# Acknowledgments

This model was developed as part of Pacific Gas and Electric Company’s Code Readiness project number ET18PGE1901. Development was supported by Pacific Gas and Electric Company and performed by Beyond Efficiency in collaboration with Frontier Energy. The project was guided and managed by Andrew Doeschot and Kelly Cunningham at Pacific Gas and Electric Company. Key technical contributions were also provided by Marc Hoeschele (Frontier Energy), Ben Larson (Larson Energy Research), and Garth Torvestad (2050 Partners).

# Introduction

Flexible HPWH Performance Predictor (Flexi-HPWH) is a simulation model of HPWHs that was developed to predict the field performance of HPWHs based on field data collected by Frontier Energy at the Davis, California Creekside multifamily project. The Creekside project is a 90-unit low-income affordable project with 23 HPWHs generally serving four single occupant apartments. Pacific Gas and Electric Company (PG&E) sponsored the monitoring of ten HPWHs with an interest in documenting performance in both standard (fixed setpoint operating mode) and load-shifting operating modes. To support end-of-project evaluations Beyond Efficiency was contracted to develop a validated simulation model to utilize input data from the field monitoring to characterize annual energy performance in various operating modes. This document provides an overview of the model capabilities and how-to-use instructions. Familiarity with Python is helpful in using the tool.

Flexi-HPWH uses a user-defined hot water draw profile and user-specified parameters describing the HPWH to calculate the energy transfer into and out of the water in the tank over the course of the simulation. The calculated energy transfers to and from the water stored in the tank enable the model to track the temperature of the water and the energy flows that occur over time. These abilities allow the model to predict the performance of the device under varying hot water load, cold water inlet temperature, and ambient temperature conditions.

Specifically, the simulation model tracks the heat withdrawn from the tank via hot water draws, heat lost from the storage tank to the ambient surroundings via convection, and heat added to the water by the heat pump and backup electric resistance elements. These heat transfers are then applied to the water in the storage tank, assuming a fully-mixed tank, to track the changing amount of energy stored and temperature of the water in the tank.

A schematic of the system is shown in Figure 1.

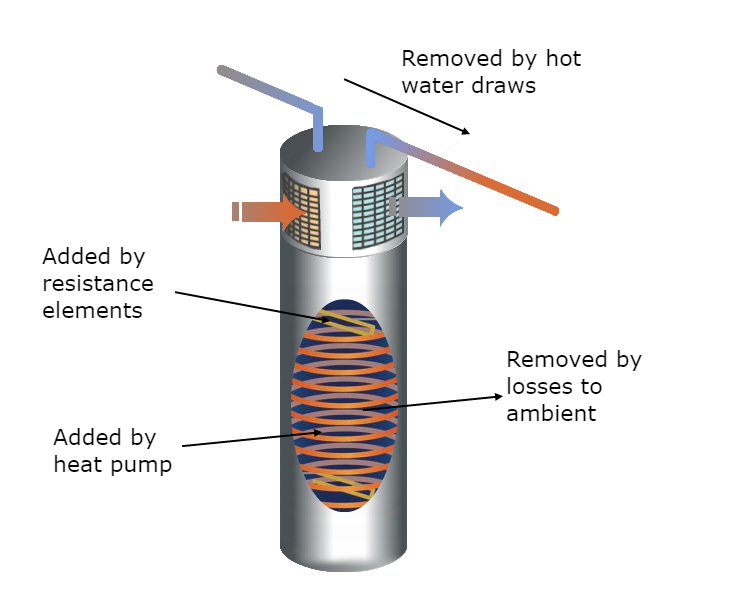


Figure 1: Schematic Showing the Heat Transfer Processes in the HPWH Simulation Model

Flexi-HPWH was developed to support a field monitoring project studying the performance of load shifting controls when applied to unitary residential-style HPWHs. As a result, it was designed to accept monitored data describing the operational conditions such as the inlet water temperature, the ambient temperature, and the hot water use patterns. The current iteration of the model utilizes monitored data for the inputs. That said, the model can accept any set of source data so long as it matches the inputs needed by the model and users can modify and use their draw profiles as desired. The required inputs are described in detail later in this Guide.

To ascertain the performance of load shifting control strategies, the model must be able to identify both a baseline (No load shifting) and a test result (With load shifting) for each day under varying loading and ambient conditions. By utilizing a validated simulation model, both baseline annual profiles and load-shifting profiles can be derived. Additionally, multiple load shift strategies can be simulated to assess impacts on total kWh consumption, time-of-use profiles, and resulting carbon impacts.

This model supports that goal by being a highly accurate representation of HPWHs, and by being flexible enough to accept either simulation default assumptions or monitoring data that Frontier Energy collects during the monitoring process. Development of this model was pursued because other available options did not readily suit the needs of the project. The Title 24 CBECC-Res compliance software includes a HPWH simulation model adapted to California’s needs; however, it is only capable of performing simulations using assumed hot water draw profiles and therefore is unable to use monitored data as inputs. Ecotope’s HPWHsim[[1]](#footnote-1) is another advanced HPWH simulation model that could perform similar work, but needs a constant load shifting schedule. Frontier Energy is testing different load shifting strategies on different days, and a model designed specifically to accept that flexibility was necessary. Since Beyond Efficiency already had developed a Python-based simulation model for gas absorption HPWHs[[2]](#footnote-2) which was designed to accept either field data or simulation assumption inputs, modifying the logic to represent electric HPWHs was considered to be the best approach.

Flexi-HPWH contains multiple different algorithms that can be used for different purposes. The existing algorithms are:

* **Model\_HPWH\_MixedTank**: This model uses the assumption of a well-mixed, single node tank. Control logic does not differentiate between temperatures at the upper and lower thermostats because the model cannot estimate separate temperatures at the two heights. This model is useful for situations when detailed performance data and parameters describing the HPWH’s operation is not available. Due to the decreased complexity it is less able to match real HPWH performance, and will present less accurate results.
* **HPWH\_MultipleNodes**: This model splits the volume of water in the tank into a user-specified number of nodes. The nodes each represent a layer of water at different depths in the tank, sequentially rising from the bottom to the top. Flexi-HPWH tracks the energy transfer and temperature in each node during each timestep in the simulation. Since this model uses multiple nodes at different temperatures it is able to track temperatures at the height of the upper and lower thermostats, enabling more complex controls. This model is useful in situations when detailed data describing the control logic of the HPWH is available. Since this mode is more complex it is more able to match real HPWH control logic decisions, and will present more accurate results.

# Key Model Development Goals

The main goal of the model development was to develop a public domain simulation tool that accurately characterizes HPWH operation in both fixed setpoint and load-shifting strategies, whereby setpoints are varied by time of day to bias operation to off-peak periods (by “overheating” storage) and therefore reduce later on-peak operation. This has increasing benefit for California as electrification activities proceed the impact of HPWHs on the grid are ideally configured to maximize midday operation (when renewables are most prevalent) and minimize on-peak operation.

# Single-Node Model

## Assumptions and Base Principles

There are several key assumptions used in this simulation model. Understanding these assumptions is key to using the model to its intended potential.

First, this model assumes a fully-mixed tank. Instead of modeling multiple nodes in the storage tank to represent tank stratification effects more precisely, this model assumes a single thermal node with a uniform temperature from top to bottom. This simplification dramatically reduces the simulation time while still enabling accurate predictions of energy use over long draw profiles. The downside of this assumption is that the model will be less accurate in predicting the hot water outlet temperature. In a stratified tank the hot water outlet temperature will remain relatively constant as the hot water draw occurs, until at some point the outlet water will start to get colder as the tank depletes. A well-mixed tank will not be able to capture that effect, instead reporting gradually reducing outlet water temperatures as hot water leaves the tank. The decision to use a fully-mixed tank model at this time was based on factors including simulation tool complexity, and the fact that the model can meet current project needs with this assumption.

The efficiency of the heat pump, expressed as coefficient of performance (COP), is assumed to be primarily a function of only the fully mixed tank water temperature, with the impact of ambient temperature neglected. It is treated as a second order regression though most HPWH performance curves are nearly linear. This assumption is based on previously published works by Ecotope[[3]](#footnote-3) showing a good match between the derived regression relationships and monitored data.

Though many of the transient operating conditions such as inlet water temperature and ambient temperature are currently designed to be read from the monitored dataset it is possible to override those inputs with assumptions instead. Time-varying assumptions can be read from an input data file in the same manner as monitored data. The code also includes a parameter providing the ability to set a user-defined constant value if desired.. This is useful for simulation studies where monitored data is not available and an assumption must be used.

## COP Regression Equation

The mixed temperature calculated in the simulation model is the only input needed to predict the COP of the heat pump.

The general form of the equation estimating the COP is a single variable, second order regression expressed as follows:

Where

= Coefficient\_2ndOrder\_COP[[4]](#footnote-4)

= Coefficient\_1stOrder\_COP

= Constant\_COP

= The fully-mixed temperature of the water in the storage tank, represented in the “Tank Temperature (deg C)” column in the model dataframe

The default values for ,, and are 7.67, -0.037, and 0, leading to a default implementation of:

## Parameters

This simulation model uses several different parameters to describe the HPWH being studied. Changing the parameters will change the behavior of the simulation model and control logic choices, so care must be taken to ensure that the values specified for each parameter match the reality of the installation. The parameters available in the model are:

* **Temperature\_Tank\_Initial**: This parameter sets the temperature of the water in the storage tank at the beginning of the simulation. Since this model assumes that the water in the tank is well mixed at a single temperature, only one temperature specification is needed. This and all water temperatures listed below are specified in degrees Celsius.
* **Temperature\_Tank\_Set**: This represents the set-point temperature of the water heater. To enable simulation of load shifting controls, this parameter is entered as a dictionary with separate key:value pairs for each hour of the day. The set temperature is specified by entering the desired set temperature for each hour of the day. Note that midnight – 1A is represented as hour ‘0’, 1A – 2A is represented as hour ‘1’, and so on up to 11P – midnight being represented as hour ‘23’. An example showing a set temperature of 48.9 ̊ C (120 ̊ F) for most of the day, and 60 ̊C (140 ̊F) during the 10A-4P load shifting period follows.

Temperature\_Tank\_Set = {'0': 48.9, '1': 48.9, '2': 48.9, '3': 48.9, '4': 48.9, '5': 48.9, '6':

48.9, '7': 48.9, '8': 48.9, '9': 48.9, '10': 60, '11': 60, '12': 60,

'13': 60, '14': 60, '15': 60, '16': 48.9, '17': 48.9, '18': 48.9,

'19': 48.9, '20': 48.9, '21': 48.9, '22': 48.9, '23': 48.9}

A library of set temperature profiles is available to facilitate performing simulations using different load shifting control strategies. The library is contained in the accompanying Set\_Temperature\_Profiles.py, and a specific control strategy can be used by calling the get\_profile() function on a specific draw profile when specifying Temperature\_Tank\_Set. For example, a control strategy with a constant set temperature of 60 °C could be simulated by calling the ‘Static\_60’ set temperature profile as in the following code:

from Set\_Temperature\_Profiles import get\_profile

Temperature\_Tank\_Set = get\_profile(‘Static\_60’)

Appendix A presents a list of the currently available set temperature profiles contained in Set\_Temperature\_Profiles.py.

* **Temperature\_Tank\_Set\_Deadband**: This sets the deadband range below the set temperature used to engage/disengage heat pump operation.. For instance, a deadband of 10 degrees Celsius would indicate that 1) the heating sources activate when the water temperature is 10 degrees below the set temperature and 2) the heating sources deactivate when the water heater returns to the set temperature.
* **Temperature\_Water\_Inlet**: This parameter specifies the temperature of cold water entering the HPWH. Depending on the input files that the user has, they may or may not have access to data showing changing inlet cold water temperatures at the device. If that data is not available, this parameter sets a constant value for inlet water temperature to be used in the simulation. If the data is available, the user can choose whether to use it or the constant temperature in the simulation.
* **Temperature\_Ambient**: This parameter defines the temperature of the air surrounding the HPWH. It is used to calculate the heat loss through the walls of the water heater to the surrounding environment and does not vary during the simulation. This parameter is only influential if using a constant ambient temperature assumption instead of including time-varying ambient temperature in the input data file.
* **Temperature\_Cutoff:** This parameter defines the cutoff temperature below which the heat pump does not operate to prevent freezing the coils. When the ambient temperature is less than the set temperature, the model sets the heat pump heat addition and electric consumption to zero and increases the backup electric resistance activation temperature to match the heat pump set temperature.
* **Volume\_Tank**: This specifies the actual (not nominal) volume of hot water stored in the tank. It is specified in liters.
* **Coefficient\_JacketLoss**: This parameter provides a heat transfer coefficient for the heat transfer between the water in the tank and the surrounding air. The heat loss rate is defined as this term multiplied by the temperature difference between the tank and the ambient air. It is specified in W/K.
* **Power\_Backup**: This parameter specifies the electric power of the backup resistance heating elements while they are running. The model assumes 100% efficiency for the backup elements, so this specifies both the rate at which electricity is consumed and the rate at which heat is added to the water. It is specified in Watts.
* **Threshold\_Activation\_Backup**: This specifies the water temperature at which the backup resistance elements activate. After the mixed water temperature falls below this temperature both the resistance elements and the heat pump will be active.
* **Threshold\_Deactivation\_Backup**: If the backup resistance elements are active, this parameter defines when they disengage. Once the water temperature is higher than this threshold, the resistance elements will deactivate, and the heat pump alone will heat the water. This parameter is typically set to Temperature\_Tank\_set.
* **HeatAddition\_HeatPump**: This parameter defines the heat consumed by the heat pump when operational. It is used to determine the heating capacity of the heat pump and is multiplied by the calculated COP to determine the rate heat is added to the water. It is specified in Watts.
* **ElectricityConsumption\_Active**: This parameter specifies the HPWH electrical demand when the heat pump is active. It represents the electricity consumed by the sensors, controller, fans, etc. It is specified in Watts.
* **ElectricityConsumption\_Idle**: This parameter specifies the HPWH electrical demand when the heat pump is idle (not heating). It represents the electricity consumed by the sensors and controller but does not include components such as the evaporator fan. It is specified in Watts.
* **CO2\_Output\_Electricity**: This is a multiplier used to calculate the CO2 produced by operating the HPWH. It can either be a constant value, assuming that the CO2 production rate of the electric grid is constant, or time-varying data can be used, if available. It is expressed in ton/MWh.
* **Coefficient\_2ndOrder\_COP**: This is the second order coefficient used in the calculation to identify the COP of the heat pump as a function of the temperature of water stored in the tank. It is described in more detail in COP regression equation.
* **Coefficient\_FirstOrder\_COP:** This is the first order coefficient used in the calculation to identify the COP of the heat pump as a function of the temperature of water stored in the tank. It is described in more detail in COP regression equation.
* **Constant\_COP:** This is the constant used in the calculation to identify the COP of the heat pump as a function of the temperature of water stored in the tank. It is described in more detail in COP regression equation.
* **Installation\_Configuration:** This informs the model about the location and installation details of the HPWH. The installation impacts how much the HPWH impacts the surrounding ambient temperature and the airflow available for venting the exhaust air, both of which impacts the performance of the HPWH. Appendix B presents the different options and how they impact HPWH performance.

## Inputs

The HPWH simulation model was developed in Python and uses a two-script architecture. The two scripts are as follows:

* **HPWH\_Model.py**: This script contains the code representing HPWH performance. It includes the heat transfer, COP, and control logic calculations based on the provided inputs and parameters. This script is intended to be a collection of functions each simulating HPWHs with different assumptions, so the user has the capability to redefine the performance relationships based on the goals of their project. Currently it contains a single function representing the performance of a HPWH with a well-mixed tank assumption.
* **A wrapper script**: Since HPWH\_Model.py only contains functions, simulating the performance of a HPWH requires a wrapper script that defines the HPWH using the previously described parameters, pre-processes the data, calls the desired HPWH\_Model.py function, and post-processes the data. This script needs to be created specifically for each project, converting the input data into the required format and performing the necessary calculations of creating the desired plots after HPWH\_Model performs the simulation.

The simulation model needs an input file detailing the hot water draw profile to be used in the simulation. This draw profile should describe the time and volume of each hot water draw. This two-script architecture makes the HPWH simulation process quite flexible, and able to accept most data sets as an input. The requirement is that the wrapper script converts the input draw profile into the format that HPWH\_Model.py’s functions can read. Specifically, the wrapper script must create a table spanning the full duration of the simulation and containing the following inputs when calling the HPWH\_Model.py functions. The table is passed to the HPWH\_Model.py script as an input. Note that this table contains both input data that the model needs to perform calculations, and placeholder columns for the model to store output data.

* **Timestep**: This data represents the duration of time between two rows in the table, measured in minutes. The name of the column must be ‘Timestep (min)’ in order for HPWH\_Model.py’s functions to locate it. Note that HPWH\_Model.py’s functions can accept draw profiles with varying timesteps, allowing the user to specify a desired timestep interval (Including varying the timestep during the simulation). Shorter timesteps (1 minute is generally recommended) lead to more accurate predictions of the HPWHs control choices and better simulation results. This data is required for the model to perform simulations.
* **Water Temperature**: This column represents the temperature of water in the storage tank. Since the model currently uses an assumption that the water is well-mixed this is represented by a single column named ‘Tank Temperature (deg C)’. When other functions that do not assume a well-mixed tank are available they will require multiple columns specify the temperature of each node in the simulated tank. The water temperature needs to be specified in degrees Celsius. This data is required for the model to perform simulations.
* **Jacket Losses**: This column shows the amount of heat lost from the water to the ambient conditions during each timestep in the simulation. It is calculated using the water temperature, the specified UA parameter, and the ambient temperature. It is specified in Joules and referenced using the column name ‘Jacket Losses (kWh)’. This is a placeholder column that the model will fill with output data during simulation.
* **Energy Added by the Backup Resistance Elements**: This column represents the calculated energy added to the water by the backup resistance elements in each timestep. It is calculated using the specified Power\_Backup parameter, and the user-defined control logic. If the control logic determines that the backup resistance element will operate it adds the energy specific by Power\_Backup to the water. This value is in Joules and referenced using a column named ‘Energy Added Backup (kWh)’. This is a placeholder column that the model will fill with output data during simulation.
* **Energy Withdrawn**: This term calculates the energy withdrawn from the tank by hot water draws during the timestep. It is calculated using the volumetric flow rate during the timestep, the temperature of water in the tank, and the inlet water temperature. It is represented in Joules and referenced using a column named ‘Energy Withdrawn (kWh)’. This is a placeholder column that the model will fill with output data during simulation.
* **Energy Added by the Heat Pump**: This column represents the energy added to the water by the heat pump. If the user-specified control logic indicates that the heat pump will be active it adds heat to the water at a rate equal to the HeatAddition\_HeatPump parameter. This value is in Joules and is represented with a column called ‘Energy Added Heat Pump (kWh)’. This is a placeholder column that the model will fill with output data during simulation.
* **Total Energy Change**: This column combines all of the energy gains/losses over the timestep to calculate the total amount of energy change during that time. It is in Joules and referenced using a column named ‘Total Energy Change (kWh)’. This is a placeholder column that the model will fill with output data during simulation.

## Output

HPWH\_Model.py does not by default save output files, and the output file location must be specified in the wrapper script. The example wrapper script contains code to save the output file, but this code should be modified as needed for each project’s requirements. To use the code in the example wrapper script you must create a folder named ‘Output’ in the project folder. The script will then save the output file in that folder. The filename will start with the word ‘Output\_’ followed by a section of the provided draw profile name as provided by the user. The code to edit the used filename is specified in the INPUTS section of the code and filters the filename to only include the desired portion of the name. It is recommended that one include enough of the input data filename to make it clear what data set was used when running the simulation and no more. Filenames that are not descriptive enough will make it hard to keep track of which output file corresponds to which input file. Filenames that are too long (More than 260 characters including the folder path structure) cannot be opened by Windows.

The format of the output file will depend on the code used in the wrapper script. It is considered best practice to never remove data from the table, and to save it all in the output file, as this facilitates any error checking steps that may be required. This means that the output file is likely to include:

* All input data in the format originally provided (E.g. Temperature data may be provided in degrees Fahrenheit),
* All input data converted to the format required by HPWH\_Model.py (e.g. Input temperature data converted from degrees Fahrenheit to degrees Celsius),
* All outputs from the called HPWH\_Model.py function, and
* All data from post-processing calculations.

## Model Limitations

When using this model, it is important to keep in mind the limitations that arise from the current design of the tool. Some important considerations include:

* **Outlet Water Temperature**: As previously stated, the mixed tank assumption means the model cannot properly track stratification in the tank. This causes fundamentally different hot water outlet temperature behavior, and this model has limited ability to estimate the hot water delivery performance of the device.
* **COP Simplification**: The regression calculating the COP of the heat pump is simplified. As a single variable regression it can’t predict the impacts of any variables other than the water temperature. So far this has shown to provide accurate results. Studying the impact of other inputs on the COP will require new datasets and modifications to the simulation model.
* **Default Parameter Values**: The parameters presented as the defaults are typical values based on one particular HPWH, and results will vary if using different equipment. Simulations predicting the energy use of specific HPWHs can be generated with this model, but it requires carefully identifying the parameters that describe that specific HPWH as described in Parameters.

## Detailed Description and Tutorial

There is currently one HPWH simulation function available in HPWH\_Model.py. It is called Model\_HPWH\_MixedTank. This model assumes a well-mixed tank. It is called by importing HPWH\_Model into the wrapper script and calling the function. Model\_HPWH\_MixedTank requires three inputs. They are:

* **Model**: This input is the previously defined table that contains all the input data described in Inputs. The example wrapper code includes steps creating this table using a Pandas dataframe from the provided input data and storing it in the variable ‘Model’. It can then be referenced by specifying ‘Model’ as the first input to the function.
* **Parameters**: This is an array containing the parameters that describe the HPWH being simulated. In the example wrapper code the user-specified parameters, described in Parameters, are added to an array called ‘Parameters’. It can then be referenced by specifying ‘Parameters’ as the second input to the function.
* **COP\_Regression**: This is a function describing the COP of the heat pump dependent upon the existing conditions in the HPWH. It is generated by combining the user-specified COP parameters into a numpy regression. In the example wrapper code this is stored as COP\_Regression, and can be referenced by specifying ‘Regression\_COP’ as the third input when calling the function.

This script can be run on a PC or a Mac.

### Wrapper Script

The following screenshots show example code used to create the input data set, call the HPWH\_Model.py function, and save the data.

Figure 2 shows the code used to specify the parameters in the example wrapper code. Text in black is the name of each parameter, and the list of parameters matches those specified in Parameters. The red text after the equals sign is the value specified for the parameter in this simulation. These values can be edited to create different starting assumptions, to simulate different HPWHs, or to simulate the HPWH in different locations. The grey text after the hashtags are comments describing each parameter, so the user can understand each line without constantly referring back to this User’s Guide.

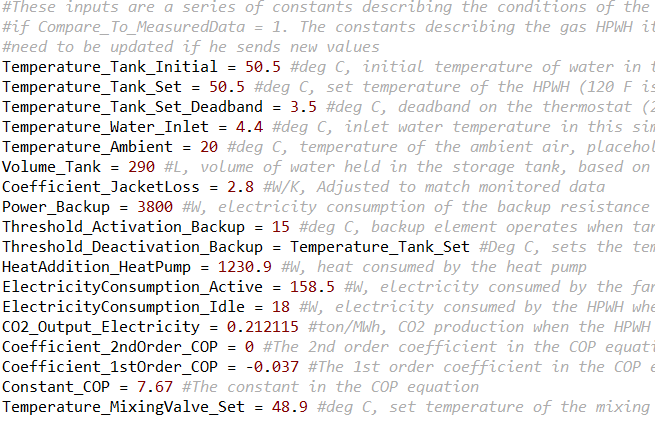


Figure 2: Specifying the parameters for HPWH\_Model.py

Figure 3 shows the code used in the example wrapper file that specifies the location of the input file data and creates the filename to be used when saving the file. The location of the input data is specified and stored in a variable called “Path\_DrawProfile” so it can later be referenced using that variable name. The remaining three lines of code specify the section of Path\_DrawProfile to use when creating the output filename, and store those characters in the variable “Filename”.

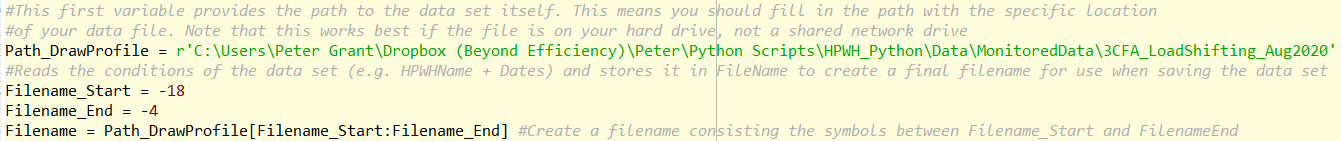


Figure 3: Specifying the input data file

Figure 4 shows the code used in the example wrapper to create the Parameters array describing the modeled HPWH. First it calculates the thermal mass of the tank, for later use in calculating the temperature change over each timestep, using the user-specified volume of the tank and constant values for the density and specific heat of water. Second it stores the user-specified parameters describing the HPWH and the calculated thermal mass in an array called ‘Parameters’. The comments with numbers after each entry in ‘Parameters’ help the user identify which entry in the array corresponds to which parameter, and helps ensure they are used correctly in the HPWH\_Model.py functions (Note: This was very important when creating and debugging the model, but is not likely to be important in a typical simulation process).

Text, letter

Description automatically generated

Figure 4: Creating the Parameters Array

Figure 5 shows some of the code in the example wrapper used to convert the user-provided data into a format that can be used by HPWH\_Model.py functions.

The first lines of code convert the user-specified parameters describing the COP of the heat pump into a regression equation. HPWH\_Model.py functions can then calculate the COP of the heat pump during each timestep using this equation and the available water temperature.

The third line reads the file previously specified in Path\_DrawProfile and stores it in a Pandas dataframe titles Draw\_Profile. This is now a tabular representation of the input data stored in the Python environment.

The fourth line creates a new Pandas dataframe that will provide the model inputs and store the outputs from HPWH\_Model.py functions. This dataframe is created by copying selected columns from Draw\_Profile. This step is critical as this is how you simultaneously gather the needed input data and limit the data in the model to avoid potentially many columns of extraneous information.

Text, letter

Description automatically generated

Figure 5: Converting Input Data to Useable Format 1

Figure 6 shows the next step in preparing the input data for use in HPWH\_Model.py functions. Note that this step will not always be necessary. Since the example provided when creating the example wrapper with in IP units this code converts all columns in the Model dataframe to SI units as required for use in HPWH\_Model.py functions.

A picture containing letter

Description automatically generated

Figure 6: Converting Input Data to Useable Format 2

Figure 7 shows the process of creating the necessary output columns and specifying data values as necessary.

As described in Inputs the functions in HPWH\_Model.py require several columns to exist in the supplied dataframe. Some of those columns (E.g. Timestep) were already available with the correct name in the data. Others were not included in the supplied data set. Figure 7 contains eight lines of code specifying that the value of each cell in columns are equal to 0. This is done to create the column so that it is available for HPWH\_Model.py functions to read and write to, while also ensuring that the dataframe is expecting numerical values.

Two of the lines describing ‘Tank Temperature (deg C)’ set the value of the first and second cells to the Temperature\_Tank\_Initial parameter specified by the user. The first row is set to Temperature\_Tank\_Initial to ensure that the model is representing the desired initial condition in the first timestep. In the second row HPWH\_Model.py calculates the total heat transfer during that timestep, then updates the tank temperature for the start of the third row. This means that the second row also needs to be set to Temperature\_Tank\_Initial.

The remaining two lines of code set the ambient temperature data and inlet water temperature data equal to the data provided in the input data set. Those two lines of code create a new column with the same data as in the input data set with the name that HPWH\_Model.py functions are expecting.

Text, letter

Description automatically generated

Figure 7: Creating Output Columns and Specifying Data Values

Figure 8 shows the code used to call the HPWH\_Model.py function and save the output data.

The first line calls the function stored within HPWH\_Model.py representing a simulation model of a HPWH with a well-mixed tank. It provides the three inputs previously described (Model provides the dataframe containing all input/output data in the model, Parameters provides the user-specified values describing the HPWH, and Regression\_COP provides the function for calculating the COP of the heat pump) and assigns the output to the Model dataframe. In this way the HPWH\_Model.py function adds new data to the dataframe and returns the result to the same dataframe.

The second line saves the Model dataframe to the user-specified filename. By default it is stored in an ‘Output’ folder within the current project folder, with a filename starting with ‘Output\_’ and ending with the user-defined portion of the input data file name. It is stored as a .csv file for future use in other Python applications or in Excel.

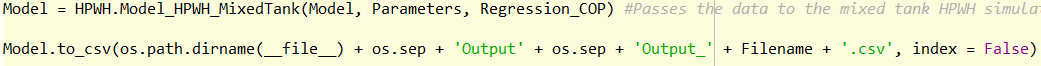


Figure 8: Calling HPWH\_Model and Saving the Data

### Input and Output Data

As described in Inputs and Output the model needs a dataframe containing the required input data and placeholder columns for output data to be supplied when calling the function. The following screenshots show portions of an example dataframe used for this purpose.

Figure 9 shows calculations preparing the time input data necessary for the model to function. Remember that the model requires a column showing the timestep in minutes to perform calculations, and the column must be labeled ‘Timestep (min). The monitoring data used in this project provided a timestamp in hour:minute format (As shown in column A), so pre-processing the data was necessary to perform the model calculations. The first step was to calculate the timestamp, which was analogous to the time since the start of the data set, in seconds (As shown in column B). That data was divided by 60 to convert the data to time in minutes (As shown in column C). Finally, the timestamp in each row was converted to a timestep between each row by subtracting the elapsed time in the previous row from the elapsed time in the current row, and stored as 'Timestep (min)' (As shown in column Q). Having both timestep and time in the dataframe enables both performing the calculations using timestep and post-processing, including plotting, using the time since the start of the simulation.

Note that the columns in an input dataframe can be in any order so long as the correct column headers are provided. Replicating the shown dataframe structure is not necessary.

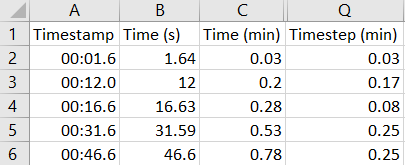


Figure : Time Calculations for the Input Data

Figure 10 shows the columns of the data frame describing the operating conditions in the simulation. Note that columns C and Q, showing the time and timestep, are included for reference. Column V presents the set temperature of the HPWH. This was necessary because the simulation included load shifting controls and the set temperature was a time-varying input, so the parameter could not be used. Column AJ shows the ambient temperature, as reported by the monitoring data. Column AK provides the measured inlet water temperature. Hot Water Draw Volume, presented in column L, provides the volume of hot water consumed during each timestep. Note that this value was calculated in the wrapper scripts by subtracting the total volume of water consumed prior to the timestep from the total volume of water consumed at the end of the timestep.

Graphical user interface

Description automatically generated with low confidence

Figure : Operating Conditions in the Input Data

Figure 11 shows an example data set output from the simulation model showing the energy calculations performed by the model. Note that columns C and Q are again retained for reference. Column AC shows the jacket losses from the tank to the surrounding environment during each time step. This heat is removed from the water stored in the tank as indicated by the fact that the example data are all negative. Column AD shows the energy withdrawn from the tank through hot water draws. This energy flow is typically removed from the water stored in the tank as well[[5]](#footnote-5). Column AE shows the energy added to the water by the backup electric resistance elements. These only operate when the water in the tank falls below the setpoint for the backup resistance elements. Column AF shows the energy added by the heat pump during each timestep. The heat pump only operates when the water temperature falls below the set temperature. Column AG shows the total energy added to the water by the heat pump and the backup resistance elements. Finally, column AI shows the total energy change during this timestep. Since the only column showing any heat transfer in this data set is the jacket losses, the total energy change is equal to the jacket losses in the figure.

Remember that, although these values are calculated by the model during the simulation, the input dataframe must include these columns. They then act as placeholders, allowing the model to enter the data as the simulation progresses. The most common approach is to create the columns in the wrapper script and fill them with zeros.

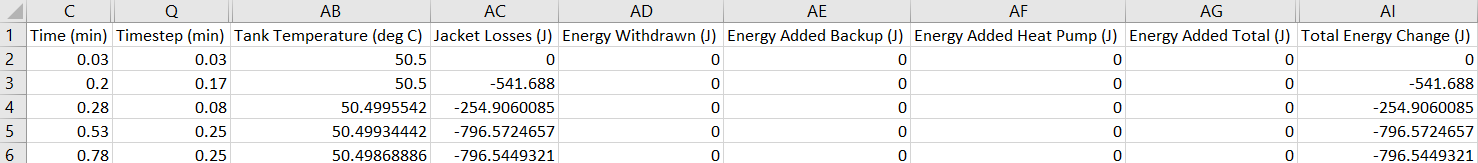


Figure : Energy Calculations Output from the Model

Figure 12 shows the calculations performed to report the electricity consumption during each timestep. Note that columns C and Q provide the time and timestep information for reference. Column AH provides the COP of the heat pump. This value is calculated from the temperature of water in the tank whether the heat pump is active or not. Column AO provides the electric power consumed by the HPWH during each timestep, calculated by 1) dividing Energy Added Total by the timestep converted to seconds, and 2) dividing again by the COP. Column AP provides the electricity consumed by the HPWH during each timestep, calculated by multiplying the electric power by the timestep and converting to kWh.

Table

Description automatically generated

Figure : Electricity Consumption Calculations Output from the Model

### Clarifications

The example wrapper code was created for use in a specific field monitoring project studying HPWHs. As a result the example wrapper contains code specific to that project, both converting the input data as needed for that project and performing calculations specific to that project. It is likely that this code will need to be replaced for every project.

Some examples of this are obvious and straightforward. Figure 6 showed code converting data in IP units to SI units. If the input data in a project is provided in SI units then this conversion will not be necessary and that code can be deleted. Alternatively, those lines could be edited to directly convert the input data to the column names required by HPWH\_Model.py functions.

Other examples are not as obvious. For example, the project for which this code was developed utilized a varying set temperature in the HPWHs. To make the simulated case match the modeled case several lines of code were added overwriting the Temperature\_Tank\_Set parameter forcing the model to use the monitored set temperature instead. This code will have to be deleted for projects where the set temperature is not being read from monitoring data. Another example occurs in the line creating “Model[‘Hot Water Draw Volume (L)’]”. Since the hot water withdrawn from the storage tank was not measured in the project it had to be calculated using other measurements. This is not likely to be the case in most simulation projects, and this code should be edited or deleted as necessary.

## Model Validation

This model has passed a preliminary validation process, as described in Model Validation Status Update Memo: Creekside Project Load Shifting HPWH Tool[[6]](#footnote-6). The model validation process was based on comparisons to field measurements taken at the Creekside multifamily project and focused on the model’s ability to correctly predict electricity consumption and timing under both static and load shifting controls.

The validation process consisted of the following comparisons:

* The modeled and monitored electricity consumption were compared during a 10 day period with no hot water draws. This test isolated the jacket losses, and enabled accurate estimation of the heat loss coefficient parameter for the monitored HPWHs.
* The modeled and monitored electricity consumption were compared during a 10 day period with typical hot water draws and no load shifting controls. This test demonstrated more typical HPWH operation and enabled accurate estimation of the deadband, verification the COP performance map, and verification the heat loss coefficient in a more complex situation.
* The electricity consumptions were compared in a single day including a very large hot water draw that triggered the electric resistance element operation. This test enabled accurate estimation of the control logic governing electric resistance element operation.
* The model results were compared to monitored data during a second 10 day period with hot water draws and no load shifting controls. This test verified that the previously identified parameters caused the model to closely match the monitored data in a second data set.
* Finally the model results were compared to monitored data in a 10 day period with hot water draws and load shifting controls. This test enabled verification that the model could accurately predict the load shifting behavior, and electricity consumption profile when load shifting controls were implemented.

The model results closely matched the monitored data in all cases. The results for each individual case are as follows:

* After adjusting the heat loss coefficient, the model results closely matched the monitored data in the heat loss coefficient test. Heating events occurred with very similar timing, and the model overpredicted the electricity consumption by 2.4%
* After adjusting the heating control parameters, the model also closely matched the 10 day period of hot water draws without load shifting. The model overpredicted the electricity consumption in the entire 10 day period by 2.2%, and by 0.018 kWh/day during a theoretical 3-9P peak period.
* After adjusting the backup resistance element control parameters, the model correctly estimated the control logic behavior and electricity consumption of the resistance elements.
* Using the pre-determined parameters, the model closely matched the second 10 day period. It overpredicted electricity consumption by 5.9% and underpredicted the 4-9 PM peak period consumption by 0.04 kWh/day.
* The model closely matched the curve when tested on a load-shifting day. It correctly captured the mid-day load-up operation, and correctly predicted 0 kWh electricity consumption during the 4-9 PM peak period. The morning heating curve didn’t match perfectly, but the total energy consumption was quite close indicating that this is an issue of timing and not of energy consumption.

The model still has a few unresolved issues which could be improved to yield better performance:

* The model sometimes engages a heating cycle earlier than the physical units. This is likely not a matter of theoretical processes and is only a matter of fine-tuning the parameters describing the specific HPWH. A rigorous calibration process, using an optimization algorithm to tune the parameters, could yield a closer fit and reduce these errors.
* The model cannot accurately predict the outlet water temperature. The trouble predicting outlet temperature is a natural consequence of the well-mixed tank assumption. In a well-mixed tank the outlet temperature is equal to the average temperature of the tank, which gradually decreases as hot water is consumed. In a stratified tank the temperature at the bottom of the tank decreases, and the temperature at the top stays relatively constant until all the water in the tank is consumed. This issue could be resolved by implementing a stratified tank model.
* The COP curve, which is based on the average temperature of water in the tank, underestimates the COP for cases where there is significant stratification during a load shifting event. When hot water is removed from the tank it is replaced with cold water at the bottom. The heat exchanger is exposed to the cold water at the bottom, thus the COP is really determined by that temperature. Since the COP curves were generated using tests at 51.7 °C (125 °F) they capture this effect. However, when this same instance occurs at a 60 °C (140 °F) the temperature difference between the COP curve input and the temperature at the bottom is greater. In that case the COP of the HPWH is greater than the COP curve predicts. This issue could be resolved by implementing a stratified tank model and developing COP curves based on the water temperature at the bottom of the tank.

# Multi-Node Model

## Assumptions and Base Principles

The multiple node model is designed to be flexible, with the ability to change parameters as needed to match the operation of different HPWHs. At this time it is programmed to emulate the control logic of the HPWHs monitored in the Creekside load shifting field study project, but control logic for other makes/models can be implemented as needed.

The main assumptions in the multi-node HPWH model are:

1. The water temperature varies with heigh in the tank, but is uniform side to side. This means that the multiple nodes in the tank each represent vertical slices encompassing the entire volume of water within that vertical slice.
2. The water flows through the tank in a plug flow manner. When a hot water draw occurs the water flows from one node to the next, and is then cooled by colder water entering from the inlet or from the lower node. This means that the water moving to the next node is hotter than if the model assumed that the volume in the node mixed fully before passing to the next node.
3. Operation of the heat pump and resistance elements employed cause enough temperature difference to cause buoyancy flow. The upper element heats all nodes in the tank above the resistance element. The lower element and the heat pump heat all nodes below a stratification layer. The stratification layer is identified based on the temperature difference between nodes.

The second and third assumptions may or may not be accurate. Prior work on piping has indicated that water tends to partially mix before moving to the next node which differs from the plug flow assumption. However, not enough data showing the temperatures inside the tank are currently available so this assumption is used for now. HPWH design also typically includes a heat exchanger wrapped around the outside of the tank, extending from the bottom to roughly ½ of the way up the tank. In cases where the stratification layer is below the top of the heat exchanger the third assumption incorrectly tracks the heat. Sections of the heat pump below the stratification layer would act as assumed, but sections of the heat pump above the stratification layer would heat the water higher in the tank instead. This assumption should be reviewed when data with more temperatures inside the tank is available.

The multi-node model is intended to be called one time step at a time, with all input data being passed as a numpy array. It then performs calculations for that timestep and returns the input data numpy array with extra data for the output data added. Some information, such as the temperature of water in each node of the tank, is stored as an attribute of the model and readily available for use in the next timestep. Between the data provided in the output and the values stored in the model enough information is provided to determine the input data for the next timestep, which enables passing input data back to the model. The tutorial provides an example of this process.

The multi-node version of Flexi-HPWH is designed to read parameters describing the operation of the HPWH from an external configuration file. This enables storing a library of files containing important parameters for HPWHs from different manufacturers and reading the correct information as needed for a specific simulation. While this process is recommended, it is also possible to hard-code the parameters into a Python script and avoid using external configuration files if desired.

The COP of the heat pump is defined using two separate second order two variable regressions. Both regressions use the rated heat addition rate of the heat pump, read from the manufacturer’s specification sheet, as a parameter and use the current tank water and ambient air temperatures as inputs. These regressions then calculate 1) the rate at which heat is added to the water during the current timestep, and 2) the electricity consumption of the heat pump for the current timestep.

## COP Regression Equations

The multi-node model uses two second order two variable regressions to define the performance of the heat pump. The two regressions use the rated heat addition rate from the manufacturer’s specification sheet as a parameter, and both the ambient air temperature and water temperature at the lower thermostat as inputs. The equations both follow the following form:

)

Where:

Two sets of coefficients need to be determined for each HPWH simulated, with one representing the heat addition rate of the heat pump and the other representing the electric demand. The coefficients are stored as a list in the configuration file.

The regressions both use the temperature of water specifically at the height of the lower thermostat, not the average temperature of water in the tank or the average temperature of water connected to an immersed heat exchanger. This is a result of the data set available for creating and validating the model, which only included the temperatures at the thermostats. These regressions could be refined in the future if testing data includes temperature measurements at different depths in the tank. That would provide better insight into how the water temperature changes with depth in the tank and how that impacts heat pump performance.

## Supporting Functions

Several supporting functions streamlining the process of performing Flexi-HPWH simulations are available and stored in separate packages which can be imported as needed. These packages and functions are as follows:

* **Set\_Temperature\_Profiles.py/get\_profile**: This function provides 24 hr set temperature profiles as needed for simulations, enabling easy simulations with different control strategies. To obtain a set temperature profile call this function with a string describing the set temperature profile. The string must match the name of one of the profiles stored in the function, and copy/pasting is recommended to avoid typos. The profile is returned as a dictionary, and can be applied to a draw profile dataframe using the .map function.
* **Installation\_Configuration.py/get\_temperatures**: This function changes the ambient air and evaporator inlet air temperatures depending on the HPWH’s installation configuration. It requires the user to pass a dataframe containing outdoor air temperature data, then it returns the dataframe with assumed ambient and evaporator inlet air temperatures. The ambient temperature is identified by applying empirical correlations showing the difference between outdoor air temperature and ambient temperature in the different configurations. The evaporator air inlet temperature is reduced if ducting is used, again based on empirical correlations. To use this function the user must specify an installation configuration that matches the options available in the function, and copy/pasting is recommended to avoid typos. The modified dataframe is then returned, providing the input data needed for multi-node Flexi-HPWH simulations.
* **CZ\_Assumptions.py/overwrite\_parameter**: This function enables overwriting outdoor air temperature or inlet water temperature using assumptions from CBECC-Res. It receives a dataframe which it then modifies and returns. Note that this does overwrite the original data so be sure you’re able to recover the original data in case of mistakes. It requires four parameters to operate. The first is the climate zone you wish to read data from. The second is the dataframe to which you want the simulation data to be added. The third is the name of the column in the original data set that you wish to overwrite. The fourth is the name of the column in the assumption data that you wish to read. The function then creates a dictionary of the assumption data with keys representing the month, day, and hour of each point. It then uses Python’s map function to apply this data to the dataframe, and returns the dataframe. To use this function the input data set for the specified climate zone must be available, so you may need to obtain and add other files as needed. It is possible to modify this function to use inputs other than CBECC assumptions.
* **HPWH\_Utilities.py/Prepare\_Creekside\_InputData**: This function converts a pandas dataframe to a numpy array and creates a column dictionary enabling interpretation of the data set. Since the multi-node version of Flexi-HPWH requires a numpy array and a column dictionary, this function is a convenient way of preparing the inputs before simulations. The function was created specifically for the Creekside project so modification may be necessary to ensure that this function can read the dataset prior to use.
* **HPWH\_Utilities.py/Simulate\_MonitoredData**: This function performs a simulation using Creekside monitored data as the inputs. It calls the correct forms of the other functions as required for a specific simulation. The advantage of this function lies in that it can be used to automatically perform large parametric studies. To do so create a list of tests to perform, a for loop iterating through the tests and extracting the required test parameters, then passing those parameters to this function. This function will perform each simulation, save the results, and store some high-level simulation result data to a summary file. Similarly to Prepare\_Creekside\_InputData this function was designed specifically for the Creekside project and will require modification before being used for other projects.

## Parameters

Several parameters describing the performance of the simulated HPWH must be supplied to use the multi-node version of Flexi-HPWH. It is recommended that these parameters be supplied using an external configuration file, saved in .json format, but so long as the data is available as a dictionary that detail is not necessary. The configuration dictionary then gets passed to Flexi-HPWH when initializing HPWH\_MultipleNodes, enabling the model to interpret and store the parameters.

The parameters that must be provided in configuration are:

* **Jacket Loss Coefficient (W/K)**: This parameter specifies the rate at which the HPWH loses heat to the surroundings based on the temperature difference between the water in the tank and the surrounding air. This parameter is divided by the number of nodes in the tank and applied evenly, creating a heat loss multiplier for each node.
* **Backup Element Power (W)**: This represents the rated power of the backup electric resistance elements. This is the electric power that will be consumed by the resistance elements and added to the water when they are active.
* **Resistance Deadband (deg C)**: This represents how far below the set temperature the water at the upper thermostat must fall before the HPWH activates second stage heating.
* **Resistance Deadband, Heat Pump Active (deg C)**: This represents the same as the prior parameter, but applies when the heat pump is already active. This parameter will not be used in other situations, and overwrites the typical resistance deadband if the heat pump is active.
* **Cutoff Temperature (deg C)**: This parameter specifies the heat pump inlet air temperature below which the heat pump will not operate. HPWHs employ cutoff temperatures to protect the heat pump from freezing when operating under cold air conditions. If the heat pump inlet temperature is below this temperature Flexi-HPWH uses only the electric resistance elements for heating and will not use the heat pump.
* **Heat Pump Activation Deadband (deg C)**: This represents how far below the set temperature the water at the lower thermostat must fall before the HPWH activates the heat pump.
* **Heat Pump Activation Deadband, Recent Set Temperature Change (deg C)**: Some HPWH controllers utilize a separate deadband when the user changes the set temperature. This parameter enables emulating that control logic by providing a separate deadband for times when the set temperature has been recently changed. If the set temperature has been changed recently Flexi-HPWH uses this deadband instead of the typical heat pump activation deadband.
* **Heat Pump Deadband Time Period (s)**: This represents the time period during which the “Recent Set Temperature Change” deadband is used for the heat pump. If the set temperature was changed more recently than this value Flexi-HPWH uses the “Recent Set Temperature Change” deadband. If not, Flexi-HPWH uses the typical deadband.
* **Heat Pump Heat Addition Rate (W)**: This is the rated heat pump heat addition rate taken from the manufacturer’s specification sheet. It is used as the base for the performance curves describing the heat pump performance.
* **Heat Rate Coefficients**: This parameter specifies the coefficients used in the performance curve stating the rate at which the heat pump adds heat to the water in the tank. The performance curve is a function of the temperature of the water at the lower thermostat and the temperature of the air entering the heat pump. The coefficients must be expressed in a list.
* **Power Coefficients**: This parameter specifies the coefficients used in the performance curve stating the electric power demand of the heat pump. The curve is a function of the temperature of water at the lower thermostat and the temperature of the air entering the heat pump. The coefficients must be expressed in a list.
* **Volume Tank (L)**: The volume of the water storage tank in the HPWH. The nominal volume of the storage tank is available on manufacturer’s specification sheets. Note that the actual volume of the tank is typically 10% lower than the rated volume.
* **Varying Set Temperature**: This parameter states whether the simulation is using a varying set temperature or a static set temperature. A “1” indicates that the set temperature varies, while a “0” indicates that the set temperature is fixed.
* **Set Temperature (deg C)**: If the simulation uses a static set temperature this parameter expresses the set temperature to use. If the simulation uses a varying set temperature this parameter is ignored. It must be provided either way to avoid errors during initialization.
* **Number of Nodes**: This parameter specifies the number of nodes in the tank to use in the simulation. As specified in “Assumptions and Base Principles” the nodes each represent a volume of water covering the area of the storage tank at a specific height.
* **Upper Thermostat Node**: The node in which the upper thermostat is located. Note that Python uses zero as the first index, so the specified upper thermostat node is lower than expected by one. For instance if the upper thermostat is in the top-most node of a 20 node tank then it is located in node 19.
* **Lower Thermostat Node**: The node in which the lower thermostat is located. Note that Python uses zero as the first index, so the lower thermostat node is one lower than expected. For example if the lower thermostat is the bottommost node of a tank that would be node zero.
* **Resistance Lockout Time (min)**: Some HPWHs employ a lockout time after the heat pump activates during which the resistance elements cannot engage. This parameter specifies that time period. For instance, if the lockout time is five minutes and the heat pump has just activated the HPWH cannot use the resistance elements for five minutes. This behavior was not identified in the monitoring data used to create Flexi-HPWH so this control logic is not currently implemented.

## Inputs

The multi-node version of Flexi-HPWH is a class representing the operation of the HPWH. It is typically used by initializing an instance of the class with the previously specified parameters, then the do\_step() function is iteratively called for each time step in the simulation. As such, use of multimode Flexi-HPWH typically requires two Python scripts. The two are:

* **HPWH\_Model.py**: This package contains the different Flexi-HPWH model options. The multi-node model is stored in the HPWH\_MultipleNodes class and calculations are performed by calling the do\_step function. This function performs the calculations for a single timestep and returns the results.
* **A wrapper script**: This script imports the hot water draw, ambient temperature, and inlet temperature profiles then formats the data as needed for use with HPWH\_Model.py. It also initializes an instance of HPWH\_MultipleNodes and passes the configuration as necessary. It is important that the wrapper script prepare the input data as necessary, then uses a for loop to iterate through each timestep in the input data and call HPWH\_MultipleNodes.do\_step. If the hot water draw volume is calculated using the temperature of water at the top of the tank it is also important that the wrapper function calculate the draw volume using the new tank temperature in each iteration of the for loop.

The input to HPWH\_MultipleNodes.do\_step during each timestep is a row of the input array for each timestep in the simulation. Note that the input row must contain both the input data and cells for the output from the timestep. This section will discuss the inputs required but it’s important to include the output columns, typically initialized to 0, when creating the input array.

The inputs required in the array are as follows. They must be stored in columns with headings matching the names specified here.

* **Set Temperature (deg C)**: This column represents the set temperature in the tank during each timestep of the simulation. It can be read from monitored data, created manually, or filled out calling Set\_Temperature\_Profiles.py/get\_profile. If the “Varying Set Temperature” parameter is set to 0 then Flexi-HPWH assumes a static set temperature matching the value in configuration and this input column is not needed.
* **Timestep (min)**: This input represents the duration of the current timestep in minutes. All calculations use this time period. Shorter timesteps yield more accurate simulations, with 15 seconds being recommended.
* **Evaporator Air Inlet Temperature (deg C)**: This input states the temperature of air entering the heat pump during each timestep in the simulation. If the HPWH is installed in a large space with no ducting it is acceptable to use the ambient air. If not, the evaporator air inlet temperature can be calculated from the using Installation\_Configuration.py/get\_temperatures.
* **Ambient Temperature (deg C)**: This is the temperature of the air surrounding the HPWH itself. It is used to calculate the standby losses from the tank and can be used to calculate the evaporator air inlet temperature.
* **Inlet Water Temperature (deg C)**: This is the temperature of the cold water entering the tank as the occupants consume hot water. It is used to calculate the heat loss in the bottom node of the tank during hot water draws.
* **Hot Water Draw Volume (L)**: This is the volume of hot water removed from the tank during the active timestep. It is used to calculate the heat transfer due to water flows in each step. If the hot water volume is calculated as a function of the temperature at the top of the tank it is important to calculate this input iteratively during each timestamp.

## Outputs

The multiple node version of Flexi-HPWH returns the provided row of the array having filled in entries corresponding to the outputs. To avoid errors it is important to include columns for these outputs in the input array. Flexi-HPWH will then fill in the cells corresponding to the outputs during the calculations, and return that row of the array. That row of data should then be used to overwrite the row in the array, thus storing the outputs in the array.

The array used to provide the inputs and store the outputs must include the following columns. Typically the columns are pre-filled with 0, though that convention is not necessary.

* **Jacket Losses (kWh)**: After the simulation this column will contain data showing the energy lost to the surrounding environment from each node during the each timestep of the simulation.
* **Energy Withdrawn (kWh)**: This output presents the energy withdrawn from each node of the tank during each timestep in the simulation.
* **Heat Added Heat Pump (kWh)**: This stores the heat added to each node during each timestep of the simulation.
* **Heat Added Backup (kWh)**: This output represents the heat added to each node during each timestep by the backup resistance elements.
* **Total Energy Change (kWh)**: This output represents the total energy change for each node during each timestep of the simulation including the jacket losses, heat pump, resistance elements, and hot water flow.
* **Node Temperatures (deg C)**: This output presents the temperature of each node in the storage tank during each timestep in the simulation.
* **Electricity Consumed Heat Pump (kWh)**: This output presents the electricity consumed by the heat pump during each timestep of the simulation. It is determined using 1) the control logic dictating whether or not the heat pump heats the water during each timestep, 2) the coefficients describing the performance of the heat pump supplied as a parameter, and 3) the current ambient and tank water temperatures.
* **Electricity Consumed Resistance (kWh)**: This provides the electricity consumed by the resistance elements during each timestep. It is determined using 1) the control logic determining whether or not the electric resistance elements are operating during a given timestep, and 2) the rated power of the resistance elements provided as a parameter.
* **Electricity Consumed Total (kWh)**: This output provides the total electricity consumed by the HPWH during each timestep of the simulation. Since the on-board controls are not currently tracked by Flexi-HPWH this output includes the heat pump and resistance element electricity consumption.
* **Total Jacket Losses (kWh)**: This output represents the jacket losses of the entire tank during each timestep of the simulation. It is the sum of the values provided by the Jacket Losses (kWh) output.
* **Total Energy Withdrawn (kWh)**: This provides the total energy withdrawn from all of the nodes in the tank during each timestep. It is the sum of the values provided by Energy Wightdrawn (kWh).
* **Total Heat Added Heat Pump (kWh)**: This output presents the total amount of heat added to the tank by the heat pump during each timestep in the simulation. It is the sum of the values for each node provided in Heat Added Heat Pump (kWh).
* **Total Heat Added Backup (kWh)**: The total heat added to the tank by the backup resistance heating elements during each timestep in the simulation. This is the sum of the values provided in Energy Added Heat Pump (kWh).
* **Total Heat Added (kWh)**: This is the total heat added to the tank during each timestep by timestep by both the heat pump and the resistance elements. It is the sum of Total Heat Added Heat Pump (kWh) and Total Heat Added Backup (kWh).
* **Node Energy Change (kWh)**: The energy change in each node in the tank during each timestep in the simulation. This is in the form of a list. The energy change is the sum of the impacts of the jacket losses, water flows, heat pump heat addition, and resistance element heat addition.
* **Heat Pump Heat Addition (kW)**: This presents the rate at which the heat pump added heat to the tank during each timestep in the simulation. It is equal to the rated heat pump power times the regression defined by the heat rate coefficients for the current water temperature and ambient air temperatures.

## Configuring the Simulation Model

Flexi-HPWH uses several parameters, as previously described, to set the thermal and control characteristics of the studied HPWH. In this way it is able to quickly estimate the performance of HPWHs from different manufacturers, with different storage tank volumes, and with different size compressors. This flexibility gives it the ability to support projects deriving recommendations on which heat pump or control approach will work best in any given installation.

To configure the multi-node version of Flexi-HPWH you must supply the parameters as a Python dictionary. This can either be done directly in the code, or it can be stored as a .json file which the code reads as a dictionary. Storing the configuration as a .json file is recommended, as it is then possible to store multiple configurations representing different models of HPWH. Comparative analysis then only requires loading different configuration files.

Figure 13 presents an example configuration file as is needed to describe the parameters previously listed. This file is meant to show the structure of the configuration file only, and all values have been replaced. To simulate a specific HPWH each of these parameters must be tuned based on the thermal characteristics and control logic displayed by the simulated HPWH. The standard distribution contains a configuration file for the HPWHs monitored in the Creekside project.

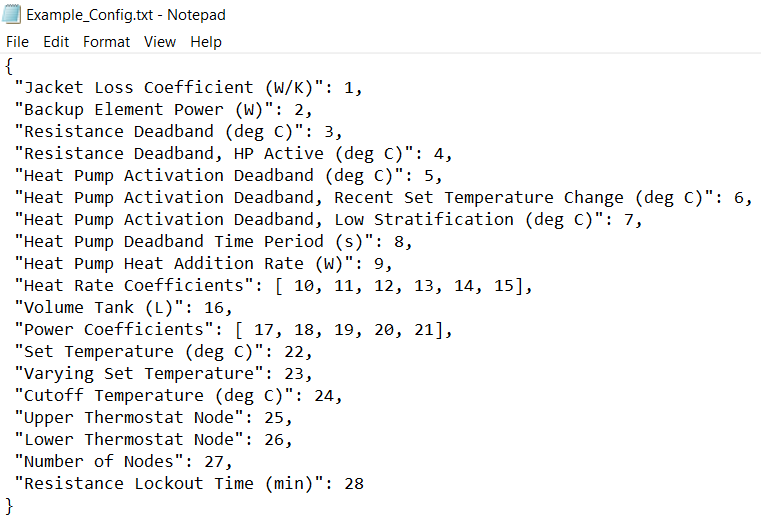
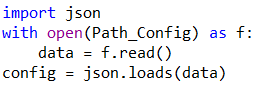


Figure 13: Example Configuration File

Once the configuration file is created and saved, the next step is to read it into the simulation model and stored as the configuration dictionary. This can be done with the following code.



## Preparing the Input/Output Array

Typically the input/output data array is created as a Pandas dataframe, then converted to a numpy array. The pandas dataframe can be created by reading a spreadsheet, creating the dataframe within the script, or calling the supporting functions to provide the necessary columns and data. This User’s Guide assumes familiarity with Python and Pandas.

It is then necessary to both convert the dataframe to a numpy array and to add a column index dictionary to the configuration dictionary. Converting the dataframe to a numpy array dramatically increases the simulation time. Adding the column index dictionary to the configuration tells the model how to find the columns representing each input. Note that the column index must be added to the configuration before initializing the model.

The column index dictionary can be created and added to the configuration using the following code. This example assumes that the dataframe is called “input\_data”.



The dataframe can be converted to a numpy array using the following code. This example assumes that the dataframe is called “input\_data”.



Note that in this example the code converts input\_data to a numpy array without changing the name. If following this example its necessary to create the column index dictionary before converting the dataframe to a numpy array, as the columns headings needed to create the dictionary will no longer exist after conversion to an array.

## Performing a Simulation

Once the configuration file and input data are created correctly it is possible to run a simulation. To do so use the following steps:

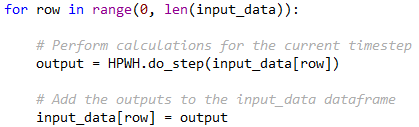
1. Initialize the model with the configuration file,
2. Iterate through the input data set and perform a simulation on each timestamp.
3. Store the output data from each timestamp in the output data array.
4. Convert the output data from a numpy array to a pandas dataframe, and interpret the results.

To initialize the model you need to import the HPWH\_MultipleNodes class and create an instance of it. HPWH\_MultipleNodes is stored in the HPWH\_Model library, so this can be accomplished with the following code.



Now the ‘HPWH’ variable represents a HPWH as defined by the provided configuration file.

The next task is to perform the calculations for each timestamp in the input data set and store the output in the array for later analysis. This can be done using the following code.



Note that each call of .do\_step() returns the provided input data with additional entries for the output data. As a result overwriting the current row of input\_data with the output simultaneously retains the input data and adds the output data.

After the simulation has performed calculations for each timestamp the last step is to convert the numpy array back to a more user-friendly pandas dataframe. This can be done with the following command.



This command provides a dataframe named ‘result’ containing all input and output data from the simulation. It can be saved, viewed, or plotted as needed to interpret the data.

## Acquiring Further Information

This documentation provides a high-level overview and general user’s tutorial for performing initial simulations using the multi-node version of Flexi-HPWH. More detailed information describing the control logic and specifics of the calculations can be obtained by reading the code itself. The code is very well documented, providing both explanations of what the code is doing and insight into why the code functions the way it does.

# Acquiring the model files

The model files are currently hosted on Beyond Efficiency’s GitHub page and can be found at <https://github.com/PeterGrant/Flexi-HPWH>.

# Appendix A: Available Set Temperature Profiles

Table 1 presents the set temperature control strategies available for use in simulations. Each row represents a different control strategy. The “Load Shifting Hours” column specifies the hours during which the set temperature is increased to implement load shifting controls. The “Elevated Temperature (°C)” column specifies the maximum temperature employed during the load shifting hours in the control strategy. The “Default Temperature (°C)” column states that set temperature used outside of load shifting hours, and represents the typical set temperature in that dwelling. The “Notes” column specifies details about how the load shifting control strategy is implemented.

Table : Available Set Temperature Control Strategies



Figures 13 – 15 present the set temperature profiles available in Flexi-HPWH. They present the set temperature during each hour of the day for each profiles. Figure 13 shows the available non load shifting profiles, Figure 14 shows the available non-stepped load shifting control profiles, and Figure 15 shows the available stepped load shifting control profiles.

A picture containing chart

Description automatically generated

Figure : Available Non-Load Shifting Set Temperature Profiles

Chart, line chart

Description automatically generated

Figure : Available Non-Stepped Load Shifting Set Temperature Profiles

Chart, line chart

Description automatically generated

Figure : Available Stepped Load Shifting Set Temperature Profiles

# Appendix B: Available Installation Configurations

**Outdoor**: This represents a HPWH that is installed outside of the building, exposed to the outdoor conditions with adequate airflow across the evaporator. It specifies simulations using the following assumptions:

* Ambient temperature = Outdoor temperature
* Evaporator air inlet temperature = Ambient temperature

**Closet\_AdequateAirflow**: This option represents a HPWH that is installed in a closet with adequate airflow across the evaporator. It assumes that the jacket losses from the HPWH, solar gains and thermal mass effects all impact the air temperature in the space. At the same time, this option assumes that the evaporator receives adequate airflow and no de-rating is required. This option specifies the following assumptions based on field monitoring observations:

* Ambient temperature = Outdoor temperature + regressions for each month and hour of the day representing the temperature difference between outdoor air and closet air temperatures
* Evaporator air inlet temperature = Ambient temperature

The data describing the regressions estimating the air temperature difference are provided in TCloset–Toutdoor\_C.csv

**Open\_Area**: This option represents a HPWH that is installed in a large space with adequate airflow. The space is large enough that the ambient temperature is not impacted by either the HPWH’s jacket losses or the heat pump cooling effect. This case represents a case similar to a HPWH being installed in a garage.

Ambient temperature = Either measured or assumed ambient temperature

Evaporator air inlet temperature = Ambient temperature

**Unducted\_Closet**: This option represents a HPWH that is installed in a small exterior closet with no ducting. It assumes that the jacket losses from the HPWH, solar gains, and thermal mass effects all impact the closet space temperature. Additionally, it assumes that the exhaust from the heat pump is not ducted and cools the space. It uses the following assumptions based on field monitoring observations:

* Ambient temperature = Outdoor temperature + regressions for each month and hour of the day representing the temperature difference between outdoor air and closet air temperatures – 6.1 deg C
* Evaporator air inlet temperature = Ambient temperature

**Ducted\_Exhaust**: This option represents a HPWH that is installed in a small exterior closet with the exhaust ducted to the outdoor environment. It assumes that the jacket losses from the HPWH, solar gains, and thermal mass effects all impact the closet space temperature. It assumes that the exhaust from the heat pump is ducted to the outdoor conditions, avoiding cooling the space while limiting the airflow (and energy available) across the evaporator. It uses the following assumptions based on field monitoring assumptions:

* Ambient temperature = Outdoor temperature + regressions for each month and hour of the day representing the temperature difference between outdoor air and closet air temperatures
* Evaporator air inlet temperature = Ambient temperature – 4.17 deg C

**StandardAttic**: This option represents a HPWH that is installed in a typical standard attic of a single family home. The simulation assumes that the attic air temperature is impacted by the standby jacket losses of the HPWH but not by the cooling effect of operating the heat pump. The cooling effect probably does impact closet temperatures, but data showing that relationship and enabling simulation assumptions is not currently available. This setting does not use ducting, so there is no evaporator air inlet temperature reduction. It uses the following assumptions:

* Ambient temperature = Outdoor temperature + regressions for each month and hour of the day representing the temperature difference between outdoor air and standard attic air temperatures
* Evaporator air inlet temperature = Ambient temperature

**Ducted\_StandardAtticInlet**: This option represents a HPWH installed in a small closet with inlet air ducted to the HPWH from a standard attic, and exhaust air ducted out of the HPWH. It assumes that the jacket losses impact the closet air temperature, but not the inlet air temperature. The evaporator air inlet temperature is reduced by the ducting. This setting uses the following assumptions:

* Ambient temperature = Outdoor temperature + regressions for each month and hour of the day representing the temperature difference between outdoor air and closet air temperatures
* HPWH supply temperature = Outdoor temperature + regressions for each month and hour of the day representing the temperature difference between outdoor air and standard attic air temperatures
* Evaporator air inlet temperature = HPWH supply temperature – 4.17 deg C

**HPAttic**:

This option represents a HPWH that is installed in a high performance attic of a single family home. The simulation assumes that the attic air temperature is impacted by the standby jacket losses of the HPWH but not by the cooling effect of operating the heat pump. The cooling effect probably does impact closet temperatures, but data showing that relationship and enabling simulation assumptions is not currently available. This setting does not use ducting, so there is no evaporator air inlet temperature reduction. It uses the following assumptions:

* Ambient temperature = Outdoor temperature + regressions for each month and hour of the day representing the temperature difference between outdoor air and high performance attic air temperatures
* Evaporator air inlet temperature = Ambient temperature

**Ducted\_HPAtticInlet**: This option represents a HPWH installed in a small closet with inlet air ducted to the HPWH from a high performance attic, and exhaust air ducted out of the HPWH. It assumes that the jacket losses impact the closet air temperature, but not the inlet air temperature. The evaporator air inlet temperature is reduced by the ducting. This setting uses the following assumptions:

* Ambient temperature = Outdoor temperature + regressions for each month and hour of the day representing the temperature difference between outdoor air and closet air temperatures
* HPWH supply temperature = Outdoor temperature + regressions for each month and hour of the day representing the temperature difference between outdoor air and high performance attic air temperatures
* Evaporator air inlet temperature = HPWH supply temperature – 4.17 deg C

1. http://www.bwilcox.com/BEES/docs/Ecotope%20-%20Heat%20Pump%20Water%20Heater%20Model%20Validation%20Study.pdf [↑](#footnote-ref-1)
2. https://gti.sharefile.com/share/view/sfa072b77ab947549 [↑](#footnote-ref-2)
3. https://github.com/EcotopeResearch/HPWHsim/blob/master/doc/Guide\_to\_HPWHsim\_v1\_1.pdf [↑](#footnote-ref-3)
4. These are the variable names used in the Python code. [↑](#footnote-ref-4)
5. This energy would be added to the water in the tank only in the extremely rare situation where the inlet water temperature is higher than the temperature of water in the tank. [↑](#footnote-ref-5)
6. P. Grant, M. Hoeschele. 2020. Model Validation Status Update Memo: Creekside Project Load Shifting HPWH Tool. Submitted to PG&E. [↑](#footnote-ref-6)