THE UNIVERSITY *of* EDINBURGH
School of Engineering

Simulation of a Heat Pump and Hot Water Storage System

Computational Methods and Modelling 3 (Group 15)

Authors and Contributions

Name	**Contribution**	(%)
Gregor Peters	Domestic hot water model, GUI, simulation scripts and formulae, building analysis, report write-up	35
Neil Sinclair	Simulation scripts and formulae, code restructuring and efficiency improvements, GUI, report write-up	35
Lewis Gambles	Report write-up and formatting	10
Yuzhu Xiao	Report write-up and early GUI	10
Ayat Kuri	Report write-up and Data collection	10

November 2024

1 Introduction

This project explores the performance and application of an air-to-water heat pump system integrated with a hot water tank through the simulation of a simplified system model. The simulation replicates the operational and thermal behavior of the system, providing key performance metrics to evaluate its efficiency and effectiveness under user-specified conditions.

The Python-based simulation employs numerical methods designed to address the dynamic and constrained behavior of the heat pump system. These methods were carefully chosen to accurately model and predict the system's thermal and operational performance.

The model was applied in an engineering context to analyze system performance across various building types, accounting for differences in insulation, setpoint temperature, and system sizing. The model was further enhanced to simulate the heat pump's demand under typical domestic hot water usage during peak periods, improving its realism and practical applicability.

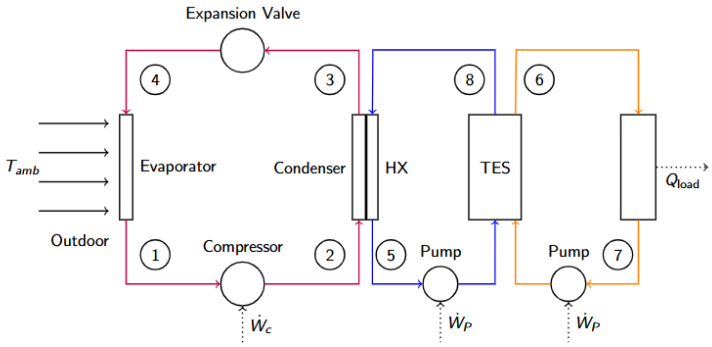

Figure 1: System diagram

2 User Interface Design and Implementation

The graphical user interface (GUI) offers customizable simulation parameters within the "Simulation Parameters" tab, enabling users to configure the model for different building characteristics and system setups. Additionally, the GUI provides an option to switch between the simplified simulation model and the enhanced model, which incorporates domestic hot water (DHW) usage effects for greater realism.

The GUI features a tabbed structure that organizes the simulation outputs based on the user-defined inputs. These outputs are presented as visual plots (See Figure 2) and scalar metrics, detailing the thermal behavior of the tank and heat pump over time. Performance metrics are computed and displayed, allowing users to conduct a thorough analysis under the selected parameters.

The implementation of the GUI setup is critical for maintaining computational performance, especially when re-running the simulation multiple times within a single session. To optimize performance, the GUI uses Python's class functionality to ensure that only parameters affected by user changes are recalculated, while static variables remain unchanged. This design minimizes unnecessary computations and ensures the smooth execution of the simulation model.

3 Model Evaluation and insights

3.1 Model Structure

The Python simulation model is organized into four primary folders, each serving a specific purpose in the simulation workflow:

Docs Folder: This folder contains essential documentation for the operation of the simulation model, including information about required Python libraries and installation guidelines. It serves as a reference for users to ensure proper setup before running the simulation.

GUI Folder: This folder houses the Python scripts that define the setup and functionality of the graphical user interface (GUI). These scripts enable user interaction and are accessed by the main script upon execution, ensuring seamless integration between the interface and the simulation.

Simulation Folder: This folder contains the core simulation scripts, including both the simplified model and the enhanced model incorporating domestic hot water (DHW) usage. It also includes the thermodynamic and mathematical formulae required for the simulation calculations, which are accessed by the simulation scripts.

Data Folder: This folder holds configuration data, such as user-defined parameters and manufacturer COP data, stored in YAML format. It also includes the JSON file that contains the simulation results. The data in this folder is accessed and modified by the GUI and simulation scripts, facilitating dynamic interaction and versatile output storage.

3.2 COP data fitting

The empirical relationship for the COP of the heat pump is given as:

$$\text{COP} = a + \frac{b}{\Delta T}$$

where ΔT represents the temperature difference between the outdoor temperature (retrieved using the Meteostat Python package) and the condenser temperature (specified in the input data file).

The unknown parameters a and b were determined using SciPy's `curve_fit` feature, which minimizes the sum of squared differences between the observed noisy COP values and the predicted values from the empirical formula. This approach identifies the optimal parameters by iteratively adjusting them to achieve the best fit.

The nonlinear nature of the relationship between COP and ΔT necessitates the use of a robust curve-fitting algorithm. This ensures that the parameters a and b are estimated accurately, capturing the system's physical behavior while accounting for noise and variability in the data.

While not utilized within the current simulation, the `getCOPparameters()` function provides the option to calculate the standard error of the fit by computing the square root of the diagonal elements of the covariance matrix. This feature could be useful for assessing the reliability of the fitted parameters.

The parameters a and b are subsequently passed as arguments to the Euler ODE solver function, enabling the dynamic calculation of the COP over the simulation period.

3.3 Tank dimension estimation

In order to calculate the heat loss of the tank the surface area of the tank the volume of the tank is first estimated using the mass of the water in the tank and using an idealised density of water taken to be 1000kg/m3(assuming a cylinderical shape). The volume is calculated as follows:

$$V = \frac{m}{\rho}$$

Subsequently the height and radius of the tank can be found:

$$r = \left(\frac{V}{\pi \cdot \text{RHratio}} \right)^{\frac{1}{3}}$$

$$h = \text{RHratio} \cdot r$$

Using the estimated tank dimensions the surface area is calculated:

$$A_{\text{total}} = 2\pi rh + 2\pi r^2$$

Finally the total area which heat is lost over is found by subtracting the area over which the condenser is providing heat:

$$A_{\text{loss}} = A_{\text{total}} - A_{\text{condenser}}$$

This value is stored as Atank and is passed into the Euler ODE solver for calculation of heat loss.

3.4 Solving the energy balance of the water tank

The thermal behaviour of the water tank is given by the following ordinary differential equation:

$$\frac{dT_{\text{tank}}}{dt} = \frac{Q_{\text{transfer}} - Q_{\text{loss}} - Q_{\text{load}}}{C_{\text{thermal}}}$$

Where Q_{transfer} is the heat supplied from the heat pump, Q_{loss} is the heat loss from the tank, Q_{load} is the heat load of the building, and C_{thermal} is the thermal capacity of the tank.

In order to solve this initial value problem for the tank temperature , the employment of a numerical ODE solving technique is necessary. For the finalised simulation model the simple Euler method was chosen.

Previous iterations of the model utilised SciPy's solve ivp() functionality which allows use of the Runge-Kuttra numerical techniques however it was found using this method that the operating condition of the heat pump was not adequatley handled due to the nature of solve ivp's adaptive step size feature.

Adaptive step sizing leads to non-linear evaluation of time points unless an incredibly small time step is defined , creating a huge computational demand. This effectively means that using solve ivp , the solver will jump backwards and forwards between previous and future time points , leading to incorrect evaluation of the heat pumps current operational status. A solution to this problem was explored by implementing peak flagging logic , and time range identification which in theory would correct this issue , however it added far more complexity than necessary.

Instead of using SciPy's ODE solving features it was decided to define a simple Euler ODE solving method , which uses a fixed time-step and allows for simple

handling of the heat-pump operation status , and is adequate for this scenario where high precision isn't necessary and it allows for a higher computational efficiency.

The mathematical process of the Euler ODE solver is as follows:

$$T_{\text{tank},n+1} = T_{\text{tank},n} + h \cdot \frac{Q_{\text{transfer}} - Q_{\text{load}} - Q_{\text{loss}}}{C_{\text{thermal}}}$$

where T_0 is set to the initial tank temperature defined in the input file, and h is the time step calculated as:

$$h = \frac{\text{Time Frame}}{\text{Number of Points}}$$

Where Q_{loss} and Q_{load} are calculated dynamically, and Q_{transfer} is only calculated when the heat pump is on. The heat pump status control is outlined below:

$$\text{Pump Status} = \begin{cases} \text{Off}, & \text{if } T_{\text{tank},n} \geq T_{\text{off}} \\ \text{On}, & \text{if } T_{\text{tank},n} \leq T_{\text{on}} \end{cases}$$

3.5 Simulation output

Running the simulation under the default system configuration defined in the brief will result in the heat pump never turning off as it is thermodynamically impossible for the tank temperature to match the condenser temperature due to heat loss from the tank. instead the default temperature off threshold was adjusted to 54 °C. The outputs of the simulation model are shown in Figures 2-4.

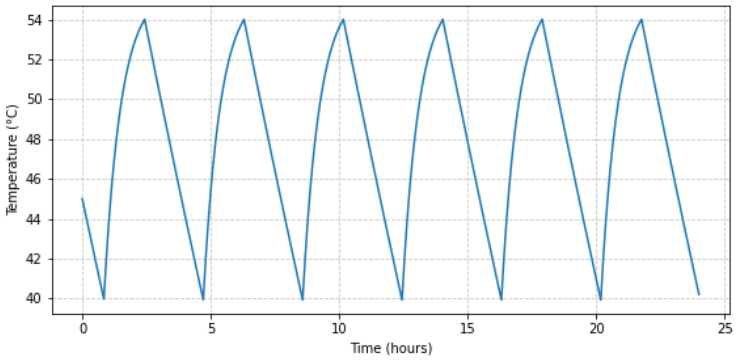

Figure 2: Tank temperature variation under default system configuration

4 Performance tuning of selected building types

The simulation model was used to simulate the behaviour of a heat pump and hot water tank heating

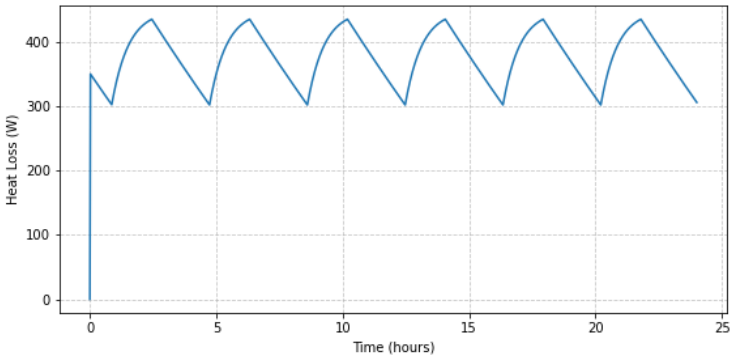

Figure 3: Heat loss variation under system default configuration

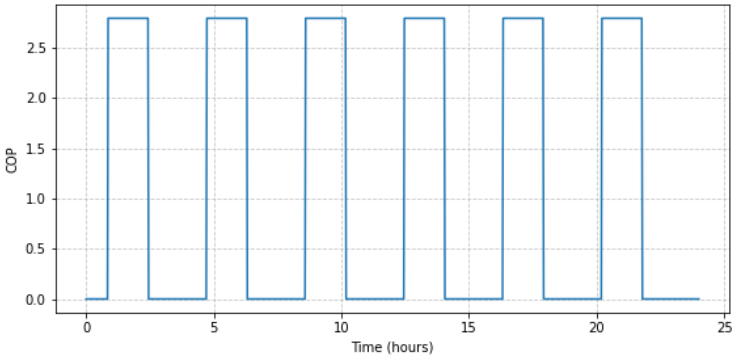

Figure 4: COP variation under default system configuration

system for the three different building types shown in table 1.

Type	**Roof Area (m^2)**	**Wall Area (m^2)**	$\mathbf{U_{wall}}$	$\mathbf{U_{roof}}$	$T_{\mathbf{sp}}$ (°C)
Apartment	60	80	0.25	0.2	20
1-floor house	120	100	0.3	0.25	25
Family sized house	150	140	0.35	0.3	30

Table 1: Building Characteristics for Different Types

The simulation model ran using these building characteristics under the default configuration values for the heat pump provided the following metrics of performance in table 2.

4.1 Analysis and Recommendations

The low thermal efficiency of the apartment building type suggests that the heat load is relatively small compared to the heat supplied by the heat pump. To better

Type	$\mathbf{E_T}$ (kJ)	$\mathbf{Q}_{\text{hp,max}}$ (W)	**TE** (%)
Apartment	6.7	6681.2	48.98
1-floor house	11.7	6671.0	75.0
Family sized house	22.0	6725.0	80.88

Table 2: Energy consumption , maximum heat pump output and thermal efficiency of selected building types

align the tank size with the load demand and reduce energy consumption, it is recommended to use a smaller hot water tank with a lower capacity, such as 100 kg.

The 1-floor house exhibits a moderate energy demand that could be reduced by improving roof and wall insulation. This would lower the building's heat load and reduce the required heat from the heat pump. For instance, setting the wall U-value to 0.2 and the roof U-value to 0.15 would optimize energy efficiency.

The family-sized house has the highest energy demand due to its larger size. Enhancing roof and wall insulation is recommended to decrease the heat load, allowing for a smaller tank. Additionally, lowering the off-threshold temperature to around 52°C would further minimize energy consumption. However, the most significant impact would come from reducing the indoor setpoint temperature in conjunction with improved insulation and a downsized tank. For example, setting the wall U-value to 0.25, the roof U-value to 0.2, the tank capacity to 150 kg, and the indoor setpoint temperature to 25°C would achieve reduced energy consumption and improved thermal efficiency.

Type	$\mathbf{E_T}$ (kJ)
Apartment	6.7
1-floor house	9.46
Family sized house	12.63

Table 3: Energy consumption of building types using suggested improvements

5 Model improvements

5.1 Domestic hot water usage model

The simulation model was improved to model the effect of heat loss of cold water entering the hot water tank during peak hot water usage hours in the morning and evening when completing everyday tasks such as running a shower or washing dishes.

The number of times per day hot water is used is modelled using a poisson's distribution , which is suitable as it assumes the events are independent and allows for future customisability for different household sizes.

$$P(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

(Where k is the number of events and λ is the expected number of events). The morning and evening peaks can be modeled as normal random variables with a mean start time of 7am and 7pm , and a standard deviation of 1 hour.

$$t_{\text{start}} \sim \begin{cases} \mathcal{N}(7am, 1hour) & \textbf{morning peak} \\ \mathcal{N}(7pm, 1hour) & \textbf{evening peak} \end{cases}$$

The duration of each event is modelled using an exponential distribution using an average event duration of 5 minutes to replicate the independence of event duration to reflect the higher frequency of short duration events.

$$d_{\text{event}} \sim \text{Exp}(\frac{1}{300})$$

The mass flow rate for the hot water usage events is modelled as a uniform distribution assuming all events are equally likely. However this could be later improved to model the frequency of running a tap compared to a shower for example.

$$\dot{m}_{\text{DHW}} \sim \mathcal{U}(0.3, 0.5)\,\text{kg/s}$$

This model is then added to the time series to provide mass flow rate values for each time point. The energy balance of the hot water tank now includes the heat loss effect from domestic hot water usage:

$$\frac{dT}{dt} = \frac{Q_{\text{transfer}} - Q_{\text{load}} - Q_{\text{loss}} - Q_{\text{DHW}}}{C_{\text{thermal}}}$$

Where the heat loss effect is calculated dynamically using the same Euler method used before with the following formula:

$$Q_{\text{DHW}} = \dot{m}_{\text{DHW}} \cdot c_p \cdot (T_{\text{tank}} - T_{\text{inlet}})$$

Adjusting the average number of events , duration and inlet tank temperature (from the water supply)

will determine the aggressiveness of this model. And by running the simulation using this new model there is a much larger demand on the heat pump during the peak times which is observed from huge tank temperature drops (see Figure 5). In order to allow the heat pump to react quick enough it was found that increasing the off temperature threshold was favourable. If this model was developed further , a suitable improvement would be to put the on and off temperature thresholds on a timer.

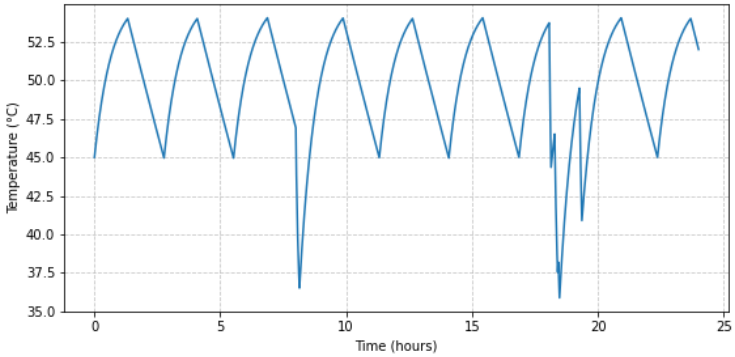

Figure 5: Temperature variation with DHW model using on/off thresholds of 45 °C and 54°C respectively

5.2 Optmisation algorithm

Another improvement that was considered for the simulation was incorporating an optimisation algorithm that would allow the user to specify a weighting of priority to energy consumption , thermal comfort (maintenance of the set point temperature) , and insulation cost to optimise the heat pump configuration and building insulation values for the user's building properties.

This would use the weighted sum method (see [1]):

$$F = w_1 f_1(x) + w_2 f_2(x) + \cdots + w_n f_n(x)$$

Where F is the total weighted objective, w_i is the weight assigned to the i-th objective, $f_i(x)$ is the i-th objective function to be optimized, and x represents the decision variables of the optimization problem.

Developing this in python would involve defining an objective function which takes in user defined priority weighting , the simulation parameters to optimise and outputs the objective function value which is a function of the total energy input , thermal comfort (the unmet load demand) and the insulation cost function which would be determined by a specified penalty of choosing costlier insulation.

This function could then be minimised using the SciPy optimise submodule from the SciPy library. However the method of minimisation would have to allow user defined constraints to avoid physically nonsensical values.

References

[1] ScienceDirect, "Weighted sum method," n.d. Available at: `https://www.sciencedirect.com/topics/computer-science/weighted-sum-method`. Accessed: November 21, 2024.