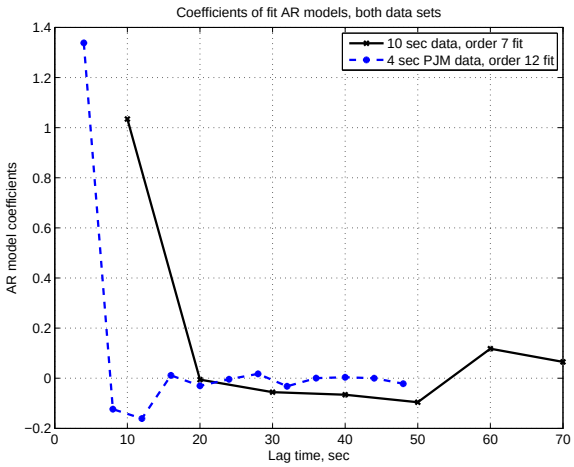

Figure 6.47: Model coefficients (IIR filter tap weights) for autoregressive models for both data sets.

Section 8.5]. The calculated coefficients for a model of data set A of order 7 and for a model of data set B of order 12 are included in Fig. 6.47. The frequency response for the two models, if viewed as IIR filters, are included as Figs. 6.48 and 6.49.

There are several ways to test if a model might be a good representation of the associated data set. One test is to look for structure in the residuals $\hat{\epsilon}_t$, i.e. the differences between the one-step prediction based on the existing data and the model and the following data point. Based on Equation 6.3, the prediction $\hat{y}_t$ is:

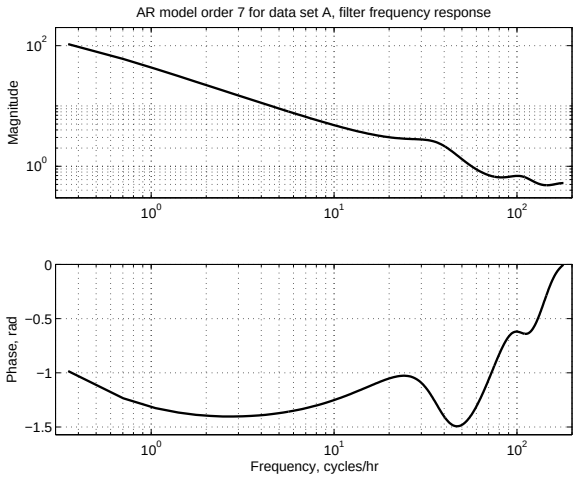

Figure 6.48: Power spectral density of autoregressive model order 7 of data set A, in the sense of an IIR filter.

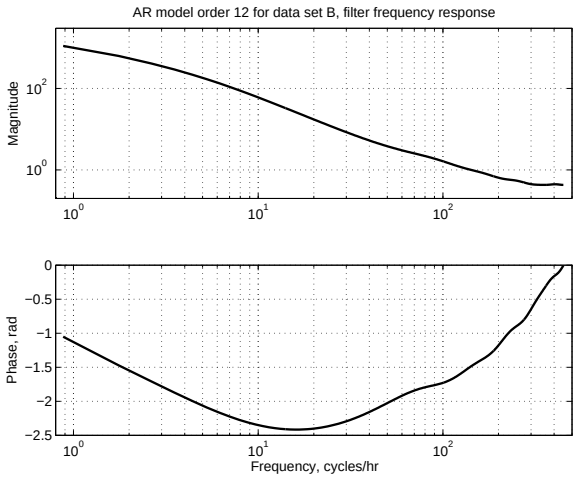

Figure 6.49: Power spectral density of autoregressive model order 12 of data set B, in the sense of an IIR filter.

$$\hat{y}_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} \tag{6.12}$$

$$\hat{\epsilon}_t = y_t - \hat{y}_t \tag{6.13}$$

$$\text{compare } a_t = y_t - \phi_1 y_{t-1} - \phi_2 y_{t-2} - \ldots - \phi_p y_{t-p} \tag{6.14}$$

The residuals should be uncorrelated (white). This indicates that the model is capturing all the structure inherent in the data. The autocorrelation of the residuals from the order 7 model of data set A is shown in Fig. 6.50 and for the order 12 model of data set B is in Fig. 6.51, for each day and week separately. The residuals exhibit small autocorrelation, indicating that the models capture the majority of the structure in the data sets.

Another way to tell whether the model captures the relevant behavior is to generate synthetic data and then compare it to the original data sets. To generate synthetic data with these models, the model coefficients are used as filters on a white noise (IID) input (sometimes called "shocks") with an appropriate distribution. In order to get a good match between the real data and the synthetic data, it is desirable to use a similar distribution for the input shocks as that exhibited by the residuals from the models (see Equation 6.14). While selecting a normal distribution with the same variance as the residuals for the input shocks produces a synthetic data set with some similar characteristics to the original data set, the distribution of the residuals here does not closely mirror a normal distribution. A simple way to select the distribution to be used to generate input shocks is to use the empirical distribution of the residuals for each data set. Samples from the empirical distribution may be drawn by stacking up the residual values and normalizing to create

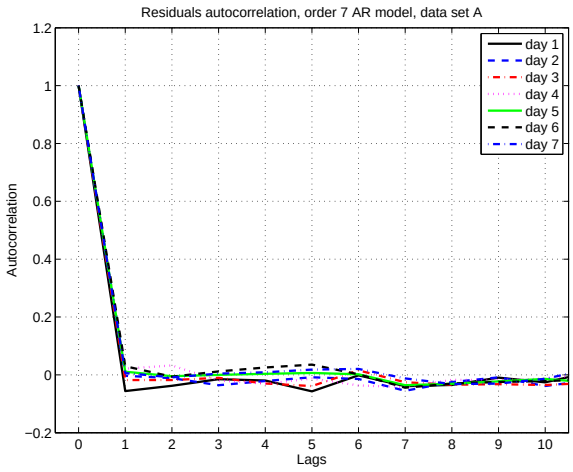

Figure 6.50: Autocorrelation of the residuals from the order 7 model of data set A. Note that the autocorrelation is small for all nonzero lags.

a cumulative distribution function (CDF) and then sampling from a uniform distribution on $[0, 1]$. Samples from the empirical distribution are then selected by finding the residual value corresponding to the cumulative value drawn from $[0, 1]$ [86, Section 2.1]. Because both data sets are large, this CDF is already quite smooth and well-suited to using in reverse. This technique is used to generate a set of input shocks for each of the two data sets, based on the respective residual values.

Once an acceptable set of input shocks has been generated, it is processed by the AR filters described above to generate synthetic data. Short sections of this synthetic data

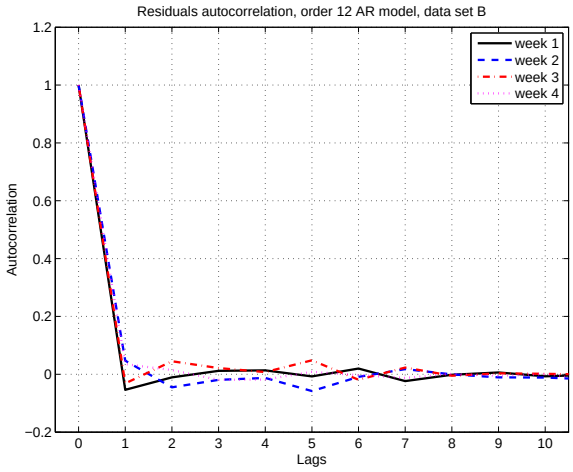

Figure 6.51: Autocorrelation of the residuals from the order 12 model of data set B. Note that the autocorrelation is small for all nonzero lags.

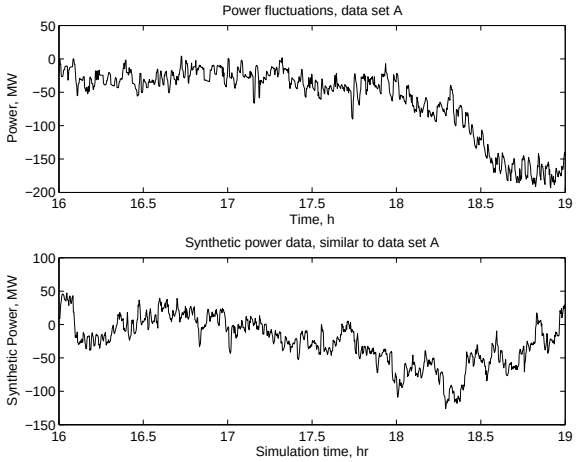

Figure 6.52: Synthetic data mimicking data set A (bottom) along with the original data (top).

are pictured in Figs. 6.52 and 6.53 alongside sections of the corresponding real data for comparison. The power and ramp rate duration curves for the synthetic data and the real data are also compared in Figs. 6.54–6.57. The performance of the synthetic data when filtered using the Chebyshev high-pass filters of Chapter 5 and Section 6.2 in terms of mean absolute ramp rate versus energy storage requirement are in Figs. 6.58 and 6.59. From the figures it is clear that the synthetic data does a pretty good job of mimicking data set A, both visually in the time domain and statistically in the duration domain. The match with data set B is less close, although the synthetic 4 second data based on data set B still displays many characteristics of the original, such as an increase in ramping when filtered with certain Chebyshev high pass filters and a similar ramp rate distribution. In particular, the synthetic data includes a much smaller power range than data set B, indicating that there may be some other effect, such as very slow trends, in data set B which is not captured by the AR(12) model. Any additional effect does not seem to manifest itself in the ramping behavior of the synthetic data compared to the real data. Overall, these synthetic data sets appear to capture most of the important characteristics of the original data sets. This indicates that the partial autocorrelations, upon which the models are based which created the synthetic data, also capture most of the important differences between the two data sets.

One consideration which was not mentioned during the description of the model above is the order of the AR models to be used. In general, the desired model order is the lowest possible value which is large enough to capture the relevant behavior. For data produced by an AR(k) model, the partial autocorrelation should be zero for lags $n > k$, so one might choose the proper model order as the number of nonzero values in the partial autocorrelation function for the time series data. However, this requires a determination of when values of

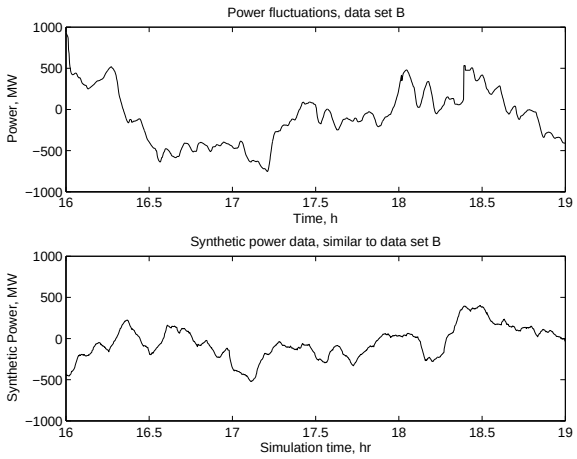

Figure 6.53: Synthetic data mimicking data set B (bottom) along with a portion of data set B (top).

the partial autocorrelation function change from being "significant" to being "near zero." Looking at the partial autocorrelation functions of Figs. 6.22–6.24, we do expect the most useful models to be of an order in the approximate range of 6-15. Based on these graphs, one might choose an order 6 model for data set A and an order 12 model for data set B.

Another important aspect of the model is how it behaves in this application, for example when filtered with the Chebyshev high pass filters used throughout this Chapter. A sequence of AR(k) models were created for each data set, with $1 \leq k \leq 15$, and synthetic data was created from each model (using the residuals from the corresponding model to produce

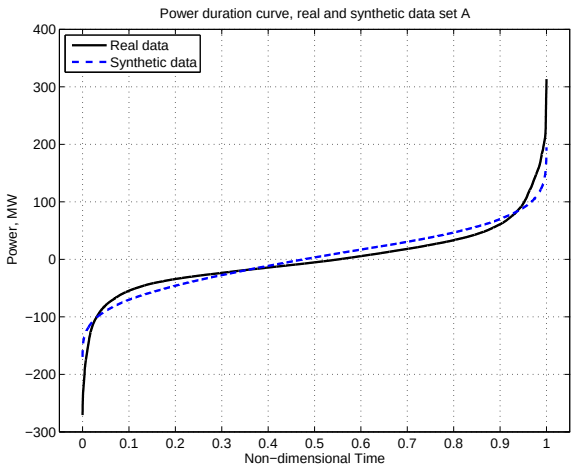

Figure 6.54: Power duration curve for synthetic 10 second data (modeled after data set A), compared to curve for original data. Note that the synthetic data has a very similar power distribution to the real data.

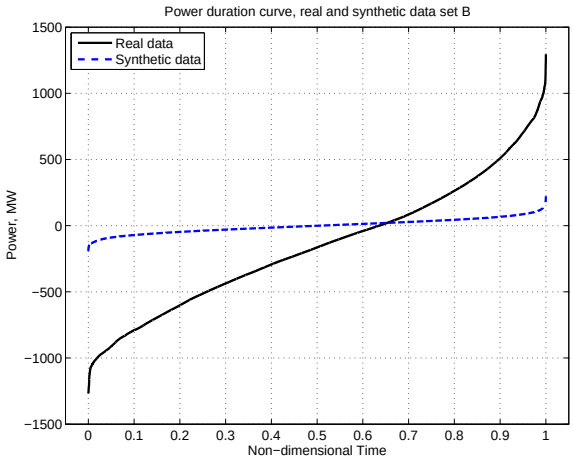

Figure 6.55: Power duration curve for synthetic 4 second data (modeled after data set B), compared to curve for original data. While the curves have generally a similar shape, the real data exhibits a much larger range of power requirements than the synthetic data. This may indicate some slower fluctuation in data set B not captured by the model.

an empirical distribution for the shocks). These synthetic data sets were filtered with the Chebyshev filters with cutoff frequencies at 3 minutes, 7 minutes, and 10 minutes, and the mean absolute ramp rate of the low frequency portion of the data was compared across model order. This is plotted in Fig. 6.60 for the 10 second synthetic data (modeled after data set A) and in Fig. 6.61 for the 4 second synthetic data (modeled after data set B). These graphs show the response of the synthetic data to the filters changing with model order for low orders, then stabilizing around order 7 for the synthetic 10 second data and

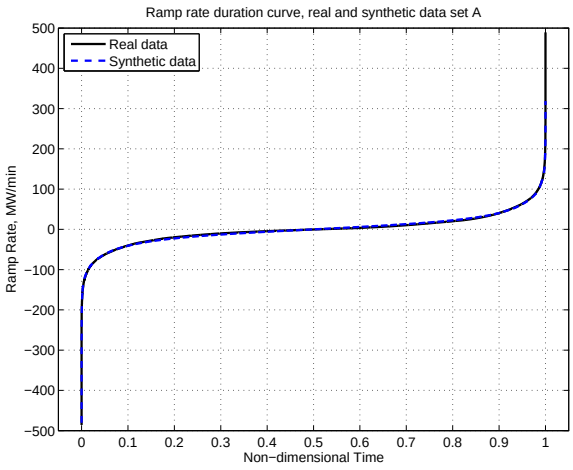

Figure 6.56: Ramp rate duration curve for synthetic 10 second data (modeled after data set A), compared to curve for original data. There is a close match in ramping characteristics between the real and the synthetic data.

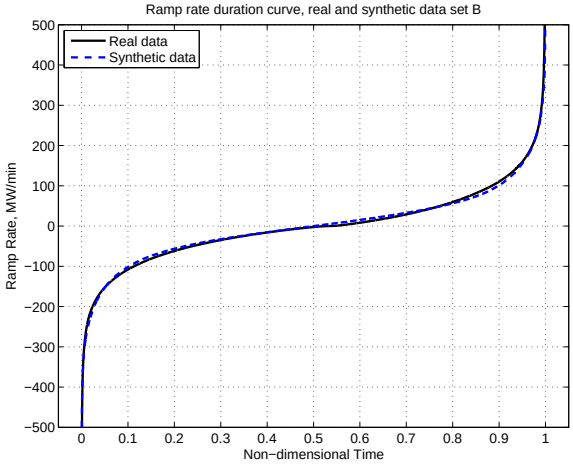

Figure 6.57: Ramp rate duration curve for synthetic 4 second data (modeled after data set B), compared to curve for original data. There is a close match in ramping characteristics between the real and the synthetic data.

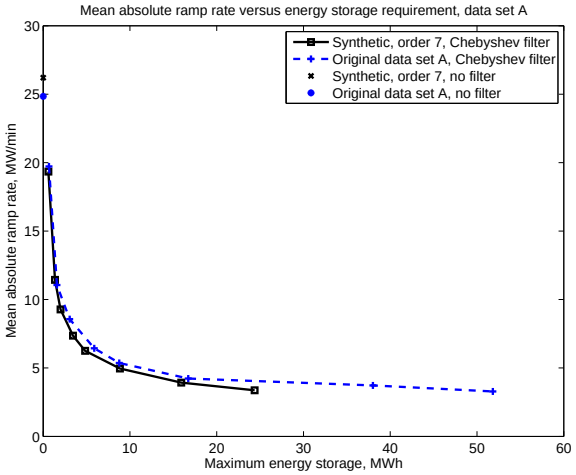

Figure 6.58: Mean absolute ramp rate versus maximum energy storage requirement for synthetic 10 second data (modeled after data set A) using Chebyshev type 1 high pass order 3 filters, compared to original data. Filter cutoff frequencies correspond to 3 minutes, 7 minutes, 10 minutes, 15 minutes, 20 minutes, 30 minutes, 45 minutes, and 60 minutes, from upper left to lower right.

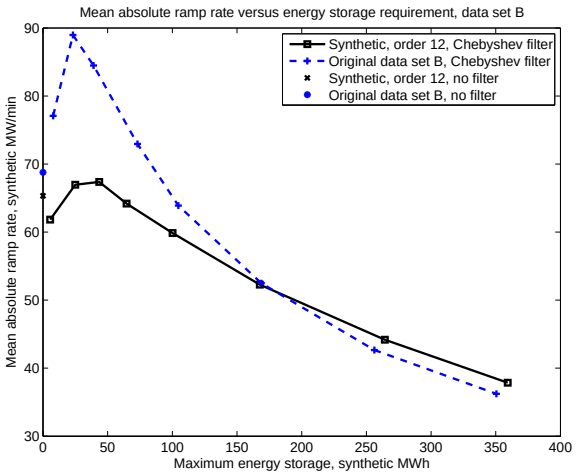

Figure 6.59: Mean absolute ramp rate versus maximum energy storage requirement for synthetic 4 second data (modeled after data set B) using Chebyshev type 1 high pass order 3 filters, compared to original data. Filter cutoff frequencies correspond to 3 minutes, 7 minutes, 10 minutes, 15 minutes, 20 minutes, 30 minutes, 45 minutes, and 60 minutes, from upper left to lower right. Note that while the match between real and synthetic data is only approximate, the synthetic data does exhibit the problematic increase in ramp rate for the low frequency portion of the signal obtained using Chebyshev high pass filters with lower cutoff frequencies.

around order 12 for the synthetic 4 second data. Since this is largely consistent with the indications of the partial autocorrelation graphs, these model orders were used to create the synthetic data sets described above.

6.8 Conclusions

While Chapter 5 described the benefits attainable from using linear filtering to divide the total regulation signal into a high frequency component, to be sent to a small energy storage unit, and a low frequency component, to be sent to traditional thermal generators, this chapter demonstrated the difficulty of this approach when working with certain other types of signals. When the desired signal has a large amount of fast "fuzz," simple filters which remove the high frequency fluctuations are readily able to substantially decrease the total ramping required of the slow thermal generators. By contrast, when the desired regulation signal is dominated by slower but larger changes, these simple filters fall short, and the effects of the nonlinear ramping metric become evident. The characteristics of the frequency regulation signal are influenced by both fixed factors such as system size and interconnection degree and more flexible factors such as control philosophy and controller design. This

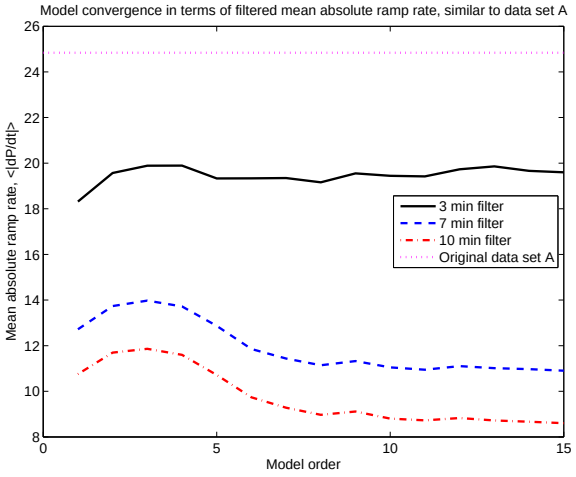

Figure 6.60: Change in mean absolute ramp rate of low frequency portion of synthetic 10 second signal (modeled after data set A) for several Chebyshev high pass order 3 filters as the order of the autoregressive model is increased. Note that the ramp rate has settled to an approximately constant value with models of order 7 and above.

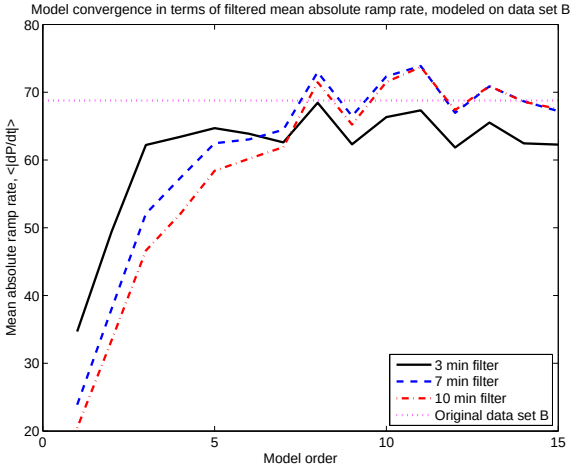

Figure 6.61: Change in mean absolute ramp rate of low frequency portion of synthetic 4 second signal (modeled after data set B) for several Chebyshev high pass order 3 filters as the order of the autoregressive model is increased. Note that the ramp rate has settled to an approximately constant value with models of order 12 and above.

chapter indicates that if a reduction in thermal generator ramping is desired by way of the use of energy storage to perform part of frequency regulation, close attention must be paid to those aspects of the frequency regulation signal which may be changed via design choices. It is no longer necessarily desirable to reduce the degree of small fluctuations as far as possible in the regulation signal, prior to its separation into controls for storage and traditional generators. Finally, even in systems where the native fluctuations tend to be dominated by slower components like data set B, specialized linear or nonlinear filter design may still produce valuable results in terms of separation of the fast and the slow components of the regulation signal for a combination of slow control of the traditional thermal generators with a small energy storage requirement. For particularly difficult signals, a sophisticated real-time ramp minimization algorithm which were based upon the nonlinear ramping metric rather than the linear frequency separation could also possibly provide good results.

Chapter 7

Conclusions

7.1 Contributions of the Thesis

THIS thesis demonstrates the valuable properties of energy storage for frequency regulation on the electric grid. However, fast energy storage is best used on the grid in concert with technologies which can deliver net energy and are more limited in their rates of change of output power than in their total energy delivery. Furthermore, the thesis describes a way to determine how much energy storage would be useful, in the form of energy-duration curves. These curves show how energy storage would be used in a dispatch strategy and hence how much storage is needed. The value of the energy storage to the system is shown by ramp rate duration curves, which illustrate the improvements in ramp rate of thermal generators when energy storage is used to provide part of the frequency regulation for the system. The thermal generators can then operate at a more stable power output, and have higher efficiency and lower wear. Dispatch strategies can be compared with ramp rate and energy duration curves, or with single metrics which reflect the cost aspects of both energy storage and thermal ramping: the total energy capacity required, for storage, and the time average absolute ramp rate, for the thermal units. These curves make clear the trade-offs between different classes of filters for dividing the frequency regulation signal between the thermal generators and the energy storage units.

This work also demonstrates the importance of signal characteristics for operating energy storage as frequency regulation. The frequency regulation power signals are quasi-random, but they do have structure, and that structure is important to the operation of both the energy storage unit and to the strategy used to divide the signal into portion for the storage unit and the traditional resources. Frequency regulation is nominally a zero-energy resource, but the existing frequency regulation signals actually tend to include relatively large amounts of energy delivered and sunk over hours, and are not always zero average power. This means that the raw frequency regulation signals are not suitable for provision by energy storage alone. Furthermore, the structure of the signal determines how much benefit is available using a simple filtering strategy. When there are a lot of the fastest fluctuations, the energy storage produces a lot of benefit by removing them from the signal.

When the signal is slower moving, this benefit is less strong and more difficult to deliver by using a simple filter.

7.2 Limitations and Future Work

This research is limited in certain ways, and some associated investigations could grow out of some of these limitations. One clear limitation is the amount of data that was available for this work. Future work with access to additional data sets, including data sets of the power command signal and power output from the same control area and the same time period, would be helpful to strengthen the conclusions of the work. Additionally, this work intentionally sidesteps the question of the appropriateness or sufficiency or performance of the regulation signal, instead relying on the existing signals from compliant US control areas. It may be that a different control signal would give equivalent or better performance and still be more suitable for provision by energy storage or for a divided signal. An investigation of the creation of the frequency regulation signal might also determine that fast regulation can reduce the total amount of power capacity which is required for regulation [68, 82]. Such work would require detailed models of the dynamic capabilities of both thermal generators and loads as well as of the storage units themselves, and would best be executed with respect to a particular control area. The addition of storage to perform a portion of frequency regulation is likely to be more valuable in some control areas, such as those which have difficulty maintaining good frequency control, than in others. In concert with the investigation of the regulation signals themselves, more sophisticated filter design as well as storage state of charge control would be likely to produce better results than the simple linear filters which were investigated here.

The design of a closed-loop feedback system for energy storage control in grid-connected applications presents several opportunities for future work. In addition to the detailed design of a system of the type described here for a particular control area, the implications of a combined feed-forward and feed-back structure like that used here may be considered in more depth. If the thermal and storage subsystems can be sufficiently separated in frequency, they can be designed separately, as with minor-loop feedback compensation [67]. Additionally, the high-frequency action of the storage unit may be extended to perform near-instantaneous frequency response (the fast increase in power for decreases in frequency due to the disconnection of generation that is currently performed by all generators under speed governor control) in addition to frequency regulation (responding to automatically telemetered setpoints). The addition of well-behaved and fast-acting storage-based frequency response might even change the frequency regulation performance of an area. This

is related to the FAPER concept discussed in [93], where loads respond directly to grid frequency.

Another limitation of this work is the scarcity of information on the costs associated with frequency regulation by traditional generators in the literature. In the past, when there was little alternative to having generators perform frequency regulation, the penalties of frequency regulation by thermal generators mattered less because they were inevitable. Now that energy storage technologies can play a major part in the electric grid, a more complete understanding of the problems with using thermal generators to perform frequency regulation is useful for evaluating the parts of the signal that each type of asset can best respond to. A related point which this work did not address is the revenues and costs associated with frequency regulation by energy storage units. A complete evaluation of the capital and operating costs of a particular technology as well as of the available revenues is important before the implementation of any project to use energy storage for frequency regulation. Additionally, such an analysis could compare the relative merits and costs of different energy storage technologies. For this work, different energy storage technologies were treated as largely equivalent, but for a particular application, only one type will be purchased. This is a rapidly developing field, so the storage technologies which were mentioned in Chapter 4 may be joined by other technologies, and their relative costs are likely to change in the near future. These are all things that are important for the consideration of a particular storage installation in the electric grid as well as points for more general research.

7.3 Discussion

This thesis demonstrates some of the advantages of dividing the burden of LFC between fast energy storage units and slower traditional thermal generators. The use of fast energy storage can lead to substantial reductions in the ramp rate requirement of the thermal units and thus to reduced costs. Energy-duration curves and ramp-rate-duration curves are useful tools for evaluating the performance of dispatch methods. Slope-duration curves in particular may also prove useful in other applications which focus on ramping at different time scales, such as for economic dispatch. The characteristics of the particular frequency regulation signal are key to the benefits of incorporating energy storage.

Appendix A

Generator Models

THE AIM of this appendix is to create an analytical model for a power system which gives good results for timescales from large fractions of a second to tens of minutes, that is, the time scales of interest for frequency regulation. The model includes both conventional resources and fast energy storage, as well as network constraints. This is a decoupled model, with only the real power flows accounted for. Voltage magnitude is assumed to be constant at 1 p.u. throughout the system because of sufficient reactive power compensation. This model may be useful for analysis of the system, such as the evaluation of stability or the formulation of an optimal control strategy; for numerical studies and simulations; and for contributing to understanding by making clear the relationships among the components.

A.1 Power Plant Modeling

Power plants are very large systems with complex dynamics. The time domain of interest is longer than the electrical transients within the machine, but the mechanical interactions of the prime mover and the rotor must be accounted for. The model therefore comprises a very simple synchronous machine model and a turbine model which is primarily linear but which includes plant and control dynamics.

A.1.1 Machine Modeling

This work uses a highly simplified cylindrical-rotor synchronous machine model, with a Thevenin equivalent source dependent on the rotor dynamics [34]. An illustration of this model is pictured in Fig. A.1.

The angle of the voltage source in the equivalent circuit model is the same as the physical rotor (torque) angle in electrical units. The electrical power delivered is determined by the relationship between that angle δ_G and the voltage phase angles on the rest of the system.

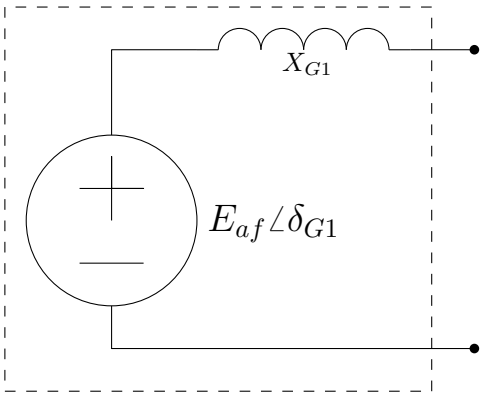

Figure A.1: The simplified equivalent circuit model for the generator.

The physical rotor angle is governed by the generator swing equation, which describes the relationship between generator power, turbine power, and shaft acceleration.[1]

$$J\dot{\omega}_G = P_{therm} - P_{elec} \tag{A.1}$$

That is, the shaft power imbalance is proportional to the rate of change of rotor speed (Newton's Second Law). The generator speed itself is related to the rate of change of torque angle:

$$\dot{\delta}_G = \omega_G - \omega_{sched} \tag{A.2}$$

Together with the generator equivalent circuit model, this is a complete, though highly simplified, description of the generator real power behavior.

A.1.2 Turbine Modeling

There are several possible conventional plant types that might be of interest for fast dynamic modeling for frequency regulation and related grid services. These include combustion turbines, hydraulic turbines, and some steam turbines, especially those close to the margins for energy production which may be scheduled to provide ancillary services instead of providing bulk energy. What is desired is a model which, by choice of parameters, can approximately accommodate any of these turbine types. Then, for a particular system, parameters can

[1] All angles are referenced to a single virtual synchronous reference spinning at scheduled speed $\omega_{sched.}$.

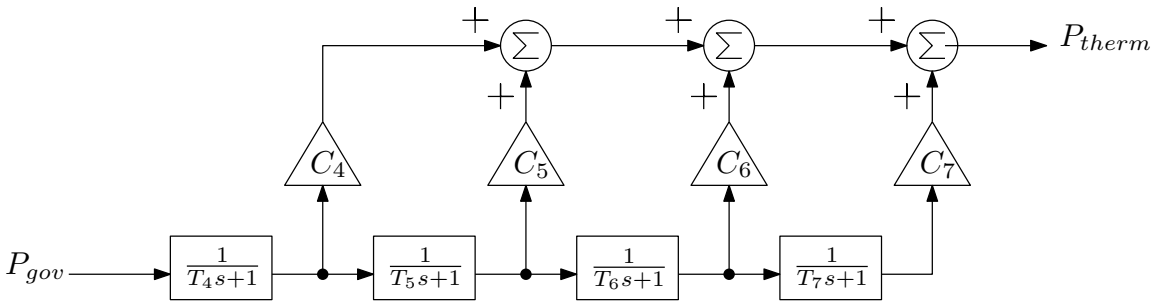

Figure A.2: General steam turbine model, applicable to single or multiple stages with or without reheat, from [45].

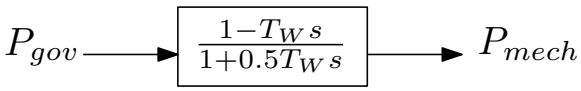

Figure A.3: General hydraulic turbine model, from [3].

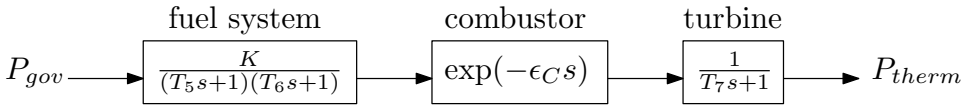

Figure A.4: Simplified combustion turbine model, derived from [89, 40].

be chosen to represent the prime movers in the balancing area, without changes to the fundamental structure of the model.

The literature on modeling prime movers includes several IEEE task force papers and some standardized models [89, 45, 40, 113, 18, 111, 46]. In particular, the models for each of the prime mover types can be combined to fit into a single model type with only a small amount of additional complexity. In [45], a steam turbine model is suggested with a series of single poles, where the result is added after each pole. This general model and is a good candidate to use here. It is shown in Fig. A.2. Also provided in [45] is a model for hydraulic turbines,

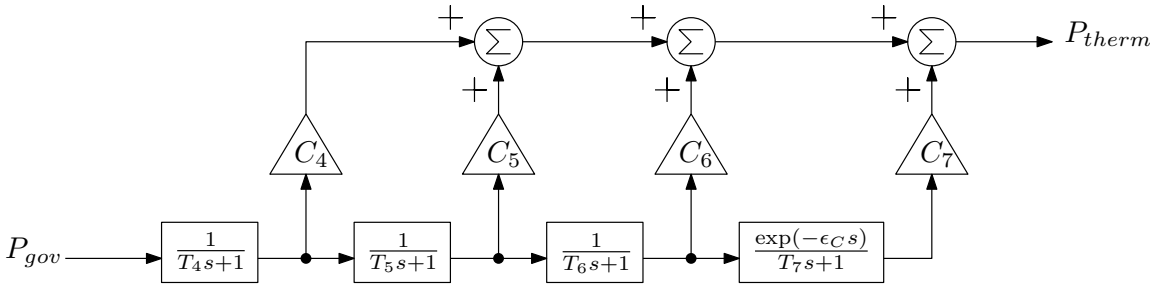

Figure A.5: General turbine model, for which parameters may be chosen to model steam, hydraulic, or combustion turbines as in Figs. A.2, A.3, or A.4.

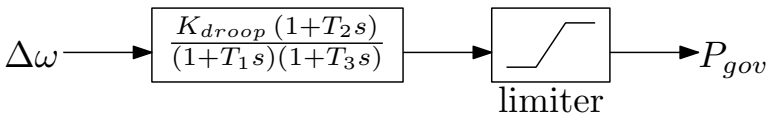

Figure A.6: The governor model which captures most dynamics for steam, hydraulic, and combustion turbines [89, 45].

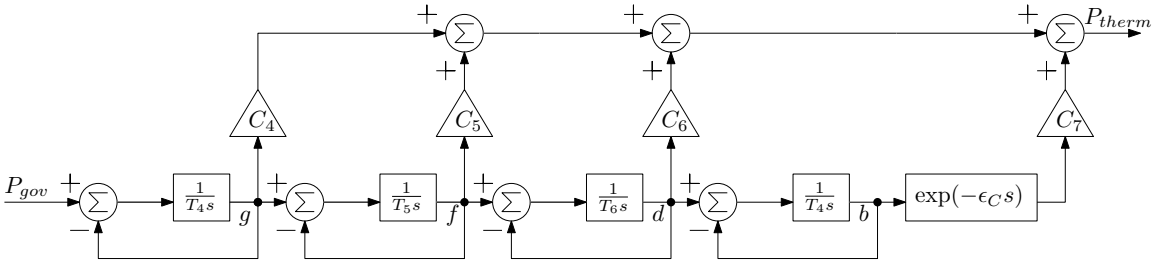

Figure A.7: The turbine model of Fig. A.5 rearranged to facilitate a state-space representation.

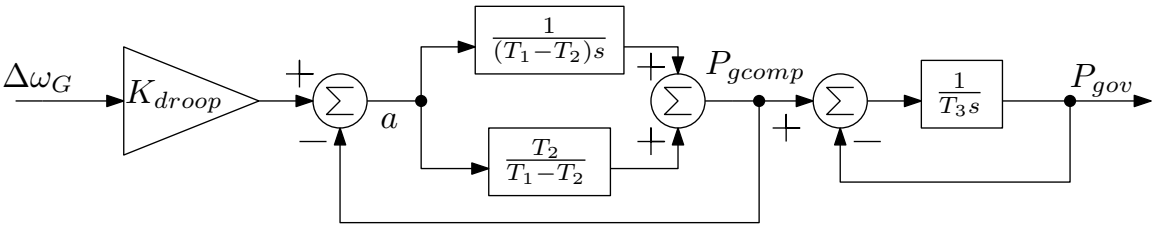

Figure A.8: The governor model of Fig. A.6 rearranged to facilitate a state-space representation.

with a single pole and a single right-half-plane zero, shown in Fig. A.3. This hydroturbine model can be accommodated by the steam turbine model if the gain of one of the forward blocks is negative as the authors suggest.

A general combustion turbine model is presented in [89, 40]. It includes several blocks for which the constants are generally zero, and these are omitted. The resulting combustion turbine model is pictured in Fig. A.4.

This combustion turbine model may be accommodated by the framework of the steam- and hydroturbine model by adding the delay term to one of the blocks of Fig. A.2. The terms not in use can then be set to zero for any particular plant configuration. The resulting general model is diagrammed in Fig. A.5.

All the turbine models above use as an input the power output signal produced by a governor, P_{gov}. The governor model, which relates the speed error to the power signal, can also have the same form for all three turbine types. The steam and hydro turbine governor

model includes two poles, a zero, and a gain factor [45]. The combustion turbine governor model includes one pole, one zero, and a gain factor [89]. Governors may also include limitations on total power or power ramp rate, but these are not included in the analytical model. The general governor model used here is illustrated in Fig. A.6.

The general turbine model and the governor model can be rearranged to be easily expressed as a complete state-space system model. The rearranged turbine model is shown in Fig. A.7, and the rearranged governor model is shown in Fig. A.8.

The state-space model of the governor can then be written out as below:

$$T_3\,\dot{P}_{gov} = P_{gcomp} - P_{gov} \tag{A.3}$$

$$T_1\,\dot{P}_{gcomp} = \frac{K_{droop}\,T_2}{J}(P_{therm} - P_{elec}) + a \tag{A.4}$$

$$\frac{1}{K_{droop}}\,\dot{a} = \frac{1}{J}\left(1 - \frac{T_2}{T_1}\right)(P_{therm} - P_{elec}) + \frac{a}{T_1\,K_{droop}} \tag{A.5}$$

The state space model for the general turbine is then:

$$\dot{P}_{therm} = C_7 \cdot \text{delay}(b, \epsilon_C) + [C_6\,C_5\,C_4] \cdot \begin{bmatrix} d \\ f \\ g \end{bmatrix} \tag{A.6}$$

$$T_4\,\dot{q} = \frac{T_4 + T_W}{T_3}\,(P_{gcomp} - P_{gov}) - q \tag{A.7}$$

$$T_4\,\dot{g} = \frac{-T_W}{T_3}\,(P_{gcomp} - P_{gov}) + q \tag{A.8}$$

$$T_5\,\dot{f} = g - f \tag{A.9}$$

$$T_6\,\dot{d} = f - d \tag{A.10}$$

$$T_7\,\dot{b} = d - b \tag{A.11}$$

The model includes a number of constants which are selected based on the turbine type and parameters. Typical values for the turbine constants, as reported in [89, 45, 40, 58], are listed in Table A.1.

Table A.1: Typical values for the constants in the turbine models, as reported in [89, 45, 40, 58].

Constant	Typical Value			Unit
	Steam (Reheat)	Hydro	Combustion	
T_4	0.1–0.5	0	0.05	sec
T_5	4–11	0.2 – 2	0.2	sec
T_6	0.3–11	0	0	sec
T_7	0–0.5	0	0.2	sec
ϵ_C	0	0	0.01	sec
C_4	0.2–0.3	–2	0	–
C_5	0–0.4	3	0	–
C_6	0–0.5	0	0	–
C_7	0–0.5	0	1	–

A.2 Storage Modeling

The storage unit was modeled as a controllable power source dependent on only internal dynamics (rather than external variables like frequency or phase angle). The storage unit could represent any of a number of fast energy storage technologies, including high-speed flywheels or fast batteries. Rather than focusing on a particular energy storage technology, this work aims to focus on the use of the energy storage in a grid context.

The technologies under consideration for this work are assumed to be fast enough that the dynamics are much faster than the dynamics of the system (see Chapter 4). In particular, electronically interfaced storage technologies are considered here so there are no slow mechanical transients involved in their operation. For the purposes of this model, the energy storage unit is assumed to have a single dominant pole at a frequency slower than the slowest internal dynamics, but still fast compared to the rest of the system. The storage time constant might be expected to be on the order of a single 60 Hz cycle, or a few milliseconds.

The storage unit can be imagined to have a minimum of two state variables: the current state of charge (SOC) and the current delivered power (P_{stor}). Other state variables depend more closely on the technical details of the chosen energy storage unit. The storage unit also has an input signal, the current desired delivered power (P_{stor}^{ref}). In the general case, the dynamics of the storage unit may be expressed as a function of the two state variables, the input signal, and possibly other variables:

$$T_{stor}\dot{P}_{stor} = h(P_{stor},\, P_{stor}^{ref},\, \mathrm{SOC}, \dots) \tag{A.12}$$

If there is assumed to be a first-order delay between the input reference and the change in output, this relationship may be linearized as:

$$T_{stor}\dot{P}_{stor} \approx P_{stor} - P_{stor}^{ref} \tag{A.13}$$

Note that if the energy storage unit is assumed, for simplicity, to be lossless, the relationship between the delivered power and the state of charge is:

$$\dot{\text{SOC}} = -P_{stor} \tag{A.14}$$

A loss model of any complexity may be added to this model, but the general structure will remain the same.

A.3 Load Modeling

The load was modeled as a fixed real power, which is used as an input or disturbance to the system. Load dynamics were not modeled here, although the introduction of some changes in load based on changes in frequency would be possible. Much of the literature on load dynamics relates the load power to voltage in particular, and this type of model does not mesh well with the decoupled power flow which was used here, as voltage effects are completely unmodeled and would contribute significant nonlinearities.

A.4 Network Modeling

The network constraints were modeled using linearized, decoupled power flow. The general power flow equation is:

$$P_{flow,ij} = \frac{V_i\, V_j}{X_{line}} \sin(\delta_{ij}) \tag{A.15}$$

If V is assumed to be a constant and the angle δ_{ij} is assumed to be small, then we have the decoupled linearized power flow equation, expressed in per unit:

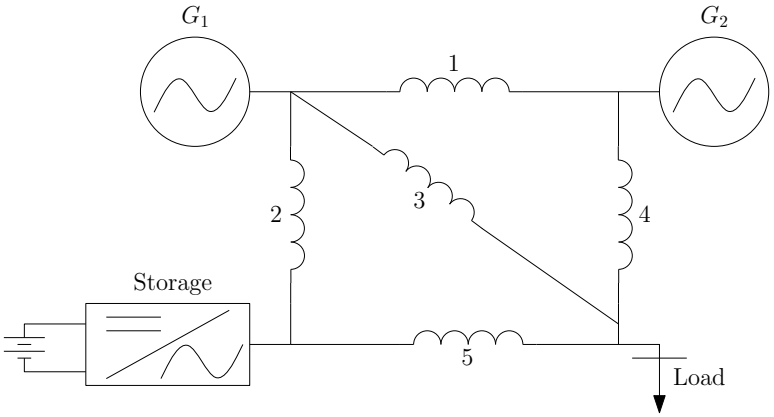

Figure A.9: The toy system used to illustrate the network analysis technique, with two generator nodes, one storage node, and one load node.

$$P_{flow} = \frac{1}{X_{line}} \delta_{ij} \tag{A.16}$$

This becomes an analog to Ohm's law, where we treat power flow as "current," angle (with respect to a fixed reference or infinite bus) as "potential," and have "conductance" $\frac{1}{Xline}$. The angle differences across lines are then "potential differences" which drive "current" (power) flow.

This linear circuit model is of great value because it enables the use of linear algebra techniques to directly solve for the unknown variables, the angles at the load and/or storage nodes, the power flows from each generator, and the line flows. In fact, these variables need not be expressed explicitly in the complete model; the state variables and inputs as well as the constants derived from network topology are sufficient. The state of the "circuit" is fully determined by the power injection at independent-power nodes and the machine torque angle at generator nodes. This strategy closely follows the network analysis in [103].

A toy system model, pictured in Fig. A.9 is used to illustrate the linearization technique. The toy model has two generator nodes, one storage node, and one load node. Five lines connect the four nodes. Although this system is tractable to analyze by hand, the technique is suitable for modeling even very large systems with relative ease. Figure A.9 may be converted to the directed circuit model of Fig. A.10. The direction is necessary to keep track of the direction of power flows within the system.

The linear circuit approach of [103] requires that all current sources be connected between two nodes and all voltage sources be in series with a branch. For this reason, the directed graph of Fig. A.10 must be rearranged slightly. The ground node is used as the source or

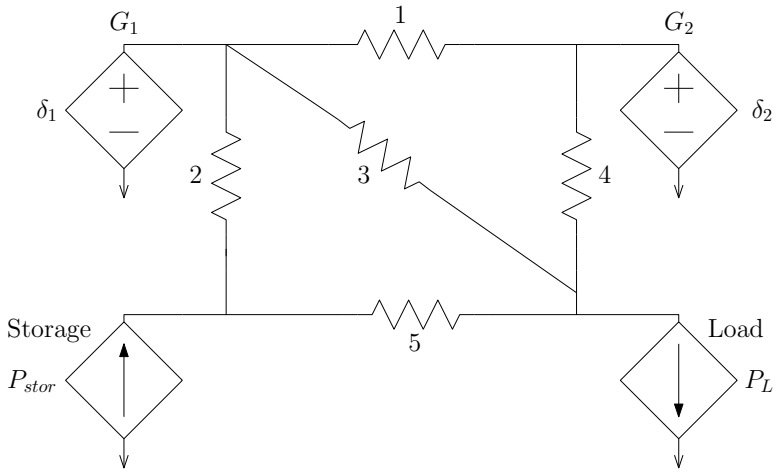

Figure A.10: The circuit analog of the toy system of Fig. A.9. Voltage angle is the driving potential, and power is the resulting flow. Sources are drawn as dependent because their values depend on dynamics or inputs from outside the circuit.

sink for the voltage and current sources. In order to have voltage sources in series with individual branches, the generators are broken out into equal voltage sources, one in series with each line to which they are connected. These are voltages (i.e. angles) relative to ground (i.e. reference), so they may be separated in this fashion. Figure A.11 is equivalent to Fig. A.10 after it has been rearranged in this way. Note also that the series impedance of the generator itself is added to the line impedance in the network.

Once the rearranged circuit model in the style of Fig. A.11 has been created, network techniques may be used to solve the circuit in one step. The topology of the network is captured in the incidence matrix (A), of dimension $m \times n$, where m is the number of lines and n is the number of nodes. The incidence matrix for a directed graph indicates for each node (column) and edge (row), whether the indicated edge leaves the node (-1), enters the node ($+1$), or is not connected to the node (0). The ground node is omitted from the incidence matrix to maintain full rank. For the example system, the incidence matrix is:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 0 \end{bmatrix} \tag{A.17}$$

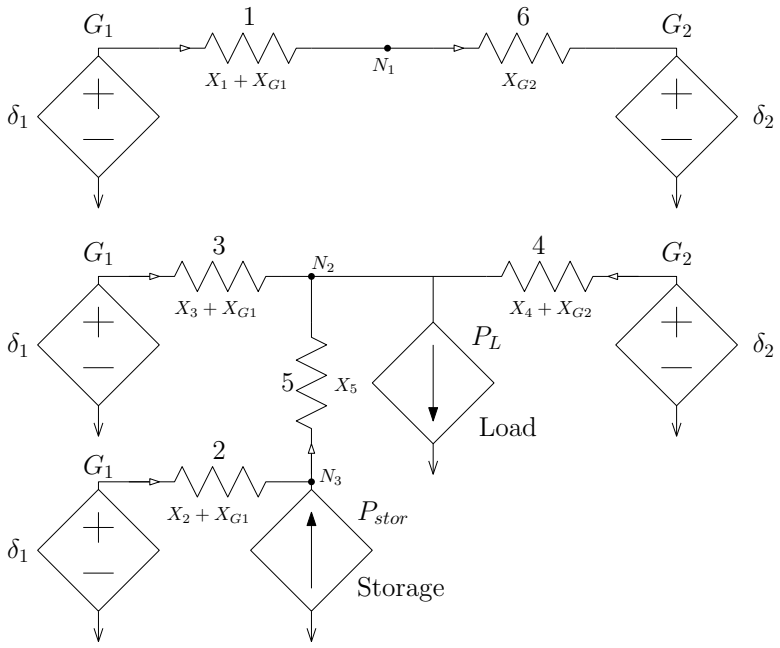

Figure A.11: The circuit model of Fig. A.10 rearranged with voltage sources in series with branches and current sources connected to nodes. This format facilitates the direct solution of the circuit.

The strength of the lines in the system is captured in the conductance matrix ($C = Y_{line}^{-1}$). This is simply an $m \times m$ diagonal matrix with the conductance of each branch (in this case, the inverse of the reactance) along the diagonal. For this simple example, the conductance matrix is:

$$C = Y_{line}^{-1} = \begin{bmatrix} \frac{1}{X_1+X_{G1}} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{X_2+X_{G1}} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{X_3+X_{G1}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{X_4+X_{G2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{X_5} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{X_{G2}} \end{bmatrix} \tag{A.18}$$

The matrices describing the network may now be assembled. The angle differences across branches are the difference between the potential sources and the potential drops.

$$\Delta\delta_{lines} = \delta_G - A \cdot \delta_{nodes} \tag{A.19}$$

The power flows resulting from the angle differences are calculated using the branch conductance matrix:

$$P_{lines} = Y_{line}^{-1} \cdot \Delta\delta_{lines} \tag{A.20}$$

Finally, the conservation of energy means that the net power delivered into any node must be zero. The vector $P_{sources}$ is simply the $n \times 1$ vector of power injections into each node by either loads or storage units.

$$P_{sources} = A^T \cdot P_{lines} \tag{A.21}$$

These three equations may be assembled into a single mixed-unit matrix (K) as follows:

$$\begin{bmatrix} \delta_G \\ P_{sources} \end{bmatrix} = \begin{bmatrix} Y_{line}^{-1} & A \\ A^T & 0 \end{bmatrix} \cdot \begin{bmatrix} P_{lines} \\ \delta_{nodes} \end{bmatrix} = K \cdot \begin{bmatrix} P_{lines} \\ \delta_{nodes} \end{bmatrix} \tag{A.22}$$

The product $K^{-1} \cdot \begin{bmatrix} \delta_G \\ P_{sources} \end{bmatrix}$ is the mixed-unit vector $\begin{bmatrix} P_{lines} \\ \delta_{nodes} \end{bmatrix}$ [2] Because we have assumed that both storage and loads draw or inject power independent of the local power angle (although their behavior does affect the power angle), we do not need to observe the node angles further. For this reason, a reduced K_{red}^{-1} is produced by taking only the first m rows of K^{-1}. Then $K_{red}^{-1} \cdot \begin{bmatrix} \delta_G \\ P_{sources} \end{bmatrix} = P_{lines}$, which is a vector of line flows in the directed graph in numerical order by branch.

Because of the way the generators were separated out in the circuit model to be "potential" sources in series with branches (lines), the power flows need to be grouped to yield the power delivered by each generator. This can be simply done with a generator incidence matrix (G), with as many rows as different generators and as many columns as lines. The generator incidence matrix simply reflects whether the flow in the positive direction of each branch is flowing into the generator (-1), out of the generator ($+1$), or is not flowing through the generator (0). For example, the generator incidence matrix for this toy example would be:

[2] K^{-1} exists as long as A is full rank and Y_{line}^{-1} exists (i.e. as long as there are no unused lines or nodes).

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{A.23}$$

Multiplying the generator incidence matrix times the P_{lines} vector gives the vector of electrical power delivered by each generator, in order of generator number.

$$G \cdot K_{red}^{-1} \cdot \begin{bmatrix} \delta_G \\ P_{sources} \end{bmatrix} = P_G \tag{A.24}$$

In sum, the inputs to the network model are the load and storage power injections (draws), and the generator angles, and the outputs of the network model are the generator power flows. This is in contrast to models of generators as sources of a fixed real power or a fixed power angle, because it explicitly models the dynamics of the power angle of the machine and uses a physical interpretation of the network to determine the power delivered based on that angle. The power delivered then affects the shaft power imbalance, and that is in turn reflected in the acceleration of the rotor and thus the rate of change of the power angle, as described in Section A.1.

A.5 Full Model

Assembling the modeled pieces together, we have a full state-space model of the power system.

$$\dot{\delta}_{sources} = \omega_G - \omega_0^{sched} \tag{A.25}$$

$$J\dot{\omega}_G = P_{therm} - G \cdot K_{red}^{-1} \cdot \begin{bmatrix} \delta_{sources} \\ P_{loads} \\ P_{stor} \end{bmatrix} \tag{A.26}$$

$$T_{stor}\dot{P}_{stor} = h(P_{stor},\, P_{stor}^{ref},\, \mathrm{SOC}) \tag{A.27}$$

$$\dot{\mathrm{SOC}} = -P_{stor} \tag{A.28}$$

$$\dot{P}_{therm} = C_7 \cdot \mathrm{delay}(b, \epsilon_C) + [C_6\, C_5\, C_4] \cdot \begin{bmatrix} d \\ f \\ g \end{bmatrix} \tag{A.29}$$

$$T_4\dot{g} = P_{gov} - g \tag{A.30}$$

$$T_5\dot{f} = g - f \tag{A.31}$$

$$T_6\dot{d} = f - d \tag{A.32}$$

$$T_7\dot{b} = d - b \tag{A.33}$$

$$T_3\dot{P}_{gov} = P_{gcomp} - P_{gov} \tag{A.34}$$

$$T_1\dot{P}_{gcomp} = \frac{K_{droop}T_2}{J}\left(P_{therm} - G \cdot K_{red}^{-1} \cdot \begin{bmatrix} \delta_{sources} \\ P_{loads} \\ P_{stor} \end{bmatrix}\right) + a \tag{A.35}$$

$$\frac{1}{K_{droop}}\dot{a} = \frac{1}{J}\left(1 - \frac{T_2}{T_1}\right)\left(P_{therm} - G \cdot K_{red}^{-1} \cdot \begin{bmatrix} \delta_{sources} \\ P_{loads} \\ P_{stor} \end{bmatrix}\right) + \frac{a}{T_1 K_{droop}} \tag{A.36}$$

A.5.1 Linearized Model

The full model of Section A.5 already contains a number of linearizations and simplifications, for example the linearized decoupled power flow used in the network models and the simplified turbine models, but it also retains two nonlinear terms. As explained in Section A.2, the energy storage model is not explicitly given, but it will be be linearized according to the dominant-pole strategy in that discussion.

The second nonlinear term is the delay term in the turbine model. This term may only be explicitly present (nonzero) in combustion turbine models, where it represents the delay time between fuel injection and fuel combustion. For combustion turbines, this delay is generally very short (about 10 ms) so it can be neglected without much loss of accuracy since timescales of concern are larger fractions of a second.

The fully linearized model follows.

$$\dot{\delta}_{sources} = \omega_G - \omega_0^{sched} \tag{A.37}$$

$$J\dot{\omega}_G = P_{therm} - G \cdot K_{red}^{-1} \cdot \begin{bmatrix} \delta_{sources} \\ P_{loads} \\ P_{stor} \end{bmatrix} \tag{A.38}$$

$$T_{stor}\dot{P}_{stor} = P_{stor} - P_{stor}^{ref} \tag{A.39}$$

$$\dot{P}_{therm} = [C_7\ C_6\ C_5\ C_4] \cdot \begin{bmatrix} b \\ d \\ f \\ g \end{bmatrix} \tag{A.40}$$

$$T_4\dot{g} = P_{gov} - g \tag{A.41}$$

$$T_5\dot{f} = g - f \tag{A.42}$$

$$T_6\dot{d} = f - d \tag{A.43}$$

$$T_7\dot{b} = d - b \tag{A.44}$$

$$T_3\dot{P_{gov}} = P_{gcomp} - P_{gov} \tag{A.45}$$

$$T_1\dot{P_{gcomp}} = \frac{K_{droop}T_2}{J}\left(P_{therm} - G \cdot K_{red}^{-1} \cdot \begin{bmatrix} \delta_{sources} \\ P_{loads} \\ P_{stor} \end{bmatrix}\right) + a \tag{A.46}$$

$$\frac{1}{K_{droop}}\dot{a} = \frac{1}{J}\left(1 - \frac{T_2}{T_1}\right)\left(P_{therm} - G \cdot K_{red}^{-1} \cdot \begin{bmatrix} \delta_{sources} \\ P_{loads} \\ P_{stor} \end{bmatrix}\right) + \frac{a}{T_1 K_{droop}} \tag{A.47}$$

A.6 Discussion

The model developed here is applicable to the study of fast energy storage as a grid resource for frequency regulation. The model captures the relevant dynamics of the generators and their prime movers, of the network, and of the storage itself. While load dynamics are not included, their effect is not clear because the relationship between frequency and load power is difficult to determine.

When looking at the model, the different time scales of the subsystems become apparent. While the energy storage unit operates at timescales of a single 60 Hz cycle or less, portions of the power turbine have time constants of several seconds or more. This can facilitate analysis by allowing the formal separation of the model into a fast and a slow subsystem, which

interact only in a quasi-steady-state sense. For a particular system with fixed parameters, this timescale separation could be especially helpful.

The power plants in the model are under closed-loop control based on governor response, and a system study would be likely to include closed-loop control for the energy storage unit(s) as well. This control could react to local frequency, to its own internal states (especially state of charge) and possibly to load or tieline power (insofar as that is being monitored). The energy storage in the model would then react to the changes in load power which are input to the system.

One way to select a feedback control scheme would be to produce cost functions based on system behavior and choose an optimal control strategy based on those costs. The costs would likely be related to system performance such as frequency error and to control effort in the form of total storage capacity or rate of change of thermal unit setpoints. If the costs can be approximated by quadratic functions, then linear optimal control theory may be used to solve for the best linear controller. If, however, a quadratic cost function does not capture the interesting features of the problem, other control must be chosen, either linear or nonlinear.

As discussed above, energy storage can be appropriate for frequency regulation, which is a secondary control function acting at time scales longer than a few seconds. Another fast control task on the power grid is frequency response, which is the change in power that traditional power plants generate as the frequency changes. This effect can be seen in the model as the droop constant of the generators. This faster primary response may also be an appropriate application for energy storage, although it has not been applied to this area before. Energy storage units respond quickly enough to participate in control at the faster timescale.

Appendix B

Partial Autocorrelation Development

PARTIAL autocorrelation is the linear relationship between successive points in a time series with the effect of the intervening points removed. For example, given a sequence of time series values x_0, x_1, , $\ldots x_t$, if point x_i depends on point x_{i-1}, then the time series will also exhibit a dependency between x_i and x_{i-2} because point x_i depends on point x_{i-1} and point x_{i-1} depends on point x_{i-2}. Partial autocorrelation describes the direct relationship between points x_i and x_{i-2} with the linear effect of point x_{i-1} removed. This also generalizes to lags higher than 2, in which case the linear effects of all intermediate points are removed. The following derivation is adapted from [7].

The calculation of the partial autocorrelation at lag k is equivalent to finding the best-fit model coefficients for an autoregressive model of order k. An autoregressive model has the form:

$$x_t = \phi_1 \, x_{t-1} + \phi_2 \, x_{t-2} + \cdots + \phi_k \, x_{t-k} + a_t \tag{B.1}$$

where a_t is a independent random shock term (input noise) and the ϕ_i coefficients describe the model.

For any k, the sample autocorrelation at lag k, r_k, is:

$$r_k = \frac{\sum_0^{n-k}(x_t - \bar{x})(x_{t+k} - \bar{x})}{\sum_0^n (x_t - \bar{x})^2} \tag{B.2}$$

In this case, $\bar{x}$ represents the sample mean. For simplicity, assume that the sample mean $\bar{x} = 0$. Then, a model of order j can be fit to the data, with the $\hat{\phi}_i$ terms indicating that they are an approximate model fit:

$$x_t = \hat{\phi}_1 \, x_{t-1} + \hat{\phi}_2 \, x_{t-2} + \cdots + \hat{\phi}_j \, x_{t-j} + a_t \tag{B.3}$$

Multiplying through by x_{t-k}, for an integer value of k between 1 and j:

$$x_t\, x_{t-k} = \hat{\phi}_1\, x_{t-1}\, x_{t-k} + \hat{\phi}_2\, x_{t-2}\, x_{t-k} + \cdots + \hat{\phi}_j\, x_{t-j}\, x_{t-k} + a_t\, x_{t-k} \tag{B.4}$$

Equation B.4 may be summed over all the values of t to yield:

$$\sum_{t=0}^{n-k} x_t\, x_{t-k} = \hat{\phi}_1 \sum_{t=0}^{n-k} x_{t-1}\, x_{t-k} + \hat{\phi}_2 \sum_{t=0}^{n-k} x_{t-2}\, x_{t-k} + \cdots + \hat{\phi}_j \sum_{t=0}^{n-k} x_{t-j}\, x_{t-k} + \sum_{t=0}^{n-k} a_t\, x_{t-k} \tag{B.5}$$

The term $\sum_{t=0}^{n-k} a_t\, x_{t-k}$ vanishes because both a_t and x_t are assumed to be zero mean, and since a_t is random and independent of the sequence, their cross-correlation is zero. If Equation B.5 is normalized by the sample variance ($\sum_0^n x_t^2$), then this yields a linear relationship among the autocorrelations:

$$r_k = \hat{\phi}_{j1}\, r_{k-1} + \hat{\phi}_{j2}\, r_{k-2} + \cdots + \hat{\phi}_{jj}\, r_{k-j} \tag{B.6}$$

These are known as the Yule-Walker equations and can be rearranged as follows:

$$\begin{array}{lcllllll} r_1 & = & \hat{\phi}_{j1} & + & \hat{\phi}_{j2}\, r_1 & + \ \ldots \ + & \hat{\phi}_{jj}\, r_{j-1}, & k=1 \\ r_2 & = & \hat{\phi}_{j1}\, r_1 & + & \hat{\phi}_{j2} & + \ \ldots \ + & \hat{\phi}_{jj}\, r_{j-2}, & k=2 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots & \\ r_j & = & \hat{\phi}_{j1}\, r_{j-1} & + & \hat{\phi}_{j2}\, r_{j-2} & + \ \ldots \ + & \hat{\phi}_{jj}, & k=j \end{array} \tag{B.7}$$

or equivalently:

$$\begin{bmatrix} 1 & r_1 & r_2 & \ldots & r_{k-1} \\ r_1 & 1 & r_1 & \ldots & r_{k-2} \\ r_2 & r_1 & 1 & \ldots & r_{k-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{k-1} & r_{k-2} & r_{k-3} & \ldots & 1 \end{bmatrix} \cdot \begin{bmatrix} \hat{\phi}_{k1} \\ \hat{\phi}_{k2} \\ \hat{\phi}_{k3} \\ \vdots \\ \hat{\phi}_{kk} \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_k \end{bmatrix} \tag{B.8}$$

Then, the sample partial autocorrelation at lag k is $\hat{\phi}_{kk}$, which is also the linear relationship between a point and the point k sample periods earlier, with the effects of the intervening points removed.

Appendix C

MATLAB *Scripts*

This appendix includes the MATLAB R2010b code which was used to perform the signal processing in this thesis. The code which builds the plots themselves is omitted, as are some top-level files whose main purpose is to run subfiles.

C.1 Duration Curves and Basic Processing

C.1.1 Example Top Level File

`thesis_makefigs_ch_load_durcurves_2jun11_noplots.m`

Includes some functions which are used throughout the code and defined in top-level files for each part.

```
%% Olivia Leitermann  MIT LEES  2 jun 11
%% script to make the plots for thesis chapter on loads and duration curves

clear all;
close all;

set(0,'DefaultAxesColorOrder',[0 0 0],...
      'DefaultAxesLineStyleOrder','-|--|-.|:', ...
        'DefaultAxesFontSize', 12);

mitred = [153, 51, 51]/256;

% make sure I have a consistent ramp rate and integral with the right
    sample spacing
rr10sec = inline('[0;␣(vect(2:end)␣-␣vect(1:end-1)).*6];', 'vect');
int10sec = inline('-cumtrapz(vect)./360', 'vect');

rr5min = inline('[0;␣(vect(2:end)␣-␣vect(1:end-1))./5];', 'vect');
int5min = inline('-cumtrapz(vect)./12', 'vect');

rr4sec = inline('[0;␣(vect(2:end)␣-␣vect(1:end-1)).*15];', 'vect');
```

```
int4sec = inline('-cumtrapz(vect)./900', 'vect');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Mystery BA data
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% script to take care of loading, analyzing, and plotting 10s data
thesis_makefigs_ch_loaddur_subfile_10sdata_20jul11

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PJM Data %%%%%%%%%%%%%%%%%%%%%%%%%

%% script to take care of loading, analyzing, and plotting pjm reg data
thesis_makefigs_ch_loaddur_subfile_pjmdata_20jul11


'done'
```

C.1.2 Processing Data Set A

thesis_makefigs_ch_loaddur_subfile_10sdata_20jul11_noplots.m

```
%% Olivia Leitermann  separated 20 Jul 11  MIT LEES
%% Script to do 10-sec data portion of data loading, analysis, plotting

loaddata10sec = load('NEISO_AGC_NEload.csv');
% columns are 10-sec (!) samples of NE load on 10 non-consecutive days,
% midnight to midnight.  first 3 cols are sundays 23 mar 08, 28 sept 08, 30
% aug 09, followed by 2 saturdays, 11 apr 09, 2 jan 10, followed by
% mon-friday 16 jun 08, 26 may 09, 29 jul 09, 1 oct 09, 22 feb 08.  Total
% system load MW

% use a 5-point median filter to get rid of spikes
flddata10sec = medfilt1(loaddata10sec, 5);

f7lddata10sec = medfilt1(loaddata10sec, 7);
f3lddata10sec = medfilt1(loaddata10sec, 3);

% sift out days if I need to...
pickdays = [1 2 3 4 7 8 9];
numdays10sec = numel(pickdays);

sload10sec = loaddata10sec(:,pickdays);
sfload10sec = flddata10sec(:,pickdays);

sf7load10sec = f7lddata10sec(:,pickdays);
sf3load10sec = f3lddata10sec(:,pickdays);
```

```
numpts10s = numel(sload10sec);
time10sec = linspace(0, (numpts10s - 1)/360, numpts10s);

%% Plot original data with anomalies indicated

orignumpts10s = numel(loaddata10sec);
origloaddata10svec = reshape(loaddata10sec, orignumpts10s, 1);
origtime10sec = linspace(0, (orignumpts10s-1)/360, orignumpts10s);

%% find ramp rate of original 10-sec data with and without median filter

init10s = zeros(size(sload10sec));

sloadrr10sec = init10s;
sfloadrr10sec = init10s;
sf3loadrr10sec = init10s;
sf7loadrr10sec = init10s;

for index = 1:numdays10sec

    sloadrr10sec(:,index) = rr10sec(sload10sec(:,index));
    sfloadrr10sec(:,index) = rr10sec(sfload10sec(:,index));
    sf3loadrr10sec(:,index) = rr10sec(sf3load10sec(:,index));
    sf7loadrr10sec(:,index) = rr10sec(sf7load10sec(:,index));
end

sload10secvec = reshape(sload10sec, [], 1);
sfload10secvec = reshape(sfload10sec, [], 1);

sf3load10secvec = reshape(sf3load10sec, [], 1);
sf7load10secvec = reshape(sf7load10sec, [], 1);

sloadrr10secvec = reshape(sloadrr10sec, [], 1);
sfloadrr10secvec = reshape(sfloadrr10sec, [], 1);

sf3loadrr10secvec = reshape(sf3loadrr10sec, [], 1);
sf7loadrr10secvec = reshape(sf7loadrr10sec, [], 1);

sload10secdur = sort(sload10secvec);
sfload10secdur = sort(sfload10secvec);
sloadrr10secdur = sort(sloadrr10secvec);
sfloadrr10secdur = sort(sfloadrr10secvec);

%% Plot difference between median filtered and raw data

meddiff10svec = sload10secvec - sfload10secvec;

med53diff10svec = sfload10secvec - sf3load10secvec;
med57diff10svec = sfload10secvec - sf7load10secvec;
```

```
%% Find the bulk energy of the curves and subtract to get fuzz only

window90 = 90*6+1; % number of samples to make up 90 min, want an odd number
filtvec90 = ones(window90, 1)./window90;

init = zeros(size(sload10sec));

sfloadavg90 = init;
sfleftover90 = init;

sloadavg90 = init;
sleftover90 = init;

for index = 1:numdays10sec

    offset = sfload10sec(1,index) - sfload10sec(end,index);
    [waitforit, zi] = filter(filtvec90,1, (sfload10sec(:,index) + offset));
        % calculate appropriate initial conditions: jigger so it ends where
        the next one starts
    working = filter(filtvec90,1,sfload10sec(:,index), zi); % use ICs from
        as if two identical days had been appended
    % BUT need to slide it so that the data points are centered in the
    % boxcars
    chunk = working(1:(window90+1)/2)-offset;
    working = [working((window90+3)/2:end); chunk];
    sfloadavg90(:,index) = working;
    sfleftover90(:,index) = sfload10sec(:,index) - working;

    [waitforit, zi] = filter(filtvec90,1, (sload10sec(:,index) + offset));
        % calculate appropriate initial conditions: jigger so it ends where
        the next one starts
    working = filter(filtvec90,1,sload10sec(:,index), zi); % use ICs from
        as if two identical days had been appended
    % BUT need to slide it so that the data points are centered in the
    % boxcars
    chunk = working(1:(window90+1)/2)-offset;
    working = [working((window90+3)/2:end); chunk];
    sloadavg90(:,index) = working;
    sleftover90(:,index) = sload10sec(:,index) - working;

end

sfavgvec = reshape(sfloadavg90, numpts10s, 1);
sffuzzvec = reshape(sfleftover90, numpts10s, 1);

savgvec = reshape(sloadavg90, numpts10s, 1);
sfuzzvec = reshape(sleftover90, numpts10s, 1);
```

```
sfavgdur = sort(sfavgvec);
sffuzzdur = sort(sffuzzvec);

savgdur = sort(savgvec);
sfuzzdur = sort(sfuzzvec);

sf3fuzz = sf3load10sec - sloadavg90;
sf7fuzz = sf7load10sec - sloadavg90;

%% Get all the ramping and energy characteristics of the separated curves

for index = 1:numdays10sec

        sfavgrr(:,index) = rr10sec(sfloadavg90(:,index));
        sffuzzrr(:,index) = rr10sec(sfleftover90(:,index));

        savgrr(:,index) = rr10sec(sloadavg90(:,index));
        sfuzzrr(:,index) = rr10sec(sleftover90(:,index));

        sffuzzint(:,index) = int10sec(sfleftover90(:,index));
        sfuzzint(:,index) = int10sec(sleftover90(:,index));

    sf3fuzzrr(:,index) = rr10sec(sf3fuzz(:,index));
    sf7fuzzrr(:,index) = rr10sec(sf7fuzz(:,index));
end

sfavgrrvec = reshape(sfavgrr, numpts10s, 1);
sffuzzrrvec = reshape(sffuzzrr, numpts10s, 1);

sf3fuzzrrvec = reshape(sf3fuzzrr, numpts10s, 1);
sf7fuzzrrvec = reshape(sf7fuzzrr, numpts10s, 1);

savgrrvec = reshape(savgrr, numpts10s, 1);
sfuzzrrvec = reshape(sfuzzrr, numpts10s, 1);

sffuzzintvec = reshape(sffuzzint, numpts10s, 1);
sfuzzintvec = reshape(sfuzzint, numpts10s, 1);

sfavgrrdur = sort(sfavgrrvec);
sffuzzrrdur = sort(sffuzzrrvec);

sf3fuzzrrdur = sort(sf3fuzzrrvec);
sf7fuzzrrdur = sort(sf7fuzzrrvec);

savgrrdur = sort(savgrrvec);
sfuzzrrdur = sort(sfuzzrrvec);

sffuzzintdur = sort(sffuzzintvec);
sfuzzintdur = sort(sfuzzintvec);
```

```
sffuzzabsrr = abs(sffuzzrrvec);
sffuzzmabrr = mean(sffuzzabsrr) % print this out to put in the paper

%print 10th %ile and 90th %ile
per10pts10s = floor(numpts10s/10);
per10sfrr10s = sffuzzrrdur(per10pts10s) % print out 10th %ile
per90sfrr10s = sffuzzrrdur(numpts10s - per10pts10s) % print out 90th %ile


%% Plot time series of 10-sec data curves

% shift vertically to get a minimum at zero energy
efshift = min(sffuzzintvec);
eshift = min(sfuzzintvec);

%% energy durcurves

%same minimum as in time domain curves, same reason to shift
```

C.1.3 Processing Data Set B

thesis_makefigs_ch_loaddur_subfile_pjmdata_20jul11_noplots.m

```
%% Olivia Leitermann  20 Jul 11  MIT LEES
%% script to load, analyze, and plot pjm 4-second regulation data

%% load hourly data, for information
numwkdays = 7;
numwks = 4;


bulkfilenames = {'jun10_pjm_rmcp_bulkload.csv',
    'sep10_pjm_rmcp_bulkload.csv', ...
            'jan11_pjm_rmcp_bulkload.csv', 'mar11_pjm_rmcp_bulkload.csv'};

bulkpjmload = [];

for index = 1:numwks
    fid = fopen(bulkfilenames{index});
    bulkpjmload_cell = textscan(fid, '%*s␣%*s␣%*f␣%f␣%*f␣%*f␣%*f␣%*f␣%f',
        'Delimiter', ',', ...
        'CommentStyle', '%'); % only want 4th col: total hourly bulk load
            (MWh)
    % Cols are: EPT Hour Ending,GMT Hour Ending,RMCP ($/MWh),Total PJM RT
        Load (MWh),
```

```
    % Total PJM Lost Opportunity Cost Credit ($),Total PJM Reg Purchases
        (MWh),
    % Total PJM Self-Scheduled Reg (MWh),Total PJM-Assigned Reg (MWh)

    % data are hourly and start with hour ending 1am on 1 xxx 201x.
    fclose(fid);
    bulkpjmload = [bulkpjmload bulkpjmload_cell{1}];
end

% trim bulkpjmload so we have just the relevant points, for the weeks
% available:
bulkpjmload = bulkpjmload(1:numwkdays*24,:);
hourlytime1wk = [0:1:numwkdays*24-1]';
hourlytime2wk = [0:1:numwkdays*24*2-1]';
bulkpjmload2col = reshape(bulkpjmload, [], 2);
meanbulkpjmload = mean(mean(bulkpjmload));
save('meanbulkpjmload.mat', 'meanbulkpjmload');

%% load data, Jun, Sep 1-7, 2010; Jan, Mar 1-7, 2011


filenames = {'rto-reg-data-external-1-7jun-2010.csv',
    'rto-reg-data-external-1-7sep-2010.csv', ...
                'rto-reg-data-external-1-7jan-2011.csv',
                    'rto-reg-data-external-1-7mar-2011.csv'};

tregpjmmat = [];
tregpjmintmat = [];
tregpjmrrmat = [];

for index = 1:numwks
    %index
    fid = fopen(filenames{index});
    treg4s_cell = textscan(fid, '%*s␣%f␣%f␣%f␣%*s␣%*s␣%*s␣%*s␣%*s',
        'Delimiter', ',', ...
        'CommentStyle', '%'); % only want last 3 cols: dispatched reg (MW),
            total online reg capacity (MW), quotient
    % (also is some extra columns in the file; ignore those, they're empty
    % anyway)
    %"Reg. Test Data from",01/01/10,"to",1/2/2010
    %"Time","RTO RegA","RTO Treg","% raise/lower"
    % data are every 4 seconds and start with midnight on 1 xxx 201x.
    fclose(fid);
    %numptsnow = numel(marreg4s_cell{1})
    tregpjmmat = [tregpjmmat treg4s_cell{1}]; % traditonal reg

    % compute ramp rates and energy
    tregpjmintmat = [tregpjmintmat int4sec(treg4s_cell{1})];
    tregpjmrrmat = [tregpjmrrmat rr4sec(treg4s_cell{1})];
```

```
end

numpts1wk = numel(tregpjmmat)/numwks;

% slide storage to have a minimum at zero and start at the same place each
% week
tregpjmintmat = tregpjmintmat - min(min(tregpjmintmat));

tregpjmvec = reshape(tregpjmmat, [], 1);
tregpjmintvec = reshape(tregpjmintmat, [], 1);
tregpjmrrvec = reshape(tregpjmrrmat, [], 1);

%% compute durations

tregpjmdur = sort(tregpjmvec);
tregpjmintdur = sort(tregpjmintvec);
tregpjmrrdur = sort(tregpjmrrvec);

%% frequency content

tregfftH = fft(tregpjmmat); % force the n-point fft, each col has result

% the frequency content from the guy starts with the dc and then is #
    cycles/full vec.
% this means that each frequency step is 1/wk or
numsecperwk = 3600*24*7;
df = 1/numsecperwk; % step frequency in Hz

% the highest frequency that I can actually sense is N/2 cycles per full
    vec, or half the sampling freq
% in this case 1/8 Hz
% the fourier guys will give me data for up to N cycles per full vec, but
    these just overlap the lower freqs

fmax = 1/8;

tregfftf = [0:df:fmax];

tregfftHmag = abs(tregfftH(1:numpts1wk/2+1,:));
tregfftHph = angle(tregfftH(1:numpts1wk/2+1,:));


%%%%%%%%%%%%%%%%%%%%%%%%% plots %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% vectors for the x axes

timehr1wk = linspace(0, 24*numwkdays-4/3600, numpts1wk);
timehr4wks = linspace(0,24*(numwkdays*numwks)-4/3600, numpts1wk*numwks);
```

```
durx1wk = linspace(0, 1, numpts1wk);
durx4wks = linspace(0, 1, numpts1wk*numwks);


%% time series plots

%% duration plots
```

C.2 Dividing the Regulation Burden

C.2.1 Open-Loop Filters

thesis_makefigs_ch_dividereg_subfile_10s_simpfilt.m

```
%% Olivia Leitermann   29 Jul 11  MIT LEES
%% subfile for doing filtering on 10 sec data
%% called by thesis_makefig_ch_dividereg_<date>.m


loaddata10sec = load('NEISO_AGC_NEload.csv');
% columns are 10-sec (!) samples of NE load on 10 non-consecutive days,
% midnight to midnight.  first 3 cols are sundays 23 mar 08, 28 sept 08, 30
% aug 09, followed by 2 saturdays, 11 apr 09, 2 jan 10, followed by
% mon-friday 16 jun 08, 26 may 09, 29 jul 09, 1 oct 09, 22 feb 08.  Total
% system load MW

% use a 5-point median filter to get rid of spikes
flddata10sec = medfilt1(loaddata10sec, 5);

% sift out days if I need to...
pickdays = [1 2 3 4 7 8 9];
numdays10s = numel(pickdays);

sload10sec = loaddata10sec(:,pickdays);
sfload10sec = flddata10sec(:,pickdays);

numpts10s = numel(sload10sec);



%% find ramp rate of original 10-sec data with and without median filter

init10s = zeros(size(sload10sec));
```

```
sloadrr10sec = init10s;
sfloadrr10sec = init10s;

for index = 1:numdays10s

    sloadrr10sec(:,index) = rr10sec(sload10sec(:,index));
    sfloadrr10sec(:,index) = rr10sec(sfload10sec(:,index));

end

sload10secvec = reshape(sload10sec, numel(sload10sec), 1);
sfload10secvec = reshape(sfload10sec, numel(sfload10sec), 1);

sloadrr10secvec = reshape(sloadrr10sec, numel(sloadrr10sec), 1);
sfloadrr10secvec = reshape(sfloadrr10sec, numel(sfloadrr10sec), 1);

sload10secdur = sort(sload10secvec);
sfload10secdur = sort(sfload10secvec);
sloadrr10secdur = sort(sloadrr10secvec);
sfloadrr10secdur = sort(sfloadrr10secvec);

%% Find the bulk energy of the curves and subtract to get fuzz only

window90 = 90*6+1; % number of samples to make up 90 min, want an odd number
filtvec90 = ones(window90, 1)./window90;

init = zeros(size(sload10sec));

sfavg90mat = init;
sffuzz90mat = init;

savg90mat = init;
sfuzz90mat = init;

for index = 1:numdays10s

    offset = sfload10sec(1,index) - sfload10sec(end,index);
    [waitforit, zi] = filter(filtvec90,1, (sfload10sec(:,index) + offset));
        % calculate appropriate initial conditions: jigger so it ends where
        the next one starts
    working = filter(filtvec90,1,sfload10sec(:,index), zi); % use ICs from
        as if two identical days had been appended
    % BUT need to slide it so that the data points are centered in the
    % boxcars
    chunk = working(1:(window90+1)/2)-offset;
    working = [working((window90+3)/2:end); chunk];
    sfavg90mat(:,index) = working;
    sffuzz90mat(:,index) = sfload10sec(:,index) - working;
```

```
    [waitforit, zi] = filter(filtvec90,1, (sload10sec(:,index) + offset));
        % calculate appropriate initial conditions: jigger so it ends where
        the next one starts
    working = filter(filtvec90,1,sload10sec(:,index), zi); % use ICs from
        as if two identical days had been appended
    % BUT need to slide it so that the data points are centered in the
    % boxcars
    chunk = working(1:(window90+1)/2)-offset;
    working = [working((window90+3)/2:end); chunk];
    savg90mat(:,index) = working;
    sfuzz90mat(:,index) = sload10sec(:,index) - working;

end

sfavgvec = reshape(sfavg90mat, numpts10s, 1);
sffuzzvec = reshape(sffuzz90mat, numpts10s, 1);

savgvec = reshape(savg90mat, numpts10s, 1);
sfuzzvec = reshape(sfuzz90mat, numpts10s, 1);

sfavgdur = sort(sfavgvec);
sffuzzdur = sort(sffuzzvec);

savgdur = sort(savgvec);
sfuzzdur = sort(sfuzzvec);

%% Get all the ramping and energy characteristics of the separated curves

for index = 1:numdays10s

        sfavgrr(:,index) = rr10sec(sfavg90mat(:,index));
        sffuzzrr(:,index) = rr10sec(sffuzz90mat(:,index));

        savgrr(:,index) = rr10sec(savg90mat(:,index));
        sfuzzrr(:,index) = rr10sec(sfuzz90mat(:,index));

        sffuzzint(:,index) = int10sec(sffuzz90mat(:,index));
        sfuzzint(:,index) = int10sec(sfuzz90mat(:,index));

end

sfavgrrvec = reshape(sfavgrr, numpts10s, 1);
sffuzzrrvec = reshape(sffuzzrr, numpts10s, 1);

savgrrvec = reshape(savgrr, numpts10s, 1);
sfuzzrrvec = reshape(sfuzzrr, numpts10s, 1);

sffuzzintvec = reshape(sffuzzint, numpts10s, 1);
sfuzzintvec = reshape(sfuzzint, numpts10s, 1);
```

```
sfavgrrdur = sort(sfavgrrvec);
sffuzzrrdur = sort(sffuzzrrvec);

savgrrdur = sort(savgrrvec);
sfuzzrrdur = sort(sfuzzrrvec);

sffuzzintdur = sort(sffuzzintvec);
sfuzzintdur = sort(sfuzzintvec);

%% Design filters: Type 1 chebyshev, passband ripple

n = 3; % filter order
R = 0.1; % passband ripple allowance, dB

% analog prototyping and digital conversion
fs10s = 1/10; % sampling frequency 1 cycle/10 sec
Wpa = 2*pi/60./[3 7 10 15 20 30 45 60]; % cutoff is 1/n min, different filts
numfilts = 8;
numfiltpts = 512*16;

init2 = zeros(numfiltpts, numfilts);
hatfs10s = init2;
was10s = init2;
hadtfs10s = init2;
wads10s = init2;

ta10s = [0:1:60*60*6]; % 6 h time vec
tad10s = [0:10:60*60*6];

for index2 = 1:numfilts
    %[ba, aa] = cheby1(n,R,Wpa(index2),'high', 's'); %zeros and poles
    % design cont time as state space
    [Aa10s, Ba10s, Ca10s, Da10s] = cheby1(n, R, Wpa(index2), 'high', 's');
    sysana = ss(Aa10s, Ba10s, Ca10s, Da10s); %create ss model
    [ba10s, aa10s] = tfdata(sysana, 'v'); % convert to tf
    [hatf10s,wa10s] = freqs(ba10s,aa10s, numfiltpts); % frequency response
    ya10s = step(sysana, ta10s); % generate step resp

    % convert to discrete time
    [Aad10s, Bad10s, Cad10s, Dad10s] = bilinear(Aa10s, Ba10s, Ca10s, Da10s,
        fs10s);
    sysdig = ss(Aad10s, Bad10s, Cad10s, Dad10s, 1/fs10s); % discrete ss
        model
    [bad10s, aad10s] = tfdata(sysdig, 'v'); % convert to tf
    [hadtf10s, fad10s] = freqz(bad10s, aad10s, numfiltpts, fs10s); % freq
        resp
    yad10s = step(sysdig, tad10s); % step resp
```

```
    % store vars
    hatfs10s(:,index2) = hatf10s;
    was10s(:,index2) = wa10s;
    bads10s(:,index2) = bad10s;
    aads10s(:,index2) = aad10s;
    hadtfs10s(:,index2) = hadtf10s;
    fads10s(:,index2) = fad10s;
    yas10s(:,index2) = ya10s;
    yads10s(:,index2) = yad10s;

end

fas10s = was10s/2/pi;

maghatfs10s = abs(hatfs10s);
phhatfs10s = angle(hatfs10s);

maghadtfs10s = abs(hadtfs10s);
phhadtfs10s = angle(hadtfs10s);

numpts10s1day = numpts10s/numdays10s;

%% Peform filtering

init = zeros(numpts10s/numdays10s, numdays10s);
init3 = zeros(numpts10s, numfilts);

hifreq10s = init3;
lofreq10s = init3;
hifreqint10s = init3;
lofreqrr10s = init3;
hifreqforfft = zeros(numpts10s1day, numfilts);
lofreqforfft = zeros(numpts10s1day, numfilts);

for index2 = 1:numfilts;
    hifreqs1 = init;
    lofreqs1 = init;
    hifreqints1 = init;
    lofreqrrs1 = init;
    hifreqrrs1 = init;

    for index = 1:numdays10s
        offset = sffuzz90mat(1,index) - sffuzz90mat(end,index);
        % also important to use the filter indicated by index2
        [waitforit, zi] = filter(bads10s(:,index2), aads10s(:,index2),
            (sffuzz90mat(:,index) + offset)); % calculate appropriate
            initial conditions: jigger so it ends where the next one starts
```

```
        hifreqs1(:,index) = filter(bads10s(:,index2), aads10s(:,index2),
            sffuzz90mat(:,index), zi); % use ICs from as if two identical
            days had been appended
        lofreqs1(:,index) = sffuzz90mat(:,index) - hifreqs1(:,index);

        % calculate integral and ramp rate
        hifreqints1(:,index) = int10sec(hifreqs1(:,index));
        lofreqrrs1(:,index) = rr10sec(lofreqs1(:,index));
        hifreqrrs1(:,index) = rr10sec(hifreqs1(:,index));

    end

        % save first day for fft
        lofreqforfft(:,index2) = lofreqs1(:,1);
        hifreqforfft(:,index2) = hifreqs1(:,1);

        %store all vars in a vector
    lofreqs1 = reshape(lofreqs1, numpts10s, 1);
    hifreqs1 = reshape(hifreqs1, numpts10s, 1);
    lofreqrrs1 = reshape(lofreqrrs1, numpts10s, 1);
    hifreqints1 = reshape(hifreqints1, numpts10s, 1);
    hifreqrrs1 = reshape(hifreqrrs1, numpts10s, 1);

    % put vectors in a matrix
    lofreq10s(:,index2) = lofreqs1;
    hifreq10s(:,index2) = hifreqs1;
    lofreqrr10s(:,index2) = lofreqrrs1;
    hifreqint10s(:,index2) = hifreqints1;
    hifreqrr10s(:,index2) = hifreqrrs1;
end

lofreqdur10s = sort(lofreq10s);
hifreqdur10s = sort(hifreq10s);
lofreqrrdur10s = sort(lofreqrr10s);
hifreqintdur10s = sort(hifreqint10s);
hifreqrrdur10s = sort(hifreqrr10s);

bigdurx10s = linspace(0, 1, numpts10s);
bigtime10s = linspace(0, 24*numdays10s-1/360, numpts10s); % time in hours

%% Frequency content
% since it's a pain to have all the days, just do a single day for each
    filter

phifreq10sH = fft(hifreqforfft); % each col is one day for one filt
plofreq10sH = fft(lofreqforfft);

numsecperday = 3600*24;
df = 1/numsecperday; % step freq in Hz
```

```
% highest frequency to sense is N/2 cycles per full vector
% in this case 1/20 Hz

fmax = 1/20;

ppjmfftf = [0:df:fmax]; % should be the right frequency vect

phifreq10sHmag = abs(phifreq10sH(1:numpts10s1day/2+1,:));
plofreq10sHmag = abs(plofreq10sH(1:numpts10s1day/2+1,:));

phifreq10sHph = angle(phifreq10sH(1:numpts10s1day/2+1,:));
plofreq10sHph = angle(plofreq10sH(1:numpts10s1day/2+1,:));

% Wmax vs <|dP/dt|>

cheb10swmax = hifreqintdur10s(end,:)-hifreqintdur10s(1,:); % shifted the
    plot, not the column!
cheb10saadpdt = mean(abs(lofreqrr10s)); % mean for each column
orig10saadpdt = mean(abs(sffuzzrrvec)); % mean for original data

cheb10srmsdpdt = sqrt(mean(lofreqrr10s.^2));
orig10srmsdpdt = sqrt(mean(sffuzzrrvec.^2));
%% figuring out some stuff for the closed-loop filter:

cheb10swmax25pc = 1.25 * cheb10swmax; % print to terminal
cheb10swmax20pc = 1.2 * cheb10swmax;
cheb10swmax100pc = 2*cheb10swmax;

maxenergyforsim = ceil(cheb10swmax20pc)
```

C.2.2 Closed Loop Filtering, No Loss Feedforward

`thesis_makefigs_ch_dividereg_subfile_cl_nolossff.m`

This file calls Simulink to process the closed-loop models.

```
%% Partial script called by thesis_makefigs_ch_dividereg
%% Olivia Leitermann file created 1 Jul 11  MIT LEES

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% No loss feedforward
%%%%%%%%%%%%%%%%%%%%%%%%%% Closed Loop Filtering, Mystery BA data
    %%%%%%%%%%%%%%%%

% grabbed from
% run_simulink_stormodel_neisodata_NLFB_15mar11_nobulkenergy_60m.m
```

```
model10nff = 'virtual_PP_th_stor_w_NL_FB_rails_useasbaseline_10_nolossff';
    % my model name
model15nff = 'virtual_PP_th_stor_w_NL_FB_rails_useasbaseline_15_nolossff';
    % my model name
model20nff = 'virtual_PP_th_stor_w_NL_FB_rails_useasbaseline_20_nolossff';
    % my model name
model30nff = 'virtual_PP_th_stor_w_NL_FB_rails_useasbaseline_30_nolossff';
    % my model name
% outputs are: out1: total power; out2: thermal power; out3: storage
% state of charge; out4: load power; out5: storage balancing power; out6:
% storage power

modelvecnff = [model10nff; model15nff; model20nff; model30nff];

%filenames for input are neiso_dayn where n = [1 2 3 4 7 8 9] - the days
%without anomalies, median-filtered.



%% first, plot the feedback fxns with the new FB cubic gain

SOCx = linspace(-1, 1, 1e3);

Kg = 80; % 45 MW/norm stor^3
gofSOC = Kg*SOCx.^3;

figure
a=plot(SOCx, gofSOC, 'k-');
xlabel('Normalized␣State␣of␣Charge');
ylabel('Feedback␣Signal,␣MW');
title('Nonlinear␣State␣Feedback␣Function');
set(a, 'LineWidth', 2);
grid on;
saveas(gcf,'NLFB_plot.eps', 'epsc');


%% Set up the variables for the sims

load('neiso_all_pad.mat');
%includes 'fulltimesec', 'paddat', 'paddathifreq30', 'paddathifreq60',
    'paddathifreq90'
negpts = 360*2; % 2 hours: amount of padded constant data

%paddat already has the padded bulk data in it; don't need to re-create

load('neiso_noncausal_306090_fuzz.mat');

% Parameters: need U(-.005, 0.005) for variable efficiency and U(0.7, 1.3)
% for variable self-discharge power. Changes rather fast, but go with it
```

```
% for now

% vareff = rand(numpts, numdays)/100-0.005;
% varPsd = rand(numpts, numdays)*0.6+0.7;

numrealsimpts10s = 86401;
simfactor10s = 10; % 10 times as many sim pts as real pts

pouttotall10sclnff = [];
thpowall10sclnff = [];
socall10sclnff = [];
ploadall10sclnff = [];
pbalanceall10sclnff = [];
pstorall10sclnff = [];
thrrall10sclnff = [];
storrrall10sclnff = [];
balrrall10sclnff = [];
railflagall10sclnff = [];
lossfraxall10sclnff = [];
tall10sclnff = [];

pouttotall10sclmatnff = [];
thpowall10sclmatnff = [];
socall10sclmatnff = [];
ploadall10sclmatnff = [];
pbalanceall10sclmatnff = [];
pstorall10sclmatnff = [];
thrrall10sclmatnff = [];
storrrall10sclmatnff = [];
balrrall10sclmatnff = [];
railflagall10sclmatnff = [];
lossfraxall10sclmatnff = [];
tall10sclmatnff = [];

%% Note on how to get only the output points I want (each second):
%% Configuration Parameters/(Data Import/Export)/Output Options:
%% Choose "Produce Specified Output Only," time is [0:86400]



numfiltsim = 4;
storsizevec = maxenergyforsim(3:6); % 10, 15, 20, 30 minute filters

numsimptstot = numel(fulltimesec);

%for filtindexnff = 4;
for filtindexnff = 1:numfiltsim % cols are concat days, dim 2 holds filts
    filtindexnff % print so we know where we are
    storsize = storsizevec(filtindexnff);
```

```
modelnff = modelvecnff(filtindexnff,:);

% need to reset vars each time because I'm being mean and making them
% grow
pouttotall10sclnff = [];
thpowall10sclnff = [];
socall10sclnff = [];
ploadall10sclnff = [];
pbalanceall10sclnff = [];
pstorall10sclnff = [];
thrrall10sclnff = [];
storrrall10sclnff = [];
balrrall10sclnff = [];
railflagall10sclnff = [];
lossfraxall10sclnff = [];
tall10sclnff = [];

for dayindex = 1:numdays10s

    %ut = [fulltimesec' paddatsift(dayindex,:)' vareff(:,dayindex)
        varPsd(:,dayindex)];
    ut = [fulltimesec' paddathifreq90(dayindex,:)'
        repmat(storsize,numsimptstot, 1)];
        timespan = [-7200 86400];
        [t,x,pout_tot, pth, soc, pload, pbalance, pstor, thrr, storrr,
            ...
            balrr, railflag, lossfrax, absPin] =
                sim(modelnff,timespan,[],ut);
        % units of power quantities are MW, units of SOC are MJ (?)

        % find where the real part of the sim begins
        ind = find(t>=0, 1, 'first'); % should return the first time
            step where t>=0
        % snag only those values, to start

        % concatenate all waveforms, truncating to the "real" part
        pouttotall10sclnff = [pouttotall10sclnff; pout_tot(ind:end)];
        thpowall10sclnff = [thpowall10sclnff; pth(ind:end)];
        socall10sclnff = [socall10sclnff; soc(ind:end)];
        ploadall10sclnff = [ploadall10sclnff; pload(ind:end)];
        pbalanceall10sclnff = [pbalanceall10sclnff; pbalance(ind:end)];
        pstorall10sclnff = [pstorall10sclnff; pstor(ind:end)];
        thrrall10sclnff = [thrrall10sclnff; thrr(ind:end)];
        storrrall10sclnff = [storrrall10sclnff; storrr(ind:end)];
        balrrall10sclnff = [balrrall10sclnff; balrr(ind:end)];
        railflagall10sclnff = [railflagall10sclnff; railflag(ind:end)];
        lossfraxall10sclnff = [lossfraxall10sclnff; lossfrax(ind:end)];
        tall10sclnff = [tall10sclnff;
            t(ind:end)+(dayindex-1)*numrealsimpts10s];
```

```
    end
    pouttotall10sclmatnff = [pouttotall10sclmatnff pouttotall10sclnff];
    thpowall10sclmatnff = [thpowall10sclmatnff thpowall10sclnff];
    socall10sclmatnff = [socall10sclmatnff socall10sclnff];
    ploadall10sclmatnff = [ploadall10sclmatnff ploadall10sclnff];
    pbalanceall10sclmatnff = [pbalanceall10sclmatnff pbalanceall10sclnff];
    pstorall10sclmatnff = [pstorall10sclmatnff pstorall10sclnff];
    thrrall10sclmatnff = [thrrall10sclmatnff thrrall10sclnff];
    storrrall10sclmatnff = [storrrall10sclmatnff storrrall10sclnff];
    balrrall10sclmatnff = [balrrall10sclmatnff balrrall10sclnff];
    railflagall10sclmatnff = [railflagall10sclmatnff railflagall10sclnff];
    lossfraxall10sclmatnff = [lossfraxall10sclmatnff lossfraxall10sclnff];
    tall10sclmatnff = [tall10sclmatnff tall10sclnff];
end

% clear the partial vars so I don't accidentally use them:
clear pouttotall10sclnff thpowall10sclnff socall10sclnff ploadall10sclnff
    ...
    pbalanceall10sclnff pstorall10sclnff thrrall10sclnff storall10sclnff ...
    balrrall10sclnff railflagall10sclnff lossfraxall10sclnff tall10sclnff;


% i think this is ok because of the way I've set up the sims now...
% % flag if I'm not getting the data sampled properly
% mintimestep10scl = min(tall10scl(2:end)-tall10scl(1:end-1));
% if mintimestep10scl < 1
%     'warning: min time step < 1! check simulation parameters'
%     mintimestep10scl
% end

%flag if storage is filling up or emptying out
[railflagmax10sclnff, railflagind10sclnff] =
    max(max(abs(railflagall10sclmatnff)));
if railflagmax10sclnff > 0
    'warning: storage is emptying or filling completely! results may be
        inaccurate'
    disp('one time of violations:');
    railflagind10sclnff
end

thr10sclmatnff = tall10sclmatnff/60/60;

%% compute mean efficiencies

lossfraxmeannff = mean(lossfraxall10sclmatnff); % mean over cols for 4
    different filts
%NOT the same as efficiency over the whole time: avg. instant. eff.
```

```
lossfraxmeannff

%% compute durations:

pouttotdur10sclmatnff = sort(pouttotall10sclmatnff); % should properly sort
    cols
thpowdur10sclmatnff = sort(thpowall10sclmatnff);
socdur10sclmatnff = sort(socall10sclmatnff);
ploaddur10sclmatnff = sort(ploadall10sclmatnff);
pbalancedur10sclmatnff = sort(pbalanceall10sclmatnff);
pstordur10sclmatnff = sort(pstorall10sclmatnff);
thrrdur10sclmatnff = sort(thrrall10sclmatnff);
storrrdur10sclmatnff = sort(storrrall10sclmatnff);
balrrdur10sclmatnff = sort(balrrall10sclmatnff);
railflagdur10sclmatnff = sort(railflagall10sclmatnff);

wmaxsocnff = socdur10sclmatnff(end,:) - socdur10sclmatnff(1,:);
dpdtmeanabsnff = mean(abs(thrrdur10sclmatnff));

%% Now with the unfiltered data: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pouttotall10sclnffwbulk = [];
thpowall10sclnffwbulk = [];
socall10sclnffwbulk = [];
ploadall10sclnffwbulk = [];
pbalanceall10sclnffwbulk = [];
pstorall10sclnffwbulk = [];
thrrall10sclnffwbulk = [];
storrrall10sclnffwbulk = [];
balrrall10sclnffwbulk = [];
railflagall10sclnffwbulk = [];
lossfraxall10sclnffwbulk = [];
tall10sclnffwbulk = [];

pouttotall10sclmatnffwbulk = [];
thpowall10sclmatnffwbulk = [];
socall10sclmatnffwbulk = [];
ploadall10sclmatnffwbulk = [];
pbalanceall10sclmatnffwbulk = [];
pstorall10sclmatnffwbulk = [];
thrrall10sclmatnffwbulk = [];
storrrall10sclmatnffwbulk = [];
balrrall10sclmatnffwbulk = [];
railflagall10sclmatnffwbulk = [];
lossfraxall10sclmatnffwbulk = [];
tall10sclmatnffwbulk = [];

durx10ssim = linspace(0, 1, numdays10s*numrealsimpts10s);
```

```
numfiltsim = 4;
storsizevec = maxenergyforsim(3:6); % 10, 15, 20, 30 minute filters

numsimptstot = numel(fulltimesec);
ind = 1;
%for filtindexnffwbulk = 4;
for filtindexnffwbulk = 1:numfiltsim % cols are concat days, dim 2 holds
    filts
    filtindexnffwbulk % print so we know where we are
    storsize = storsizevec(filtindexnffwbulk);
    modelnff = modelvecnff(filtindexnffwbulk,:);

    % need to reset vars each time because I'm being mean and making them
    % grow
    pouttotall10sclnffwbulk = [];
    thpowall10sclnffwbulk = [];
    socall10sclnffwbulk = [];
    ploadall10sclnffwbulk = [];
    pbalanceall10sclnffwbulk = [];
    pstorall10sclnffwbulk = [];
    thrrall10sclnffwbulk = [];
    storrrall10sclnffwbulk = [];
    balrrall10sclnffwbulk = [];
    railflagall10sclnffwbulk = [];
    lossfraxall10sclnffwbulk = [];
    tall10sclnffwbulk = [];

    for dayindex = 1:numdays10s

        %ut = [fulltimesec' paddatsift(dayindex,:)' vareff(:,dayindex)
            varPsd(:,dayindex)];
        ut = [fulltimesec' paddat(dayindex,:)'
            repmat(storsize,numsimptstot, 1)];
            timespan = [-7200 86400];
            [t,x,pout_tot, pth, soc, pload, pbalance, pstor, thrr, storrr,
                ...
                balrr, railflag, lossfrax, absPin] =
                    sim(modelnff,timespan,[],ut);
            % units of power quantities are MW, units of SOC are MJ (?)

            % find where the real part of the sim begins
%            ind = find(t>=0, 1, 'first'); % should return the first time
    step where t>=0
            % snag only those values, to start

            % concatenate all waveforms, truncating to the "real" part
            pouttotall10sclnffwbulk = [pouttotall10sclnffwbulk;
                pout_tot(ind:end)];
```

```
            thpowall10sclnffwbulk = [thpowall10sclnffwbulk; pth(ind:end)];
            socall10sclnffwbulk = [socall10sclnffwbulk; soc(ind:end)];
            ploadall10sclnffwbulk = [ploadall10sclnffwbulk; pload(ind:end)];
            pbalanceall10sclnffwbulk = [pbalanceall10sclnffwbulk;
                pbalance(ind:end)];
            pstorall10sclnffwbulk = [pstorall10sclnffwbulk; pstor(ind:end)];
            thrrall10sclnffwbulk = [thrrall10sclnffwbulk; thrr(ind:end)];
            storrrall10sclnffwbulk = [storrrall10sclnffwbulk;
                storrr(ind:end)];
            balrrall10sclnffwbulk = [balrrall10sclnffwbulk; balrr(ind:end)];
            railflagall10sclnffwbulk = [railflagall10sclnffwbulk;
                railflag(ind:end)];
            lossfraxall10sclnffwbulk = [lossfraxall10sclnffwbulk;
                lossfrax(ind:end)];
            tall10sclnffwbulk = [tall10sclnffwbulk;
                t(ind:end)+(dayindex-1)*numrealsimpts10s];

    end
    pouttotall10sclmatnffwbulk = [pouttotall10sclmatnffwbulk
        pouttotall10sclnffwbulk];
    thpowall10sclmatnffwbulk = [thpowall10sclmatnffwbulk
        thpowall10sclnffwbulk];
    socall10sclmatnffwbulk = [socall10sclmatnffwbulk socall10sclnffwbulk];
    ploadall10sclmatnffwbulk = [ploadall10sclmatnffwbulk
        ploadall10sclnffwbulk];
    pbalanceall10sclmatnffwbulk = [pbalanceall10sclmatnffwbulk
        pbalanceall10sclnffwbulk];
    pstorall10sclmatnffwbulk = [pstorall10sclmatnffwbulk
        pstorall10sclnffwbulk];
    thrrall10sclmatnffwbulk = [thrrall10sclmatnffwbulk
        thrrall10sclnffwbulk];
    storrrall10sclmatnffwbulk = [storrrall10sclmatnffwbulk
        storrrall10sclnffwbulk];
    balrrall10sclmatnffwbulk = [balrrall10sclmatnffwbulk
        balrrall10sclnffwbulk];
    railflagall10sclmatnffwbulk = [railflagall10sclmatnffwbulk
        railflagall10sclnffwbulk];
    lossfraxall10sclmatnffwbulk = [lossfraxall10sclmatnffwbulk
        lossfraxall10sclnffwbulk];
    tall10sclmatnffwbulk = [tall10sclmatnffwbulk tall10sclnffwbulk];
end

% clear the partial vars so I don't accidentally use them:
clear pouttotall10sclnffwbulk thpowall10sclnffwbulk socall10sclnffwbulk
    ploadall10sclnffwbulk ...
    pbalanceall10sclnffwbulk pstorall10sclnffwbulk thrrall10sclnffwbulk
        storall10sclnffwbulk ...
    balrrall10sclnffwbulk railflagall10sclnffwbulk lossfraxall10sclnffwbulk
        tall10sclnffwbulk;
```

```
% i think this is ok because of the way I've set up the sims now...
% % flag if I'm not getting the data sampled properly
% mintimestep10scl = min(tall10scl(2:end)-tall10scl(1:end-1));
% if mintimestep10scl < 1
%     'warning: min time step < 1! check simulation parameters'
%     mintimestep10scl
% end

%flag if storage is filling up or emptying out
[railflagmax10sclnffwbulk, railflagind10sclnffwbulk] =
    max(max(abs(railflagall10sclmatnffwbulk)));
if railflagmax10sclnffwbulk > 0
    'warning: storage is emptying or filling completely! results may be
        inaccurate'
    disp('one time of violations:');
    railflagind10sclnffwbulk
end

thr10sclmatnffwbulk = tall10sclmatnffwbulk/60/60;

%% compute mean efficiencies

lossfraxmeannffwbulk = mean(lossfraxall10sclmatnffwbulk); % mean over cols
    for 4 different filts
%NOT the same as efficiency over the whole time: avg. instant. eff.

lossfraxmeannffwbulk

%% compute durations:

pouttotdur10sclmatnffwbulk = sort(pouttotall10sclmatnffwbulk); % should
    properly sort cols
thpowdur10sclmatnffwbulk = sort(thpowall10sclmatnffwbulk);
socdur10sclmatnffwbulk = sort(socall10sclmatnffwbulk);
ploaddur10sclmatnffwbulk = sort(ploadall10sclmatnffwbulk);
pbalancedur10sclmatnffwbulk = sort(pbalanceall10sclmatnffwbulk);
pstordur10sclmatnffwbulk = sort(pstorall10sclmatnffwbulk);
thrrdur10sclmatnffwbulk = sort(thrrall10sclmatnffwbulk);
storrrdur10sclmatnffwbulk = sort(storrrall10sclmatnffwbulk);
balrrdur10sclmatnffwbulk = sort(balrrall10sclmatnffwbulk);
railflagdur10sclmatnffwbulk = sort(railflagall10sclmatnffwbulk);

wmaxsocnffwbulk = socdur10sclmatnffwbulk(end,:) -
    socdur10sclmatnffwbulk(1,:);
dpdtmeanabsnffwbulk = mean(abs(thrrdur10sclmatnffwbulk));
```

```
%% Plot simulation results %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% time series results for checking

figure
plot(thr10sclmatnff(:,1), pstorall10sclmatnff(:,1), 'k-', ...
   thr10sclmatnff(:,2), pstorall10sclmatnff(:,2), 'b--', ...
   thr10sclmatnff(:,3), pstorall10sclmatnff(:,3), 'r:', ...
   thr10sclmatnff(:,4), pstorall10sclmatnff(:,4), 'm-.');
title('Power␣out␣of␣storage␣device,␣no␣loss␣feedforward');
xlabel('Time,␣hours');
ylabel('Power,␣MW');
legend('10␣min', '15␣min', '20␣min', '30␣min');
xlim([10 14]);
ylim([-50 50]);
saveas(gcf,'pvst_10s_cl_stor_nolossff.eps', 'epsc');

figure
plot(thr10sclmatnff(:,1), socall10sclmatnff(:,1), 'k-', ...
      thr10sclmatnff(:,2), socall10sclmatnff(:,2), 'b--', ...
      thr10sclmatnff(:,3), socall10sclmatnff(:,3), 'r:', ...
      thr10sclmatnff(:,4), socall10sclmatnff(:,4), 'm-.');
hold on;
plot([0; thr10sclmatnff(end,1)], repmat(maxenergyforsim(3), 2,1), 'k-', ...
     [0; thr10sclmatnff(end,1)], repmat(maxenergyforsim(4), 2,1), 'b--', ...
     [0; thr10sclmatnff(end,1)], repmat(maxenergyforsim(5), 2,1), 'r:', ...
     [0; thr10sclmatnff(end,1)], repmat(maxenergyforsim(6), 2,1), 'm-.');
title('Storage␣device␣state␣of␣charge,␣no␣loss␣feedforward');
xlabel('Time,␣hours');
ylabel('Energy,␣MWh');
%xlim([10 15]);
xlim([0 thr10sclmatnff(end,1)]);
ylim([0 22]);
legend('10␣min', '15␣min', '20␣min', '30␣min', '10␣min␣size', '15␣min␣
    size',...
        '20␣min␣size', '30␣min␣size','Location', 'NW');
saveas(gcf,'evst_10s_cl_stor_nolossff.eps', 'epsc');


figure
plot(thr10sclmatnff(:,1), pouttotall10sclmatnff(:,1), 'k-', ...
      thr10sclmatnff(:,2), pouttotall10sclmatnff(:,2), 'b--', ...
      thr10sclmatnff(:,3), pouttotall10sclmatnff(:,3), 'r:', ...
      thr10sclmatnff(:,4), pouttotall10sclmatnff(:,1), 'g-.', ...
      thr10sclmatnff(:,1), thpowall10sclmatnff(:,1), 'm-', ...
      thr10sclmatnff(:,2), thpowall10sclmatnff(:,2), 'k--', ...
      thr10sclmatnff(:,3), thpowall10sclmatnff(:,3), 'b:', ...
```

```
      thr10sclmatnff(:,4), thpowall10sclmatnff(:,4), 'r-.');
title('Power␣demand␣and␣thermal␣power␣delivered,␣10-second␣data,␣no␣loss␣
    feedforward');
xlabel('Time,␣hours');
ylabel('Power,␣MW');
legend('Total␣Power␣10␣min','Total␣Power␣15␣min','Total␣Power␣20␣min', ...
    'Total␣Power␣30␣min', 'Thermal␣10␣min', 'Thermal␣15␣min', 'Thermal␣20␣
        min', ...
    'Thermal␣30␣min', 'Location', 'SW');
xlim([10 15]);
ylim([-100 100]);
saveas(gcf,'pvstx2_10s_cl_th_tot_nolossff.eps', 'epsc');

figure
plot(thr10sclmatnff, railflagall10sclmatnff);
title('Flag␣for␣full␣or␣empty␣storage,␣no␣loss␣feedforward');
xlabel('Time,␣hours');
ylabel('+1␣=␣full;␣-1␣=␣empty');
legend('10␣min', '15␣min', '20␣min', '30␣min');
ylim([-1.1 1.1]);

% figure
% plot(thr30, socall30, 'k-', thr60, socall60, 'b--', thr90, socall90,
    'r-.');
% title('Storage State of Charge, SOC');
% xlabel('Time, hours');
% ylabel('Energy, MWh');
% legend('30-min window', '60-min window', '90-min window');
%
% figure
% plot(thr30, balrrall30, 'k-',thr60, balrrall60, 'b--',thr90, balrrall90,
    'r-.', thr30, thrrall30, 'g:', thr60, thrrall60, 'm-', thr90,
    thrrall90, 'c--');
% title('Component ramp rates')
% legend('balancing power, 30-min window', 'balancing power, 60-min
    window', 'balancing power, 90-min window', ...
%         'thermal power, 30-min window', 'thermal power, 60-min window',
    'thermal power, 90-min window', 'Location', 'NW')
% xlabel('Time, hours');
% ylabel('Power Ramp Rate, MW/min');
%


% power duration curves, thermal and storage

figure
a=plot(durx10ssim, pstordur10sclmatnff(:,1), 'k-', ...
      durx10ssim, pstordur10sclmatnff(:,2), 'b--', ...
      durx10ssim, pstordur10sclmatnff(:,3), 'r:', ...
```

```
      durx10ssim, pstordur10sclmatnff(:,4), 'm-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(bigdurx10s, sffuzzdur, 'm--');
title('Storage Power-Duration Curve, 10-second data, no loss feedforward');
xlabel('Non-dimensional Time');
ylabel('Power, MW');
xlim([-0.05 1.05]);
legend('10 min', '15 min', '20 min', '30 min', 'Original', 'Location',
    'SE');
ylim([-130 130]);
saveas(gcf,'pdurx2_10s_cl_stor_tot_nolossff.eps', 'epsc');

%

figure
a=plot(durx10ssim, thpowdur10sclmatnff(:,1), 'k-', ...
      durx10ssim, thpowdur10sclmatnff(:,2), 'b--', ...
      durx10ssim, thpowdur10sclmatnff(:,3), 'r:', ...
      durx10ssim, thpowdur10sclmatnff(:,4), 'm-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(bigdurx10s, sffuzzdur, 'r--');
title('Thermal Power-Duration Curves Compared, 10-second data, no loss
    feedforward');
xlabel('Non-dimensional Time');
ylabel('Power, GW');
legend('10 min', '15 min', '20 min', '30 min', 'Original', 'Location',
    'SE');
xlim([-0.05 1.05]);
ylim([-350 350]);
saveas(gcf,'pdurx2_10s_cl_th_tot_nolossff.eps', 'epsc');

%ramp rate duration curves, storage

figure
a = plot(durx10ssim, storrrdur10sclmatnff(:,1), 'k-', ...
      durx10ssim, storrrdur10sclmatnff(:,2), 'b--', ...
      durx10ssim, storrrdur10sclmatnff(:,3), 'r:', ...
      durx10ssim, storrrdur10sclmatnff(:,4), 'm-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(bigdurx10s, sffuzzrrdur,'r--');
title('Storage Ramp Rate Duration Curves Compared, 10-second data, no loss
    feedforward');
xlabel('Non-dimensional Time');
ylabel('Ramp Rate, MW/min');
legend('10 min', '15 min', '20 min', '30 min', 'Original', 'Location',
    'SE');
```

```
xlim([-0.05 1.05]);
ylim([-500, 500]);
saveas(gcf,'rrdurx2_10s_cl_stor_tot_nolossff.eps', 'epsc');

%ramp rate duration curves, thermal

figure
a = plot(durx10ssim, thrrdur10sclmatnff(:,1), 'k-', ...
      durx10ssim, thrrdur10sclmatnff(:,2), 'b--', ...
      durx10ssim, thrrdur10sclmatnff(:,3), 'r:', ...
      durx10ssim, thrrdur10sclmatnff(:,4), 'm-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(bigdurx10s, sffuzzrrdur,'r--');
title('Thermal␣Ramp␣Rate␣Duration␣Curves␣Compared,␣10-second␣data,␣no␣loss␣
    feedforward');
xlabel('Non-dimensional␣Time');
ylabel('Ramp␣Rate,␣MW/min');
legend('10␣min', '15␣min', '20␣min', '30␣min', 'Original', 'Location',
    'SE');
xlim([-0.05 1.05]);
ylim([-50, 50]);
saveas(gcf,'rrdurx2_10s_cl_th_tot_nolossff.eps', 'epsc');

% figure
% a=plot(durx10ssim, storrrdur10scl, 'k-', durx10ssim, thrrdur10scl,
    'b--',bigdurx10s, sffuzzrrdur,'r:');
% title('Ramp-Rate-Duration Curves Compared, 10-second data');
% xlabel('Non-dimensional Time');
% ylabel('Ramp Rate, MW/min');
% legend('Storage Ramping', 'Thermal Ramping', 'Original Ramping',
    'Location', 'SE');
% xlim([-0.05 1.05]);
% ylim([-100, 100]);
% set(a, 'LineWidth', 2);
% saveas(gcf,'rrdurx3_10s_cl_stor_th_tot_zoomin.eps', 'epsc');
%


% energy duration curves, storage only

figure
%plot(bigdurx, socdur, 'k-', bigdurx, 25, 'r-.')
a=plot(durx10ssim, socdur10sclmatnff(:,1), 'k-', durx10ssim,
    socdur10sclmatnff(:,2), 'b--', ...
      durx10ssim, socdur10sclmatnff(:,3), 'r:', durx10ssim,
          socdur10sclmatnff(:,4), 'm-.');
hold on;
```

```
plot([-0.05 1.05], repmat(maxenergyforsim(3), 2,1), 'k-', [-0.05 1.05],
    repmat(maxenergyforsim(4), 2,1), 'b--', ...
     [-0.05 1.05], repmat(maxenergyforsim(5), 2,1), 'r:', [-0.05 1.05],
        repmat(maxenergyforsim(6), 2,1), 'm-.');
title('Storage␣Energy-Duration␣Curve,␣no␣loss␣feedforward');
xlabel('Non-dimensional␣Time');
ylabel('Energy,␣MWh');
xlim([-0.05 1.05]);
ylim([0, 22]);
set(a, 'LineWidth', 2);
legend('10␣min', '15␣min', '20␣min', '30␣min', '10␣min␣size', '15␣min␣
    size',...
         '20␣min␣size', '30␣min␣size','Location', 'NW');
saveas(gcf,'edur_10s_cl_stor_nolossff.eps', 'epsc');

%% Just make all the same plots for sims with bulk data included %%%%%%%%%

% time series results for checking

figure
plot(thr10sclmatnffwbulk(:,1), pstorall10sclmatnffwbulk(:,1), 'k-', ...
      thr10sclmatnffwbulk(:,2), pstorall10sclmatnffwbulk(:,2), 'b--', ...
      thr10sclmatnffwbulk(:,3), pstorall10sclmatnffwbulk(:,3), 'r:', ...
      thr10sclmatnffwbulk(:,4), pstorall10sclmatnffwbulk(:,4), 'm-.');
title('Power␣out␣of␣storage␣device,␣no␣loss␣feedforward,␣bulk␣input');
xlabel('Time,␣hours');
ylabel('Power,␣MW');
legend('10␣min', '15␣min', '20␣min', '30␣min');
% xlim([10 14]);
% ylim([-50 50]);
saveas(gcf,'pvst_10s_cl_stor_nolossff_wbulk.eps', 'epsc');

figure
plot(thr10sclmatnffwbulk(:,1), socall10sclmatnffwbulk(:,1), 'k-', ...
      thr10sclmatnffwbulk(:,2), socall10sclmatnffwbulk(:,2), 'b--', ...
      thr10sclmatnffwbulk(:,3), socall10sclmatnffwbulk(:,3), 'r:', ...
      thr10sclmatnffwbulk(:,4), socall10sclmatnffwbulk(:,4), 'm-.');
hold on;
plot([0; thr10sclmatnffwbulk(end,1)], repmat(maxenergyforsim(3), 2,1),
    'k-', ...
      [0; thr10sclmatnffwbulk(end,1)], repmat(maxenergyforsim(4), 2,1),
          'b--', ...
      [0; thr10sclmatnffwbulk(end,1)], repmat(maxenergyforsim(5), 2,1),
          'r:', ...
      [0; thr10sclmatnffwbulk(end,1)], repmat(maxenergyforsim(6), 2,1),
          'm-.');
title('Storage␣device␣state␣of␣charge,␣no␣loss␣feedforward,␣bulk␣input');
xlabel('Time,␣hours');
ylabel('Energy,␣MWh');
```

```
%xlim([10 15]);
% xlim([0 thr10sclmatnffwbulk(end,1)]);
% ylim([0 22]);
legend('10 min', '15 min', '20 min', '30 min', '10 min size', '15 min
    size',...
        '20 min size', '30 min size','Location', 'NW');
saveas(gcf,'evst_10s_cl_stor_nolossff_wbulk.eps', 'epsc');


figure
plot(thr10sclmatnffwbulk(:,1), pouttotall10sclmatnffwbulk(:,1), 'k-', ...
      thr10sclmatnffwbulk(:,2), pouttotall10sclmatnffwbulk(:,2), 'b--',...
      thr10sclmatnffwbulk(:,3), pouttotall10sclmatnffwbulk(:,3), 'r:', ...
      thr10sclmatnffwbulk(:,4), pouttotall10sclmatnffwbulk(:,1), 'g-.', ...
      thr10sclmatnffwbulk(:,1), thpowall10sclmatnffwbulk(:,1), 'm-', ...
      thr10sclmatnffwbulk(:,2), thpowall10sclmatnffwbulk(:,2), 'k--', ...
      thr10sclmatnffwbulk(:,3), thpowall10sclmatnffwbulk(:,3), 'b:', ...
      thr10sclmatnffwbulk(:,4), thpowall10sclmatnffwbulk(:,4), 'r-.');
title('Power demand and thermal power delivered, 10-second data, no loss
    feedforward, bulk input');
xlabel('Time, hours');
ylabel('Power, MW');
legend('Total Power 10 min','Total Power 15 min','Total Power 20 min', ...
    'Total Power 30 min', 'Thermal 10 min', 'Thermal 15 min', 'Thermal 20
        min', ...
    'Thermal 30 min', 'Location', 'SW');
% xlim([10 15]);
% ylim([-100 100]);
saveas(gcf,'pvstx2_10s_cl_th_tot_nolossff_wbulk.eps', 'epsc');

figure
plot(thr10sclmatnffwbulk, railflagall10sclmatnffwbulk);
title('Flag for full or empty storage, no loss feedforward, bulk input');
xlabel('Time, hours');
ylabel('+1 = full; -1 = empty');
legend('10 min', '15 min', '20 min', '30 min');
ylim([-1.1 1.1]);

% figure
% plot(thr30, socall30, 'k-', thr60, socall60, 'b--', thr90, socall90,
    'r-.');
% title('Storage State of Charge, SOC');
% xlabel('Time, hours');
% ylabel('Energy, MWh');
% legend('30-min window', '60-min window', '90-min window');
%
% figure
```

```
% plot(thr30, balrrall30, 'k-',thr60, balrrall60, 'b--',thr90, balrrall90,
    'r-.', thr30, thrrall30, 'g:', thr60, thrrall60, 'm-', thr90,
    thrrall90, 'c--');
% title('Component ramp rates, bulk input')
% legend('balancing power, 30-min window', 'balancing power, 60-min
    window', 'balancing power, 90-min window', ...
%         'thermal power, 30-min window', 'thermal power, 60-min window',
    'thermal power, 90-min window', 'Location', 'NW')
% xlabel('Time, hours');
% ylabel('Power Ramp Rate, MW/min');
%


% power duration curves, thermal and storage

figure
a=plot(durx10ssim, pstordur10sclmatnffwbulk(:,1), 'k-', ...
      durx10ssim, pstordur10sclmatnffwbulk(:,2), 'b--', ...
      durx10ssim, pstordur10sclmatnffwbulk(:,3), 'r:', ...
      durx10ssim, pstordur10sclmatnffwbulk(:,4), 'm-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(bigdurx10s, sffuzzdur, 'm--');
title('Storage␣Power-Duration␣Curve,␣10-second␣data,␣no␣loss␣feedforward,␣
    bulk␣input');
xlabel('Non-dimensional␣Time');
ylabel('Power,␣MW');
xlim([-0.05 1.05]);
legend('10␣min', '15␣min', '20␣min', '30␣min', 'Original', 'Location',
    'SE');
% ylim([-130 130]);
saveas(gcf,'pdurx2_10s_cl_stor_tot_nolossff_wbulk.eps', 'epsc');

%

figure
a=plot(durx10ssim, thpowdur10sclmatnffwbulk(:,1), 'k-', ...
      durx10ssim, thpowdur10sclmatnffwbulk(:,2), 'b--', ...
      durx10ssim, thpowdur10sclmatnffwbulk(:,3), 'r:', ...
      durx10ssim, thpowdur10sclmatnffwbulk(:,4), 'm-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(bigdurx10s, sffuzzdur, 'r--');
title('Thermal␣Power-Duration␣Curves␣Compared,␣10-second␣data,␣no␣loss␣
    feedforward,␣bulk␣input');
xlabel('Non-dimensional␣Time');
ylabel('Power,␣GW');
legend('10␣min', '15␣min', '20␣min', '30␣min', 'Original', 'Location',
    'SE');
```

```
xlim([-0.05 1.05]);
% ylim([-350 350]);
saveas(gcf,'pdurx2_10s_cl_th_tot_nolossff_wbulk.eps', 'epsc');

%ramp rate duration curves, storage

figure
a = plot(durx10ssim, storrrdur10sclmatnffwbulk(:,1), 'k-', ...
      durx10ssim, storrrdur10sclmatnffwbulk(:,2), 'b--', ...
      durx10ssim, storrrdur10sclmatnffwbulk(:,3), 'r:', ...
      durx10ssim, storrrdur10sclmatnffwbulk(:,4), 'm-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(bigdurx10s, sffuzzrrdur,'r--');
title('Storage␣Ramp␣Rate␣Duration␣Curves␣Compared,␣10-second␣data,␣no␣loss␣
    feedforward,␣bulk␣input');
xlabel('Non-dimensional␣Time');
ylabel('Ramp␣Rate,␣MW/min');
legend('10␣min', '15␣min', '20␣min', '30␣min', 'Original', 'Location',
    'SE');
xlim([-0.05 1.05]);
% ylim([-500, 500]);
saveas(gcf,'rrdurx2_10s_cl_stor_tot_nolossff_wbulk.eps', 'epsc');

%ramp rate duration curves, thermal

figure
a = plot(durx10ssim, thrrdur10sclmatnffwbulk(:,1), 'k-', ...
      durx10ssim, thrrdur10sclmatnffwbulk(:,2), 'b--', ...
      durx10ssim, thrrdur10sclmatnffwbulk(:,3), 'r:', ...
      durx10ssim, thrrdur10sclmatnffwbulk(:,4), 'm-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(bigdurx10s, sffuzzrrdur,'r--');
title('Thermal␣Ramp␣Rate␣Duration␣Curves␣Compared,␣10-second␣data,␣no␣loss␣
    feedforward,␣bulk␣input');
xlabel('Non-dimensional␣Time');
ylabel('Ramp␣Rate,␣MW/min');
legend('10␣min', '15␣min', '20␣min', '30␣min', 'Original', 'Location',
    'SE');
xlim([-0.05 1.05]);
% ylim([-50, 50]);
saveas(gcf,'rrdurx2_10s_cl_th_tot_nolossff_wbulk.eps', 'epsc');

% figure
% a=plot(durx10ssim, storrrdur10scl, 'k-', durx10ssim, thrrdur10scl,
    'b--',bigdurx10s, sffuzzrrdur,'r:');
% title('Ramp-Rate-Duration Curves Compared, 10-second data');
% xlabel('Non-dimensional Time');
```

```
% ylabel('Ramp Rate, MW/min');
% legend('Storage Ramping', 'Thermal Ramping', 'Original Ramping',
    'Location', 'SE');
% xlim([-0.05 1.05]);
% ylim([-100, 100]);
% set(a, 'LineWidth', 2);
% saveas(gcf,'rrdurx3_10s_cl_stor_th_tot_zoomin.eps', 'epsc');
%


% energy duration curves, storage only

figure
%plot(bigdurx, socdur, 'k-', bigdurx, 25, 'r-.')
a=plot(durx10ssim, socdur10sclmatnffwbulk(:,1), 'k-', durx10ssim,
    socdur10sclmatnffwbulk(:,2), 'b--', ...
      durx10ssim, socdur10sclmatnffwbulk(:,3), 'r:', durx10ssim,
          socdur10sclmatnffwbulk(:,4), 'm-.');
hold on;
plot([-0.05 1.05], repmat(maxenergyforsim(3), 2,1), 'k-', [-0.05 1.05],
    repmat(maxenergyforsim(4), 2,1), 'b--', ...
     [-0.05 1.05], repmat(maxenergyforsim(5), 2,1), 'r:', [-0.05 1.05],
        repmat(maxenergyforsim(6), 2,1), 'm-.');
title('Storage␣Energy-Duration␣Curve,␣no␣loss␣feedforward,␣bulk␣input');
xlabel('Non-dimensional␣Time');
ylabel('Energy,␣MWh');
xlim([-0.05 1.05]);
% ylim([0, 22]);
set(a, 'LineWidth', 2);
legend('10␣min', '15␣min', '20␣min', '30␣min', '10␣min␣size', '15␣min␣
    size',...
         '20␣min␣size', '30␣min␣size','Location', 'NW');
saveas(gcf,'edur_10s_cl_stor_nolossff_wbulk.eps', 'epsc');
```

C.2.3 Closed Loop Filtering, With Loss Feedforward

`thesis_makefigs_ch_dividereg_subfile_cl_10s_wlossff.m`

This file calls Simulink to process the closed-loop models.

```
%% Partial script called by thesis_makefigs_ch_dividereg
%% Olivia Leitermann file created 3 Aug 11  MIT LEES

%% Closed-loop filtering with loss ff

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% With Loss Feedforward
```

```
% grabbed from
% run_simulink_stormodel_neisodata_NLFB_15mar11_nobulkenergy_60m.m

model10 = 'virtual_PP_th_stor_w_NL_FB_rails_useasbaseline_10'; % my model
    name
model15 = 'virtual_PP_th_stor_w_NL_FB_rails_useasbaseline_15'; % my model
    name
model20 = 'virtual_PP_th_stor_w_NL_FB_rails_useasbaseline_20'; % my model
    name
model30 = 'virtual_PP_th_stor_w_NL_FB_rails_useasbaseline_30'; % my model
    name
% outputs are: out1: total power; out2: thermal power; out3: storage
% state of charge; out4: load power; out5: storage balancing power; out6:
% storage power

modelvec = [model10; model15; model20; model30];

%filenames for input are neiso_dayn where n = [1 2 3 4 7 8 9] - the days
%without anomalies, median-filtered.


%% first, plot the feedback fxns with the new FB cubic gain

SOCx = linspace(-1, 1, 1e3);

Kg = 80; % 38 MW/norm stor^3
gofSOC = Kg*SOCx.^3;

figure
a=plot(SOCx, gofSOC, 'k-');
xlabel('Normalized␣State␣of␣Charge');
ylabel('Feedback␣Signal,␣MW');
title('Nonlinear␣State␣Feedback␣Function');
set(a, 'LineWidth', 2);
grid on;
saveas(gcf,'NLFB_plot.eps', 'epsc');

%% Set up variables for the sims

load('neiso_all_pad.mat');
%includes 'fulltimesec', 'paddat', 'paddathifreq30', 'paddathifreq60',
    'paddathifreq90'
negpts10s = 360*2; % 2 hours: amount of padded constant data

load('neiso_noncausal_306090_fuzz.mat');

% Parameters: need U(-.005, 0.005) for variable efficiency and U(0.7, 1.3)
% for variable self-discharge power. Changes rather fast, but go with it
% for now
```

```
numrealsimpts10s = 86401;
simfactor10s = 10; % 10 times as many sim pts as real pts

pouttotall10scl = [];
thpowall10scl = [];
socall10scl = [];
ploadall10scl = [];
pbalanceall10scl = [];
pstorall10scl = [];
thrrall10scl = [];
storrrall10scl = [];
balrrall10scl = [];
railflagall10scl = [];
lossspredall10scl = [];
lossfraxall10scl = [];
tall10scl = [];

pouttotall10sclmat = [];
thpowall10sclmat = [];
socall10sclmat = [];
ploadall10sclmat = [];
pbalanceall10sclmat = [];
pstorall10sclmat = [];
thrrall10sclmat = [];
storrrall10sclmat = [];
balrrall10sclmat = [];
railflagall10sclmat = [];
lossspredall10sclmat = [];
lossfraxall10sclmat = [];
tall10sclmat = [];

%% Note on how to get only the output points I want (each second):
%% Configuration Parameters/(Data Import/Export)/Output Options:
%% Choose "Produce Specified Output Only," time is [0:86400]

numfiltsim = 4;
storsize10svec = maxenergyforsim(3:6); % 10, 15, 20, 30 minute filters

numsimptstot = numel(fulltimesec); % cols are all days concat, rows are
    filts
for filtindex = 1:numfiltsim
    filtindex % print so we know where we are
    model = modelvec(filtindex,:);

    % 10 sec data %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    storsize10s = storsize10svec(filtindex);

    % need to reset vars each time because I'm being mean and making them
```

```
% grow
pouttotall10scl = [];
thpowall10scl = [];
socall10scl = [];
ploadall10scl = [];
pbalanceall10scl = [];
pstorall10scl = [];
thrrall10scl = [];
storrrall10scl = [];
balrrall10scl = [];
railflagall10scl = [];
losspredall10scl = [];
lossfraxall10scl = [];
tall10scl = [];

for dayindex = 1:numdays10s

    %ut = [fulltimesec' paddatsift(dayindex,:)' vareff(:,dayindex)
        varPsd(:,dayindex)];
    ut = [fulltimesec' paddathifreq90(dayindex,:)'
        repmat(storsize10s,numsimptstot, 1)];
    timespan = [-7200 86400];
    [t,x,pout_tot, pth, soc, pload, pbalance, pstor, thrr, storrr, ...
        balrr, railflag, losspred, lossfrax, absPin] =
            sim(model,timespan,[],ut);
    % units of power quantities are MW, units of SOC are MJ

    % find where the real part of the sim begins
    ind = find(t>=0, 1, 'first'); % should return the first time step
        where t>=0
    % snag only those values, to start

    % concatenate all waveforms, truncating to the "real" part
    pouttotall10scl = [pouttotall10scl; pout_tot(ind:end)];
    thpowall10scl = [thpowall10scl; pth(ind:end)];
    socall10scl = [socall10scl; soc(ind:end)];
    ploadall10scl = [ploadall10scl; pload(ind:end)];
    pbalanceall10scl = [pbalanceall10scl; pbalance(ind:end)];
    pstorall10scl = [pstorall10scl; pstor(ind:end)];
    thrrall10scl = [thrrall10scl; thrr(ind:end)];
    storrrall10scl = [storrrall10scl; storrr(ind:end)];
    balrrall10scl = [balrrall10scl; balrr(ind:end)];
    railflagall10scl = [railflagall10scl; railflag(ind:end)];
    losspredall10scl = [losspredall10scl; losspred(ind:end)];
    lossfraxall10scl = [lossfraxall10scl; lossfrax(ind:end)];
    tall10scl = [tall10scl; t(ind:end)+(dayindex-1)*numrealsimpts10s];

end
pouttotall10sclmat = [pouttotall10sclmat pouttotall10scl];
```

```
    thpowall10sclmat = [thpowall10sclmat thpowall10scl];
    socall10sclmat = [socall10sclmat socall10scl];
    ploadall10sclmat = [ploadall10sclmat ploadall10scl];
    pbalanceall10sclmat = [pbalanceall10sclmat pbalanceall10scl];
    pstorall10sclmat = [pstorall10sclmat pstorall10scl];
    thrrall10sclmat = [thrrall10sclmat thrrall10scl];
    storrrall10sclmat = [storrrall10sclmat storrrall10scl];
    balrrall10sclmat = [balrrall10sclmat balrrall10scl];
    railflagall10sclmat = [railflagall10sclmat railflagall10scl];
    losspredall10sclmat = [losspredall10sclmat losspredall10scl];
    lossfraxall10sclmat = [lossfraxall10sclmat lossfraxall10scl];
    tall10sclmat = [tall10sclmat tall10scl];

end

% clear the partial vars so I don't accidentally use them:
clear pouttotall10scl thpowall10scl socall10scl ploadall10scl ...
    pbalanceall10scl pstorall10scl thrrall10scl storall10scl ...
    balrrall10scl railflagall10scl losspredall10scl ... %lossfraxall10scl
    tall10scl;

% i think this is ok because of the way I've set up the sims now...
% % flag if I'm not getting the data sampled properly
% mintimestep10scl = min(tall10scl(2:end)-tall10scl(1:end-1));
% if mintimestep10scl < 1
%     'warning: min time step < 1! check simulation parameters'
%     mintimestep10scl
% end

%flag if storage is emptying or filling completely
[railflagmax10scl, railflagind10scl] = max(max(abs(railflagall10sclmat)));
if railflagmax10scl > 0
    'warning: storage is emptying or filling completely! results may be
       inaccurate'
    disp('one time of violations:');
    railflagind10scl
end

thr10sclmat = tall10sclmat/60/60;

%% compute mean efficiencies

lossfraxmeanwlff = mean(lossfraxall10sclmat); % mean over cols for 4
    different filts
%NOT the same as efficiency over the whole time: avg. instant. eff.

lossfraxmeanwlff

%% compute durations
```

```
pouttotdur10sclmat = sort(pouttotall10sclmat); % should properly sort cols
thpowdur10sclmat = sort(thpowall10sclmat);
socdur10sclmat = sort(socall10sclmat);
ploaddur10sclmat = sort(ploadall10sclmat);
pbalancedur10sclmat = sort(pbalanceall10sclmat);
pstordur10sclmat = sort(pstorall10sclmat);
thrrdur10sclmat = sort(thrrall10sclmat);
storrrdur10sclmat = sort(storrrall10sclmat);
balrrdur10sclmat = sort(balrrall10sclmat);
railflagdur10sclmat = sort(railflagall10sclmat);
lossfraxdur10sclmat = sort(lossfraxall10sclmat);

%% Do same simulations using bulk data as input %%%%%%%%%%%%%%%%%%%%%%%%%%%%

pouttotall10sclwbulk = [];
thpowall10sclwbulk = [];
socall10sclwbulk = [];
ploadall10sclwbulk = [];
pbalanceall10sclwbulk = [];
pstorall10sclwbulk = [];
thrrall10sclwbulk = [];
storrrall10sclwbulk = [];
balrrall10sclwbulk = [];
railflagall10sclwbulk = [];
lossspredall10sclwbulk = [];
lossfraxall10sclwbulk = [];
tall10sclwbulk = [];

pouttotall10sclwbulkmat = [];
thpowall10sclwbulkmat = [];
socall10sclwbulkmat = [];
ploadall10sclwbulkmat = [];
pbalanceall10sclwbulkmat = [];
pstorall10sclwbulkmat = [];
thrrall10sclwbulkmat = [];
storrrall10sclwbulkmat = [];
balrrall10sclwbulkmat = [];
railflagall10sclwbulkmat = [];
lossspredall10sclwbulkmat = [];
lossfraxall10sclwbulkmat = [];
tall10sclwbulkmat = [];

%% Note on how to get only the output points I want (each second):
%% Configuration Parameters/(Data Import/Export)/Output Options:
%% Choose "Produce Specified Output Only," time is [0:86400]

numfiltsim = 4;
```

```
storsize10svec = maxenergyforsim(3:6); % 10, 15, 20, 30 minute filters

numsimptstot = numel(fulltimesec); % cols are all days concat, rows are
    filts
for filtindex = 1:numfiltsim
    filtindex % print so we know where we are
    model = modelvec(filtindex,:);

    % 10 sec data %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    storsize10s = storsize10svec(filtindex);

    % need to reset vars each time because I'm being mean and making them
    % grow
    pouttotall10sclwbulk = [];
    thpowall10sclwbulk = [];
    socall10sclwbulk = [];
    ploadall10sclwbulk = [];
    pbalanceall10sclwbulk = [];
    pstorall10sclwbulk = [];
    thrrall10sclwbulk = [];
    storrrall10sclwbulk = [];
    balrrall10sclwbulk = [];
    railflagall10sclwbulk = [];
    lossspredall10sclwbulk = [];
    lossfraxall10sclwbulk = [];
    tall10sclwbulk = [];

    for dayindex = 1:numdays10s

        %ut = [fulltimesec' paddatsift(dayindex,:)' vareff(:,dayindex)
            varPsd(:,dayindex)];
        ut = [fulltimesec' paddathifreq90(dayindex,:)'
            repmat(storsize10s,numsimptstot, 1)];
        timespan = [-7200 86400];
        [t,x,pout_tot, pth, soc, pload, pbalance, pstor, thrr, storrr, ...
            balrr, railflag, losspred, lossfrax, absPin] =
                sim(model,timespan,[],ut);
        % units of power quantities are MW, units of SOC are MJ

        % find where the real part of the sim begins
        ind = find(t>=0, 1, 'first'); % should return the first time step
            where t>=0
        % snag only those values, to start

        % concatenate all waveforms, truncating to the "real" part
        pouttotall10sclwbulk = [pouttotall10sclwbulk; pout_tot(ind:end)];
        thpowall10sclwbulk = [thpowall10sclwbulk; pth(ind:end)];
        socall10sclwbulk = [socall10sclwbulk; soc(ind:end)];
        ploadall10sclwbulk = [ploadall10sclwbulk; pload(ind:end)];
```

```
        pbalanceall10sclwbulk = [pbalanceall10sclwbulk; pbalance(ind:end)];
        pstorall10sclwbulk = [pstorall10sclwbulk; pstor(ind:end)];
        thrrall10sclwbulk = [thrrall10sclwbulk; thrr(ind:end)];
        storrrall10sclwbulk = [storrrall10sclwbulk; storrr(ind:end)];
        balrrall10sclwbulk = [balrrall10sclwbulk; balrr(ind:end)];
        railflagall10sclwbulk = [railflagall10sclwbulk; railflag(ind:end)];
        lossspredall10sclwbulk = [lossspredall10sclwbulk; lossspred(ind:end)];
        lossfraxall10sclwbulk = [lossfraxall10sclwbulk; lossfrax(ind:end)];
        tall10sclwbulk = [tall10sclwbulk;
            t(ind:end)+(dayindex-1)*numrealsimpts10s];

    end
    pouttotall10sclwbulkmat = [pouttotall10sclwbulkmat
        pouttotall10sclwbulk];
    thpowall10sclwbulkmat = [thpowall10sclwbulkmat thpowall10sclwbulk];
    socall10sclwbulkmat = [socall10sclwbulkmat socall10sclwbulk];
    ploadall10sclwbulkmat = [ploadall10sclwbulkmat ploadall10sclwbulk];
    pbalanceall10sclwbulkmat = [pbalanceall10sclwbulkmat
        pbalanceall10sclwbulk];
    pstorall10sclwbulkmat = [pstorall10sclwbulkmat pstorall10sclwbulk];
    thrrall10sclwbulkmat = [thrrall10sclwbulkmat thrrall10sclwbulk];
    storrrall10sclwbulkmat = [storrrall10sclwbulkmat storrrall10sclwbulk];
    balrrall10sclwbulkmat = [balrrall10sclwbulkmat balrrall10sclwbulk];
    railflagall10sclwbulkmat = [railflagall10sclwbulkmat
        railflagall10sclwbulk];
    lossspredall10sclwbulkmat = [lossspredall10sclwbulkmat
        lossspredall10sclwbulk];
    lossfraxall10sclwbulkmat = [lossfraxall10sclwbulkmat
        lossfraxall10sclwbulk];
    tall10sclwbulkmat = [tall10sclwbulkmat tall10sclwbulk];

end

% clear the partial vars so I don't accidentally use them:
clear pouttotall10sclwbulk thpowall10sclwbulk socall10sclwbulk
    ploadall10sclwbulk ...
    pbalanceall10sclwbulk pstorall10sclwbulk thrrall10sclwbulk
        storall10sclwbulk ...
    balrrall10sclwbulk railflagall10sclwbulk lossspredall10sclwbulk ...
        %lossfraxall10sclwbulk
    tall10sclwbulk;

% i think this is ok because of the way I've set up the sims now...
% % flag if I'm not getting the data sampled properly
% mintimestep10sclwbulk =
    min(tall10sclwbulk(2:end)-tall10sclwbulk(1:end-1));
% if mintimestep10sclwbulk < 1
%     'warning: min time step < 1! check simulation parameters'
%     mintimestep10sclwbulk
```

```
% end

%flag if storage is emptying or filling completely
[railflagmax10sclwbulk, railflagind10sclwbulk] =
    max(max(abs(railflagall10sclwbulkmat)));
if railflagmax10sclwbulk > 0
    'warning: storage is emptying or filling completely! results may be
        inaccurate'
    disp('one time of violations:');
    railflagind10sclwbulk
end

thr10sclwbulkmat = tall10sclwbulkmat/60/60;

%% compute mean efficiencies

lossfraxmeanwlffwbulk = mean(lossfraxall10sclwbulkmat); % mean over cols
    for 4 different filts
%NOT the same as efficiency over the whole time: avg. instant. eff.

lossfraxmeanwlffwbulk

%% compute durations


pouttotdur10sclwbulkmat = sort(pouttotall10sclwbulkmat); % should properly
    sort cols
thpowdur10sclwbulkmat = sort(thpowall10sclwbulkmat);
socdur10sclwbulkmat = sort(socall10sclwbulkmat);
ploaddur10sclwbulkmat = sort(ploadall10sclwbulkmat);
pbalancedur10sclwbulkmat = sort(pbalanceall10sclwbulkmat);
pstordur10sclwbulkmat = sort(pstorall10sclwbulkmat);
thrrdur10sclwbulkmat = sort(thrrall10sclwbulkmat);
storrrdur10sclwbulkmat = sort(storrrall10sclwbulkmat);
balrrdur10sclwbulkmat = sort(balrrall10sclwbulkmat);
railflagdur10sclwbulkmat = sort(railflagall10sclwbulkmat);
lossfraxdur10sclwbulkmat = sort(lossfraxall10sclwbulkmat);


%% Plot simulation results: 10 sec data, fuzz input %%%%%%%%%%%%%%%%%%%%%%

durx10ssim = linspace(0, 1, numdays10s*numrealsimpts10s);

% time series results for checking

figure
plot(thr10sclmat(:,1), pstorall10sclmat(:,1), 'k-', ...
    thr10sclmat(:,2), pstorall10sclmat(:,2), 'b--', ...
    thr10sclmat(:,3), pstorall10sclmat(:,3), 'r:', ...
```

```
    thr10sclmat(:,4), pstorall10sclmat(:,4), 'g-.');
title('Power␣out␣of␣storage␣device');
xlabel('Time,␣hours');
ylabel('Power,␣MW');
legend('10␣min', '15␣min', '20␣min', '30␣min');
%xlim([0 10]);
ylim([-50 50]);
saveas(gcf,'pvst_10s_cl_stor.eps', 'epsc');

figure
plot(thr10sclmat(:,1), socall10sclmat(:,1), 'k-', ...
    thr10sclmat(:,2), socall10sclmat(:,2), 'b--', ...
    thr10sclmat(:,3), socall10sclmat(:,3), 'r:', ...
    thr10sclmat(:,4), socall10sclmat(:,4), 'g-.');
title('Storage␣device␣state␣of␣charge');
xlabel('Time,␣hours');
ylabel('Energy,␣MWh');
%xlim([0 10]);
%ylim([-50 50]);
legend('10␣min', '15␣min', '20␣min', '30␣min');
saveas(gcf,'evst_10s_cl_stor.eps', 'epsc');


figure
plot(thr10sclmat(:,1), pouttotall10sclmat(:,1), 'k-', ...
    thr10sclmat(:,2), pouttotall10sclmat(:,2), 'b--',...
    thr10sclmat(:,3), pouttotall10sclmat(:,3), 'r:',...
    thr10sclmat(:,4), pouttotall10sclmat(:,1), 'g-.', ...
    thr10sclmat(:,1), thpowall10sclmat(:,1), 'm-', ...
    thr10sclmat(:,2), thpowall10sclmat(:,2), 'k--', ...
    thr10sclmat(:,3), thpowall10sclmat(:,3), 'b:', ...
    thr10sclmat(:,4), thpowall10sclmat(:,4), 'r-.');
title('Power␣demand␣and␣thermal␣power␣delivered,␣10-second␣data');
xlabel('Time,␣hours');
ylabel('Power,␣MW');
legend('Total␣Power␣10␣min','Total␣Power␣15␣min','Total␣Power␣20␣min',...
    'Total␣Power␣30␣min', 'Thermal␣10␣min', 'Thermal␣15␣min', ...
    'Thermal␣20␣min', 'Thermal␣30␣min', 'Location', 'NW');
xlim([0 10]);
ylim([-100 100]);
saveas(gcf,'pvstx2_10s_cl_th_tot.eps', 'epsc');

figure
plot(thr10sclmat, railflagall10sclmat);
title('Flag␣for␣full␣or␣empty␣storage');
xlabel('Time,␣hours');
ylabel('+1␣=␣full;␣-1␣=␣empty');
legend('10␣min', '15␣min', '20␣min', '30␣min');
ylim([-1.1 1.1]);
```

```
figure
a=plot(durx10ssim, pstordur10sclmat(:,1), 'k-', ...
    durx10ssim, pstordur10sclmat(:,2), 'b--', ...
    durx10ssim, pstordur10sclmat(:,3), 'r:', ...
    durx10ssim, pstordur10sclmat(:,4), 'g-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(bigdurx10s, sffuzzdur, 'm--');
title('Storage Power-Duration Curve, 10-second data');
xlabel('Non-dimensional Time');
ylabel('Power, MW');
xlim([-0.05 1.05]);
legend('10 min', '15 min', '20 min', '30 min', 'Original', 'Location',
    'SE');
ylim([-130 130]);
saveas(gcf,'pdurx2_10s_cl_stor_tot.eps', 'epsc');

%

figure
a=plot(durx10ssim, thpowdur10sclmat(:,1), 'k-', ...
    durx10ssim, thpowdur10sclmat(:,2), 'b--', ...
    durx10ssim, thpowdur10sclmat(:,3), 'r:', ...
    durx10ssim, thpowdur10sclmat(:,4), 'g-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(bigdurx10s, sffuzzdur, 'r--');
title('Thermal Power-Duration Curves Compared, 10-second data');
xlabel('Non-dimensional Time');
ylabel('Power, GW');
legend('10 min', '15 min', '20 min', '30 min', 'Original', 'Location',
    'SE');
xlim([-0.05 1.05]);
ylim([-200 200]);
saveas(gcf,'pdurx2_10s_cl_th_tot.eps', 'epsc');

%ramp rate duration curves, storage

figure
a = plot(durx10ssim, storrrdur10sclmat(:,1), 'k-', ...
    durx10ssim, storrrdur10sclmat(:,2), 'b--', ...
    durx10ssim, storrrdur10sclmat(:,3), 'r:', ...
    durx10ssim, storrrdur10sclmat(:,4), 'g-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(bigdurx10s, sffuzzrrdur,'r--');
title('Storage Ramp Rate Duration Curves Compared, 10-second data');
xlabel('Non-dimensional Time');
```

```
ylabel('Ramp␣Rate,␣MW/min');
legend('10␣min', '15␣min', '20␣min', '30␣min', 'Original', 'Location',
    'SE');
xlim([-0.05 1.05]);
ylim([-500, 500]);
saveas(gcf,'rrdur_10s_cl_stor_tot.eps', 'epsc');

%ramp rate duration curves, thermal

figure
a = plot(durx10ssim, thrrdur10sclmat(:,1), 'k-', durx10ssim,
    thrrdur10sclmat(:,2), 'b--', ...
     durx10ssim, thrrdur10sclmat(:,3), 'r:', durx10ssim,
        thrrdur10sclmat(:,4), 'm-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(durx10ssim, balrrdur10sclmat(:,1), 'g-', durx10ssim,
    balrrdur10sclmat(:,2), 'k--', ...
     durx10ssim, balrrdur10sclmat(:,3), 'r:', durx10ssim,
        balrrdur10sclmat(:,4), 'b-.');
set(a, 'LineWidth', 2);
a = plot(bigdurx10s, sffuzzrrdur,'m--');
title('Thermal␣Ramp␣Rate␣Duration␣Curves␣Compared,␣10-second␣data');
xlabel('Non-dimensional␣Time');
ylabel('Ramp␣Rate,␣MW/min');
legend('Thermal␣10␣min', 'Thermal␣15␣min', 'Thermal␣20␣min', 'Thermal␣30␣
    min', ...
         'Balancing␣10␣min', 'Balancing␣15␣min', 'Balancing␣20␣min',
            'Balancing␣30␣min', ...
         'Original', 'Location', 'SE');
xlim([-0.05 1.05]);
ylim([-180, 180]);
saveas(gcf,'rrdur_10s_cl_th_bal_tot.eps', 'epsc');

% energy duration curves, storage only

figure
%plot(bigdurx, socdur, 'k-', bigdurx, 25, 'r-.')
a=plot(durx10ssim, socdur10sclmat(:,1), 'k-', durx10ssim,
    socdur10sclmat(:,2), 'b--',...
     durx10ssim, socdur10sclmat(:,3), 'r:', durx10ssim, socdur10sclmat(:,4),
        'm-.');
hold on;
plot([-0.05 1.05], repmat(maxenergyforsim(3), 2,1), 'k-', [-0.05 1.05],
    repmat(maxenergyforsim(4), 2,1), 'b--', ...
     [-0.05 1.05], repmat(maxenergyforsim(5), 2,1), 'r:', [-0.05 1.05],
        repmat(maxenergyforsim(6), 2,1), 'm-.');
title('Storage␣Energy-Duration␣Curve');
xlabel('Non-dimensional␣Time');
```

```
ylabel('Energy, MWh');
xlim([-0.05 1.05]);
ylim([0, 22]);
set(a, 'LineWidth', 2);
legend('10 min', '15 min', '20 min', '30 min', '10 min size', '15 min
    size',...
        '20 min size', '30 min size','Location', 'NW');
saveas(gcf,'edur_10s_cl_stor.eps', 'epsc');

% <|dP/dt|> vs Wmax
cl10swmax = socdur10sclmat(end,:)-socdur10sclmat(1,:);
cl10saadpdt = mean(abs(thrrdur10sclmat));

figure
a = plot(cheb10swmax, cheb10saadpdt, 'k-s');
set(a, 'LineWidth', 2);
hold on;
a = plot(wmaxsocnff, dpdtmeanabsnff, 'r:o');
set(a, 'LineWidth', 2);
a = plot(cl10swmax, cl10saadpdt, 'b--+');
set(a, 'LineWidth', 2);
a = plot(0, orig10saadpdt, 'mx');
set(a, 'LineWidth', 2);
xlabel('Maximum Energy Required, MWh');
ylabel('Mean Absolute Ramping, MW/min');
title('Comparison of Chebyshev filters with 10-second data');
legend('Open loop', 'Closed-loop, no loss feed-forward', 'Closed loop, with
    loss feed-forward', 'Original data', 'Location', 'NE');
saveas(gcf,'wvsdpdt_10s_5cheb_cl.eps', 'epsc');

%% Plot results with bulk input also %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% time series results for checking

figure
plot(thr10sclwbulkmat(:,1), pstorall10sclwbulkmat(:,1), 'k-', ...
    thr10sclwbulkmat(:,2), pstorall10sclwbulkmat(:,2), 'b--', ...
    thr10sclwbulkmat(:,3), pstorall10sclwbulkmat(:,3), 'r:', ...
    thr10sclwbulkmat(:,4), pstorall10sclwbulkmat(:,4), 'g-.');
title('Power out of storage device, with bulk input');
xlabel('Time, hours');
ylabel('Power, MW');
legend('10 min', '15 min', '20 min', '30 min');
%xlim([0 10]);
ylim([-50 50]);
saveas(gcf,'pvst_10s_cl_stor_wbulk.eps', 'epsc');

figure
plot(thr10sclwbulkmat(:,1), socall10sclwbulkmat(:,1), 'k-', ...
```

```
    thr10sclwbulkmat(:,2), socall10sclwbulkmat(:,2), 'b--', ...
    thr10sclwbulkmat(:,3), socall10sclwbulkmat(:,3), 'r:', ...
    thr10sclwbulkmat(:,4), socall10sclwbulkmat(:,4), 'g-.');
title('Storage device state of charge, with bulk input');
xlabel('Time, hours');
ylabel('Energy, MWh');
%xlim([0 10]);
%ylim([-50 50]);
legend('10 min', '15 min', '20 min', '30 min');
saveas(gcf,'evst_10s_cl_stor_wbulk.eps', 'epsc');


figure
plot(thr10sclwbulkmat(:,1), pouttotall10sclwbulkmat(:,1), 'k-', ...
    thr10sclwbulkmat(:,2), pouttotall10sclwbulkmat(:,2), 'b--',...
    thr10sclwbulkmat(:,3), pouttotall10sclwbulkmat(:,3), 'r:',...
    thr10sclwbulkmat(:,4), pouttotall10sclwbulkmat(:,1), 'g-.', ...
    thr10sclwbulkmat(:,1), thpowall10sclwbulkmat(:,1), 'm-', ...
    thr10sclwbulkmat(:,2), thpowall10sclwbulkmat(:,2), 'k--', ...
    thr10sclwbulkmat(:,3), thpowall10sclwbulkmat(:,3), 'b:', ...
    thr10sclwbulkmat(:,4), thpowall10sclwbulkmat(:,4), 'r-.');
title('Power demand and thermal power delivered, 10-second data, with bulk
   input');
xlabel('Time, hours');
ylabel('Power, MW');
legend('Total Power 10 min','Total Power 15 min','Total Power 20 min',...
    'Total Power 30 min', 'Thermal 10 min', 'Thermal 15 min', ...
    'Thermal 20 min', 'Thermal 30 min', 'Location', 'NW');
xlim([0 10]);
ylim([-100 100]);
saveas(gcf,'pvstx2_10s_cl_th_tot_wbulk.eps', 'epsc');

figure
plot(thr10sclwbulkmat, railflagall10sclwbulkmat);
title('Flag for full or empty storage, with bulk input');
xlabel('Time, hours');
ylabel('+1 = full; -1 = empty');
legend('10 min', '15 min', '20 min', '30 min');
ylim([-1.1 1.1]);

figure
a=plot(durx10ssim, pstordur10sclwbulkmat(:,1), 'k-', ...
    durx10ssim, pstordur10sclwbulkmat(:,2), 'b--', ...
    durx10ssim, pstordur10sclwbulkmat(:,3), 'r:', ...
    durx10ssim, pstordur10sclwbulkmat(:,4), 'g-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(bigdurx10s, sffuzzdur, 'm--');
title('Storage Power-Duration Curve, 10-second data, with bulk input');
```

```
xlabel('Non-dimensional Time');
ylabel('Power, MW');
xlim([-0.05 1.05]);
legend('10 min', '15 min', '20 min', '30 min', 'Original', 'Location',
    'SE');
ylim([-130 130]);
saveas(gcf,'pdurx2_10s_cl_stor_tot_wbulk.eps', 'epsc');

%

figure
a=plot(durx10ssim, thpowdur10sclwbulkmat(:,1), 'k-', ...
    durx10ssim, thpowdur10sclwbulkmat(:,2), 'b--', ...
    durx10ssim, thpowdur10sclwbulkmat(:,3), 'r:', ...
    durx10ssim, thpowdur10sclwbulkmat(:,4), 'g-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(bigdurx10s, sffuzzdur, 'r--');
title('Thermal Power-Duration Curves Compared, 10-second data, with bulk
    input');
xlabel('Non-dimensional Time');
ylabel('Power, GW');
legend('10 min', '15 min', '20 min', '30 min', 'Original', 'Location',
    'SE');
xlim([-0.05 1.05]);
ylim([-200 200]);
saveas(gcf,'pdurx2_10s_cl_th_tot_wbulk.eps', 'epsc');

%ramp rate duration curves, storage

figure
a = plot(durx10ssim, storrrdur10sclwbulkmat(:,1), 'k-', ...
    durx10ssim, storrrdur10sclwbulkmat(:,2), 'b--', ...
    durx10ssim, storrrdur10sclwbulkmat(:,3), 'r:', ...
    durx10ssim, storrrdur10sclwbulkmat(:,4), 'g-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(bigdurx10s, sffuzzrrdur,'r--');
title('Storage Ramp Rate Duration Curves Compared, 10-second data, with
    bulk input');
xlabel('Non-dimensional Time');
ylabel('Ramp Rate, MW/min');
legend('10 min', '15 min', '20 min', '30 min', 'Original', 'Location',
    'SE');
xlim([-0.05 1.05]);
ylim([-500, 500]);
saveas(gcf,'rrdur_10s_cl_stor_tot_wbulk.eps', 'epsc');

%ramp rate duration curves, thermal
```

```
figure
a = plot(durx10ssim, thrrdur10sclwbulkmat(:,1), 'k-', durx10ssim,
    thrrdur10sclwbulkmat(:,2), 'b--', ...
     durx10ssim, thrrdur10sclwbulkmat(:,3), 'r:', durx10ssim,
        thrrdur10sclwbulkmat(:,4), 'm-.');
set(a, 'LineWidth', 2);
hold on;
a = plot(durx10ssim, balrrdur10sclwbulkmat(:,1), 'g-', durx10ssim,
    balrrdur10sclwbulkmat(:,2), 'k--', ...
     durx10ssim, balrrdur10sclwbulkmat(:,3), 'r:', durx10ssim,
        balrrdur10sclwbulkmat(:,4), 'b-.');
set(a, 'LineWidth', 2);
a = plot(bigdurx10s, sffuzzrrdur,'m--');
title('Thermal␣Ramp␣Rate␣Duration␣Curves␣Compared,␣10-second␣data,␣with␣
    bulk␣input');
xlabel('Non-dimensional␣Time');
ylabel('Ramp␣Rate,␣MW/min');
legend('Thermal␣10␣min', 'Thermal␣15␣min', 'Thermal␣20␣min', 'Thermal␣30␣
    min', ...
         'Balancing␣10␣min', 'Balancing␣15␣min', 'Balancing␣20␣min',
            'Balancing␣30␣min', ...
         'Original', 'Location', 'SE');
xlim([-0.05 1.05]);
ylim([-180, 180]);
saveas(gcf,'rrdur_10s_cl_th_bal_tot_wbulk.eps', 'epsc');

% energy duration curves, storage only

figure
%plot(bigdurx, socdur, 'k-', bigdurx, 25, 'r-.')
a=plot(durx10ssim, socdur10sclwbulkmat(:,1), 'k-', durx10ssim,
    socdur10sclwbulkmat(:,2), 'b--',...
     durx10ssim, socdur10sclwbulkmat(:,3), 'r:', durx10ssim,
        socdur10sclwbulkmat(:,4), 'm-.');
hold on;
plot([-0.05 1.05], repmat(maxenergyforsim(3), 2,1), 'k-', [-0.05 1.05],
    repmat(maxenergyforsim(4), 2,1), 'b--', ...
     [-0.05 1.05], repmat(maxenergyforsim(5), 2,1), 'r:', [-0.05 1.05],
        repmat(maxenergyforsim(6), 2,1), 'm-.');
title('Storage␣Energy-Duration␣Curve,␣with␣bulk␣input');
xlabel('Non-dimensional␣Time');
ylabel('Energy,␣MWh');
xlim([-0.05 1.05]);
ylim([0, 22]);
set(a, 'LineWidth', 2);
legend('10␣min', '15␣min', '20␣min', '30␣min', '10␣min␣size', '15␣min␣
    size',...
         '20␣min␣size', '30␣min␣size','Location', 'NW');
```

```
saveas(gcf,'edur_10s_cl_stor_wbulk.eps', 'epsc');

% <|dP/dt|> vs Wmax
cl10swmax = socdur10sclwbulkmat(end,:)-socdur10sclwbulkmat(1,:);
cl10saadpdt = mean(abs(thrrdur10sclwbulkmat));

figure
a = plot(cheb10swmax, cheb10saadpdt, 'k-s');
set(a, 'LineWidth', 2);
hold on;
a = plot(wmaxsocnff, dpdtmeanabsnff, 'r:o');
set(a, 'LineWidth', 2);
a = plot(cl10swmax, cl10saadpdt, 'b--+');
set(a, 'LineWidth', 2);
a = plot(0, orig10saadpdt, 'mx');
set(a, 'LineWidth', 2);
xlabel('Maximum␣Energy␣Required,␣MWh');
ylabel('Mean␣Absolute␣Ramping,␣MW/min');
title('Comparison␣of␣Chebyshev␣filters␣with␣10-second␣data,␣with␣bulk␣
    input');
legend('Open␣loop', 'Closed-loop,␣no␣loss␣feed-forward', 'Closed␣loop,␣with␣
    loss␣feed-forward', 'Original␣data', 'Location', 'NE');
saveas(gcf,'wvsdpdt_10s_5cheb_cl_wbulk.eps', 'epsc');
```

C.3 Importance of Signal Characteristics

C.3.1 Load and Prepare Data

`thesis_makefigs_ch_newpjmdata_subfile_load_prep_data.m`

Similar to Subsection C.1.2.

```
%% Olivia Leitermann  MIT LEES  10 jan 12
%% subfile to make thesis figures, chapter (new pjm data), load and prep
    data

%% load 10-sec data

loaddata10sec = load('NEISO_AGC_NEload.csv');
% columns are 10-sec (!) samples of NE load on 10 non-consecutive days,
% midnight to midnight.  first 3 cols are sundays 23 mar 08, 28 sept 08, 30
% aug 09, followed by 2 saturdays, 11 apr 09, 2 jan 10, followed by
% mon-friday 16 jun 08, 26 may 09, 29 jul 09, 1 oct 09, 22 feb 08.  Total
% system load MW
```

```
% use a 5-point median filter to get rid of spikes
flddata10sec = medfilt1(loaddata10sec, 5);

% sift out days if I need to...
pickdays = [1 2 3 4 7 8 9];
numdays10s = numel(pickdays);

sload10sec = loaddata10sec(:,pickdays);
sfload10sec = flddata10sec(:,pickdays);
sfload10secvec = reshape(sfload10sec, [], 1);

numpts10s = numel(sload10sec);
numpts1day10s = numpts10s/numdays10s;

%% find ramp rate of original 10-sec data with and without median filter

init10s = zeros(size(sload10sec));

sloadrr10sec = init10s;
sfloadrr10sec = init10s;

for index = 1:numdays10s

    sloadrr10sec(:,index) = rr10sec(sload10sec(:,index));
    sfloadrr10sec(:,index) = rr10sec(sfload10sec(:,index));

end

sload10secvec = reshape(sload10sec, numel(sload10sec), 1);
sfload10secvec = reshape(sfload10sec, numel(sfload10sec), 1);

sloadrr10secvec = reshape(sloadrr10sec, numel(sloadrr10sec), 1);
sfloadrr10secvec = reshape(sfloadrr10sec, numel(sfloadrr10sec), 1);

sload10secdur = sort(sload10secvec);
sfload10secdur = sort(sfload10secvec);
sloadrr10secdur = sort(sloadrr10secvec);
sfloadrr10secdur = sort(sfloadrr10secvec);

meansfload = mean(sfload10secvec);

%% Find the bulk energy of the curves and subtract to get fuzz only

window90 = 90*6+1; % number of samples to make up 90 min, want an odd number
filtvec90 = ones(window90, 1)./window90;

init = zeros(size(sload10sec));

sfavg90mat = init;
```

```
sffuzz90mat = init;

savg90mat = init;
sfuzz90mat = init;

for index = 1:numdays10s

    offset = sfload10sec(1,index) - sfload10sec(end,index);
    [waitforit, zi] = filter(filtvec90,1, (sfload10sec(:,index) + offset));
        % calculate appropriate initial conditions: jigger so it ends where
        the next one starts
    working = filter(filtvec90,1,sfload10sec(:,index), zi); % use ICs from
        as if two identical days had been appended
    % BUT need to slide it so that the data points are centered in the
    % boxcars
    chunk = working(1:(window90+1)/2)-offset;
    working = [working((window90+3)/2:end); chunk];
    sfavg90mat(:,index) = working;
    sffuzz90mat(:,index) = sfload10sec(:,index) - working;

    [waitforit, zi] = filter(filtvec90,1, (sload10sec(:,index) + offset));
        % calculate appropriate initial conditions: jigger so it ends where
        the next one starts
    working = filter(filtvec90,1,sload10sec(:,index), zi); % use ICs from
        as if two identical days had been appended
    % BUT need to slide it so that the data points are centered in the
    % boxcars
    chunk = working(1:(window90+1)/2)-offset;
    working = [working((window90+3)/2:end); chunk];
    savg90mat(:,index) = working;
    sfuzz90mat(:,index) = sload10sec(:,index) - working;

end

sfavgvec = reshape(sfavg90mat, numpts10s, 1);
sffuzzvec = reshape(sffuzz90mat, numpts10s, 1);

savgvec = reshape(savg90mat, numpts10s, 1);
sfuzzvec = reshape(sfuzz90mat, numpts10s, 1);

sfavgdur = sort(sfavgvec);
sffuzzdur = sort(sffuzzvec);

savgdur = sort(savgvec);
sfuzzdur = sort(sfuzzvec);

%% Get all the ramping and energy characteristics of the separated curves

for index = 1:numdays10s
```

```
        sfavgrr(:,index) = rr10sec(sfavg90mat(:,index));
        sffuzzrr(:,index) = rr10sec(sffuzz90mat(:,index));

        savgrr(:,index) = rr10sec(savg90mat(:,index));
        sfuzzrr(:,index) = rr10sec(sfuzz90mat(:,index));

        sffuzzint(:,index) = int10sec(sffuzz90mat(:,index));
        sfuzzint(:,index) = int10sec(sfuzz90mat(:,index));

end

sfavgrrvec = reshape(sfavgrr, numpts10s, 1);
sffuzzrrvec = reshape(sffuzzrr, numpts10s, 1);

savgrrvec = reshape(savgrr, numpts10s, 1);
sfuzzrrvec = reshape(sfuzzrr, numpts10s, 1);

sffuzzintvec = reshape(sffuzzint, numpts10s, 1);
sfuzzintvec = reshape(sfuzzint, numpts10s, 1);

sfavgrrdur = sort(sfavgrrvec);
sffuzzrrdur = sort(sffuzzrrvec);

savgrrdur = sort(savgrrvec);
sfuzzrrdur = sort(sfuzzrrvec);

sffuzzintdur = sort(sffuzzintvec);
sfuzzintdur = sort(sfuzzintvec);

sffmeanabsrrorig = mean(abs(sffuzzrrvec));
sffrmsrrorig = sqrt(mean(sffuzzrrvec.^2));

%% load data, Jun, Sep 1-7, 2010; Jan, Mar 1-7, 2011

numwkdays = 7;
numwks = 4;

filenames = {'rto-reg-data-external-1-7jun-2010.csv',
    'rto-reg-data-external-1-7sep-2010.csv', ...
                'rto-reg-data-external-1-7jan-2011.csv',
                    'rto-reg-data-external-1-7mar-2011.csv'};

tregpjmmat = [];
tregpjmintmat = [];
tregpjmrrmat = [];

for index = 1:numwks
    %index
```

```
    fid = fopen(filenames{index});
    treg4s_cell = textscan(fid, '%*s␣%f␣%f␣%f␣%*s␣%*s␣%*s␣%*s', ...
        'Delimiter', ',', ...
        'CommentStyle', '%'); % only want last 3 cols: dispatched reg (MW),
            total online reg capacity (MW), quotient
    % (also is some extra columns in the file; ignore those, they're empty
    % anyway)
    %"Reg. Test Data from",01/01/10,"to",1/2/2010
    %"Time","RTO RegA","RTO Treg","% raise/lower"
    % data are every 4 seconds and start with midnight on 1 xxx 201x.
    fclose(fid);
    %numptsnow = numel(marreg4s_cell{1})
    tregpjmmat = [tregpjmmat treg4s_cell{1}]; % traditonal reg

    % compute ramp rates and energy
    tregpjmintmat = [tregpjmintmat int4sec(treg4s_cell{1})];
    tregpjmrrmat = [tregpjmrrmat rr4sec(treg4s_cell{1})];

end

numpts1wk = numel(tregpjmmat)/numwks;

% slide storage to have a minimum at zero and start at the same place each
% week
tregpjmintmat = tregpjmintmat - min(min(tregpjmintmat));

tregpjmvec = reshape(tregpjmmat, [], 1);
tregpjmintvec = reshape(tregpjmintmat, [], 1);
tregpjmrrvec = reshape(tregpjmrrmat, [], 1);

tmeanabsrrorig = mean(abs(tregpjmrrvec));
trmsrrorig = sqrt(mean(tregpjmrrvec.^2));

load('meanbulkpjmload.mat');


%% compute durations

tregpjmdur = sort(tregpjmvec);
tregpjmintdur = sort(tregpjmintvec);
tregpjmrrdur = sort(tregpjmrrvec);

%% Do median filtering with 5 and 13 points (just in case)

for index = 1:numwks
    tregmf5(:,index) = medfilt1(tregpjmmat(:,index),5);
    tregmf13(:,index) = medfilt1(tregpjmmat(:,index),13);

    tregmf5int(:,index) = int4sec(tregmf5(:,index));
```

```
    tregmf13int(:,index) = int4sec(tregmf13(:,index));

    tregmf5rr(:,index) = rr4sec(tregmf5(:,index));
    tregmf13rr(:,index) = rr4sec(tregmf13(:,index));

end

tregmf5vec = reshape(tregmf5, [],1);
tregmf13vec = reshape(tregmf13, [],1);

tregmf5rrvec = reshape(tregmf5rr, [],1);
tregmf13rrvec = reshape(tregmf13rr, [],1);

tregmf5intvec = reshape(tregmf5int, [],1);
tregmf13intvec = reshape(tregmf13int, [],1);

tregmf5dur = sort(tregmf5vec);
tregmf13dur = sort(tregmf13vec);

tregmf5rrdur = sort(tregmf5rrvec);
tregmf13rrdur = sort(tregmf13rrvec);

tregmf5intdur = sort(tregmf5intvec);
tregmf13intdur = sort(tregmf13intvec);

tregmf5meanabsrr = mean(abs(tregmf5rrvec));
tregmf5rmsrr = sqrt(mean(tregmf5rrvec.^2));

tregmf13meanabsrr = mean(abs(tregmf13rrvec));
tregmf13rmsrr = sqrt(mean(tregmf13rrvec.^2));

%% resample data to 10s

tregresamp10smat = resample(tregpjmmat,4,10); % uses 10 samples on either
% side of point, w/ LPF to avoid aliasing, downsample to 10 sec/sample

%trim first and last 10 samples to be safe
tregresamp10smat = tregresamp10smat(11:end-10,:);

numresamppts = size(tregresamp10smat);
numresamppts = numresamppts(1);

% compute ramp rates and integrals
for index = 1:numwks
    tregresamp10srrmat(:,index) = rr10sec(tregresamp10smat(:,index));
    tregresamp10sintmat(:,index) = int10sec(tregresamp10smat(:,index));
end

figure
```

```
plot([0:10:10000], tregresamp10smat(1:1001,1), 'k-');
hold on;
plot([0:4:10000], tregpjmmat(1:2501,1), 'b--');
title('reg␣and␣resampled␣pjm␣data');

% slide energy to be a minimum at zero, start at same place every day
tregresamp10sintmat = tregresamp10sintmat - min(min(tregresamp10sintmat));

% duration curves and vectors
tregresamp10svec = reshape(tregresamp10smat, [],1);
tregresamp10srrvec = reshape(tregresamp10srrmat, [],1);
tregresamp10sintvec = reshape(tregresamp10sintmat, [],1);

tregresamp10sdur = sort(tregresamp10svec);
tregresamp10srrdur = sort(tregresamp10srrvec);
tregresamp10sintdur = sort(tregresamp10sintvec);

% mean absolute ramp rate
tmeanabsresamp10srr = mean(abs(tregresamp10srrvec));

%% vectors for the x axes

timehr10s = linspace(0, 24*7-1/360, numpts10s)';
durx10s = linspace(0, 1, numpts10s)';

timehr4s1wk = linspace(0, 24*numwkdays-4/3600, numpts1wk)';
timehr4s4wks = linspace(0,24*(numwkdays*numwks)-4/3600, numpts1wk*numwks)';

durx4s1wk = linspace(0, 1, numpts1wk)';
durx4s4wks = linspace(0, 1, numpts1wk*numwks)';
durx10s4wks = linspace(0, 1, numel(tregresamp10smat))';

figure
plot(durx10s4wks, tregresamp10srrdur, 'k-', durx4s4wks, tregpjmrrdur,
    'b--');
title('resampled␣and␣original␣ramp␣duration');

%% Pull out "bulk" load from pjm data, using same 90-min filter

window90pjm = 90*15+1; % number of samples to make up 90 min, want an odd
    number
filtvec90pjm = ones(window90pjm, 1)./window90pjm;

init = zeros(size(tregpjmmat));

tregavg90mat = init;
tregfuzz90mat = init;

for index = 1:4 % 4 weeks
```

```
    offset = tregpjmmat(1,index) - tregpjmmat(end,index);
    [waitforit, zi] = filter(filtvec90pjm,1, (tregpjmmat(:,index) +
        offset)); % calculate appropriate initial conditions: jigger so it
        ends where the next one starts
    working = filter(filtvec90pjm,1,tregpjmmat(:,index), zi); % use ICs
        from as if two identical days had been appended
    % BUT need to slide it so that the data points are centered in the
    % boxcars
    chunk = working(1:(window90pjm+1)/2)-offset;
    working = [working((window90pjm+3)/2:end); chunk];
    tregavg90mat(:,index) = working;
    tregfuzz90mat(:,index) = tregpjmmat(:,index) - working;

    tregavg90rrmat(:,index) = rr4sec(working);
    tregfuzz90rrmat(:,index) = rr4sec(tregfuzz90mat(:,index));

    tregfuzz90intmat(:,index) = int4sec(tregfuzz90mat(:,index));

end

tregavg90vec = reshape(tregavg90mat, [], 1);
tregfuzz90vec = reshape(tregfuzz90mat, [], 1);

tregavg90dur = sort(tregavg90vec);
tregfuzz90dur = sort(tregfuzz90vec);

tregfuzz90rrvec = reshape(tregfuzz90rrmat, [], 1);

tfuzz90meanabsrrorig = mean(abs(tregfuzz90rrvec));

%%%%% can use this later to prove that the averaging filter doesn't make a
    big difference

%% frequency spectrum

%% frequency content and PSD

% use welch's method (chop up and average PSD sets, then average across
% days)
for index = 1:7
    [PSDsff10swelch(:,index),w10swelch] = pwelch(sffuzz90mat(:,index));
    % just overwrite the frequency vector each time - should be the same
    % (same length)
end
PSDsff10smeanwelch = mean(PSDsff10swelch, 2);
f10srpmwelch = w10swelch/2/pi/10*60; % should be cycles/min
Pdistsff10smean = cumtrapz(w10swelch, PSDsff10smeanwelch);
Pdistsff10smeannorm = Pdistsff10smean/Pdistsff10smean(end);
```

```
for index = 1:4
    [PSDtregpjmwelch(:,index),w4swelch] = pwelch(tregpjmmat(:,index));
end
PSDtregpjmmeanwelch = mean(PSDtregpjmwelch, 2);
f4srpmwelch = w4swelch/2/pi/4*60; % should be cycles/min
Pdisttregpjmmean = cumtrapz(w4swelch, PSDtregpjmmeanwelch);
Pdisttregpjmmeannorm = Pdisttregpjmmean/Pdisttregpjmmean(end);

%% Autocorrelation

% 10 s data
ACF10smat = [];
Lags10smat = [];
Bounds10smat = [];

for index = 1:numdays10s

    [ACF,Lags,Bounds] =
        autocorr(sffuzz90mat(:,index),floor(numpts1day10s/2));
    %M and nSTDs are not specified -> default values
    ACF10smat = [ACF10smat ACF]; %append ACF and don't care if matlab cries
    Lags10smat = [Lags10smat Lags];
    Bounds10smat = [Bounds10smat Bounds];

end

Lags10smatmin = Lags10smat/6;
ACF10smean = mean(ACF10smat, 2); % should average over 2nd dim., ie rows

% 4 s pjm data
ACF4smat = [];
Lags4smat = [];
Bounds4smat = [];

for index = 1:numwks

    [ACF,Lags,Bounds] = autocorr(tregpjmmat(:,index),floor(numpts1wk/2));
    %M and nSTDs are not specified -> default values
    ACF4smat = [ACF4smat ACF]; %append ACF and don't care if matlab cries
    Lags4smat = [Lags4smat Lags];
    Bounds4smat = [Bounds4smat Bounds];

end

Lags4smatmin = Lags4smat/15;
ACF4smean = mean(ACF4smat, 2);
```

```
save('Autocorr_2sets.mat', 'ACF*', 'Lags*', 'Bounds*');

%% partial autocorrelation

% 10 s data
numparcorr = 30;

PACF10smat = [];
plags10smat = [];
pbounds10smat = [];

for index = 1:numdays10s

    [PACF,plags,pbounds] = parcorr(sffuzz90mat(:,index),numparcorr);
    %M and nSTDs are not specified -> default values
    PACF10smat = [PACF10smat PACF];
    plags10smat = [plags10smat plags];
    pbounds10smat = [pbounds10smat pbounds];

end

plags10smatmin = plags10smat/6;
PACF10smean = mean(PACF10smat, 2); % should average over 2nd dim., ie rows

% 4 s pjm data
PACF4smat = [];
plags4smat = [];
pbounds4smat = [];

for index = 1:numwks

    [PACF,plags,pbounds] = parcorr(tregpjmmat(:,index),numparcorr);
    %M and nSTDs are not specified -> default values
    PACF4smat = [PACF4smat PACF]; %append PACF and don't care if matlab
        cries
    plags4smat = [plags4smat plags];
    pbounds4smat = [pbounds4smat pbounds];

end

plags4smatmin = plags4smat/15;
PACF4smean = mean(PACF4smat, 2);

save('Parcorr_2sets.mat', 'PACF*', 'plags*', 'pbounds*');
```

C.3.2 Building the Filters

thesis_makefigs_ch_newpjmdata_subfile_build_all_filters.m

```
%% Olivia Leitermann  MIT LEES  10 jan 12
%% subfile to make thesis figures, new pjm data chapter
%% called by thesis_makefigs_ch_newpjmdata_<date>.m
%% build all filters required in this chapter

%% Chebyshev HP

n = 3; % filter order
R = 0.1; % passband ripple allowance, dB

% analog prototyping and digital conversion: build 4-sec cheb filter
fs4s = 1/4; % sampling frequency 1 cycle/4 sec
Wpa = 2*pi/60./[3 7 10 15 20 30 45 60]; % cutoff is 1/n min, different
    filts; Wpa is in rad/sec
numfilts = 8;
numfiltpts = 512*64;

init2 = zeros(numfiltpts, numfilts);
hatfs4s = init2;
was4s = init2;
hadtfs4s = init2;
wads4s = init2;

ta4s = [0:.25:60*60*4];
tad4s = [0:4:60*60*4];

for index2 = 1:numfilts
    %[ba4s, aa4s] = cheby1(n,R,Wpa(index2),'high', 's'); % tf form

    % continuous time
    [Aa4s, Ba4s, Ca4s, Da4s] = cheby1(n,R,Wpa(index2),'high', 's'); % state
        space
    sysana = ss(Aa4s, Ba4s, Ca4s, Da4s); % cont ss model
    [ba4s, aa4s] = tfdata(sysana, 'v'); % convert to tf
    [za4s, pa4s, ka4s] = zpkdata(sysana, 'v');
    [hatf4s,wa4s] = freqs(ba4s,aa4s, numfiltpts); % frequency response
    ya4s = step(sysana, ta4s);

    % convert to discrete time
    [Aad4s, Bad4s, Cad4s, Dad4s] = bilinear(Aa4s, Ba4s, Ca4s, Da4s,fs4s); %
        state space
    sysdig = ss(Aad4s, Bad4s, Cad4s, Dad4s, 1/fs4s); % discrete ss model
    [bad4s, aad4s] = tfdata(sysdig, 'v'); % convert to tf
    [zad4s, pad4s, kad4s] = zpkdata(sysdig, 'v');
    [hadtf4s, fad4s] = freqz(bad4s, aad4s, numfiltpts, fs4s);
```

```
    yad4s = step(sysdig, tad4s);

    % store variables
    hatfs4s(:,index2) = hatf4s;
    was4s(:,index2) = wa4s;
    bads4s(:,index2) = bad4s;
    aads4s(:,index2) = aad4s;
    hadtfs4s(:,index2) = hadtf4s;
    fads4s(:,index2) = fad4s;
    yas4s(:, index2) = ya4s;
    yads4s(:, index2) = yad4s;
    zas4s{index2} = complexize(za4s); % different #s of elts: need to be
        cell arrays?
    pas4s{index2} = complexize(pa4s); % make sure it thinks of the p/z in
        complex plane
    kas4s(:,index2) = ka4s;
    zads4s{index2} = complexize(zad4s);
    pads4s{index2} = complexize(pad4s);
    kads4s(:,index2) = kad4s;
%     tas4s(:,index2) = ta4s;
%     tads4s(:,index2) = tad4s;
%
end

% build 10-sec cheb filter
% analog prototyping and digital conversion
fs10s = 1/10; % sampling frequency 1 cycle/10 sec
Wpa = 2*pi/60./[3 7 10 15 20 30 45 60]; % cutoff is 1/n min, different filts

init2 = zeros(numfiltpts, numfilts);
hatfs10s = init2;
was10s = init2;
hadtfs10s = init2;
wads10s = init2;

ta10s = [0:1:60*60*6]; % 6 h time vec
tad10s = [0:10:60*60*6];

for index2 = 1:numfilts
    %[ba, aa] = cheby1(n,R,Wpa(index2),'high', 's'); %zeros and poles
    % design cont time as state space
    [Aa10s, Ba10s, Ca10s, Da10s] = cheby1(n, R, Wpa(index2), 'high', 's');
    sysana = ss(Aa10s, Ba10s, Ca10s, Da10s); %create ss model
    [ba10s, aa10s] = tfdata(sysana, 'v'); % convert to tf
    [hatf10s,wa10s] = freqs(ba10s,aa10s, numfiltpts); % frequency response
    ya10s = step(sysana, ta10s); % generate step resp

    % convert to discrete time
```

```
    [Aad10s, Bad10s, Cad10s, Dad10s] = bilinear(Aa10s, Ba10s, Ca10s, Da10s,
        fs10s);
    sysdig = ss(Aad10s, Bad10s, Cad10s, Dad10s, 1/fs10s); % discrete ss
        model
    [bad10s, aad10s] = tfdata(sysdig, 'v'); % convert to tf
    [hadtf10s, fad10s] = freqz(bad10s, aad10s, numfiltpts, fs10s); % freq
        resp
    yad10s = step(sysdig, tad10s); % step resp

    % store vars
    hatfs10s(:,index2) = hatf10s;
    was10s(:,index2) = wa10s;
    bads10s(:,index2) = bad10s;
    aads10s(:,index2) = aad10s;
    hadtfs10s(:,index2) = hadtf10s;
    fads10s(:,index2) = fad10s;
    yas10s(:,index2) = ya10s;
    yads10s(:,index2) = yad10s;

end

fas10s = was10s/2/pi;

%% Chebyshev LP

% use same n and R values as above

init2 = zeros(numfiltpts, numfilts);
hatfs4scheblp = init2;
was4scheblp = init2;
hadtfs4scheblp = init2;
wads4scheblp = init2;

ta4scheblp = [0:.25:60*60*4];
tad4scheblp = [0:4:60*60*4];

for index2 = 1:numfilts
    %[ba4s, aa4s] = cheby1(n,R,Wpa(index2),'high', 's'); % tf form

    % continuous time
    [Aa4scheblp, Ba4scheblp, Ca4scheblp, Da4scheblp] =
        cheby1(n,R,Wpa(index2),'low', 's'); % state space
    sysana = ss(Aa4scheblp, Ba4scheblp, Ca4scheblp, Da4scheblp); % cont ss
        model
    [ba4scheblp, aa4scheblp] = tfdata(sysana, 'v'); % convert to tf
    [za4scheblp, pa4scheblp, ka4scheblp] = zpkdata(sysana, 'v');
    [hatf4scheblp,wa4scheblp] = freqs(ba4scheblp,aa4scheblp, numfiltpts); %
        frequency response
    ya4scheblp = step(sysana, ta4scheblp);
```

```
    % convert to discrete time
    [Aad4scheblp, Bad4scheblp, Cad4scheblp, Dad4scheblp] =
        bilinear(Aa4scheblp, Ba4scheblp, Ca4scheblp, Da4scheblp,fs4s); %
        state space
    sysdig = ss(Aad4scheblp, Bad4scheblp, Cad4scheblp, Dad4scheblp,
        1/fs4s); % discrete ss model
    [bad4scheblp, aad4scheblp] = tfdata(sysdig, 'v'); % convert to tf
    [zad4scheblp, pad4scheblp, kad4scheblp] = zpkdata(sysdig, 'v');
    [hadtf4scheblp, fad4scheblp] = freqz(bad4scheblp, aad4scheblp,
        numfiltpts, fs4s);
    yad4scheblp = step(sysdig, tad4scheblp);

    % store variables
    hatfs4scheblp(:,index2) = hatf4scheblp;
    was4scheblp(:,index2) = wa4scheblp;
    bads4scheblp(:,index2) = bad4scheblp;
    aads4scheblp(:,index2) = aad4scheblp;
    hadtfs4scheblp(:,index2) = hadtf4scheblp;
    fads4scheblp(:,index2) = fad4scheblp;
    yas4scheblp(:, index2) = ya4scheblp;
    yads4scheblp(:, index2) = yad4scheblp;
    zas4scheblp{index2} = complexize(za4scheblp); % different #s of elts:
        need to be cell arrays?
    pas4scheblp{index2} = complexize(pa4scheblp); % make sure it thinks of
        the p/z in complex plane
    kas4scheblp(:,index2) = ka4scheblp;
    zads4scheblp{index2} = complexize(zad4scheblp);
    pads4scheblp{index2} = complexize(pad4scheblp);
    kads4scheblp(:,index2) = kad4scheblp;
%     tas4s(:,index2) = ta4s;
%     tads4s(:,index2) = tad4s;
%
end

% build 10-sec cheb filter
% analog prototyping and digital conversion
init2 = zeros(numfiltpts, numfilts);
hatfs10scheblp = init2;
was10scheblp = init2;
hadtfs10scheblp = init2;
wads10scheblp = init2;

ta10scheblp = [0:1:60*60*6]; % 6 h time vec
tad10scheblp = [0:10:60*60*6];

for index2 = 1:numfilts
    %[ba, aa] = cheby1(n,R,Wpa(index2),'high', 's'); %zeros and poles
    % design cont time as state space
```

```
    [Aa10scheblp, Ba10scheblp, Ca10scheblp, Da10scheblp] = cheby1(n, R,
        Wpa(index2), 'low', 's');
    sysana = ss(Aa10scheblp, Ba10scheblp, Ca10scheblp, Da10scheblp);
        %create ss model
    [ba10scheblp, aa10scheblp] = tfdata(sysana, 'v'); % convert to tf
    [hatf10scheblp,wa10scheblp] = freqs(ba10scheblp,aa10scheblp,
        numfiltpts); % frequency response
    ya10scheblp = step(sysana, ta10scheblp); % generate step resp

    % convert to discrete time
    [Aad10scheblp, Bad10scheblp, Cad10scheblp, Dad10scheblp] =
        bilinear(Aa10scheblp, Ba10scheblp, Ca10scheblp, Da10scheblp, fs10s);
    sysdig = ss(Aad10scheblp, Bad10scheblp, Cad10scheblp, Dad10scheblp,
        1/fs10s); % discrete ss model
    [bad10scheblp, aad10scheblp] = tfdata(sysdig, 'v'); % convert to tf
    [hadtf10scheblp, fad10scheblp] = freqz(bad10scheblp, aad10scheblp,
        numfiltpts, fs10s); % freq resp
    yad10scheblp = step(sysdig, tad10scheblp); % step resp

    % store vars
    hatfs10scheblp(:,index2) = hatf10scheblp;
    was10scheblp(:,index2) = wa10scheblp;
    bads10scheblp(:,index2) = bad10scheblp;
    aads10scheblp(:,index2) = aad10scheblp;
    hadtfs10scheblp(:,index2) = hadtf10scheblp;
    fads10scheblp(:,index2) = fad10scheblp;
    yas10scheblp(:,index2) = ya10scheblp;
    yads10scheblp(:,index2) = yad10scheblp;

end

fas10scheblp = was10scheblp/2/pi;

%% Butterworth HP (same order)

% analog prototyping and digital conversion: build 4-sec butterworth filter

% init2 = zeros(numfiltpts, numfilts);
% hatfs4s = init2;
% was4s = init2;
% hadtfs4s = init2;
% wads4s = init2;
%
% % build 10-sec butterworth filter
% % analog prototyping and digital conversion
%
% init2 = zeros(numfiltpts, numfilts);
% hatfs10s = init2;
% was10s = init2;
```

```
% hadtfs10s = init2;
% wads10s = init2;

for index2 = 1:numfilts
    %[ba4s, aa4s] = cheby1(n,R,Wpa(index2),'high', 's'); % tf form

    % continuous time
    [Aa4sbutter, Ba4sbutter, Ca4sbutter, Da4sbutter] =
        butter(n,Wpa(index2),'high', 's'); % state space
    sysana = ss(Aa4sbutter, Ba4sbutter, Ca4sbutter, Da4sbutter); % cont ss
        model
    [ba4sbutter, aa4sbutter] = tfdata(sysana, 'v'); % convert to tf
    [za4sbutter, pa4sbutter, ka4sbutter] = zpkdata(sysana, 'v');
    [hatf4sbutter,wa4sbutter] = freqs(ba4sbutter,aa4sbutter, numfiltpts); %
        frequency response
    ya4sbutter = step(sysana, ta4s);

    % convert to discrete time 4 sec
    [Aad4sbutter, Bad4sbutter, Cad4sbutter, Dad4sbutter] =
        bilinear(Aa4sbutter, Ba4sbutter, Ca4sbutter, Da4sbutter,fs4s); %
        state space
    sysdig = ss(Aad4sbutter, Bad4sbutter, Cad4sbutter, Dad4sbutter,
        1/fs4s); % discrete ss model
    [bad4sbutter, aad4sbutter] = tfdata(sysdig, 'v'); % convert to tf
    [zad4sbutter, pad4sbutter, kad4sbutter] = zpkdata(sysdig, 'v');
    [hadtf4sbutter, fad4sbutter] = freqz(bad4sbutter, aad4sbutter,
        numfiltpts, fs4s);
    yad4sbutter = step(sysdig, tad4s);

    % store variables
    hatfs4sbutter(:,index2) = hatf4sbutter;
    was4sbutter(:,index2) = wa4sbutter;
    bads4sbutter(:,index2) = bad4sbutter;
    aads4sbutter(:,index2) = aad4sbutter;
    hadtfs4sbutter(:,index2) = hadtf4sbutter;
    fads4sbutter(:,index2) = fad4sbutter;
    yas4sbutter(:, index2) = ya4sbutter;
    yads4sbutter(:, index2) = yad4sbutter;
    zas4sbutter{index2} = complexize(za4sbutter); % different #s of elts:
        need to be cell arrays?
    pas4sbutter{index2} = complexize(pa4sbutter); % make sure it thinks of
        the p/z in complex plane
    kas4sbutter(:,index2) = ka4sbutter;
    zads4sbutter{index2} = complexize(zad4sbutter);
    pads4sbutter{index2} = complexize(pad4sbutter);
    kads4sbutter(:,index2) = kad4sbutter;
%     tas4s(:,index2) = ta4s;
%     tads4s(:,index2) = tad4s;
%
```

```
    % convert to discrete time 10 sec
    [Aad10sbutter, Bad10sbutter, Cad10sbutter, Dad10sbutter] =
        bilinear(Aa4sbutter, Ba4sbutter, Ca4sbutter, Da4sbutter, fs10s);
    sysdig = ss(Aad10sbutter, Bad10sbutter, Cad10sbutter, Dad10sbutter,
        1/fs10s); % discrete ss model
    [bad10sbutter, aad10sbutter] = tfdata(sysdig, 'v'); % convert to tf
    [hadtf10sbutter, fad10sbutter] = freqz(bad10sbutter, aad10sbutter,
        numfiltpts, fs10s); % freq resp
    yad10sbutter = step(sysdig, tad10s); % step resp

    % store vars
    bads10sbutter(:,index2) = bad10sbutter;
    aads10sbutter(:,index2) = aad10sbutter;
    hadtfs10sbutter(:,index2) = hadtf10sbutter;
    fads10sbutter(:,index2) = fad10sbutter;
    yads10sbutter(:,index2) = yad10sbutter;


end


%% Butterworth LP

% use same n value as above

init2 = zeros(numfiltpts, numfilts);
hatfs4sbutterlp = init2;
was4sbutterlp = init2;
hadtfs4sbutterlp = init2;
wads4sbutterlp = init2;

ta4sbutterlp = [0:.25:60*60*4];
tad4sbutterlp = [0:4:60*60*4];

hatfs10sbutterlp = init2;
was10sbutterlp = init2;
hadtfs10sbutterlp = init2;
wads10sbutterlp = init2;

ta10sbutterlp = [0:1:60*60*6]; % 6 h time vec
tad10sbutterlp = [0:10:60*60*6];

for index2 = 1:numfilts
    %[ba4s, aa4s] = cheby1(n,R,Wpa(index2),'high', 's'); % tf form

    % continuous time
    [Aa4sbutterlp, Ba4sbutterlp, Ca4sbutterlp, Da4sbutterlp] =
        butter(n,Wpa(index2),'low', 's'); % state space
```

```
    sysana = ss(Aa4sbutterlp, Ba4sbutterlp, Ca4sbutterlp, Da4sbutterlp); %
        cont ss model
    [ba4sbutterlp, aa4sbutterlp] = tfdata(sysana, 'v'); % convert to tf
    [za4sbutterlp, pa4sbutterlp, ka4sbutterlp] = zpkdata(sysana, 'v');
    [hatf4sbutterlp,wa4sbutterlp] = freqs(ba4sbutterlp,aa4sbutterlp,
        numfiltpts); % frequency response
    ya4sbutterlp = step(sysana, ta4sbutterlp);

    % convert to discrete time 4 sec
    [Aad4sbutterlp, Bad4sbutterlp, Cad4sbutterlp, Dad4sbutterlp] =
        bilinear(Aa4sbutterlp, Ba4sbutterlp, Ca4sbutterlp,
        Da4sbutterlp,fs4s); % state space
    sysdig = ss(Aad4sbutterlp, Bad4sbutterlp, Cad4sbutterlp, Dad4sbutterlp,
        1/fs4s); % discrete ss model
    [bad4sbutterlp, aad4sbutterlp] = tfdata(sysdig, 'v'); % convert to tf
    [zad4sbutterlp, pad4sbutterlp, kad4sbutterlp] = zpkdata(sysdig, 'v');
    [hadtf4sbutterlp, fad4sbutterlp] = freqz(bad4sbutterlp, aad4sbutterlp,
        numfiltpts, fs4s);
    yad4sbutterlp = step(sysdig, tad4sbutterlp);

    % store variables
    hatfs4sbutterlp(:,index2) = hatf4sbutterlp;
    was4sbutterlp(:,index2) = wa4sbutterlp;
    bads4sbutterlp(:,index2) = bad4sbutterlp;
    aads4sbutterlp(:,index2) = aad4sbutterlp;
    hadtfs4sbutterlp(:,index2) = hadtf4sbutterlp;
    fads4sbutterlp(:,index2) = fad4sbutterlp;
    yas4sbutterlp(:, index2) = ya4sbutterlp;
    yads4sbutterlp(:, index2) = yad4sbutterlp;
    zas4sbutterlp{index2} = complexize(za4sbutterlp); % different #s of
        elts: need to be cell arrays?
    pas4sbutterlp{index2} = complexize(pa4sbutterlp); % make sure it thinks
        of the p/z in complex plane
    kas4sbutterlp(:,index2) = ka4sbutterlp;
    zads4sbutterlp{index2} = complexize(zad4sbutterlp);
    pads4sbutterlp{index2} = complexize(pad4sbutterlp);
    kads4sbutterlp(:,index2) = kad4sbutterlp;
%     tas4s(:,index2) = ta4s;
%     tads4s(:,index2) = tad4s;
%
% convert to discrete time 10 sec
    [Aad10sbutterlp, Bad10sbutterlp, Cad10sbutterlp, Dad10sbutterlp] =
        bilinear(Aa4sbutterlp, Ba4sbutterlp, Ca4sbutterlp, Da4sbutterlp,
        fs10s);
    sysdig = ss(Aad10sbutterlp, Bad10sbutterlp, Cad10sbutterlp,
        Dad10sbutterlp, 1/fs10s); % discrete ss model
    [bad10sbutterlp, aad10sbutterlp] = tfdata(sysdig, 'v'); % convert to tf
    [hadtf10sbutterlp, fad10sbutterlp] = freqz(bad10sbutterlp,
        aad10sbutterlp, numfiltpts, fs10s); % freq resp
```

```
    yad10sbutterlp = step(sysdig, tad10sbutterlp); % step resp

    % store vars
    bads10sbutterlp(:,index2) = bad10sbutterlp;
    aads10sbutterlp(:,index2) = aad10sbutterlp;
    hadtfs10sbutterlp(:,index2) = hadtf10sbutterlp;
    fads10sbutterlp(:,index2) = fad10sbutterlp;
    yads10sbutterlp(:,index2) = yad10sbutterlp;

end

fas10sbutterlp = was10sbutterlp/2/pi;
fas4sbutterlp = was4sbutterlp/2/pi;


%% Bessel LP

% analog prototyping and digital conversion: build 4-sec cheb filter
Wpa = 2*pi/60./[3 7 10 15 20 30 45 60]; % cutoff is 1/n min, different
    filts; Wpa is in rad/sec
numfilts = 8;
numfiltpts = 512*64;

init2 = zeros(numfiltpts, numfilts);
hatfs4sbess = init2;
was4sbess = init2;
hadtfs4sbess = init2;
wads4sbess = init2;

ta4sbess = [0:.25:60*60*4];
tad4sbess = [0:4:60*60*4];

for index2 = 1:numfilts
    %[ba4s, aa4s] = cheby1(n,R,Wpa(index2),'high', 's'); % tf form

    % continuous time
    [Aa4sbess, Ba4sbess, Ca4sbess, Da4sbess] = besself(n,Wpa(index2)); %
        state space
    sysana = ss(Aa4sbess, Ba4sbess, Ca4sbess, Da4sbess); % cont ss model
    [ba4sbess, aa4sbess] = tfdata(sysana, 'v'); % convert to tf
    [za4sbess, pa4sbess, ka4sbess] = zpkdata(sysana, 'v');
    [hatf4sbess,wa4sbess] = freqs(ba4sbess,aa4sbess, numfiltpts); %
        frequency response
    ya4sbess = step(sysana, ta4sbess);

    % convert to discrete time
    [bad4sbess, aad4sbess] = impinvar(ba4sbess, aa4sbess, fs4s);
    sysdig = tf(bad4sbess, aad4sbess, 1/fs4s); % discrete ss model
```

```
    [Aad4sbess, Bad4sbess, Cad4sbess, Dad4sbess] = ssdata(sysdig); % state
        space
    [zad4sbess, pad4sbess, kad4sbess] = zpkdata(sysdig, 'v');
    [hadtf4sbess, fad4sbess] = freqz(bad4sbess, aad4sbess, numfiltpts,
        fs4s);
    yad4sbess = step(sysdig, tad4sbess);

    % store variables
    hatfs4sbess(:,index2) = hatf4sbess;
    was4sbess(:,index2) = wa4sbess;
    bads4sbess(:,index2) = bad4sbess;
    aads4sbess(:,index2) = aad4sbess;
    hadtfs4sbess(:,index2) = hadtf4sbess;
    fads4sbess(:,index2) = fad4sbess;
    yas4sbess(:, index2) = ya4sbess;
    yads4sbess(:, index2) = yad4sbess;
    zas4sbess{index2} = complexize(za4sbess); % different #s of elts: need
        to be cell arrays?
    pas4sbess{index2} = complexize(pa4sbess); % make sure it thinks of the
        p/z in complex plane
    kas4sbess(:,index2) = ka4sbess;
    zads4sbess{index2} = complexize(zad4sbess);
    pads4sbess{index2} = complexize(pad4sbess);
    kads4sbess(:,index2) = kad4sbess;
%     tas4s(:,index2) = ta4s;
%     tads4s(:,index2) = tad4s;
%
end

% build 10-sec bessel filter
% analog prototyping and digital conversion
Wpa = 2*pi/60./[3 7 10 15 20 30 45 60]; % cutoff is 1/n min, different filts
numfilts = 8;
numfiltpts = 512*16;

init2 = zeros(numfiltpts, numfilts);
hatfs10sbess = init2;
was10sbess = init2;
hadtfs10sbess = init2;
wads10sbess = init2;

ta10sbess = [0:1:60*60*6]; % 6 h time vec
tad10sbess = [0:10:60*60*6];

for index2 = 1:numfilts
    %[ba, aa] = cheby1(n,R,Wpa(index2),'high', 's'); %zeros and poles
    % design cont time as state space
   % continuous time
```

```
    [Aa10sbess, Ba10sbess, Ca10sbess, Da10sbess] = besself(n,Wpa(index2));
        % state space
    sysana = ss(Aa10sbess, Ba10sbess, Ca10sbess, Da10sbess); % cont ss model
    [ba10sbess, aa10sbess] = tfdata(sysana, 'v'); % convert to tf
    [za10sbess, pa10sbess, ka10sbess] = zpkdata(sysana, 'v');
    [hatf10sbess,wa10sbess] = freqs(ba10sbess,aa10sbess, numfiltpts); %
        frequency response
    ya10sbess = step(sysana, ta10sbess);

    % convert to discrete time
    [bad10sbess, aad10sbess] = impinvar(ba10sbess, aa10sbess, fs10s);
    sysdig = tf(bad10sbess, aad10sbess, 1/fs10s); % discrete ss model
    [Aad10sbess, Bad10sbess, Cad10sbess, Dad10sbess] = ssdata(sysdig); %
        state space
    [zad10sbess, pad10sbess, kad10sbess] = zpkdata(sysdig, 'v');
    [hadtf10sbess, fad10sbess] = freqz(bad10sbess, aad10sbess, numfiltpts,
        fs10s);
    yad10sbess = step(sysdig, tad10sbess); % step resp

    % store vars
    hatfs10sbess(:,index2) = hatf10sbess;
    was10sbess(:,index2) = wa10sbess;
    bads10sbess(:,index2) = bad10sbess;
    aads10sbess(:,index2) = aad10sbess;
    hadtfs10sbess(:,index2) = hadtf10sbess;
    fads10sbess(:,index2) = fad10sbess;
    yas10sbess(:,index2) = ya10sbess;
    yads10sbess(:,index2) = yad10sbess;

end

fas10sbess = was10sbess/2/pi;


%% FIR

fcutoffs = [3, 1, 1/3, 1/7, 1/10, 1/15, 1/20, 1/30]/60;
numFIRfilts = 8;

fcnorm10s = fcutoffs/fs10s;
fcnorm4s = fcutoffs/fs4s;

% long delay filters

longfiltorder10s = 60*60/10; % 60 min filter, 30 min delay
longfiltorder4s = 60*60/4;

for index = 1:numFIRfilts % filters are in each ROW
    longfir1filt10s(index,:) = fir1(longfiltorder10s+1, fcnorm10s(index));
```

```
    longfir1filt4s(index,:) = fir1(longfiltorder4s+1, fcnorm4s(index));
    % frequency response
    [hfirlong10s(:,index),ffirlong10s(:,index)]=freqz(longfir1filt10s(index,:),1,numfiltpt
    [hfirlong4s(:,index),ffirlong4s(:,index)]=freqz(longfir1filt4s(index,:),1,numfiltpts,1
end

longfiltdelaysamps10s = longfiltorder10s/2;
longfiltdelaysamps4s = longfiltorder4s/2;

% short delay filters - make a selection of them
numorder = 1;
%shortfiltorders10s = [3 6 9];
%shortfiltorders4s = [3 6 15];
shortfiltorders10s = 6; % filts are approx. 1 minute, that means approx. 30
    sec delay
shortfiltorders4s = 14;

% highest order should be at the end

% need to initialize these so there's zeros appropriately
shortfir1filts10s = zeros(shortfiltorders10s(end)+1,numfilts, numorder);
shortfir1filts4s = zeros(shortfiltorders4s(end)+1,numfilts, numorder);

for ordidx = 1:numorder
    for filtidx = 1:numFIRfilts % filters come out in rows, put them in cols
        shortfir1filts10s(1:shortfiltorders10s(ordidx)+1,filtidx, ordidx) =
            fir1(shortfiltorders10s(ordidx), fcnorm10s(filtidx))';
        shortfir1filts4s(1:shortfiltorders4s(ordidx)+1,filtidx, ordidx) =
            fir1(shortfiltorders4s(ordidx), fcnorm4s(filtidx))';
    % frequency response
    [hfirshort10s(:,filtidx,ordidx),ffirshort10s(:,filtidx,ordidx)]=freqz(shortfir1filts10
    [hfirshort4s(:,filtidx,
        ordidx),ffirshort4s(:,filtidx,ordidx)]=freqz(shortfir1filts4s(:,filtidx,ordidx),1,
    end
end

shortfiltdelaysamps10s = shortfiltorders10s/2;
shortfiltdelaysamps4s = shortfiltorders4s/2;
```

C.3.3 Performing Open Loop Filtering

thesis_makefigs_ch_newpjmdata_subfile_do_all_filtering.m

```
%% Olivia Leitermann  MIT LEES  10 jan 12
%% subfile to make thesis figures, new pjm data chapter
%% called by thesis_makefigs_ch_newpjmdata_<date>.m
%% perform all the (separation) filtering required for this chapter
```

```
%% Cheb HP filtering

for filtindex = 1:numfilts;
    for dayindex = 1:numdays10s
        offset = sffuzz90mat(1,dayindex) - sffuzz90mat(end,dayindex);
        % also important to use the filter indicated by index2
        [waitforit, zi] = filter(bads10s(:,filtindex),
            aads10s(:,filtindex), (sffuzz90mat(:,dayindex) + offset)); %
            calculate appropriate initial conditions: jigger so it ends
            where the next one starts
        hifreqtemp = filter(bads10s(:,filtindex), aads10s(:,filtindex),
            sffuzz90mat(:,dayindex), zi); % use ICs from as if two
            identical days had been appended
        lofreqtemp = sffuzz90mat(:,dayindex) - hifreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int10sec(hifreqtemp);
        lofreqrrtemp = rr10sec(lofreqtemp);
        hifreqrrtemp = rr10sec(hifreqtemp);


        %store all vars in a vector
        sfflofreqmat3(:,dayindex, filtindex) = lofreqtemp;
        sffhifreqmat3(:,dayindex, filtindex) = hifreqtemp;
        sfflofreqrrmat3(:,dayindex, filtindex) = lofreqrrtemp;
        sffhifreqrrmat3(:,dayindex, filtindex) = hifreqrrtemp;
        sffhifreqintmat3(:,dayindex, filtindex) = hifreqinttemp;

    end
end

sfflofreqmat2 = reshape(sfflofreqmat3, [], numfilts);
sffhifreqmat2 = reshape(sffhifreqmat3, [], numfilts);
sfflofreqrrmat2 = reshape(sfflofreqrrmat3, [], numfilts);
sffhifreqrrmat2 = reshape(sffhifreqrrmat3, [], numfilts);
sffhifreqintmat2 = reshape(sffhifreqintmat3, [], numfilts);

sffhifreqintmat2 = sffhifreqintmat2 - repmat(min(sffhifreqintmat2),
    numpts10s, 1);

sfflofreqdur = sort(sfflofreqmat2);
sffhifreqdur = sort(sffhifreqmat2);
sfflofreqrrdur = sort(sfflofreqrrmat2);
sffhifreqrrdur = sort(sffhifreqrrmat2);
sffhifreqintdur = sort(sffhifreqintmat2);

% dPdt vs Wmax
sffmeanabslfrr = mean(abs(sfflofreqrrdur));
```

```
sffrmslfrr = sqrt(mean(sfflofreqrrdur.^2));
sffmaxhfint = sffhifreqintdur(end,:);


bigdurx10s = linspace(0, 1, numpts10s)';
bigtime10s = linspace(0, 24*numdays10s-1/360, numpts10s)'; % time in hours

% data structure: columns are time-series data for a particular week and
% filter, along dim 2 (rows) is the same filter for different weeks, and
% dim 3 is the same week, different filters

for filtindex = 1:numfilts;
    for wkindex = 1:numwks
        offset = tregpjmmat(1,wkindex) - tregpjmmat(end,wkindex);

        [waitforit, zi] = filter(bads4s(:,filtindex), aads4s(:,filtindex),
            (tregpjmmat(:,wkindex) + offset)); % calculate appropriate
            initial conditions: jigger so it ends where the next one starts
        hifreqtemp = filter(bads4s(:,filtindex), aads4s(:,filtindex),
            tregpjmmat(:,wkindex), zi); % use ICs from as if two identical
            days had been appended
        lofreqtemp = tregpjmmat(:,wkindex) - hifreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int4sec(hifreqtemp);
        lofreqrrtemp = rr4sec(lofreqtemp);
        hifreqrrtemp = rr4sec(hifreqtemp);

        % save data to the matrices
        thifreqpjmmat3(:, wkindex, filtindex) = hifreqtemp;
        tlofreqpjmmat3(:, wkindex, filtindex) = lofreqtemp;
        thifreqpjmintmat3(:, wkindex, filtindex) = hifreqinttemp;
        thifreqpjmrrmat3(:, wkindex, filtindex) = hifreqrrtemp;
        tlofreqpjmrrmat3(:, wkindex, filtindex) = lofreqrrtemp;
    end
end

% reshape into column vectors, one for each filter

thifreqpjmmat2 = reshape(thifreqpjmmat3, [], numfilts, 1);
tlofreqpjmmat2 = reshape(tlofreqpjmmat3, [], numfilts, 1);
thifreqpjmintmat2 = reshape(thifreqpjmintmat3, [], numfilts, 1);
thifreqpjmrrmat2 = reshape(thifreqpjmrrmat3, [], numfilts, 1);
tlofreqpjmrrmat2 = reshape(tlofreqpjmrrmat3, [], numfilts, 1);

% shift integral to have minimum at zero: subtract min for each column
thifreqpjmintmat2 = thifreqpjmintmat2 - repmat(min(thifreqpjmintmat2),
    numpts1wk*numwks, 1);
```

```
% durations (sort each column)
thifreqpjmdur = sort(thifreqpjmmat2);
tlofreqpjmdur = sort(tlofreqpjmmat2);
thifreqpjmintdur = sort(thifreqpjmintmat2);
thifreqpjmrrdur = sort(thifreqpjmrrmat2);
tlofreqpjmrrdur = sort(tlofreqpjmrrmat2);

% dPdt vs Wmax
tmeanabslfrr = mean(abs(tlofreqpjmrrdur));
trmslfrr = sqrt(mean(tlofreqpjmrrdur.^2));
tmaxhfint = thifreqpjmintdur(end,:);

% data with the 90-minute average taken out

for filtindex = 1:numfilts;
    for wkindex = 1:numwks
        offset = tregfuzz90mat(1,wkindex) - tregfuzz90mat(end,wkindex);

        [waitforit, zi] = filter(bads4s(:,filtindex), aads4s(:,filtindex),
            (tregfuzz90mat(:,wkindex) + offset));
        % calculate appropriate initial conditions: jigger so it ends where
            the next one starts
        hifreqtemp = filter(bads4s(:,filtindex), aads4s(:,filtindex),
            tregfuzz90mat(:,wkindex), zi);
        % use ICs from as if two identical days had been appended
        lofreqtemp = tregfuzz90mat(:,wkindex) - hifreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int4sec(hifreqtemp);
        lofreqrrtemp = rr4sec(lofreqtemp);
        hifreqrrtemp = rr4sec(hifreqtemp);

        % save data to the matrices
        tfuzz90hifreqpjmmat3(:, wkindex, filtindex) = hifreqtemp;
        tfuzz90lofreqpjmmat3(:, wkindex, filtindex) = lofreqtemp;
        tfuzz90hifreqpjmintmat3(:, wkindex, filtindex) = hifreqinttemp;
        tfuzz90hifreqpjmrrmat3(:, wkindex, filtindex) = hifreqrrtemp;
        tfuzz90lofreqpjmrrmat3(:, wkindex, filtindex) = lofreqrrtemp;
    end
end

% reshape into column vectors, one for each filter

tfuzz90hifreqpjmmat2 = reshape(tfuzz90hifreqpjmmat3, [], numfilts, 1);
tfuzz90lofreqpjmmat2 = reshape(tfuzz90lofreqpjmmat3, [], numfilts, 1);
tfuzz90hifreqpjmintmat2 = reshape(tfuzz90hifreqpjmintmat3, [], numfilts, 1);
tfuzz90hifreqpjmrrmat2 = reshape(tfuzz90hifreqpjmrrmat3, [], numfilts, 1);
tfuzz90lofreqpjmrrmat2 = reshape(tfuzz90lofreqpjmrrmat3, [], numfilts, 1);
```

```
% shift integral to have minimum at zero: subtract min for each column
tfuzz90hifreqpjmintmat2 = tfuzz90hifreqpjmintmat2 -
    repmat(min(tfuzz90hifreqpjmintmat2), numpts1wk*numwks, 1);

% durations (sort each column)
tfuzz90hifreqpjmdur = sort(tfuzz90hifreqpjmmat2);
tfuzz90lofreqpjmdur = sort(tfuzz90lofreqpjmmat2);
tfuzz90hifreqpjmintdur = sort(tfuzz90hifreqpjmintmat2);
tfuzz90hifreqpjmrrdur = sort(tfuzz90hifreqpjmrrmat2);
tfuzz90lofreqpjmrrdur = sort(tfuzz90lofreqpjmrrmat2);

% dPdt vs Wmax
tfuzz90meanabslfrr = mean(abs(tfuzz90lofreqpjmrrdur));
tfuzz90rmslfrr = sqrt(mean(tfuzz90lofreqpjmrrdur.^2));
tfuzz90maxhfint = tfuzz90hifreqpjmintdur(end,:);


% data resampled to 10s

for filtindex = 1:numfilts;
    for wkindex = 1:numwks
        offset = tregresamp10smat(1,wkindex) -
            tregresamp10smat(end,wkindex);

        [waitforit, zi] = filter(bads10s(:,filtindex),
            aads10s(:,filtindex), (tregresamp10smat(:,wkindex) + offset));
            % calculate appropriate initial conditions: jigger so it ends
            where the next one starts
        hifreqtemp = filter(bads10s(:,filtindex), aads10s(:,filtindex),
            tregresamp10smat(:,wkindex), zi); % use ICs from as if two
            identical days had been appended
        lofreqtemp = tregresamp10smat(:,wkindex) - hifreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int10sec(hifreqtemp);
        lofreqrrtemp = rr10sec(lofreqtemp);
        hifreqrrtemp = rr10sec(hifreqtemp);

        % save data to the matrices
        thifreqresamp10smat3(:, wkindex, filtindex) = hifreqtemp;
        tlofreqresamp10smat3(:, wkindex, filtindex) = lofreqtemp;
        thifreqresamp10sintmat3(:, wkindex, filtindex) = hifreqinttemp;
        thifreqresamp10srrmat3(:, wkindex, filtindex) = hifreqrrtemp;
        tlofreqresamp10srrmat3(:, wkindex, filtindex) = lofreqrrtemp;
    end
end

% reshape into column vectors, one for each filter
```

```
thifreqresamp10smat2 = reshape(thifreqresamp10smat3, [], numfilts, 1);
tlofreqresamp10smat2 = reshape(tlofreqresamp10smat3, [], numfilts, 1);
thifreqresamp10sintmat2 = reshape(thifreqresamp10sintmat3, [], numfilts, 1);
thifreqresamp10srrmat2 = reshape(thifreqresamp10srrmat3, [], numfilts, 1);
tlofreqresamp10srrmat2 = reshape(tlofreqresamp10srrmat3, [], numfilts, 1);

% shift integral to have minimum at zero: subtract min for each column
thifreqresamp10sintmat2 = thifreqresamp10sintmat2 -
    repmat(min(thifreqresamp10sintmat2), numresamppts*numwks, 1);

% durations (sort each column)
thifreqresamp10sdur = sort(thifreqresamp10smat2);
tlofreqresamp10sdur = sort(tlofreqresamp10smat2);
thifreqresamp10sintdur = sort(thifreqresamp10sintmat2);
thifreqresamp10srrdur = sort(thifreqresamp10srrmat2);
tlofreqresamp10srrdur = sort(tlofreqresamp10srrmat2);

% dPdt vs Wmax
tmeanabslfrrresamp10s = mean(abs(tlofreqresamp10srrdur));
trmslfrrresamp10s = sqrt(mean(tlofreqresamp10srrdur.^2));
tmaxhfintresamp10s = thifreqresamp10sintdur(end,:);

%%%%% median filtered version

for filtindex = 1:numfilts;
    for wkindex = 1:numwks
        offset = tregmf5(1,wkindex) - tregmf5(end,wkindex);

        [waitforit, zi] = filter(bads4s(:,filtindex), aads4s(:,filtindex),
            (tregmf5(:,wkindex) + offset)); % calculate appropriate initial
            conditions: jigger so it ends where the next one starts
        hifreqtemp = filter(bads4s(:,filtindex), aads4s(:,filtindex),
            tregmf5(:,wkindex), zi); % use ICs from as if two identical
            days had been appended
        lofreqtemp = tregmf5(:,wkindex) - hifreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int4sec(hifreqtemp);
        lofreqrrtemp = rr4sec(lofreqtemp);
        hifreqrrtemp = rr4sec(hifreqtemp);

        % save data to the matrices
        thifreqmf5mat3(:, wkindex, filtindex) = hifreqtemp;
        tlofreqmf5mat3(:, wkindex, filtindex) = lofreqtemp;
        thifreqmf5intmat3(:, wkindex, filtindex) = hifreqinttemp;
        thifreqmf5rrmat3(:, wkindex, filtindex) = hifreqrrtemp;
        tlofreqmf5rrmat3(:, wkindex, filtindex) = lofreqrrtemp;
        offset = tregmf5(1,wkindex) - tregmf5(end,wkindex);
```

```
        [waitforit, zi] = filter(bads4s(:,filtindex), aads4s(:,filtindex),
            (tregmf13(:,wkindex) + offset)); % calculate appropriate
            initial conditions: jigger so it ends where the next one starts
        hifreqtemp = filter(bads4s(:,filtindex), aads4s(:,filtindex),
            tregmf13(:,wkindex), zi); % use ICs from as if two identical
            days had been appended
        lofreqtemp = tregmf13(:,wkindex) - hifreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int4sec(hifreqtemp);
        lofreqrrtemp = rr4sec(lofreqtemp);
        hifreqrrtemp = rr4sec(hifreqtemp);

        % save data to the matrices
        thifreqmf13mat3(:, wkindex, filtindex) = hifreqtemp;
        tlofreqmf13mat3(:, wkindex, filtindex) = lofreqtemp;
        thifreqmf13intmat3(:, wkindex, filtindex) = hifreqinttemp;
        thifreqmf13rrmat3(:, wkindex, filtindex) = hifreqrrtemp;
        tlofreqmf13rrmat3(:, wkindex, filtindex) = lofreqrrtemp;
    end
end

% reshape into column vectors, one for each filter

thifreqmf5mat2 = reshape(thifreqmf5mat3, [], numfilts, 1);
tlofreqmf5mat2 = reshape(tlofreqmf5mat3, [], numfilts, 1);
thifreqmf5intmat2 = reshape(thifreqmf5intmat3, [], numfilts, 1);
thifreqmf5rrmat2 = reshape(thifreqmf5rrmat3, [], numfilts, 1);
tlofreqmf5rrmat2 = reshape(tlofreqmf5rrmat3, [], numfilts, 1);

thifreqmf13mat2 = reshape(thifreqmf13mat3, [], numfilts, 1);
tlofreqmf13mat2 = reshape(tlofreqmf13mat3, [], numfilts, 1);
thifreqmf13intmat2 = reshape(thifreqmf13intmat3, [], numfilts, 1);
thifreqmf13rrmat2 = reshape(thifreqmf13rrmat3, [], numfilts, 1);
tlofreqmf13rrmat2 = reshape(tlofreqmf13rrmat3, [], numfilts, 1);

% shift integral to have minimum at zero: subtract min for each column
thifreqmf5intmat2 = thifreqmf5intmat2 - repmat(min(thifreqmf5intmat2),
    numpts1wk*numwks, 1);

thifreqmf13intmat2 = thifreqmf13intmat2 - repmat(min(thifreqmf13intmat2),
    numpts1wk*numwks, 1);

% durations (sort each column)
thifreqmf5dur = sort(thifreqmf5mat2);
tlofreqmf5dur = sort(tlofreqmf5mat2);
thifreqmf5intdur = sort(thifreqmf5intmat2);
thifreqmf5rrdur = sort(thifreqmf5rrmat2);
tlofreqmf5rrdur = sort(tlofreqmf5rrmat2);
```

```
thifreqmf13dur = sort(thifreqmf13mat2);
tlofreqmf13dur = sort(tlofreqmf13mat2);
thifreqmf13intdur = sort(thifreqmf13intmat2);
thifreqmf13rrdur = sort(thifreqmf13rrmat2);
tlofreqmf13rrdur = sort(tlofreqmf13rrmat2);

% dPdt vs Wmax
tmf5meanabslfrr = mean(abs(tlofreqmf5rrdur));
tmf5rmslfrr = sqrt(mean(tlofreqmf5rrdur.^2));
tmf5maxhfint = thifreqmf5intdur(end,:);

tmf13meanabslfrr = mean(abs(tlofreqmf13rrdur));
tmf13rmslfrr = sqrt(mean(tlofreqmf13rrdur.^2));
tmf13maxhfint = thifreqmf13intdur(end,:);

%% Cheb LP filtering

for filtindex = 1:numfilts;
    for dayindex = 1:numdays10s
        offset = sffuzz90mat(1,dayindex) - sffuzz90mat(end,dayindex);
        % also important to use the filter indicated by index2
        [waitforit, zi] = filter(bads10scheblp(:,filtindex),
            aads10scheblp(:,filtindex), (sffuzz90mat(:,dayindex) +
            offset)); % calculate appropriate initial conditions: jigger so
            it ends where the next one starts
        lofreqtemp = filter(bads10scheblp(:,filtindex),
            aads10scheblp(:,filtindex), sffuzz90mat(:,dayindex), zi); % use
            ICs from as if two identical days had been appended
        hifreqtemp = sffuzz90mat(:,dayindex) - lofreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int10sec(hifreqtemp);
        lofreqrrtemp = rr10sec(lofreqtemp);
        hifreqrrtemp = rr10sec(hifreqtemp);

        %store all vars in a vector
        sfflofreqcheblpmat3(:,dayindex, filtindex) = lofreqtemp;
        sffhifreqcheblpmat3(:,dayindex, filtindex) = hifreqtemp;
        sfflofreqcheblprrmat3(:,dayindex, filtindex) = lofreqrrtemp;
        sffhifreqcheblprrmat3(:,dayindex, filtindex) = hifreqrrtemp;
        sffhifreqcheblpintmat3(:,dayindex, filtindex) = hifreqinttemp;
    end
end

sfflofreqcheblpmat2 = reshape(sfflofreqcheblpmat3, [], numfilts);
sffhifreqcheblpmat2 = reshape(sffhifreqcheblpmat3, [], numfilts);
sfflofreqcheblprrmat2 = reshape(sfflofreqcheblprrmat3, [], numfilts);
sffhifreqcheblprrmat2 = reshape(sffhifreqcheblprrmat3, [], numfilts);
```

```
sffhifreqcheblpintmat2 = reshape(sffhifreqcheblpintmat3, [], numfilts);

sffhifreqcheblpintmat2 = sffhifreqcheblpintmat2 -
    repmat(min(sffhifreqcheblpintmat2), numpts10s, 1);

sfflofreqcheblpdur = sort(sfflofreqcheblpmat2);
sffhifreqcheblpdur = sort(sffhifreqcheblpmat2);
sfflofreqcheblprrdur = sort(sfflofreqcheblprrmat2);
sffhifreqcheblprrdur = sort(sffhifreqcheblprrmat2);
sffhifreqcheblpintdur = sort(sffhifreqcheblpintmat2);

% dPdt vs Wmax
sffmeanabslfrrcheblp = mean(abs(sfflofreqcheblprrdur));
sffrmslfrrcheblp = sqrt(mean(sfflofreqcheblprrdur.^2));
sffmaxhfintcheblp = sffhifreqcheblpintdur(end,:);


% pjm data also
for filtindex = 1:numfilts;
    for wkindex = 1:numwks
        offset = tregpjmmat(1,wkindex) - tregpjmmat(end,wkindex);
        % also important to use the filter indicated by index2
        [waitforit, zi] = filter(bads4scheblp(:,filtindex),
            aads4scheblp(:,filtindex), (tregpjmmat(:,wkindex) + offset)); %
            calculate appropriate initial conditions: jigger so it ends
            where the next one starts
        lofreqtemp = filter(bads4scheblp(:,filtindex),
            aads4scheblp(:,filtindex), tregpjmmat(:,wkindex), zi); % use
            ICs from as if two identical days had been appended
        hifreqtemp = tregpjmmat(:,wkindex) - lofreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int4sec(hifreqtemp);
        lofreqrrtemp = rr4sec(lofreqtemp);
        hifreqrrtemp = rr4sec(hifreqtemp);

        %store all vars in a vector
        tlofreqcheblpmat3(:,wkindex, filtindex) = lofreqtemp;
        thifreqcheblpmat3(:,wkindex, filtindex) = hifreqtemp;
        tlofreqcheblprrmat3(:,wkindex, filtindex) = lofreqrrtemp;
        thifreqcheblprrmat3(:,wkindex, filtindex) = hifreqrrtemp;
        thifreqcheblpintmat3(:,wkindex, filtindex) = hifreqinttemp;
    end
end

tlofreqcheblpmat2 = reshape(tlofreqcheblpmat3, [], numfilts);
thifreqcheblpmat2 = reshape(thifreqcheblpmat3, [], numfilts);
tlofreqcheblprrmat2 = reshape(tlofreqcheblprrmat3, [], numfilts);
thifreqcheblprrmat2 = reshape(thifreqcheblprrmat3, [], numfilts);
```

```
thifreqcheblpintmat2 = reshape(thifreqcheblpintmat3, [], numfilts);

thifreqcheblpintmat2 = thifreqcheblpintmat2 -
    repmat(min(thifreqcheblpintmat2), numpts1wk*numwks, 1);

tlofreqcheblpdur = sort(tlofreqcheblpmat2);
thifreqcheblpdur = sort(thifreqcheblpmat2);
tlofreqcheblprrdur = sort(tlofreqcheblprrmat2);
thifreqcheblprrdur = sort(thifreqcheblprrmat2);
thifreqcheblpintdur = sort(thifreqcheblpintmat2);

% dPdt vs Wmax
tmeanabslfrrcheblp = mean(abs(tlofreqcheblprrdur));
trmslfrrcheblp = sqrt(mean(tlofreqcheblprrdur.^2));
tmaxhfintcheblp = thifreqcheblpintdur(end,:);


%% Butterworth HP filtering

for filtindex = 1:numfilts;
    for dayindex = 1:numdays10s
        offset = sffuzz90mat(1,dayindex) - sffuzz90mat(end,dayindex);
        % also important to use the filter indicated by index2
        [waitforit, zi] = filter(bads10sbutter(:,filtindex),
            aads10sbutter(:,filtindex), (sffuzz90mat(:,dayindex) +
            offset)); % calculate appropriate initial conditions: jigger so
            it ends where the next one starts
        hifreqtemp = filter(bads10sbutter(:,filtindex),
            aads10sbutter(:,filtindex), sffuzz90mat(:,dayindex), zi); % use
            ICs from as if two identical days had been appended
        lofreqtemp = sffuzz90mat(:,dayindex) - hifreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int10sec(hifreqtemp);
        lofreqrrtemp = rr10sec(lofreqtemp);
        hifreqrrtemp = rr10sec(hifreqtemp);

        %store all vars in a vector
        sfflofreqbuttermat3(:,dayindex, filtindex) = lofreqtemp;
        sffhifreqbuttermat3(:,dayindex, filtindex) = hifreqtemp;
        sfflofreqbutterrrmat3(:,dayindex, filtindex) = lofreqrrtemp;
        sffhifreqbutterrrmat3(:,dayindex, filtindex) = hifreqrrtemp;
        sffhifreqbutterintmat3(:,dayindex, filtindex) = hifreqinttemp;
    end
end

sfflofreqbuttermat2 = reshape(sfflofreqbuttermat3, [], numfilts);
sffhifreqbuttermat2 = reshape(sffhifreqbuttermat3, [], numfilts);
sfflofreqbutterrrmat2 = reshape(sfflofreqbutterrrmat3, [], numfilts);
```

```
sffhifreqbutterrrmat2 = reshape(sffhifreqbutterrrmat3, [], numfilts);
sffhifreqbutterintmat2 = reshape(sffhifreqbutterintmat3, [], numfilts);

sffhifreqbutterintmat2 = sffhifreqbutterintmat2 -
    repmat(min(sffhifreqbutterintmat2), numpts10s, 1);

sfflofreqbutterdur = sort(sfflofreqbuttermat2);
sffhifreqbutterdur = sort(sffhifreqbuttermat2);
sfflofreqbutterrrdur = sort(sfflofreqbutterrrmat2);
sffhifreqbutterrrdur = sort(sffhifreqbutterrrmat2);
sffhifreqbutterintdur = sort(sffhifreqbutterintmat2);

% dPdt vs Wmax
sffmeanabslfrrbutter = mean(abs(sfflofreqbutterrrdur));
sffrmslfrrbutter = sqrt(mean(sfflofreqbutterrrdur.^2));
sffmaxhfintbutter = sffhifreqbutterintdur(end,:);


% pjm data also
for filtindex = 1:numfilts;
    for wkindex = 1:numwks
        offset = tregpjmmat(1,wkindex) - tregpjmmat(end,wkindex);
        % also important to use the filter indicated by index2
        [waitforit, zi] = filter(bads4sbutter(:,filtindex),
            aads4sbutter(:,filtindex), (tregpjmmat(:,wkindex) + offset)); %
            calculate appropriate initial conditions: jigger so it ends
            where the next one starts
        hifreqtemp = filter(bads4sbutter(:,filtindex),
            aads4sbutter(:,filtindex), tregpjmmat(:,wkindex), zi); % use
            ICs from as if two identical days had been appended
        lofreqtemp = tregpjmmat(:,wkindex) - hifreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int4sec(hifreqtemp);
        lofreqrrtemp = rr4sec(lofreqtemp);
        hifreqrrtemp = rr4sec(hifreqtemp);

        %store all vars in a vector
        tlofreqbuttermat3(:,wkindex, filtindex) = lofreqtemp;
        thifreqbuttermat3(:,wkindex, filtindex) = hifreqtemp;
        tlofreqbutterrrmat3(:,wkindex, filtindex) = lofreqrrtemp;
        thifreqbutterrrmat3(:,wkindex, filtindex) = hifreqrrtemp;
        thifreqbutterintmat3(:,wkindex, filtindex) = hifreqinttemp;
    end
end

tlofreqbuttermat2 = reshape(tlofreqbuttermat3, [], numfilts);
thifreqbuttermat2 = reshape(thifreqbuttermat3, [], numfilts);
tlofreqbutterrrmat2 = reshape(tlofreqbutterrrmat3, [], numfilts);
```

```
thifreqbutterrrmat2 = reshape(thifreqbutterrrmat3, [], numfilts);
thifreqbutterintmat2 = reshape(thifreqbutterintmat3, [], numfilts);

thifreqbutterintmat2 = thifreqbutterintmat2 -
    repmat(min(thifreqbutterintmat2), numpts1wk*numwks, 1);

tlofreqbutterdur = sort(tlofreqbuttermat2);
thifreqbutterdur = sort(thifreqbuttermat2);
tlofreqbutterrrdur = sort(tlofreqbutterrrmat2);
thifreqbutterrrdur = sort(thifreqbutterrrmat2);
thifreqbutterintdur = sort(thifreqbutterintmat2);

% dPdt vs Wmax
tmeanabslfrrbutter = mean(abs(tlofreqbutterrrdur));
trmslfrrbutter = sqrt(mean(tlofreqbutterrrdur.^2));
tmaxhfintbutter = thifreqbutterintdur(end,:);


%% Butterworth LP filtering

for filtindex = 1:numfilts;
    for dayindex = 1:numdays10s
        offset = sffuzz90mat(1,dayindex) - sffuzz90mat(end,dayindex);
        % also important to use the filter indicated by index2
        [waitforit, zi] = filter(bads10sbutterlp(:,filtindex),
            aads10sbutterlp(:,filtindex), (sffuzz90mat(:,dayindex) +
            offset)); % calculate appropriate initial conditions: jigger so
            it ends where the next one starts
        lofreqtemp = filter(bads10sbutterlp(:,filtindex),
            aads10sbutterlp(:,filtindex), sffuzz90mat(:,dayindex), zi); %
            use ICs from as if two identical days had been appended
        hifreqtemp = sffuzz90mat(:,dayindex) - lofreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int10sec(hifreqtemp);
        lofreqrrtemp = rr10sec(lofreqtemp);
        hifreqrrtemp = rr10sec(hifreqtemp);

        %store all vars in a vector
        sfflofreqbutterlpmat3(:,dayindex, filtindex) = lofreqtemp;
        sffhifreqbutterlpmat3(:,dayindex, filtindex) = hifreqtemp;
        sfflofreqbutterlprrmat3(:,dayindex, filtindex) = lofreqrrtemp;
        sffhifreqbutterlprrmat3(:,dayindex, filtindex) = hifreqrrtemp;
        sffhifreqbutterlpintmat3(:,dayindex, filtindex) = hifreqinttemp;
    end
end

sfflofreqbutterlpmat2 = reshape(sfflofreqbutterlpmat3, [], numfilts);
sffhifreqbutterlpmat2 = reshape(sffhifreqbutterlpmat3, [], numfilts);
```

```
sfflofreqbutterlprrmat2 = reshape(sfflofreqbutterlprrmat3, [], numfilts);
sffhifreqbutterlprrmat2 = reshape(sffhifreqbutterlprrmat3, [], numfilts);
sffhifreqbutterlpintmat2 = reshape(sffhifreqbutterlpintmat3, [], numfilts);

sffhifreqbutterlpintmat2 = sffhifreqbutterlpintmat2 -
    repmat(min(sffhifreqbutterlpintmat2), numpts10s, 1);

sfflofreqbutterlpdur = sort(sfflofreqbutterlpmat2);
sffhifreqbutterlpdur = sort(sffhifreqbutterlpmat2);
sfflofreqbutterlprrdur = sort(sfflofreqbutterlprrmat2);
sffhifreqbutterlprrdur = sort(sffhifreqbutterlprrmat2);
sffhifreqbutterlpintdur = sort(sffhifreqbutterlpintmat2);

% dPdt vs Wmax
sffmeanabslfrrbutterlp = mean(abs(sfflofreqbutterlprrdur));
sffrmslfrrbutterlp = sqrt(mean(sfflofreqbutterlprrdur.^2));
sffmaxhfintbutterlp = sffhifreqbutterlpintdur(end,:);


% pjm data also
for filtindex = 1:numfilts;
    for wkindex = 1:numwks
        offset = tregpjmmat(1,wkindex) - tregpjmmat(end,wkindex);
        % also important to use the filter indicated by index2
        [waitforit, zi] = filter(bads4sbutterlp(:,filtindex),
            aads4sbutterlp(:,filtindex), (tregpjmmat(:,wkindex) + offset));
            % calculate appropriate initial conditions: jigger so it ends
            where the next one starts
        lofreqtemp = filter(bads4sbutterlp(:,filtindex),
            aads4sbutterlp(:,filtindex), tregpjmmat(:,wkindex), zi); % use
            ICs from as if two identical days had been appended
        hifreqtemp = tregpjmmat(:,wkindex) - lofreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int4sec(hifreqtemp);
        lofreqrrtemp = rr4sec(lofreqtemp);
        hifreqrrtemp = rr4sec(hifreqtemp);

        %store all vars in a vector
        tlofreqbutterlpmat3(:,wkindex, filtindex) = lofreqtemp;
        thifreqbutterlpmat3(:,wkindex, filtindex) = hifreqtemp;
        tlofreqbutterlprrmat3(:,wkindex, filtindex) = lofreqrrtemp;
        thifreqbutterlprrmat3(:,wkindex, filtindex) = hifreqrrtemp;
        thifreqbutterlpintmat3(:,wkindex, filtindex) = hifreqinttemp;
    end
end

tlofreqbutterlpmat2 = reshape(tlofreqbutterlpmat3, [], numfilts);
thifreqbutterlpmat2 = reshape(thifreqbutterlpmat3, [], numfilts);
```

```
tlofreqbutterlprrmat2 = reshape(tlofreqbutterlprrmat3, [], numfilts);
thifreqbutterlprrmat2 = reshape(thifreqbutterlprrmat3, [], numfilts);
thifreqbutterlpintmat2 = reshape(thifreqbutterlpintmat3, [], numfilts);

thifreqbutterlpintmat2 = thifreqbutterlpintmat2 -
    repmat(min(thifreqbutterlpintmat2), numpts1wk*numwks, 1);

tlofreqbutterlpdur = sort(tlofreqbutterlpmat2);
thifreqbutterlpdur = sort(thifreqbutterlpmat2);
tlofreqbutterlprrdur = sort(tlofreqbutterlprrmat2);
thifreqbutterlprrdur = sort(thifreqbutterlprrmat2);
thifreqbutterlpintdur = sort(thifreqbutterlpintmat2);

% dPdt vs Wmax
tmeanabslfrrbutterlp = mean(abs(tlofreqbutterlprrdur));
trmslfrrbutterlp = sqrt(mean(tlofreqbutterlprrdur.^2));
tmaxhfintbutterlp = thifreqbutterlpintdur(end,:);


%% Bessel LP filtering

for filtindex = 1:numfilts;
    for dayindex = 1:numdays10s
        offset = sffuzz90mat(1,dayindex) - sffuzz90mat(end,dayindex);
        % also important to use the filter indicated by index2
        [waitforit, zi] = filter(bads10sbess(:,filtindex),
            aads10sbess(:,filtindex), (sffuzz90mat(:,dayindex) + offset));
            % calculate appropriate initial conditions: jigger so it ends
            where the next one starts
        lofreqtemp = filter(bads10sbess(:,filtindex),
            aads10sbess(:,filtindex), sffuzz90mat(:,dayindex), zi); % use
            ICs from as if two identical days had been appended
        hifreqtemp = sffuzz90mat(:,dayindex) - lofreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int10sec(hifreqtemp);
        lofreqrrtemp = rr10sec(lofreqtemp);
        hifreqrrtemp = rr10sec(hifreqtemp);

        %store all vars in a vector
        sfflofreqbessmat3(:,dayindex, filtindex) = lofreqtemp;
        sffhifreqbessmat3(:,dayindex, filtindex) = hifreqtemp;
        sfflofreqbessrrmat3(:,dayindex, filtindex) = lofreqrrtemp;
        sffhifreqbessrrmat3(:,dayindex, filtindex) = hifreqrrtemp;
        sffhifreqbessintmat3(:,dayindex, filtindex) = hifreqinttemp;
    end
end

sfflofreqbessmat2 = reshape(sfflofreqbessmat3, [], numfilts);
```

```
sffhifreqbessmat2 = reshape(sffhifreqbessmat3, [], numfilts);
sfflofreqbessrrmat2 = reshape(sfflofreqbessrrmat3, [], numfilts);
sffhifreqbessrrmat2 = reshape(sffhifreqbessrrmat3, [], numfilts);
sffhifreqbessintmat2 = reshape(sffhifreqbessintmat3, [], numfilts);

sffhifreqbessintmat2 = sffhifreqbessintmat2 -
    repmat(min(sffhifreqbessintmat2), numpts10s, 1);

sfflofreqbessdur = sort(sfflofreqbessmat2);
sffhifreqbessdur = sort(sffhifreqbessmat2);
sfflofreqbessrrdur = sort(sfflofreqbessrrmat2);
sffhifreqbessrrdur = sort(sffhifreqbessrrmat2);
sffhifreqbessintdur = sort(sffhifreqbessintmat2);

% dPdt vs Wmax
sffmeanabslfrrbess = mean(abs(sfflofreqbessrrdur));
sffrmslfrrbess = sqrt(mean(sfflofreqbessrrdur.^2));
sffmaxhfintbess = sffhifreqbessintdur(end,:);

% pjm data also
for filtindex = 1:numfilts;
    for wkindex = 1:numwks
        offset = tregpjmmat(1,wkindex) - tregpjmmat(end,wkindex);
        % also important to use the filter indicated by index2
        [waitforit, zi] = filter(bads4sbess(:,filtindex),
            aads4sbess(:,filtindex), (tregpjmmat(:,wkindex) + offset)); %
            calculate appropriate initial conditions: jigger so it ends
            where the next one starts
        lofreqtemp = filter(bads4sbess(:,filtindex),
            aads4sbess(:,filtindex), tregpjmmat(:,wkindex), zi); % use ICs
            from as if two identical days had been appended
        hifreqtemp = tregpjmmat(:,wkindex) - lofreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int4sec(hifreqtemp);
        lofreqrrtemp = rr4sec(lofreqtemp);
        hifreqrrtemp = rr4sec(hifreqtemp);

        %store all vars in a vector
        tlofreqbessmat3(:,wkindex, filtindex) = lofreqtemp;
        thifreqbessmat3(:,wkindex, filtindex) = hifreqtemp;
        tlofreqbessrrmat3(:,wkindex, filtindex) = lofreqrrtemp;
        thifreqbessrrmat3(:,wkindex, filtindex) = hifreqrrtemp;
        thifreqbessintmat3(:,wkindex, filtindex) = hifreqinttemp;
    end
end

tlofreqbessmat2 = reshape(tlofreqbessmat3, [], numfilts);
thifreqbessmat2 = reshape(thifreqbessmat3, [], numfilts);
```

```
tlofreqbessrrmat2 = reshape(tlofreqbessrrmat3, [], numfilts);
thifreqbessrrmat2 = reshape(thifreqbessrrmat3, [], numfilts);
thifreqbessintmat2 = reshape(thifreqbessintmat3, [], numfilts);

thifreqbessintmat2 = thifreqbessintmat2 - repmat(min(thifreqbessintmat2),
    numpts1wk*numwks, 1);

tlofreqbessdur = sort(tlofreqbessmat2);
thifreqbessdur = sort(thifreqbessmat2);
tlofreqbessrrdur = sort(tlofreqbessrrmat2);
thifreqbessrrdur = sort(thifreqbessrrmat2);
thifreqbessintdur = sort(thifreqbessintmat2);

% dPdt vs Wmax
tmeanabslfrrbess = mean(abs(tlofreqbessrrdur));
trmslfrrbess = sqrt(mean(tlofreqbessrrdur.^2));
tmaxhfintbess = thifreqbessintdur(end,:);

%% FIR

% long filters
for filtidx = 1:numFIRfilts
    for dayidx = 1:7
        offset = sffuzz90mat(1,dayidx) - sffuzz90mat(end,dayidx);
        [waitforit, zi] = filter(longfir1filt10s(filtidx,:), 1,
            (sffuzz90mat(:,dayidx) + offset));
        working = filter(longfir1filt10s(filtidx,:), 1,
            sffuzz90mat(:,dayidx));
        lofreq = working(longfiltdelaysamps10s:end);
        hifreq = sffuzz90mat(1:end-longfiltdelaysamps10s+1,dayidx) - lofreq;

        lofreqrr = rr10sec(lofreq);
        hifreqrr = rr10sec(hifreq);
        hifreqint = int10sec(hifreq);

        sfflofreqfirlongmat3(:,dayidx, filtidx) = lofreq;
        sffhifreqfirlongmat3(:,dayidx, filtidx) = hifreq;
        sffhifreqfirlongintmat3(:,dayidx, filtidx) = hifreqint;
        sfflofreqfirlongrrmat3(:,dayidx, filtidx) = lofreqrr;
        sffhifreqfirlongrrmat3(:,dayidx, filtidx) = hifreqrr;

        clear working lofreq hifreq lofreqrr hifreqrr hifreqint

    end

    sffhifreqfirlongintmat3(:,:,filtidx) = ...
        sffhifreqfirlongintmat3(:,:,filtidx) -
            min(min(sffhifreqfirlongintmat3(:,:,filtidx)));
```

```
    for wkidx = 1:4
        offset = tregpjmmat(1,wkidx) - tregpjmmat(end,wkidx);
        [waitforit, zi] = filter(longfir1filt4s(filtidx,:), 1,
            (tregpjmmat(:,wkidx) + offset));
        working = filter(longfir1filt4s(filtidx,:), 1, tregpjmmat(:,wkidx));
        lofreq = working(longfiltdelaysamps4s:end);
        hifreq = tregpjmmat(1:end-longfiltdelaysamps4s+1,wkidx) - lofreq;

        lofreqrr = rr4sec(lofreq);
        hifreqrr = rr4sec(hifreq);
        hifreqint = int4sec(hifreq);

        tlofreqfirlongmat3(:,wkidx, filtidx) = lofreq;
        thifreqfirlongmat3(:,wkidx, filtidx) = hifreq;
        thifreqfirlongintmat3(:,wkidx, filtidx) = hifreqint;
        tlofreqfirlongrrmat3(:,wkidx, filtidx) = lofreqrr;
        thifreqfirlongrrmat3(:,wkidx, filtidx) = hifreqrr;

        clear working lofreq hifreq lofreqrr hifreqrr hifreqint
    end

    thifreqfirlongintmat3(:,:,filtidx) = ...
        thifreqfirlongintmat3(:,:,filtidx) -
            min(min(thifreqfirlongintmat3(:,:,filtidx)));
end

sfflofreqfirlongmat2 = reshape(sfflofreqfirlongmat3, [], numfilts);
sffhifreqfirlongmat2 = reshape(sffhifreqfirlongmat3, [], numfilts);
sfflofreqfirlongrrmat2 = reshape(sfflofreqfirlongrrmat3, [], numfilts);
sffhifreqfirlongrrmat2 = reshape(sffhifreqfirlongrrmat3, [], numfilts);
sffhifreqfirlongintmat2 = reshape(sffhifreqfirlongintmat3, [], numfilts);

%already did this above
%sffhifreqfirlongintmat2 = sffhifreqfirlongintmat2 -
    repmat(min(sffhifreqfirlongintmat2), numpts10s, 1);

sfflofreqfirlongdur = sort(sfflofreqfirlongmat2);
sffhifreqfirlongdur = sort(sffhifreqfirlongmat2);
sfflofreqfirlongrrdur = sort(sfflofreqfirlongrrmat2);
sffhifreqfirlongrrdur = sort(sffhifreqfirlongrrmat2);
sffhifreqfirlongintdur = sort(sffhifreqfirlongintmat2);

% dPdt vs Wmax
sffmeanabslfrrfirlong = mean(abs(sfflofreqfirlongrrdur));
sffrmslfrrfirlong = sqrt(mean(sfflofreqfirlongrrdur.^2));
sffmaxhfintfirlong = sffhifreqfirlongintdur(end,:);

%and pjm
```

```
tlofreqfirlongmat2 = reshape(tlofreqfirlongmat3, [], numfilts);
thifreqfirlongmat2 = reshape(thifreqfirlongmat3, [], numfilts);
tlofreqfirlongrrmat2 = reshape(tlofreqfirlongrrmat3, [], numfilts);
thifreqfirlongrrmat2 = reshape(thifreqfirlongrrmat3, [], numfilts);
thifreqfirlongintmat2 = reshape(thifreqfirlongintmat3, [], numfilts);

%already did this above
%thifreqfirlongintmat2 = thifreqfirlongintmat2 -
    repmat(min(thifreqfirlongintmat2), numpts10s, 1);

tlofreqfirlongdur = sort(tlofreqfirlongmat2);
thifreqfirlongdur = sort(thifreqfirlongmat2);
tlofreqfirlongrrdur = sort(tlofreqfirlongrrmat2);
thifreqfirlongrrdur = sort(thifreqfirlongrrmat2);
thifreqfirlongintdur = sort(thifreqfirlongintmat2);

% dPdt vs Wmax
tmeanabslfrrfirlong = mean(abs(tlofreqfirlongrrdur));
trmslfrrfirlong = sqrt(mean(tlofreqfirlongrrdur.^2));
tmaxhfintfirlong = thifreqfirlongintdur(end,:);

% short filters
for filtidx = 1:numFIRfilts
    for dayidx = 1:7
        offset = sffuzz90mat(1,dayidx) - sffuzz90mat(end,dayidx);
        [waitforit, zi] = filter(shortfir1filts10s(:,filtidx), 1,
            (sffuzz90mat(:,dayidx) + offset));
        working = filter(shortfir1filts10s(:,filtidx), 1,
            sffuzz90mat(:,dayidx));
        lofreq = working(shortfiltdelaysamps10s:end);
        hifreq = sffuzz90mat(1:end-shortfiltdelaysamps10s+1,dayidx) -
            lofreq;

        lofreqrr = rr10sec(lofreq);
        hifreqrr = rr10sec(hifreq);
        hifreqint = int10sec(hifreq);

        sfflofreqfirshortmat3(:,dayidx, filtidx) = lofreq;
        sffhifreqfirshortmat3(:,dayidx, filtidx) = hifreq;
        sffhifreqfirshortintmat3(:,dayidx, filtidx) = hifreqint;
        sfflofreqfirshortrrmat3(:,dayidx, filtidx) = lofreqrr;
        sffhifreqfirshortrrmat3(:,dayidx, filtidx) = hifreqrr;

        clear working lofreq hifreq lofreqrr hifreqrr hifreqint

    end

    sffhifreqfirshortintmat3(:,:,filtidx) = ...
```

```
        sffhifreqfirshortintmat3(:,:,filtidx) -
            min(min(sffhifreqfirshortintmat3(:,:,filtidx)));

    for wkidx = 1:4
        offset = tregpjmmat(1,wkidx) - tregpjmmat(end,wkidx);
        [waitforit, zi] = filter(shortfir1filts4s(:,filtidx), 1,
            (tregpjmmat(:,wkidx) + offset));
        working = filter(shortfir1filts4s(:,filtidx), 1,
            tregpjmmat(:,wkidx));
        lofreq = working(shortfiltdelaysamps4s:end);
        hifreq = tregpjmmat(1:end-shortfiltdelaysamps4s+1,wkidx) - lofreq;

        lofreqrr = rr4sec(lofreq);
        hifreqrr = rr4sec(hifreq);
        hifreqint = int4sec(hifreq);

        tlofreqfirshortmat3(:,wkidx, filtidx) = lofreq;
        thifreqfirshortmat3(:,wkidx, filtidx) = hifreq;
        thifreqfirshortintmat3(:,wkidx, filtidx) = hifreqint;
        tlofreqfirshortrrmat3(:,wkidx, filtidx) = lofreqrr;
        thifreqfirshortrrmat3(:,wkidx, filtidx) = hifreqrr;

        clear working lofreq hifreq lofreqrr hifreqrr hifreqint
    end

    thifreqfirshortintmat3(:,:,filtidx) = ...
        thifreqfirshortintmat3(:,:,filtidx) -
            min(min(thifreqfirshortintmat3(:,:,filtidx)));
end

sfflofreqfirshortmat2 = reshape(sfflofreqfirshortmat3, [], numfilts);
sffhifreqfirshortmat2 = reshape(sffhifreqfirshortmat3, [], numfilts);
sfflofreqfirshortrrmat2 = reshape(sfflofreqfirshortrrmat3, [], numfilts);
sffhifreqfirshortrrmat2 = reshape(sffhifreqfirshortrrmat3, [], numfilts);
sffhifreqfirshortintmat2 = reshape(sffhifreqfirshortintmat3, [], numfilts);

%already did this above
%sffhifreqfirshortintmat2 = sffhifreqfirshortintmat2 -
    repmat(min(sffhifreqfirshortintmat2), numpts10s, 1);

sfflofreqfirshortdur = sort(sfflofreqfirshortmat2);
sffhifreqfirshortdur = sort(sffhifreqfirshortmat2);
sfflofreqfirshortrrdur = sort(sfflofreqfirshortrrmat2);
sffhifreqfirshortrrdur = sort(sffhifreqfirshortrrmat2);
sffhifreqfirshortintdur = sort(sffhifreqfirshortintmat2);

% dPdt vs Wmax
sffmeanabslfrrfirshort = mean(abs(sfflofreqfirshortrrdur));
sffrmslfrrfirshort = sqrt(mean(sfflofreqfirshortrrdur.^2));
```

```
sffmaxhfintfirshort = sffhifreqfirshortintdur(end,:);

%and pjm

tlofreqfirshortmat2 = reshape(tlofreqfirshortmat3, [], numfilts);
thifreqfirshortmat2 = reshape(thifreqfirshortmat3, [], numfilts);
tlofreqfirshortrrmat2 = reshape(tlofreqfirshortrrmat3, [], numfilts);
thifreqfirshortrrmat2 = reshape(thifreqfirshortrrmat3, [], numfilts);
thifreqfirshortintmat2 = reshape(thifreqfirshortintmat3, [], numfilts);

%already did this above
%thifreqfirshortintmat2 = thifreqfirshortintmat2 -
    repmat(min(thifreqfirshortintmat2), numpts10s, 1);

tlofreqfirshortdur = sort(tlofreqfirshortmat2);
thifreqfirshortdur = sort(thifreqfirshortmat2);
tlofreqfirshortrrdur = sort(tlofreqfirshortrrmat2);
thifreqfirshortrrdur = sort(thifreqfirshortrrmat2);
thifreqfirshortintdur = sort(thifreqfirshortintmat2);

% dPdt vs Wmax
tmeanabslfrrfirshort = mean(abs(tlofreqfirshortrrdur));
trmslfrrfirshort = sqrt(mean(tlofreqfirshortrrdur.^2));
tmaxhfintfirshort = thifreqfirshortintdur(end,:);
```

C.3.4 Building Autoregressive Models

thesis_makefigs_ch_newpjmdata_subfile_build_AR_models.m

```
%% Olivia Leitermann  MIT LEES  10 jan 12
%% subfile to make thesis figures, new pjm data chapter
%% called by thesis_makefigs_ch_newpjmdata_<date>.m
%% build and analyze AR models needed in this chapter

%% Fitting an AR model to the various data sets

% make the data objects

datapjm1 = iddata(tregpjmmat(:,1), [], 4); % sampling time is 4sec
datapjm2 = iddata(tregpjmmat(:,2), [], 4);
datapjm3 = iddata(tregpjmmat(:,3), [], 4);
datapjm4 = iddata(tregpjmmat(:,4), [], 4);

data10s1 = iddata(sffuzz90mat(:,1), [], 10); % sampling time is 10 sec
data10s2 = iddata(sffuzz90mat(:,2), [], 10);
data10s3 = iddata(sffuzz90mat(:,3), [], 10);
data10s4 = iddata(sffuzz90mat(:,4), [], 10);
```

```
data10s5 = iddata(sffuzz90mat(:,5), [], 10);
data10s6 = iddata(sffuzz90mat(:,6), [], 10);
data10s7 = iddata(sffuzz90mat(:,7), [], 10);

p10smax = 15;
ppjmmax = 15;

% store phi tap weights COLUMNWISE, not how they come rowwise
ARmeanparam10s = zeros(p10smax+1, p10smax);
ARmeanparampjm = zeros(ppjmmax+1, ppjmmax);

for index = 1:p10smax

    % default method is "forward-backward" approach with no padding outside
    % data set, seems ok
    armod10s1 = ar(data10s1, index);
    armod10s2 = ar(data10s2, index);
    armod10s3 = ar(data10s3, index);
    armod10s4 = ar(data10s4, index);
    armod10s5 = ar(data10s5, index);
    armod10s6 = ar(data10s6, index);
    armod10s7 = ar(data10s7, index);
    % ar creates an "idpoly" model
    % get parameters with arxdata
    [A10s1,~,sigA10s1,~] = arxdata(armod10s1);
    [A10s2,~,sigA10s2,~] = arxdata(armod10s2);
    [A10s3,~,sigA10s3,~] = arxdata(armod10s3);
    [A10s4,~,sigA10s4,~] = arxdata(armod10s4);
    [A10s5,~,sigA10s5,~] = arxdata(armod10s5);
    [A10s6,~,sigA10s6,~] = arxdata(armod10s6);
    [A10s7,~,sigA10s7,~] = arxdata(armod10s7);

    meanA10s = mean([A10s1; A10s2; A10s3; A10s4; A10s5; A10s6; A10s7]);
    meansigA10s = mean([sigA10s1; sigA10s2; sigA10s3; sigA10s4; sigA10s5;
        sigA10s6; sigA10s7]);
    ARmeanparam10s(1:index+1,index) = meanA10s';

    % first element is zero to get strict causality, negative sign b/c of
    % conventions in ardxata format
    predictfilt10s = [0, -meanA10s(2:end)]'; % column of filter for 10s
        prediction

    %filter operates on each column of data matrix separately.  denom is 1.
    predictresult10s = filter(predictfilt10s, 1, sffuzz90mat);

    % truncate to eliminate bad points at beginning: ppjm or p10s + 2
    predictresult10s = predictresult10s(p10smax+2:end, :);

    % calculate residuals z - zhat
```

```
    resid10s(:,:,index) = -predictresult10s + sffuzz90mat(p10smax+2:end,:);

end

resid10smat2 = reshape(resid10s, [], p10smax);
resid10smat2dur = sort(resid10smat2);

for index = 1:ppjmmax
    armodpjm1 = ar(datapjm1, index);
    armodpjm2 = ar(datapjm2, index);
    armodpjm3 = ar(datapjm3, index);
    armodpjm4 = ar(datapjm4, index);
    % ar creates an "idpoly" model

    % get the parameters with arxdata
    [Apjm1,~,sigApjm1,~] = arxdata(armodpjm1);
    [Apjm2,~,sigApjm2,~] = arxdata(armodpjm2);
    [Apjm3,~,sigApjm3,~] = arxdata(armodpjm3);
    [Apjm4,~,sigApjm4,~] = arxdata(armodpjm4);

    meanApjm = mean([Apjm1; Apjm2; Apjm3; Apjm4]);
    meansigApjm = mean([sigApjm1; sigApjm2; sigApjm3; sigApjm4]);

    ARmeanparampjm(1:index+1,index) = meanApjm';

    % first element is zero to get strict causality, negative sign b/c of
    % conventions in ardxata format
    predictfiltpjm = [0, -meanApjm(2:end)]'; % "b" vector for filtering

    %filter operates on each column of data matrix separately.  denom is 1.
    predictresultpjm = filter(predictfiltpjm, 1, tregpjmmat);

    % truncate to eliminate bad points at beginning: ppjm or p10s + 2
    predictresultpjm = predictresultpjm(ppjmmax+2:end, :);

    % calculate residuals z - zhat
    residpjm(:,:,index) = -predictresultpjm + tregpjmmat(ppjmmax+2:end,:);
end

residpjmmat2 = reshape(residpjm, [], ppjmmax);
residpjmmat2dur = sort(residpjmmat2);


%% need autocorrelation of the residuals
pick10sorder = 7;
pickpjmorder = 12;
nlags10s = 2*3600/10; % 2 hr worth of lags
nlags4s = 2*3600/4;
M = 3; % should definitely not have anything going on beyond 3 lags
```

```
for index = 1:numdays10s
    [acfresid10s7temp,acfresid10s7lags,acfresid10s7bounds] =
        autocorr(resid10s(:,index,7),nlags10s,M,[]);
    acfresid10s7(:,index) = acfresid10s7temp;
end

meanacfresid10s7 = mean(acfresid10s7,2); % mean of each row (should be mean
    for a lag across days)

for index = 1:numwks
    [acfresidpjm12temp,acfresidpjm12lags,acfresidpjm12bounds] =
        autocorr(residpjm(:,index,12),nlags4s,M,[]);
    acfresidpjm12(:,index) = acfresidpjm12temp;
end

meanacfresidpjm12 = mean(acfresidpjm12,2); % mean of each row (should be
    mean across weeks for a given lag

%% also need frequency domain behavior of the AR model filters
numfreqsamps = 512;
fs10s = 1/10;
fs4s = 1/4;

[armodfreqresp10s7,armodfreq10s7] =
    freqz(1,ARmeanparam10s(:,7),numfreqsamps,fs10s);
[armodfreqresppjm12,armodfreqpjm12] =
    freqz(1,ARmeanparampjm(:,12),numfreqsamps,fs4s);

armodfreqmagresp10s7 = abs(armodfreqresp10s7);
armodfreqphresp10s7 = angle(armodfreqresp10s7);

armodfreqmagresppjm12 = abs(armodfreqresppjm12);
armodfreqphresppjm12 = angle(armodfreqresppjm12);
```

C.3.5 Creating and Analyzing Synthetic Data

thesis_makefigs_ch_newpjmdata_subfile_make_ana_syndat.m

```
%% Olivia Leitermann  MIT LEES  10 jan 12
%% subfile to make thesis figures, new pjm data chapter
%% called by thesis_makefigs_ch_newpjmdata_<date>.m
%% create and analyze synthetic data (needs filters)

numsimpts = 1e5;
syndurx = linspace(0, 1, numsimpts)';
syntimehr10s = linspace(0, (numsimpts-1)*10/3600, numsimpts)';
```

```
syntimehr4s = linspace(0, (numsimpts-1)*4/3600, numsimpts)';
% shocks with the empirical dist
% need to do this for every model if I want syndata for every p order


for index = 1:p10smax
    % choose the appropriate residual distribution for the guys
    tempresiddur = resid10smat2dur(:,index);
    tempurand10s = round(rand(numsimpts,1)*numel(tempresiddur)); % becomes
        the index of the cdf
    % in case any of the guys are too small and round to zero
    ind = find(tempurand10s == 0);
    tempurand10s(ind) = 1;
    tempempshocks10s = tempresiddur(tempurand10s);

    % now use these shocks to model the data and see if it's any better
    % filter the shocks to get the AR data
    tempsyndat = filter(1, ARmeanparam10s(:,index), tempempshocks10s);
    empsyndat10s(:,index) = tempsyndat;
    % rr, int, duration
    tempsyndatrr = rr10sec(tempsyndat);
    empsyndatrr10s(:,index) = tempsyndatrr;
    tempsyndatint = int10sec(tempsyndat);
    empsyndatint10s(:,index) = tempsyndatint - min(tempsyndatint);

    empsyndat10sdur(:,index) = sort(tempsyndat);
    empsyndatrr10sdur(:,index) = sort(tempsyndatrr);
    empsyndatint10sdur(:,index) = sort(tempsyndatint);

end

empsyndat10smeanabsrr = mean(abs(empsyndatrr10s)); % one row, cols are
    orders

for index = 1:ppjmmax
    tempresiddur = residpjmmat2dur(:,index);
    tempurandpjm = round(rand(numsimpts, 1)*numel(tempresiddur));
    ind = find(tempurandpjm == 0);
    tempurandpjm(ind) = 1;
    tempempshockspjm = tempresiddur(tempurandpjm);
    empsyndatpjm(:,index) = filter(1, ARmeanparampjm(:,index),
        tempempshockspjm);

    tempsyndatrr = rr4sec(tempsyndat);
    empsyndatrrpjm(:,index) = tempsyndatrr;
    tempsyndatint = int4sec(tempsyndat);
    empsyndatintpjm(:,index) = tempsyndatint - min(tempsyndatint);

    empsyndatpjmdur(:,index) = sort(tempsyndat);
```

```
    empsyndatrrpjmdur(:,index) = sort(tempsyndatrr);
    empsyndatintpjmdur(:,index) = sort(tempsyndatint);
end

empsyndatpjmmeanabsrr = mean(abs(empsyndatrrpjm)); % one row, cols are
    orders

syntime4shr = linspace(0, (numsimpts-1)*4/3600, numsimpts)';

syntime10shr = linspace(0, (numsimpts-1)*10/3600, numsimpts)';



%% filter with the cheby hp filter
clear hifreqinttemp lofreqrrtemp hifreqrrtemp

for filtindex = 1:numfilts;
    for orderindex = 1:p10smax
        offset = empsyndat10s(1,orderindex) - empsyndat10s(end,orderindex);
        % also important to use the filter indicated by index2
        [waitforit, zi] = filter(bads10s(:,filtindex),
            aads10s(:,filtindex), (empsyndat10s(:,orderindex) + offset)); %
            calculate appropriate initial conditions: jigger so it ends
            where the next one starts
        hifreqtemp = filter(bads10s(:,filtindex), aads10s(:,filtindex),
            empsyndat10s(:,orderindex), zi); % use ICs from as if two
            identical days had been appended
        lofreqtemp = empsyndat10s(:,orderindex) - hifreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int10sec(hifreqtemp) ;
        hifreqinttemp = hifreqinttemp - min(hifreqinttemp);
        % slide to zero minimum on energy
        lofreqrrtemp = rr10sec(lofreqtemp);
        hifreqrrtemp = rr10sec(hifreqtemp);


        %store all vars in a vector
        syn10slofreqmat3(:,orderindex, filtindex) = lofreqtemp;
        syn10shifreqmat3(:,orderindex, filtindex) = hifreqtemp;
        syn10slofreqrrmat3(:,orderindex, filtindex) = lofreqrrtemp;
        syn10shifreqrrmat3(:,orderindex, filtindex) = hifreqrrtemp;
        syn10shifreqintmat3(:,orderindex, filtindex) = hifreqinttemp;
    end

end

syn10slofreqmat3dur = sort(syn10slofreqmat3);
syn10shifreqmat3dur = sort(syn10shifreqmat3);
```

```
syn10shifreqrrmat3dur = sort(syn10shifreqrrmat3);
syn10slofreqrrmat3dur = sort(syn10slofreqrrmat3);
syn10shifreqintmat3dur = sort(syn10shifreqintmat3);

syn10smeanabslfrr = mean(abs(syn10slofreqrrmat3dur)); % should be mean of
    each col (in each page)
syn10smeanabslfrr = squeeze(syn10smeanabslfrr); % now rows are order and
    cols are filter
syn10smaxhfint = syn10shifreqintmat3dur(end,:,:);
syn10smaxhfint = squeeze(syn10smaxhfint); % again rows are order and cols
    are filter

% and the pjm like data

for filtindex = 1:numfilts;
    for orderindex = 1:ppjmmax
        offset = empsyndatpjm(1,orderindex) - empsyndatpjm(end,orderindex);
        % also important to use the filter indicated by index2
        [waitforit, zi] = filter(bads4s(:,filtindex), aads4s(:,filtindex),
            (empsyndatpjm(:,orderindex) + offset)); % calculate appropriate
            initial conditions: jigger so it ends where the next one starts
        hifreqtemp = filter(bads4s(:,filtindex), aads4s(:,filtindex),
            empsyndatpjm(:,orderindex), zi); % use ICs from as if two
            identical days had been appended
        lofreqtemp = empsyndatpjm(:,orderindex) - hifreqtemp;

        % calculate integral and ramp rate
        hifreqinttemp = int4sec(hifreqtemp) ;
        hifreqinttemp = hifreqinttemp - min(hifreqinttemp);
        % slide to zero minimum on energy
        lofreqrrtemp = rr4sec(lofreqtemp);
        hifreqrrtemp = rr4sec(hifreqtemp);

        %store all vars in a vector
        synpjmlofreqmat3(:,orderindex, filtindex) = lofreqtemp;
        synpjmhifreqmat3(:,orderindex, filtindex) = hifreqtemp;
        synpjmlofreqrrmat3(:,orderindex, filtindex) = lofreqrrtemp;
        synpjmhifreqrrmat3(:,orderindex, filtindex) = hifreqrrtemp;
        synpjmhifreqintmat3(:,orderindex, filtindex) = hifreqinttemp;
    end
end

synpjmlofreqmat3dur = sort(synpjmlofreqmat3);
synpjmhifreqmat3dur = sort(synpjmhifreqmat3);
synpjmhifreqrrmat3dur = sort(synpjmhifreqrrmat3);
synpjmlofreqrrmat3dur = sort(synpjmlofreqrrmat3);
synpjmhifreqintmat3dur = sort(synpjmhifreqintmat3);
```

```
synpjmmeanabslfrr = mean(abs(synpjmlofreqrrmat3dur)); % should be mean of
    each col (in each page)
synpjmmeanabslfrr = squeeze(synpjmmeanabslfrr); % now rows are order and
    cols are filter
synpjmmaxhfint = synpjmhifreqintmat3dur(end,:,:);
synpjmmaxhfint = squeeze(synpjmmaxhfint); % again rows are order and cols
    are filter
```

C.3.6 Comparing a Structured Signal

thesis_makefigs_ch_newpjmdata_subfile_trapwave_noplots.m

```
%% Olivia Leitermann  MIT LEES  10 jan 12
%% subfile to make thesis figures, new pjm data chapter
%% called by thesis_makefigs_ch_newpjmdata_<date>.m
%% create the trapezoidal wave experiment and the required plots

% figures needed
%      better filters
% trapwave_partsum_2
% trap_ramp_vs_partsum


numptsa = 1e2;
numptst = 1e4;

% unit trapezoid waveform: amplitude 1, period 1, symmetric, each flat part
% has length a
t = linspace(0, 1, numptst);

a = linspace(0, 1/2-1/numptsa, numptsa)'; %vector of a values

n = [1 3 5 7 9 11 13]';
numterms = numel(n);

for indexn = 1:numterms
   nnow = n(indexn);

   for indexa = 1:numptsa
      anow = a(indexa);
      % each term in the df/dt series has a range of values for n, a, t
      term(:,indexa,indexn) =
          1/nnow*cos(nnow*pi*anow).*sin(2*pi*nnow*t)/(1-2*anow);
   end
end

term = term*8/pi; % scale by the constant factors
```

```
% now sum the terms up to get the values for each choice of max n

for indexn = 1:numterms
   psums(:,:,indexn) = sum(term(:,:,1:indexn),3); % sum along 3rd dim, up
       to current choice of n_max
end

% then we can take the absolute value (for a given value of a and t)

abspsums = abs(psums);
sqpsums = psums.^2;

% finally take the time average over t for a given a and max n

intabspsums = trapz(t, abspsums); % should integrate over t from 0 to 1
% intabspsums should be 1 x numpts(a) x numterms
intsqpsums = trapz(t, sqpsums);

intabspsums = squeeze(intabspsums);
% then rows are values of a, columns are different n_max
intsqpsums = squeeze(intsqpsums);

rmspsums = sqrt(intsqpsums);

%% look at the RMS - calculate for exact and plot for partial sums

exactrms = 2./sqrt(1-2.*a);

% just check the first two psums for rms

calcrms1 = 4*sqrt(2)/pi./(1-2*a).*cos(pi*a);
calcrms2 = 4*sqrt(2)/pi./(1-2*a).*sqrt(cos(pi*a).^2 + cos(3*pi*a).^2/9);


errcalc1rms = calcrms1 - rmspsums(:,1);
maxerrcalc1rms = max(errcalc1rms);
errcalc2rms = calcrms2 - rmspsums(:,2);
maxerrcalc2rms = max(errcalc2rms);

maxerrcalc1rms
maxerrcalc2rms

%% want to check the calculations that I did: do 2 terms out by "hand", and
%% compare the curves

for index = 1:numptsa
    anow = a(index);
    % value of f(x) for first two terms for all t for each a
```

```
    f1(:,index) = 4/pi^2/(1-2*anow)*cos(pi*anow)*cos(2*pi*t);
    f2(:,index) = 4/pi^2/(1-2*anow)*(cos(pi*anow)*cos(2*pi*t) +
        1/9*cos(3*pi*anow)*cos(6*pi*t));
end

df2dt = numptst*diff(f2); % should do diff for each a (vector)
df1dt = numptst*diff(f1);
%df2dt = [zeros(1,numptsa); df2dt]; % add a first zero row because that's
    the way I'd been doing it
absdf2dt = abs(df2dt);
meanabsdf2dt = mean(absdf2dt); % mean along each col: cols are different a
    vals
absdf1dt = abs(df1dt);
meanabsdf1dt = mean(abs(df1dt));



% a = .25 exact
anow = .25;
exacttrap25 = zeros(numptst,1);
exacttrap25(1:numptst*anow/2) = .5;
exacttrap25(numptst/2-anow*numptst/2:numptst/2+anow*numptst/2) = -.5;
exacttrap25(end-anow*numptst/2:end) = .5;
exacttrap25(numptst*anow/2+1:numptst/2*(1-anow)) = linspace(.5, -.5,
    numptst/4);
exacttrap25(numptst/2*(1+anow)+1:end-anow*numptst/2) = linspace(-.5, .5,
    numptst/4);

dexactdt25 = numptst*diff(exacttrap25);
absdexactdt25 = abs(dexactdt25);
meanabsdexactdt25 = mean(absdexactdt25);
meanabsdexactdt25
meanabsdf2dt(50)
intabspsums(50,2)


rmsdexactdt25 = sqrt(mean(dexactdt25.^2));
calcexactrms25 = 2*sqrt(2);

rmsdexactdt25
calcexactrms25
```

Bibliography

[1] N. Atić, D. Rerkpreedapong, A. Hasanovic, and A. Feliachi, "NERC compliant decentralized load frequency control design using model predictive control," in *IEEE PES General Meeting*. IEEE Power Engineering Society, July 2003, pp. 554–559.

[2] A. R. Bergen and V. Vittal, *Power Systems Analysis*, 2nd ed. Prentice Hall, 2000.

[3] R. W. Boom, "Superconductive energy storage for diurnal use by electric utilities," *IEEE Transactions on Magnetics*, vol. MAG-17, no. 1, pp. 340–343, Jan 1981.

[4] R. R. Booth, "Power system simulation model based on probability analysis," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-91, no. 1, pp. 62–69, Jan 1972.

[5] A. Bose and I. Atiyyah, "Regulation errorin load frequency control," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-99, no. 2, pp. 650–657, 1980.

[6] B. L. Bowerman, R. T. O'Connell, and A. B. Koehler, *Forecasting, Time Series, and Regression*, 4th ed. Thomson Brooks Cole, 2005.

[7] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, 1st ed. Holden-Day, 1970.

[8] D. Bradwell, G. Ceder, L. Ortiz, and D. R. Sadoway, "Liquid electrode battery," US Application 2011/0 014 505 A1, 2011.

[9] D. Bradwell, H. Kim, A. H. C. Sirk, and D. R. Sadoway, "Magnesium-antimony liquid metal battery for stationary energy storage," *Journal of the American Chemical Society*, vol. 134, no. 4, pp. 1895–1897, 2012.

[10] V. Brandwajn, A. Ipakchi, and V. Sherkat, "Tracking evolutionary trends in generation control," *IEEE Computer Applications in Power*, vol. 6, no. 1, pp. 22–26, January 1993.

[11] R. P. Broehm, R. L. Earle, F. C. Graves, T. J. Jenkin, and D. M. Murphy, "Mechanisms for evaluating the role of hydroelectric generation in ancillary service markets," EPRI, Palo Alto, CA, Tech. Rep. TR-111707, 1998.

[12] D. W. Bunn, "Forecasting loads and prices in competitive power markets," *Proceedings of the IEEE*, vol. 88, no. 2, pp. 163–169, February 2000.

[13] D. W. Bunn and E. D. Farmer, Eds., *Comparative Models for Electrical Load Forecasting.* John Wiley and Sons, 1985.

[14] A. M. Conner, J. E. Francfort, and B. N. Rinehart, "U.S. hydropower resource assessment final report," Idaho National Engineering and Environmental Laboratory, National Lab DOE Report DOE/ID-10430.2, Dec 1998.

[15] G. M. Cook, W. C. Spindler, and G. Grefe, "Overview of battery power regulation and storage," *IEEE Transactions on Energy Conversion*, vol. 6, no. 1, pp. 204–211, Mar 1991.

[16] F. Crotogino, K.-U. Mohmeyer, and R. Scharf, "Huntorf CAES: More than 20 years of successful operation," in *Solution Mining Research Institute Spring Meeting*, Apr 2001.

[17] Current Operational Problems Working Group, M. D. Anderson, A. J. Connor, F. I. Denny, J. R. Huff, T. Kennedy, and C. J. Frank, "Current operating problems associated with automatic generation control," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-98, no. 1, pp. 88–96, January 1979.

[18] F. P. de Mello, "Boiler models for system dynamic performance studies," *IEEE Transactions on Power Systems*, vol. 6, no. 1, pp. 66–74, 1991.

[19] F. P. de Mello and R. J. Mills, "Automatic generation control part II: Digital control techniques," in *IEEE PES Summer Meeting.* San Francisco, CA: IEEE Power and Energy Society, July 1972, pp. 716–724.

[20] J. K. Delson, "Thermal stress computation for steam-electric generator dispatch," *IEEE Transactions on Power Systems*, vol. 9, no. 1, pp. 120–127, Feb. 1994.

[21] E. Denny, "The economics of tidal power," in *IEEE Power and Energy Society General Meeting*, 2010.

[22] L. D. Douglas, T. A. Green, and R. A. Kramer, "New approaches to the AGC nonconforming load problem," *IEEE Transactions on Power Systems*, vol. 9, no. 2, pp. 619–628, May 1994.

[23] E. D. Eason, A. A. Merton, and E. E. Nelson, "Correlating cycle duty with cost at fossil fuel power plants," Electric Power Research Institute, Palo Alto, CA, Tech. Rep. 1008351, September 2001.

[24] C. L. Edgar, "Practical experience with storage batteries in central stations," in *101st Meeting of the American Institute of Electrical Engineers*, New York and Chicago, November 1895, pp. 592–598.

[25] I. Egido, F. Fernandez-Bernal, L. Rouco, E. Porras, and A. Saiz-Chicharro, "Modeling of thermal generating units for automatic generation control purposes," *IEEE Transactions on Control Systems Technology*, vol. 12, no. 1, pp. 205–210, January 2004.

[26] Electric Power Research Institute, K. Q. Chang, and F. P. de Mello, "Transient efficiencies in electric power plants," Electric Power Research Institute, Palo Alto, CA, Research Project Final Report EL-2439, June 1982.

[27] Electric Power Research Institute, I. Gyuk, and S. Eckroad, "EPRI-DOE handbook of energy storage for transmission and distribution applications," U.S. Department of Energy, Palo Alto, CA and Washington, D.C., Tech. Rep. 1001834, Dec 2003.

[28] ENBALA Power Networks, Inc., "Pennsylvania american water connects to the smart grid," `http://www.enbala.com/media/casestudies/Pennsylvania American Water Connects to the Smart Grid.pdf`.

[29] I. A. Erinmez, D. O. Bickers, G. F. Wood, and W. W. Hung, "NGC experience with frequency control in England and Wales—provision of frequency response by generators," in *IEEE Power Engineering Society Winter Meeting*, 1999, pp. 590–596.

[30] I. Erlich and M. Wilch, "Primary frequency control by wind turbines," in *IEEE Power and Energy Society General Meeting*, 2010.

[31] P. Fairley, "Flywheels keep the grid in tune," *IEEE Spectrum*, July 2011.

[32] *Frequency Regulation Compensation in the Organized Wholesale Power Markets*, Federal Energy Regulatory Commission Order 755 Docket Nos. RM11-7-000 and AD10-11-000, October 20 2011. [Online]. Available: `http://www.ferc.gov/whats-new/comm-meet/2011/102011/E-28.pdf`

[33] D. Feroldi, M. Serra, and J. Riera, "Design and analysis of fuel-cell hybrid systems oriented to automotive applications," *IEEE Transactions on Vehicular Technology*, vol. 58, pp. 4720–4729, 2009.

[34] A. E. Fitzgerald, C. K. Jr., and S. D. Umans, *Electric Machinery*, 6th ed. McGraw-Hill, New York, NY, 2003, ch. 5 Synchronous Machines, pp. 245–297.

[35] B. Geng, J. K. Mills, and D. Sun, "Two-stage energy management control of fuel cell plug-in hybrid electric vehicles considering fuel cell longevity," *IEEE Transactions on Vehicular Technology*, vol. 61, pp. 498–508, 2012.

[36] R. K. Green, "Transformed automatic generation control," *IEEE Transactions on Power Systems*, vol. 11, no. 4, pp. 1799–1804, November 1996.

[37] G. Gross, "Analysis of load frequency control performance assessment criteria," *IEEE Transactions on Power Systems*, vol. 16, no. 3, pp. 520–525, August 2001.

[38] I. Gyuk, P. Kulkarni, J. H. Sayer, J. D. Boyes, G. P. Corey, and G. H. Peek, "The United States of storage," *IEEE Power and Energy Magazine*, pp. 31–39, Mar/Apr 2005.

[39] Y. Hain, R. Kulessky, and G. Nudelman, "Identification-based power unit model for load-frequency control purposes," *IEEE Transactions on Power Systems*, vol. 15, no. 4, pp. 1313–1321, November 2000.

[40] L. N. Hannett and A. H. Khan, "Combustion turbine dynamic model validation from tests," *IEEE Transactions on Power Systems*, vol. 8, no. 1, pp. 152–158, 1993.

[41] W. V. Hassenzahl, "Superconducting magnetic energy storage," *Proceedings of the IEEE*, vol. 71, no. 9, pp. 1089–1098, September 1983.

[42] P. D. Henderson, H. Kalaiman, J. Ginetti, T. Snodgrass, N. Cohn, S. Bloor, and L. VanSlyck, "Cost aspects of AGC, inadvertent energy and time error," *IEEE Transactions on Power Systems*, vol. 5, no. 1, pp. 111–118, February 1990.

[43] E. Hirst and B. Kirby, "Defining intra- and inter-hour load swings," *IEEE Transactions on Power Systems*, vol. 13, no. 4, pp. 1379–1385, Nov. 1998.

[44] D. Howell, "Progress report for energy storage research and development, fiscal year 2008," U.S. Department of Energy, Office of Vehicle Technologies, Washington, D.C., Tech. Rep., January 2009, available at `http://www1.eere.energy.gov/vehiclesandfuels/pdfs/program/2008_energy_storage.pdf`.

[45] IEEE Committee Report, "Dynamic models for steam and hydro turbines in power system studies," *IEEE Transactions on Power Apparatus and Systems*, no. 6, pp. 1904–1915, 1973.

[46] IEEE Working Group, "MW response of fossil fueled steam units," *IEEE Transactions on Power Apparatus and Systems*, no. 2, pp. 455–463, 1973.

[47] IEEE Working Group on Power Plant Response to Load Changes, "Mw response of fossil fueled steam units," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-92, no. 2, pp. 455–463, March 1973.

[48] M. Ilic, P. Skantze, C. N. Yu, L. Fink, and J. Cardell, "Power exchange for frequency control (PXFC)," in *IEEE PES Winter Meeting*. IEEE Power Engineering Society, 1999, pp. 809–819.

[49] E. D. Ingersoll, J. A. Abhorn, and S. M. Chomyszak, "Compressor and/or expander device," US Patent 8 096 117, January 17, 2012.

[50] T. Inoue and H. Amano, "A thermal power plant model for dynamic simulation of load frequency control," in *Power Systems Conference and Exposition*. IEEE Power Engineering Society, October 2006, pp. 1442–1447.

[51] N. Jaleeli, D. N. Ewart, L. H. Fink, L. S. VanSlyck, and A. G. Hoffman, "Understanding automatic generation control," *IEEE Transactions on Power Systems*, vol. 7, no. 3, pp. 1106–1122, Aug. 1992, a report of the AGC Task Force of the IEEE/PES/PSE/System Control Subcommittee.

[52] G. M. Jenkins and D. G. Watts, *Spectral analysis and its applications*. San Francisco: Holden-Day, 1968.

[53] J. G. Kassakian and R. S. et al., "The future of the electric grid," Massachusetts Institute of Technology, Tech. Rep. ISBN 978-0-9828008-6-7, 2011, http://web.mit.edu/mitei/research/studies/the-electric-grid-2011.shtml.

[54] B. Kirby, M. Milligan, and E. Ela, "Providing minute-to-minute regulation from wind plants," in *International Workshop in Large-Scale Integration of Wind Power into Power Systems and Transmission Networks for Offshore Wind Power Plant*, Quebec, Canada, October 2010.

[55] B. Kirby and E. Hirst, "Generator response to intrahour load fluctuations," *IEEE Transactions on Power Systems*, vol. 13, no. 4, pp. 1373–1378, November 1998.

[56] B. J. Kirby, "Frequency regulation basics and trends," Oak Ridge National Laboratory, Oak Ridge, TN, Tech. Rep. ORNL/TM-2004/291, Dec. 2004.

[57] D. Kottick, M. Blau, and D. Edelstein, "Battery energy storage for frequency regulation in an island power system," *IEEE Transactions on Energy Conversion*, vol. 8, no. 3, pp. 455–459, September 1993.

[58] P. Kundur, *Power System Stability and Control*, N. J. Balu and M. G. Lauby, Eds. McGraw-Hill, 1994.

[59] H.-J. Künisch, K. G. Krämer, and H. Dominik, "Battery energy storage: Another option for load-frequency-control and instantaneous reserve," *IEEE Transactions on Energy Conversion*, vol. EC-1, no. 3, pp. 41–46, September 1986.

[60] J. Kure-Jensen, "Cost of providing ancillary services from power plants," EPRI, Palo Alto, CA, Tech. Rep. TR-107270-V2 4161, April 1997.

[61] K. D. Le, R. R. Jackups, J. Feinstein, H. H. Thompson, H. M. Wolf, E. C. Stein, A. D. Gorski, and J. S. Griffith, "Operational aspects of generation cycling," *IEEE Transactions on Power Systems*, vol. 5, no. 4, pp. 1194–1203, November 1990.

[62] S. Lefton, J. Edmonds, J. Foulds, and J. Montrose, "Effects of flexible operation on turbines and generators," Electric Power Research Institute, Palo Alto, CA, Tech. Rep. 1008351, December 2004.

[63] S. A. Lefton, P. M. Besuner, and G. P. Grimsrud, "Managing utility power plant assets to economically optimize power plant cycling costs, life, and reliability," in *Fossil Plant Cycling Conference*, 1994, pp. 195–208.

[64] ——, "The real cost of cycling powerplants: What you don't know will hurt you," *Power Magazine*, vol. 146, no. 8, pp. 29–34, Nov./Dec. 2002.

[65] O. Leitermann and J. L. Kirtley, "Energy storage for use in load frequency control," in *Proc. IEEE Conf. Innovative Technologies for an Efficient and Reliable Electricity Supply (CITRES)*, 2010, pp. 292–296.

[66] S. Lemofouet and A. Rufer, "A hybrid energy storage system based on compressed air and supercapacitors with maximum efficiency point tracking (MEPT)," *IEEE Transactions on Industrial Electronics*, vol. 53, no. 4, pp. 1105–1115, August 2006.

[67] K. H. Lundberg, "Notes on feedback systems," MIT 6.302 Course Notes Fall 2004, version 3.1.

[68] Y. V. Makarov, J. Ma, S. Lu, and T. B. Nguyen, "Assessing the value of regulation resources based on their time response characteristics," Pacific Northwest National Laboratory, Richland, WA, Tech. Rep. PNNL-17632, Jun 2008.

[69] Y. V. Makarov, C. Loutan, J. Ma, and P. de Mello, "Operational impacts of wind generation on california power systems," *IEEE Transactions on Power Systems*, vol. 24, no. 2, pp. 1039–1050, May 2009.

[70] J. Marple, S. Lawrence, *Digital Spectral Analysis with Applications*, ser. Prentice-Hall Signal Processing Series, A. V. Oppenheim, Ed. Englewood Cliffs, NJ: PTR Prentice Hall, Inc., 1987.

[71] T. R. Matthew L. Laszarewicz, "Grid-scale frequency regulation using flywheels," 2010, `http://www.beaconpower.com/files/ISO-NE-performance-paper-2010.pdf`.

[72] N. W. Miller and K. Clark, "Advanced controls enable wind plants to provide ancillary services," in *IEEE Power and Energy Society General Meeting*, 2010.

[73] N. W. Miller, R. S. Zrebiec, R. W. Delmerico, and G. Hunt, "A VRLA battery energy storage system for metlakatla, alaska," in *Eleventh Annual Battery Conference on Applications and Advances*. IEEE, January 1996, pp. 241–248.

[74] NERC Control Criteria Task Force, "Control performance standard and disturbance control standard frequently asked questions," `http://www.nerc.com/docs/oc/rs/cpsfaq.pdf`, November 1996.

[75] I. Ngamroo, Y. Mitani, and K. Tsuji, "Application of SMES coordinated with solid-state phase shifter to load frequency control," *IEEE Transactions on Applied Superconductivity*, vol. 9, no. 2, pp. 322–325, June 1999.

[76] *Real Power Balancing Control Performance*, North American Electric Reliability Corporation Std. BAL-001-0.1a, Oct 2008. [Online]. Available: `http://www.nerc.com/files/BAL-001-0_1a.pdf`

[77] *Frequency Response and Bias*, North American Electric Reliability Council Std. BAL-003-0a, October 2007. [Online]. Available: `http://www.nerc.com/files/BAL-003-0a.pdf`

[78] *Operating Reserves*, North American Electric Reliability Council (NERC) Std. BAL-STD-002-0, June 2007. [Online]. Available: `http://www.nerc.com/files/BAL-STD-002-0.pdf`

[79] C. D. Parker, "Lead-acid battery energy-storage systems for electricty supply networks," *Journal of Power Sources*, vol. 100, pp. 18–28, 2001.

[80] T. W. Parks and C. S. Burrus, *Digital Filter Design*, ser. Wiley Interscience. John Wiley and Sons, 1987.

[81] B. Parsons, M. Milligan, B. Zavadil, D. Brooks, B. Kirby, K. Dragoon, and J. Caldwell, "Grid impacts of wind power: A summary of recent studies in the United States," *Wind Energy*, vol. 7, pp. 87–108, 2004.

[82] PJM Interconnection, L.L.C., "FERC order no. 755 compliance filing," Norristown, PA, Tech. Rep. ER12, March 2012, `http://elibrary.ferc.gov/IDMWS/common/opennat.asp?fileID=12908801`.

[83] P. Poonpun and W. T. Jewell, "Analysis of the cost per kilowatt hour to store electricity," *IEEE Transactions on Energy Conversion*, vol. 23, no. 2, pp. 529–534, June 2008.

[84] D. C. H. Prowse, "Improvements to a standard automatic generation control filter algorithm," *IEEE Transactions on Power Systems*, vol. 8, no. 3, pp. 1204–1210, Aug. 1993.

[85] D. C. H. Prowse, P. Koskela, T. A. Grove, and L. R. Larson, "Experience with joint agc regulation," *IEEE Transactions on Power Systems*, vol. 9, no. 4, pp. 1974–1979, November 1994.

[86] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*, 2nd ed., ser. Springer Texts in Statistics. Springer, 2004.

[87] B. Roberts and J. McDowall, "Commercial successes in power storage," *IEEE Power and Energy Magazine*, pp. 24–30, March/April 2005.

[88] C. W. Ross, "Error adaptive control computer for interconnected power systems," *IEEE Transactions on Power Apparatus and Systems*, vol. 85, no. 7, pp. 742–749, July 1966.

[89] W. I. Rowen, "Simplified mathematical representations of heavy-duty gas turbines," *Journal of Engineering for Power, Transactions of the ASME*, vol. 105, pp. 865–869, October 1983.

[90] T. Sasaki, T. Kadoya, and K. Enomoto, "Study on load frequency control using redox flow batteries," *IEEE Transactions on Power Systems*, vol. 19, no. 1, pp. 660–667, February 2004.

[91] R. Saupe, "The power conditioning system for the +/- 8, 5/17 mw - energy storage plant of BEWAG," in *Third International Conference on Power Electronics and Variable Speed Drives*. London, UK: IEEE, July 1988, pp. 218–220.

[92] S. M. Schoenung, "Energy storage systems cost update," Sandia National Laboratories, Albuquerque, NM, Tech. Rep. SAND2011-2730, April 2011.

[93] F. C. Schweppe, R. D. Tabors, J. L. Kirtley, H. R. Outhred, F. F. H. Pickel, and A. J. Cox, "Homeostatic utility control," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-99, no. 3, pp. 1151 – 1163, 1980.

[94] H. Shayeghi, H. A. Shayanfar, and A. Jalili, "Load frequency control strategies: A state-of-the-art survey for the researcher," *Energy Conversion and Management*, vol. 50, no. 2, pp. 344–353, February 2009.

[95] M. R. I. Sheikh, S. M. Muyeen, R. Takahashi, T. Murata, and J. Tamura, "Improvement of load frequency control with fuzzy gain scheduled superconducting magnetic energy storage unit," in *Proceedings of the International Conference on Electrical Machines*, no. Paper ID 1026, 2008.

[96] A. Shibli, J. Gostling, and F. Starr, "Damage to power plants due to cycling," Electric Power Research Institute, Palo Alto, CA, Tech. Rep. 1001507, July 2001.

[97] R. R. Shoults, R. Kelm, D. Maratukulam, and M. Yao, "Improved system AGC performance with arc furnace steel mill loads," *IEEE Transactions on Power Systems*, vol. 13, no. 2, pp. 630–635, May 1998.

[98] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. Pearson Education, Inc., 1991.

[99] P. Sørensen, N. A. Cutululis, A. Vigueras-Rodriguez, L. E. Jensen, J. Hjerrild, M. H. Donovan, and H. Madsen, "Power fluctuations from large wind farms," *IEEE Transactions on Power Systems*, vol. 22, no. 3, pp. 958–965, Aug. 2007.

[100] P. Spicer and P. Galow, "Determine the effect of providing regulation and frequency response service on the efficiency of pulverized coal boilers," Electric Power Research Institute, Palo Alto, CA, Tech. Rep. 1000744, September 2000.

[101] J. T. Stauth and S. R. Sanders, "Optimum biasing for parallel hybrid switching-linear regulators," *IEEE Transactions on Power Electronics*, vol. 22, no. 5, pp. 1978–1985, September 2007.

[102] P. Stoica and R. Moses, *Spectral Analysis of Signals*. Pearson Prentice Hall, 2005.

[103] G. Strang, *Computational Science and Engineering*. Wellesley Cambridge Press, 2007, ch. 2.4 Graph Models and Kirchhoff's Laws, pp. 142–155.

[104] P. A. Taylor, "Update on the Puerto Rico Electric Power Authority's spinning reserve battery system," in *11th Annual Battery Conference on Applications and Advances*, 1996.

[105] U.S. Energy Information Administration, "Electric power monthly," U.S. Department of Energy, Washington, D.C., Tech. Rep., February 2012, available at `http://www.eia.gov/cneaf/electricity/epm/epm_sum.html`.

[106] C. Vartanian, "Grid stability battery systems for renewable energy success," in *IEEE Energy Conversion Conference and Expo*, September 2010, pp. 132–135.

[107] R. Wagner, "Large lead/acid batteries for frequency regulation, load levelling and solar power applications," *Journal of Power Sources*, vol. 67, pp. 163–172, 1997.

[108] R. A. Walling, L. A. Freeman, and W. P. Lasher, "Regulation requirements with high wind generation penetration in the ERCOT market," in *Power Systems Conference and Expo, PSCE09*, Seattle, WA, 15-18 March 2009.

[109] C. Wang and S. M. Shahidehpour, "Optimal generation scheduling with ramping costs," *IEEE Transactions on Power Systems*, vol. 10, no. 1, pp. 60–67, February 1995.

[110] W. D. Wilder and H. J. Thielke, "Effect of swinging loads on steam plant economy," in *AIEE Winter General Meeting*, New York, NY, January 1953.

[111] Working Group on Prime Mover and Energy Supply Models for System Dynamic Performance Studies, "Dynamic models for fossil fueled steam units in power system studies," *IEEE Transactions on Power Systems*, vol. 6, no. 2, pp. 753–761, 1991.

[112] ——, "Hydraulic turbine and turbine control models for system dynamic studies," *IEEE Transactions on Power Systems*, vol. 7, no. 1, pp. 167–179, February 1992.

[113] ——, "Dynamic models for combined cycle plants in power system studies," *IEEE Transactions on Power Systems*, vol. 9, no. 3, pp. 1698–1708, August 1994.

[114] Y. Zhang and A. Bose, "Design of wide-area damping controllers for interarea oscillations," *IEEE Transactions on Power Systems*, vol. 23, no. 3, pp. 1136–1143, Aug 2008.

[115] D. Zwillinger, Ed., *Standard Mathematical Tables and Formulae*, 31st ed. Chapman and Hall/CRC, 2003.