Spring 6-13-2018

Electric Water Heater Modeling for Distributed Energy Resource Aggregation and Control

Anne E. Clarke
Portland State University

Let us know how access to this document benefits you.

Follow this and additional works at: https://pdxscholar.library.pdx.edu/open_access_etds

Part of the Electrical and Computer Engineering Commons

Recommended Citation

10.15760/etd.6333

This Thesis is brought to you for free and open access. It has been accepted for inclusion in Dissertations and Theses by an authorized administrator of PDXScholar. For more information, please contact pdxscholar@pdx.edu.
Abstract

Today’s utilities face new challenges due to the continually increasing penetration of residential solar and other distributed, stochastic generation sources. In order to maintain balance and stability in the grid without building costly, large-scale generation plants, utilities are turning to distributed energy resources for use in demand response programs. Demand response is a cost-efficient way to balance grid load/generation without the need for increased capital investment in traditional generation resources. Demand response programs are excellent exploiters of end-user devices that otherwise further accentuate the daily load curve and thus, add to the difficulties created by daily load peaks.

Electric water heaters are excellent candidates for use in demand response programs for a variety of reason. One, electric water heaters represent a large portion of daily household loads due to their high nominal power ratings (1.5 kW - 5.5kW), and frequent use estimated to account for approximately one third of all daily residential power demand [1]. Two, they are composed of strictly resistive elements, which greatly simplifies modeling, aggregation and control. And third, they can be used for load "shedding" during periods of high electrical demand as well as load "absorbing" during periods of excess generation due to their thermal storage capabilities.

With improved access and control, electric water heaters could become a major distributed energy resource for utilities. In order to properly control and use a distributed
energy resource, it is important to know how these resources operate and their patterns of behavior in different environments. This thesis presents a single-element, single mass electric water heater model for analyzing the effectiveness of using electric water heaters as distributed energy resources and for participation in demand response programs. The CTA-2045 communication protocol was used for testing demand response signals. The electric water heater is modeled in Python and the communication pathway was built in C++ and Python.
This thesis is dedicated to a man of tremendous integrity: Dr. Bob Bass. And to my mom and dad, thank you for being my biggest fans.
Acknowledgements

I wouldn’t be the student I am today without the efforts of Dr. Bob Bass and Dr. Doug Hall. You both have believed in me beyond what I thought I deserved. You have supported my desire to become an engineer and have encouraged me to keep learning and reaching for my dreams. You are champions for women in STEM and you have been a champion for me. Thank you.

I would also like to thank the Power Lab team: Crystal, Tylor, Kevin, Leighton and Jamie. Crystal, you are my hero and a gift of a friend. Tylor, I can’t imagine my engineering life without you. You’re my favorite lab partner and a wonderful person. Kevin, you always told me where to find free food on campus and generously shared all your weird life wisdom with me. Leighton, thank you for being my water heater guy and for your unyielding willingness to help. Jamie, thank you for your levity and for keeping me young.

And finally, I would like to thank my parents. You showed me love and gentleness on the weekends when I would come home stressed and grumpy. You kept me focused on the present task and never let me lose myself in the hustle of graduate life.
# Contents

## Abstract i

## Dedication iii

## Acknowledgements iv

## List of Tables vii

## List of Figures viii

## 1 Introduction 1

1.1 Demand Side Management ........................................ 1  
1.2 Impact of Solar Penetration ...................................... 2  
1.3 Electric Water Heaters as DERs .................................. 3  
1.4 DER Modeling ....................................................... 7  
1.5 Communication for DERs .......................................... 8  
1.6 Problem Statement ................................................ 11  
1.6.1 Time Scale ..................................................... 12  
1.6.2 Usage Profiles ................................................. 13  
1.6.3 EWH Size and Manufacturer .................................. 13  
1.6.4 Device Parameters ............................................. 14  
1.6.5 Model Precision ............................................... 14

## 2 Literature Review 15

2.1 Dual Element modeling ............................................. 15  
2.2 Single Element, Multi-Zone Modeling ............................ 16  
2.3 Models with Unknown Usage Profiles ............................ 17  
2.4 Defined Model Criteria .......................................... 17  
2.5 Objectives of Work ................................................ 19

## 3 Design Considerations and Methodology 20

3.1 Electric Water Heater Modeling .................................. 20  
3.1.1 Electric Water Heater Operation ............................. 20  
3.1.2 CTA-2045 Definitions ......................................... 23  
3.1.3 EWH Model Parameters ....................................... 26
3.1.4 EWH Model Flow and Operation ........................................... 31
3.1.5 Usage Profiles ................................................................. 33
3.1.6 EWH Model Temperature and Energy Take Calculations ........... 38
3.1.7 Stand-by Loss and Water Draw Functions .............................. 42
   3.1.7.1 EWH Model Assumptions ........................................... 45
3.1.8 Output Information and Plotting ........................................ 46
3.2 Physical Electric Water Heater Design and Instrumentation ........... 48
   3.2.1 Installation and Data Collection ....................................... 49

4 Results 53
   4.1 Heat-up Validation ............................................................ 54
   4.2 Stand-by Loss Validation ..................................................... 56
   4.3 Water Draw Validation ....................................................... 57
   4.4 Usage Profile Validation: Temperature ................................. 59
   4.5 Usage Profile Testing: Energy Take ..................................... 63
   4.6 Shed Command Testing ..................................................... 64

5 Discussion 68

6 Conclusion 72

Bibliography 75

Appendix A: Appendix 79
   A.1 EWH Emulator Python Script ........................................... 79
   A.2 Physical EWH Control Python Script ................................... 94
List of Tables

<table>
<thead>
<tr>
<th></th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>EWH Model Operating States</td>
<td>30</td>
</tr>
<tr>
<td>3.2</td>
<td>Configuration File: Default Values</td>
<td>31</td>
</tr>
<tr>
<td>3.3</td>
<td>Hot Water Demand by Event Type for U.S. Residential Home [2]</td>
<td>34</td>
</tr>
<tr>
<td>3.4</td>
<td>Hot Water Demand by Event Type for Residential Home in Sweden [2]</td>
<td>34</td>
</tr>
<tr>
<td>3.5</td>
<td>Nomenclature for Energy Flow and Temperature Calculations</td>
<td>39</td>
</tr>
<tr>
<td>3.6</td>
<td>Nomenclature for Stand-by and Water Draw Losses</td>
<td>42</td>
</tr>
<tr>
<td>3.7</td>
<td>EWH Simulation: Example Configuration File</td>
<td>47</td>
</tr>
<tr>
<td>3.8</td>
<td>EWH Simulation: Example Configuration File</td>
<td>49</td>
</tr>
<tr>
<td>3.9</td>
<td>Physical EWH Installation and Testing Equipment</td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>EWH Usage Profile [3]</td>
<td>54</td>
</tr>
<tr>
<td>5.1</td>
<td>Usage Profile Test: Temperature and Energy Take Accuracy</td>
<td>70</td>
</tr>
</tbody>
</table>
# List of Figures

1.1 Accentuated “Duck curve” from Solar Generation [4] ............................. 3  
1.2 Daily Residential Load Profile for Total Demand and Water Heater Demand [5] 4  
1.3 CTA-2045 Connections with an EWH [6] ........................................ 9  
1.4 CTA-2045 Communication Design [6] ............................................. 9  

3.2 Dual Element Electric Water Heater Control Circuit [8] .......................... 23  
3.4 Electric Water Heater Model Control Flow Diagram. .......................... 32  
3.5 Residential electric water heater daily usage profile [3] .......................... 35  
3.8 EWH Emulator: Output Log File ..................................................... 47  
3.9 EWH Emulator: Output Log File ..................................................... 48  
3.10 Electric water heater installation and set-up. ...................................... 49  
3.11 Physical EWH Set-up ................................................................. 52  

4.1 Heat-up Cycle: Emulation vs. Physical Water Heater .......................... 55  
4.2 Ambient Loss Cycle: Emulation vs. Physical Water Heater .................... 56  
4.3 Water Draw Validation Plot ............................................................. 58  
4.4 Usage Profile Temperature Plot: Emulation (1:60 seconds) vs. Physical Water Heater ................................................................. 59  
4.5 Usage Profile Temperature Plot: Physical Water Heater Data from Four Sensors 60  
4.6 Usage Profile Temperature Plot: Emulation (1:120 seconds) vs. Physical Water Heater ................................................................. 61  
4.7 Usage Profile Temperature Plot: Emulation (1:240 seconds) vs. Physical Water Heater ................................................................. 62  
4.8 Usage Profile Energy Take Plot: Emulation (1:60 seconds) vs. Physical Water Heater ................................................................. 63  
4.9 Shed Command Test: Emulation (1:120 seconds) ............................... 65  
4.10 Shed Command Test: Emulation (1:120 seconds) with Energy Take .......... 66
1 Introduction

Many states within the United States are required to or interested in expanding their renewable energy resources due to new Renewable Portfolio Standard (RPS) requirements. The state of Oregon’s RPS requires that 50% of all demand be supplied by renewable energy resources by 2040 [9]. Utilities can accomplish this by increasing the amount of renewable generation sources and increasing the amount of utility-controlled distributed energy resources (DERs) such as electric water heaters (EWHs). This shift away from carbon-emitting, on-demand generation sources leaves utilities with fewer dispatchable assets that can respond to daily load spikes or unexpected grid disturbances.

1.1 Demand Side Management

As stochastic resources continue to increase in power capacity and quantity, utilities are presented with new challenges in maintaining a reliable and stable grid and can no longer rely on purely supply-side management programs. To minimize costs, reduce losses and increase power system stability, many utilities are turning to demand side management (DSM) programs. DSM programs have controlled access to end-use devices (i.e. specific customer loads) and can use them to respond to certain grid conditions in accordance to customer agreements and device limitations [10]. DSM programs increase a utility’s reserve of controllable assets, allowing for load-shifting, valley-filling, and load following
control schedules to be implemented while minimizing effects on customer lifestyle and comfort [11], and reducing customer electricity bills.

The participation of EWHs in DSM programs is well researched and with the deployment of the smart grid, the opportunity for utilities to gain access to usage data and control of these assets is easier than ever. There are two main types of DSM control programs: direct and indirect load control [11]. In indirect load control, customers manually set the power parameters of the DER; the utility can not directly control the device. This paper focuses on direct load control where the utility has direct control over the power consumption of the DER with little operational input from the customer during DR events. Depending on the DER type and manufacturer, control limitations may be placed by the utility in order to maintain the efficiency ratings of the device.

1.2 Impact of Solar Penetration

Estimations in the U.S. from 2016 report 19.5 TWh of electric energy was produced by distributed, small-scale rooftop solar [12], where small-scale solar is defined as a system with a generating capacity up to one megawatt. The main difficulty with increased residential solar is the stochastic nature of its generating schedule, much of which peaks during midday (10am - 3pm; regionally dependent). This influx of stochastic power generation complicates the process of developing a balanced generation/load schedule along with added difficulties in maintaining grid stability [7]. The accentuated "duck curve" due to solar generation can be seen in Figure 1.1.
1.3 Electric Water Heaters as DERs

It is estimated that the installation of solar, coupled with residential batteries, will continue to increase through 2021 [13]. This increasing penetration of residential photovoltaic and other distributed generation sources has lead to an increase in demand for access to and control over DERs that can be used to help mitigate the effect of increased solar without increased capital investment in traditional generation sources (e.g. peaker plants) [7]. EWHs are excellent candidates for mitigating these new challenges because their daily load profile closely follows that of the aggregated daily residential load curve. As shown in Figure 1.2, the peaks and valleys of total residential demand mimic the demand curve of EWHs [1, 5]. Due to these correlated load curves, adjusting residential EWH usage directly affects the
overall shape of the total daily load curve. EWHs used for manufacturing and industry have different curves that do not follow the average daily load curve [5] and therefore can add complexity when used for DSM programs.

![Figure 1.2: Daily Residential Load Profile for Total Demand and Water Heater Demand [5]](image)

Depending on the type of asset, DERs can be used to provide a variety of grid services. EWHs can provide services including but not limited to peak shaving, valley filling, load following, increasing reserve capacity, voltage support, frequency regulation and other fast responding ancillary services [11]. A single EWH will not affect the overall load curve, but through aggregation, utilities will gain the ability to control large groups of EWHs, thereby increasing their capacity for load shifting and other ancillary services.

DERs can be any range of devices from electric vehicle (EV) chargers, heat pumps, HVAC systems, solar panels, small scale wind farms (i.e. wind turbines with a generating
capacity of less than 50 kW) and EWHs. EWHs are ideal for participating in DSM programs for several reasons.

1. EWHs are abundant grid assets. One study reported that EWHs account for approximately 30% of all residential loads [1] making them large components in the daily load curve. In 2016, an estimated 3.94 million EWHs were shipped to the U.S. [14]. Each of these EWHs is a possible DER that can be exercised with utility control.

2. EWHs have high nominal power consumption levels as compared to other residential appliances. Small-capacity residential systems have a power rating of around 1500 W, with more moderately-sized systems having a power rating of around 4500 W - 5500 W. This is a valuable attribute for DERs as it minimizes the number of DERs that need to take part in a DR event for a given desired outcome [2]. By aggregating large numbers of these EWHs, utilities could have at their disposal hundreds of megawatts and megawatt-hours of shift-able load.

3. EWHs are primarily resistive loads, therefore, when used as DERs, there is no need for reactive power support. This makes using these assets in DSM programs a much simpler, more manageable, and predictable process. Other DERs, like battery storage systems, require a more advanced control system than the simple on/off requirements of an EWH due to the reactive component presented by the device inverter.

4. And finally, EWHs can be used to "shed" load but also can act as energy storage units and "absorb" load. EWHs have thermal storage capability that can be exploited as
a load-shifting technique (e.g. heat EWH tank during peak solar generation, and use during times of high demand and low generation). This is an especially useful quality as more photovoltaic generation is added to the grid. These dual response capabilities are provided by turning off EWHs (i.e. "shedding" load) and through using the hot water in the tank as thermal storage (i.e. "absorbing" load). By aggregating and controlling large groups of EWHs, utilities will have a virtual power plant (VPP) of DERs at their disposal for grid stabilization and support.

The main disadvantage to using EWHs in DSM programs is that their behavior is difficult to predict. The state and capacity of an EWH is primarily dependent on the usage of hot water in the home. Hot water usage is stochastic and inconsistent and thus difficult to define. Because this research is focused on building an EWH model for aggregation purposes, the impact of the stochastic nature of usage will be reduced due to aggregating large numbers of these assets. Controlling EWHs in a group allows for more tolerance and flexibility for the state of individual devices and emphasizes statistical group averaging over extreme accuracy of specific assets.

In order for utilities to effectively use EWHs for demand response (DR), utilities need to be able to aggregate and control a large number of devices simultaneously. To do this properly, they need the ability to accurately predict the state and available capacities for each connected device or group of similarly behaving devices, preferably without constant communication. This is where models play an important role.

---

1 A virtual power plant is a collection of DERs that are combined and controlled as a single entity [15]
1.4 DER Modeling

Models can be used to virtually mimic the physical behavior of a DER. Models are valuable resources because they allow for a wide variety of devices to be tested and verified without the need for separate testing environments for a multitude of EWH styles, sizes and manufacturers. Different environments include the size of the water heater, the size of the household, the usage profile of hot water, the manufacturing type, and the geographical environment where the EWH resides. Modeling allows utilities to minimize the installation and testing costs, which can be expensive and invasive to the owners of DERs, while still exploring behavioral differences due to changes in environment and usage. Having an accurate testing model also enables utilities to test DR commands, event schedules and communication pathways to ensure programs are effective, responsive and useful for specific grid services. The model developed in this research can be used to test the behavior of a wide range of EWHs, as long as the relevant parameters and a usage profile of water draws are input into the program before running the model.

Models help minimize the amount of communication needed between each DER and the utility controlling it. Instead of sending telemetry data to and from a device, repeatedly updating its state of operation, models predict what state and capacity a DER has based on physical parameters, past states and, in the case of EWHs, usage profiles.

For the purpose of this study, usage profiles are tracked in gallons per hour. These profiles can be changed to study EWHs in various residential environments (e.g. two person working household or a four person household with two children). And can also be used to
study different types of water heaters such as those used in industrial and manufacturing
settings, which commonly have very different daily demands and usage profiles [5].

1.5 Communication for DERs

Using EWHs for use in DSM programs requires communication from the device and the
central controller (i.e. utilities). There are a variety of communication protocols in use
throughout the utility sector, none of which presently have enough influence to develop a
national standard. Therefore, given the available communication options, CTA-2045 has
been chosen as the communication protocol for this research. This protocol was developed
in 2013 by the Consumer Technology Association [6]. CTA-2045 is a communication
protocol that provides a communication standard that can be applied and used by any DER,
not just EWHs. As defined within the standard, communication between the device and the
DSM facilitator is routed through the modular communication interface [6].

There are two main pieces to the design of this communication: the data link layer and
the application layer. The data link layer connects directly to the physical device (i.e. DER)
to identify device characteristics and to verify that all received commands are supported by
the DER. The application layer checks the message package for errors and ensures that all
required information has been provided to the end device. Figure 1.3 illustrates the direct
connection between an EWH and a CTA-2045 modular communication port, and Figure
1.4 shows the general communication network designed to send telemetry and command
signals to a DER device and from the DER to the manager of the DSM program.
This protocol is equipped with defined command signals such as "shed," "endShed," and absorb-like commands such as "setSetpoint" which can adjust the temperature set-point of the EWH and thus increase the thermal storage capacity of the device. "Load-up" is another supported command of CTA-2045. "Load-up" increases the temperature of the EWH to the upper bounds of the thermostat deadband, thus acting as a smaller version of the "setSetPoint" command. The EWH model uses an interface capable of receiving and
responding to such command signals. All signals are processed through a data link and application link layer to ensure that all received commands are supported by the device and can be achieved by the device depending on its thermostat temperature set-point and its present state of operation. This research utilizes the "shed," and "endShed" and future work involves incorporating absorb-like commands such as "setSetpoint."

This thesis presents completed research for modeling and controlling electric water heaters as DERs, specifically for demand response purposes through CTA-2045 communication protocol.
1.6 Problem Statement

Portland General Electric has requested for the construction of an EWH model to provide a virtual environment for testing and analyzing the practicality of using residential EWHs for demand response through the CTA-2045 communication protocol. The EWH model presented in this research was developed using Python script. This thesis focuses on building an EWH model that is capable of representing a wide variety of water heater types, sizes, household usage profiles, and geographical environments (i.e. inlet water temperatures based on region and water usage amounts based on societal norms). The EWH model acts as a stand-in for the physical device, not only in its behavior, but also in its ability to communicate, respond, and reply to CTA-2045 message commands and requests.

Overall, the developed model acts as a virtual test bed for EWH behavior and CTA-2045 communication. Such a model could also be used to test the security of a communication path, to verify system requirements for telemetry data, and to study the use of aggregated groups of EWHs for creating VPPs. All presented research involves one, single element, 12 gallon physical electric water heater located in the Portland State University Power Lab. The emulated model mimics this EWH by using a single element, single mass model for state prediction.

Electric water heaters, though strictly resistive devices, are complex systems with a variety of physical and digital inputs that each greatly impact the device’s performance and its effect on the daily demand curve. The obvious factors that impact device performance include the physical size of the EWH, the element ratings, and the location of the devices.
The greatest factor influencing EWH behavior is a household’s hot water usage profile. As stated in past research, it is difficult to properly model an EWH without extensive knowledge of the usage behavior of the household [16]. All of these variables and usage profiles can be manually inputted into each model through a configuration file. Each model has a unique configuration file that can be used to simulate most types, sizes, styles and usage profiles of a residential EWH.

Given accurate inputs into the configuration file, the model will output power consumption values as well as estimated temperatures of the EWH in real-time or simulated time.

1.6.1 Time Scale

To expedite testing, the configuration file is designated with a time scale value that can be adjusted according to the needs of the user. The time scale value is a ratio of seconds. The default value creates a simulation to real-time ratio of 1:120 (i.e. one second in simulation time is equal to two minutes in "real" time). The granularity of the output data decreases as the time scale ratio increases due to the increased time gap between data calculations. For accuracy, it is best to not increase the time scale beyond the 1:300 ratio (i.e. one second in simulation time is equal to five minutes in "real" time). This is a suggested limit but is up to the discretion of the user to determine how frequently temperature calculations should be logged. The time scale value is most sensitive when modeling smaller EWHs as their heat-up and cool-down periods are much smaller. In order to obtain accurate output data with high granularity for these smaller capacity devices, the time scale should be decreased.
1.6.2 Usage Profiles

Knowing the hot water usage of a residential household is imperative to building an accurate EWH model. The Python model built in this research allows the user to input a csv file with the estimated, or measured, hot water usage for a 24 hour period. Each model can run off of a unique usage profile or a group of EWHs can use the same profile for aggregation testing. The csv input allows for testing of any style of residential home and its specific hot water needs.

These usage profiles and their testing are especially important in maintaining customer comfort and provides a safe, non-intrusive platform for evaluating how DSM programs can effect and alter residential comfort and hot water availability. Because this research focuses primarily on direct control DSM programs, the impact on customers was not extensively studied. Though customer comfort wasn’t explicitly defined as an objective for the study, the model is capable of providing insight into that field of concern through minor model adjustments.

1.6.3 EWH Size and Manufacturer

The dimensions and construction of an EWH also impact the device behavior and possible utility for use in DSM programs. Physical parameters such as diameter, height, water capacity and element ratings are input into the model manually through the configuration file. The diameter of the tank can be approximated to be 1/5 of the height if not provided by the user [17]. The height, water capacity and element rating are mandatory inputs for an
1.6.4 Device Parameters

There are several other values that must be input into the model for accuracy: thermostat set-point (°F) as defined by the owner of the device and the thermostat deadband (°F) as defined by the device manufacturer. The default thermostat set-point is 120 °F and the default thermostat deadband is ±2°F. These values can be adjusting in the configuration file if different from the default values.

1.6.5 Model Precision

The EWH model built in this research is a one-element, single mass residential electric water heater model. The purpose of using a single-element model is to greatly reduce the computational intensity and complexity of the simulations. Dual element EWHs operate similarly to two, single element EWHs in that the elements are mutually exclusive in their operation (i.e. only one element can be on at a time unless otherwise stated in the manufacturers operation manual). By using a single element model, simulation tests can analyze the power flow and thermodynamics of the device while minimizing calculations and complexity. This is a standard approach and will be discussed further in Chapter 2.
2 Literature Review

The scientific literature on EWH modeling greatly varies based on the objectives of the model and the known and unknown parameters available to the modeling program. Over half of the papers reviewed for this thesis use single-element, single-mass models, similar to the one presented in this research and the other half use a two-element, multi-zone model using partial differential equations [7]. The models also vary in required parameter inputs (i.e. tank size, dimension, and hot water usage profiles). The following sections explore the benefits and drawbacks of each of these choices and presents justification for the research model provided in this thesis.

2.1 Dual Element modeling

Due to the operational nature of EWHs, using a dual-element model can add unnecessary complexity and increase the computational intensity of the program, especially for research like the one presented in this thesis, which focuses primarily on power consumption and available energy take (i.e. how much thermal capacity is available in the EWH before it reaches its thermostat set-point). Dual element models also require additional information that may be difficult or expensive to collect. Kondoh et al. presented an EWH model using a dual-element EWH and a dual-zone model for dynamic thermal analysis [2]. In order for this model to accurately predict the thermal state of the EWH, the flow rate of water usage
must be defined. This would require additional sensors that could record and send flow rates to the model to ensure accurate estimations. Additional sensors and increased telemetry data increases costs, installation time and the overall complexity of the system. To encourage residential customers to partake in such a DSM program, the process should be as cheap, simple and as non-intrusive as possible.

Xu et al. presented an EWH model that used a second-order differential equation. This model is able to predict the temperature of water within the EWH at any time and for any specific position within the tank. This is a complex and precise model that requires 25 input parameters in order to model properly. These parameters include values such as the thermal expansion coefficient, buoyancy factor and the thickness of the heating elements [7]. Parameters such as these can be difficult to define and again, increase complexity for residential customers that might be required to provide the device data. Kepplinger et al. used a similar model that required 29 input parameters per EWH being modeled [18]. Requiring these parameter inputs is extensive and, with respect to this thesis, does not add adequate value compared to the computation costs and added parameter complexity.

2.2 Single Element, Multi-Zone Modeling

There are several types of single element, multi-zone models presented in the literature such as that by Fernández-Seara et al. This model uses two operation modes to model the complete behavior of the EWH in its transition between an "on" (i.e. heating) and an "off" (i.e. idling) state. The problem with this model is that it does not provide the reader with a
way to accurately model the interactions between the zones and how heat transfers between
the upper and lower sections of the tank.

2.3 Models with Unknown Usage Profiles

The literature also contains several studies with a focus on predicting and identifying water
usage profiles through power consumption data [19]. The model presented by Shad et al.
uses daily power consumption data to build a hot water usage profile. This works well when
usage profiles are unknown and when power demand from the DER is accessible to the
utility. As a trade-off, it adds to the computational intensity of the model and increases
the need for frequent real-time device telemetry in order for the usage profile to be built
accurately. For the purpose of this research, water usage profiles were predetermined in
order to accurately verify the model with the physical device and to reduce computational
time.

2.4 Defined Model Criteria

Through this literature review the following criteria were defined for the model presented in
this thesis:

1. The model should be simple and accurate. All parameters that do not significantly
   add to the precision of the model in terms of the energy flow and averaged water
temperature should be eliminated.
2. Model simulations should be able to be performed within a reasonable computational time frame. This involves decreasing the complexity of calculations as well as providing user-defined time scale parameters that can expedite testings and analysis according to the needs of the research.

3. All configuration file input parameters need to have physical significance that can be easily identified and provided by the EWH owner. For example, the thermal expansion coefficient and thickness of the heating element(s), as used in several other research models [7, 18] are not easily defined EWH characteristics. Though they increase the accuracy of energy flow and temperature analysis for a single EWH, they do not add value for a statistical averaging model.
2.5 Objectives of Work

The overall objectives of this research are as follows:

1. Develop a single element, single mass model of an EWH that has variable inputs for ease of testing a variety of EWH models, styles and usage behaviors.

2. Create a model that can manipulate time scale values for fast and efficient testing.

3. Verify the model using a single element EWH equipped with the same input values as the model and operating off of an identical, predetermined hot water usage profile.

4. Build a model compatible with the CTA-2045 communication protocol and test CTA-2045 command communication to research the effectiveness of using EWHs in DSM programs.
3 Design Considerations and Methodology

3.1 Electric Water Heater Modeling

The model presented in this thesis uses a single mass, single element model to emulate the interior temperature and energy usage of a residential EWH. The design of the model is based on the average water temperature within the tank and energy flow analysis (i.e. power consumption over time). It is important to differentiate between residential, industrial and manufacturing because their patterns of usage and, overall size of EWHs vary. Residential EWHs have water capacities ranging from approximately 10 to 80 gallons, depending on the size of the house, and generally produce a usage profile that mimics the general shape of the daily load profile seen by utilities. It is also important to be aware of what time of the year the model is being used to test as this can affect the idling losses as well as hot water usage. These factors, and more, are discussed in the sections below.

3.1.1 Electric Water Heater Operation

As previously stated, EWHs are strictly resistive loads that turn on and off depending on the temperature of the water inside of the tank. Each EWH has a temperature set-point parameters that tell the device when to turn the resistive element(s) on and off. For dual element EWHs, the thermostat set-points for the top and bottom element are commonly
identical values, though this is not true for every system in use today. These temperature set-points are determined by the customer, and are commonly initialized to 120 °F. This is the parameter that would be adjusted for an "absorb" event. Each EWH also has a predetermined deadband value. The deadband range sets the upper and lower limits of the thermostat set point temperature and is determined by the manufacturer. This value is more fully defined in Section 3.1.3.

The tank is pressurized by the cold water supply line. When hot water is being used (i.e. flowing out), cold water is simultaneously flowing into the tank at an equivalent rate, keeping the EWH at a full and consistent capacity. An influx of cold water, due to hot water usage, lowers the average water temperature in the tank. If the temperature within the tank drops below the lower bounds of the thermostat deadband temperature, a resistive element turns on until the water temperature returns to the temperature set-point. The general layout and temperature profile for a two-element EWH is shown in Figure 3.1.
To extrapolate what a single mass, single element model would look like in comparison to Figure 3.1, consider summing the total amount of electric energy put into the water through operation of the elements. When the element is on, the power rating of the element (e.g. 1.5 kW) will add energy to the tank over time, slowly increasing its temperature. Instead of the elements sharing the heating responsibility as dual element EWHs do, with the top element having priority over the bottom element, imagine one element within the tank, heating the entire volume of water evenly until the temperature reaches the desired set point. The control circuit for a two element EWH is shown in Figure 3.2 [8].

It is helpful to view a dual element EWH as two, single element EWHs stacked on top of one another. Because of this similarity, a single mass, single element EWH model expanded
to account for size variations is a good approximation of the behavior of the larger, two element tanks common in most residential homes.

Figure 3.2: Dual Element Electric Water Heater Control Circuit [8]

### 3.1.2 CTA-2045 Definitions

The CTA-2045 Modular Communications Interface for Energy Management defines all DR commands in this research [6]. CTA-2045 is a defined communication protocol designed to integrate a variety of DERs into DSM control and aggregation programs. This
communication interface defines "shed" and "endShed" events as "Basic DR" commands and defines "absorb" as an "Intermediate DR" command. The difference between a "Basic DR" command and an intermediate command is the message format and length of each command. "Basic DR" commands have a fixed length data format, shown in Figure 3.3, while "Intermediate DR" commands can have messages of varying length depending on the command type and depending on the type of action being requested [6]. The "shed," "endShed," and "absorb" commands are defined as follows:

<table>
<thead>
<tr>
<th>Message Type = 0x08, 0x01</th>
<th>Reserved, Must be '0'</th>
<th>Payload Length = 0x00, 0x02</th>
<th>Opcode1</th>
<th>Opcode2</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Bytes</td>
<td>3 Bits</td>
<td>13 Bits</td>
<td>1 Byte</td>
<td>1 Byte</td>
<td>2 Bytes</td>
</tr>
</tbody>
</table>

Figure 3.3: Basic DR Message: Data Format for Shed and EndShed Events [6]

- **Shed**: an event intended to decrease power demand either by turning off an asset or by adjusting its settings to a lower power consumption level. As strictly defined by CTA-2045, "Shed" is an event requesting a complete drop in power consumption from the end device [6]. In the case of an EWH, this entails turning off the element(s) and shifting the device into an idling state.

The identification of the command as a "shed" event is defined in the "opcode1" section of the message. "opcode2" is a hexadecimal value that sets the desired duration of the event. For the purposes of this research no duration value was sent for a "shed" command (i.e. 0x00 was sent as the "opcode2" value). Instead, an "endShed" command was used to return the device to its normal state of operation.
• EndShed: this event removes the device from its previous "shed" state and moves it into a state of normal operation [6].

• Absorb: an event intended to increase power demand by either turning an asset from an off state to an on state, or by adjusting its settings to increase its energy consumption [6]. In the case of an EWH, this would mean raising the thermostat temperature set-point beyond the value set by the customer, as long as it is within the limitations of the device and its manufacturer settings.

When a DR event is received by the EWH model, the operating state of the system will change according to the type of DR event. The model can operate in three operational modes:

• Operation state → 0
  
  An operation state of 0 indicates that no DR command has been received or the received command was an "endShed" command. In this state the EWH operates in a non-DR mode (i.e. normal operation).

• Operation state → 1
  
  An operation state of 1 indicates that a properly constructed "shed" command was received through the CTA-2045 communication interface and that the EWH is operating in a DR-shed mode (i.e. the power consumption of the EWH has been reduced to zero (element(s) are in an OFF state).

• Operation state → 2
An operation state of 2 indicates that a properly constructed "absorb" command has been received through the CTA-2045 interface and that the EWH is operating in a DR-absorb mode. This functionality will raise the temperature set point of the EWH by a specified amount, thus immediately turning ON the water heater element(s) if $T_{\text{set}} > T_{\text{tank}}$.

### 3.1.3 EWH Model Parameters

In order to accurately model an EWH, there are a variety of parameter inputs that must be defined in the configuration file of the program. Required parameters for this model include the following:

- **Water Heater Size (gallons)**
  
  Volume capacity of water heater. This defines the total amount of water that can be housed within the EWH and is a main physical characteristic that determines the overall behavior of the device.

- **Water heater height & diameter (inches)**
  
  These are the physical dimensions of the exterior walls of the EWH. As previously stated, if the diameter of the EWH is not provided, it can be approximated to be $\frac{1}{5}$ of the device’s height [17]. These values are used to determine the rate of stand-by losses that incur while the water heater is in an idling state. Properly modeling standby losses is dependent upon accurate device dimensions, which helps utilities and DSM managers understand and predict spikes in power.
consumption that are a result of idling states and their accompanied re-heating cycles.

- **Element power rating (kW),** $P_{\text{rated}}$

  This is a predetermined value set by the device manufacturer. When the device is in an "on" state, this is the power consumption of the system. Because element operations in most dual element EWHs are mutually exclusive, the power consumption levels are the same for single and double element EWHs with identical element power ratings. This is a constant, unchangeable value regardless of the EWH size and value of the thermostat temperature value (i.e. power consumption is either equal to zero, or the power rating of the element(s)).

- **Customer thermostat set point ($^\circ F$),** $T_{\text{set}}$

  This is the desired tank temperature set by the customer before starting operation. This value is manually set for non-"smart" water heaters, commonly done through an analog temperature dial behind the top element panel of the device and is a constant for all operating states. For a "smart" water heater, this value can be remotely set and changed for partial "shed" or "absorb" events, if permitted by the manufacturer.

- **Temperature deadband ($^\circ F$),** $T_{\text{db}}$

  The temperature deadband value is the range of allowed temperatures where the elements will not respond to changes in temperature value. This value is a
constant defined by the device manufacturer and is generally ± 2 to ± 5°F of the thermostat set-point. For example, an EWH with a deadband value of ± 2°F with a thermostat value of 120°F will heat the water up to 122°F and will not begin its reheating cycle until the temperature drops below 118°F.

The physical EWH used for validation in this research does not have an upper deadband value and is thus defined as having a deadband value of −2 (i.e. the deadband range of the device is 120°F to 118°F.)

- Ambient room temperature (°F), $T_{\text{amb}}$

  This is the temperature of the room where the EWH is located. This value is largely affected by region and location. The default value is defined to be 65°F. This value is used for calculating the idling losses of the EWH.

- Cold water inlet temperature (°F), $T_{\text{inlet}}$

  Temperature of the tap water entering the tank upon start-up and during a water draw. When a water draw event occurs, this value plays a large role in determining the duration of the heat-up cycle. 52°F is the default value defined in the configuration file for each EWH but can be adjusted according to regional and climate differences. This is the "cold-start" value for the average inlet water temperature in Portland, Oregon [20].

- Initial water temperature (°F), $T_{\text{initial}}$
This is the initial average temperature of the water heater upon start-up. This value can be adjusted for ease of testing. For example, if analyzing stand-by losses of an EWH device, initialize the water temperature somewhere within the deadband range (preferable at the value of the upper bounds) then run the model. The inverse can be done if analyzing a full heating cycle. Running a "cold-start," which initializes the water temperature to be the same as the inlet temperature, ensures a full heating cycle will be run from start-up.

- Daily usage hourly profile (.csv)

See Section 3.1.5 for a full definition of usage profiles and their use within the EWH model.

- Time scale, \( t_{\text{scale}} \)

This value dictates the ratio of real-time to simulation time and is set to expedite running time but can also decrease accuracy if set to a high ratio value. The default time scale for this project was 300 (i.e. one second real time is equivalent to 300 seconds (five minutes) simulation time).

- Loop sleep timer (seconds)

The loop sleep timer sets the amount of time between energy take, which is defined as the amount of energy needed to raise the temperature of the water to the customer set-point, and temperature calculations. Each sleep loop cycle prints the current state and energy take capacity of the EWH. A smaller loop
sleep time will result in a more accurate model as this increases the granularity of the output data. A sleep timer of 1, with a time scale value of 300, results in an output of data every second, which in real-time, corresponds to a data logging rate of once every five minutes.

- **Initial Operation State**

  This value defines the start-up state of the EWH model (i.e. does the user want the model to begin in a state of normal operation or in a command state, such as "shed?"). State value definitions are shown in table 3.1, as translated from CTA-2045 command messages, and are fully defined in Section 3.1.2.

<table>
<thead>
<tr>
<th>State</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>0</td>
</tr>
<tr>
<td>Shed</td>
<td>1</td>
</tr>
<tr>
<td>Absorb</td>
<td>2</td>
</tr>
</tbody>
</table>

  Table 3.1: EWH Model Operating States

- **Initial Element State**

  This value sets the start-up state of the element within the model. A value of zero corresponds to the element(s) beginning in an off state and a value of one corresponds to the elements beginning in an on state. The state of operation takes priority over the element state value. For example, if the element begins in an on state and the model is defined to start in a "shed" event, the element will turn off upon start up to comply with the operation state.
Each of these values is input into the EWH model through a configuration file that can be unique for each EWH being modeled. Table 3.2 lists the default values for each parameter. The default parameter inputs were chosen based on average values for Portland, Oregon and the surrounding area. If no default value is listed, this indicates the value must be set manually for each model because of the high variability of possible values and due to the importance of that value in accurately modeling the EWH.

<table>
<thead>
<tr>
<th>Input Parameter</th>
<th>Default Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>EWH Size</td>
<td>–</td>
<td>Gallons</td>
</tr>
<tr>
<td>Height</td>
<td>–</td>
<td>Inches</td>
</tr>
<tr>
<td>Diameter</td>
<td>1/5 x Height</td>
<td>Inches</td>
</tr>
<tr>
<td>Element Rating</td>
<td>–</td>
<td>Watts</td>
</tr>
<tr>
<td>Thermostat Set-point</td>
<td>120</td>
<td>°F</td>
</tr>
<tr>
<td>Temperature Deadband</td>
<td>±2</td>
<td>°F</td>
</tr>
<tr>
<td>Ambient Room Temperature</td>
<td>60</td>
<td>°F</td>
</tr>
<tr>
<td>Inlet Water Temperature</td>
<td>52</td>
<td>°F</td>
</tr>
<tr>
<td>Initial Water Temperature</td>
<td>52</td>
<td>°F</td>
</tr>
<tr>
<td>Time Scale</td>
<td>300</td>
<td>–</td>
</tr>
<tr>
<td>Loop Sleep Timer</td>
<td>1</td>
<td>Seconds</td>
</tr>
<tr>
<td>Initial Operation State</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>Initial Element State</td>
<td>0</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 3.2: Configuration File: Default Values

3.1.4 EWH Model Flow and Operation

Given the configuration file parameter inputs, the normal operation of the EWH model functions as described in Figure 3.4.
Figure 3.4: Electric Water Heater Model Control Flow Diagram.
Figure 3.4 is a simplified flow diagram of the EWH single element, single mass model. This figure does not depict the DR loop. The DR loop is responsible for determining when a DR command is received and checks the CTA-2045 interface to see if there is a new input every one second, simulation time. All command messages are sent to the model via the command line interface of the Universal Control Module (UCM) [6]. If the CTA-2045 application and data link layer approve the message structure and command type (i.e. the command is sent in proper format and the command is supported by the receiving device), then the command is sent on to the model. The model responds within a second and adjusts the parameters of the model to mimic the request of the command. For example: if the received command is a "shed" event, then the EWH model will turn off the element, move to an operation state of one (see Section 3.1.2) and will remain in an idle state until the next command is received.

Data are collected from the model every sleep loop cycle. During each data collection cycle, the energy take, which is defined as the amount of energy needed to raise the temperature of the water to the customer set-point, and the averaged water temperature are printed to the terminal screen as well as a log file. These data can be extracted and plotted for visual analysis of residential EWH behavior.

3.1.5 Usage Profiles

As identified by the U.S. Department of Energy, there are five major sources of use for a residential EWH: showers, baths, sinks, dishwasher, and clothes washer [21]. Differentiating between types of usage is generally done through the volume of water drawn during each
draw event. Estimated hot water demands for the largest of these types of hot water events are defined in Tables 3.3 and 3.4 [2].

The differences in these usage amounts highlight the change in device behavior depending on the geographical location of the EWH. These values can be scaled to accommodate testing for smaller EWHs. For testing statistical averages, NREL determined the average hot water usage for a high income household to be 86 gallons per day, 41 gallons per day for a low-income household and a national residential average of 68 gallons per day [16].

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Water Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shower</td>
<td>30 gallons (114 liters)</td>
</tr>
<tr>
<td>Bath</td>
<td>20 gallons (76 liters)</td>
</tr>
<tr>
<td>Laundry</td>
<td>20 gallons (76 liters)</td>
</tr>
<tr>
<td>Dishwasher</td>
<td>15 gallons (57 liters)</td>
</tr>
</tbody>
</table>

Table 3.3: Hot Water Demand by Event Type for U.S. Residential Home [2]

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Water Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shower</td>
<td>40 liters</td>
</tr>
<tr>
<td>Bath</td>
<td>100 liters</td>
</tr>
<tr>
<td>Dishwasher</td>
<td>39 liters</td>
</tr>
</tbody>
</table>

Table 3.4: Hot Water Demand by Event Type for Residential Home in Sweden [2]

The usage profile is a central part of the EWH model and is also a parameter that is difficult to accurately define as described in the literature [16]. This is why studies often avoid emphasis on extreme precision of usage profiles and instead use averaged hot water demand curves as representation of aggregated groups of EWHs [17]. All water draw profiles are predetermined in this research. The determined usage profile is read into the program as an hour-by-hour, full day water draw schedule that is automatically repeated
after the 24th hour. Each hourly value defines the amount of water drawn (in gallons) for that hour in the 24 hour period.

Within the model, water draws happen at the top of the hour. The usage profiles can be expanded to increments of 15 minutes and is up to the user’s discretion to determine the water draw time intervals. Usage profiles are built and tested based on a generalized duck curve and residential water heater usage data [3]. A sample residential water heater usage profile is shown in Figure 3.5. In order to test these data on a small, 12 gallon EWH, values were scaled down to fit the capacity of the physical EWH used in this study.

![Residential electric water heater daily usage profile](image)

Figure 3.5: Residential electric water heater daily usage profile [3]

The purpose of these usage profiles is to allow testing of EWHs in varying environments. These different environments include households of varying sizes, households of varying dynamics (e.g. children v.s. no children), households where the home is empty during the day, households occupied during the day, etc. There is also a distinct difference between
usage profiles of residential EWHs and EWHs used in manufacturing. As previously stated, residential EWHs hot water usage profiles closely follow that of the daily demand curve, while EWHs in a manufacturing setting follow a different pattern.

There is also a difference in the total mean loads presented by these two usage climates. The residential EWHs have, on average, mean peak loads that are double in magnitude than those in a manufacturing environment [5]. These differences in behavior are illustrated in Figure 3.6 and Figure 3.7. This emphasizes the benefits of using residential EWHs for participation in DSM programs as they average a much higher amount of energy usage and provide utilities with greater capacity for load shifting and other services. They also have demand curves similar to the daily "duck curve." Being able to test these different usage profiles allows utilities to better understand EWH load trends and gives them insight into when and what EWHs are available for DR and how much load or absorb capacity they have to offer during specific times of the day and specific months of the year.
Figure 3.6: Hot Water Monthly Load Profiles: Manufacturing [5]

Figure 3.7: Hot Water Monthly Load Profiles: Residential [5]
3.1.6 EWH Model Temperature and Energy Take Calculations

The following table shows the defined variables for temperature and energy take calculations for the heating cycle of the EWH model.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
<th>Units</th>
<th>Calculation or Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{\text{rated}}$</td>
<td>Power rating of the device element(s)</td>
<td>Watts</td>
<td></td>
</tr>
<tr>
<td>$T_{\text{set}}$</td>
<td>Temperature set point</td>
<td>°F</td>
<td></td>
</tr>
<tr>
<td>$T_{\text{inlet}}$</td>
<td>Inlet water temperature as defined in configuration file and minimum water temperature of tank</td>
<td>°F</td>
<td></td>
</tr>
<tr>
<td>$T_{\text{initial}}$</td>
<td>Initial water temperature as defined in the configuration file</td>
<td>°F</td>
<td></td>
</tr>
<tr>
<td>$T_{\text{current}}$</td>
<td>Current average water temperature</td>
<td>°F</td>
<td></td>
</tr>
<tr>
<td>$T_{\text{old}}$</td>
<td>Previously calculated average water temperature</td>
<td>°F</td>
<td></td>
</tr>
<tr>
<td>$\Delta T_{\text{increase}}$</td>
<td>Change in temperature due to energy input</td>
<td>°F</td>
<td></td>
</tr>
<tr>
<td>$T_{\text{db}}$</td>
<td>Temperature deadband</td>
<td>°F</td>
<td></td>
</tr>
<tr>
<td>$T_{\text{hi}}$</td>
<td>Upper Bounds of temperature deadband</td>
<td>°F</td>
<td>$T_{\text{set}} + T_{\text{db}}$</td>
</tr>
<tr>
<td>$T_{\text{lo}}$</td>
<td>Lower Bounds of temperature deadband</td>
<td>°F</td>
<td>$T_{\text{set}} - T_{\text{db}}$</td>
</tr>
<tr>
<td>$t_{\text{scale}}$</td>
<td>Time scale value as defined in the configuration</td>
<td></td>
<td>–</td>
</tr>
<tr>
<td>$t_{\text{initial}}$</td>
<td>Initial time stamp when program starts</td>
<td>Seconds</td>
<td></td>
</tr>
<tr>
<td>$t_{\text{old}}$</td>
<td>Time stamp taken after calculations</td>
<td>Seconds</td>
<td></td>
</tr>
<tr>
<td>$t_{\text{new}}$</td>
<td>Time stamp taken before calculations</td>
<td>Seconds</td>
<td></td>
</tr>
<tr>
<td>$t_{\text{elapsed}}$</td>
<td>Time that has passed since the last calculation</td>
<td>Hours</td>
<td></td>
</tr>
<tr>
<td>$W h_{\text{gallon}}$</td>
<td>Watt-hours to increase the temperature of one gallon of water by one degree °F</td>
<td>Watt-hours/°F</td>
<td>2.44 [22]</td>
</tr>
<tr>
<td>$W h_{\text{tank}}$</td>
<td>Watt-hours to increase the temperature of the entire tank of water by one degree °F</td>
<td>Watt-hours/°F</td>
<td>2.44 x EWH size</td>
</tr>
<tr>
<td>$Q_{\text{initial}}$</td>
<td>Initial energy take capacity</td>
<td>Watt-hours</td>
<td></td>
</tr>
<tr>
<td>$Q_{\text{max}}$</td>
<td>Maximum energy take capacity</td>
<td>Watt-hours</td>
<td></td>
</tr>
<tr>
<td>$Q_{\text{in}}$</td>
<td>Energy input into the EWH</td>
<td>Watt-hours</td>
<td></td>
</tr>
<tr>
<td>$Q_{\text{current}}$</td>
<td>Current energy take capacity</td>
<td>Watt-hours</td>
<td></td>
</tr>
<tr>
<td>$Q_{\text{old}}$</td>
<td>Previously calculated energy take capacity</td>
<td>Watt-hours</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.5: Nomenclature for Energy Flow and Temperature Calculations
The value of 2.44 Watt-hours of energy needed to increase the temperature of one gallon of water by one degree °F was found through the following steps [22]:

- 1,055 Joules to increase the temperature of one pound of water by one degree °F
- 8.34 x 1,055 Joules = 8,798.7 Joules to increase the temperature of one gallon of water by one degree °F
- 8,798.7 Joules x \( \frac{1\text{Watt-second}}{1\text{Joules}} \times \frac{1\text{Watt-hour}}{3,600\text{Watt-seconds}} \) = 2.44 Watt-hours to increase the temperature of one gallon of water by one degree °F

Using the variables listed in Table 3.5 and Table 3.6, the average thermal behavior and energy flow of an EWH can be calculated by summing the energy source and the energy losses of the system. This includes stand-by losses and losses due to water draw events. Equations 3.1 through 3.7 show the calculations used throughout the model flow to display and record the energy take and average temperature during each logging cycling. After an energy take or average temperature has been calculated, it is printed then stored as \( Q_{\text{old}} \) and \( T_{\text{old}} \), respectively. For example, once the initial energy take capacity, \( Q_{\text{initial}} \) is calculated, it is then immediately copied and stored as \( Q_{\text{old}} \). Equation 3.1 shows this procedure for the initial energy take calculation. The same process is used for the time stamp values. Once the calculations have been completed, the \( t_{\text{new}} \) value is copied and stored into \( t_{\text{old}} \).

\[
Q_{\text{initial}} = (T_{\text{hi}} - T_{\text{current}})(Wh_{\text{tank}}) = Q_{\text{old}}
\] (3.1)
\[ Q_{\text{max}} = (T_{\text{hi}} - T_{\text{inlet}})(W_{\text{tank}}) \]  

(3.2)

\[ t_{\text{elapsed}} = \frac{(t_{\text{new}} - t_{\text{old}})}{3600} t_{\text{scale}} \]  

(3.3)

\[ Q_{\text{in}} = P_{\text{rated}} t_{\text{elapsed}} \]  

(3.4)

\[ Q_{\text{current}} = Q_{\text{old}} - Q_{\text{in}} \]  

(3.5)

\[ \Delta T_{\text{increase}} = \frac{Q_{\text{in}}}{W_{\text{tank}}} \]  

(3.6)

\[ T_{\text{current}} = T_{\text{old}} + \Delta T_{\text{increase}} \]  

(3.7)

The \( Q \) values defined in Table 3.5 and Table 3.6 represent the energy take capacity of the EWH in various states. For example, Equation 3.1 determines the initial energy take capacity of the system upon start-up. When the EWH is at its maximum temperature the \( Q_{\text{current}} \) equals zero. The \( Q_{\text{current}} \) is determined by the inlet water temperature of the EWH and the upper bounds of the thermostat set-point.

The upper limit of the deadband temperature or of the thermostat set-point can be adjusted to increase the energy take capacity of the system. This is a useful tool for "absorb" and "load-up" events. This can also be used to prepare an EWH for a "shed" event. With
temperature regulation through mixing valves, an EWH with a raised temperature set-point value can provide a larger reserve of hot water without the need for addition heating after a water draw event.

### 3.1.7 Stand-by Loss and Water Draw Functions

The following table shows the defined variables for temperature and energy take calculations for stand-by (i.e. idling) loss and water draw loss calculations used in the EWH model.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_s$</td>
<td>Surface area of tank</td>
<td>in$^2$</td>
</tr>
<tr>
<td>$A_{So}$</td>
<td>Surface area of tank from BPA study</td>
<td>in$^2$</td>
</tr>
<tr>
<td>$P_{standByLoss}$</td>
<td>Stand-by state power loss</td>
<td>W</td>
</tr>
<tr>
<td>$Q_{standByLoss}$</td>
<td>Stand-by state energy loss</td>
<td>J</td>
</tr>
<tr>
<td>$Q_{waterDrawloss}$</td>
<td>Water draw energy loss</td>
<td>J</td>
</tr>
<tr>
<td>$\Delta T_{standByLoss}$</td>
<td>Change in temperature due to idling</td>
<td>$^\circ$F</td>
</tr>
<tr>
<td>$\Delta T_{waterDrawloss}$</td>
<td>Change in temperature due to a water draw event</td>
<td>$^\circ$F</td>
</tr>
<tr>
<td>$T_{amb}$</td>
<td>Ambient temperature of EWH exterior environment</td>
<td>$^\circ$F</td>
</tr>
<tr>
<td>$T_{inlet}$</td>
<td>Inlet water temperature as defined in configuration file</td>
<td>$^\circ$F</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>Change in time</td>
<td>hours</td>
</tr>
<tr>
<td>$X_{waterDraw}$</td>
<td>Water draw amount for the hour as defined in the usage profile</td>
<td>gallons</td>
</tr>
</tbody>
</table>

Table 3.6: Nomenclature for Stand-by and Water Draw Losses

There are two types of losses modeled in this research: loss through water draw events, and standby loss where energy is lost through the exterior walls of the EWH into the surrounding environment [17]. In the stand-by loss calculations, having access to accurate tank dimensions is imperative to properly modeling the device’s behavior while idling.
There are a variety of approaches to modeling stand-by losses of an EWH. Dolan et al. uses an approach similar to the one presented in this paper but differs by including the thermal resistance of the tank insulation as a calculation factor [17]. In order to simplify calculations and the computational intensity of the model, a predetermined \( W/\circ F \) value was used from a previously conducted study. This study was conducted by Bonneville Power Administration in which the average ambient loss value for a 52 gallon, dual element electric water heater with a total surface area of 3958.41 in\(^2\) \((A_{so})\), was found to be 1.3868 \( W/\circ F \) [23]. This value can be scaled to any size cylindrical EWH by multiplying the calculated \( W/\circ F \) value by a ratio of surface areas. This calculation is shown in Equation 3.9. An ambient loss of 0.5283 \( W/\circ F \) was calculated for the 12 gallon electric water heater being used in this study. This is the amount of power lost while the water heater is idling (element(s) in an OFF state) with respect to the temperature in the tank.

To calculate the total amount of energy lost or decrease in temperature at any given time, use Equations 3.10 and 3.11.

\[
\dot{Q}_{\text{standByLoss}} = \frac{dQ}{dt} \quad (3.8)
\]

\[
\dot{Q}_{\text{standByLoss}} = 1.3868 \frac{A_s}{A_{so}} (T_{\text{current}} - T_{\text{amb}}) \quad (3.9)
\]

\[
Q_{\text{standByLoss}} = \dot{Q}_{\text{standByLoss}} \Delta t \quad (3.10)
\]
The EWH model checks for a water draw event at the top of every hour. If \(X_{\text{waterDraw}}\) is greater than zero, then a water draw of the size defined in the usage profile will be emulated using the equations below. All water draws are modeled as instantaneous events that occur at the start of the hour. This is not a realistic time frame for a water draw event as the flow rates of EWHs usually range within 1 to 5 gallons per minute (gpm) with clothes washers averaging a flow rate of 3.5 gpm, and 0.75 gpm for fixtures such as kitchen sinks and bathroom faucets [16].

A 2003 study conducted by The National Renewable Energy Laboratory (NREL) took the varying flow rates throughout a 24 hour period and used the average as a constant flow rate for analysis of water usage in a virtual environment [16]. The research goals of this thesis are focused on power flow and temperature change analysis. Using an instantaneous, hourly water draw model is a good approximation of the affects of water draw events on the overall system state. Also, the intent of this model is to be used in aggregated control platforms which means most simulations will be run in accelerated time. In accelerated time, most water draws will look instantaneous due to minutes of time passing between each calculation cycle. For a model meant to be run solely in real-time, a flow rate value would be applicable but for this model, instantaneous water draws provide enough accuracy. For a water draw event, the energy loss and average temperature change are calculated using

\[
\Delta T_{\text{standByLoss}} = \frac{Q_{\text{standByLoss}}}{W_{\text{tank}}}
\]  

(3.11)
Equations 3.12 and 3.13. The impacts of water draw events and stand-by losses on the system energy are shown in Equations 3.14 and 3.15.

\[ Q_{\text{waterDrawloss}} = (T_{\text{current}} - T_{\text{inlet}}) \times \text{waterDraw} \times W_{\text{gallon}} \] (3.12)

\[ \Delta T_{\text{waterDrawloss}} = \frac{Q_{\text{waterDrawLoss}}}{W_{\text{tank}}} \] (3.13)

\[ Q_{\text{current}} = Q_{\text{old}} + Q_{\text{waterDrawloss}} + Q_{\text{standByLoss}} \] (3.14)

\[ T_{\text{current}} = T_{\text{old}} - \Delta T_{\text{waterDrawloss}} - \Delta T_{\text{standByLoss}} \] (3.15)

### 3.1.7.1 EWH Model Assumptions

The calculations used in this model are founded on several assumptions. The following assumptions are used to simplify the model flow and to minimize the computational intensity of the program.

- The density and weight of water is a constant value and does not change with temperature.
- Convection losses while the tank is in a heating cycle are neglected.
- Losses through installation pipes and connections are not included in the stand-by loss calculations and are assumed to be zero.
• Water draw flow rates are neglected and all water draws are considered to be instantaneous.

### 3.1.8 Output Information and Plotting

When running the EWH model, the program will consistently output the calculated average temperature of the water inside the tank as well as the energy take capacity. Using this information, and by aggregating large numbers of these assets, the DSM operator can calculate a total capacity of reserve generation (i.e. "shed") and energy consumption (i.e. "absorb") availability to a utility for their use in stabilizing the grid and performing ancillary services such as frequency regulation and voltage support.

An example output of the program can be seen in Figure 3.8. The figure shows an EWH emulation of a 12 gallon EWH with a 1:300 simulation to real-time ratio. The EWH has a small capacity and the time scale ratio is high, causing the granularity of the data to decrease and the EWH to heat beyond the upper bounds of the thermostat setting. By decreasing the time scale ratio, the output log records the elements turning off at a temperature much closer to the designated upper limits of the temperature set-point. Figure 3.9 shows the same simulation but with a time scale ratio of 1:60. The parameter input values for the configuration files are listed in Table 3.8.
<table>
<thead>
<tr>
<th><strong>Input Parameter</strong></th>
<th><strong>Value</strong></th>
<th><strong>Units</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Outside Temperature</td>
<td>60</td>
<td>°F</td>
</tr>
<tr>
<td>Initial Temperature</td>
<td>100</td>
<td>°F</td>
</tr>
<tr>
<td>Inlet Water Temperature</td>
<td>55</td>
<td>°F</td>
</tr>
<tr>
<td>Thermostat Set-point</td>
<td>120</td>
<td>°F</td>
</tr>
<tr>
<td>Deadband Range</td>
<td>−2</td>
<td>°F</td>
</tr>
<tr>
<td>EWH Capacity</td>
<td>12</td>
<td>Gallons</td>
</tr>
<tr>
<td>Element Rating</td>
<td>1500</td>
<td>Watts</td>
</tr>
</tbody>
</table>

Table 3.7: EWH Simulation: Example Configuration File

Figure 3.8: EWH Emulator: Output Log File
3.2 Physical Electric Water Heater Design and Instrumentation

The physical EWH used for validation and testing in this study is a 12 gallon, single-element Whirlpool electric water heater. This EWH was chosen as the validating device for this research because of its small water capacity, which enables for ease of transport, simplified installation and expedited testing. This is due to the decreased amount of time need to perform full heating cycles and stand-by loss analysis. The physical dimensions and nameplate ratings of the EWH used in this study are listed below.
Figure 3.10: Electric water heater installation and set-up.

<table>
<thead>
<tr>
<th>EWH Parameter</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
<td>22.75</td>
<td>in</td>
</tr>
<tr>
<td>Diameter</td>
<td>16.00</td>
<td>in</td>
</tr>
<tr>
<td>Rated Voltage</td>
<td>120</td>
<td>V</td>
</tr>
<tr>
<td>Element Rating</td>
<td>1500</td>
<td>Watts</td>
</tr>
<tr>
<td>Number of Elements</td>
<td>1</td>
<td>–</td>
</tr>
<tr>
<td>EWH Capacity</td>
<td>12</td>
<td>Gallons</td>
</tr>
</tbody>
</table>

Table 3.8: EWH Simulation: Example Configuration File

3.2.1 Installation and Data Collection

Four thermal sensors were evenly placed with vertical displacement inside the EWH through the anode rod opening. This keeps the sensors in a constant position in order to maintain consistent and repeatable data collection. The sensors are attached to a long copper pipe
installed vertically through the anode rod tube and run down the length of the EWH. The anode rod opening is located on the top of the EWH tank and is normally used for placement of a steel rod coated in aluminum, magnesium, or zinc to keep the EWH from oxidizing [24]. Due to the short-term nature of these experiments, removing the anode rod is assumed to not affect the behavior of the device.

The anode rod opening was resealed after placing the pipe and sensors inside the EWH to ensure the tank remained pressurized throughout the entirety of testing. The thermal sensors are water proof and calibrated for accuracy before being placed within the EWH. Each thermal sensor has its own serial number and is connected to a unique GPIO pin on a Raspberry Pi 3, model B, microcontroller. The Raspberry Pi sends temperature data every two minutes to a logging file for data analysis and plotting. The sample rate of the thermal sensors are adjustable and can be set based on user needs and desired granularity of the data. Although the default sample rate is set to five minutes, for this study it was found that a sample rate of two minutes provided more accurate data for tracking energy capacity and temperature. The need for this smaller sample rate is due to the small size of the water heater; larger water heaters (i.e. greater than a 30 gallon capacity) would suffice with a five minutes sample rate due to their larger thermal lag. The logged temperature data are converted to energy capacity. When the energy capacity of the system is less than or equal to zero, the water heater turns off. The general design and installation of the EWH, the implementation of the thermal sensors and Raspberry Pi microcontroller are shown in Figure 3.10.
Usage profiles are input into the Raspberry Pi through a water draw file, similar to that of the model. Water draw events were controlled by a valve located on the hot water outlet of the EWH and were monitored using a flow meter. The meter and valve were also connected to the Raspberry Pi. All equipment used in the installation and implementation of the physical device is shown in Table 3.9. The overall physical installation of the EWH is shown in Figure 3.11(a) and 3.11(b).

<table>
<thead>
<tr>
<th>Measurement / Use</th>
<th>Equipment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>DS18B20 Temperature Sensor (4)</td>
</tr>
<tr>
<td>Sensor Alignment</td>
<td>1/2 inch copper pipe</td>
</tr>
<tr>
<td>Sealant</td>
<td>JB Weld Plastic Bonder</td>
</tr>
<tr>
<td>Data Collection</td>
<td>Raspberry Pi 3</td>
</tr>
<tr>
<td>EWH Connections</td>
<td>3/4 inch PEX piping, 3/4 inch brass thread-to-hose fittings and copper ring clamps</td>
</tr>
<tr>
<td>Water Flow Control</td>
<td>ASCO Solenoid Valve</td>
</tr>
<tr>
<td>Water Flow Meter</td>
<td>Omega FPR300/310 Series Flow Meter</td>
</tr>
</tbody>
</table>

Table 3.9: Physical EWH Installation and Testing Equipment
(a) Installation and Piping

(b) Anode Rod Opening

Figure 3.11: Physical EWH Set-up
4 Results

Tests were conducted to validate the EWH model’s heating cycle, stand-by loss function, water draw function, and usage profiles. For each test, data were collected from the model into a csv file at a variety of time scales ratios to compare data granularity and to determine the limitations created by the time scale value of the model. Data were collected from the physical EWH by thermal sensors at a rate of once every two minutes and were sent and stored in a log file on the Raspberry Pi. The data collected for each test and the comparison to the emulated model are plotted and shown in Sections 4.1-4.6. Correlation percentages were found using the Pearson product-moment correlation method. This method takes the covariance of two data sets and divides by the product of their standard deviations. The Pearson formula is shown in Equation 4.1, where "x" is the data set for the model and "y" is the data set for the physical EWH.

\[
\rho_{x,y} = \frac{\text{cov}(x, y)}{\sigma_x \sigma_y} \tag{4.1}
\]

The usage profile used for testing is shown in Figure 3.5. This water draw profile was input into the EWH model and provided to the physical EWH through a csv file similar to the format of the one shown below in Table 4.1 using 24-hour time. Table 4.1 values are
identical to the water draw values used in the usage profile validation test. A total of 45 gallons of hot water were used during the 24-hour profile.

<table>
<thead>
<tr>
<th>Hour</th>
<th>Water Draw (gallons)</th>
<th>Hour</th>
<th>Water Draw (gallons)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>13</td>
<td>1.75</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>14</td>
<td>1.5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>15</td>
<td>1.75</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>17</td>
<td>2.5</td>
</tr>
<tr>
<td>6</td>
<td>3.5</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>3.5</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>3.5</td>
<td>20</td>
<td>2.75</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>21</td>
<td>2.25</td>
</tr>
<tr>
<td>10</td>
<td>2.75</td>
<td>22</td>
<td>1.5</td>
</tr>
<tr>
<td>11</td>
<td>2.25</td>
<td>23</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.1: EWH Usage Profile [3]

4.1 Heat-up Validation

The validation test was a full heat-up cycle. Testing a full heating cycle entails running a "cold-start" (i.e. filling the tank with inlet temperature water) and allowing the EWH to heat the water up to the thermostat set-point without interruption from a water draw or DR event. The test began with an initial temperature of 70 °F and was heated to 120 °F. The inlet temperature of the water in the PSU Power Lab is much higher than the expected inlet temperature for a building in Portland, OR. The water inlet temperature fluctuates slightly throughout the day, but for this research is assumed to be a constant 69 °F. Data were gathered every two minutes for the physical EWH and every five minutes for the EWH model. Both tests ran for a total of 60 minutes. Figure 4.1 shows the comparison of the
physical water heater to the EWH model.

The heat-up validation test has a 0.998 correlation between the physical EWH and the emulated model. Due to the strictly resistive nature of an EWH, the linear model was straightforward in its implementation and this is supported by the two, almost identical behaviors. In a real-life setting, there will be very few instances of "cold-start" heating cycles as EWHs, unless responding to a DR event with a duration of multiple days (incredibly unlikely), are in constant operation and will continually keep the water inside of the tank within a few degrees of the thermostat set-point. This validation is important for the overall operation of an EWH and is represented, in smaller time increments, in the usage profile validation test.
4.2 Stand-by Loss Validation

The second test was a validation for the stand-by loss function. For this test the EWHs, both physical and emulated, were initialized to their customer set-point temperature, 120 °F. The 12 gallon EWH used in testing does not have an upper deadband value, so it was unnecessary to begin the test at 120 °F + T_{db}. In order to ensure the heating element did not operate during the test, the power supply was removed from the physical EWH, and for the EWH model, a "shed" event was sent without an "end shed" command. Both systems were left idle for 12 hours and data were collected at a sample rate of 5 minutes. The compared plots can be seen in Figure 4.2.

![Figure 4.2: Ambient Loss Cycle: Emulation vs. Physical Water Heater](image)

The correlation between the physical and emulated EWH is approximately 98%. There are two main causes for the 2% difference in behavior. The first is the constant $W/\circ F$ value for stand-by losses that was found in another study and used for this research. This value is an average ambient loss value found from a variety of different sized EWHs. Scaling the value by surface area is a good approximation but is not an exact indicator of long duration idling losses. In order to get a more exact correlation, individual tests would have to be
completed for each EWH and the stand-by loss constant would become a changing value that is specific to each device and its measured losses. This would be a time consuming and expensive process. Because of this, the correlation shown in Figure 4.2 is considered a good enough approximation for the purposes of this research.

Another factor contributing to the deviations in value are the uninsulated iron and brass connections on the physical EWH. These connections add surface area to the device that contribute to its rate of temperature loss. EWH installation is unique to each resident and affects the stand-by losses of the system. Future work could include testing this factor. If confirmed, the EWH will be adjusted to account for this added surface area.

It is also important to note that during normal operation an EWH will rarely be in an idling state for more than a couple of hours. The time spend in an idling state is dependent on the ambient temperature, water usage and the DR state of the device. Idling states with an extended duration, such as the one shown in this test, would be the result of a resident deliberately turning off the EWH.

4.3 Water Draw Validation

The third test was a water draw validation test. The purpose of this test is to show a large water draw cycle followed by a duration of idling with a slowed time scale ratio. The water draw amount used during this test was 9 gallons and occurred at the top of the first hour. The emulator was operating on a 1:30 second time scale ratio and the physical EWH was collecting data every sixty seconds. These scales were chosen to provide high granularity of
Figure 4.3 shows the comparison between the emulated and physical EWH. The physical EWH is shown through two line plots: the blue line shows data collected from the middle-top sensor and the dotted red line shows the average temperature of the three top sensors (data from the bottom sensor were thrown out during data analysis due to large differences in temperature readings, see Figure 4.5). The averaged temperature of the tank is slightly lower than the temperature readings of the middle-top sensor but overall, is very similar and provides a good estimation of the tanks temperature state.

The emulated EWH has a 98.9% correlation with the middle-top sensor temperatures and a 98.7% correlation with the averaged temperatures. The main discrepancies occur at the lower temperature values after a water draw event and in the idling state. The lower temperature values of the emulated model can be attributed to the physical EWH data set.
lacking the colder temperatures from the bottom sensor. Differences for the idling state and stand-by losses are hypothesized and explained in Section 4.2.

4.4 Usage Profile Validation: Temperature

The fourth set of tests were validation tests for a known usage profile. For this test, a 24 hour profile was run on the physical water heater as well as the emulated EWH. The following three figures show data comparison for three different time scales set in the emulator configuration time. Figure 4.4 shows the virtual EWH data running at a scale of 1 second emulation time to 60 seconds real-time.

![Figure 4.4: Usage Profile Temperature Plot: Emulation (1:60 seconds) vs. Physical Water Heater](image)

The temperature data collected using a 1:60 second time ratio value have a correlation of approximately 93% with the physical EWH. There are differences in temperature changes due to water draw events. This is because the EWH model calculates the total average
temperature of the tank while the physical EWH is limited to four thermal sensors, which are used to calculate the average temperature. Of the four sensors placed within the tank, the bottom sensor was of little use to the temperature calculations. Its location within the tank was close to the cold water inlet valve causing its temperature readings to rarely reach above 110 °F. The temperature data collected from the bottom sensor was thrown out before calculating the average. This could explain why the emulated model shows temperature drops slightly lower than that of the physical model. The differences in temperature readings of the four sensors are shown in Figure 4.5.

![Figure 4.5: Usage Profile Temperature Plot: Physical Water Heater Data from Four Sensors](image)

Figure 4.5 shows the comparison for the physical EWH usage profile data and the emulated model running at a time ratio of 1:120 seconds. The correlation between the two sets of data still follows the same general pattern of temperature change but has more
variations than the previous test. This is directly related to the time scale value of the EWH model. The higher the time scale value, the lower the data resolution will be. This is due to the computational time needed by the emulator to produce the temperature and energy take calculations for each loop and to also check the CTA-2045 interface for a command. It takes the program about two to three seconds each loop to output the new data values, so for a time scale of 1:120, that is a gap of four to six minutes between data points.

Once again, the temperature change values during a water draw event are slightly more dramatic for the EWH model. And also, the temperature value when the EWH model turns off and moves to an idling state is greater than the previous test. This is due to the time scale value which creates more time, without comparison to the thermostat set-point, for the water to be heated beyond its upper limit.

![Figure 4.6: Usage Profile Temperature Plot: Emulation (1:120 seconds) vs. Physical Water Heater](image)

The final comparison test, as seen in Figure 4.7, shows EWH emulation data with a time
ratio value of 1:240 seconds. The temperature data for this test are even more skewed than the previous two tests. Running the emulator with a 1 second to 4 minute time ratio causes inconsistent temperature data especially at the points of transition from an on state to an idling state. This is also accentuated by the small size of the physical EWH in testing and would be less dramatic if done on a tank with a larger capacity. Though the data aren’t perfectly correlated, there is still fairly consistent behavior between the two temperature sets. Overall, the EWH model can mimic the behavior of a physical EWH with a known usage profile but will lose accuracy as the time scale value increases and as the size of the EWH is reduced.

Figure 4.7: Usage Profile Temperature Plot: Emulation (1:240 seconds) vs. Physical Water Heater
4.5 Usage Profile Testing: Energy Take

Figure 4.8 is an energy take plot of a usage profile test running a 1:60 second time ratio, and a water draw schedule starting at 6:00 and ending at 22:00. The water draw amounts used in this test are identical to the respective hourly draws seen in Table 4.1. The energy take values for the emulated EWH were calculate by the emulator using Equations 3.1, 3.4, 3.5, 3.10, 3.12, and 3.14. The energy take values for the physical EWH were calculated from the recorded temperature data using the same equations.

This test highlights the available "absorb" capacity of the EWH at any given time. Large water draw events create higher energy take capacities than smaller water events or idling losses. The correlation evaluation of this test, which used the Pearson product-moment correlation method to compare the two data sets showed an accuracy of 94%. The main
discrepancies of these data are similar to the differences seen in Figure 4.4. Higher energy take capacities occur in the EWH emulator due to the colder temperature values for large water draws such as those at 6:00, 7:00 and 8:00. Because the physical EWH is plotted as average temperature values, the energy take of the system fluctuates into negative values further than the model. This indicates that the EWH has an average internal temperature higher than the upper limit of the thermostat set-point.

4.6 Shed Command Testing

Figure 4.9 demonstrates the "shed" command functionality of the emulator. During this test, a "shed" command signal was sent to the EWH emulator at 6am and lasted for approximately 1 hour and again around 5pm with a duration of 2 hours. The emulator was running off of the same usage profile as used for previous tests and had a time scale value of 1:120 seconds, similar to the second usage profile test.
Figure 4.9: Shed Command Test: Emulation (1:120 seconds)

The purpose of this test is to show the emulator response to a CTA-2045 "shed" command. Upon receiving the command, the emulator response is almost instantaneous and the program remains in a "shed" state until an "endShed" command is received. During the DR event, the water draw schedule continues to be processed as shown in the quick drops in temperature during the two events. Running this type of test is important for understanding the availability of hot water when the device is idling in order to ensure that a satisfactory level of customer comfort is maintained throughout the time during and after a "shed" event. These results can give utilities a better understanding of suitable duration values for DR events and how to best stagger them among customers to avoid large amounts of hot water depletion. Lower limits on temperature can be designated by the user or the utility to prevent customers from being
deprived of hot water for too long of a time. This could mean an EWH does not remain in a "shed" state for the duration of the event but will help maintain customer comfort.

![Figure 4.10: Shed Command Test: Emulation (1:120 seconds) with Energy Take](image)

Figure 4.10 shows the same test as Figure 4.9 but with energy take capacities plotted for comparison. This figure shows the amount of energy take available in the system after and during a "shed" event. Knowing these values is important for understanding how EWH loads will respond when a "shed" event ends and how long it will take the system to reach its temperature set-point again. This information also provides insight into how many EWHs would be needed to provide a given amount of load shift. Overall, the energy take capacity is directly proportional to the average temperature of the EWH. This is visible in
the mirror-like representations of the temperature and energy take plots.
The EWH model presented in this research produces accurate temperature data for a known usage profile given a small time scale ratio, relative to the size of the EWH being modeled. For the 12 gallon tank used in this research, a time scale value of 1:60 seconds produced the best results. A time scale value of 1:30 seconds was even more accurate but it did not provide enough added data granularity to justify the increased computational and simulation running time.

The heat-up validation test showed a 99% correlation with the physical device. This is a highly accurate function within the model and an imperative piece in regards to post-DR event power demand and hot water availability time-lines for customers participating in DSM programs. The stand-by loss test produced a correlation of 98%. This is another important piece of information that can be used to estimate appropriate DR event duration values with respect to customer comfort.

The DR function within the model and its compatibility with CTA-2045 communication protocol is another useful tool for testing DSM schedules and for determining time frames for stand-by losses and heating cycles. The shed command test provides insight into the effect of DR events on hot water availability and gives utilities a platform for analyzing DR schedules and their affects on customer comfort. For example, if a group of customers’ (i.e. a VPP) peak hot water usage usually occurs around 5pm, DSM controllers should ensure
that a fully heated EWH is available at that time. They should also provide the appropriate amount of time for reheating depending on the stand-by losses that could have occurred earlier as a result of a DR event.

The usage profile test using a time scale ratio of 1:60 had a correlation accuracy of around 93% when compared to the physical data. Though this is a lower correlation than the previous two tests, it is justifiable. When modeling real residential EWHs, the usage profile of the model will be emulating hot water usage of unknown quantity and timing and thus, does not need to have high accuracy; it simply needs to provide an estimation. From point of view of a utility, this function will be modeling averages of large groups of EWHs and will be used as an estimation of aggregated energy capacities.

The usage profile test and energy take test running at a 1 to 60 second time scale ratio show an average margin of error of approximately 0.5% for every two hour period. The largest error for a two hour period was 2.3% and the smallest margin of error for a two hour period was 0.14%. These differences vary depending on the stopping temperature of the heating cycle and the magnitude of water draw events. Table 5.1 shows the percent error for each two hour time period from Figure 4.4 and Figure 4.8. The hours and error values in bold relate to both figures.
<table>
<thead>
<tr>
<th>Two hour period</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:00 - 2:00</td>
<td>0.3%</td>
</tr>
<tr>
<td>2:00 - 4:00</td>
<td>0.15%</td>
</tr>
<tr>
<td>4:00 - 6:00</td>
<td>0.14%</td>
</tr>
<tr>
<td>6:00 - 8:00</td>
<td>0.9%</td>
</tr>
<tr>
<td>8:00 - 10:00</td>
<td>2.3%</td>
</tr>
<tr>
<td>10:00 - 12:00</td>
<td>1.3%</td>
</tr>
<tr>
<td>12:00 - 14:00</td>
<td>0.6%</td>
</tr>
<tr>
<td>14:00 - 16:00</td>
<td>0.8%</td>
</tr>
<tr>
<td>16:00 - 18:00</td>
<td>1.7%</td>
</tr>
<tr>
<td>18:00 - 20:00</td>
<td>0.8%</td>
</tr>
<tr>
<td>20:00 - 22:00</td>
<td>0.29%</td>
</tr>
<tr>
<td>22:00 - 24:00</td>
<td>0.3%</td>
</tr>
</tbody>
</table>

Table 5.1: Usage Profile Test: Temperature and Energy Take Accuracy

Model heating cycles with higher temperature stopping points had higher margins of error due to the physical device always stopping the heating events within 0.3°F of the set-point. The water draw events also affect the data correlation. The larger water draws produced more differences in data values than the smaller draw events. These differences are considered acceptable due to the purpose of the model. The EWH model will be used in a DR aggregation program where the model will receive telemetry data from the physical assets every hour to reinitialize the state of the simulation and ensure the model is properly mimicking the behavior of the physical EWHs. Accuracy drift over time was not shown in the results and in aggregated use will also be deterred due to hourly telemetry updates.

Given the +93% accuracy for the four main pieces of the model, the model was found to be accurate enough for use in aggregation programs that focus on averaging large groups of devices into VPPs. For a model running as the stand-in for real-time telemetry of individual EWHs, the model is able to provide utilities with information regarding available capacity.
and temperature data for determining DR event schedules. When the model is being used for this purpose it will also have telemetry data every hour from the physical devices that can be used to initialize the state to emulate the physical state of the devices. This limits the amount of time the model has to operate under unknown conditions and thus requires accurate simulations for the short time frames between received telemetry.

The simulations with a larger time scale value (e.g. 1:120 and 1:240) produced temperature data that had a weaker correlation to the physical temperature data but still provided a good estimation for the EWH state and possible energy take. Larger EWHs and groups of EWHs will have more capacity to allow for these larger time scale values due to their expanded heating cycles. Overall, aggregation systems emphasize statistical averaging over calculating highly accurate models for individual EWHs. For this purpose, larger time scale ratios could be used to provide good estimations of temperature and energy take availability.
6 Conclusion

Using EWHs for use in DSM programs is a research area gaining momentum due to the increasing penetration of stochastic generation sources such as wind and solar. This influx of non-dispatchable generation creates new challenges for maintaining a stable and balanced grid. One response to this issue is to build VPPs by aggregating large groups of DERs for use in DR events. DSM programs utilize aggregation to simultaneously control asset groups based on types of usage, availability and operation state. Though these programs are becoming more popular, large amounts of research still needs to be done to provide utilities with accurate data sets of how different DERs operate and how they can be utilized in a way that is cheap, efficient and has a minimal effect on customer comfort. Testing large groups of physical devices is time consuming and expensive. One way around this issue is to exploit the use of virtual models.

Virtual models that mimic physical devices provide a platform for flexible and efficient testing of DER behavior in a large variety of environments and initialization states. They provide grid authorities with the opportunity to test corner cases, practice sending DR events without obstructing the function of real end-use devices, and to quantify customer behavior without added instrumentation to residential assets. Another valuable aspect of a virtual model is that it can run in expedited time. This makes testing a much simpler and less time intensive process. Models can also be used to provide data regarding the statistical averaging
of groups of similarly behaving assets without the need for real-time telemetry and extensive communication networks. All of these benefits combine to give utilities accurate data for predicting load and generation availability, and in turn, provide them the information needed to effectively and intelligently control their connected DERs.

EWHs are excellent candidates for use in DSM programs for a variety of reasons. One, they are abundant grid assets that make up a sizable portion of the residential daily load curve. Two, they are strictly resistive loads with high power consumption, which simplifies control algorithms and decreases the number of assets needed for large shifts in power demand. And three, they have thermal storage capability that can be used for "absorb" events, as well as the more common "shed" events. In order to test the usefulness of aggregating and controlling EWHs for DR, an EWH model was built in Python script, and a physical water heater was installed and instrumented for data collection, comparison and verification.

The EWH model presented in this research currently supports a 24 hour usage profile and the "shed" DR event command using CTA-2045 communication protocol. The 24 hour usage profile is set by the user and can be used for an aggregated group or can be uniquely defined for each connected EWH model. A heating cycle test, a stand-by loss test, a usage profile test and a "shed" event test were completed to verify the model and to show the benefits of using virtual models in DSM programs. All of the tests that could be mimicked on the physical EWH (i.e. all except the "shed" test) produced a correlation value of 93% or higher. The errors in these results are attributed to the constant $W/°F$ value used in the stand-by loss calculations, the added surface area of the physical EWH and the time-scale
ratios used in the model simulation. Because the goal of this model was to provide a virtual
twin for a residential EWH that could be used for testing large groups of aggregated assets,
the accuracy of these results is sufficient.

Future work for this research will involve verifying the "shed" event response on the
physical water heater. A duration parameter should also be added to the "shed" command
input to ensure event testing is consistent and repeatable. The next step in the implementation
of the CTA-2045 communication protocol is to build functionality for the "absorb" command,
also with a duration input. Also, large groups of these virtual models need to be run
simultaneously to test aggregation controls and to collect EWH temperature and energy take
response data.

Another tangential study to be completed is a "partial shed" event study. This involves
decreasing the power consumption of the EWH to a level below its power rating while still
keeping the EWH in operation. For the model, this would decrease the element rating by a
specified percentage and would be implemented on the physical model by using pulse-width
modulation on a solid state relay. Pulse width modulation can be used to decrease the power
supply of an EWH to a determined percentage by varying the length of the switching period
of the solid state relay. This is a great alternative to full "shed" events because it can help
maintain a higher level of customer comfort while still providing load-shifting and other
ancillary services to utility operators.
Bibliography


[22] International Correspondence Schools. *Incandescent and arc lighting : electric-railway station equipment, interior wiring, mercury-vapor converters, voltage regulation, mod-


A.1 EWH Emulator Python Script

```python
#!/usr/bin/env python

import csv
import pandas as pd
import time
from time import sleep
import logging
from operator import truediv
import math
from math import pi
from ConfigParser import ConfigParser
from environment import Environment as Env

class updateWH(object):
    ''' Simulates electric water heater'''
    ''' Outputs temperature and energy take capacity'''
```
```python
def __init__(self, **kwargs):

    # Reading Configuration File
    # Saving parameters, initial values and time scales
    config = ConfigParser()
    config.read('config.ini')
    timeScale = config.get('Time', 'scale')
    self.timeScale = float(timeScale)
    sleepRate = config.get('Time', 'sleep')
    self.sleepRate = int(sleepRate)
    outsideTempF = config.get('Temp', 'outside')
    self.outsideTempF = int(outsideTempF)
    initialTempF = config.get('Temp', 'initial')
    self.initialTempF = int(initialTempF)
    inletTempF = config.get('Temp', 'inlet')
    self.inletTempF = int(inletTempF)
    setPointTempF = config.get('Temp', 'setPoint')
    self.setPointTempF = int(setPointTempF)
    deadbandTempF = config.get('Temp', 'deadband')
    self.deadbandTempF = int(deadbandTempF)
    maxTempF = config.get('Temp', 'maxTemp')
```

self.maxTempF = int(maxTempF)

size = config.get('WH', 'size')
self.size = int(size)

height = config.get('WH', 'height')
self.height = int(height)

diameter = config.get('WH', 'diameter')
self.diameter = int(diameter)

radius = self.diameter/2
self.radius = radius

elementRating = config.get('WH', 'elementRating')
self.elementRating = int(elementRating)

# Read usage profile
dp = pd.read_csv('DailyProfile.csv')

self.usageProfile = []
row = 0

while row < len(dp):
    self.usageProfile.append(dp.get_value(row, 'gallons'))
    row += 1
# Calculating ambient loss

```python
self.surfaceAreaStudy = 3958.41

self.surfaceAreaTank = 2*pi*self.radius*self.height + 
    2*pi*self.radius**2
```

# Conversion Values and initial parameters

```python
self.whToRaise1Gall1DegreeF = 2.44

self.whToRaiseTank1DegreeF = self.whToRaise1Gall1DegreeF * self.size

self.hoursToRaiseTank1DegreeF = 
    self.whToRaiseTank1DegreeF / self.elementRating

self.standByLosses = 1.3868 * (self.surfaceAreaTank / 
    self.surfaceAreaStudy)
```

def main(self):
    energyTake = (self.setPointTempF - 
        self.initialTempF) * self.whToRaiseTank1DegreeF
    temp = self.initialTempF

    print '********* FROM updateWH.main() *********' 
    print 'Initial energy take capacity of tank = %.2f' 
        % energyTake
```
```
print 'Initial temperature of water in tank = %.2f

→ [degrees F]' % temp

global data
data = open('waterheaterdata.csv','wb')

global data_transfer
data_transfer = csv.writer(data)

global usageTimeStamp
usageTimeStamp = time.time()# time stamp in seconds
timeStamp = time.time()# time stamp in seconds
elapsedTime = 0
hourCount = 1

# opState 0 = Normal operation
# opState 1 = Shed operation
# opState 2 = SetSetPoint operation

opState = 0
commandStamp = 'new'
elementState = 0
```
return {'opState': opState, 'energyTake': energyTake,
        'temp': temp, 'timeStamp': timeStamp,
        'commandStamp': commandStamp,
        'hourCount': hourCount,
        'elementState': elementState}

def updateWH(self, **whStateValues):
    opState = whStateValues['opState']
    energyTake = whStateValues['energyTake']
    temp = whStateValues['temp']
    timeStamp = whStateValues['timeStamp']
    commandStamp = whStateValues['commandStamp']
    hourCount = whStateValues['hourCount']
    elementState = whStateValues['elementState']

    if opState == 0:
        commandStamp_new = 'none'
        if commandStamp_new != commandStamp:
            print 'Water heater entering Normal Operation state'
            commandStamp = commandStamp_new
timeStamp_new = time.time()

data_transfer.writerow(['Time (min)'] +
                      ['Temperature (F)'] + ['Energy (Wh)'])

data_transfer.writerow([timeStamp] +
                        [round(temp,2)] + [round(energyTake,2)])

elif commandStamp_new == commandStamp:
    if energyTake > 0:
        if elementState == 0:
            elementState = 1
            print 'Element has been turned ON'

    timeStamp_new = time.time()

elif elementState == 1:
    timeStamp_new = time.time()
    elapsedTime = (timeStamp_new -
                    timeStamp)*self.timeScale / 3600  # this much of
    # an hr has passed
    energyInput = self.elementRating * elapsedTime  # in
    # watt-hour
energyTake -= energyInput

print 'Energy take available = %.2f [wh]'

tempIncrease = energyInput /

→ self.whToRaiseTank1DegreeF

temp += tempIncrease

print 'Temperature of the tank = %.2f [degrees F]'

data_transfer.writerow([timeStamp_new] + [round(temp,2)] + [round(energyTake,2)])

elif energyTake <= 0:

   if elementState == 1:

      elementState = 0

   print 'Element has been turned OFF'

timeStamp_new = time.time()

elif elementState == 0:

timeStamp_new = time.time()

elapsedTime = (timeStamp_new -

→ time Stamp)*self.timeScale / 3600
wattLoss = self.standByLosses * (temp -
    self.outsideTempF)
wattHourLoss = wattLoss * elapsedTime
tempLoss = wattHourLoss / self.whToRaiseTank1DegreeF
temp -= tempLoss

print 'Temperature = %.2f [degrees F]' %temp

energyTake = ((self.setPointTempF +
    self.deadbandTempF) - temp) *
    self.whToRaiseTank1DegreeF
data_transfer.writerow([timeStamp_new] +
    [round(temp,2)] + ['0'])

if temp < self.setPointTempF - self.deadbandTempF:
    energyTake = (self.setPointTempF - temp) *
        self.whToRaiseTank1DegreeF

print 'READY TO START REHEAT'
print 'Energy take available = %.2f [wh]'
    %energyTake
data_transfer.writerow([timeStamp_new] +
    [round(temp,2)] + [round(energyTake,2)])
timeStamp = timeStamp_new

totalElapsedTime = (timeStamp -

usageTimeStamp) * self.timeScale / 3600

# print 'TOTAL ELAPSED TIME USAGE = %f'

%totalElapsedTime# hours of use

hour = hourCount - totalElapsedTime

if hour <= 0:
    # has more than an hour passed?

    print 'Checking water draw schedule'

    profilePlace = hourCount - 1

    waterDraw = self.usageProfile[profilePlace]

    print 'Water draw for hour %d = %.2f [gal.]

    %(profilePlace, waterDraw)

data_transfer.writerow(['Hour '] + [profilePlace] +

['Water Draw = '] + [waterDraw])

data_transfer.writerow(['timeStamp_new'] + [round(temp,2)]

+ [round(energyTake,2)])

if waterDraw > 0:
    tempDiff = temp - self.inletTempF
energyOutput = tempDiff * waterDraw * 

→ self.whToRaise1Gall1DegreeF

energyTake += energyOutput

print 'Energy take available = %.2f [wh]'

→ %energyTake

tempDecrease = energyOutput / 

→ self.whToRaiseTank1DegreeF

temp -= tempDecrease

print 'Temperature of the tank = %.2f [degrees F]'

→ %temp

data_transfer.writerow([timeStamp_new] + 

→ [round(temp,2)] + [round(energyTake,2)])

hourCount = hourCount + 1

if hourCount > 24:

    print 'RESTARTING USAGE PROFILE'

    hourCount = 1# restarts usage profile if full day has passed

# if a Shed command has been recieved enter this loop

elif opState == 1:

    commandStamp_new = 'shed'
if commandStamp_new != commandStamp:

    print 'Shed command was recieved! Moving to
    Shed loop.'

    commandStamp = commandStamp_new
    timeStamp_new = time.time()
    data_transfer.writerow([timeStamp] +
    [round(temp,2)] + [round(energyTake,2)])

elif commandStamp_new == commandStamp:

    if elementState == 1:
        elementState = 0
        print 'Element has been turned OFF'
        timeStamp_new = time.time()

elif elementState == 0:

    timeStamp_new = time.time()
    elapsedTime = (timeStamp_new -
    timeStamp)*self.timeScale / 3600
    wattLoss = self.standByLosses * (temp -
    self.outsideTempF)
    wattHourLoss = wattLoss * elapsedTime
    tempLoss = wattHourLoss / self.whToRaiseTank1DegreeF
temp -= tempLoss

# sets lower bounds on temperature of tank

if temp < self.outsideTempF:
    temp = self.outsideTempF
else:
    temp = temp

print 'Temperature = %f [degrees F]' % temp

energyTakeAdded = self.whToRaiseTank1DegreeF * tempLoss

energyTake += energyTakeAdded

print 'Energy take available = %.2f [wh]'

data_transfer.writerow([timeStamp_new] + [round(temp,2)] + [round(energyTake,2)])

timeStamp = timeStamp_new

totalElapsedTime = (timeStamp - usageTimeStamp)*self.timeScale / 3600

#print 'TOTAL ELAPSED TIME USAGE = %f'  

# hours of use

hour = hourCount - totalElapsedTime

if hour <= 0:
    # has more than an hour passed?
print 'Checking water draw schedule'

profilePlace = hourCount - 1

waterDraw = self.usageProfile[profilePlace]

print 'Water draw for hour %d = %.2f [gal.]

data_transfer.writerow(['Hour '] + [profilePlace] + ['Water Draw = '] + [waterDraw])

data_transfer.writerow([timeStamp_new] + [round(temp,2)] + [round(energyTake,2)])

if waterDraw > 0:
    tempDiff = temp - self.inletTempF
    energyOutput = tempDiff * waterDraw * self.whToRaise1Gal1DegreeF
    energyTake += energyOutput

    print 'Energy take available = %.2f [wh]'

    tempDecrease = energyOutput / self.whToRaiseTank1DegreeF
    temp -= tempDecrease

    print 'Temperature of the tank = %.2f [degrees F]' %temp

print

92
data_transfer.writerow([timeStamp_new] + 
  [round(temp,2)] + [round(energyTake,2)])

hourCount = hourCount + 1

if hourCount > 24:
    print 'RESTARTING USAGE PROFILE'
    hourCount = 1  # restarts usage profile if full day has passed

sleep(self.sleepRate)

return {'opState':opState, 'energyTake':energyTake,
  'temp':temp, 'timeStamp':timeStamp,
  'commandStamp':commandStamp,
  'hourCount':hourCount,
  'elementState':elementState}
import os
import glob
from time import time, sleep
import RPi.GPIO as GPIO
import csv
import pandas as pd
from datetime import datetime

# Initialize GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

FMPIN = 6  # flow meter GPIO pin
VPIN = 17  # valve GPIO pin
HEPIN = 16  # heating element pin
TSPIN = 23  # temperature sensor pin

GPIO.setup(FMPIN, GPIO.IN, GPIO.PUD_UP)  # setup flow meter pin as input
GPIO.setup(VPIN, GPIO.OUT, initial=GPIO.LOW)  # setup valve pin as output
GPIO.add_event_detect(FMPIN, GPIO.RISING)  # add rising edge detection

GPIO.setup(HEPIN, GPIO.OUT, initial=GPIO.LOW)  # setup heating element pin as output

GPIO.setup(TSPIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)  # setup temp. sensor pin as input

# Read csv to create sensor ID array
sensorFile = open('Sensor_List.csv', 'r').read()
sensors = sensorFile.split('
')
sensors.remove(sensors[0])  # Remove header
sensIDs = []  # Initialize array of Sensor IDs
for eachLine in sensors:
    if len(eachLine) > 1:
        eachLine = eachLine.split(',')[1]
sensIDs.append(eachLine[1])

# Initialize temperature sensors and create array of file locations
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')
base_dir = '/sys/bus/w1/devices/

Sensor_files = []

for ID in sensIDs:
    device_folder = glob.glob(base_dir + ID)[0]
    Sensor_files.append(device_folder + '/w1_slave')

#Define functions for reading from temperature sensor

def read_temp_raw(device_file):
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp(device_file):
    lines = read_temp_raw(device_file)
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
temp_c = float(temp_string) / 1000.0

temp_f = temp_c * 9.0 / 5.0 + 32.0

return temp_f

# Define function to draw water

def draw_water(target):
    if target == 0:
        print('No draw for this hour')
        return()

print('Drawing %.2f gallon(s).' % target)

volume = 0
numPulses = 0

GPIO.output(VPIN, GPIO.HIGH)  # open valve

start_time = time()

while volume < target:  # keep valve open until desired volume has passed
    if GPIO.event_detected(FMPIN):
        numPulses += 1  # Count pulses from flow meter

        volume = float(numPulses) / 458.8  # Calculate volume

run_time = time()
elapsed_time = run_time - start_time

if elapsed_time > 180:
    print ('Timeout Error."

    break

GPIO.output(VPIN, GPIO.LOW) #close valve

print ('Volume drawn: %.2f gallon(s).' % volume)

#Read csv file with daily usage profile (one column for
# hours, one for gallons)

dp = pd.read_csv('DailyProfile.csv')

hours = []
gallons = []
row = 0

while row < len(dp):
    hours.append(dp.get_value(row,'Hour '))
gallons.append(dp.get_value(row,'gallons'))
    row += 1

state = 0  #Variable to mark heating element state (0 is
# off)

#Enter main program loop
while True:

    now = datetime.now()  # Update date/time

    filename = 'WH_Data_' + str(now.month) + '-' + str(now.day) + '-' + str(now.year) + '.csv'

    if not os.path.isfile(filename):  # Create new datafile
        data = open(filename, 'w')

        data.write('Time,Sensor 1,Sensor 2,Sensor 3,Sensor 4
        
        ')  # 4\n'

        data.close

        print('Creating new data file for ' + str(now.month) + '-' + str(now.day) + '-' + str(now.year))

    # Read middle temperature sensor, and adjust heating
    # Element if too hot or cold

    temp = read_temp(Sensor_files[1])

    if temp > 120 and state == 1:

        GPIO.output(HEPIN, GPIO.LOW)

        state = 0

        print('Temperature has exceeded 120 degrees -
        turning off heating element.')
elif temp < 118 and state == 0:
    GPIO.output(HEPIN, GPIO.HIGH)
    state = 1
    print('Temperature is below 118 degrees - turning on
    → heating element. ')

# Log data every 2 minutes
if now.minute % 2 == 0 and now.second < 2:
    data = open(filename, 'a')
    data.write(datetime.strftime(now, '%H:%M'))
    for eachSensor in Sensor_files:
        data.write(', ' + str(read_temp(eachSensor)))
    data.write('\n')
    data.close
    print('Logging Temperature data.')

# Draw water at the start of each hour
for i in range(len(hours)):
    if hours[i] == now.hour and now.minute == 0 and
    → now.second < 10:
```
draw_water(gallons[i])  #Draw scheduled volume
  ↪  for current hour

sleep(10)  #Wait 10 seconds to prevent
draw_water call from repeating
```