December 2021

Real-Time Battery Energy Storage Scheduling for an Educational Building

Armien Harrell
Clemson University, armienkh@gmail.com

REAL-TIME BATTERY ENERGY STORAGE SCHEDULING FOR AN EDUCATIONAL BUILDING

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Electrical Engineering

---

by
Armien K. Harrell
December 2021

Accepted by:
Dr. Ramtin Hadidi, Committee Chair
Dr. Edward Collins
Dr. Zheyu Zhang

Abstract

Special electricity rates are offered by some utilities to incentivize certain energy consumption behaviors in an attempt to garner a cost effective demand profile to service. One area of concern for a utility is the maximum demand of a building during the times of day when electricity usage is typically at its peak. Regulated utilities are required to have the capacity to meet the electrical demand of customers, though, the peak demand of customers can oftentimes be relatively brief. Therefore, it is not cost effective for a utility to have generation units online to only cover momentary peak demands. Electricity provider Dominion Energy offers a time-of-use demand and energy rate structure, Rate 21A, that is available to buildings such as the Zucker Graduate Education Center (ZGEC). The ZGEC is a two-story educational building located on the Clemson University Restoration Institute (CURI) campus in Charleston, SC. The purpose of this project is to implement a battery energy storage system (BESS) scheduler for demand-side management of the ZGEC which has attached to it a 50 kW, 96 kWh battery. There are three components of the demand management algorithm that are investigated for the real-time implementation:

1. Demand Data Collection
2. Building Load Forecaster
3. BESS Scheduler

Demand data collection is done using a database management system developed in MATLAB 2019b capable of reading power meters that communicate using Modbus. By design, only three types of power meters are compatible: PowerLogic Series 800 (PM800), PSL PQube 3, EIG Shark 100. Load forecasting and BESS scheduling are done by a Simulink model, developed in MATLAB 2020b, that operates in Real-Time Normal Mode. A time series forecasting model provides 24 hours of ZGEC demand predictions to a BESS scheduler which uses Model Predictive Control (MPC). Currently, the ZGEC is subjected to a standard energy charge rate structure, Rate 9. Cost comparisons are provided for switching to Rate 21A as well as for implementing the developed forecaster and BESS scheduler.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Demand for electricity which must be met by power plants varies significantly dependent upon the time of day as well as time of year. Electricity's generation and consumption must be balanced to ensure a reliable power grid and, since electricity is not easily stored, grid operators are tasked with performing this balancing act by scheduling generation units for operation based on estimated demand [1]. Too little available generation capacity would result in loss of electricity for many consumers, faults throughout the power grid, and fines as well as other punishments from organizations such as NERC and FERC. In contrast, faults on the power grid will also arise if generation overtakes load. The goal of utilities is to have just enough generation online to always satisfy load and be cost effective.

In order to match the continuously increasing or decreasing demand of electricity, operators must make proper use of power plants such as baseload plants and peakers. Baseload plants are the foundation of electricity generation and account for much of the energy supplied. These plants operate continuously for long periods of time at lower costs relative to peakers, though, their initial cost to build is higher. Peakers are utilized during times when demand grows beyond what baseload plants are able to supply alone. While their initial build costs are low relative to baseload plants, their operation costs are also more expensive [1]. Reliance on peakers tend to rise during certain times of a day, such as on-peak hours, when demand usually reaches its highest point. These on-peak hours shift over the course of a year, and spikes in demand during these periods have a critical impact on the cost of generating electricity.

At least 10% of electricity generation costs are due to meeting high levels of demand which occur less than 1% of the time. As a way to improve the reliability of the grid, the Energy Policy Act was passed in part to encourage electricity consumers to alter their demand profile [1]. The act promotes the offering of time-based pricing from utilities, deployment of technologies, and other cost saving incentives to support consumers engaging in demand response activities. Demand response,

also known as load management or demand-side management, can be initiated by the consumer through different means. The scope of approaches can differ for the types of consumers. In the case of commercial buildings, demand response could entail shifting electrical loads and deploying building automation systems (BASs) to control such things as lighting and HVAC.

It is stated in [2] that accurate sensing and effective controls could eliminate up to 30% of energy consumption according to studies. Cost savings can be maximized through making use of both demand reduction offered through BASs and demand shifting. This thesis focuses on shifting the demand of the Zucker Graduate Education Center (ZGEC) via an installed BESS rated at 50 kW (96 kWh). The sizing of the battery was not considered as a parameter to be optimized within this project since the battery was installed prior to the work detailed in this thesis. Implementing load management with the given battery relies on three mechanisms: demand data collection, building load forecaster, BESS scheduler.

1.1 Demand Data Collection

In order to implement sophisticated demand response methods, advanced meters are needed to measure relevant physical characteristics to be recorded at suitable intervals. Recording large amounts of data requires proper storage and handling to yield any functional information. Storage is done through use of a database whose design is described in Chapter 2. The handling of the data to yield information is provided by a database management system (DBMS) described in Chapter 3 along with a description of the power meters which interface with the system. Only three power meters are installed or expected to be installed in the ZGEC: PowerLogic Series 800 (PM800), PSL PQube 3, EIG Shark 100. While all of the meters do not share the same transport protocols, they each use MODBUS for their application protocol.

1.2 Building Load Forecaster

The designed forecaster is a time series model performing real-time very short-term load forecasting. It operates by predicting the demand for the upcoming period followed by the next until the range of interest is completely forecasted. The range of interest may be two days or less depending on how much time remains for model operation. This particular model takes only

historical data supplied from the database as its input. Further descriptions and comparisons to other types of models are provided in Chapter 4.

1.3 BESS Scheduler

A number of works consider the perspective of the utility when discussing the use of a BESS; renewable generation typically accompanies the storage devices discussed. In some papers the utility owns both the BESS and the renewables. In others, it is the customer who owns the technology. A common element throughout, however, is that cost to the customer is not usually the main priority but instead the focus is typically centered on coordinating renewable generation and BESS in an optimal manner which avoids causing issues on the grid. There are also few papers detailing the results of real-time application of their proposed methods, instead, conclusions are drawn from simulations alone.

Authors in [3] discuss simulated deployment of a BESS owned by a utility for the purposes of voltage management as well as managing power flow on the grid. The objective function specified focuses on minimizing the total cost of the installed BESS, and the constraints considered are the voltage requirement of the network and operation condition of the BESS. In [4], the authors focus on demand response for commercial/industrial buildings with priority given to customer savings while also considering installed BESS and distributed generation. Provided by the authors are mathematical formulations meant to help a customer determine the appropriate demand response program and integrate a BESS along with distributed generation.

Within [5], residential owned BESS and PV system are the focus, and the authors discuss the application of Model Predictive Control (MPC) for energy management of such systems. A cost minimizing objective function is provided in which power from the grid is weighted such that generated PV power is utilized as much as possible to reduce costs and battery degradation. It is explained in [5] that in Li-Ion batteries cycling ageing is accelerated through long dwell times at a high state-of-charge (SOC) and also excessive BESS cycling, so a high SOC is penalized in the objective function. Power from the grid is weighted to reduce interaction, but there is no consideration of maximizing time-of-use rates. Also, the authors of [5] are prioritizing the prevention of grid congestion when deploying the BESS. Scheduling for the BESS took in forecasts for PV generation and load demand.

In [6], peak load shaving using a BESS is discussed. The purpose of this shaving is to minimize the difference between peak and valley loads as well as minimize the daily load variance. While minimizing the peak demand will inherently reduce demand costs, the charging/discharging model discussed by the authors does not acknowledge any time-of-use rates, so scheduling is not necessarily optimized to reduce the consumers electricity bill relative to a specific rate structure. A rolling load forecasting technique is implemented that relies on historical data and produces regression parameters. Results from this method of forecasting and scheduling are generated through simulations.

In this thesis, MPC is used for real-time BESS scheduling with the main objective being to reduce the electricity cost of an educational building based upon time-of-use energy and demand rates provided by Rate 21A from Dominion Energy. The scheduler takes as input the demand profile provided by the developed time series forecasting model and is discussed thoroughly in Chapter 5. The forecaster itself also only uses a singular input of relevant historical data. Data taken from the designated power meters are timestamped and stored in the developed database which is accessed by its unique DBMS. Results of the implementation are discussed in Chapter 6. The contributions of this project are:

- Custom database and DBMS designed to handle the retrieval, allocation, display, and export of high resolution data.
- An adaptive very short term time series forecasting model reliant only on historical data. The forecasting model is designed specifically for the application of MPC for BESS scheduling at intervals of 5 or 15 minutes.
- The real-time deployment of a BESS for the purposes of demand-side management for an educational building.

Chapter 2

Database Design

Advancement is driven by the ability to better understand sets of discrete occurrences or data. The modernization of power systems relies upon the prevalence of sophisticated measurement technologies which are capable of supplying large volumes of data at a high frequency. These measurements are the foundation of a number of special interest areas such as complex control algorithms, automation techniques, and real-time response applications. However, in order to utilize the data, it must be properly managed and given a contextual basis, converting it to information. It is information that is provided by a database.

2.1 Traditional Database

A database is defined in [7] as a collection of files that have some relation to each other. The files are generally separated for the sake of grouping data items of the same unit or category. The relationship between files is usually in the form of one file contains the complete list of available power meters, a second file contains the list of installed power meters, and a third file contains what each power meter is capable of measuring. Gathering these separate yet related files into a database allows for more efficient organization, retrieval, and modification of information.

Data organization is the first fundamental component of a database as it dictates how all of the stored data is perceived and the overall database efficiency which also pertains to the amount of redundancy present. From a simplistic viewpoint, devoid of models or functionality, the organization of a database can be broken down to basic structure components as seen in Figure 2.1: file, record, field, data item. A file has already been described in the context of a database, but the information that is held within a file is grouped into what are known as records.

Records within a file all pertain to a specific function. For example, a power meter performs measurements which can be grouped based on their relevance to one another. Three records that could be contained within a file are Power, Voltage and Current. Within these records are fields

which are the lowest level of information as they are just above the data. The record Power would contain fields such as Phase A Real Power, Phase B Real Power, etc.

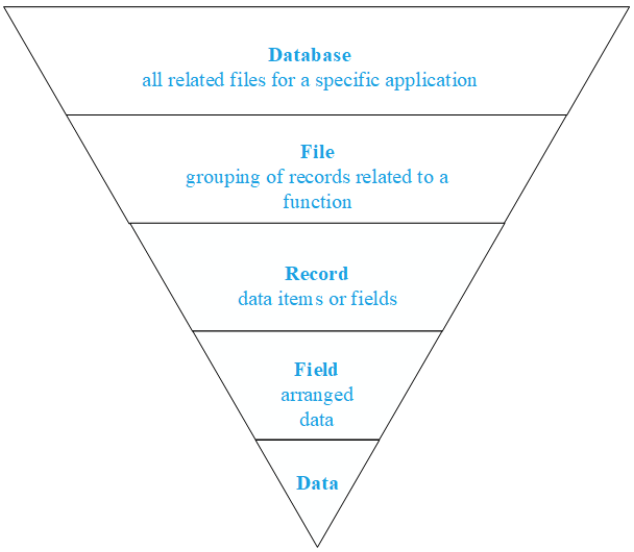

Figure 2.1. Data Organization Hierarchy

Each record usually contains fields that are of the same data unit, so all data within a record no matter its field share the same unit. Therefore, within the record that is Power, all of the phase power fields will be in units of kilowatts. In accordance with the methodology set forth, a description of the unit for each data item would be available in either another file or record.

2.2 Database Models

Database models are oftentimes abstractions that provide a description of the methodology for data processing. Data processing is discussed in greater detail in Section 2.5, but it can be succinctly defined as the act of handling data to convert it into useful information [7]. The data organization hierarchy is useful in conceptualizing the series of operations necessary to perform actions within a given database model. Three common database models are:

- Network
- Object-Oriented
- Hierarchical

A Network Database model is, as its name suggests, similar to the concept of a simple computer network in the sense that a local area network (LAN) could be connected to multiple other LANs via bridges. Considering the network model in terms of the data organization hierarchy pyramid, more than one record may contain the same field depending upon the application for which a database is used. In such instances, so as to reduce the amount of redundant data being stored, a network model will allow for each record to point to the same field instead of establishing multiple copies of that field.

Some power meters do offer redundant registers such that the same data is stored at multiple addresses. Within a Network Database model, each of the redundant registers would be acknowledged by their unique address or name, but only one copy of the data is stored within the database. An example, as depicted in Figure 2.2, would be if within the Power record there existed two fields which represented the same single phase real power. Only a single data item is utilized, however, both fields are referenced in the database.

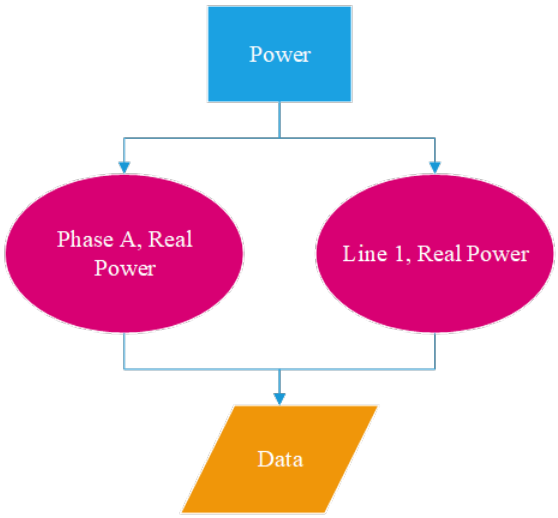

Figure 2.2. Network Database Model Example

The Object-Oriented Database, also known as the Object Database Management System (ODBMS), model is unique amongst the three listed database models. The other two models fall under the category of relational database or Relational Database Management System (RDBMS) [7]. A relational database stores data by utilizing two dimensional tables that are normalized to reduce repetition of data. The columns, or rows, of the table depend on a unique identifier for selection.

Once the specific column is chosen, some action may take place on one or more rows of that column. In an ODBMS, the database software is based upon object oriented languages such as C++.

This type of database is only useful when the data or the data relationships are complex. This complexity could appear in cases where there are multiple tables of data pointing to a multitude of other tables, or a simple rule of thumb would be if using the standard relational methods for handling information is too arduous due to its complexity then the ODBMS is the optimal choice.

However, in instances when the data is simple, the ODBMS is the inefficient option [7]. A deciding factor when choosing between an ODBMS and a RDBMS is the type of database system chosen. Database systems are discussed in further detail in Section 2.4. The last database model, Hierarchical, is the simplest as it operates on the logic of parent-child processes in computer languages: a child may only have one parent, but a parent may have many children as depicted in Figure 2.3.

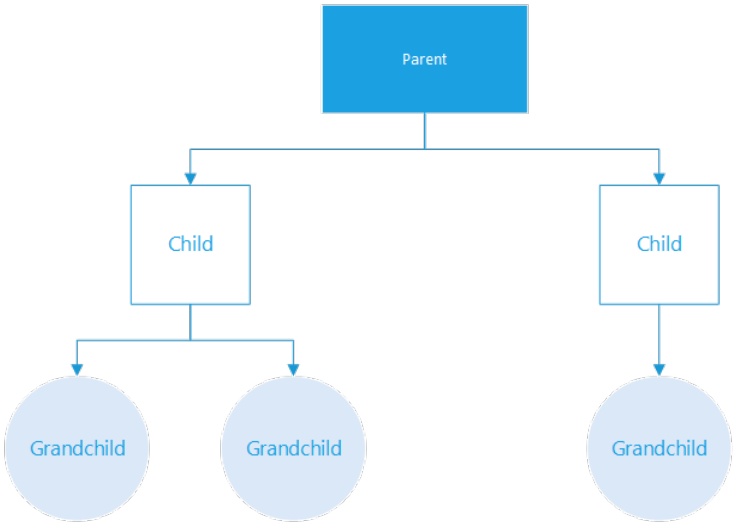

Figure 2.3. Hierarchical Database Model

In juxtaposition to the Object-Oriented Database model, the Hierarchical Database model is most efficient when the data and their relations are simple [7]. Unlike the Network model, the Hierarchy model does not handle the presence of redundant data as effectively, so it is preferred in scenarios involving definite systematic structures of one-to-many relationships instead of many-to-many relationships.

While the level of complexity is often application dependent, the data organization can either

simplify or further complicate a database structure. Continuing with the power meter example, the overall model is intuitively simpler if all power measurements from a meter are contained inside a single record than if those measurements were instead held inside of various records.

2.3 Types of Databases

The modeling of data is an important feature to consider of a database, but one must also determine the function of the database. Database types are generally categorized as either Analytic or Operational [7].

Analytic databases are also known as On Line Analytical Processing (OLAP) and are largely read-only databases meant for analysis applications of historical data. Additional data may be added but such occurrences are expected to be few and far in between such that the database can be considered static. Operational databases, also known as On Line Transaction Processing (OLTP), are the opposite as it is expected to be dynamic. Applications that involve tracking real-time information are best served by an operational database which allows for frequent modification of data.

2.4 Database Systems

A database system is essentially the method by which the database itself is stored. The database can either be distributed or centralized dependent upon the number of expected users of said database as well as the relevance of its information to all groups of users.

With a centralized database, all data is stored in a single physical location such as on a server, though, users in remote locations may access the data over network [7]. Relative to the other system, the centralized database is simple and easy to maintain. Issues may arise if multiple users require access simultaneously; in such scenarios the throughput of data to and from the database lessens.

A distributed database remedies certain concerns associated with the centralized system according to [7]. First, the database is more secure as it is stored on multiple devices within a network, so should one of the physical devices be damaged the entire database is not lost. There is also more security with respect to data access as the database can be partitioned to allow users

access only to data relevant to their work. With this feature the occurrence of a bottleneck becomes less likely as well. However, with these added features also comes notable disadvantages relative to the centralized system.

A distributed system requires far more components, increasing initial and maintenance costs, and the overall design can become complicated. Some of the complications are due to data integrity. With multiple locations storing the database, any modification of data at one location must be reflected at all locations capable of accessing that data. This leads to deciding between central and distributed control schemes to best address the issue.

2.5 Traditional Database Management System

A database management system (DBMS), or sometimes just called database manager, is the specialized software programs running on a database that allow users to interact with the data. This interaction is data processing, which can be split into three main operations: collection, conversion, manipulation [7].

The collection of data precedes all other actions as it provides the basis of their function. The DBMS is responsible for properly storing the data that it receives, so incoming data is converted to a form that streamlines processing; the form the data takes is dictated by the database model. The data is then ready to be manipulated to provide information via sorting, calculating, summarizing, etc. Overall, the DBMS is expected to perform a number of functions that are listed in [7] as:

- Data Dictionary Management
- Data Storage Management
- Security Management
- Multi-User Access Control
- Backup and Recovery Management
- Data Integrity Management
- Database Access Languages and Application Interface
- Database Communication Interface

Some of the functions are independent of user interactions so their implementations are allowed to be set solely by the preferences of the database architect, though, other functions are to be implemented based upon effectively interfacing with a user. The interaction between a user and the DBMS is generally that of client and server. An example of this relationship is depicted in the two-tier architecture for a centralized database shown in Figure 2.4.

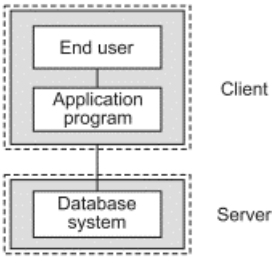

Figure 2.4. Two-tier Architecture [7]

DBMS software runs on the database, which acts as the server, and communication is initiated by the user through an interface that resides on the client side. As with any communication between separate entities, there needs to exist an agreed upon protocol by which commands and data are transferred and interpreted. Since there may exist a multitude of users with varying operating systems and application programs, it is simplest for the standardization to take place on the database itself. One of the major components of any DBMS, along with the software and hardware, is the database access language. The most commonly used language amongst notable DBMS programs such as MySQL, Microsoft Access, Oracle, etc. is Structured Query Language (SQL). With SQL as the standard, users need only ensure that the application program they use incorporates the language [7].

Though SQL is a computer language, it is easy to understand relative to other languages as it is similar to English. Users send SQL commands such as SELECT, FROM, WHERE, etc. for manipulating data and objects. Since SQL is a query language it only requires that users specify the action to take place and not how that action is to be implemented. Despite SQL's lack of difficulty to understand, some training is necessary to fully grasp the proper syntax of the language and to use it efficiently.

2.6 Developed Database

The design of a database is dependent upon its application. For the purposes of the project outlined by this thesis, the developed database is at minimum required to:

- Collect and Store Timeseries Data
- Provide Access to a Small Pool of Users
- Organize Data from Multiple Power Meters
- Export Information

Standard database management systems are not specifically tailored to collecting and storing timeseries data, however, there are timeseries databases built for such tasks. Timeseries data is simply data that is timestamped whose size is expected to grow with the passing of time. Standard applications of databases typically assign a single data item to a field. In a timeseries database, multiple data items are expected to be stored in a field and each of those data items correlate to an individual timestamp. Examples of commercial timeseries databases include InfluxDB, kdb+, and Prometheus.

Other than being customized to more effectively handle timeseries data, the commercial timeseries databases are just as standardized as any of the other. SQL is still the most prominent language used, though, there are notable instances of a commercial database using its own query language. As noted in the previous section, SQL does require some training to make use of it effectively. When considering the list of minimum requirements of the database for this project, only a small group of users, less than five, will need access to the database with no anticipation that concurrent use will be needed.

Since the user pool is not expansive, a standardized design approach is not necessary, and the overall architecture can be personalized to the needs of those few users. The most fundamental need is storing an assortment of data from various power meters on a one minute interval. Due to the sheer volume of data and the high frequency at which it must be stored, automating this process is essential. The last requirement is that information, in the form of sorted or summarized timestamped data, can be provided from the database.

Per the requirements, the developed database is OLTP since it is dynamic. As for modeling the database, a hierarchical approach was taken when considering the simple relationships provided by the data organization as depicted in Figure 2.5. The files of the database include the meter being read from, the register map of the meter, and recovery files. Records are considered to be the names of registers being continuously read from, group names of registers, and object attributes. The fields level is made up of timestamp and register description groups. Finally, the last level contains data items whose type is either datetime, numerical or string.

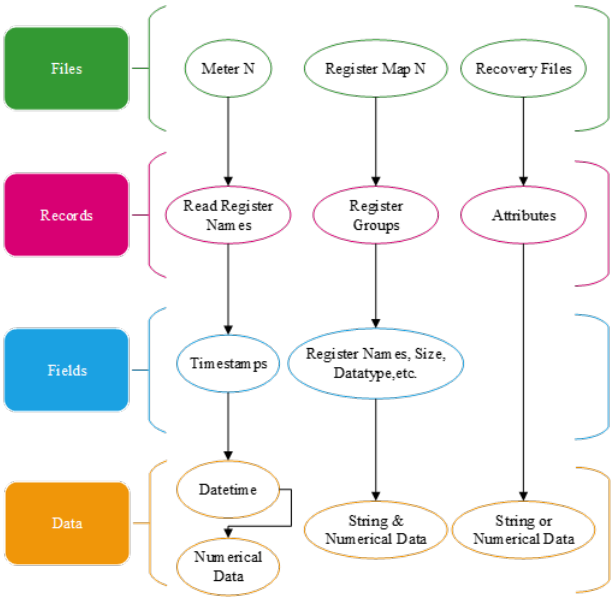

Figure 2.5. Hierarchy of Developed Database

A byproduct of having a small user group is implementing the database system to be centralized. Users are allowed to connect remotely, though not concurrently, to the database via a custom three-tier client-server architecture depicted by Figure 2.6. In this architecture, an intermediary application server is placed in between the client and the database. This application server retains the procedures for accessing the data from the database, removing the need for any of the users to learn a query language. The lack of a query language does limit the flexibility of the commands that can be given, but another byproduct of a small user group is being able to pre-program the expected necessary commands. Due to the combined requirements of the project, the architecture implemented operates with the DBMS on the application server.

While this adaptation of a DBMS is notably different from standard implementations, it

still performs many of the necessary functions that were listed in Section 2.5. Data Dictionary Management takes note of the files in the database and provides descriptions of the data in those files; this is handled with directory scans and the use of register maps for the power meters being read. Data Storage Management is done autonomously as the DBMS collects data from the power meters every minute then stores the data according to the database's defined data structure.

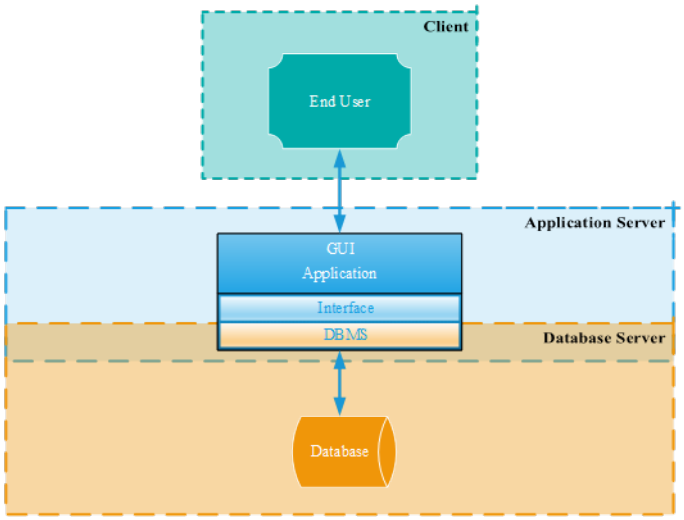

Figure 2.6. Custom Three-Tier Architecture

Data Transformation & Presentation is the manipulation of the stored data such that it provides relevant information based upon user commands. These commands are delivered via the graphical user interface (GUI) selections and then the resulting information is also presented by the GUI. Security Management is achieved through the application server, which verifies the credentials of a user before allowing access to the database. Backup & Recovery Management is also helped by the presence of the application server as it provides a physical device separate from the centralized database to store recovery files. Multi-user Access Control and Data Integrity Management are the only functions which are not wholly applied. As stated previously, concurrent use is not a priority for this application, and data integrity is less of a concern since data will not be compromised due to multiple users accessing the database simultaneously. Data integrity methods are still applied with regard to the forecasting model, described in Chapter 4, which is capable of accessing the database concurrent with the DBMS.

2.7 Summary

In summary, the developed database is centralized and functions as an OLTP with a hierarchical data model. In place of a database access language, a GUI on the intermediary application server handles all communication between the user and the database. Characteristics of the developed database are provided in Table 2.1. The purpose of this database is to store timestamped data that is autonomously retrieved from an assortment of power meters on a one minute interval continuously to then be used primarily for demand response studies. To best meet the needs of this project a three-tier client/server architecture was established in which the DBMS resides physically on the application server instead of the database.

Developed Database	
Type	Operational (OLTP)
Model	Hierarchical
System	Centralized
Access	GUI (via Application Server)
Functions	▪ Data Dictionary Management ▪ Data Storage Management ▪ Data Transformation & Presentation ▪ Security Management ▪ Backup & Recovery Management ▪ Application Interface ▪ Database Communication Interface

Table 2.1. Characteristics of the Developed Database

Chapter 3

Application Server

The application server is home to the DBMS which communicates with the database server. Power meters are being queried continuously by the DBMS on 1-minute intervals using MODBUS messaging protocol. Users are also allowed to directly query the database and power meters via the GUI that encompasses the DBMS.

3.1 MODBUS

As described in [8], MODBUS provides client/server communication between intelligent devices as an application layer messaging protocol. At the application layer, independent of lower level communication layers, a frame of information consists of a function code and data to form what is known as a protocol data unit (PDU). The function code is the basis of the request and reply nature of the protocol as it describes the action that the client wishes the server to perform or details an error occurrence. This field is one byte in size whose valid values range numerically from 1 to 255, with values 128 and above reserved for exception responses. The data field may or may not exist in the PDU depending upon the request, however, should a MODBUS related error occur at the server then an exception response will represent the data field.

Devices operating with MODBUS will address data as discrete inputs, coils, input registers and holding registers. Discrete inputs and coils are single bit addressable that are read-only and read-write respectively. Input registers and holding registers are 16-bit word addressable that are read-only and read-write respectively. Up to 65536 data items can be selected individually no matter which address method is employed.

Lower level communication utilizing the MODBUS protocol is implemented for serial transmission and TCP/IP over Ethernet. Additional fields are added to the standard PDU at these communication layers to create an application data unit (ADU), however, the amount of data that can be transferred is limited by the PDU no matter the transmission method. A MODBUS PDU is restricted to 253 bytes total and the most significant byte is sent first since this protocol uses a

big-Endian representation [8].

MODBUS over serial line builds its ADU by adding an address field at the beginning and an error checking field at the end. The header address field, one byte in size, supports referencing 256 different addresses from 0 to 255; valid individual server addressing requires a unique value in the range of 1 to 247. The error checking field can be based upon either cyclical redundancy checking (CRC) or longitudinal redundancy checking (LRC). If using CRC, the field is two bytes in size, but the field is one byte in size if using LRC. The error check method utilized depends which serial transmission mode is used for all devices on a serial line: RTU mode or ASCII mode.

RTU mode, as described in [9], transmits an ADU in continuous 8-bit increments represented as two 4-bit hexadecimal characters. This mode is preferred due to its high throughput compared to ASCII mode, which represents the 8 bits as two characters. ASCII mode is usually used when the physical link or a device does not allow for RTU mode. Due to their differing representations, the modes also differ in their error checking methods. RTU mode uses CRC and ASCII mode uses LRC.

The serial transmission method for MODBUS can use different physical interfaces of RS485 or RS232. Both options have two-wire configurations, but RS485 can also be configured as a four-wire interface. RS485 is the better option for transmitting signals over long distances at a high baud rate as it is less susceptible to noise. RS232 requires shorter distances of 20 meters if attempting to transmit a signal at a baud rate near 20 Kilo-bits per second with success. According to [9], 9600 bps and 19.2 Kbps are required to be baud rate options while other rates are optional. All rate options must meet minimum transmission and error requirements as detailed in [9].

MODBUS TCP/IP over Ethernet builds it ADU by only adding a MODBUS Application Protocol (MBAP) header that is 7 bytes in size. This dedicated header contains four fields: Transaction Identifier, Protocol Identifier, Length, Unit Identifier. The unit identifier field is to distinguish between servers if they exist on a serial line behind a gateway. According to [10], this is one of the main differences when comparing MODBUS TCP/IP with RTU; the unit identifier acts as the address field within an RTU ADU. Other differences include that for TCP/IP the recipient can verify a message is complete and, due to length information being included in the MBAP, the recipient can determine the boundaries of a message. There is also no need for the ADU to include a checksum such as CRC since the checksum is handled by the Ethernet frame which encapsulates the ADU. Within an Ethernet frame the ADU is the data field. An ADU comparison between RTU

and Ethernet is provided in Figure 3.1.

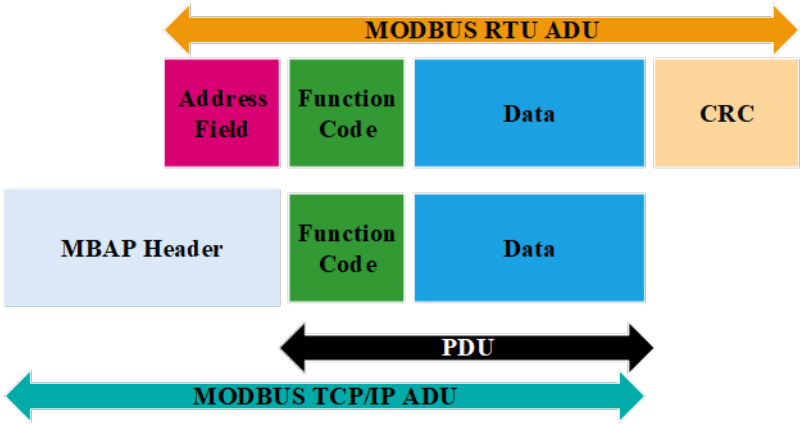

Figure 3.1. MODBUS Frames

3.2 Power Meters

By design, only three types of power meters are compatible with the application server program: PowerLogic Series 800 (PM800), PSL PQube 3, EIG Shark 100. All of the power meters used for this project implement some version of MODBUS for their messaging protocol. They each also store data in 16-bit wide registers and do not bother to distinguish between register types. All registers can effectively be considered as holding registers when addressing them, however, each register has its own read/write permissions.

The PM800 power meter is connected to the electrical control panel of the Zucker Graduate Education Center (ZGEC) to acquire real-time readings of phase current, voltage, power, etc. of the building from the perspective of the grid. Its data transmission is done serially with RS485 as the communication protocol. To improve network flexibility such that the power meter could be reached over a wide area network, a Lantronix UDS1100 [11] is used as an intermediary device between the serial lines and an Ethernet network.

It is important to note that this device is not utilizing the MODBUS TCP/IP protocol but is instead performing serial tunneling. Serial tunneling in this instance can be described as a MODBUS RTU ADU being wrapped inside of an Ethernet frame to be sent over a network to a receiving device [12]. The receiving device then unpacks the Ethernet frame to grab the ADU then

reads it as serial data. The UDS1100 handles the packing and unpacking for the PM800 and to avoid using two intermediate devices, the application server makes use of a virtual COM port. Creating this virtual COM port was done using software from Lantronix named the COM Port Redirector (CPR) manager [13]. The CPR manager will intercept any serial data directed towards the virtual COM port and perform the necessary actions whether the control application is sending or receiving.

The UDS1100 requires proper configuration before it can be used. Settings such as the serial protocol, baud rate, and parity must match with those of the PM800 and virtual COM port. Other notable fields to set are the IP address, connection type, and local port number. The IP address of the UDS1100 can be dynamic or set manually and remain static. Connection type may be either passive or active. If active, the device will attempt to create the connection with the application server, elsewise, with a passive connection, the device will wait for a connection to be made by the application server. The local port number is somewhat arbitrary and is by default set to be 10001. The IP address and port number of the UDS1100 device are static for the purposes of this project and a passive connection type was determined to be the most appropriate option.

Multiple versions of the PQube 3 power meter are compatible with the software on the application server including the base model, PQube 3r, PQube 3e and PQube 3v. The PQube 3 and PQube 3e are the versions expected to be deployed. These versions differ mainly in that the base model can monitor up to two three phase loads while 3e can handle four. This meter can communicate directly over Ethernet using MODBUS TCP/IP protocol, so no intermediary devices are needed. Before installing a PQube 3 meter, it must first be configured. The application server assumes all meters of this type have the same settings. Typically, the default settings are used and only the MODBUS settings must be verified. So, for each PQube to be called by the application server, the MODBUS server must be enabled, device address set to 1, TCP port set to 502, register start address set to 7000, and the byte order set to big endian.

In general, the TCP port for MODBUS applications will be set to 502 as it is reserved for the protocol and must always be a port option for TCP devices that utilize the protocol according to the standard outlined in [10]. The 7000 base is the default starting address for PQube and, unlike the PM800, the PQube does not automatically compensate for zero-based addressing. As mentioned previously in 3.1, up to 65536 registers can be addressed. The MODBUS protocol assumes the addressing ranges from 0 to 65535, however, it may be preferable for a device to offset that range. The PM800 internally has a register base address of 30,000 or 40,000 depending on configuration,

but the application server does not need the offset value since the PM800 accepts the zero-based address then adjusts it accordingly. The PQube 3 has no such feature, so the application server is required to know the base address in order to properly read the meter.

3.3 Application Server GUI

The main function of the database is to manage the demand data collection portion of the overall demand response algorithm, but the overall database was designed to incorporate various categories of data and supply users with relevant information.

3.3.1 Interface Overview

The interface contains four main tabs and two menus that provide an assortment of actions and information to the user. The main tabs are named Read, Graph, Download & ESIP. Within Read is another grouping of tabs named Read All, Snapshot & Custom Snapshot. The Read All tab is shown in Figure 3.2 and is the landing page of the interface upon initialization. Implementing autonomous interactions with power meters requires verification that communication is in fact proceeding without interruption and some control to remedy any issue likely to occur.

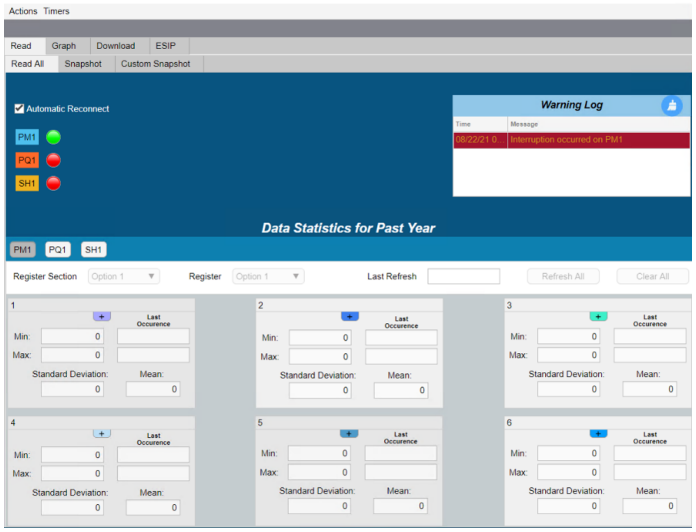

Figure 3.2. Read Tab: Read All

Seen at the top of the Read All tab is an Automatic Reconnect checkbox. When checked,

attempts to fix communication issues are done autonomously instead of requiring a user to manually address the problem; however, when the box is unchecked, such issues will require user intervention. It may be ideal for the box to be unchecked in instances when a connectivity dilemma is due to a physical cause that is impossible to fix programmatically. Whenever there is any issue related to the power meters, warning messages will appear in the log.

An entry in the warning log is added if the program notices anything undesirable as it relates to the active power meters and if the program is attempting to resolve a connection issue. Every entry of the warning log will contain the name of the specific meter for which the warning occurred or be in reference to all meters. Possible warnings include:

1. $**$#: Worker Queue Full
2. $**$#: Read Time Too Long
3. $**$#tmp Unable to Load
4. $**$#tmpStatus Unable to Load
5. Interruption Occurred on $**$#
6. $**$#, ..., #: Resetting run Status
7. $**$#, ..., #: Resetting run and wait Statuses
8. Attempting to Reconnect $**$#, ..., # After 4 Consecutive Failed Attempts
9. Too Many Consecutive Failed Attempts for $**$#, ..., #. Reconnection Not Attempted
10. Manual Reconnect Attempted

Nine of the warnings reference specific power meters with '$**$#' representing meter names such as PM1, PQ1, PQ2, SH1, etc. Warnings 1 to 5 pertain to an individual meter while warnings 6 to 9 may reference multiple meters of the same type. Warning 10 does not specifically name any meter since a user requested reconnect results in an attempted reconnect for all disconnected meters.

Directly underneath the automatic reconnect checkbox are the status lights for each activated meter. The status lights may be either green, blue or red. A green light signifies that the meter is connected and communicating with the application server. A blue status represents instances when the previous status was green but the program was unable to read from the meter.

Finally, the status is given the color of red if a read from the meter fails and the previous status was either blue or red.

The bottom portion of the Read All sub-tab is dedicated to providing the user with statistics for fields of data collected from each activated meter. By first selecting the desired meter then clicking the colored '+' symbol within one of the numbered grey boxes, a user can then proceed to select which field of data to generate the statistics for based on the most recent year's worth of data.

The Snapshot sub-tab, shown in Figure 3.3, provides further information on the meters and recent data read from them. On the left of this page is a list of active meters separated by their types. A description may also be present beside each meter to detail its physical location and/or other relevant information. A drop down menu is provided to select from the activated meters. Upon selection, another list will be shown with register group names that the user can choose from which will result in a dialogue box appearing with the registers from within that group. As registers are selected their names will appear in the blue text area. After selecting the desired registers from the available register groups, clicking the 'Submit' button will populate the table at the bottom of the page with the most recent entries of data for the fields selected by the user. It is important to note that only registers being continuously read from are available for selection and only data already stored in the database may be shown.

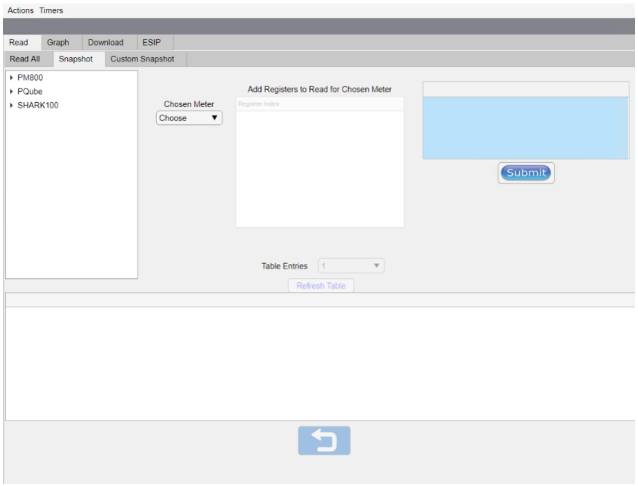

Figure 3.3. Read Tab: Snapshot

The number of entries within the table is determined by the value selected in the Table Entries drop down menu which includes 1, 10, 20 & 30 as options. When a new meter is selected

from the Chosen Meter drop down menu the table data will change to show the registers that were selected for that particular meter if any were chosen at all. To clear selections made for any one meter, the user can select the undo button at the very bottom of the page.

The final Read sub-tab is Custom Snapshot shown in Figure 3.4. Within this tab the user is capable of reading a register directly from a power meter instead of the database. The upper section of the page allows the user to make selections from drop down menus which include the power meter and the data type of the register to be read. Other inputs have to be typed such as the register/offset number, the size of the data, name, and the units of the data. The 'Name' and 'Units' inputs are not required to perform a read. The size of the data is only enabled for input if the data type is Mod10, Char or Bitmap, otherwise, the size is set. Size in this context represents the number of registers required to fully read the relevant data. Data types of Integer/SINT16, UINT16, Long, Float, DateTime, and Octets each have a constant size that is automatically set by the program with no need for input by the user.

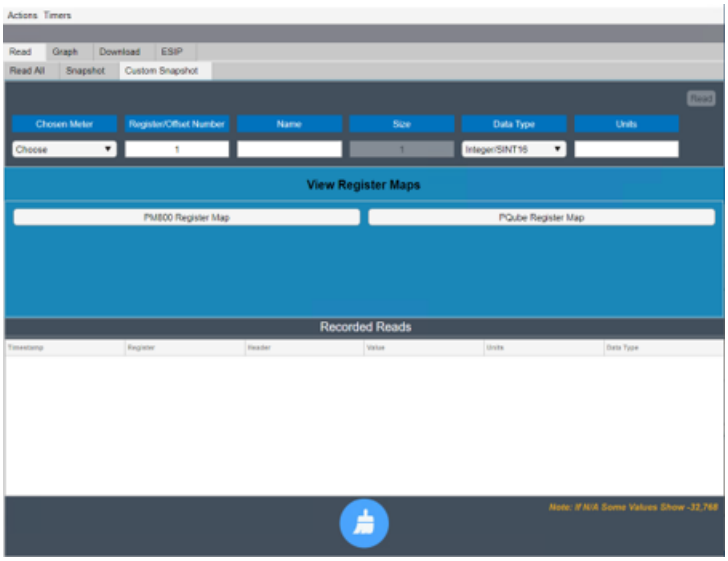

Figure 3.4. Read Tab: Custom Snapshot

The register/offset number requires the user to input the register address as seen in the register maps for the power meters which show zero-based register addressing. The user does not need to know the base address of the power meter being read as any needs to adjust the zero-based address is handled by the program. Provided in the middle section of the page are buttons to access the register maps for the types of power meters being utilized. Selecting any of these buttons will

open a separate window which will allow the user to browse a particular register map. The bottom section of the page includes a table of all of the user custom reads. Table entries in the Units and Header columns can be edited.

The Graph tab, displayed in Figure 3.5, provides the user with graphing capabilities for all of the collected data as well as some useful information not present within the Read tab. Starting from the top left of the page, the Select Meter Dataset table provides entries for each of the activated meters which include the filename in which their data is stored, the time of interface initialization and the number of interrupts that have occurred while reading from the meter since interface initialization. Selecting any entry from this table will enable a drop down menu with register group names for the meter. Once a group name is chosen from the menu, the Meter Details table is populated with the names of the registers within the chosen group, a timestamp of when data was last updated, and the last value recorded for each register.

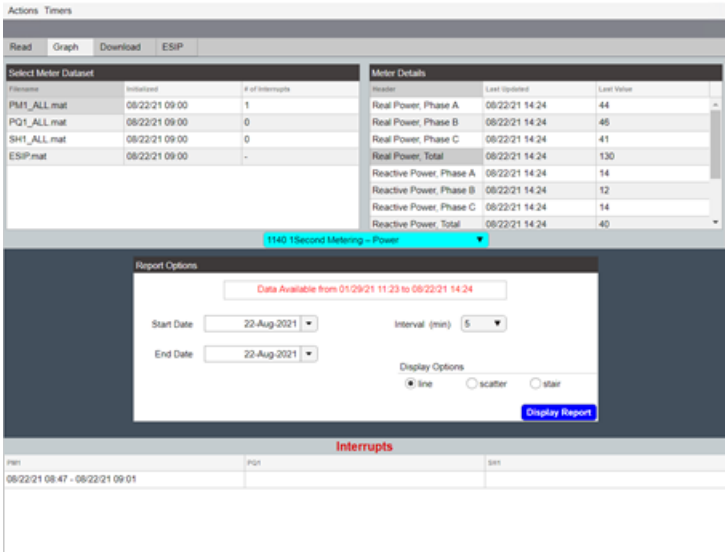

Figure 3.5. Graph Tab

Also occurring as a result of a selection from the menu is the red text seen in the Report Options section that details the date availability of the data selected. The red text and the information provided in the Meter Details table are not refreshed automatically, so in order to update the information available to be graphed the user must first either select a new entry in the Select Meter Dataset table or select the 'Choose' option of the drop down menu. Options to manipulate how the data is presented are available in the Report Options section of the page. The user selects

the date range, sample rate and plot style of the data before the Display Report button is enabled. The date ranges are only specific to the day, so the start date is assigned a time of 00:00 and the end date is assigned a time of 23:59. Specifying ranges down to the hour and minute can be done once the data has been graphed. Shown at the bottom of the page is a table of interrupts with the meter names serving as the column headers and their entries being the date range of the interrupts. If the interface is re-initialized, this list of specific date ranges will be lost.

When the display report button is selected, a new application shown in Figure 3.6 is opened which displays the graph as well as the dates of all the interrupts from the Interrupts table of the Graph tab. While the list of interrupts on this app will also disappear upon re-initialization of the main app, any missing data points in the graphed data will always be counted and displayed in the Stats of Shown Graph section. The range of the graph can be scaled smaller using two methods while this new app is opened. The first method is to utilize the time graph shown immediately under the main graph. Users can click and drag their mouse over the time graph's range of interest then select the clock symbol to the right. The original scale can be returned by selecting the home icon of the time graph then selecting the clock symbol again.

Figure 3.6. Report Example

The second method is using the section to the lower left portion of the page. The user is required to select a from date, left side drop down menus, and an end date, the right side menus, then select the Adjust Time Range button. The first method is faster while the second method

allows for more finely tuned scaling if needed. The original graph can be obtained by selecting the Redraw button in the Display Options section on upper left portion of the app. Also in this section are options to change the plot style of the shown graph.

The Download tab works similarly to the Graph tab with the main difference being that this tab is meant to select data that the user wishes to grab from the database to be used in other applications. A depiction of the Download tab is shown in Figure 3.7. The user goes about selecting the data, its range and interval just as in the Graph tab. In the Download Options section is an Add button that the user will select to include the desired register as a field to be downloaded. This process can be repeated for as many registers from as many meters as desired. Once all selections are completed, the user then selects the Download button which will present a save dialogue box to allow the user to choose the folder to which to write as well as the file type of the downloaded data.

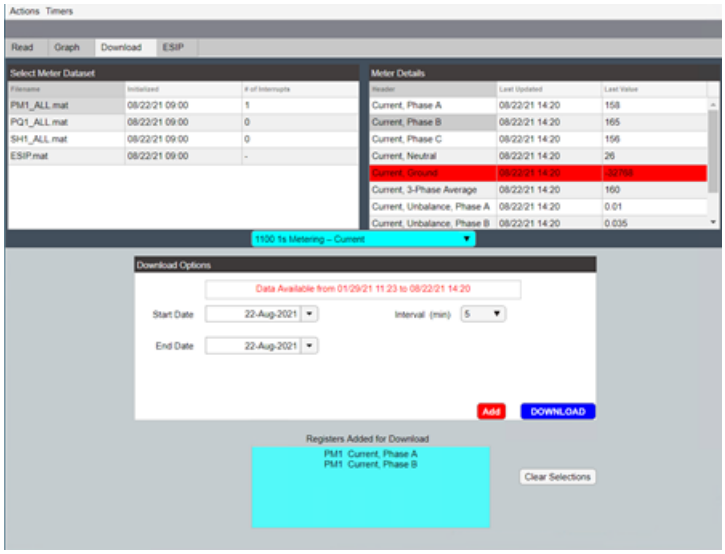

Figure 3.7. Download Tab

In the final tab, ESIP, two sub-tabs are present: Main, Alarms. Information from the Energy Storage Integration Platform (ESIP) is found on both tabs. Within the Main tab, shown in Figure 3.8, are the most recent recorded values from the Command and Read registers; the interlock bits are displayed as status lights. Shown in the middle of the page is the timestamp of when all ESIP information was last updated on the GUI. To the right of the Watchdog Time edit field near the top of the tab page is a button that allows the user to start and stop the incrementation of the watchdog timer. It is necessary that the watchdog value changes at least once per second to allow

the energy storage system to operate in Follow Power mode and allow control over the battery's charging and discharging.

The helpful info section contains buttons that display information that may be relevant to the user. Button 'P Cmd' provides a list of the recommended steps to take when sending a power command to the ESIP. The 'HIL Alarms' button displays human-in-the-loop alarms which are faults experienced by the ESIP that require a user's attention resolution; the ESIP system does not handle these alarms automatically as it does with other faults. To the left of the tab page, just below the command registers, are the interface features a user would use to perform a manual write to the ESIP. Before running the forecaster and battery scheduler described in 4 and 5, respectively, the user must first ensure the watchdog timer is incrementing and then change the ESIP's mode of operation to Follow Power. If the watchdog timer is not active when Follow Power mode is established, the Communication Error warning and the General Comms Fault alarm will both be set.

Figure 3.8. ESIP Tab: Main

On the Alarms tab, the statuses of all minor and major faults are displayed as seen in Figure 3.9. The minor faults are warnings that do not require a system reset. Warnings alone will not interfere with system operation. When a warning is set the status light and the text of the Alarms tab will turn yellow. Alarms are major faults that will interfere with operation and may require a system reset. When an alarm is set the status light changes to red; the text of the Alarms

tab will also become red regardless of any warnings that may also be set. Some alarms and warnings that may occur during Follow Power mode could potentially be reset by simply toggling the mode of operation to Standby then back to Follow Power instead of resetting the entire system. There are some warnings such as Charge Cmd Large & Discharge Cmd Large that will only reset some extended length of time after the mode of operation has been toggled.

The menus of the interface offer other information and actions the user may utilize. The Actions menu contains three commands: Save, Add Meter, Reconnect. The Save menu item allows the user to manually save the current state of the interface so that in the event the application shuts down for any reason, re-initialization will not be required. Saving is done periodically by the program itself, however, the manual save is useful if a planned shutdown is to occur and the autonomous save has yet to happen. The Add Meter menu item allows the user to add new meters of pre-approved types such as PM800 series, PQube 3, or Shark 100. Finally, the Reconnect menu item allows the user to perform a manual reconnect of all the disconnected meters. A message will be displayed in the Warning Log of the Read tab when a manual reconnect is performed.

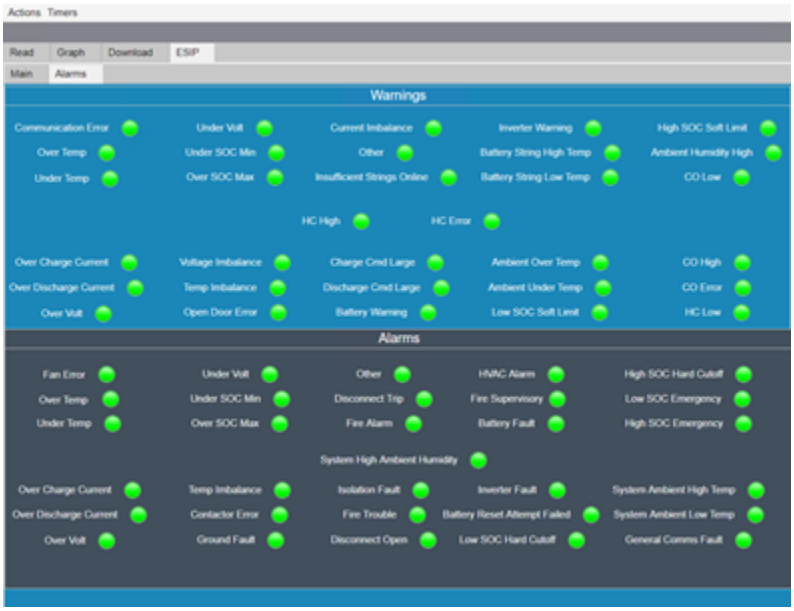

Figure 3.9. ESIP Tab: Alarms

In the Timers menu are items which allow the user to check on the status of all timers being used to perform autonomous operations. Timers for each type of meter are included as menu items. All meters of the same type share a timer. Also included as menu items are Save Timer

and Reconnect Timer. The Reconnect Timer can be turned off or on by deselecting or selecting respectively the Automatic Reconnect checkbox on the Read tab, but the other timers should always remain on. Selecting one of the Timers' menu items will produce a dialogue box with the status of the timer and its period. For the timers that should always remain on, the program continuously checks to ensure they are operational. All timers can also be re-established manually by performing a manual reconnect.

3.3.2 Parallel Computing & Timers

A timer operates by periodically calling a specific function, which is necessary for handling autonomous operations. However, the GUI itself is non-responsive while functions are being ran in the background. To reduce processes that may interfere with interface operability, MATLAB's Parallel Computing Toolbox is needed. This toolbox allows for the creation of what is called a pool, which contains a user specified number of workers. These workers can be thought of as parallel processors capable of performing an operation then returning the results to the GUI, which can be considered the main processor. While the workers carry out the actions of a function, the GUI remains responsive.

In order to adequately make use of the pool of workers, it is important to understand their limitations. For instance, as of MATLAB 2020b, workers are not inherently capable of viewing and accessing directories. This is remedied by adding specific folder paths to the pool so that all workers are capable of viewing the necessary files and folders. Limitations still persist in that a worker, while performing a task, sees the added directory as static. So, if a file is removed while the worker is carrying out an action, the worker will still see that file. This particular worker characteristic is prominent because workers are not capable of communicating amongst themselves. In fact, workers generally only communicate with the main processor when they receive a task and when they finish a task. Some form of communication is needed between workers to avoid errors and data collisions associated with multiple workers attempting to access the same power meter. Such an occurrence is possible because a timer does not wait for the completion of a parallel task before it starts counting to its next function call.

Creating and deleting temporary files or folders as a form of communication was not a viable option, however, while a worker may not be capable of seeing a change in a directory, it is capable of accessing a file whose contents have been altered by another worker. Each power meter

will have associated with it a file containing statuses. Only meter reading and data logging are done on parallel workers due to issues with accessing and changing global variables and GUI objects by workers; some response delays are still possible, though, any delay that may occur is expected to be negligible since the operations requiring the most time are completed in parallel.

3.4 Summary

The application server houses an interface, which retains all of the database rules for accessing data to void the need of a query language, and the DBMS, whose responsibilities include actively retrieving data from power meters on a 1-minute period. Communication between the application server and other devices, not including the database and client user, follows a protocol known as MODBUS for RTU and Ethernet transmission methods. The devices that require MODBUS are power meters installed within the ZGEC, however, there are intermediary devices between the application server and power meters.

Only three types of power meters were considered during the design of the application server program: PowerLogic Series 800 (PM800), PSL PQube 3, EIG Shark 100. Network switches are needed as intermediate devices for all power meters, but the PM800 also requires the use of a Lantronix UDS1100 since it operates on MODBUS RTU. Serial tunneling is needed for the PM800 to allow communication over Ethernet, but all other meters use the MODBUS TCP/IP protocol which require no further steps for communication.

The GUI housed on the application server is what the user interacts with to glean information from the DBMS and power meters; it also allows the user to take an assortment of pre-defined actions and provides diagnostics on communication with the power meters. Each tab of the interface offers its own set of actions and information for the user to employ.

Chapter 4

Real-Time Forecaster

Electrical demand of a building is typically in a state of fluctuation due to a multitude of factors. The severity of demand flux is directly proportional to the timescale being considered; viewing building demand on a 1-minute timescale normally shows far greater variability than viewing that same building's demand on a 1-hour timescale. This trait of demand is a driving factor when considering a forecasting methodology. Forecasting, as stated in [14], is the estimation of value that a variable or set of variables may be at a future time. Four categories of load forecasting, listed in [14], are: Very Short-term Load Forecasting (VSTLF), Short-term Load Forecasting (STLF), Mid-term Load Forecasting (MTLF), Long-term Load Forecasting (LTLF).

The range of time for which the forecasting is implemented is the difference between the categories. VSTLF focuses on a time range from a few minutes to a few hours while STLF is between a few hours to a few days. This trend continues with MTLF having a time period of a few weeks to a few months and LTLF encapsulating the largest range of time from one year to beyond. The requirements of a project and the parameters available determine the category of forecasting chosen which in turn drives the decision for the type of forecasting model implemented. A number of models are mentioned in [14],[15],[16],[17],[18],[19], and [20]. A few of note are:

- Time Series
- Regression
- Artificial Neural Network (ANN)
- Support Vector Machine (SVM)

Time series models are described in [21], [19], and [16]. This model is considered the simplest as it uses historical time series data to generate trends for predicting. Time series analysis is often used for VSTLF and STLF due to the complicated nature of load patterns within smaller ranges. There are multiple regression methods, as discussed in [18], but [17] describes this type of model

overall as a statistical process that develops a linear or non-linear relationship between variables. Load is usually considered the dependent variable with the independent variables being related to weather. While regression can be used effectively in all categories of forecasting, it is often used for LTLF according to [19].

ANN and SVM are both artificial intelligence forecasting methods that tend to be preferred for non-linear problems with a small amount of data [15]. ANN, SVM and certain types of regression models are better suited for the cases that rely on variables beyond just demand. These extra variables tend to be weather information such as temperature and humidity, though, other parameters could include human occupancy and other indoor conditions. Having independent variables to accompany demand is suited for 'what-if' analyses, which is an advantage over single variable models when considering wider ranges of time. A potential downside to incorporating more variables is that the model becomes more susceptible to unreliable data.

All of the forecasting models have positive and negative traits with some seeming more or less important depending upon what is needed of the forecaster. The forecasting model for this project can be characterized as time series with demand being the single variable. This model is intended to meet the accuracy and computational efficiency standards of the regression model discussed in [20] while relying solely on historical timeseries data. The model operates by first grabbing the last five years of demand data upon startup then continues to grab only the most up-to-date data as it continues to operate.

Reference days for the days to forecast are collected through a process that finds previous similar types of days as well as the previous week of days relative to each particular forecasted day. These collected days are then subjugated to a validation process in an attempt to avoid allowing outlier days to be considered as reference. The criteria of the validation process are discussed further in Section 4.1.

The forecaster model used is centered around using the reference days to predict a specified timestep ahead. A 5-minute timestep is used as the discrete period for which the entire model operates, therefore, each 5-minute timestep the forecaster predicts the building demand for the next 5 minutes. This process is explained in Section 4.2. After a 5-minute step ahead forecast is performed, it is then used along with the reference days to generate a forecast that includes the next 24 hours, which is called the full range forecast throughout this paper. The full range forecast has a 15-minute resolution and its formation is described in Section 4.3.

The full range forecast is fed to the battery scheduler that uses a Model Predictive Control (MPC) scheduler based upon the optimizer in [20]. The scheduler is called every 15 minutes during normal operation and requires all inputs to be of 15-minute resolution no matter the mode of operation. There are special scenarios which will allow for the scheduler to operate on a 5-minute basis, though, all inputs must still be converted to a 15-minute resolution. MPC scheduling during normal operation is discussed in Section 5.1, and MPC scheduling during special modes of operation is discussed in Section 5.2.

As stated previously, the model as a whole operates on a 5-minute timestep, so the execution of the algorithm must be complete within that five minute window. During startup of the model, if it has not been previously compiled, compile time and code execution requires 2 to 3 minutes. Independent of the DBMS, the implementation of the code after startup would generally be complete within ten seconds. However, since the model requires access to data files that are also accessed by the DBMS, there is a 'handshake' between the model and the DBMS that allows access to certain data files without interference. Accounting for a potential wait time due to DBMS operation, code execution requires less than a minute after startup.

4.1 Gathering Historical Data and Selecting Reference Days

Before any forecasts or schedules can be made, the appropriate timeseries data sets must be gathered. During startup of the model, up to five years of demand-related data may be retrieved from the database. As of the writing of this paper, less than a year of data has been recorded from the PM800. In order to make up for the lack of 1-minute resolution data, the 15-minute resolution data from the Dominion Energy power meter attached to the ZGEC is used. All data is converted to a 5-minute resolution. The 1-minute data is averaged to 5-minute data, but the 15-minute data is just repeated to mimic 5-minute data. The Dominion Energy power meter data is only used to fill out demand data prior to the recording of the PM800 data and is only accessed at startup.

After startup, only the most recent data is grabbed every time step. Barring any interrupts, recent data in this context is defined as all information recorded within the last completed 5-minute bin. Throughout this paper the term 'bin' will be used to describe a window of time as described in Table 4.1.

Four sets of timeseries data must be retrieved for the model every timestep:

1. PM800: 'Real Power, Total'

2. ESIP: 'ESS Total DC Power'

3. ESIP: 'Active DC Power String 1'

4. ESIP: 'ESS SOC'

Table 4.1. Bin Descriptions

5-Minute Bins [hh:mm]	15-Minute Bins [hh:mm]
hh:00 – hh:04	hh:00 – hh:14
hh:05 – hh:09	hh:15 – hh:29
hh:10 – hh:14	hh:30 – hh:44
⋮	hh:45 – hh:59
hh:55 – hh:59	

Note: The model should begin operation at the start of a 15-minute bin

The total real power read from the PM800 provides the power from the perspective of the grid. Since the battery is behind the meter along with the building, the PM800 power meter is not able to distinguish between battery output and building load. In order to properly forecast building load and schedule the battery, building load alone is required. There are no voltage, current or power measurements from the AC side of the inverter, so data sets 2 and 3 are used instead for the calculation to approximate the building load. Active DC Power String 1 provides the absolute value of ESS Total DC Power plus the DC power draw of the inverter when connected. Since the value from this register is not signed, the sign is taken from data set 2. ESS Total DC Power is negative when the battery is charging and positive when discharging. The real power read from the PM800, also considered the optimized power, and the calculated building load are resampled to a 5-minute resolution.

$$\begin{aligned} BuildingLoad = {} & [Real\ \ Power, Total] + \\ & sign([ESS\ Total\ DC\ Power]) \cdot [Active\ DC\ Power\ String\ 1] \end{aligned} \tag{4.1}$$

Data set 4 is the battery's state of charge (SOC). Unlike the other data sets, only the very last read SOC is retrieved and there is no need for resampling. The final step before finding the

proper reference days is to assign the appropriate day types to the timeseries data collected. Day type is a numerical value of the range 1 to 7 that represents days Sunday to Saturday respectively as shown in Table 4.2.

Holidays are a special exception and are designated a day type of 7 no matter what day of the week they fall. The holidays considered are New Year's Day, Memorial Day, Independence Day, Labor Day, Thanksgiving and Christmas. Also considered is the meteorological season, shown in Table 4.3, in which a day resides. Ideally, all previous similar days collected for reference must share the same day type and season type as the day being forecasted.

Table 4.2. Day Types

Day	Day Type
Sunday	1
Monday	2
Tuesday	3
Wednesday	4
Thursday	5
Friday	6
Saturday	7

Table 4.3. Season Types

Months	Meteorological Season	Season Type
Mar – May	Spring	1
Jun – Aug	Summer	2
Sep – Nov	Fall	3
Dec – Feb	Winter	4

Reference days are split into two categories: previous similar days and previous consecutive days. Previous similar days are those which share the same day type and season type as the day forecasted. The only exception to this rule is if the number of complete similar days to collect does not meet the set threshold. The threshold is currently set to an arbitrary value of 40 similar days. If there are not enough complete similar days of the particular season within the historical data, then season type is allowed to change to the previous season while day type remains the same. With the inclusion of the Dominion Energy power meter data, this exception is not likely to be encountered.

Once the previous similar days have all been collected, they are then tested for their inclusion as reference days. The first test is to check all days against each other. In order to pass this test, the demand at each timestep throughout the day must fall within the corresponding demand range. This demand range is based upon the median demand of all the days at a particular time of day. Currently, the demand range is $[0.75 * Median, 1.4 * Median]$. The assumption is that any of the

previous similar days that significantly deviates from the rest is an anomaly that should not be used for reference. An example of the described 'Median Test' is depicted in Figure 4.1. Each box represents a demand range for a timestep, and the different colored orbs are the demand for each collected day at that timestep. Based upon the figure, Day 2 and Day 5 would be dismissed because they both have at least one demand value that falls outside the set range.

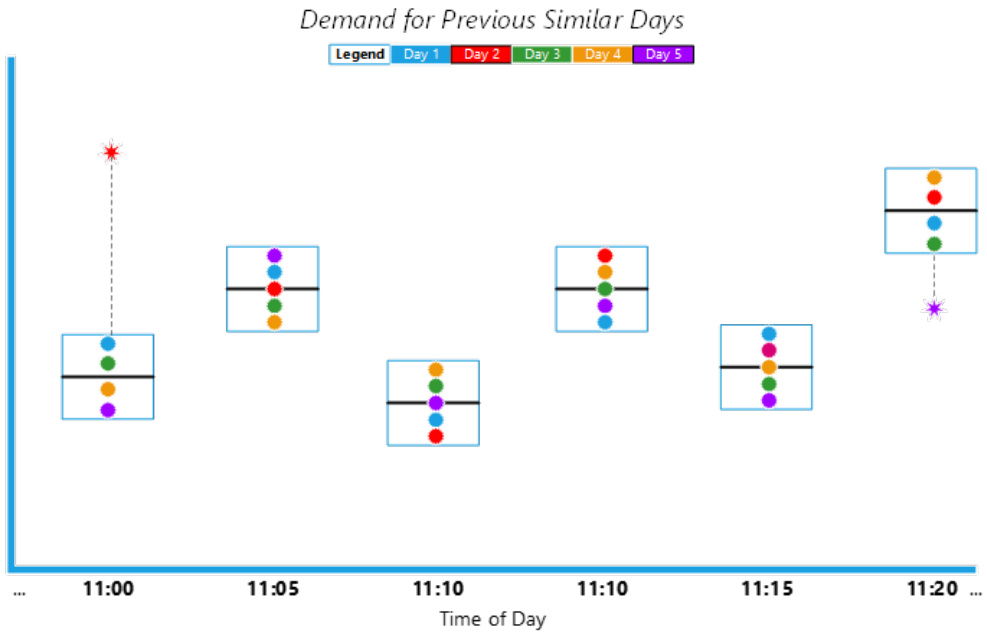

Figure 4.1. Median Test Example

The second test for the previous similar days is to check for absurd demand deviations that are short-lived. This test checks each day individually. The deviations of that day are checked against a threshold based upon that day's standard deviation. If the standard deviation for that day is greater than or equal to 40 kW, the result of the standard deviation divided by 1.5 is used for the threshold. If the standard deviation is less than 15 kW, the threshold uses a standard deviation of 15.

These manipulations of standard deviation are meant to improve the model's ability to spot momentary spikes or valleys in demand. Any of the large deviations in demand are considered short-lived if they last three hours or less. Three hours is somewhat of an arbitrary threshold that was chosen because the large unpredictable demand spikes and valleys are not consistent in their duration; the expectation is that most large deviations last less than three hours. Only the previous

similar days that pass both the median and standard deviation tests are allowed as reference days.

The other category of reference days, previous consecutive days, consists of the last 6 complete days. Each day in this group is tested individually for absurd deviations, which is the same process as the second test for previous similar days. Since the previous consecutive days will differ in day types, and potentially season types, the days are not tested against each other. Only the days from both categories that pass their respective validation process are considered reference days within the model. Although, if no days from either category pass the validation process, then all collected days would be included as reference days, but that scenario is unlikely when given sufficient historical data.

4.2 Forecasting Single Step Ahead

This model is predicated on forecasting the electrical demand of the building a single timestep ahead. The process begins with a few pre-forecasting actions, which occur every timestep, that produce inputs required for the step ahead forecast to operate. During the pre-forecast procedure a count is taken of the number of 5-minute step ahead forecasts made since the start of the model. Also, the 5-minute mean building demand for the day being forecasted is gathered as that day progresses; all demand for that day is gathered regardless of when the model starts. This demand data will be referred to as true demand. The final act of this process is to choose the reference day that the forecaster will use as the initial basis for its prediction.

Finding the base reference day starts by taking the absolute difference between the last recorded building demand and the demand of each reference day that corresponds to the time of the last recorded demand. At the start of a day, exactly at midnight, the building demand corresponding to midnight for each reference day uses the last recorded demand of the previous day for the difference comparison. The reference day with the smallest absolute difference in relation to the last recorded value is chosen as the basis reference day. If multiple days equally share the smallest difference, then a closest norm tie-breaker is used. Those reference days, up to the time of day of the last recorded value, are compared with all of the demand recorded for the day of interest thus far to determine which is closest in the sense of the Euclidean norm, or 2-norm. If, by chance, multiple days still share the minimum difference, one of those days is chosen at random.

After the pre-processing phase, there are still certain actions which must be taken to ensure

the most accurate forecast is made. The first task is to remove 'bad' data from True Demand which may adversely affect the forecast. Bad data is considered to be the large unpredictable deviations in demand. Similar deviations were discussed in Section 4.1 as being one of the causes for the disqualification of reference days. True demand undergoes an outlier test the same as a reference day except in this instance any outliers found are replaced with values calculated from taking the average of the 'normal' demands on either side of the large deviation. Any previous forecasts which were affected by the outliers are also replaced. It is important to note that temporary variables are used to hold the adjusted true demand and forecasts. Final building load and forecast results are not altered.

The other task performed is finding the median demand deviation between the previous timestep and the current time being forecasted within the reference days. This task is only performed if there exists at least one other reference day whose demand at the time of interest is within 2% of the basis day's demand at that same time. So, to best predict the behavior of the building load at a specific time, the basis day may change if one of the days within 2% also has a deviation closer to the median deviation at the time of interest. An extension of this task, which is performed regardless of any other reference days being close to the basis day, is to determine if a common deviation is shared amongst most of the reference days at the time of interest.

For an even number of reference days, a deviation is considered common if at least 50% of the days exhibit a similar change; an odd number of reference days require more than 40% share the behavior. Knowing whether a behavior is common or not helps determine how to best utilize the predictor methods that are used to forecast the next timestep. There are five predictor methods used in this model:

1. Initial Basis Day Error Prediction [F_{EP}]
2. Full True Demand Hankel [F_T]
3. Single Vector True Demand Hankel [F_{VT}]
4. Implicit Forecast Hankel [F_F]
5. Single Vector Implicit Forecast Hankel [F_{VF}]

The initial predictor method, F_{EP}, is the simplest of the five. It relies on two error values found using true demand and the basis day demand. The first error value is 5-minute based and

is essentially a slightly reduced difference between the last recorded building demand and the corresponding basis day demand. The second error value follows the same method as the first except it uses the corresponding 15-minute mean demand of the basis day. An initial forecast is generated using the corresponding 5-min basis demand, 15-min basis demand and one of the error values as shown in Eq. (4.2).

$$pf_5 = F_{EP} = 0.6 \cdot b_5 + 0.4 \cdot b_{15} + Error \tag{4.2}$$

If a common deviation is found amongst the reference days, the first error is used to generate the initial forecast and the initial forecast is the final prediction used by the model. Otherwise, if a common deviation is not shared between the reference days, the second error is used to generate the initial forecast followed by the generation of forecasts using the other predictor methods. Methods 2 through 4 use the Hankel matrix, or an adaptation of it, to describe the "algebraic relationships between elements of the sequence without pretending to approximate the analytical model of an underlying dynamical system" [22].

Predictor method 2, Full True Demand Hankel, is outlined in Eq. (4.3 - 4.6). This method builds a matrix of the true demand with d_{k-n} representing the building demand n timesteps before time of interest k. The unknown being solved for is demand at time k, d_k, which is also represented as F_T in Eq. (4.6). In all methods n is set to 6, so each row of the Hankel matrix contains thirty minutes of demand data. Method 3 is the same as method 2 except it uses the only the last row of the Hankel matrix, H_T, and output vector, Y_T. Methods 4 and 5 differ from 3 and 4 in that they use the demand forecasts generated since the start of the model and are somewhat implicit. Instead of treating the value at time k as an unknown, these methods use the initial prediction as a guess to formulate their respective forecasts. Methods 4 and 5 utilize linear least squares estimation to generate their predictions. The benefits and disadvantages of using any of the predictor methods individually are listed in Table 4.4.

True Demand

Full

$$H_T = \begin{bmatrix} \vdots & \vdots & \vdots \\ d_{k-(2n-1)} & \cdots & d_{k-n} \\ \vdots & \vdots & \vdots \\ d_{k-(n+2)} & \cdots & d_{k-3} \\ d_{k-(n+1)} & \cdots & d_{k-2} \end{bmatrix} \tag{4.3}$$

$$Y_T = \begin{bmatrix} \vdots \\ d_{k-n} \\ \vdots \\ d_{k-1} \end{bmatrix} \tag{4.4}$$

$$\alpha_T = ({H_T}^* \cdot H_T)^{-1} \cdot {H_T}^* \cdot Y_T \tag{4.5}$$

$$F_T = \begin{bmatrix} d_{k-n} & \cdots & d_{k-1} \end{bmatrix} \cdot \alpha_T \tag{4.6}$$

Single Vector

$$H_{VT} = \begin{bmatrix} d_{k-(n+1)} & \cdots & d_{k-2} \end{bmatrix} \tag{4.7}$$

$$Y_{VT} = \begin{bmatrix} d_{k-1} \end{bmatrix} \tag{4.8}$$

$$\alpha_{VT} = ({H_{VT}}^* \cdot H_{VT})^{-1} \cdot {H_{VT}}^* \cdot Y_{VT} \tag{4.9}$$

$$F_{VT} = \begin{bmatrix} d_{k-n} & \cdots & d_{k-1} \end{bmatrix} \cdot \alpha_{VT} \tag{4.10}$$

Forecasted Demand

Full

$$H_F = \begin{bmatrix} \vdots & \vdots & \vdots \\ f_{k-(2n-1)} & \cdots & f_{k-n} \\ \vdots & \vdots & \vdots \\ f_{k-(n+1)} & \cdots & f_{k-2} \\ f_{k-n} & \cdots & f_{k-1} \end{bmatrix} \tag{4.11}$$

$$Y_F = \begin{bmatrix} \vdots \\ f_{k-n} \\ \vdots \\ f_{k-1} \end{bmatrix} \tag{4.12}$$

$$\alpha_F = ({H_F}^* \cdot H_F)^{-1} \cdot {H_F}^* \cdot Y_F \tag{4.13}$$

$$F_F = \begin{bmatrix} f_{k-(n-1)} & \cdots & f_{k-1} & F_{EP} \end{bmatrix} \cdot \alpha_F \tag{4.14}$$

Single Vector

$$H_{VF} = \begin{bmatrix} f_{k-n} & \cdots & f_{k-1} \end{bmatrix} \tag{4.15}$$

$$Y_{VF} = \begin{bmatrix} f_{k-1} \end{bmatrix} \tag{4.16}$$

$$\alpha_{VF} = ({H_{VF}}^* \cdot H_{VF})^{-1} \cdot {H_{VF}}^* \cdot Y_{VF} \tag{4.17}$$

$$F_{VF} = \begin{bmatrix} f_{k-(n-1)} & \cdots & f_{k-1} & F_{EP} \end{bmatrix} \cdot \alpha_{VF} \tag{4.18}$$

Table 4.4. Characteristics of Predictor Methods

Method	Pros	Cons
F_{EP}	•Fastest Response to Large Deviations •Best Predictor for Large Deviations •Low Susceptibility to Bad Data	•Most Error During Standard Variability
F_T	•High Accuracy During Standard Variability	•Slow Response to Large Deviations •Needs More Data Points •High Susceptibility to Bad Data
F_{VT}	•Needs Less Data Points than F_T •Faster Response than F_T	•Moderate Response to Large Deviations •Moderate Susceptibility to Bad Data
F_F	•High Accuracy During Standard Variability	•Slow Response to Large Deviations •High Susceptibility to Bad Data
F_{VF}	•Faster Response than F_F •Needs Less Data Points than F_F	•Moderate Susceptibility to Bad Data •Moderate Response to Large Deviations

In order to produce the most accurate forecast at all times, all predictor methods are considered. Let $\bar{F}_{all}$ represent a vector containing the result of each method, d_k is the last recorded demand of the building and let F_{TD} be whichever member of $\bar{F}_{all}$ that is closest in value to d_k. Eq. (4.19) provides the calculation for determining the final step ahead forecast used for the model when, in most instances, there is no common deviation amongst the reference days. If it is noticed that a significant deviation is shared amongst the reference days, then Eq. (4.2) is used instead.

$$pf_5 = 0.7 \cdot F_{TD} + 0.15 \cdot d_k + 0.15 \cdot median(\bar{F}_{all}) \tag{4.19}$$

4.3 Forecasting Full Range

Since the battery scheduler takes as input 15-minute mean data and requires at least 24 hours of data, unless the remaining forecast window is shorter, it is necessary to go beyond a single 5-minute step ahead forecast. If multiple days remain in the time of interest to forecast, then only up to two days are forecasted during a single step of the model as a means of reducing run time; also, the battery scheduler only looks at a 24-hour window so there is no need to forecast beyond two complete days. Generating the forecast for the full range of time needed is an iterative process. Figure 4.2 illustrates the process. The objective of the process is to generate an optimized 15-minute forecast based on step ahead forecasts of 5 and 15-minute resolutions.

First, 5-minute and 15-minute step ahead forecasts are performed, represented as pf_5 and pf_{15} respectively. The 5-minute step ahead forecast was described in Section 4.2, and the 15-minute

step ahead forecast follows the same process as described in the previous section but with a different resolution. The optimized 15-minute forecast-pf_{Opt15}-is developed using pf_5, pf_{15}, and any true demand recorded within the 15-minute window if applicable.

Let a 15-minute demand window be comprised of 5-minute demand values x_1, x_2, and x_3; also, let n be the number of static members within a window. At the start of a 15-minute bin, n is set to 1 and x_1 is set to pf_5 as the only static member. The optimization process will find x_2 and x_3, therefore, also providing pf_{Opt15} as the mean of all three members. The next stage of the full range forecast process is to treat the members as if they were actual building load and continue to forecast for the subsequent 15-minute window. This process is repeated until the end of the range of interest.

$$Minimize\ OF_1 = \frac{x_1 + x_2 + x_3}{3} - x_4 \tag{4.20}$$

$$Minimize\ OF_2 = \frac{-x_1 - x_2 - x_3}{3} + x_4 \tag{4.21}$$

subject to

$$x_u \geq 1.1 \cdot F_{15} \tag{4.22}$$

$$x_u \leq 0.98 \cdot F_{15} \tag{4.23}$$

$$\frac{x_1 + x_2 + x_3}{3} = x_4 \geq 0.98 \cdot F_{15} \tag{4.24}$$

$$\frac{x_1 + x_2 + x_3}{3} = x_4 \leq 1.02 \cdot F_{15} \tag{4.25}$$

$$F_{15} = \frac{n}{3} \cdot \frac{\sum_{i=1}^{n} x_i}{n} + (1 - \frac{n}{3}) \cdot pf_{15} \tag{4.26}$$

$$where\ x_u = unknown\ 5 - minute\ forecast,\ n = \#\ of static\ members$$

Once the model progresses to the next five minutes in real-time , actual 5-minute building demand will have been recorded within the current 15-minute window. In this scenario, x_1 will be set to the actual true demand while x_2 is set to pf_5 and n is set to 2. Only x_3 will be found during the optimization process of the current 15-minute window. All subsequent forecasts within full range will still set x_1 to pf_5 and n to 1. Another progression of five minutes in real-time implies x_1 and

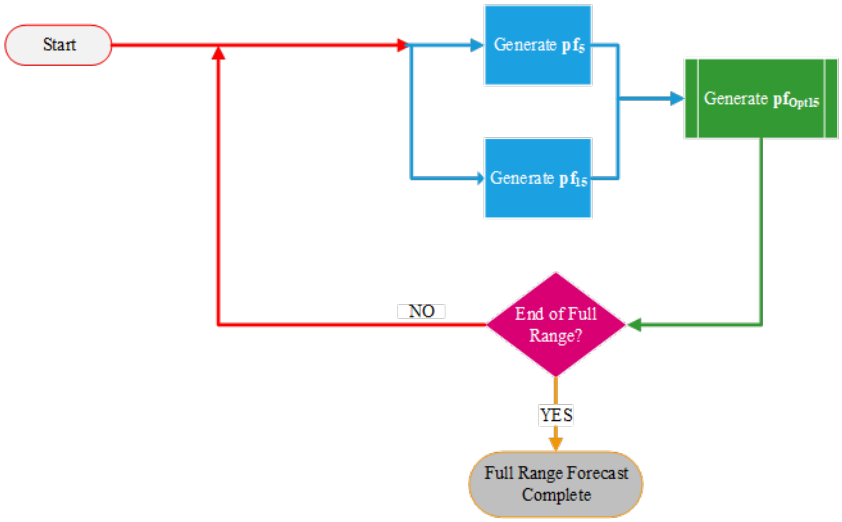

Figure 4.2. Full Range Forecast Flow Diagram

x_2 are actual recorded building demand values in the current 15-minute window. In this instance, x_3 will be set to pf_5 and no optimization is needed for the current 15-minute window. The rest of the full range forecast will be computed by setting n to 1 and x_1 to pf_5 just as in each of the previous scenarios. Only the pf_{15} and pf_{Opt15} generated at the start of the current 15-minute bin are recorded and used for forecast comparisons.

The optimized 15-minute optimization process is formulated as a linear programming problem. Eq. (4.20-4.21) are the objective functions minimized and subject to constraints in Eq. (4.22-4.26). The objective functions are negatives of each other. Different results are obtained depending on the sign of the function when using 'linprog' in MATLAB for finding a solution. The results of each objective function are averaged together to provide the values for each non-static member. In instances when no solution is found using either objective function, Eq. (4.26) provides the final result by averaging the static members with the basic 15-minute step ahead forecast.

4.4 Summary

A time series model is used for forecasting. The model is based fundamentally upon forecasting 5 minutes ahead, however, the battery scheduler described in Chapter 5 requires a resolution of 15 minutes to operate. A forecast is generated for 15 minutes ahead using the same methodology as forecasting ahead 5 minutes. Both the 5 and 15-minute step ahead forecasts are then used to

create a more accurate 15-minute forecast. This method provides 5-minute forecasts for the entire 15-minute window of interest, and these values are then used in an iterative process that generates forecasts for the next 24 hours. The 15-minute mean of that 24 hour forecast is provided to the battery scheduler.

The forecasting model only has a single input which is historical data. Having a single input allows for a simpler model but also limits the model's ability to recognize bad data, or large and unpredictable demand deviations. A collection of days taken from the historical data to be used as reference are selected based on either their corresponding day and season types or if they occurred within the last week. Reference days are subjugated to a validation process to avoid days that have bad data for a more accurate forecast.

Chapter 5

Battery Scheduler

5.1 Battery Scheduling

The battery scheduler used in this model executes in real-time on a 15-minute basis during normal operation and is based on the MPC scheduler developed in [20], formulated as a Mixed Integer Linear Programming (MILP) problem, with adjustments to better meet the needs of the developed model. Eq. (5.1) is the objective function to be minimized, incorporating time-of-use energy and demand rates of Rate 21A as described in [23]. Energy rate, $\rho_E(t)$, may change in value depending on time of day and time of year. Parameters $\rho_{D1}(t)$ and $\rho_{D2}(t)$ are demand rates for off-peak and on-peak times respectively and may change depending on time of year. Both P_{Shave1} and P_{Shave2} are determined by the MILP problem but have variable upper bounds.

During normal operation, while the lower bounds for the two variables remain static at 0 kW, the upper bounds may vary depending upon past grid demand within the one month billing window. Grid demand is defined as the sum of the building demand and the battery demand. As an equation grid demand can be defined as $P_3(t) = P_1(t) + P_L(t) - P_2(t)$. A depiction of power flow is provided in Figure 5.1. P_{Shave1} has an upper bound that is set to the maximum grid demand achieved during off-peak hours. At least two hours of demand data must be collected during the off-peak time window for maximum off-peak demand to be used as the upper bound for P_{Shave1}, otherwise, it is set to be equivalent to the upper bound of P_{Shave2}.

Derived using a similar principal, P_{Shave2}^{max} is set to the maximum grid demand achieved during the billing window. So, when the model is operating during on-peak time only past demand data during on-peak times are considered, and during off-peak time only past demand data during off-peak times are considered. At least forty minutes of data within the corresponding time window is required for this bound to be set based on past actual demand, otherwise, the bound is set to a value 5% greater than the maximum forecasted demand for the day of interest.

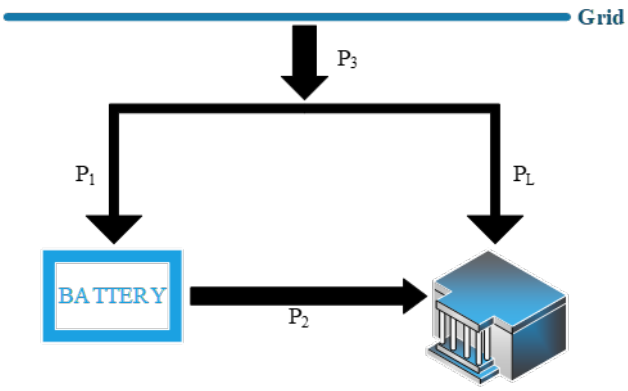

Figure 5.1. Power Flow w/ Battery

Calculated SOC is provided by Eq. (5.2) with the assumption that the energy storage system is 90% efficient when both charging and discharging; this calculated SOC is only used if SOC read from the ESIP is unavailable or at the end of the model's run-time when an ideal forecast is considered for comparative purposes. Eq. (5.3 - 5.4) ensure that only $P_1(t)$ or $P_2(t)$ is allowed to have a non-zero value at any instance in time as they both represent the charging and discharging of the battery respectively. Preferably, in Eq. (5.7), the maximum and minimum for the battery's SOC would be 90% and 10% respectively. However, to avoid soft limit SOC warnings from the ESIP due to charging and discharging, the maximum and minimum SOC are set to 89% and 11.5% in that order. The minimum SOC has an extra 0.5% difference from the ideal boundary to account for the innate power consumption of the inverter.

Eq. (5.8) is a constraint not found in [20], which works to keep the grid demand during off-peak below a previously established maximum threshold; this is equivalent to minimizing the difference between the maximum demands during off-peak and on-peak times, therefore, reducing cost. Rate 21A has an overall cost that considers the base facilities charge, energy usage, maximum demand during on-peak time, and one of three positive differences as listed in [23].

$$\min_{P_{1,2},P_{Shave1,2}} J = \sum_{t=1}^{N}[(P_1\ (t) + P_L\ (t) - P_2\ (t)) \cdot \Delta t \cdot \rho_E(t)] + [P_{Shave1} \cdot \rho_{D1}(t) + P_{Shave2} \cdot \rho_{D2}(t)] \tag{5.1}$$

subject to

$$SOC(t) = SOC(0) + \frac{\eta_C \sum_{\tau=0}^{t-1} P_1(\tau)\Delta t - \frac{1}{\eta_D} \sum_{tau=0}^{t-1} P_2(\tau)\Delta t}{B_{Cap}} \tag{5.2}$$

$$\beta_C K \geq P_1(t) \tag{5.3}$$

$$(1 - \beta_C(t))K \geq P_2(t) \tag{5.4}$$

$$0 \leq P_1(t) + P_L(t) - P_2(t) \leq P_{Shave1}\ for\ t \in t_{off-peak} \tag{5.5}$$

$$0 \leq P_1(t) + P_L(t) - P_2(t) \leq P_{Shave2}\ for\ t \in t_{on-peak} \tag{5.6}$$

$$SOC^{min} \leq SOC(t) \leq SOC^{max} \tag{5.7}$$

$$P_3(t) \leq P_3^{max}(t)\ for\ t \in t_{off-peak} \tag{5.8}$$

$$0 \leq P_{1,2}(t) \leq 50\ kW \tag{5.9}$$

The first potentially positive difference is maximum off-peak demand minus maximum on-peak demand. This is the only difference considered by the model when calculating costs. Difference two requires a contract demand value that is unavailable since the ZGEC is currently under another rate structure. Finding the ideal contract demand is beyond the scope of this project. The third difference assumes a maximum demand of 50 kVA during the entire billing period which is considered infeasible in this particular application.

5.2 Special Modes of Operation

Normal operation of the model has been described throughout this chapter, but there are other modes which allow for processes to deviate from what has previously been established. There are a total of four modes of operation:

1. Normal

2. Forecast Error Correction

3. SOC Soft Limit

4. Emergency Control

Forecast error correction operation is enabled when the 15-minute step ahead prediction made at the start of the bin differs somewhat significantly from the step ahead prediction made once actual 5-minute mean building demand from within the 15-minute window is known. This mode cannot be set if emergency control is engaged. During forecast error correction operation, the battery scheduler is given the updated full range forecast which includes the adjusted 15-minute step ahead prediction; this allows the scheduler to 'tweak' the battery power command to accommodate the new expected building demand without waiting 15 minutes as it would in normal operation. This mode of operation only lasts for a single iteration of the model before returning to normal operation.

SOC soft limit operation is engaged whenever the battery SOC is $\leq 15\%$ or $\geq 85\%$. The purpose of this mode is to avoid over charging/discharging which could potentially occur if a large enough power command sent to the battery is allowed to remain set for a full 15 minutes when the SOC is close to either of its bounds. While in this mode, the battery scheduler is called every 5 minutes for as long as the SOC within the determined limits. Normal operation may only return once SOC outside the pre-defined bounds.

The final mode, emergency control operation, also calls the battery scheduler on a 5-minute basis until normal operation is restored; it activates whenever a large unpredictable deviation in demand occurs. A significant unexpected increase in demand, referred to as a spike, or an unpredicted major decrease in demand, referred to as a valley, will cause mode four to be engaged. When a spike or valley occurs, the full range forecast becomes useless as it is no longer able to reasonably predict the behavior of the building's demand; only the 15-minute step ahead forecast is retained during this mode while the rest of the full range forecast is manipulated such that the battery scheduler may work to counteract sudden demand change.

When a valley is encountered, the model prioritizes charging as much as possible for thirty minutes without exceeding a certain demand threshold before switching back to normal operation. A valley may occur when demand drops too low during normal operation or when demand drops back to normal after spiking. The purpose of fast charging is mainly to allow the battery to regain enough charge to handle potential new spikes. Fast charging is made possible by transforming the

forecast for the day of interest to contain an artificial spike in demand thirty minutes beyond the current time of interest while all other forecast values for the day are set to the smaller demand value given by the step ahead forecast. The value for the fake spike may be adjusted to ensure that the charging is great enough to charge effectively yet not large enough for the grid demand to exceed a set limit.

The threshold for the grid demand when a valley is encountered is calculated by first finding P_{Shave2}^{max} the same as during normal operation, however, in this scenario if not enough data points are available within a particular time window, then P_{Shave2}^{max} is set to the maximum optimized grid demand achieved regardless of whether it occurred during on-peak or off-peak hours. Since P_{Shave2}^{max} is meant to set the limit of the overall 15-minute mean demand, a new shave value is calculated which reflects the average of the true demand within the current 15-minute window as each 5-minute step is taken. The method of calculating the new shave value, or demand threshold, is shown in Eq. (5.10) where x_i represents 5-minute demand that has been recorded within the 15-minute window. If there is no recorded 5-minute demand available within the window, then P_{Shave2}^{max} is used as the demand threshold.

$$Demand\ Threshold = 2 \cdot P_{Shave2}^{max} - \frac{\sum_{i=0}^{n} x_i}{n}\ ,\ n \in [1, 2] \tag{5.10}$$

When a spike is encountered during on-peak hours, the model prioritizes shaving the increased demand with a return to normal operation when off-peak hours begin. P_{Shave2}^{max} is set to be approximately 83% of the 15-minute step ahead forecast and the demand threshold is calculated as shown in Eq. (5.10). The full range forecast is transformed to show the increased demand for the remainder of on-peak hours and a valley for the remaining day of interest once on-peak hours are over. The maximum demand used to represent the spike is the step ahead forecast value while the minimum demand used to represent the valley is set to the minimum forecasted demand thus far. To ensure effective discharging, the maximum demand used to represent the spike may be adjusted. Spikes are mainly caused by the building's boiler system which can draw a minimum of $\approx$ 30 kW and maximum of $\approx$ 105 kW, though, will typically require $\approx$ 75 kW.

After thirty minutes of emergency control operation, when caused by a spike, demand is checked to verify it has grown at least to the minimum boiler demand. If demand has not grown by approximately 30 kW in this time frame then operation is returned to normal, otherwise, emergency

control continues. A spike induced emergency control may also end before on-peak hours are over if a valley occurs within peak hours. Should a spike occur during off-peak hours, operation switches to emergency control but the full range forecast is not transformed, therefore, no significant shaving is prioritized. If emergency control does not disengage after being set during off-peak hours, then full range forecast transformation will occur once on-peak hours are reached, and emergency control will remain engaged until SOC is below 15% or another reason for disengaging arises.

5.3 Summary

Scheduling of the battery is formulated as a MILP problem that works to minimize the overall cost of electricity in real-time with respect to Dominion Energy's Rate 21A. The building's energy consumption and demand are to be reduced while considering their respective time-of-use prices. Prices for demand and energy during a certain range of hours known as on-peak hours are higher than during all other hours known as off-peak hours. There are four modes of operation by which the battery scheduler may operate based upon what the forecasting model deems the highest priority.

Normal operation is the standard mode of the scheduler which prioritizes reducing costs associated with both energy usage and demand. If the forecasting model could perfectly predict load behavior, no other modes would be needed. The second mode of operation, Forecast Error Correction, is specifically for instances when the forecaster recognizes that it erred significantly. This mode allows the scheduler to run on a 5-minute timestep instead of waiting a full 15 minutes to adjust its power command.

Mode 3, SOC Soft Limit, provides protection against overcharging and over discharging. The scheduler operates on a 5-minute time step when the battery's SOC is within 5% of the SOC limits. The last mode is Emergency Control. Demand during on-peak hours contributes most to the monthly bill under Rate 21A, so whenever the building demand changes drastically in an unpredictable manner, Emergency Control is engaged. This mode prioritizes limiting demand during on-peak hours and operates on a 5-minute time step.

Chapter 6

Real-Time Results

Results from two separate real-time runs are considered in this chapter. The first run is from September 3, 2021 at 9:00 p.m to September 23, 2021 at 11:15 p.m. This run ended prematurely due to a system update on the application server; due to this, a complete forecasting analysis could not be performed on this 20-day run, but figures containing data of the battery scheduler response were saved prior to the restart of the application server. A separate, shorter run was performed to provide a non-continuous approximation to a month of scheduler operation. The second run ranges from September 24, 2021 at 11:30 a.m to October 1, 2021 at 11:15 a.m. Forecasting analysis is provided for the completed 7-day run, and a cost analysis is performed for the two runs combined.

Some of the graphs in this chapter contain red-ish and tan shading which represent on-peak and off-peak hours respectively. It should also be noted that the upper bound of on-peak hours is considered open. For example, if on-peak hours are from 1:00 p.m to 9:00 p.m, then 9:00 p.m is considered the beginning of off-peak hours. The lower bound of 1:00 p.m is closed and considered the beginning of on-peak hours.

6.1 Forecast

In Section 4.3, three forecasts are discussed: pf_5, pf_{15}, pf_{Opt15}. Figures 6.1 and 6.2 provide a graphical comparison of all three forecasts relative to the true building demand at a 15-minute resolution, therefore, it is the 15-minute average of pf_5 that is used for comparison. It is apparent from the graphs that pf_5 more closely matches the true demand than the other two forecasts. Recall that only the pf_{15} and pf_{Opt15} predictions made at the very start of a 15-minute bin are recorded for comparison, so it is expected that the average of pf_5 should be closer to the true demand since it is allowed to adjust within a 15-minute window.

Quantitatively, the accuracy of the forecasts is determined using five different metrics:

1. Mean Absolute Error

2. Mean Absolute Percentage Error (MAPE) [20]

3. Root Mean Square Error (RMSE)

4. Maximum Absolute Error

5. Minimum Absolute Error

With each metric, the closer its value is to zero, the more accurate the forecast. Table 6.1 shows a significant difference in accuracy between pf_5 and the other forecasts for all metrics. While the differences between pf_{15} and pf_{Opt15} are relatively less significant, pf_{Opt15} does yield more accurate results than pf_{15}, which is the desired outcome.

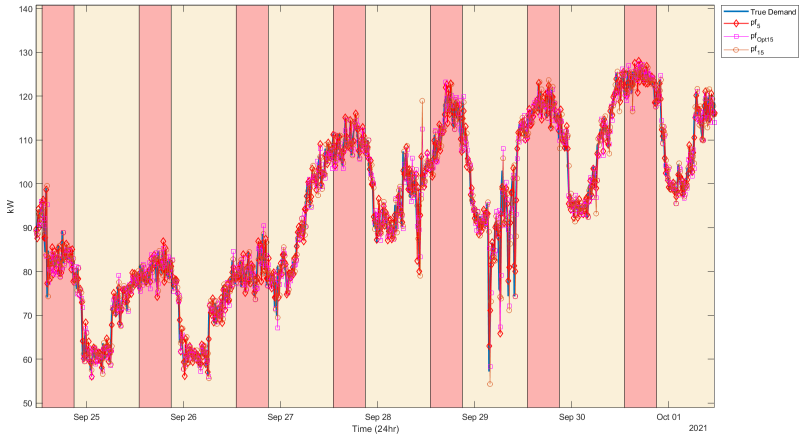

Figure 6.1. Forecasting Comparison: 7-Day Run

Figure 6.2. Zoomed Forecasting Comparison

Table 6.1. 7-Day Run Forecast Error Comparison

	Mean Abs Err	MAPE(%)	RMSE	Max Abs Err	Min Abs Err
pf_5	1.2920	1.4465	1.8067	13.866	1.0329e-3
pf_{Opt15}	2.9000	3.2353	4.2520	28.002	1.5130e-2
pf_{15}	3.1327	3.5030	4.6480	31.717	3.3532e-2

The histograms of Figure 6.3 show the forecasts' distribution of error. Error within the context of the histograms is true demand minus the respective forecast. In line with the previous results, pf_5 shows the more forecasts closer to true demand. The other forecasts are similar in their error distribution with a slight edge given to pf_{Opt15} in the number of forecast deviations closer to zero.

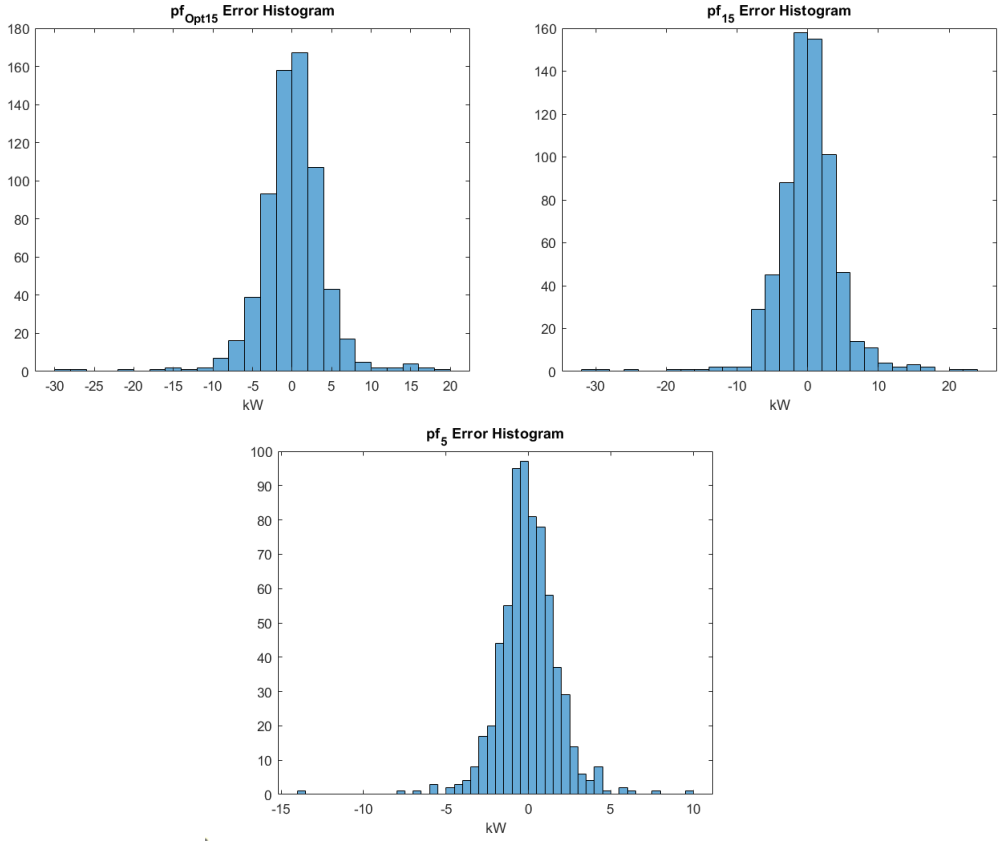

Figure 6.3. 7-Day Run Forecasting Errors

6.2 BESS Scheduling

The special modes of operation discussed in Section 5.2 are depicted in figures 6.4 and 6.5 for the 20-day and 7-day runs respectively. The modes of operation subplots may only contain values of -1, 0, and 1. A value of 0 represents normal operation as well as SOC soft limit operation. Forecast error correction mode is assigned a value of -1, and emergency control is assigned a value of 1. Each mode of operation plot is aligned with a plot depicting the true demand and pf_{Opt15} forecast; this is done to provide a reference for what is occurring with the demand and forecasting relative to specific modes.

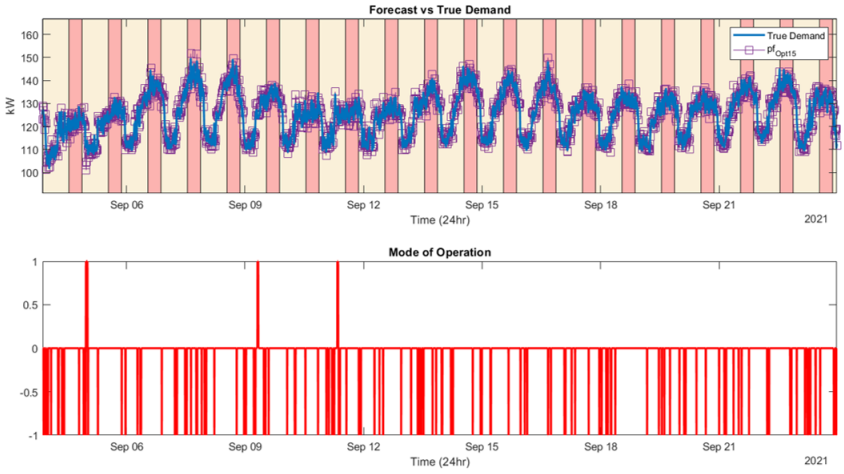

Figure 6.4. Modes of Operation: 20-day Run

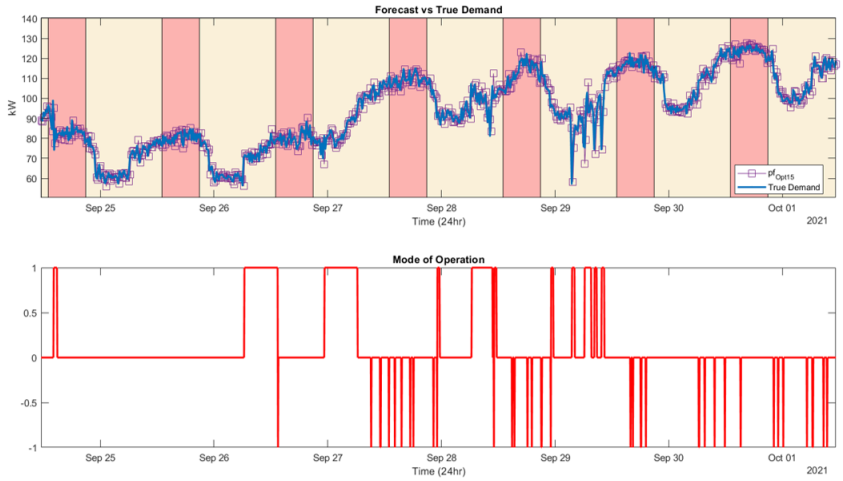

Figure 6.5. Modes of Operation: 7-day Run

Actual building demand without manipulation from the battery, NoBESS, and resulting building demand from the inclusion of the battery scheduler, MPC, are plotted together for comparison along with an SOC subplot. This power grid comparison for the 20-day run is shown in figures 6.6 and 6.7 for a battery scheduler input of pf_{Opt15} and true demand respectively. The same comparisons are given for the 7-day run in figures 6.8 and 6.9. Figures 6.6 and 6.8 show the actual resulting building demand, the real-time results, due to a battery scheduler whose input is the pf_{Opt15} forecast.

Figures 6.7 and 6.9 show the 'ideal' resulting building demand due to a battery scheduler whose input is the building demand without battery manipulation; the results in these two figures are simulated in order to compare to the real-time results in the corresponding figures 6.6 and 6.8. This case is considered ideal in the sense that there is no forecasting error introduced to the scheduler, however, the scheduler and its settings are not perfect. For example, this 'ideal' case sets both P_{Shave1} and P_{Shave2} to be equal to the maximum demand of the day of interest. While imperfect, this method provides an adequate sense of how the scheduler is improved by a more accurate forecast.

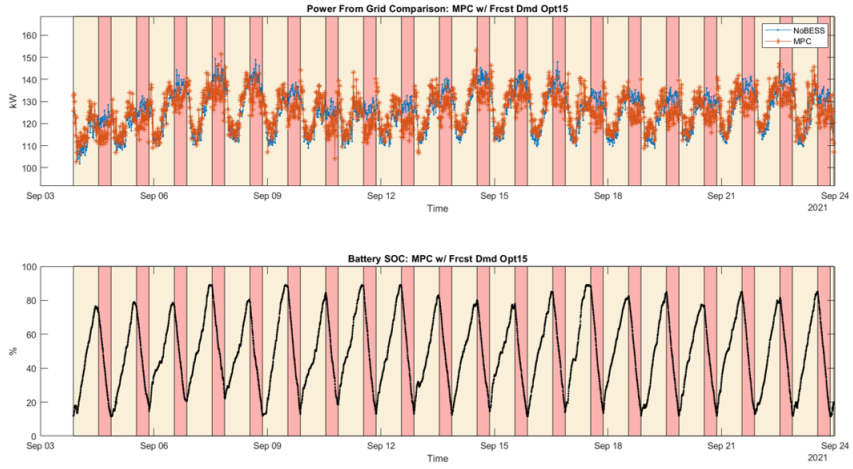

Figure 6.6. MPC Response to pf_{Opt15}: 20-Day Run

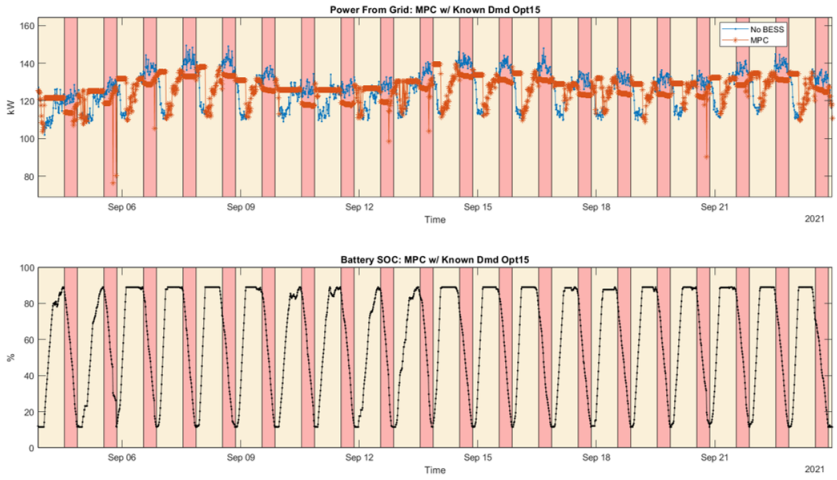

Figure 6.7. MPC Response to Ideal Forecast: 20-Day Run

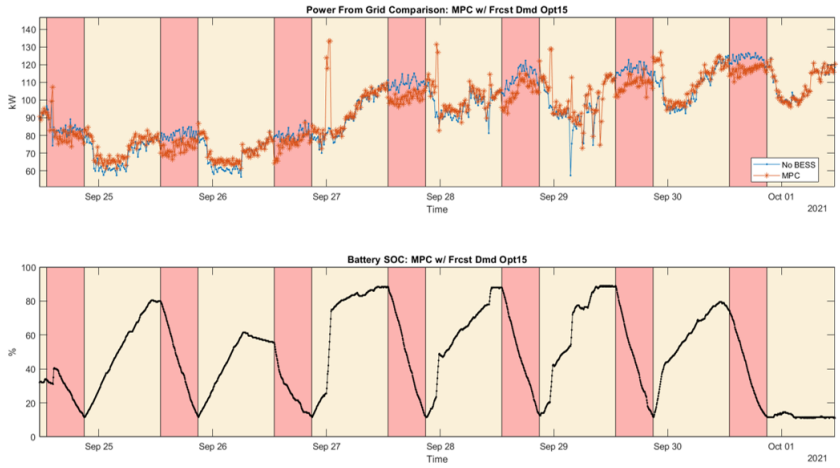

Figure 6.8. MPC Response to pf_{Opt15}: 7-Day Run

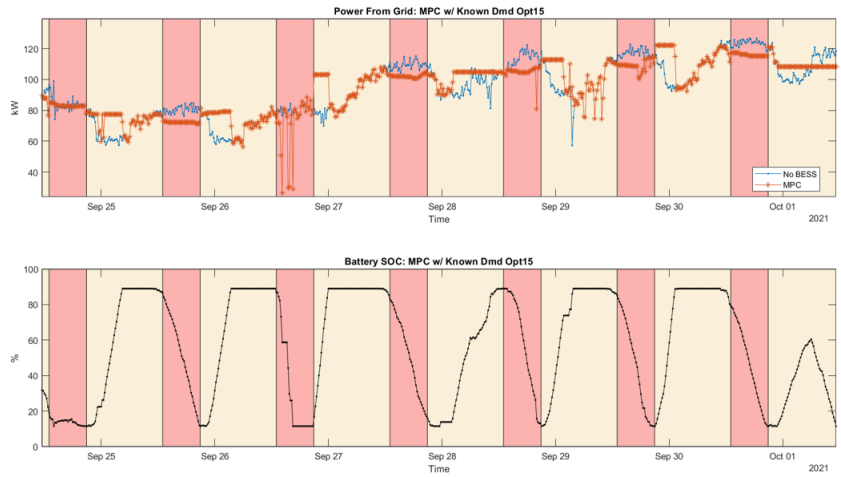

Figure 6.9. MPC Response to Ideal Forecast: 7-Day Run

6.3 Discussion

A real-time MPC-based BESS scheduler was implemented taking as input demand from a time series forecasting model. The forecasting model was designed to predict the load behavior of the ZGEC. Currently, the ZGEC is under Dominion Energy's Rate 9 [24] billing structure. Electricity use under Rate 9 is primarily based upon a standard energy consumption cost. Rate 21A [23] is a time-of-use energy and demand billing structure also offered by Dominion Energy. Should the ZGEC switch over to Rate 21A, electricity costs can be reduced with the implementation of a BESS scheduler in real-time.

The cost of the combined 20-day and 7-day runs under Rate 9 is provided in Table 6.2 for three scenarios. The first scenario considers cost of electricity without battery implementation (NoBESS), the second is the simulated case which considers the use of the battery scheduler with an ideal forecast input (MPC (Ideal)), and the last scenario is the real-time case which considers the use of the battery scheduler with the actual pf_{Opt15} forecast input (MPC (Actual)). Since the scheduler is not optimized for Rate 9, it is no surprise that the scheduler implementation does not reduce the bill under this rate.

Table 6.2. Rate 9A Cost Comparison

	Total Cost [$]
NoBESS	9535.3
MPC (Ideal)	9587.1
MPC (Actual)	9536.4

As stated previously in Section 5.1, the overall electricity cost under Rate 21A has four components: base facilities charge, time-of-use energy cost, max on-peak demand cost, and the cost associated with the greater of three positive differences. Only the positive difference between maximum off-peak demand and maximum on-peak demand is considered in the cost analysis. The other two differences are not applicable currently. One of those differences would subtract the maximum on-peak demand from 50 kVA, which is unlikely to ever be positive when considering the load behavior of the ZGEC. The other difference subtracts maximum on-peak demand from the contract demand, but that demand does not exist since the ZGEC is currently under Rate 9. Discovering the optimal contract demand is beyond the scope of this project, so the calculated Rate 21A costs shown in Table 6.3 do not entirely abide by the rate structure. Should the only applicable

difference be negative, cost associated with the maximum off-peak demand is set to zero.

Table 6.3. Rate 21A Cost Comparison

	Max On-Pk [kW—$]	Max Off-Pk [kW—$]	Energy [$]	Total Cost [$]
NoBESS	149.39—3152.8	145.97—0	4106.7	7454.6
MPC (Ideal)	134.1—2835.4	139.4—23.4	4063.5	7117.4
MPC (Actual)	151.4—3195.2	153.27—9.36	4051.1	7450.7

It is worth noting that MPC (Actual) has a cost only slightly less than NoBESS due to a greater max on-peak demand. This max demand occurred September 7, 2021 at 6:45 p.m and was caused by a corrupted persistent counter variable used by the forecaster model. This variable, after the model has run for a few days real-time, would inexplicably increment by more than one at the start of a new day. The extra increase in the counter would cause pf_{Opt15} to be not-a-number (NaN), therefore triggering a response by the model to copy the last legitimate forecast value. This response is a safety measure to ensure the model can still operate if a communication error with the Database Server occurs; it is only meant as a temporary measure until communication between the application server and database server is restored.

A remedy was applied while the model was still running real-time which consisted of using an alternate variable in an ancillary function and resetting the counter variable using MATLAB's debugging tool. Functions can only be altered during real-time operation if called using 'feval()' in the model code. Subsequent days of the 20-day run did not trigger the Database communications error response after altering the code, and Figure 6.6 shows that none of the ensuing days experienced unwarranted spikes. For the 7-day run, the persistent counter variable was replaced by a local variable found during each iteration of the model; this fix also avoided ever triggering the communications error response in the model.

6.4 Conclusion

Due to the corruption of the persistent variable in the model, the total cost of MPC (Actual) was higher than it would have otherwise been, showing less substantial savings in Table 6.4. Since the ZGEC is currently under Rate 9, the savings in Table 6.4 compare the Rate 9 electricity cost without any BESS response to the cost of each case considered for Rate 21A. Based upon this table, Rate 21A inherently provides a better cost relative to Rate 9 and implementing a real-time BESS

scheduler would offer greater savings. In the case of this non-continuous 27-day run, MPC (Ideal) would offer nearly 4% more in savings. When considering Figure 6.6 and the simulated results in Appendix B, it is reasonable to infer that MPC (Actual) should show added savings in the range of 1% to 3% for a standard demand variability month when the model operates properly.

Table 6.4. Savings Relative to Rate 9A (NoBESS)

	Cost Reduction [%]
Rate21A NoBESS	21.82
Rate21A MPC (Ideal)	25.36
Rate21A MPC (Actual)	21.86

Despite correcting the erroneous counter variable, it still impacted the remaining days due the model using the highest demand achieved during on-peak when determining the upper bound for battery charging and discharging. It is also unknown if September 7th was the only day during which the counter variable had shifted in value before being corrected, however, for the sake of approximating the expected results, Table 6.5 considers the savings if September 7th was not included when determining costs under Rate 9 and Rate 21A. The results show over 3% more in savings for the ideal forecast and over 1% for the actual forecast.

Table 6.5. Savings w/o September 7th

	Cost Reduction [%]
Rate21A NoBESS	20.35
Rate21A MPC (Ideal)	24.02
Rate21A MPC (Actual)	21.76

The scheduler is most cost effective during months that experience large spikes in demand which have a duration that does not outlast the capacity of the 50 kW, 96 kWh battery currently attached to the ZGEC. An example of this is provided in Figure 6.10, which depicts the real-time battery response during the first four days of November 2021 when spikes are encountered. Conversely, the scheduler is less cost effective during months of low demand variability such is the case for the real-time results provided previously. Forecasting is also less optimal for days whose demand is less than 100 kW as seen in Figure 6.8. The MPC spikes shown in this figure are due to emergency control being triggered by significant deviations in demand. For low demand days, the criteria for significant deviations are not as robust as they are for days with an average demand at or above 100 kW. Improving the criteria of significant deviations for low and high demand days

is a future work item to further improve the model. Other future work includes improving upon the predictor methods and full range forecast algorithm, improving the overall BESS scheduler algorithm, determining the optimal contract demand under Rate 21A, incorporating a building model with weather input, and integrating an HVAC optimizer.

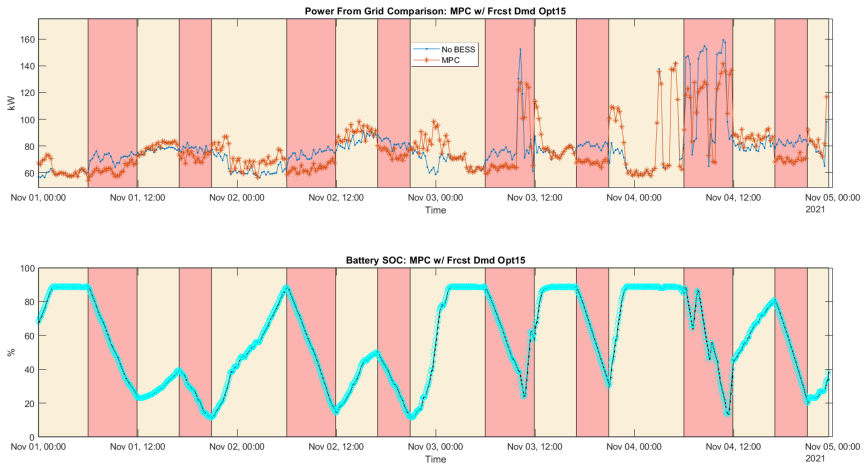

Figure 6.10. MPC Response to Actual Forecast: 4-Day Run

Appendices

Appendix A Hardware & Software Details

Hardware			
Application Server	DELL PowerEdge R240: 16 GB RAM, 1 TB Memory		
Database Server			
Battery	LG Chem JH3-R800 50kW/96kWh		
Inverter	Rhombus RES-BESS50 DC: 450-825V AC: 480V, 60 Hz, WYE 50 kW		
Power Meters	PowerLogic Series 800 (PM800)	PSL PQube 3	EIG Shark 100
Intermediate Devices	Lantronix UDS1100	RS485/232 Converter	Energy Storage Integration Platform
Application Server Software			
Matlab 2019b	Instrument Control Toolbox Parallel Computing Toolbox MATLAB Report Generator Statistics \& Machine Learning Toolbox		
Matlab 2020b	Simulink Simulink Compiler Simulink Coder Simulink Desktop Real-Time Statistics \& Machine Learning Toolbox Optimization Toolbox Instrument Control Toolbox Matlab Coder MATLAB Compiler MATLAB Support for MinGW-w64 C/C++ Compiler Real-time Kernel Installed		
Lantronix CPR Manager	Version 4.3.0.3		

Appendix B Simulation Results

This appendix displays the simulated, not real-time, results from the forecaster and battery scheduler. There are slight differences between the simulated and real-time models. The simulated model has SOC bounds of 90% & 10% and does not account for the gradual loss of SOC due to the inverter being connected. Other distinctions also exist between the models as a result of the real-time model communicating with the ESIP and sharing file access with the DBMS.

The simulated model operates on recorded demand data of past days. Upon initialization, a forecast is made without knowing any of the actual demand for the day of interest. Each subsequent step of the model assumes five minutes has passed and grabs the corresponding 5-minute demand of that day. The remaining demand data for the time of interest is still unknown to the model in order to mimic what the real-time model would experience. Other than data retrieval, the simulated model provides an approximate depiction of the real-time model's algorithm. Subsections below show the results of simulating a specific month. It should be noted that all dates before the year 2021 are based on 15-minute data taken from a Dominion Energy power meter. This data was converted to a 5-minute resolution as described in Section 4.1.

Some of the following graphs are in regard to forecasts discussed in Section 4.3 but are labeled differently. For clarification, '5to15' is the 15-minute average of pf_5, 'RawPF15' represents pf_15, and pf_Opt15 is equivalent to 'OptPF15.'

B.1 January 2020

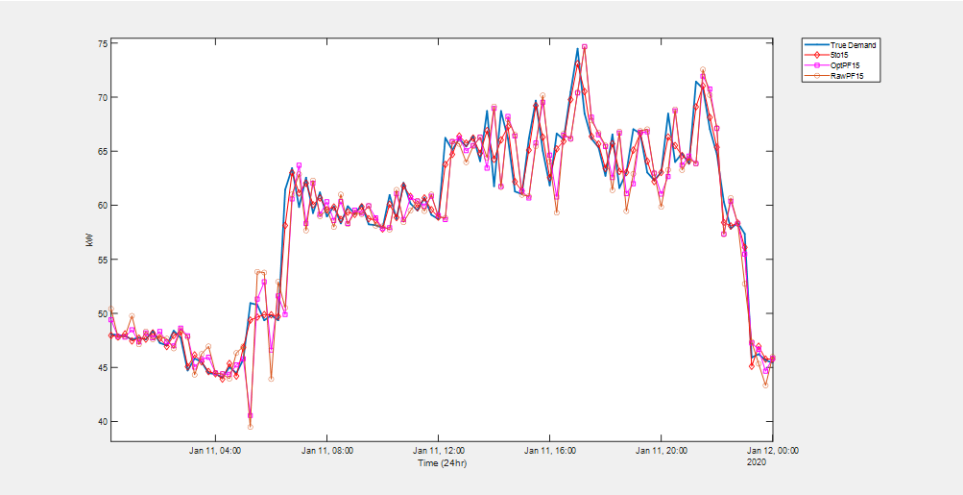

Figure 11. Jan 2020: Standard Deviation Day

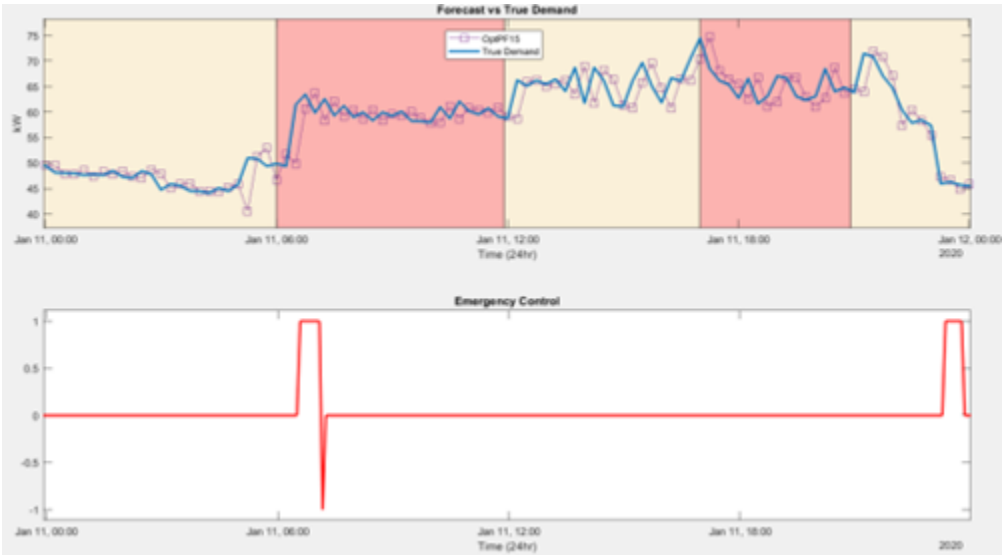

Figure 12. Jan 2020: Standard Deviation Day Modes

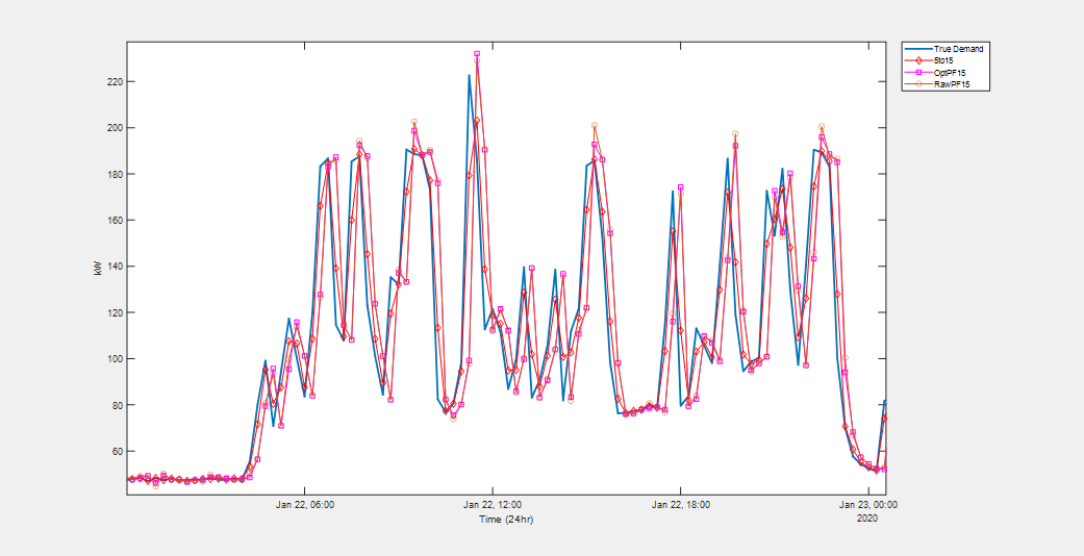

Figure 13. Jan 2020: Unpredictable Deviation Day

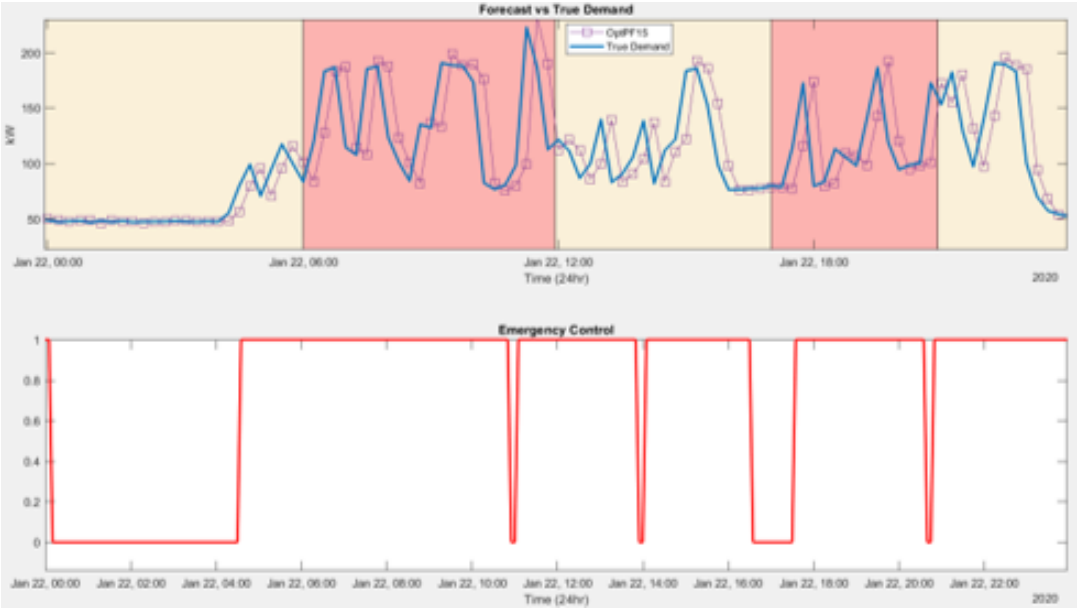

Figure 14. Jan 2020: Unpredictable Deviation Day Modes

Table 6. Jan 2020: Forecasting Error Comparison

	Mean Abs Err	MAPE(%)	RMSE	Max Abs Err	Min Abs Err
5t015	2.7485	3.0005	6.4682	43.346	2.8481e-5
$OptPF_{15}$	8.9985	9.8550	20.593	123.48	2.9465e-4
$RawPF_15$	9.2381	10.206	20.844	124.81	1.7491e-3

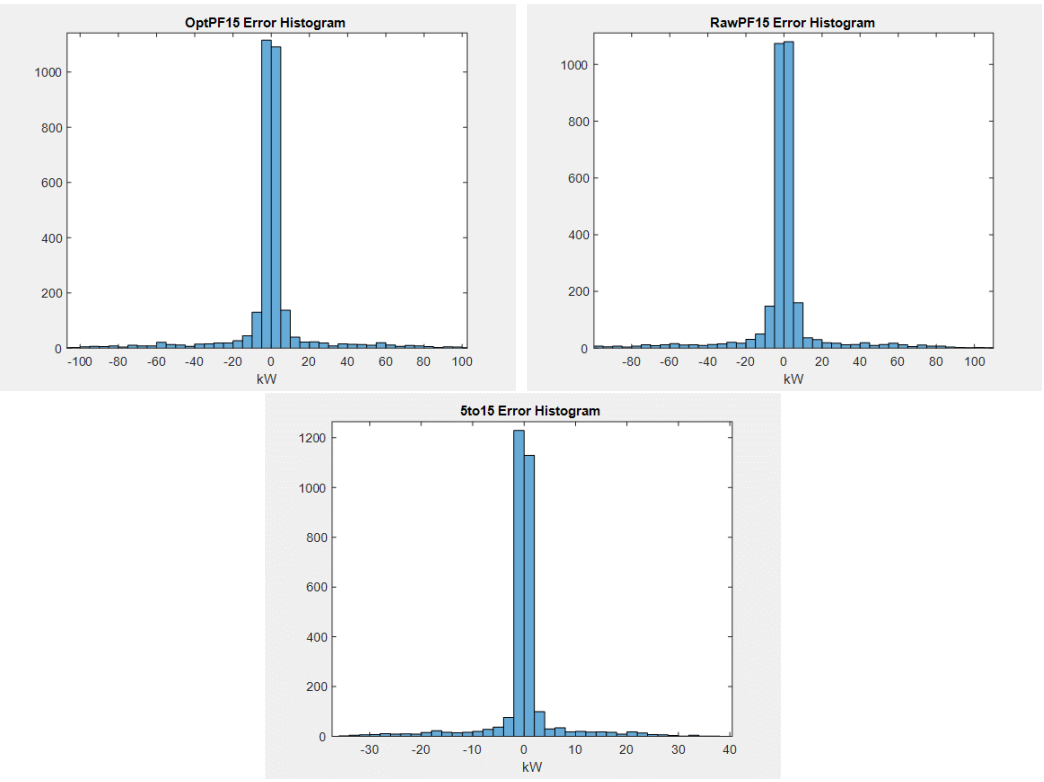

Figure 15. Jan 2020: Forecasting Errors

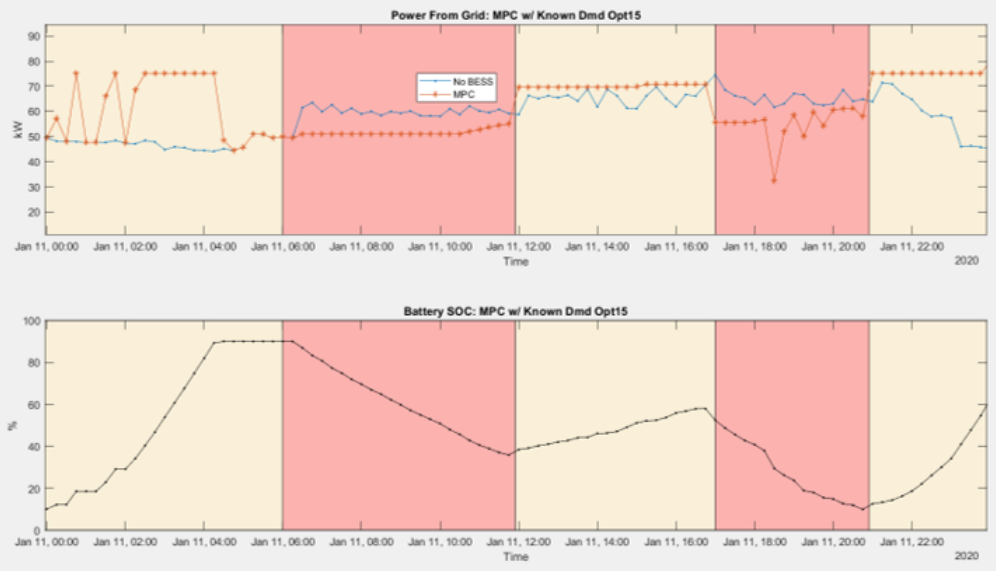

Figure 16. Jan 2020: MPC Response to Perfect Forecast (Standard Day)

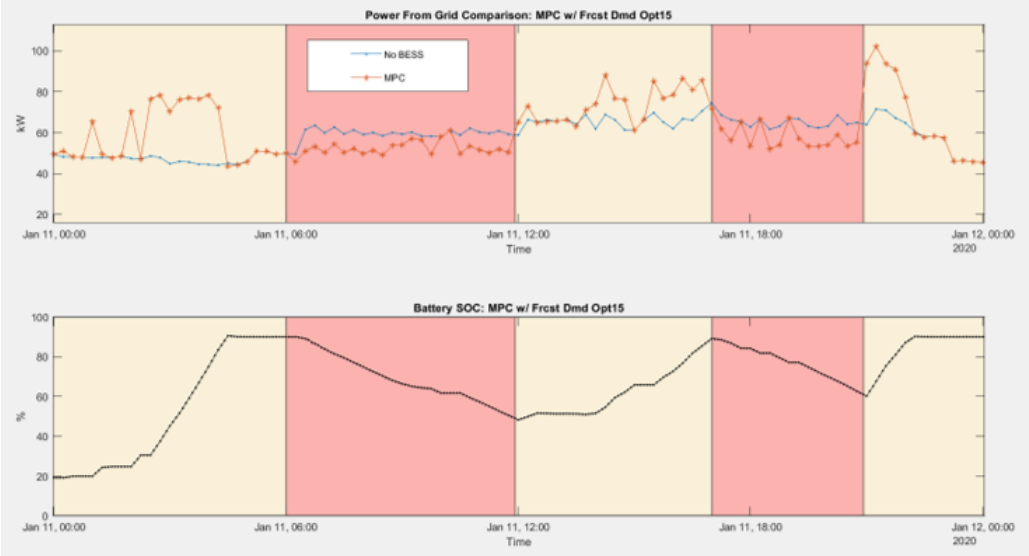

Figure 17. Jan 2020: MPC Response to Actual Forecast (Standard Day)

Table 7. Jan 2020: Rate 21A Cost Comparison

	Max On-Pk [kW—$]	Max Off-Pk [kW—$]	Energy [$]	Total Cost [$]
No BESS	22.64—2829.9	202.08—0	2602.7	5627.5
MPC (Ideal)	184.49—2335	191.2—32.76	2616	5178.7
MPC (Actual)	192.7—2449.2	200.88—37.44	2578.8	5260.4

Table 8. Jan 2020: Rate 9A Cost Comparison

	Total Cost [$]
No BESS	6282.4
MPC (Ideal)	6373.6
MPC (Actual)	6276.5

Table 9. Savings Relative to Rate 9A No BESS: Jan 2020

	Cost Reduction [%]
Rate21A NoBESS	10.42
Rate21A MPC (Ideal)	17.57
Rate21A MPC (Actual)	16.27

B.2 May 2020

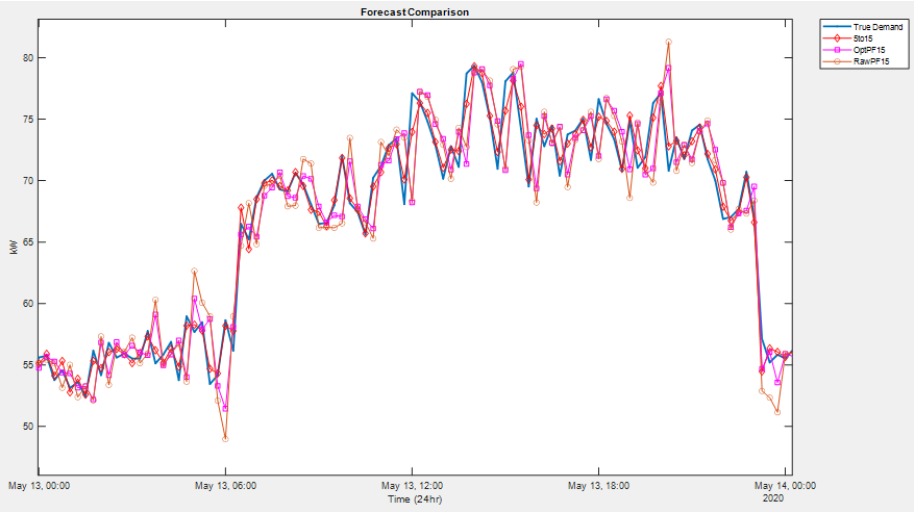

Figure 18. May 2020: Standard Deviation Day

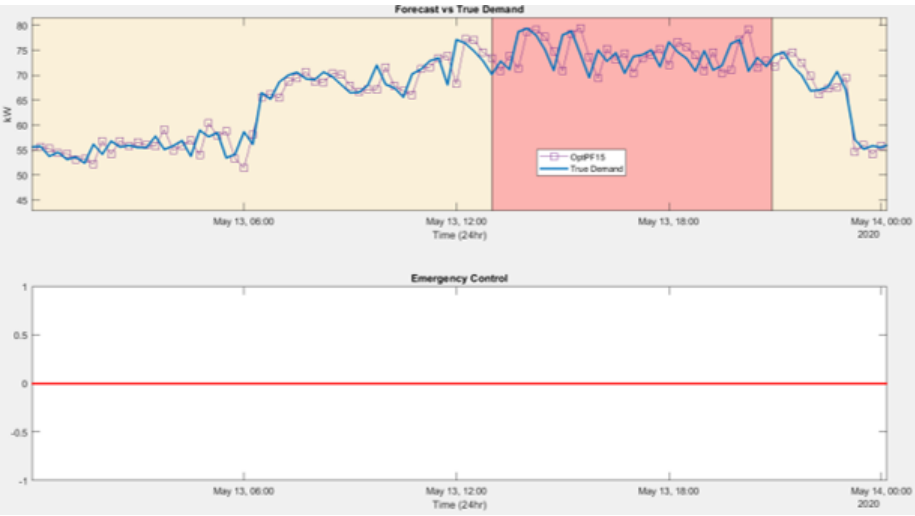

Figure 19. May 2020: Standard Deviation Day Modes

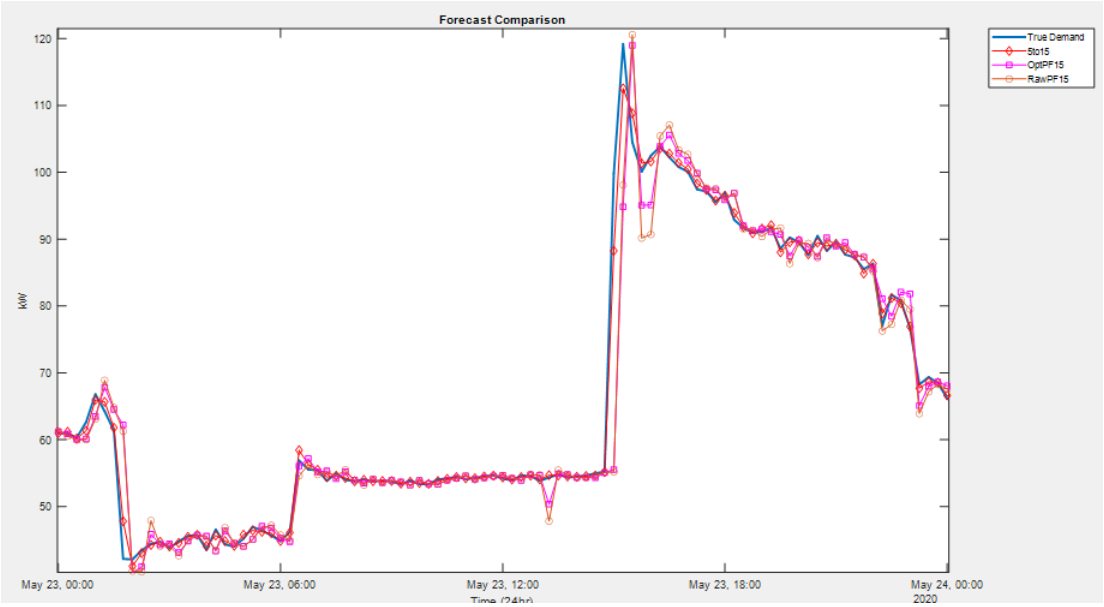

Figure 20. May 2020: Unpredictable Deviation Day

Figure 21. May 2020: Unpredictable Deviation Day Modes

Table 10. May 2020: Forecasting Error Comparison

	Mean Abs Err	MAPE(%)	RMSE	Max Abs Err	Min Abs Err
5t015	0.83816	1.1803	1.1736	11.582	0
$OptPF_{15}$	2.7463	3.8568	3.8041	44.314	2.8677e-3
$RawPF_15$	2.936	4.1355	4.0303	44.683	8.1161e-4

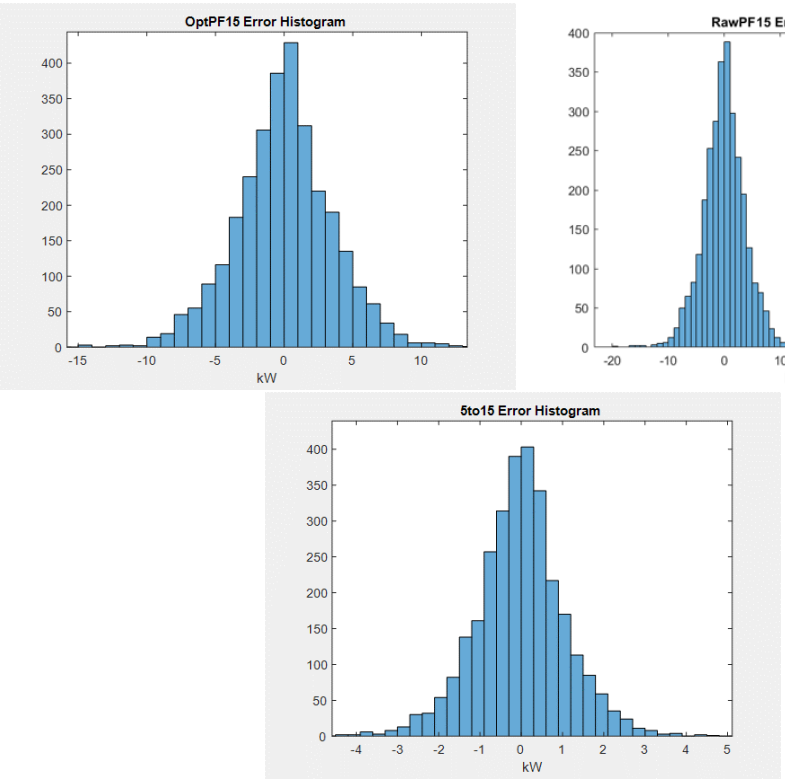

Figure 22. May 2020: Forecasting Errors

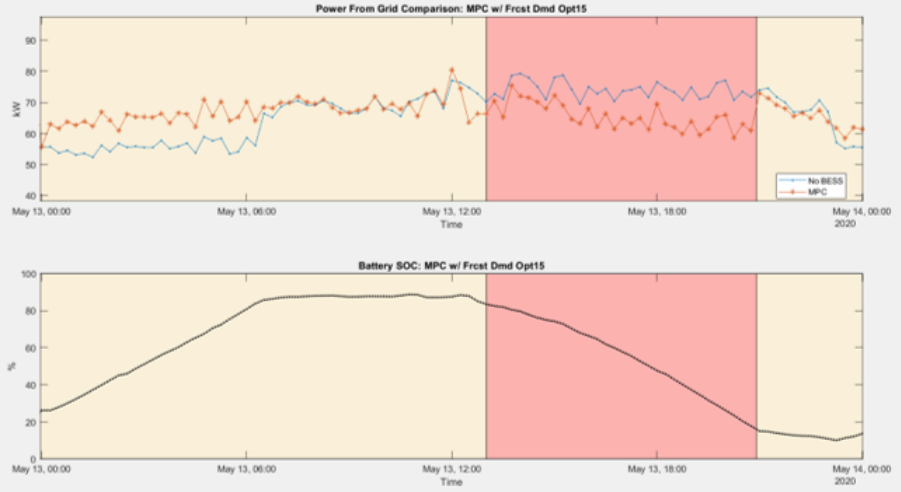

Figure 23. May 2020: MPC Response to Perfect Forecast (Standard Day)

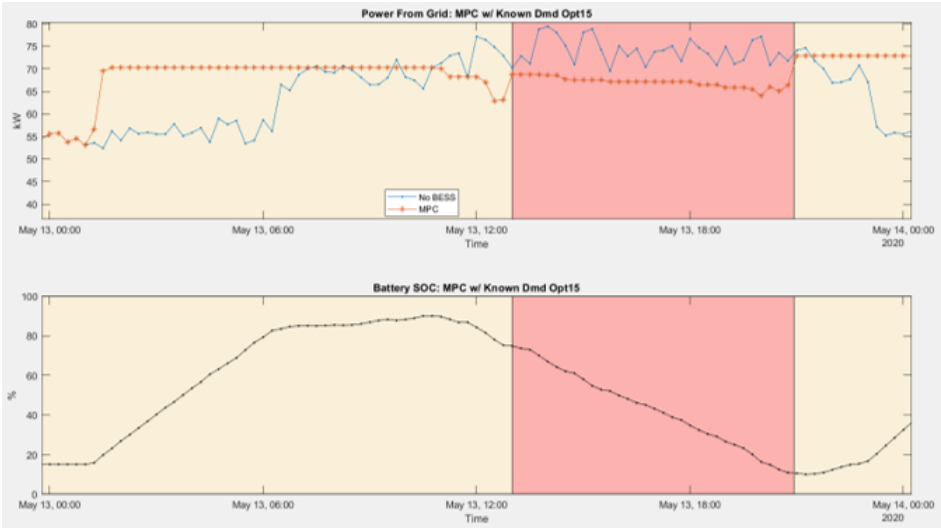

Figure 24. May 2020: MPC Response to Actual Forecast (Standard Day)

Table 11. May 2020: Rate 21A Cost Comparison

	Max On-Pk [kW—$]	Max Off-Pk [kW—$]	Energy [$]	Total Cost [$]
No BESS	119.2—1510.1	110.72—0	2345.4	4050.5
MPC (Ideal)	93.887—1192.9	98.24—18.72	2344.5	3751.1
MPC (Actual)	102.68—1307.1	113.24—51.48	2322	3875.6

Table 12. May 2020: Rate 9A Cost Comparison

	Total Cost [$]
No BESS	5807.1
MPC (Ideal)	5854.2
MPC (Actual)	5798.4

Table 13. Savings Relative to Rate 9A No BESS: May 2020

	Cost Reduction [%]
Rate21A NoBESS	30.25
Rate21A MPC (Ideal)	35.40
Rate21A MPC (Actual)	33.26

B.3 July 2020

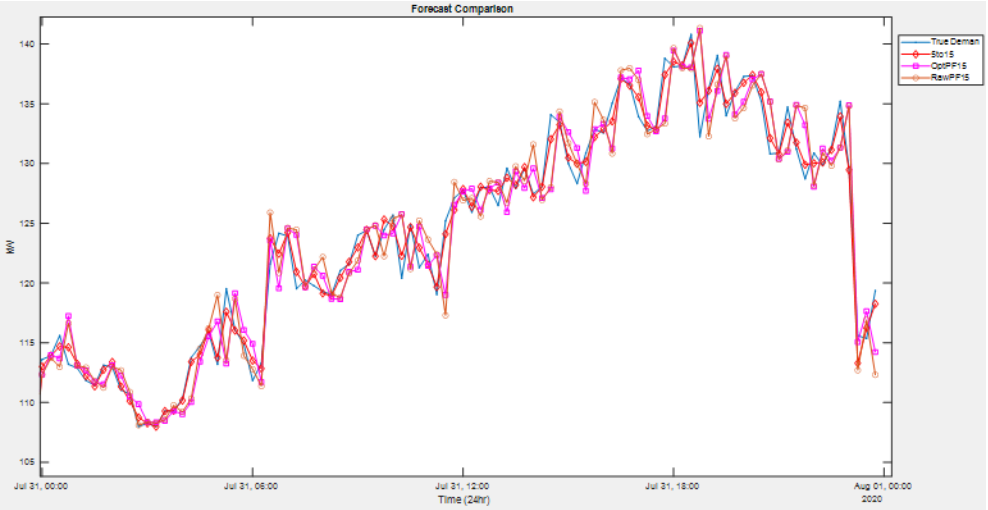

Figure 25. July 2020: Standard Deviation Day

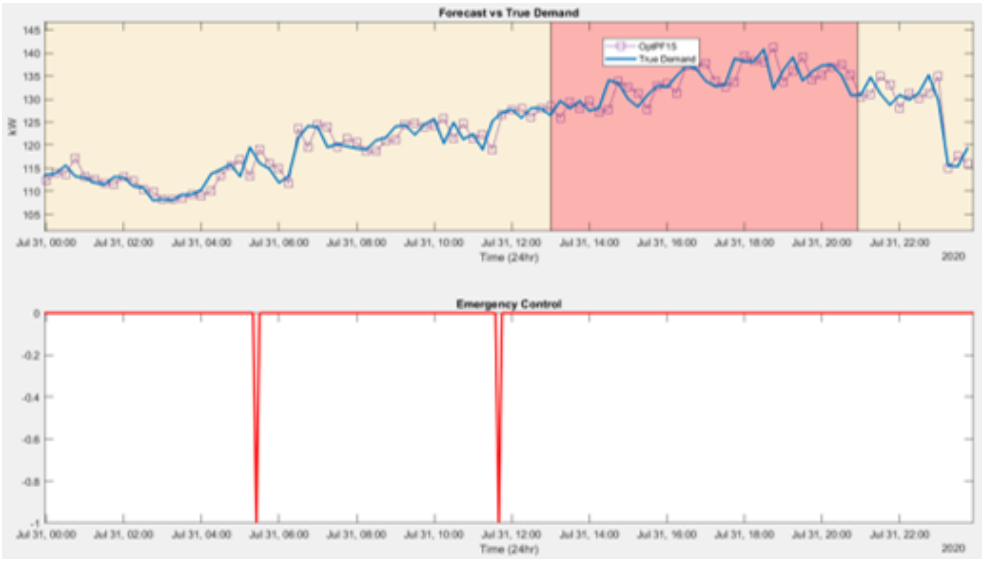

Figure 26. July 2020: Standard Deviation Day Modes

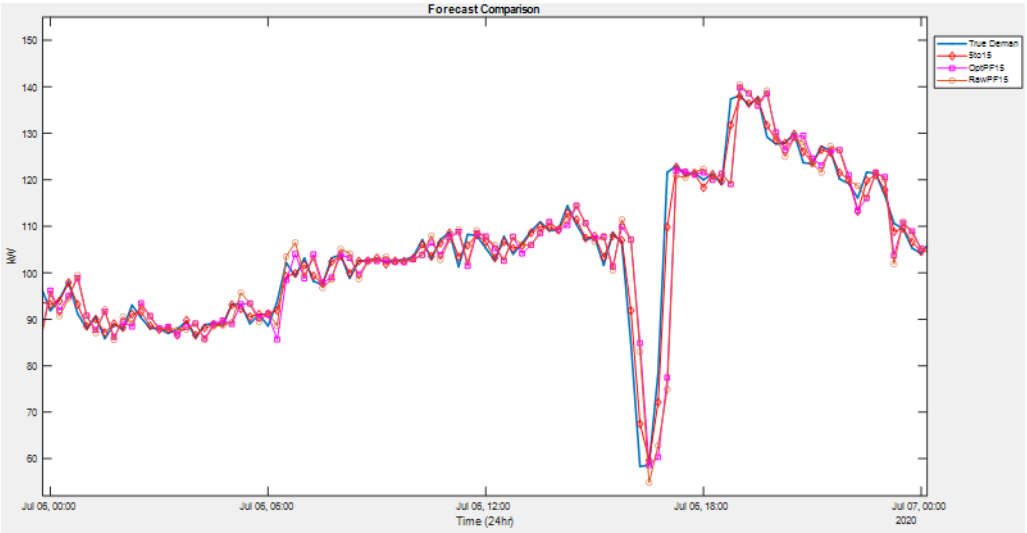

Figure 27. July 2020: Unpredictable Deviation Day

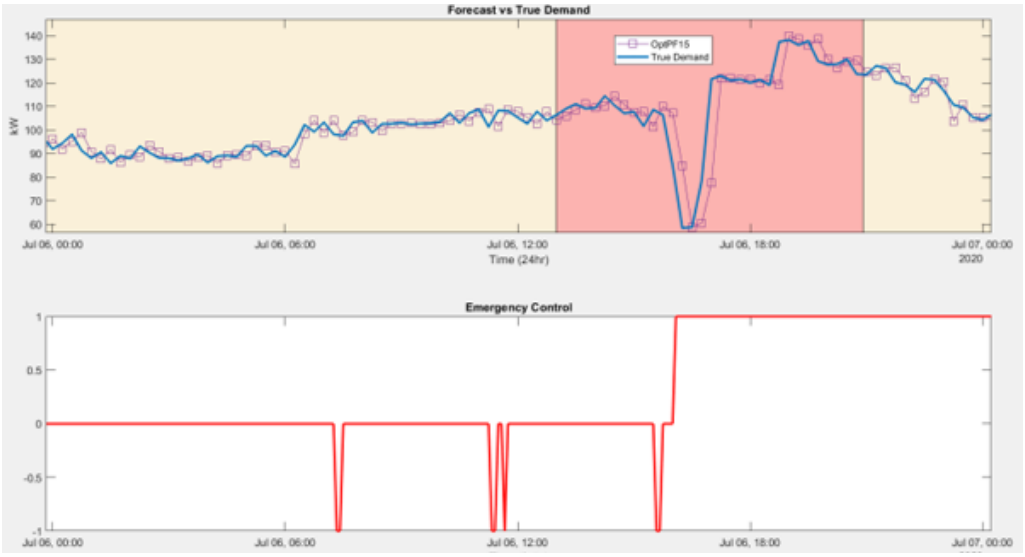

Figure 28. July 2020: Unpredictable Deviation Day Modes

Table 14. July 2020: Forecasting Error Comparison

	Mean Abs Err	MAPE(%)	RMSE	Max Abs Err	Min Abs Err
5t015	0.81831	0.71885	1.2172	19.058	1.3106e-4
$OptPF_{15}$	2.6293	2.3096	3.8365	55.686	3.4116e-4
$RawPF_15$	2.7887	4.4472	4.0269	54.539	5.1571e-4

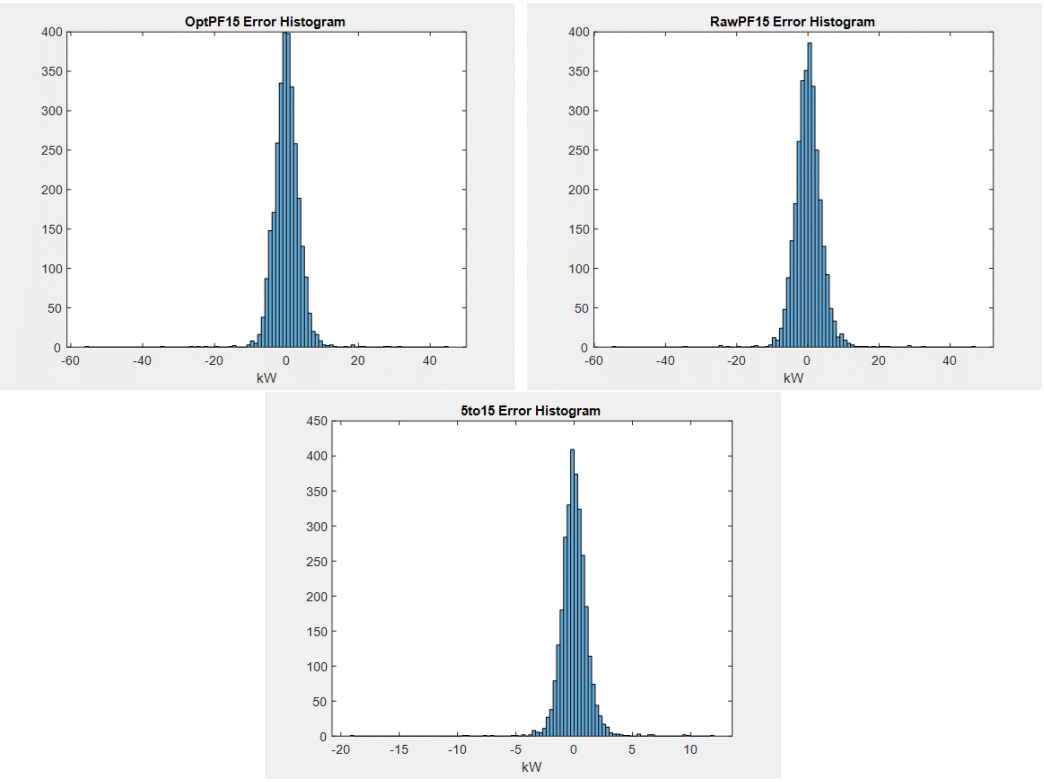

Figure 29. July 2020: Forecasting Errors

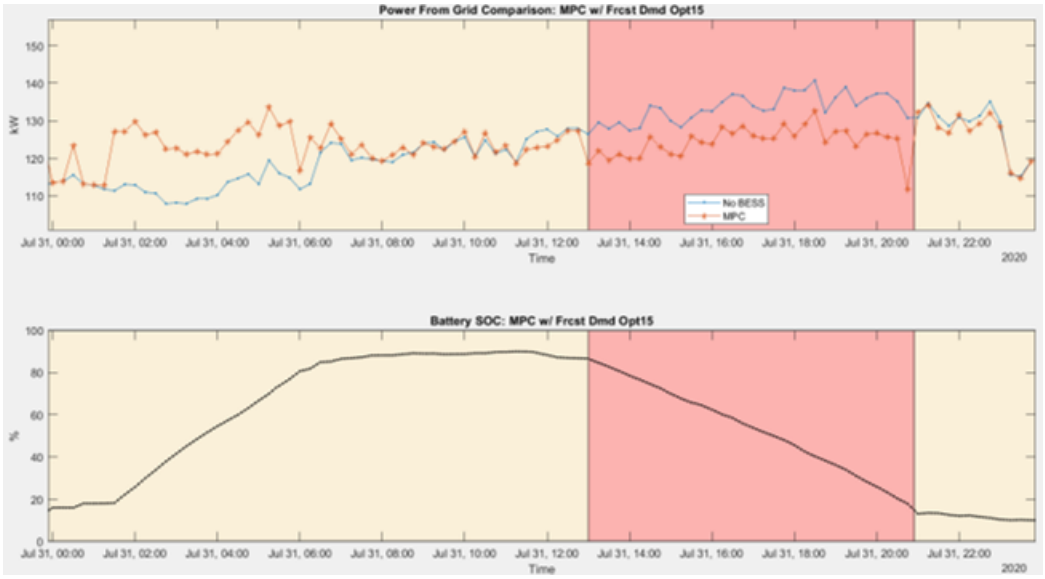

Figure 30. July 2020: MPC Response to Perfect Forecast (Standard Day)

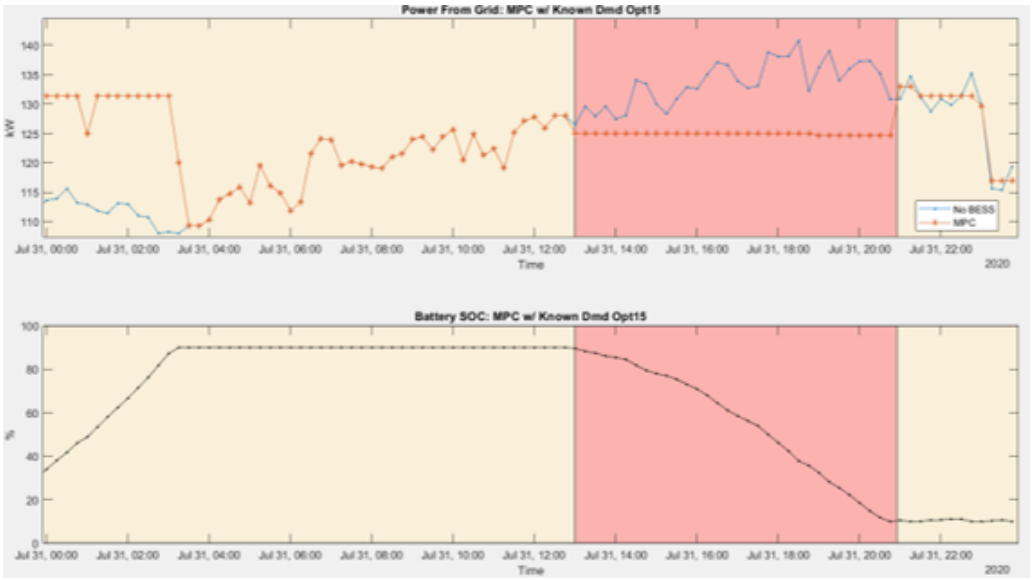

Figure 31. July 2020: MPC Response to Actual Forecast (Standard Day)

Table 15. July 2020: Rate 21A Cost Comparison

	Max On-Pk [kW—$]	Max Off-Pk [kW—$]	Energy [$]	Total Cost [$]
No BESS	140.8—2983.6	140.96—0	4673.4	7851.9
MPC (Ideal)	126.41—2666.2	136.74—46.8	4616.3	7524.2
MPC (Actual)	135.12—2856.6	161.25—121.68	4558.4	7761.7

Table 16. July 2020: Rate 9A Cost Comparison

	Total Cost [$]
No BESS	10859
MPC (Ideal)	10913
MPC (Actual)	10849

Table 17. Savings Relative to Rate 9A No BESS: July 2020

	Cost Reduction [%]
Rate21A NoBESS	27.69
Rate21A MPC (Ideal)	30.71
Rate21A MPC (Actual)	28.52

B.4 April 2021

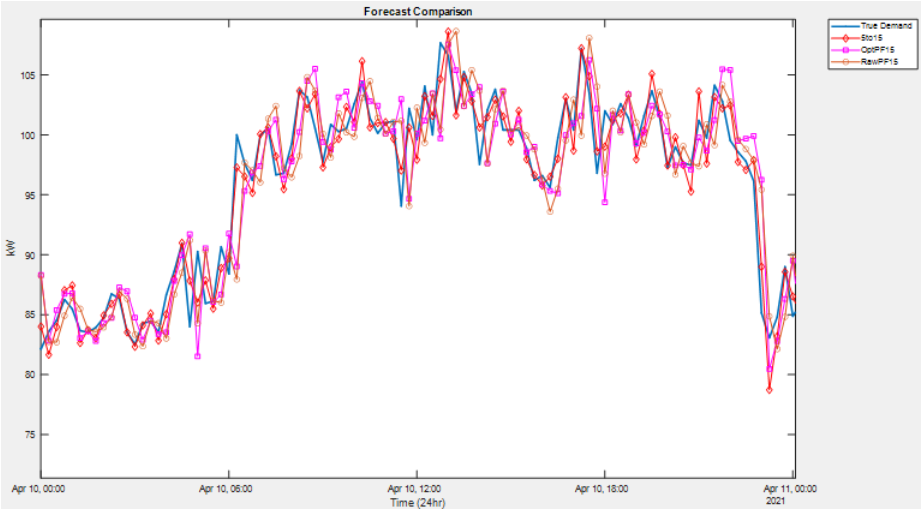

Figure 32. April 2021: Standard Deviation Day

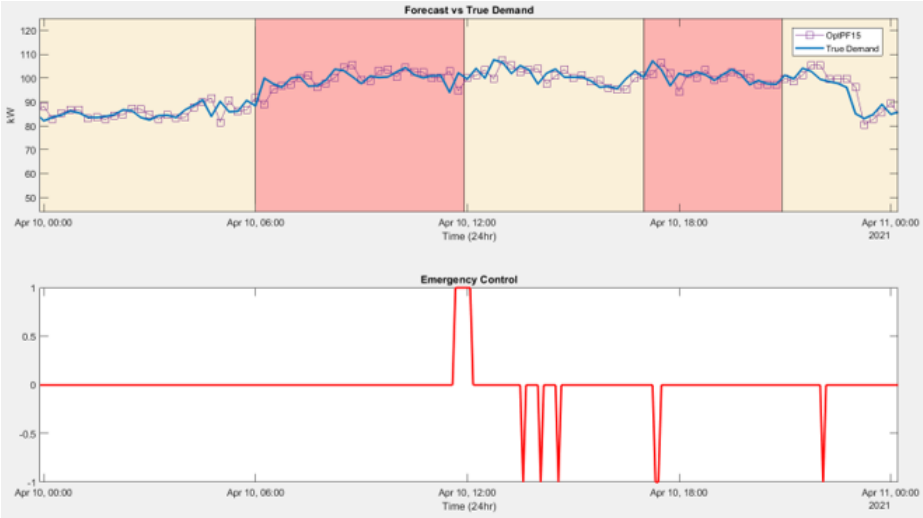

Figure 33. April 2021: Standard Deviation Day Modes

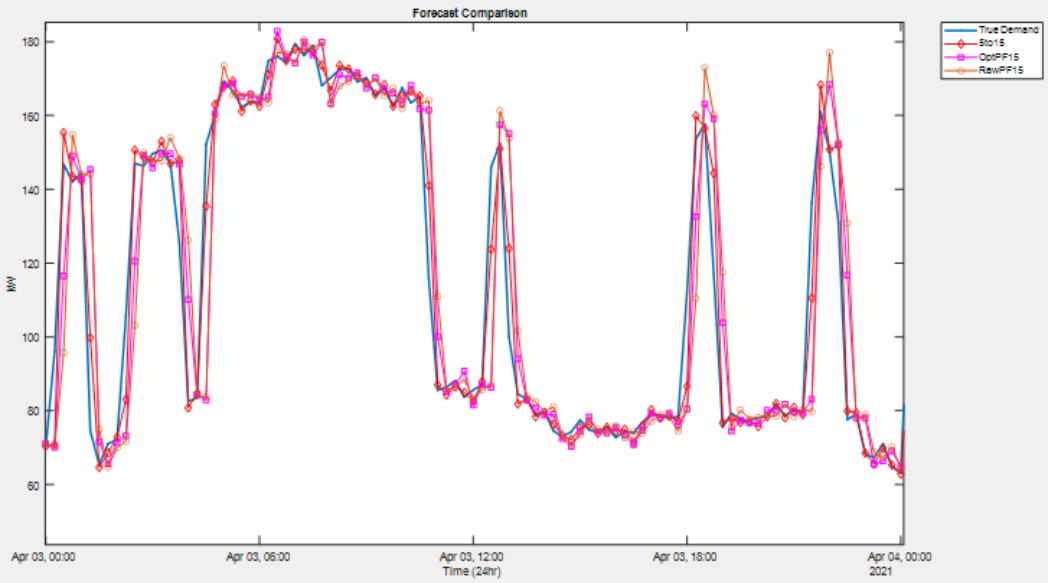

Figure 34. April 2021: Unpredictable Deviation Day

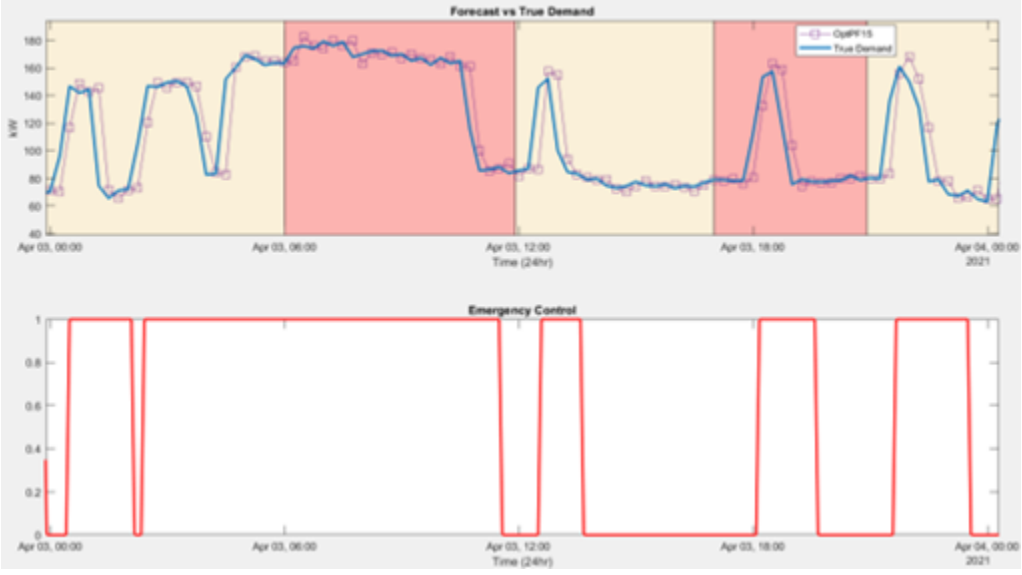

Figure 35. April 2021: Unpredictable Deviation Day Modes

Table 18. April 2021: Forecasting Error Comparison

	Mean Abs Err	MAPE(%)	RMSE	Max Abs Err	Min Abs Err
5t015	1.7019	1.9394	3.4986	27.679	1.31e-3
$OptPF_{15}$	3.7435	4.2745	7.8771	74.36	6.5417e-4
$RawPF_15$	4.0156	4.5563	8.5717	73.871	1.3364e-4

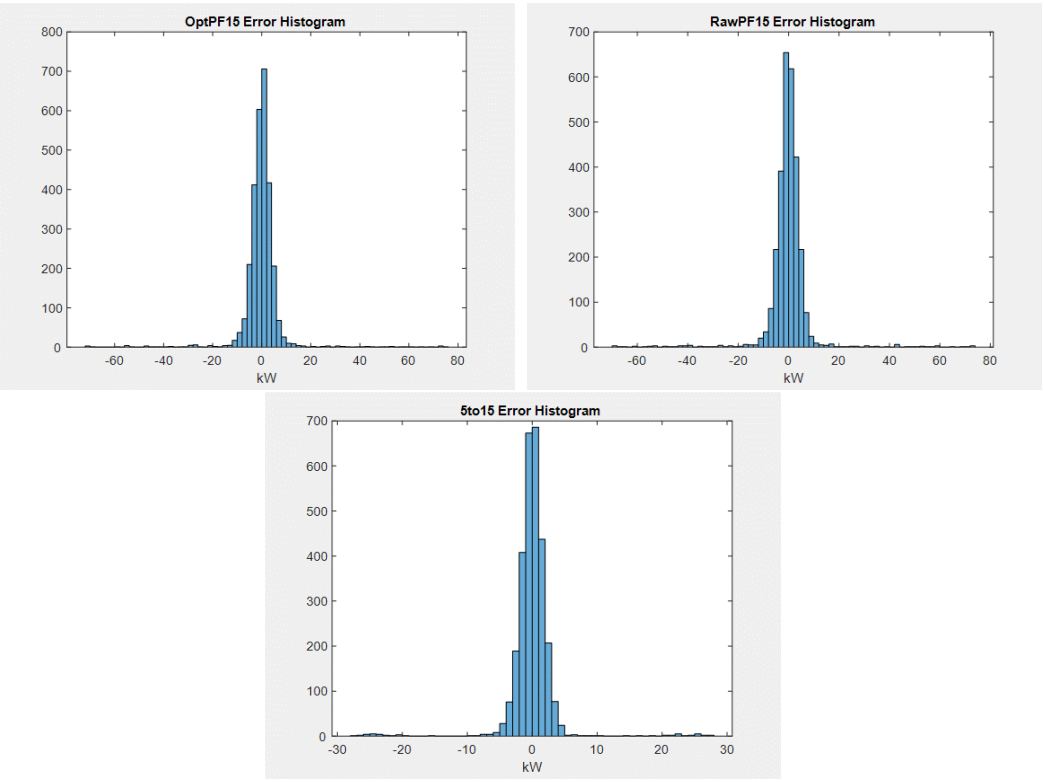

Figure 36. April 2021: Forecasting Errors

Figure 37. April 2021: MPC Response to Perfect Forecast (Standard Day)

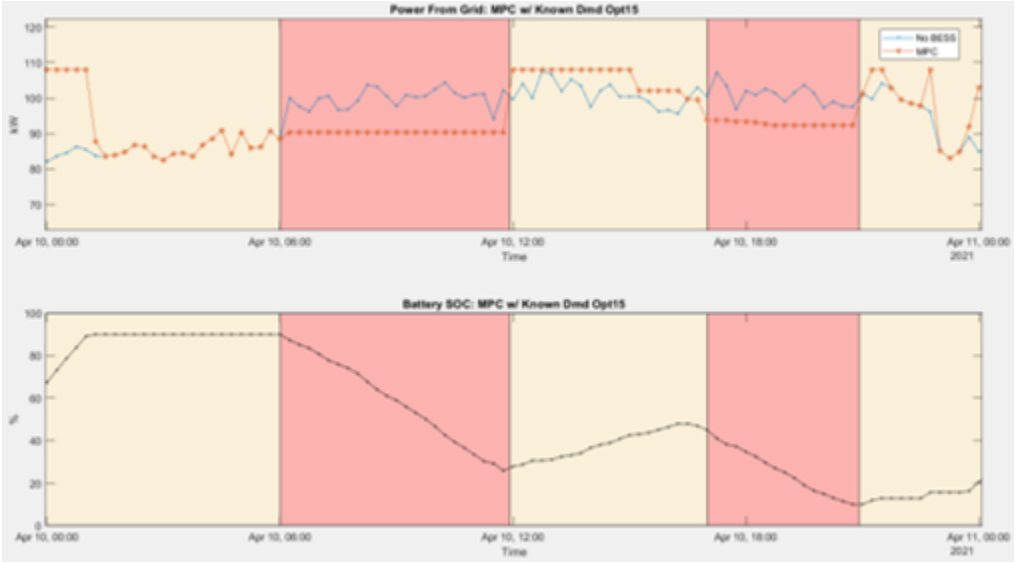

Figure 38. April 2021: MPC Response to Actual Forecast (Standard Day)

Table 19. April 2021: Rate 21A Cost Comparison

	Max On-Pk [kW—$]	Max Off-Pk [kW—$]	Energy [$]	Total Cost [$]
No BESS	181.07—2296.9	170.87—0	2757.4	5249.3
MPC (Ideal)	170.82—2170	141.77—0	2761.1	5126.1
MPC (Actual)	168.83—2144.6	170.87—9.36	2726.9	5075.9

Table 20. April 2021: Rate 9A Cost Comparison

	Total Cost [$]
No BESS	6701.6
MPC (Ideal)	6776.2
MPC (Actual)	6693.9

Table 21. Savings Relative to Rate 9A No BESS: April 2021

	Cost Reduction [%]
Rate21A NoBESS	21.67
Rate21A MPC (Ideal)	23.51
Rate21A MPC (Actual)	24.26

Appendix C Miscellaneous

C.1 Private & Callback Functions

The private functions of the GUI are private in the sense that no outside applications are allowed to call them, however, they are accessible throughout the code within the application. These functions primarily handle the independent autonomous actions of the DBMS and certain graphical updates of the GUI, but they can also be called in response to user specific interactions.

Callback functions are called solely in response to user specific interactions and are not responsible for any independent autonomous actions.

C.2 Model Implementation

Within the dotted blue line is the implemented model. Other components are to be added in future work.

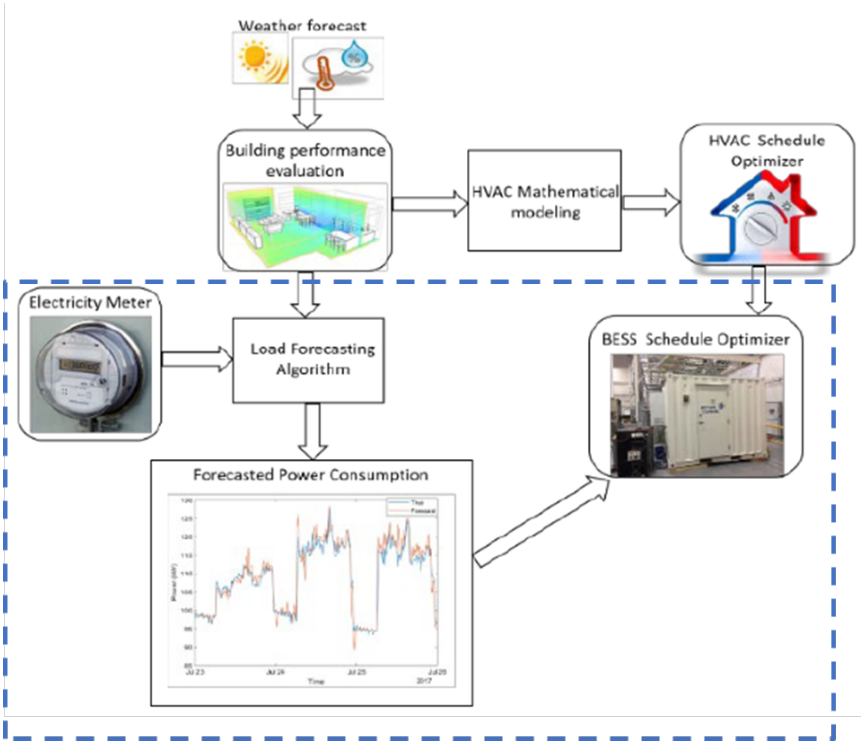

Figure 39. Model Implementation

C.3 Network Diagram

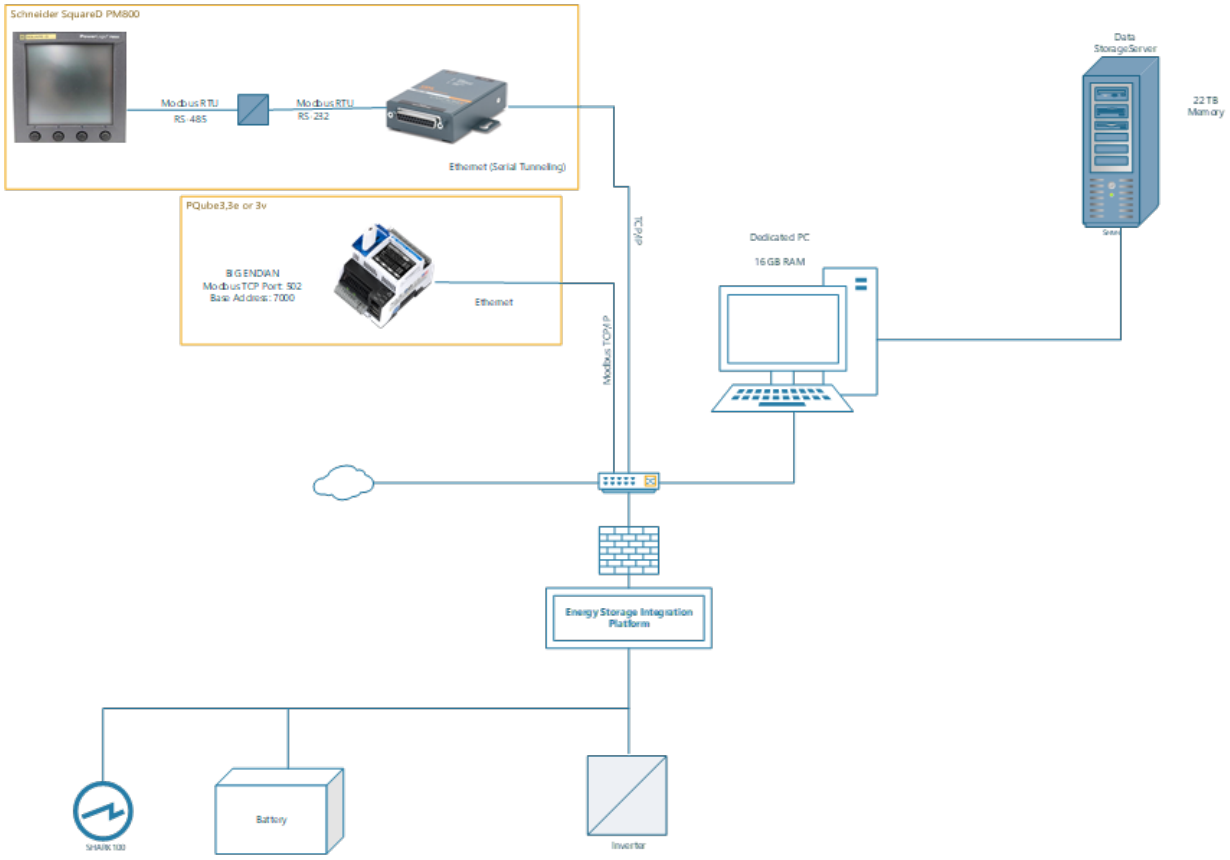

Figure 40. Layout of Hardware

C.4 Simulink Model

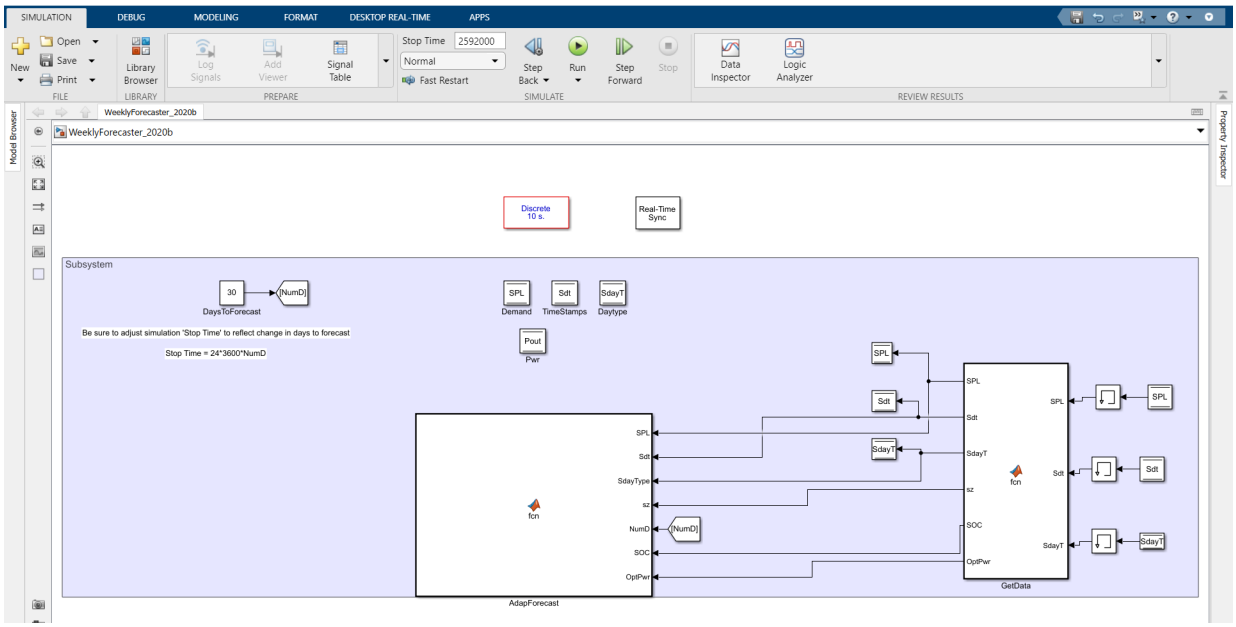

Figure 41. Simulink Model Set for Real-Time Execution in Normal Mode

Bibliography

[1] C. Reilly. *Demand-Response in the United States: Expansion Efforts and Electricity Market Activities*. Nova Science Publishers, 2014.

[2] N. Fernandez and et al. Energy savings potential from improved building controls for the us commercial building sector. *Energy Efficiency*, 11(2):393–413, 2018.

[3] A. Nguyen and et al. Optimal strategies of siting, sizing, and scheduling of bess: Voltage management solution for future lv network. *IEEJ Transactions on Electrical and Electronic Engineering*, 14(5):694–704, 2019.

[4] Z. Wang and Y. He. Two-stage optimal demand response with battery energy storage systems. *IET Generation, Transmission & Distribution*, 10(5):1286–1293, 2016.

[5] U. Nair and et al. An analysis of multi objective energy scheduling in pv-bess system under prediction uncertainty. *IEEE Transactions on Energy Conversion*, 36(3):2276–2286, 2021.

[6] C. Lu and et al. Optimal sizing and control of battery energy storage system for peak load shaving. *Energies*, 7(12):8396–8410, 2014.

[7] B. Ashima B. and V. Bansal. *Database Management System*. Alpha Science International Ltd., 2015.

[8] Modbus application protocol specification, 7 2018. `https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf`.

[9] Modbus over serial line specification and implementation guide, 12 2006. `https://modbus.org/docs/Modbus_over_serial_line_V1_02.pdf`.

[10] Modbus messaging on tcp/ip implementation guide, 10 2006. `https://modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf`.

[11] Device server user guide, 2020. `https://www.lantronix.com/wp-content/uploads/pdf/UDS1100_UG.pdf`.

[12] Com port redirector overview, 2006. `https://www.lantronix.com/products/com-port-redirector/#tab-overview`.

[13] Enable com port-based applications to communicate over a network to equipment around the globe software application, 2006. `https://www.lantronix.com/wp-content/uploads/pdf/Com-Port-Redirector_PB.pdf`.

[14] D. Ali and et al. Long-term load forecast modelling using a fuzzy logic approach. *Pacific Science Review A: Natural Science and Engineering*, 18(5):123–127, 2016.

[15] M. Azhar and et al. Building electrical energy consumption forecasting analysis using conventional and artificial intelligence methods: A review. *Renewable and Sustainable Energy Reviews*, 70(December):1108–1118, 2016.

[16] C. Deb and et al. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74(March):902–924, 2017.

[17] T. Hong and S. Fan. Probabilistic electric load forecasting: A tutorial review. *International Journal of Forecasting*, 32(3):914–938, 2016.

[18] K. Debnath and et al. Forecasting methods in energy planning models. *Renewable and Sustainable Energy Reviews*, 88(March):297–325, 2018.

[19] C. Kuster and et al. Electrical load forecasting models: A critical systematic review. *Sustainable Cities and Society*, 35(June):257–270, 2017.

[20] D. Vedullapalli. Power management system in a building with battery and hvac for demand response applications, 8 2020.

[21] L. Suganthi and A. Samuel. Energy models for demand forecasting: A review. *Renewable and Sustainable Energy Reviews*, 16(2):1223–12240, 2012.

[22] M. Capuno and et al. Very short-term load forecasting using hybrid algebraic prediction and support vector regression, 2017. `https://doi.org/10.1155/2017/8298531`.

[23] Dominion energy electricity rate 21a time-of-use demand general service, 2021. `https://www.dominionenergy.com/south-carolina/rates-and-tariffs`.

[24] Dominion energy electricity rate 9 general service, 2021. `https://www.dominionenergy.com/south-carolina/rates-and-tariffs`.