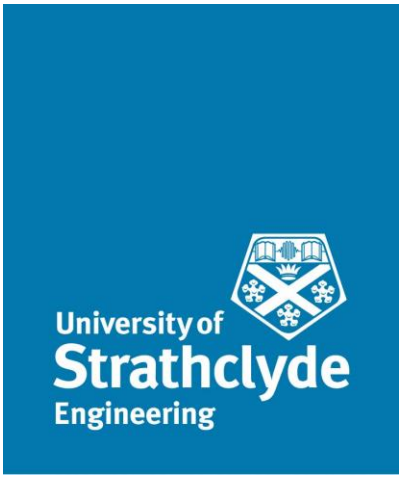
Department of Mechanical and Aerospace Engineering



The design and simulation in Python of a model predictive control system to maximise use of local renewables in a heat network with thermal storage.

Author: Richard Lane

Supervisor: Dr Paul Tuohy

A thesis submitted in partial fulfilment for the requirement of the degree

Master of Science

in

Sustainable Engineering: Renewable Energy Systems and the Environment

2019

Copyright Declaration

Signed : 

Richard Lane

Date : 22nd August 2019

Abstract

Increasing penetration of intermittent renewable generation within a power grid brings challenges with physical constraints, especially in remote locations. Energy storage is one approach to addressing this, and the possibility of providing this in thermal form offers an opportunity to create low-costs systems which respond both to the needs of energy consumers and the conditions on the local power grid.

Model Predictive Control (MPC) is a method of intelligently controlling an unpredictable system to meet multiple objectives. In this study an MPC algorithm is developed in the Python programming language to control a small district heat network with thermal storage. A development at Findhorn, an eco-village on the North Moray coast, is used as the inspiration for a modelled district heat network. Code is created to model the demand and renewable energy generation on the site, as well as model of the heating system.

The resulting Python library, named PyREmatcher, is coded in such a way as to provide a flexible structure that may be used for subsequent development, and made available on an open-source basis through the GitHub platform.

This algorithm is tested in a number of simulations of the Findhorn development to determine its performance. It is found to be very successful in reducing the amount of energy that needs to be imported from outwith the local private wire network to meet need for space and hot water heating, with all simulations tested reducing imported electricity to zero.

Possibilities for the future development of the system are discussed and recommendations are made for future development, including the implementation of this system in open source hardware.

Acknowledgements

This thesis draws heavily on the ongoing relationship between the Energy Systems Research Unit at the University of Strathclyde and the Findhorn Foundation community, and thanks are due to all who maintain this mutually beneficial relationship. In particular, I would like to thank Paddy Atkinson of the Findhorn Foundation and College for his rapid and helpful support in providing me with details of infrastructure and plans for North and West Whins. I have received generous support from many Strathclyde academics; in particular I would like to thank Graeme Flett, Andrew Lyden for getting me started with heat pump and demand modelling, and my supervisor Dr Paul Touhy, whose energy and enthusiasm for renewable energy and the civilisation-wide project to decarbonise our economy has been central to the rich experience of so many students on this course. This thesis stands on the shoulders of the ORIGIN project, and thanks are due to all who contributed to this important piece of work.

The partnership of an industrially-minded Department of Mechanical and Aerospace Engineering and a remote, spiritually-minded retreat on the Moray coast is not an obvious one but long may the association, with its spirit of joint enquiry and experimentation, persist - to the benefit of all.

Table of Contents

(continues)

List of figures

List of abbreviations

ASHP - Air source heat pump
BDST - Biomass Decision Support Tool
COP - Coefficient of performance
CH - Central heating
DHI - Direct Horizontal Irradiance
DNI - Direct Normal Irradiance
DHS - District Heating System
DHW - Domestic hot water
DSR - Demand side response
FWP - Findhorn Wind Park
GHI - Global Horizontal Irradiance
HWT - Hot water tank
LMT - Logarithmic Mean Temperature
MPC - Model predictive control
NFD - New Findhorn Directions
OEM - OpenEnergyMonitor
ORIGIN - Orchestration of Renewable Integration Generation In Neighbourhoods
PET - Parkland Ecovillage Trust
PV - Photovoltaic
RE - Renewable energy
SAP - Standard Assessment Procedure
SA:V - Surface area to volume ratio
SPF - Seasonal performance factor
TES - Thermal energy storage
VPP - Virtual power plant
WSHP - Water source heat pump
WWSHP - Wastewater source heat pump

1. Introduction

1.1. Background

In June 2019, both the UK and Scottish Parliaments passed legislation mandating a movement to net zero carbon emissions by 2050. Whilst considerable progress has been made toward the decarbonisation of the electricity supply, the supply of heat is still predominantly derived from fossil fuels, with over 85% of residential buildings in Great Britain using grid-supplied natural gas as a heating source (Committee on Climate Change, 2016). The UK Government has announced plans to prohibit the installation of gas boilers in new built domestic properties from 2030 (HM Treasury, 2019), with their role likely to be filled with technologies such as hydrogen boilers and heat pumps. With 80% of a typical UK household's energy consumption being for the provision of heating and hot water (Haslett, 2019), the increased electrification of heat - as well as the parallel effort in the transport sector - is expected to significantly increase the demands made on the electricity grid, exacerbating the stress experienced during periods of peak consumption.

Where an electricity grid has a high penetration of renewable energy (RE), the non-dispatchable nature of RE generation further exacerbates such stresses. The mismatch between irregular demand patterns and the stochastic generation patterns of RE sources is principally addressed by two mechanisms: energy storage and demand-side response (DSR), usually taking the form of peak-shifting. The electrification of heat provides an opportunity to offer both of these mechanisms cheaply and efficiently, using well-established technology.

The use of a large thermal mass to store heat has occurred since prehistory, and water has been used as a storage and transmission medium for heat for centuries. A well-insulated thermal energy store (TES) is both a vastly cheaper and more efficient means of storing energy than an electrochemical battery or pumped hydropower facility, and does not require high-cost (and potentially high-impact) materials to make. Figure 1, from a paper by Lund et al (2016) illustrates the comparison with pumped hydropower storage. The same evaluation found that the investment cost of an electrochemical battery (the Tesla PowerWall) per MWh storage is around 8 times that of pumped hydro storage.

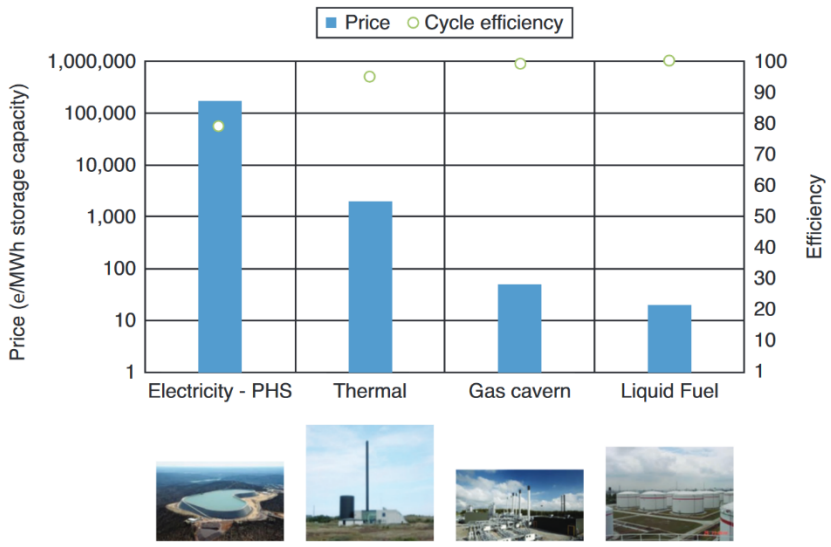


Figure 1: Investment cost and cycle efficiency comparison of electricity storage (as pumped hydropower), thermal energy storage, gas cavern storage and liquid fuel. From Lund et al (2016)

Another valuable function of a TES is to act as a shorter-term buffer between the heat source and the heat demand, to allow for smoother and more optimal operation of the heat source. This can avoid the suboptimal performance and increased mechanical wear that can occur when cycling on and off rapidly or operating below rated output to match demand directly (Millar et al 2019).

A TES, usually in the form of a hot water tank is present in 90% of the existing UK residential building stock (Fischer and Madani, 2017). The controlled use of this thermal storage may be able to make a low-cost contribution to efforts to tailor energy demands to the availability of surplus renewable energy, helping to reduce the stress on the grid and increase consumption of renewables. Figure 2 illustrates one such scenario. In an electricity grid system in which price is used as a control this can also reduce the cost of running the heating system as renewable energy.

The effectiveness of such a strategy in practice lies in being able to intelligently control the use of the TES to maximise consumption of stochastic RE. Typically systems are controlled to avoid times of peak demand, but this may not correlate well with periods of abundant RE generation. A better strategy could make a

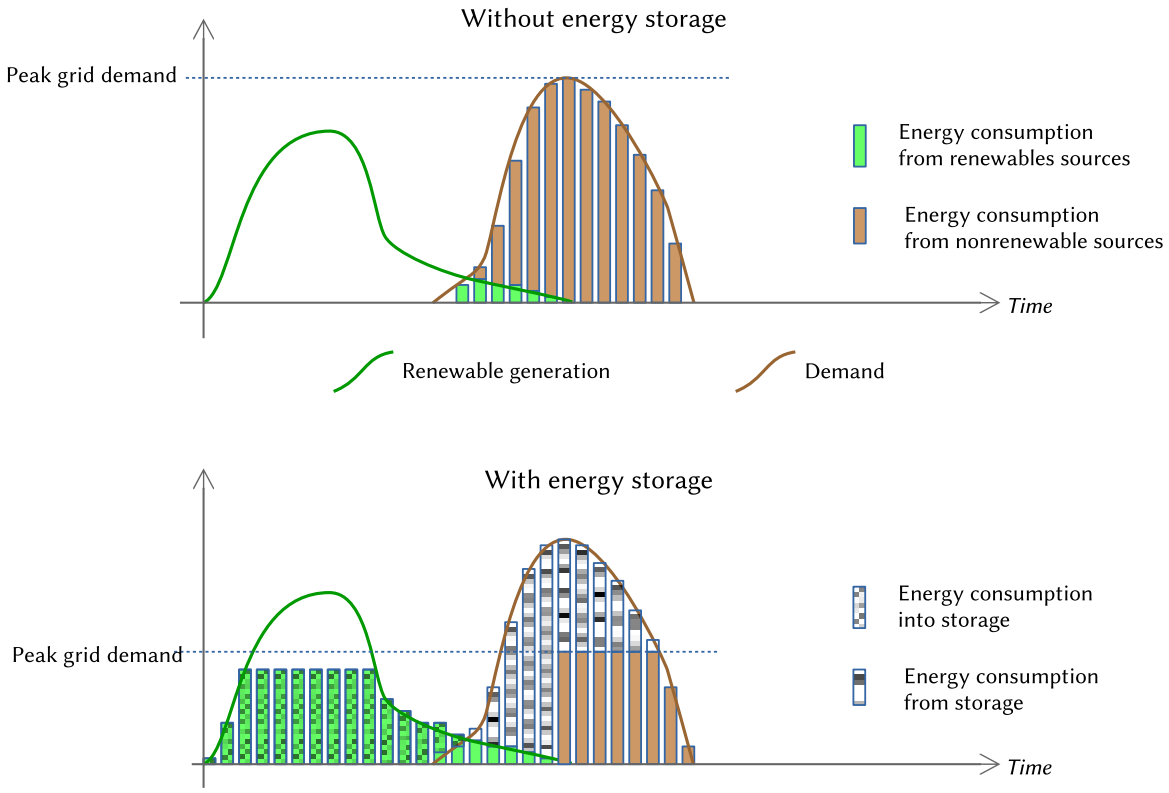


Figure 2: The intelligent use of storage to reduce grid demand and increase consumption of intermittent renewable sources.

significant contribution to tackling the decarbonisation of heat, especially if employed at a large scale.

The same problem exists in microcosm where a district heating systems (DHS) is coupled with local RE generation on a private wire network. This is the situation at the Findhorn eco-village on Scotland's Moray Coast, where the remote location results in high prices for grid electricity. The community collectively owns 675kW of wind generation capacity as well as its own private wire network, individual solar photovoltaic (PV) generation systems, solar thermal collectors, hot water tanks (HWTs) and heat pumps on residential blocks or individual houses.

The Findhorn community development company Duneland Ltd is currently going through the planning permission process for an extensive expansion project at the North Whins site (Figure 3). This development will comprise a mixture of co-housing, communal housing and independent housing, built in clusters across the site. All dwellings are to be built to similarly energy efficient standards as other

recent developments at Findhorn that have been the subject of previous study (Tuohy et al 2015).

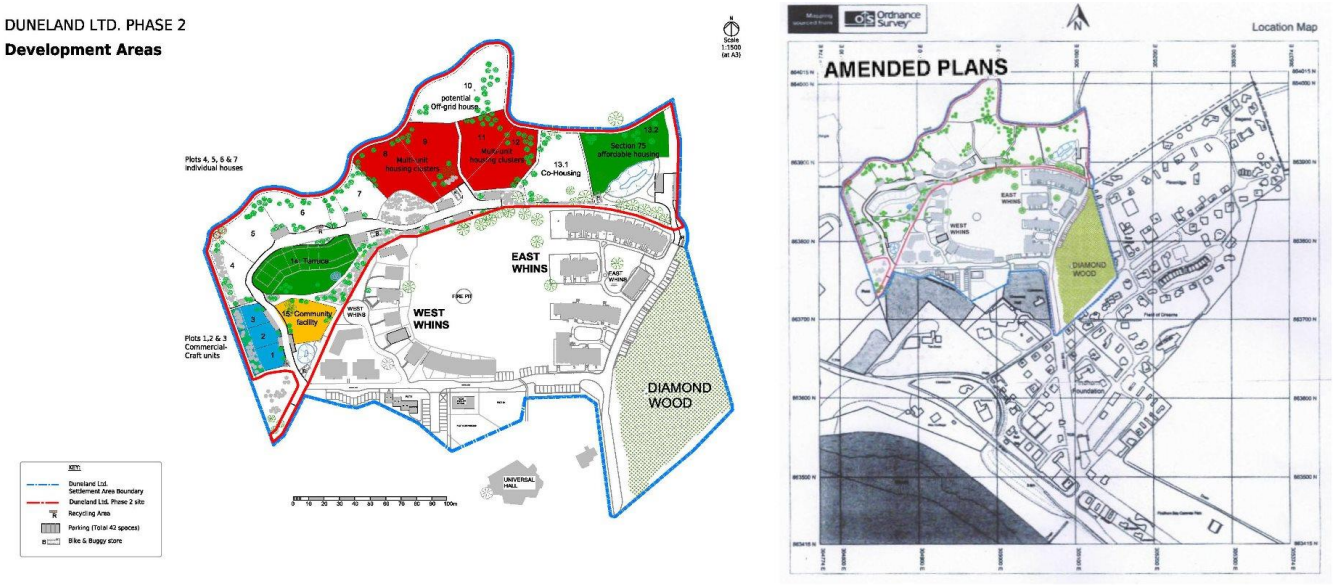


Figure 3 - Site plan for the North Whins development (outlined in red, left), and its location within Findhorn (right). From the Masterplan by Makar Architects and planning application submitted by Duneland Ltd.

This study will examine how a TES and small heat pump driven DHS supplying domestic hot water (DHW) and central heating (CH) to a one group of eight dwellings could be employed and controlled to maximise consumption of locally-generated renewables and reduce imports.

1.1.1. The case study site

The specific site to be considered in this study is a development of 8 affordable dwellings designed for the Park Ecovillage Trust (PET). It comprises a small terrace of four two-bedroomed houses and an adjacent block of four studio flats, spread over two levels. Figure 4 shows more detail on the site, including an artists impression and a site plan.

Figure 4: Site plan (left) and artists impression (right) of 8-dwelling PET development at North Whins. (Images courtesy of Findhorn Foundation.)

The local renewables to be considered comprise PV arrays mounted directly onto the buildings as well as the three turbines of the Findhorn Wind Park (FWP), the 'Three Graces'. These are 225kW Vestas V29 turbines ('Joy', 'Charm' and 'Beauty'). In the year to October 2017 the turbines generated 77% of the electricity consumed on the ecovillage site (Findhorn Wind Park, n.d.). There are also several PV arrays installed throughout the site on a "rent-a-roof" basis. As at September 2015 the total installed PV capacity was 23.85kW.

New Findhorn Directions (NFD) is the service company responsible for the village's infrastructure, including the provision of public utilities (including electricity) and management of the site's private wire network. NFD's sole supplier of electricity is FWP, who in turn trade only with a grid level supplier and NFD. Any generating plant that wishes to connect to the local network therefore has to gain permission from both organisations. FWP does not generally allow the connection of PV systems above 3.68kW.

1.2. Overall Aim

This thesis will attempt to develop a control strategy that could be employed to intelligently manage an 8-dwelling DHS at the North Whins site to maximise the

consumption of locally-generated RE. The selected strategy will then be implemented in Python code. Simulations based on data obtained from Findhorn will be run to establish the possible benefit from the use of this control system in terms of increased self-consumption of local renewables. Finally, the implementation of and improvements to this control system will be discussed, as well as the implications of more widespread of such systems.

1.3. Scope

As the North Whins development is still at the design phase, detailed thermal modelling based on building fabric will not be possible. Instead, appropriate existing profiles from other sites and literature has been used, though ways of customising them to increase their relevance to North Whins has been considered. In particular, the existing highly-insulated buildings at West Whins (shown in Figure 5), studied as part of the ORIGIN project, will be considered as surrogate models for the proposed buildings.

Figure 5: Semi-detached dwellings at West Whins (photo courtesy of Findhorn Foundation)

Plans for the site include the use of (cleansed) wastewater from the community's wastewater treatment plant, 'the Living Machine', as a heat source for a potential DHS serving much or all of the North Whins site. However, useful data on the heat resource represented by this is not available. Air sourced heat pumps (ASHP) are however widely used in buildings across the site, and the heat resource from these is readily determined from meteorological data. The model devised for this study will therefore include an ASHP as its heat source, with provisions for adapting this for a wastewater sourced heat pump (WWSHP) discussed.

1.4. Methodology

There are three major aspects to the work: the construction of a suitable control strategy, the implementation of this in code and the simulation of the North Whins site. The potential future implementation of this code in a control system must also be considered.

1.4.1. Literature review

There is extensive literature published on strategies for thermal control of heat pumps, DHSs and TES. This was reviewed for relevance to a system featuring all three of these components in order to select a control strategy to employ in the subsequent work.

Significant research was also required in devising the model for the DHS for North Whins – papers detailing suitable approaches to the modelling of a TES system, ASHP, residential demand for DHW and CH and renewable energy generation were reviewed.

1.4.2. Development of system model

The system model, bringing together the models of each of the components above, was created in Python code. Python is well suited to data-intensive applications, and as an open-source language it is highly extensible and can be implemented in a wide range of hardware environments. There is an extensive body of relevant pre-existing libraries which will be reviewed for incorporation in the model.

1.4.3. Development of control code

Once models are in place for all aspects of the system, the decision-making control system can be coded.

1.4.4. Simulation of control system

An evaluation of the effectiveness of the control system has three aspects. Firstly, how well the models reflect the behaviour of the actual system should be examined

as far as is possible given the as-yet-unbuilt site. Secondly, how well the control system responds to changing environmental factors (the variability in local renewable generation) should be examined, and finally the control system's response to unpredictable behaviour of building occupants.

1.4.5. Review, discussion and recommendations for future work

The effectiveness of the system will be reviewed, with recommendations made for identified weaknesses as well as future development to improve the system. Practical issues for the implementation of the control system in the development at North Whins will be discussed.

2. Literature Review

2.1. Modern DHS design

The DHS has a long history, but its nature has changed substantially over the past century. Lund et al (2014) identify four distinct 'generations' of DHS with distinct characteristics and means of operation, summarised in the well-known graphic by Thorsen et al (2018) shown in Figure 6.

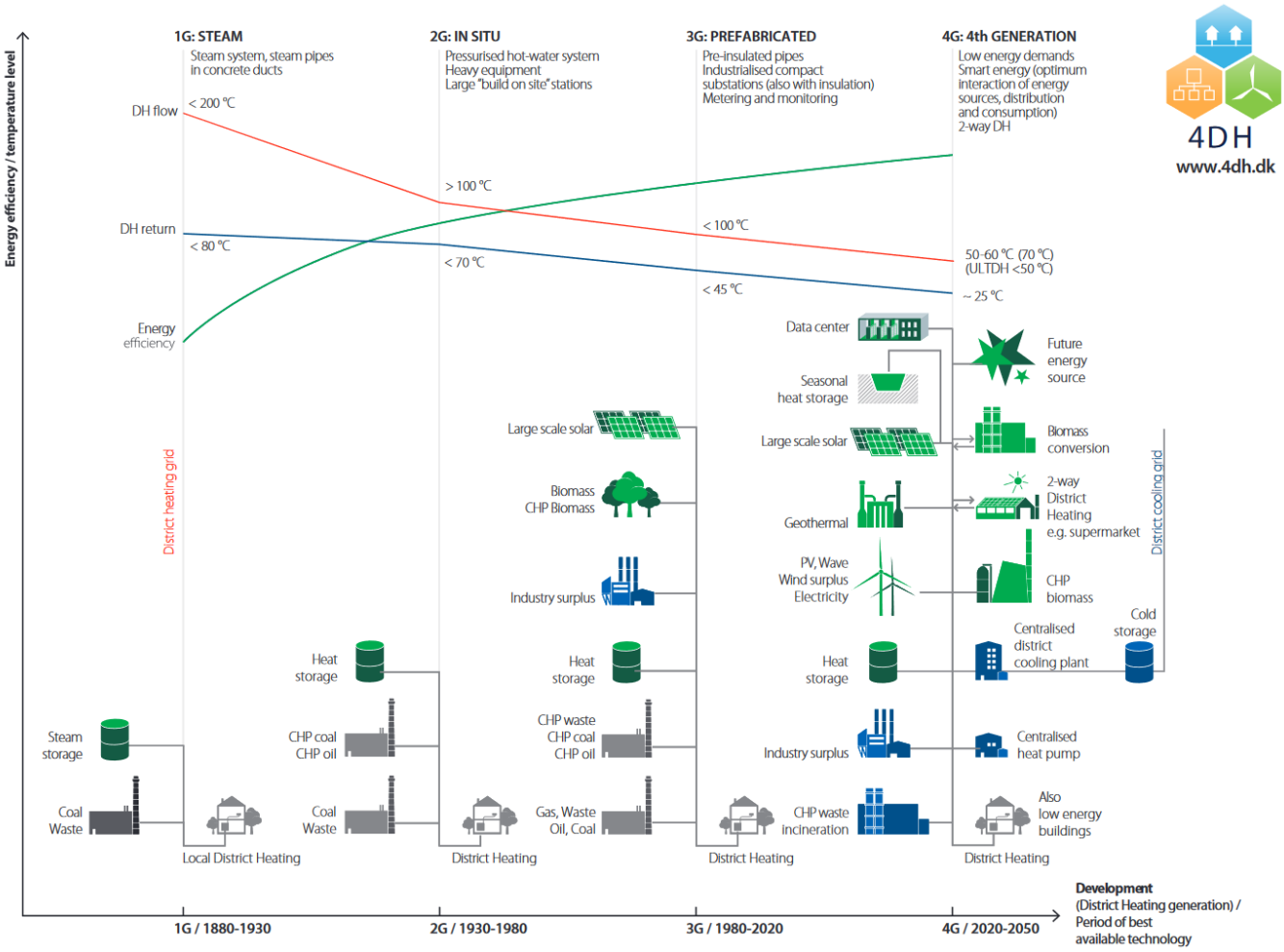


Figure 6: "Progression of District Heating – 1st to 4th generation," Thorsen et al (2018)

The proposed DHS at Findhorn fits the description of a 4th generation DHS (4GDHS), featuring a high degree of decentralised renewable energy generation, modern, well-insulated buildings and smart interaction of demand and RE supply. The temperatures in the network, and flow rates, are lower than the historic norms,

the heat typically supplied between 40-55°C. The risk posed by the growth of legionella bacteria in stored water, which thrives in water between 20-45°C, is minimised by using a closed system with indirect heating: rather than stored water being delivered into the domestic environment through hot water taps, a high-efficiency heat exchanger is present in each dwelling to raise incoming fresh water to the temperature of the DHS network. Without this separation, supply temperatures below 55°C would only be possible with the addition of a biocide (usually a chlorine-based oxidising agent) or periodic 'thermal shock' treatment in which the entire system, including all taps and appliances, is raised to between 70-80°C (European Guidelines Working Group, 2017).

The lower flow rates of 4GDH systems allow smaller pipes, and therefore more insulation for the same external duct size. Combined with the lower temperatures this means that heat losses can be significantly reduced. Typically the flow rates in the network are highly variable dependant on demand.

2.2. Thermal Energy storage

One of the crucial advantages provided by a DHS is highlighted by Lund et al (2016): the ability to implement energy storage in a more efficient form (thermal, gaseous or liquid) and in larger scale at the community level.

The overwhelming majority of TESs in operation use water as the thermal storage medium, most commonly in purpose-built highly insulated tanks (Sarbu and Sebarchievici, 2018). Water has a high thermal mass: its specific heat capacity of 4190J/kg K is greater than concrete (879 J/kg K) or oil (2200 J/kg K for the Calorie HT43 thermal oil), and an unbeatably low cost and high availability (especially in Scotland).

Many other thermal storage media have been considered and employed, most notably phase change materials (PCMs). These are a group of substances which can have potentially higher thermal energy densities by virtue of storing heat as latent energy as a substance melts and re-solidifies, instead of (or in addition to) as sensible energy. This energy stored in melting a substance, referred to as the

specific latent heat of fusion, can be much higher than sensible heat capacities: almost 200kJ/kg for stearic acid (Sharma et al, 2009). The use of these is technically more challenging as once a substance is in a solid form it becomes far more difficult to transfer heat into and out of. Any increased energy density must be weighed against the significant additional volume needed for structures (fins) to distribute heat into and out of the solid, and the greater cost of the substance – by an order of 100 times greater per kWh, as estimated by Hauer (in Sarbu and Sebarchievici, 2018).

An approach examined by Kelly et al (2014) was to use a hybrid TES of which a proportion of the volume was taken up with a PCM (an inorganic hydrated salt). For the particular case studied it was found that installing PCM modules to comprise 50% of the capacity of the TES could allow the total size of the TES to be reduced by half whilst still obtaining approximately the same performance. However, based on the relative price estimates of Hauer this 50% volume saving from the use of PCM modules would incur a cost increase of more than 800%.

In modelling the behaviour and performance of thermal storage, a range of different techniques can be applied. The simplest is to treat the store as a single mass of uniform temperature. The energy stored in can then readily be calculated from its temperature and specific heat capacity. For water used in heating systems, the specific heat capacity does not change significantly over its temperature range (the range is less than 1%).

This model omits the effect of the variance of a material's density with temperature, which is to give rise to thermal stratification. In any fluid, warmer masses at lower densities will be buoyant, floating on top of colder masses. In tanks with low flows this can be a significant effect. It can also be a beneficial effect as it means that higher temperatures can be maintained by drawing water off at the top of a tank, even as the energy stored in the tank drops, increasing the efficiency of energy storage by up to 20% over a fully-mixed tank (Ghaddar, 1994). However it can also have an injurious effect as it greatly increases the amount of energy need to raise the bottom of the tank to a high enough temperature to kill bacteria (Armstrong, 2015).

The degree of stratification is heavily influenced by the tank's shape. A taller, thinner tank will be better able to maintain a higher temperature difference between top and bottom, but the greater surface-area-to-volume ratio (SA:V) may increase thermal losses. The aspect ratio (the ratio of its height to its diameter) of HWTs is typically between 1.5 and 5. Armstrong (2015) modelled a range of aspect ratios for plausibly-insulated tanks (50mm of polyurethane) and found that the benefits from stratification outweighed the increased SA:V up to the maximum value studied (SA:V=5), but that the gains for larger tanks were less significant above an SA:V of 3.

As modelling the complex interactions of different masses of water is potentially extremely computationally expensive, a number of simplified models have been proposed.

One widely employed model is the 'plug-flow' model, which considers a tank as two bodies of variable mass, each at uniform temperature, with the thermocline forming a 'plug' of mixed water of fixed size between them. This thermocline boundary can then be considered to move vertically in the tank. This model is therefore also known as a moving boundary model and is used to model TES in software applications like energyPRO and TRNSYS.

Another approach is to consider the tank as being made up of separate nodes: volumetric spaces within which a fluid is fully mixed at a uniform temperature. Such models can be progressively simplified by considering these nodes in two dimensions only, by assuming radial symmetry within a cylindrical tank, or even by considering nodes in one dimension only, as a stack of vertical volumes. This latter approach makes the problem vastly easier to solve but neglects often significant effects such as conduction through tank walls (Lavan and Thompson, 1997), which can lead to destratification over time.

Such a one-dimensional model is described in detail by Duffie and Beckman (2013) and illustrated in Figure 7. Mass flows between the stratified nodes, and into the tank from outside are both modelled using simple logic-based functions dependant on the relative temperatures of the masses.

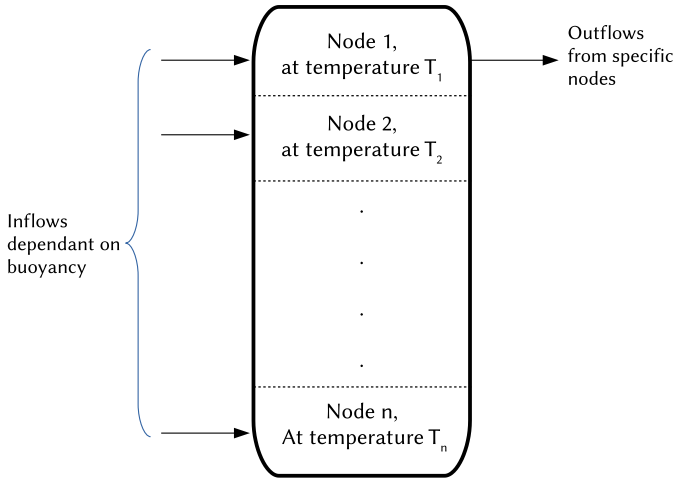


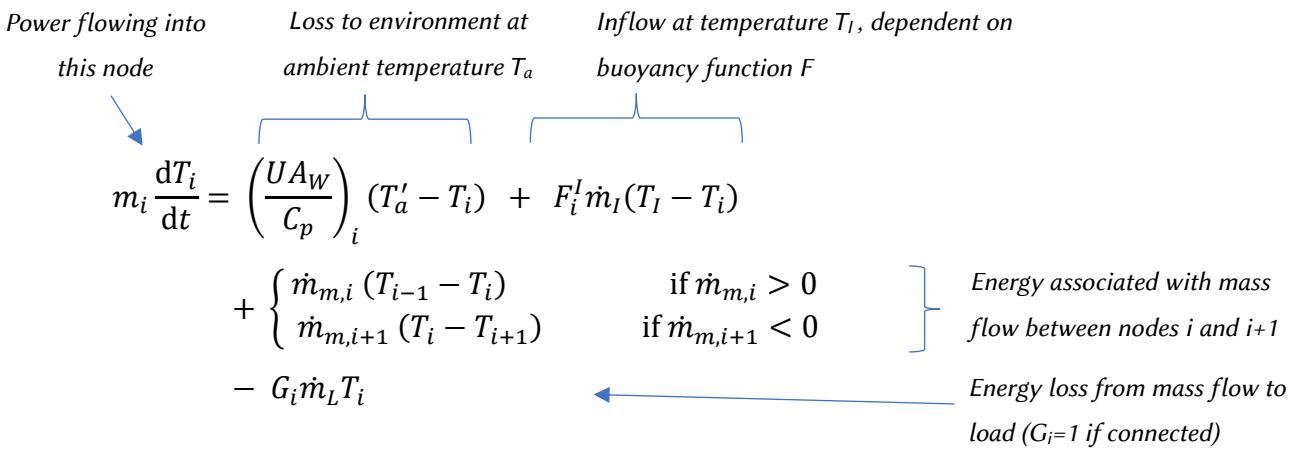
Figure 7: One dimensional multi-node model for TES

Ghaddar and Marafie (1989) formulated a similar approach to this problem which included an 'eddy conductivity factor' to consider turbulent intermixing and demonstrated that this significantly improved the validity of the model even at low flow rates.

One method of reducing the disruptive effects of turbulence and preserving stratification in a real tank is the use of a low-velocity inlet manifold which minimises mixing within the tank (Duffie and Beckman, 2013).

De Césaro Oliveski et al (2003) examined the relative accuracy of one- and two-dimensional nodal analyses of HWTs, confirming the greater validity against experimental observations of two-dimensional models. The study concluded that for long-term simulation of energy storage one dimensional analysis gave a good agreement with observed data, and recommended the inclusion of "computational artifices" such as counter-intuitively allowing node temperatures to swap to avoid temperature inversions occurring in the nodes.

In all of these approaches, determining the behaviour of a TES involves second-order differential problems: the rate of change of temperature of a node reflects the energy transfer which is in turn a function of the relative temperatures between nodes. This is shown in Equation 1, the node energy balance equation adapted from Duffie and Beckman (2013).



Equation 1 – Energy balance for node i within a multi-node tank (after Duffie and Beckman, 2013). The annotations of each term are indicative only – all terms have been divided by C_p (the specific heat capacity of water).

Note that conduction through the tank walls is not being considered here.

Buckley (2012) presents a computational method of solving this using a finite difference method to discretise this differential equation. This is then solved by an implicit method in which the temperatures of each node are modelled as a function of node temperatures in the previous timestep, creating a set of simultaneous equations of equal number to the number of nodes in the tank, which can be solved as a linear matrix equation.

Figure 8 and Equation 2 illustrate this method for an arbitrary node n in a tank. Upward mass flow through the node (m_{in}, m_{out}) is taken to be positive. Δx is the distance between the centroids of each node, A_W is the exposed area of the node wall with heat loss to the environment of U, and k is the conductivity of water through the cross-sectional tank area A.

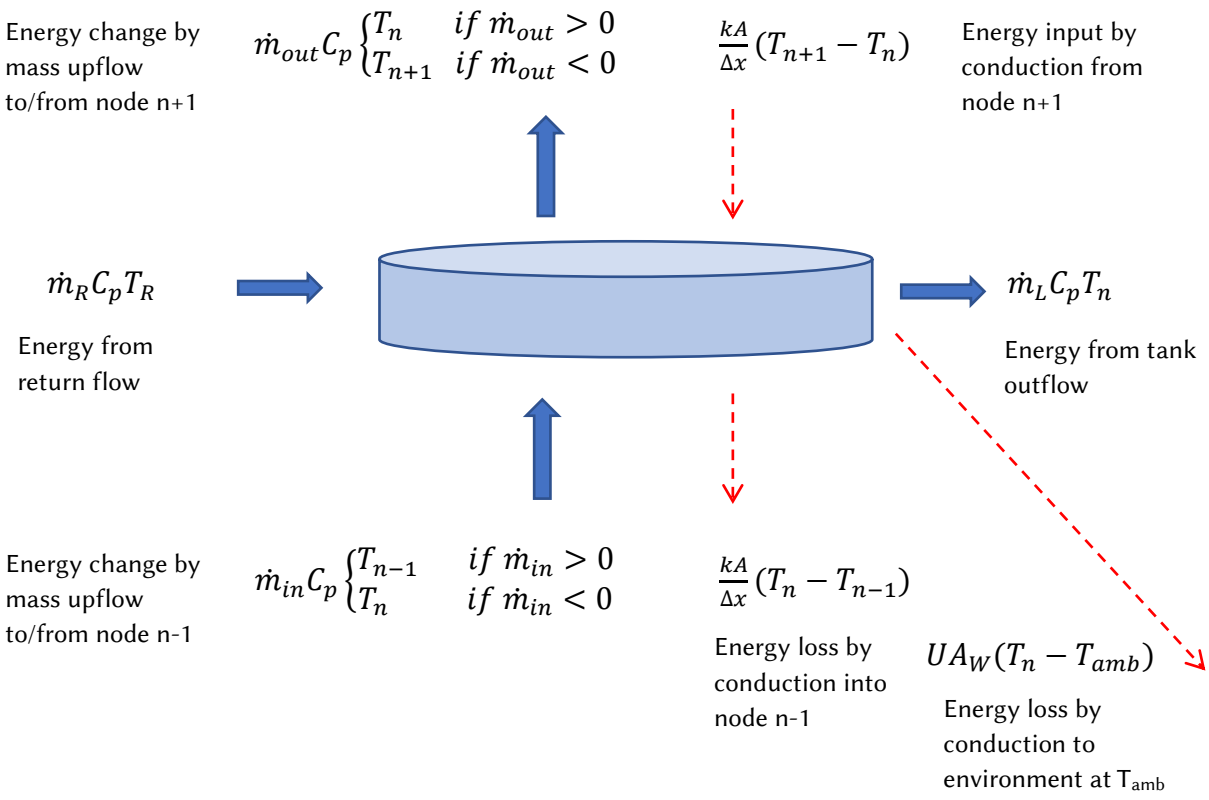


Figure 8: Energy balance on an arbitrary tank node n. m_{out} = 0 at top node, m_{in}=0 at bottom node, m_R and m_L are only nonzero for nodes with external connections.

$$m_n C_p \frac{T_n^{t-1}-T_n}{\Delta t} = \dot{m}_{out}C_p\begin{cases}T_n\\ T_{n+1}\end{cases} + UA_W(T_n-T_{amb}) + \dot{m}_L C_p T_n + \frac{kA}{\Delta x}(T_n-T_{n-1}) - \left[\dot{m}_R C_p T_R + \dot{m}_{in}C_p\begin{cases}T_{n-1}\\ T_n\end{cases} + \frac{kA}{\Delta x}(T_{n+1}-T_n)\right]$$

Equation 2 – Energy balance equation for node n shown in Figure 8 (expressed as the loss in energy from this node in this timestep)

By rearranging the terms we can arrive at Equation 3 which collects determinable coefficients of each node temperature:

$$T_n\left[\frac{m_n C_p}{\Delta t} + \begin{cases} \dot{m}_{out} C_p & \text{if } \dot{m}_{out} > 0 \\ 0 & \text{otherwise} \end{cases} + UA_W + \dot{m}_L C_p + \frac{2kA}{\Delta x} - \begin{cases} \dot{m}_{in} C_p & \text{if } \dot{m}_{in} < 0 \\ 0 & \text{otherwise} \end{cases}\right]$$
$$+T_{n-1}\left[\frac{-kA}{\Delta x} - \begin{cases} \dot{m}_{in} C_p & \text{if } \dot{m}_{in} > 0 \\ 0 & \text{otherwise} \end{cases}\right]$$
$$+T_{n+1}\left[\frac{-kA}{\Delta x} + \begin{cases} \dot{m}_{out} C_p & \text{if } \dot{m}_{out} < 0 \\ 0 & \text{otherwise} \end{cases}\right]$$
$$= \frac{m_n C_p T_n^{t-1}}{\Delta t} + UA_W T_{amb} + \dot{m}_R C_p T_R$$

Equation 3 – Rearranged energy balance equation

However, these need to vary slightly for the top and bottom node, to replace the conduction from non-existent nodes with increased losses to the environment:

$$T_{top}\left[\frac{m_n C_p}{\Delta t} + U(A_W + A) + \dot{m}_L C_p + \frac{kA}{\Delta x} - \begin{cases} \dot{m}_{in} C_p & \text{if } \dot{m}_{in} < 0 \\ 0 & \text{otherwise} \end{cases}\right]$$
$$+T_{top-1}\left[\frac{-kA}{\Delta x} - \begin{cases} \dot{m}_{in} C_p & \text{if } \dot{m}_{in} > 0 \\ 0 & \text{otherwise} \end{cases}\right]$$
$$= \frac{m_n C_p T_n^{t-1}}{\Delta t} + T_{amb} U(A_W + A) + \dot{m}_R C_p T_R$$

$$T_{bottom}\left[\frac{m_n C_p}{\Delta t} + U(A_W + A) + \dot{m}_L C_p + \frac{kA}{\Delta x} + \begin{cases} \dot{m}_{out} C_p & \text{if } \dot{m}_{out} > 0 \\ 0 & \text{otherwise} \end{cases}\right]$$
$$+T_{bottom+1}\left[\frac{-kA}{\Delta x} + \begin{cases} \dot{m}_{out} C_p & \text{if } \dot{m}_{out} < 0 \\ 0 & \text{otherwise} \end{cases}\right]$$
$$= \frac{m_n C_p T_n^{t-1}}{\Delta t} + T_{amb} U(A_W + A) + \dot{m}_R C_p T_R$$

Equations 4 & 5 – Rearranged energy balance equations – special cases for top and bottom nodes.

The resulting set of simultaneous equations can be expressed in a matrix form AT = C, where T is a vertical matrix of node temperatures at the next timestep, and solved by standard computational processing libraries.

2.3. Air-sourced heat pumps

A heat pump is a technology that applies mechanical work to force heat to move counter to its usual equalising flow, making a hot space hotter or a cold space colder. In the case of ASHPs used for space heating, low-grade heat is extracted from ambient outdoor air and 'pumped' into a space to generate useful heat.

The outline concept of all heat pumps is shown below in Figure 9.

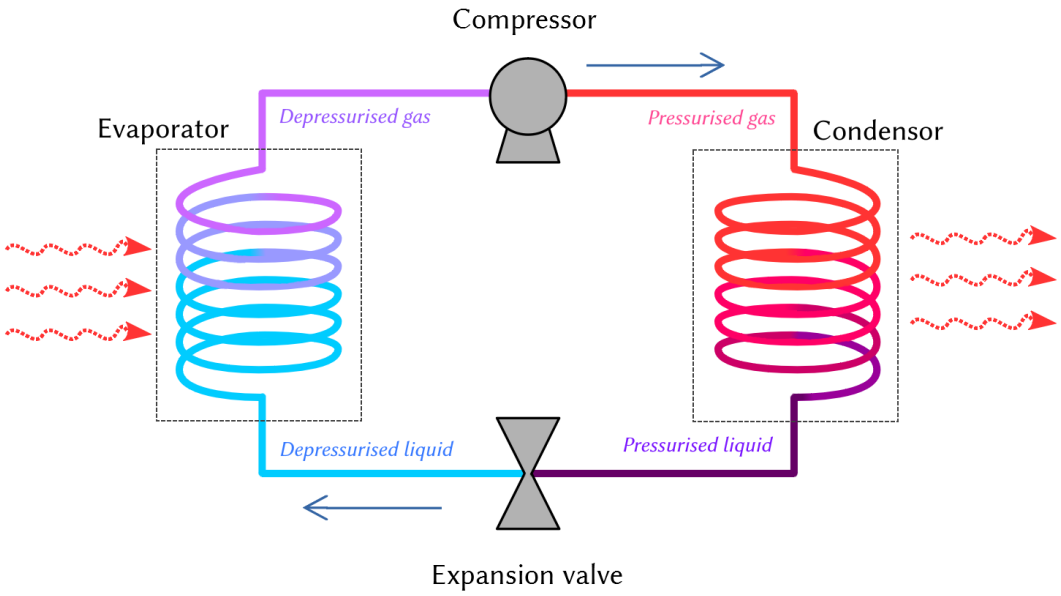


Figure 9: Conceptual diagram of a heat pump

A refrigerant is compressed, raising its temperature, and from this heat can be rejected to one environment as it cools and condenses. This liquid refrigerant is then pumped to another environment where it is allowed to return to a lower pressure. At this lower pressure it will more readily evaporate – and to do this it absorbs heat from this second environment. By this process, heat is transferred from the second environment to the first, converting low-grade heat source at ambient temperature into useful heat. That the amount of heat thus transferred greatly exceeds the amount of mechanical work applied has made heat pumps a technology of great interest in the ongoing effort to decarbonise the global demand for heating and cooling systems. This level of interest is resulting in significant improvements in performance and reliability (Chua et al, 2010).

Heat pumps can be designed specifically to pump heat in one direction or can be reversible. The performance of a heat pump is commonly measured by the ratio of heat delivered to electrical power required, the Coefficient of Performance (COP), which varies considerably in different operating circumstances. The overall average COP for a particular heat pump in a particular location in all seasons is referred to as the Seasonal Performance Factor (SPF).

A number of different methods have been employed to model the variability in COP. A basic model, as used for instance in the energyPRO modelling software, applies a constant efficiency against the Lorentz model of a heat pump:

$$COP = \eta\, COP_{Lorentz} = \eta\, \frac{T_m}{T_m - T_o}$$

Equation 6: COP modelled as Lorentz model with constant system efficiency η. In our case, T_m = logarithmic mean temperature (LMT) of delivered hot water, T_o = LMT of heat source. (EMD International, 2019)

Staffell et al (2012) derive simple formulae for COP as a function of the "lift" - the difference between the source temperature and delivered temperature. Their formula for a generic ASHP, derived from industry data and field trials, is:

$$COP_{ASHP} = 6.81 - 0.121\Delta T + 0.000630\Delta T^2 for 15 \leqslant \Delta T \leqslant 60$$

Equation 7: Generic ASHP COP model after Staffell et al (2012)

A similar basis was used by Kelly and Cockroft (2010) based on a model devised by Ferguson et al (2009) for a micro-cogeneration system. This too models COP as a function of thermal lift, but also adds secondary components to model effects such as the thermal mass of the system and parasitic heat losses.

Murphy et al (2013) provide a more complex, 'black box' method of modelling COP and thermal output based on a least-squares regression from data points obtained as part of the station European Heat Pump Association test regime. The regression

was carried out based against intake temperature (ambient air) and water return temperature as two independent variables.

Underwood et al (2017) provided a development of this approach using a 'grey box' method, again to obtain COP and thermal output. They performed similar regressions on experimental data points to obtain figures for ten parameters relating to the physical characteristics of an ASHP system: six that model the behaviour of the compressor, two that characterise the heat exchangers in compressor and evaporator, and so on.

With ASHPs one must also account for the need to defrost the condenser periodically when the surrounding environmental is at low temperatures), which adds a significant degree of variability between heat pump models and is therefore challenging to model (Underwood et al, 2017).

The heat pump employed in the recent West Whins development at Findhorn, the template being used for the current study, is an air-to-water system, the Mitsubishi Ecodan PUHZ-HW140V monobloc system. The manufacturer-supplied data for this model is provided as a range of values (from maximum/nominal performance to minimum performance) based on function of source (air) temperature, and supplied (hot water) temperature. These data show that for the same 'lift' the COP can actually vary as much as 31% (Mitsubishi 2015), and that defrost cycles are anticipated when ambient temperature reaches 2°C.

The WWSHP proposed for the North Whins development is conceptually similar to ASHP models, and in many ways simpler in practicality. The COP is likely to be far more stable due to the far smaller variations in temperature of the source water and the absence of the need for defrost cycles. One complexity that need be taken account of however is the efficiency of the in-channel heat exchanger, the design of which in turn is heavily dependent on the precise configuration of the wastewater channel and the likelihood and type of fouling that may occur (Culha et al 2015). The situation at Findhorn is somewhat distinct from the majority of cases studied, as the outflow channel is effectively cleansed water output from the natural

purification process of the 'Living Machine'. Fouling is therefore less of a problem than would be the case with a typical WWSHP.

2.4. Control systems

A wide range of different techniques have been applied to the task of optimising patterns of heat consumption with the use of thermal storage. Yu et al (2015) provide a review of the various techniques, summarised in Figure 10.

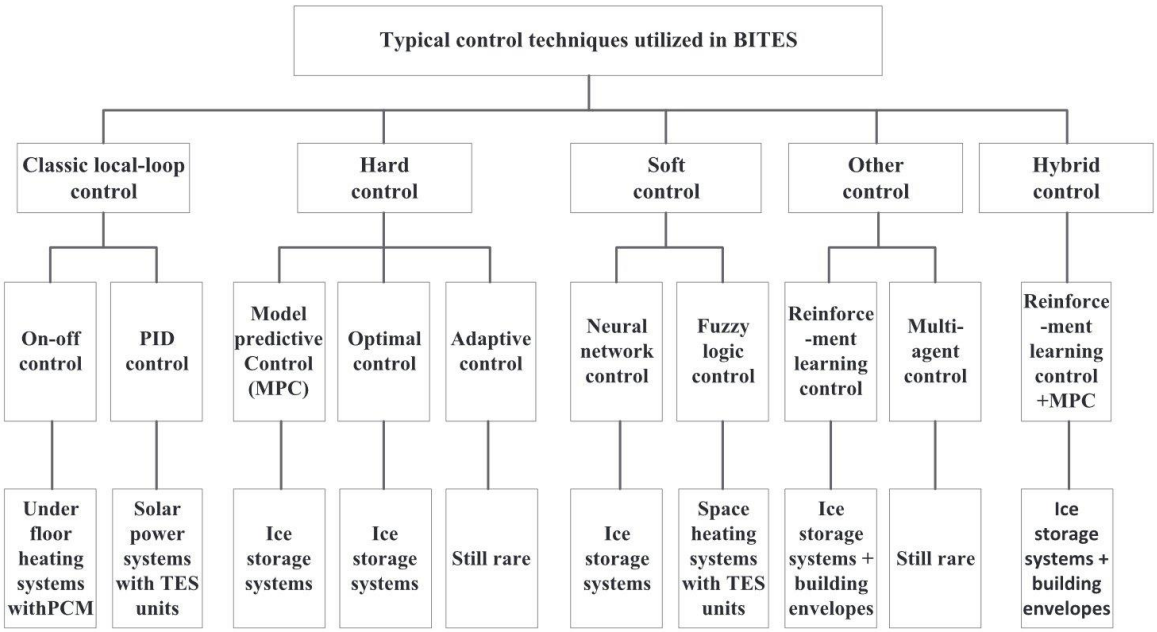


Figure 10: Control techniques used in Buildings Integrated with Thermal Storage (BITES) with some typical applications. From Yu et al (2015) after Afram and Janabi-Sharifi (2014)

Classic local-loop control, familiar to all users of central heating systems, may be based around a set time-pattern or respond to a temperature set point, usually with some hysteresis to prevent rapid cycling of heating plant. PID (proportional-integral-derivative) control is long established due to the relative ease of implementation with basic analogue electronic components – and indeed with mechanical components before this. However, neither of these systems allows a system to be able to plan ahead against external signals as we require to achieve load balancing.

Such planning can be achieved in a number of ways. Adaptive control systems and neutral networks both examine the performance of a system and optimise it against a desired outcome as a black box (or 'grey box' in the case of model-referenced adaptive control).

Model Predictive Control (MPC) is refinement of the model-referenced adaptive control technique. It has three basic elements:

1. A pre-defined system model which can be used to predict future states of that system (up to a 'prediction horizon'),
2. an objective function (usually referred to as a 'cost' function) which defines the optimality of operation, and
3. control logic to minimise (or maximise) the value of the cost function.

A typical application of MPC involves running the control logic at each decision timestep, each time optimising the value of the objective function over the prediction horizon and devising a plan of operation to minimise the cost function. The interval between runs of the control logic (the 'decision time step') is significantly less than the prediction horizon, allowing the controller to respond to unexpected responses or changes to the system state. Figure 11 illustrates this.

Variants of MPC can be applied in a wide range of circumstances, including systems with multiple inputs and outputs, systems with non-linear characteristics, hard constraints, complex interacting control loops and poor data quality.

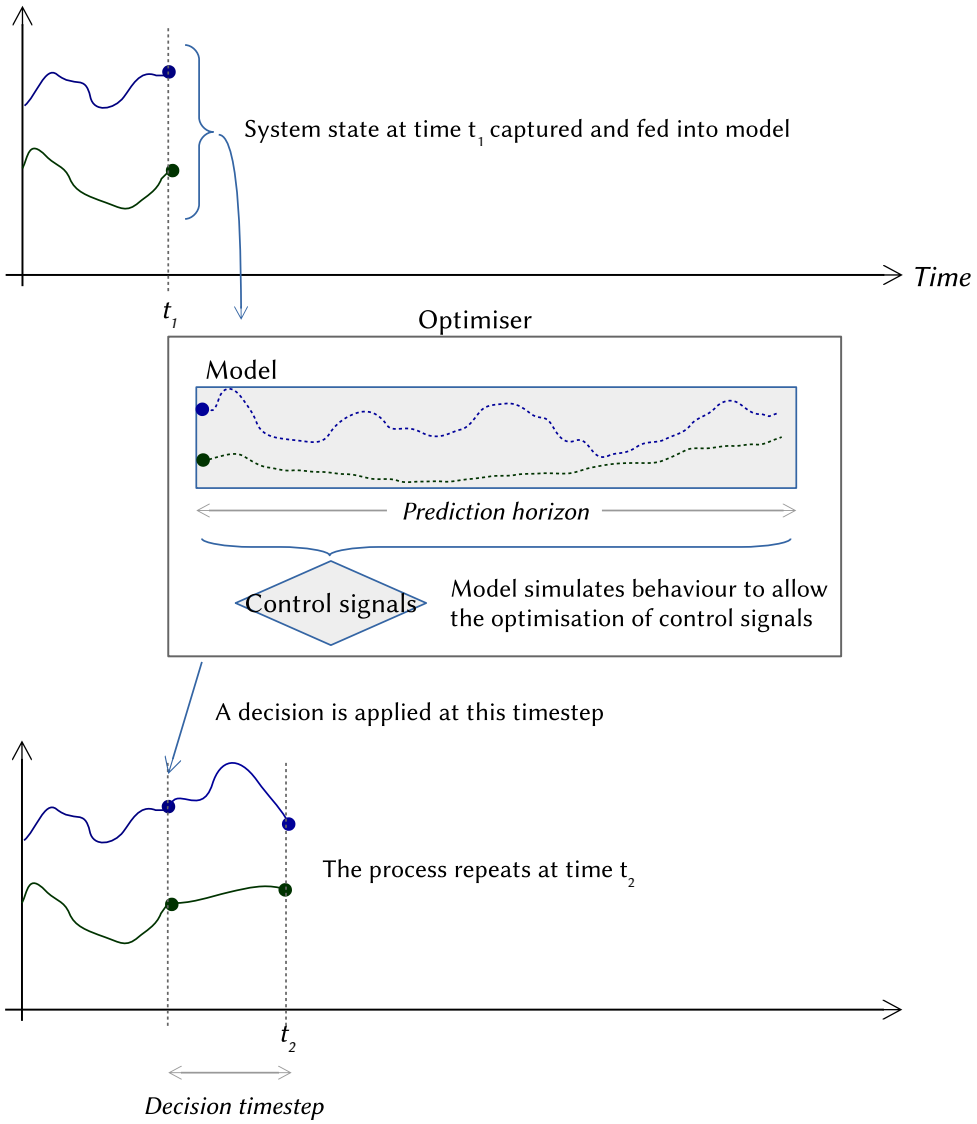


Figure 11: Conceptual outline of an MPC process

In the case of MPC used for DSR, the objective function would typically be based on the cost of energy, carbon emissions, or renewable self-consumption, and the prediction model would include forecasts of energy cost. Thieblemont et al (2017) reviewed MPC strategies applied to TES which incorporate weather forecasting, revealing also the growing interest in MPC strategies as small but computationally powerful devices become cheaper and more commonplace.

2.5. Findhorn and the ORIGIN project

A key reference point for the present study is the past work conducted in collaboration with the Findhorn Foundation Community, most notably the ORIGIN project (Orchestration of Renewable Integration Generation In Neighbourhoods) which ran from 2012 to 2015 (Tuohy et al, 2015 and Owens, 2015). Findhorn was one of three communities participating in the project, alongside Tamera in Portugal and Damanhur in Italy – all communities with significant local renewable generation. As well as automated control strategies, the project developed advanced mathematical forecasting techniques using neutral networks and investigated participatory DSR through a web-based graphical interface designed in collaboration with the communities. 12-months of demand and generation monitoring was carried out in each location, including periods of detailed monitoring of usage, performance and behaviour in individual properties.

The project demonstrated that whilst offering consumers a discounted 'green energy tariff' to incentivise load shifting only resulted in a small reduction in consumption of local renewables, implementing DSR via remotely-controlling appropriate devices could significantly decrease imports and increase consumption of local renewables (by 12% and 27% respectively).

TES formed one subset of the loads deemed appropriate to be controlled. The 'Centini' buildings at Findhorn contain a 210l storage tank (for DHW only), heated both by an electrical immersion heater and a solar thermal collector. Under normal operation, the electrical immersion heater would be active for one 'off-peak' hour once daily. An MPC algorithm was developed to shift this electrical heating demand according to the predicted solar input and local wind turbine generation whilst maintaining the availability of hot water. The flexibility of this system was reduced by the need in direct DHW storage to sterilise the tank periodically to prevent the buildup of legionella bacteria. However it was found that by shifting loads, or omitting a daily heat injection following a sterilisation cycle, significant flexibility is possible.

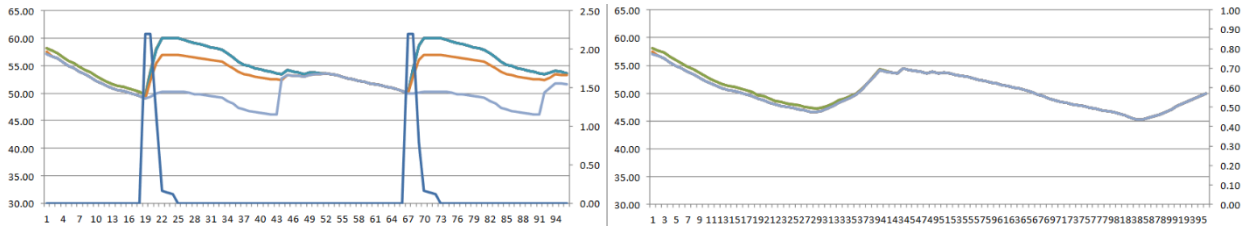


Figure 12: Modelled TES behaviour from the ORIGIN project. t=0 is 15:30 on a June day

Figure 12 shows how tank node temperatures vary with a daily heat injection (left), which can be omitted entirely, allowing thermal comfort to be achieved for a 48 hour 'coasting' period through solar heating only (right). The thermal comfort conditions can be allowed to vary throughout the year but is typically taken as being 38C – human body temperature (and therefore the 'standard' temperature for a shower).

3. Development of System Model

3.1. Demand Modelling

The prediction of time-varying demand patterns has been approached from two directions. Firstly, standard daily patterns based on residential building archetypes have been obtained from the Biomass Decision Support Tool (BDST) (Carbon Trust, 2016). The BDST is an Excel-based heating system modelling tool developed by the University of Strathclyde's Energy Systems Research Unit in collaboration with the Campbell Palmer Partnership. These provide example hourly thermal energy demands for CH and DHW combined for each housing type (defined by building type and age) for a range of daily average temperatures between -3 and 14°C. Two typical demand profiles for a semi-detached property built since 2007 are shown in Figure 13.

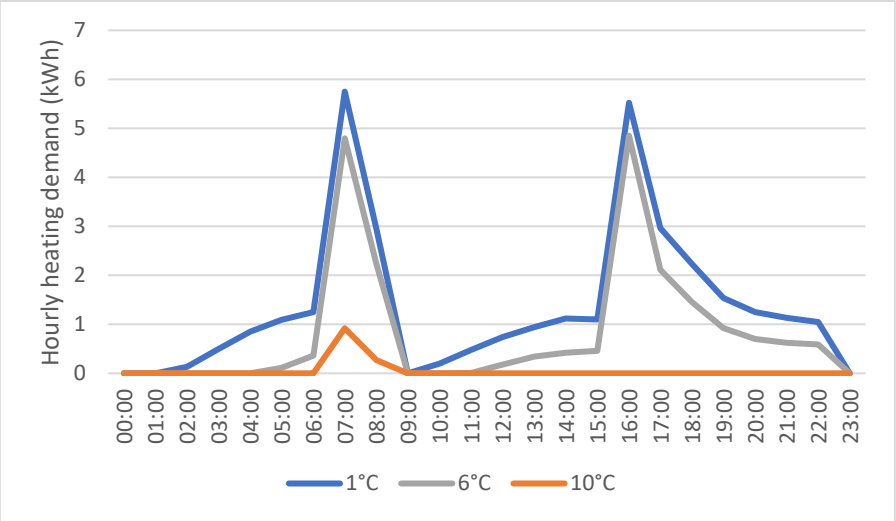


Figure 13: Typical demand profiles for three daily average temperatures, from the Biomass Decision Support Tool

The second dimension to the demand modelling is the use of data from the ORIGIN project. A data set has been obtained giving half-hourly CH+DHW demand patterns in a West Whins semi-detached comparator building for two five-week periods in February and June 2015. Average daily temperatures recorded during these periods varied between 0.7 and 22.4°C.

The dataset does not provide enough points to build up a site-specific profile based around average daily temperatures. It also contains numerous outlier readings which may be the result of malfunction of the sensing apparatus or extremely unusual behaviour. Figure 13 shows the averaged daily profile for the September period and the February period. The error bars show the range of +/- 1 standard deviation of the hourly data, which is many times larger than the average, and in some cases an order of magnitude larger than the average.

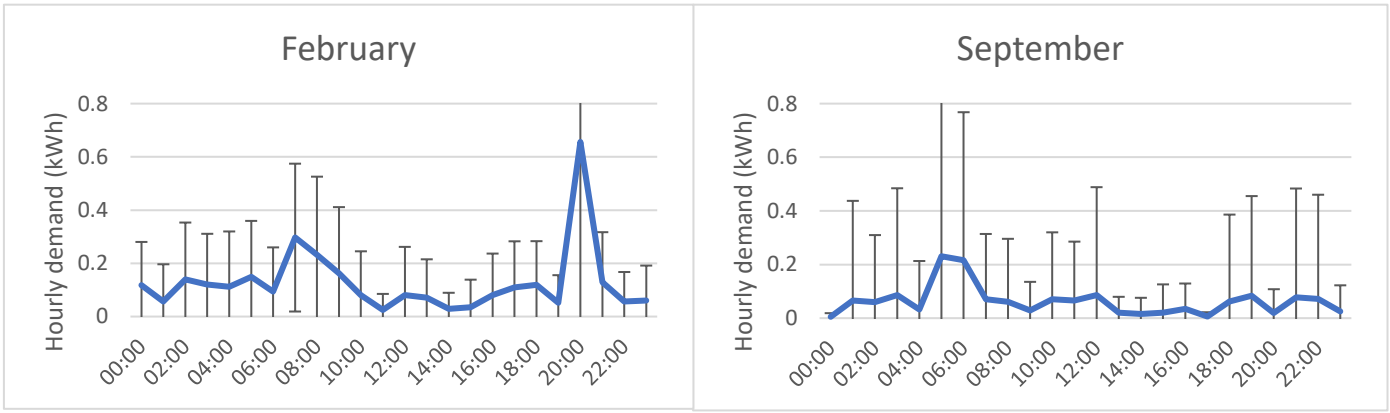


Figure 14: Average daily profiles for West Whins dwelling, from ORIGIN data, with error bars showing +/- 1 standard deviation

It would clearly not be valid to attempt to create profile data in the sort of format provided by the BDST from this data. For this study, the BDST tables were used but linearly scaled to represent the difference in performance between the archetype and the West Whins template.

The total energy used over the periods studied was 155.8kWh. Applying the same daily temperature data to the BDST archetype the total energy consumption would be expected to be 861kWh[1]. The demand profiles extracted from the BDST will therefore be scaled by a factor of 0.18 to crudely represent the average performance of the West Whins prototype.

[1] The 14°C pattern was assumed to be DWH only, and therefore was applied to all average daily temperatures above 14°C

3.2. Heat network

The demand model must also take account of losses associated with the heat network. In the absence of a detailed specification, planning guidance figures have been used from the Standard Assessment Procedure (SAP) (BRE, 2016). These provide a suggested scale to be applied to domestic demand to account for losses in a DHS. A significant change to the figures have been proposed since the last adopted guidance in 2012 (and agreed to by the Department of Business, Energy, and Industrial Strategy [BEIS] in their response): the demand scaling to a small heat network have been increased from a factor of 1.05 to 1.5. The justification for this is that it reflects the greatly reduced energy consumption of modern homes: the losses in the DHS system become far more significant as a proportion of overall demand. The revised figure of 1.5 will be applied to this model. This figure will be considered to encompass all losses from leaving the tank to the point of use in each dwelling, including the imperfect transfer of heat from the DHS loop to the incoming water at each property.

A scale of 1.5 suggests that the losses in the system are around 33%. This compares with a figure of 12% cited as typical for larger DHS systems in Sweden (Vesterlund et al 2013).

The network will be modelled as a perfectly variable flow system, with pumping energy considered to add 1% to the electrical demand as per the SAP 2016 guidance. The network supply and return temperatures will be set as 50°C and 20°C, within the archetypical ranges for 4GDH identified by Lund et al (2014) and meeting the requirements of CIBSE guidance for heat network design (CIBSE, 2015).

With a small population size of 8 units there may not be much opportunity to benefit from the smoothing effects of demand diversity. The ORIGIN project also faced this problem in modelling demand from individual households – in this case based on learned patterns rather than standardised templates. The approach adopted was to build in a prediction margin based on one standard deviation of the variability across the whole day. This has the effect of making the model more pessimistic, which is an effect that should to some extent be averaged out as the model is re-run hourly. This approach was adopted for the current study.

3.3. Thermal Storage

The TES model used in the simulation was based heavily on that developed by Buckley (2012) for simulation in MATLAB, but simplified in a number of ways. The original model allowed for the mass of water in the tank to vary, with nodal volumes allowed to become empty space as the water level dropped. The sealed DHS using indirect heating considered in this system will be considered to contain a fixed mass of circulating water, and will neglect the changes in density also modelled by Buckley. In a real installation, an expansion vessel would be provided to maintain the unvented tank at capacity and pressure safely.

PCM storage will not be considered; due to the cost it is unlikely that such technology would be implemented at Findhorn.

Four connections are made to the TES: the draw to the heating circuit load is taken from the top node of the tank, and the outlet to the heat pump is taken from lowest node. The return flows from the heat pump and the heating circuit are both considered to be injected by a perfect inlet manifold, operating such that water re-entering the tank is re-injected above the highest node whose temperature is lower than that of the inflowing fluid, with no turbulence effects. Inflowing fluid to a node is not allowed to exceed the mass of the node in a single timestep, so if the first node cannot accept all the incoming water the remainder is injected into higher nodes sequentially.

3.3.1 Tank Sizing

Clearly, the larger the tank, the greater the possible impact for DSR. The study by Kelly et al (2014) modelled a 1000L tank for single detached dwelling, and also cited a finding by Arteconi et al (2013) which found that up to 800L capacity is required to reliably achieve even a single hour of load shifting in detached dwellings insulated to 1990 UK building standards with both underfloor and radiator-based heating systems. The load shifting in this study used manually set ‘off peak’

timings; with an MPC controller responding to situations even greater flexibility may be required.

A 1000l tank at the network temperatures of 50°C /20°C theoretically provides 34.3kWh of heat storage. The daily demand obtained for the lowest monthly average temperature (2.4°C in February[2]) from the BDST when scaled by the West Whins performance factor (0.18) and the DSH loss factor (1.5) is 53.0kWh, representing a tank volume of approximately 1,550l. This will be used in the simulation.

3.3.2 Tank losses

An aspect ratio of 3 was chosen to improve stratification in the tank whilst maintaining plausible dimensions. The resultant dimensions of the modelled tank are shown in Figure 15.

An ideal tank might be modelled as 5mm of stainless steel (k=20W/mK) insulated with 35cm of mineral wool (k=0.04W/mK – though the same conductivity would be achieved using cellulose or hemp wool, which might be more in keeping with the ethos of the Findhorn community), given an overall U value of 0.11W/m²K for the tank walls.

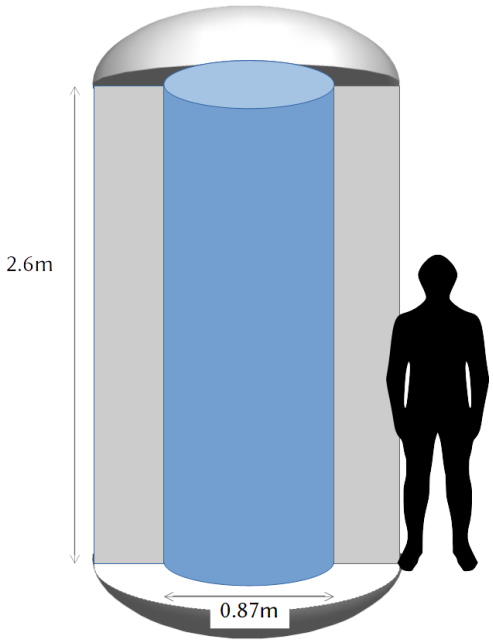


Figure 15: Dimensions of the modelled tank

3.4. Heat pump

The ASHP was modelled by regression following a method similar to that of Murphy et al (2013), using the nominal data provided by Mitsubishi. The two-dimensional regression of COP to ambient temperature and outlet temperature is shown in Figure 16 as a 3D surface plot, and compared to the raw Mitubishi data and the generic ASHP characteristics obtained by Staffell et al (2012).

[2] Climate data from climate-data.org

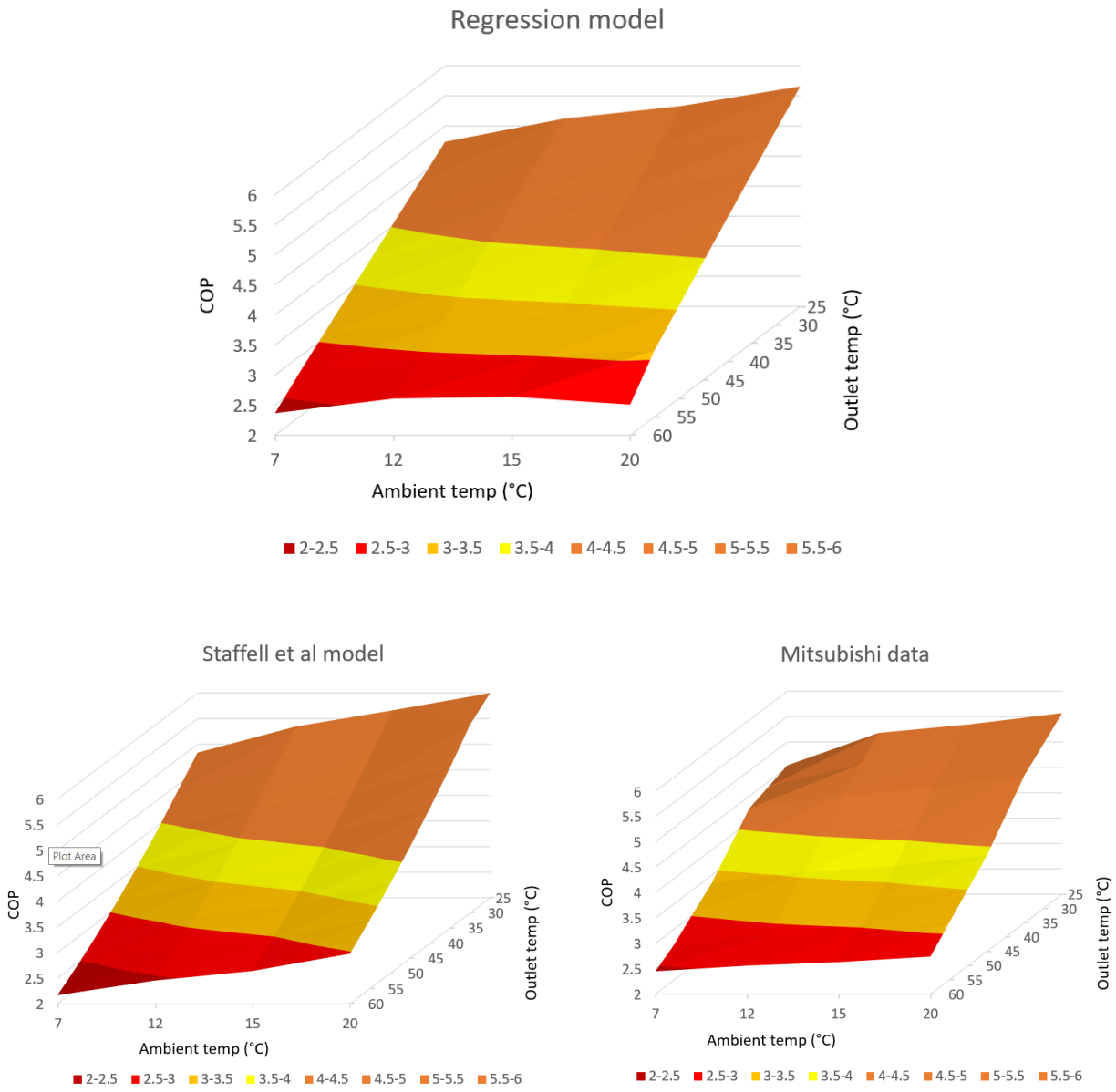
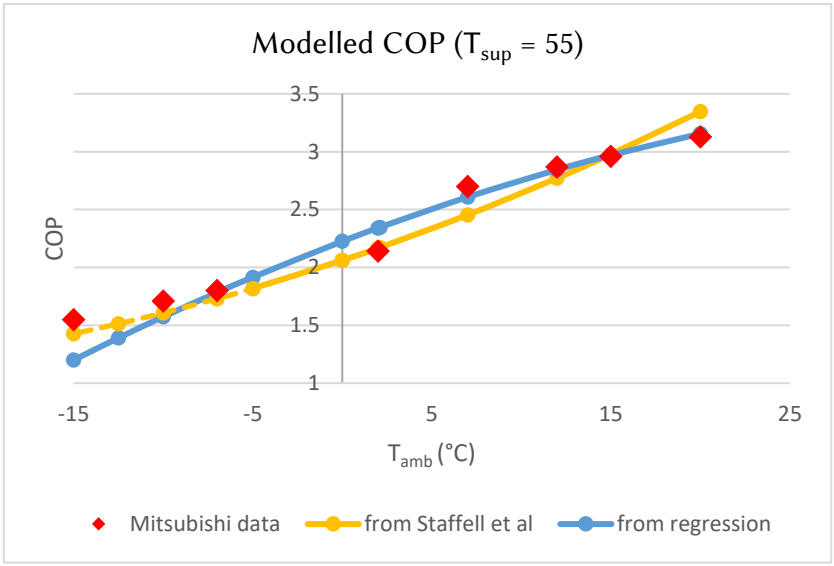


Figure 16: 2-dimensional characteristics of the ASHP, obtained by regression (top), compared to a standardised model after Staffell et al (2012) (lower left) and from raw manufacturer data (lower right)

The behaviour of the ASHP changes substantially when defrost cycles are required (below 2°C, not shown in the above figures), and the accuracy of the regression is at lower temperatures is significantly affected by this. To compensate for this, a separate regression was performed on all points supplied where defrost cycles would occur. The ASHP model will set its characteristics each time it is run, based on whether the external air temperature is below 2°C. Figure 17 shows how the

matching is improved with the introduction of this 'breakpoint' and Equation 8 shows the resultant COP formula.



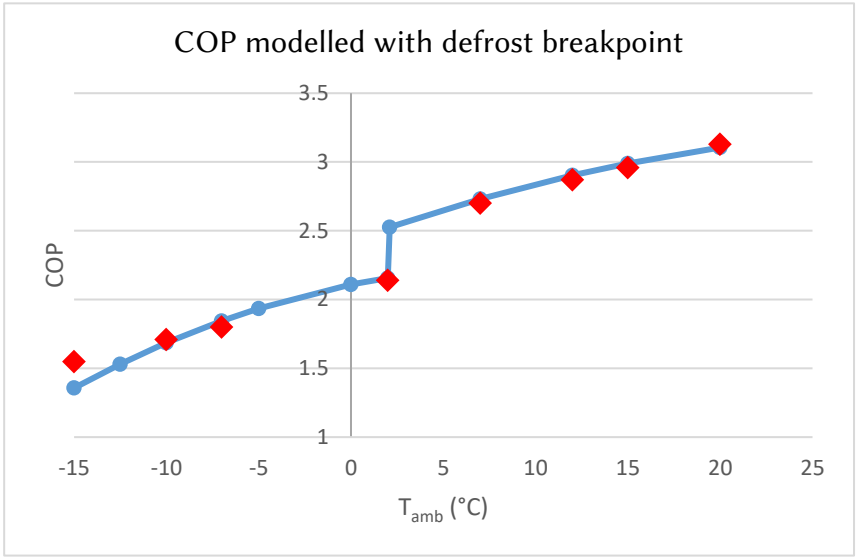


Figure 17: Modelled COP curves from datasheet regression. Top chart also shows generic regression from Staffell et al (2012) for comparison (dotted line indicates results outside of the specified valid range of lift).

$$COP = \begin{cases} 5.53 + 0.125\,T_{amb} - 7.14 \times 10^{-4}\,{T_{amb}}^2 - 5.46 \times 10^{-2}\,T_{sup} \\ \quad -3.17 \times 10^{-5}\,{T_{sup}}^2 - 1.40 \times 10^{-3}\,T_{amb}\,T_{sup} & \text{for } T_{amb} > 2 \\ 3.25 + 5.54 \times 10^{-2}\;T_{amb} - 1.55 \times 10^{-3}\,{T_{amb}}^2 + 7.18 \times 10^{-3}\,T_{sup} \\ \quad -5.09 \times 10^{-4}\,{T_{sup}}^2 - 5.19 \times 10^{-4}\,T_{amb}\,T_{sup} & \text{for } T_{amb} \leq 2 \end{cases}$$

Equation 8: COP formula from regression

In order to prevent extremely high mass flow rates (which will reduce the accuracy of the model) the heat pump will not activate unless there is a greater than 5° temperature difference between the tank's heat pump draw node temperature and the outlet temperature of the heat pump.

3.5 Renewable energy generation

The model considers the three 225kW Vestas turbines of the FWP, the Three Graces, and roof-mounted PV. Generation forecasting can be implemented in such a way that it is driven entirely off weather forecast data. This allows it to be run with historic data as easily as with forecast data.

3.5.1 Roof-mounted PV

The roof was measured from supplied plans and a plausible solar PV array size was determined (Figure 18). The eastern block of studio apartments has a 34° pitched roof facing due south onto which 11 standard size (approximately 1.7x1m) panels could be arranged. The western terrace block has a 30° pitched roof facing slightly east of south (an azimuth of 163°) onto which 2 rows of 10 standard sized panels could be arranged, making space for a skylight.

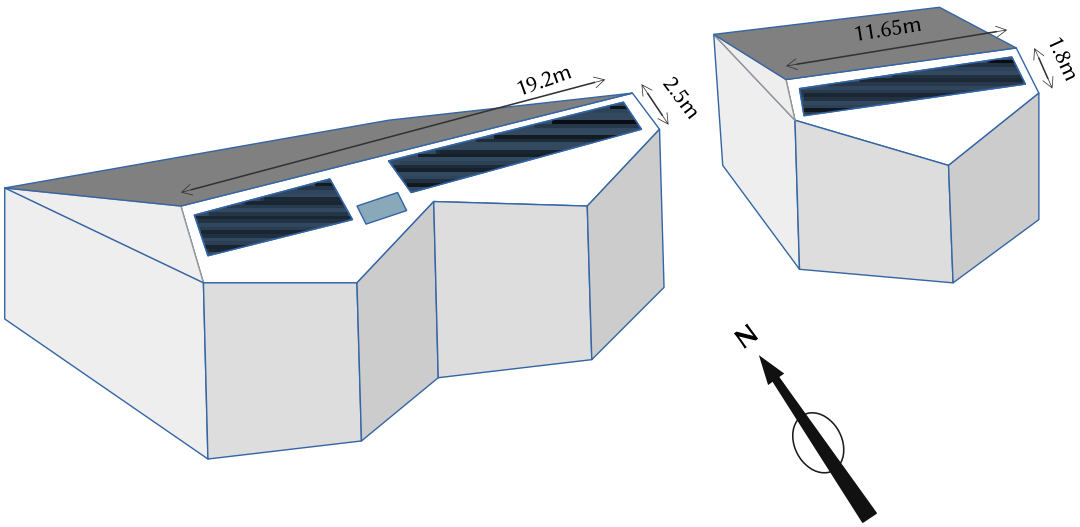


Figure 18: Plausible sizes of solar arrays on study buildings

AES Solar is a local supplier of solar powered systems (both thermal and PV) set up in 1979 by a pioneering member of the Findhorn community, Lyle Schnadt. They are a distributor of SolarEdge inverters and SunPower PV panels and have a long history of supplying to the community. Suitable models from these ranges have therefore been selected, the module being the Sunpower X22 360W panel[3]. The full specifications of the system are as shown in Table 1.

	Terrace array (W)	**Flats array (E)**
Number of panels	20 (10 x 2, landscape)	11 (11 x 1, portrait)
Nominal power	7.2kWp	3.96kWp
Tilt	30°	34°
Azimuth	163°	180°
Inverter	1x SolarEdge SE6000H (Rated at 6kVA AC)	1x SolarEdge SE3000H (Rated at 3kVA AC)

Table 1: PV array characteristics

All SolarEdge inverters allows oversizing against the total connected DC power of up to 135% - even more in their newest single-phase HD-Wave range. In locations

[3] Data specifications available at http://spectrum.sunpower.com/sites/default/files/uploads/resources/X22_360DC_RES_UK_AUS.pdf

where intense direct overhead is rarer, oversizing the inverter often has a more optimal outcome (SunPower, 2016).

It has been assumed that the usual 3.68kW upper limit specified by FWP for PV array size on the Findhorn private wire network can be circumvented for this installation.

3.5.2 Local wind turbines

The power curve for the Vestas V29 turbine is available from technical specifications. Estimates need also to be made for the turbulence caused by the local terrain, which affects the profile of wind speeds at different heights. Forecast wind speeds are usually given for a height above ground of 10m. To determine wind speeds at other heights, the Hellmann exponential law (Equation 9) is commonly used.

$$\frac{v}{v_0} = \left(\frac{H}{H_0}\right)^{\alpha}$$

Equation 9: The Hellmann exponential model for vertical wind speed profiles. v is the wind speed at height H, v_0 the speed length at height H_0 and α is the Hellmann exponent.

The local landscape is characterised by the Hellmann exponent (α as shown in Equation 9), which is derived from experimental data and reflects the roughness of the terrain. Values range from 0.1 for lakes and hardstanding to 0.4 for city areas with high rise buildings (Bañuelos-Ruedas et al, 2011). For the area around Findhorn a value of 0.2 ('tall crops, hedges and shrubs') was chosen to reflect the scrub-covered duneland around the Three Graces.

Even though the FWP allows the community to be approximately 'energy neutral' when averaged over the year, when considering instantaneous usage it is clear that local demand is not particularly well matched to wind generation. From half-hourly data obtained during the ORIGIN project for 2014-15 it was found that of the

1.282GWh generated in the year, 607MWh (47%) could not be consumed locally. The challenge with an intelligent control system would be to make better use of wind that would otherwise be exported, without further increasing the demand from local wind generation during the times where wind generation is already entirely locally consumed.

A simple and crude way of achieving this would be to set a threshold based on an average local site demand, and only allowing our scenario planner to consider wind that spills over this threshold. Figure 19 shows a frequency histogram of demand over the course of the year studied during the ORIGIN project (the year to 1st September 2015). The median half-hourly demand of 61.7kWh is also shown on the graph. Setting this as our threshold would mean that for most hours of the year our MPC controlled system will not be attempting to consume power that is already being consumed locally, and therefore should guarantee an increase in local consumption.

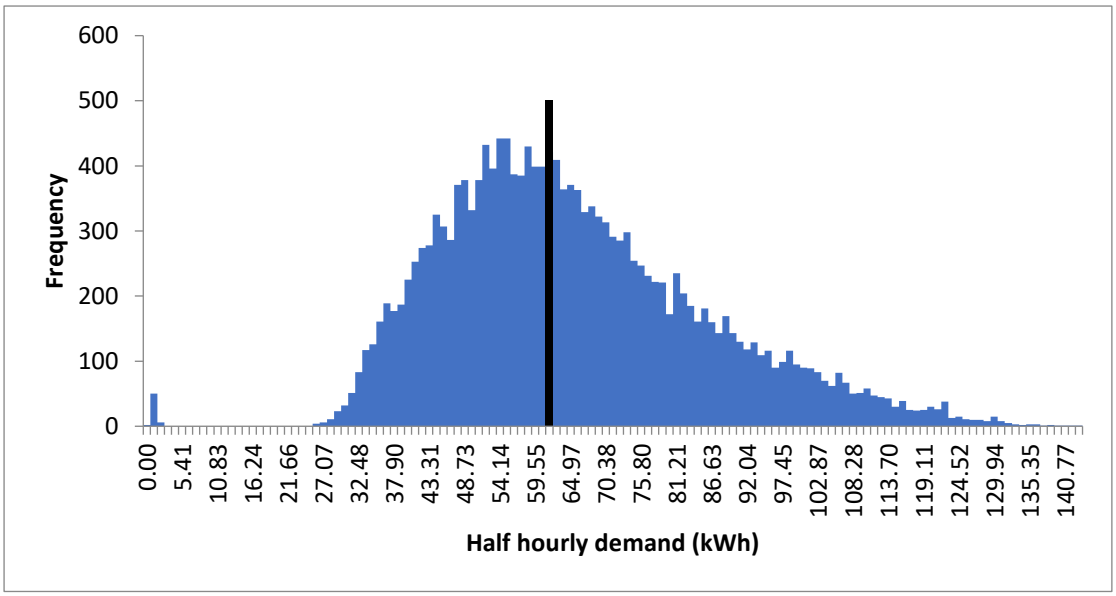


Figure 19: Frequency histogram showing half-hourly demand on the Findhorn private wire network, as measured in the ORIGIN project.

4. Development of Control System

The program flow of the MCP-controlled system is shown below in Figure 20.

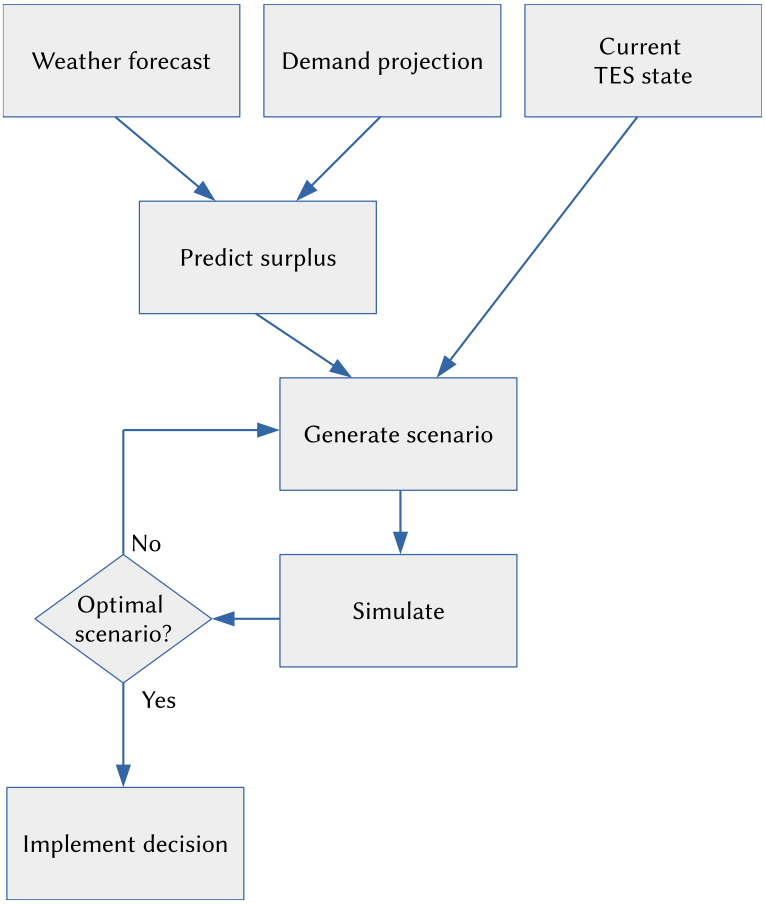


Figure 20: Process flow for MPC controller

The scheduling and optimisation algorithm in the MCP controller is a key part of the system. An hourly schedule must be devised at each decision timestep to meet the comfort criteria (the ability to provide DHW at a temperature of 38°C) with the minimum electricity consumption. This schedule is a nonlinear timeseries in which the ASHP is either 'on' or 'off'. When 'on' the ASHP will operate on a standard set-point thermostat to attempt to bring the HWT up to a temperature of 60°C.

As the time series is nonlinear, attempting to tackle it using computational optimisation is likely to be inefficient. A simpler algorithm would be to insert the minimum hours of heating required to meet comfort conditions. These hours would

be inserted into the schedule at the periods of highest available renewable energy, as shown in Figure 21.

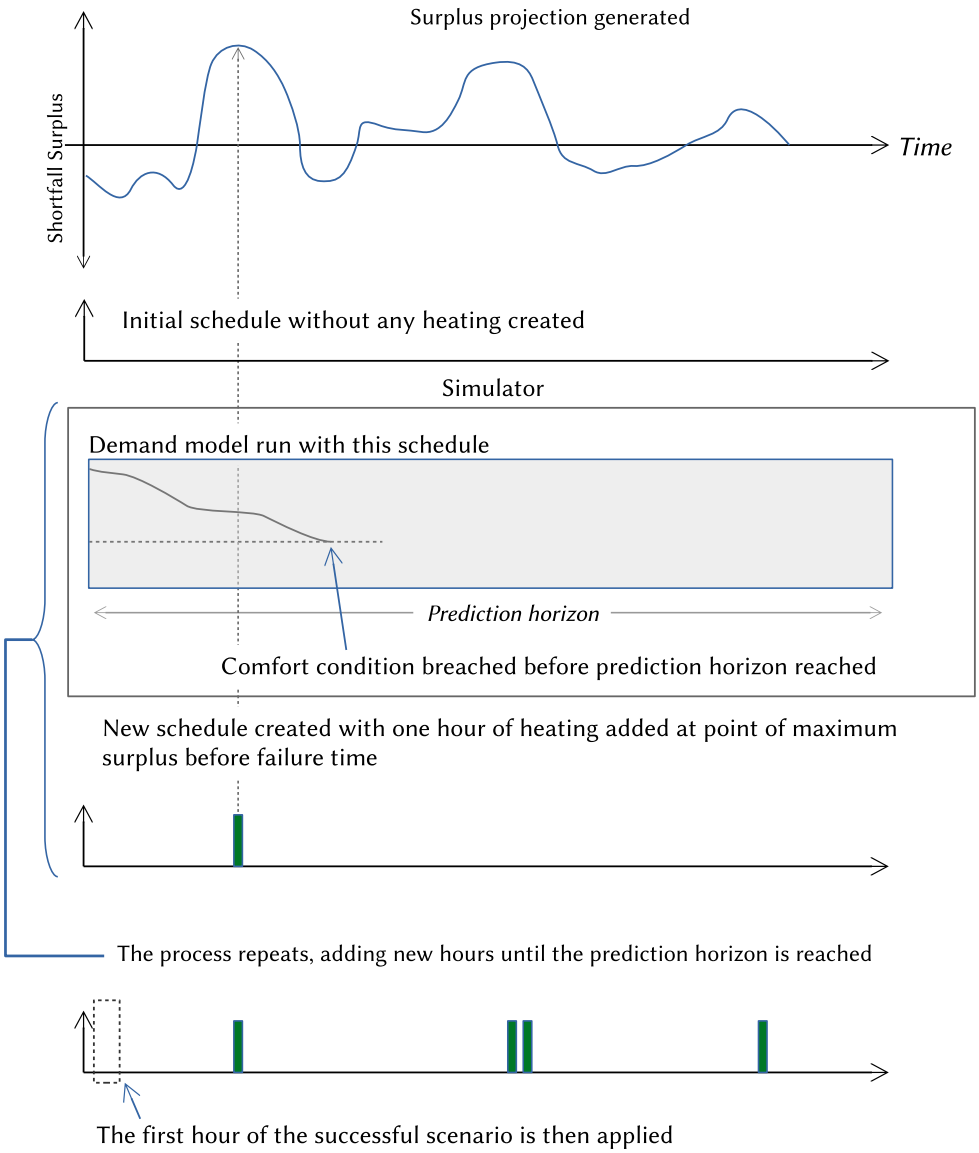


Figure 21: The scheduling algorithm

This approach minimises the uncertainty within the forecast: if generation does not match the prediction, the highest probability of generation is still likely to be around the periods of highest generation.

For the purposes of assessing the success of the algorithm the facility to add a baseline scenario will be included which will activate the heat pump on a regular daily schedule. Where this has been tested this schedule activates the tank for two off-peak hours (3am to 5am) each day.

5. Implementation in Python

The model was coded in Python 3.7 as a collection of classes, each in separate modules (.py files) to allow independent examination of different aspects of the system. The architecture of the codebase is shown in Figure 22. The codebase has been uploaded into the open-source version control hosting platform GitHub and is available at https://github.com/richplane/PyREmatcher.

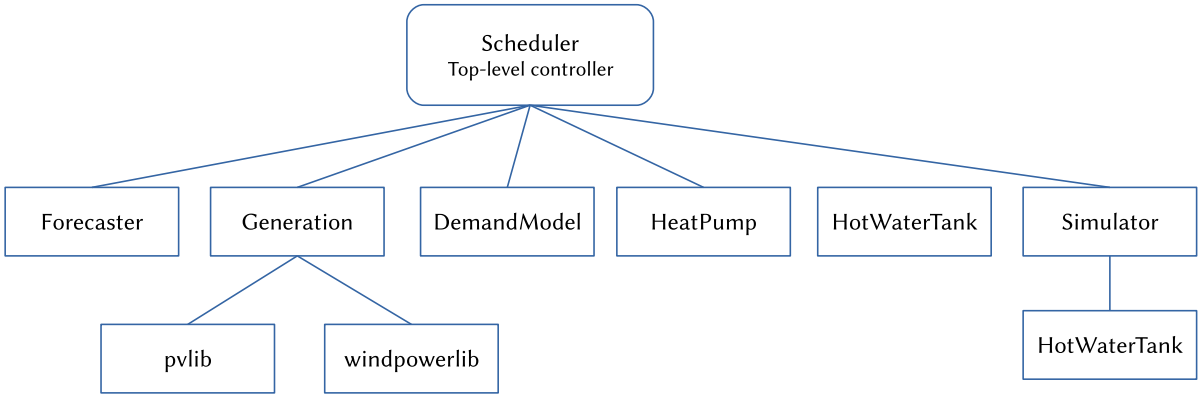


Figure 22: Top level module architecture of codebase

The model, or any part of it, can be called from an interactive Python command line or another script. In the case of most simulations in this study the model is wrapped by a testbed script which when imported as a module sets all the system characteristics and performs a single run of the scheduling model. All variables and classes are then available through this module.

All parameters that have been estimated or represent boundary conditions are accessible as class attributes, which can be set or changed by the top level controller or through the interactive Python terminal if running in command line. These are listed in the following sections, under the description of each class.

5.1. Scheduler

The top level controller defines the Scheduler class which is responsible for initiating the system parameters, holding the system state over time and

performing the simulations necessary for each timestep to arrive at a suitable scenario.

5.1.1 Class Methods

The Scheduler class has two accessible methods:

`__init__([**kwargs])`

The constructor method which creates the system simulation based on the parameters sent to it in the keyword arguments:

- `log_filename` – if supplied, the details of each simulation run will be output to this filename in a CSV format (see Appendix B for the format of this simulation output)
- `start_time` – the start time for the simulation, supplied as either a Pandas timestamp or a string that can be parsed into a timestamp. Will throw an error if it is a date in the future.
- `latitude`, `longitude`, `altitude`, `tz` (timezone) – if not passed the default values are for Findhorn
- `housing_stock` – quantities and types of all houses connected to the DHS (see Demand module)
- `minimum_temperature` – the comfort condition setting the minimum acceptable outlet temperature
- `baseline_scenario` – if supplied, the scheduler will perform an additional simulation based on a daily heating schedule for the purposes of comparison.
- `network_losses`, `pumping_energy`, `performance_factor` – all supplied as a proportion of demand – will default to 0.5, 0.01 and 1 respectively.
- `reserved_wind_power` - absolute constant value (in kW) for local consumption margin to determine surplus
- `pv_arrays` – characteristics of all PV arrays contributing to local generation (see Generation module)
- `wind_farm` – characteristics of all wind turbines contributing to local generation (see Generation module)

- `hellman_exp`, `roughness_length` – characteristics of the local wind environment (see Generation module)
- `tank_characteristics` – characteristics of the modelled TES (see Tank module)

This method then instantiates all other classes necessary for the module, loading a locally stored API key from a local file named darksky_api_key.txt to pass to the Forecast class.

`run_model(time)`

Performs the MCP algorithm to arrive at a schedule for the next 48 hours if possible. Firstly, the forecast is requested from the Forecast class. The Generation model is then used to refactor the formats and make a prediction of generation surplus. The system then creates a blank (no heating) schedule and runs the simulation based on this. Hours are then added into the schedule and the simulation rerun until a schedule is found that meets the comfort criteria over the prediction horizon. The function can also run a simulation of a baseline scenario, in which the outcome of following a fixed daily pattern of activation is simulated. The function reports on the energy consumption of the successful optimised and baseline scenarios. For example output see Appendix B. If a log file has been specified in the constructor, it is opened at the start and closed at the end of each run of the model to allow it to be read whilst the script is still active (preserving the system state). The `time` parameter (supplied as a Pandas timestamp) indicates the time that the schedule shall be devised for; if this is omitted it is run for the present hour. The system can also simulate future hours within the 2-day timeframe of a forecast already loaded into memory for the present hour – this will necessarily reduce the prediction horizon.

Two further methods are considered 'private'[4]: `_signal_heatpump(active, time)` is a placeholder for the function that in a real implementation of this code would send the command to the actuator to turn on the heat pump, but currently just writes a message to the screen. `_add_hour(failure_time)` is the prioritisation

[4] Python does not support making methods truly private, but the convention is that methods that should not under normal circumstances be accessed from outside of their class are denoted by an underscore prefix.

method that selects an hour of the (next) highest surplus energy generation for adding to the schedule.

5.1.2 Object Attributes

All the characteristics set through the constructor are accessible attributes of the object, though `start_time` is first converted into a timestamp if necessary.

5.2. Forecaster

The Forecaster class handles the retrieval of forecast data from the Dark Sky API (see section 5.8.1 below), storing it in a local folder in such a format that it can be retrieved if the forecast is required again for the same hour, avoiding a repeated call to the API.

5.2.1 Class Methods

`__init__(API_key, latitude, longitude, tz)`

The constructor merely sets the object attributes to the passed values. Again the Findhorn location and timezone are used as defaults in this class.

`get_forecast(sim_start_time)` `-> pandas.DataFrame`

This function performs the request to the Dark Sky API and constructs a 48+ hour forecast. If `sim_start_time` (in the form of a Pandas timestamp) is supplied and in the past, weather data is retrieved for that period using the Dark Sky 'Time Machine' historical data call. If the timestamp supplied is in the future an exception is raised.

Forecast data is compiled from midnight at the start of the day requested to 48 hours after the time requested. In the case of historical data this will require multiple calls; the class performs these and stitches the response data together into a single Pandas DataFrame which is then returned from the function. This is done to allow the calculation of the daily average temperature for the first day in the forecast, which is also performed here and included in the returned forecast in its own column.

If this process has already been performed for the hour requested then the forecast should be saved in a local file with a predictable filename; before any calls are made to the Dark Sky API this is checked. Correspondingly, once a forecast has been compiled from the API it is saved into a local file before being returned from the function.

The timestamp index of the forecast DataFrame becomes the common index used in the generation, demand and surplus timeseries.

There is one private method: `_call_darksky(url_suffix)`, which performs an HTTP request to the API server, returning an object converted from the JSON response. If the server does not respond an exception is raised.

5.2.2 Object Attributes

The latitude and longitude are the only intentionally accessible attributes in this class.

5.3. LocalRE

The generation module defines the LocalRE class, which wraps the two third-party renewable energy generation libraries, windpowerlib and pvlib (see section 5.8), and handles the processing of supplied Dark Sky forecasts into a format sufficient to predict the RE generation.

5.3.1 Class Methods

`__init__([wind_turbines,] [pv_arrays,] [latitude,] [longitude,] [altitude,] [roughness_length,] [hellman_exp])`

The constructor instantiates the 'model chains' for each of the generation libraries, setting up all the parameters required to predict generation from forecast data. All parameters are optional.

- `wind_turbines` – the characteristics of all local wind turbines to be modelled, supplied as a list of dictionaries. For Findhorn the list contains only one dict:

```
wind_farm = [
    {
        'name' : 'Vestas V29',
        'hub_height' : 30,          # m
        'nominal_power' : 225e3,    # W
        'rotor_diameter' : 29,      # m
        'power_curve' : pd.DataFrame(
            data={
                'value': [ p * 1000 for p in [
                    0.0, 0.0, 2.1, 7.1, 20.5, 38.3, 61.9, 92.2,
                          128, 165, 196, 216, 223, 225, 225, 0, 0
                ]],  # in W
                'wind_speed': [
                    0.0, 3.0, 3.5, 4.0, 5.0, 6.0, 7.0, 8.0,
                    9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 25.0,
                          26, 27
                ] # in m/s
            }
        ),
        'qty' : 3
    }
]
```

- `pv_arrays` – the characteristics of all local PV generation to be included in the model, again supplied as a list of dictionaries. For Findhorn the list contains two dicts:

```
pv_arrays = [
    {
        'name' : 'Terrace',
        'surface_tilt' : 30,
        'surface_azimuth' : 163,
        'surface_type' : 'grass',
        'modules_per_string' : 10,
        'strings_per_inverter' : 2,
        'module_name' : 'SunPower_SPR_X22_360_COM',
        'inverter_name' :
'SolarEdge_Technologies_Ltd___SE6000__240V__240V__CEC_2018_'
    },
    {
        'name' : 'Studios block',
        'surface_tilt' : 34,
        'surface_azimuth' : 180,
        'surface_type' : 'grass',
        'modules_per_string' : 11,
        'strings_per_inverter' : 1,
        'module_name' : 'SunPower_SPR_X22_360_COM',
        'inverter_name' :
'SolarEdge_Technologies_Ltd___SE3300__240V__240V__CEC_2018_'
    }
]
```

- `latitude`, `longitude`, `altitude` – if not passed the default values are for Findhorn
- `roughness_length`, `hellman_exp` – these characteristics are passed used in windpowerlib to characterise the local wind environment - see section 5.8.3 for details.

The constructor relies on the named PV modules and inverters being present in one of the databases of characteristics provided with pvlib (see Section 5.8.2 for more details), and will retrieve the characteristics to use in the PV modelchain.

`make_generation_forecasts(forecast)`

This calls two private functions, `_make_pv_forecast()` and `_make_wind_forecast()` which process the Dark Sky originated forecast into the DataFrame formats required by the two power generation libraries. There are differences in naming conventions and structure - windpowerlib expects a column headed 'temperature' whilst pvlib expects 'air_temp', and windpowerlib requires the columns to be multiindexed with the height above ground level for which the forecast applies. Most significantly, the code needs to generate the principle quantities necessary to calculate PV generation at the site – direct normal irradiance (DNI), direct horizontal irradiation (DHI) and global horizontal irradiance (GHI). These are not available directly in Dark Sky forecasts, so instead are calculated by combining Dark Sky's cloud coverage fraction with a clear-sky model generated by pvlib. If possible (if the required modules are available), the code will use the Ineichen and Perez clear sky model (after Ineichen and Perez, 2002) which incorporates an interpolated figure for the Linke turbidity coefficient of the atmosphere at the specified location and time. This figure, commonly denoted T_L, is the multiple that must be applied to the optical attenuation of a clear and dry atmosphere to account for the increased attenuation of clouds, water vapour and aerosols in the atmosphere at that time and location. It is determined in pvlib by interpolation from a global dataset of monthly values.

Once compiled, the forecasts are stored as object attributes `pv_forecast` and `wind_forecast`.

`predict_generation(reserved_wind_comsumption)` `-> pd.DataFrame`

Using the currently stored generation forecasts, this function runs the model chains to produce a DataFrame of generation predictions for each hour from each source. It also adds in a total column, an available_wind column (total wind generation less the supplied figure for reserved_wind_consumption, or 0 if this figure would be negative) and a surplus column (available_wind + total PV generation).

5.3.2 Object Attributes

Once instantiated, the pvlib and windpowerlib model chains are both stored as member attributes of the class (`wind_modelchain` and `pv_modelchains`), and the Location object from pvlib is stored in `pv_location` – this is used at every forecast timestep to generate the clear sky models used to calculate irradiance. The adapted forecasts are also stored as object attributes.

5.4. DemandModel

The DemandModel class deals with constructing demand models for the current housing stock based on the archetype templates derived from the BDST, which have been extracted and stored in the codebase as a 'pickled' multiindexed DataFrame file.

5.4.1 Class Methods

`__init__(houses)`

The constructor simply calls another function (the private method _get_standard_profile() to set as object attributes the initial set of daily profiles based on the supplied housing stock and the demand standard deviation values. The houses parameter is a list of dicts defining the housing types (referring to the archetypes named in the BDST), the years in which they were built and the quantity of each such house connected to the DHS. For Findhorn the value passed is:

```
housing_stock = [
    {
        'house_type' : 'Semi-detached',
        'year_built' : 2019,
```

```
        'qty' : 2
    },
    {
        'house_type' : 'Mid-terrace',
        'year_built' : 2019,
        'qty' : 2
    },
    {
        'house_type' : 'Ground-floor flat',
        'year_built' : 2019,
        'qty' : 2
    },
    {
        'house_type' : 'Top-floor flat',
        'year_built' : 2019,
        'qty' : 2
    }
]
```

`get_daily_demand(average_temp)` `-> Pandas.Series`
This function, mainly provided for debugging, returns the totalled hourly demand series for the housing stock for a day with the supplied average temperature.

`get_hourly_demand(average_temp, hour)` `-> float`
Returns the hourly demand from the stored totalled demand profile. It can accept an hour parameter in the format of a timestamp or an integer.

`predict_demand_with_margin(forecast)` `-> Pandas.Series`
This function supplies the demand profile used in the scheduling algorithm. Using the average daily temperature in the forecast data it selects the appropriate daily demand profile, adds the standard deviation of the demand over the day to each hour and returns a series matched to the timestamp index of the forecast.

In addition to the private method `_get_standard_profile()` already mentioned there is also `_get_sigmas(average_temp)` which retrieves the standard deviation for the day with the supplied average temperature.

5.4.2 Object Attributes

The profiles and standard deviations for the current housing stock are both stored locally as object attributes `profiles` and `sigmas`; both are stored as DataFrames

indexed by hour in string format ("00:00", "01:00"...) with demand profiles for daily average temperatures between -3 and 14°C in separate columns.

5.5. HeatPump

The heat pump model is chiefly used to characterise the amount of energy consumed in heating but also to set the performance limits. The model is hard coded to represent the Mitsubishi Ecodan PUHZ-HW140V, but would be easily adaptable to other systems. Unlike most of the other models in the system, the HeatPump class defined in this module is considered stateless – its operation in one timestep does not affect its operation in the next timestep in any way and it retains no data on past or future conditions.

5.5.1 Class Methods

`heatable_mass(T_in)` `-> float`

This is the function that determines how much mass could be heated from `T_in` to the outlet temperature (set as the class attribute `T_out`) in the current timestep based on the capacity of the heat pump and the maximum flow rate. This is used to set the amount of mass flowing from and to the TES in each timestep.

`deliver_heat(T_in, mass)` `-> float`

This function calculates the amount of electrical energy required by the heat pump to produce mass kg of water raised from `T_in` to `T_out`. The appropriate coefficients for current conditions are returned from a private method `_COP_coefficients()`. If the mass of water is more than the heat pump has the capacity to produce in this timestep a warning is raised. A warning is also raised if the heat required is greater than the 14kW capacity of the heat pump, though since the characteristics of the heat pump are used to determine the mass flow (from the previous function) this should not occur.

5.5.2 Object Attributes

As the object itself is stateless, any parameters relating to its performance are manipulated directly from outside of the class. `timestep` (in hours) is set upon first

instantiation, and `T_amb` is changed hourly to reflect the external conditions. Other parameters of operation are not changed during simulations: the maximum `T_out` supplied by the heatpump, `nominal_power` and `max_flow_rate`. The specific heat of water is also made a class attribute (as class constants do not exist in Python).

5.6. HotWaterTank

There are two separate instantiations of the HotWaterTank class – one that is used by the top level Scheduler class to maintain the state of the hot water tank between timesteps, and one that is used by the Simulator class to simulate future scenarios. At the start of each scenario simulation the top-level tank is deep-copied to create the starting conditions for the tank in this scenario.

The tank model was coded in such a way that it can simulate an arbitrary number of nodes (set upon instantiation of the class) to allow comparison with different studies and datasets.

5.6.1 Class Methods

`__init__(self, nodes, [**keyword_arguments])`

The constructor method sets up the tank model with the desired characteristics. The only required parameter is the number of nodes to be modelled (a minimum of 3); defaults are provided for all other keyword arguments.

- `volume` – the total tank volume, in m^3
- `diameter` – the tank diameter in m
- `height` – the tank height in m. If a volume has been supplied the diameter and height will be set based on the volume with an aspect ratio of 3 and these parameters will be ignored
- `start_node_temps` – initial temperatures for all nodes in the system, passed as a list (lowest node first)
- `outflow_node` – the node to which the outflow to the load is connected
- `heater_draw_node` – the node from which the heat pump will draw water
- `wall_U_value` – this will be used to calculate losses through the fabric of the tank

`inject_heat(mass_in, T_in)`

Instructs the tank that in the next timestep it will receive `mass_in` kg of water at `T_in` °C, drawn from the heat pump outflow node and returned by perfect reinjection above the lowest node upon which it would be buoyant (the reinjection is handled by a private method `_reinject(T_in, mass)`.

`draw_load(Q_out)`

Instructs the tank that in the next timestep it will be expected to deliver `Q_out` kWh of heat into the DHS. If the outflow node temperature is higher than the target DHS supply temperature, the amount of mass removed from the tank is calculated based on perfect mixing of return flow with tank outflow. The same amount of mass is reinjected into the system from the DHS at the return temperature.

`energy_stored()`

Mainly for debugging and inspection, this calculates a symbolic value for the energy stored in the tank at the current time based on a simple node mass × specific heat × node temperature calculation.

`process_timestep()`

This is where the matrix equation is compiled based on the inflows, outflows and current temperatures, and solved (using `linalg.solve()` from the NumPy library) to determine the tank node temperatures in the next timestep. At the end of the function all inflows and outflows are reset to zero ready for the next timestep.

`get_hp_draw_temp()`

This is a quick getter function to expose the current temperature at the nominated heat pump outflow node.

`get_outflow_temp()`

Similar to the above, this exposes the temperature at the node outflowing to the DHS.

In addition to the private method `_reinject` already mentioned, another private method is included `_mix_temps(fluids)`, which when supplied with a list of fluids, each presented as a list of `[temperature, mass]`, will calculate the resultant temperature of the perfectly mixed resultant single mass.

5.6.2 Object Attributes

The attributes of most interest are the node temperatures, stored in a list as `node_temps`. However in order to build the conditions to process each timestep, the inflows to and outflows from the tank at each node (`input_masses` and `output_masses`) need to be stored, as do the temperatures of each of these inflowing masses (`input_temps`). All are stored in lists, ordered from lowest node to highest node.

As with the heat pump model, the `timestep` (in hours) is set upon first instantiation, and `T_amb` is changed hourly to reflect the external conditions. Other parameters of operation are not changed during simulations: the target `load_supply_temp` and `load_return_temp` set for the DHS (50°C and 20°C) supplied by the heatpump, `nominal_power` and `max_flow_rate`. The specific heat capacity, density and conductance of water are made class attributes (again, as class constants do not exist in Python). The `_mass`, `_node_mass`, `_node_volume`, `_node_surface` (outer shell area), `_node_height` and `_node_area` (cross sectional) are also stored as object attributes.

5.7. Simulator

The simulator module defines the Simulator class, coded such that it can model the behaviour of the tank and heatpump in smaller timesteps than those of the forecast and demand data, the tank timestep being set as an integer division of the decision timestep. This allows higher resolution modelling of water flows in the tank and reduces the likelihood of tank nodes transferring all their mass (or more) in a single timestep.

5.7.1 Class Methods

`__init__(heatpump, minimum_temperature, tank_timestep_multiple)`

The constructor is passed the heat pump model as instantiated by the Scheduler class, which is deep copied to create a separate instantiation of the HeatPump class here. This allows the two models to operate on different time resolutions (if necessary – currently the heat pump model is only used for simulation). The `minimum_temperature` parameter sets the comfort condition for the simulation, potentially allowing it to vary at different times of the year.

`run_simulation(tank, forecast, demand, schedule, surplus, [log_file])`

```
    -> Tuple (float, float, pd.Timestamp/bool)
```

This function is responsible for running the system simulation using the provided schedule through to the prediction horizon, set by the length of the forecast DataFrame. The `demand`, `surplus` and `schedule` data are provided as a series with the same timestamp index as forecast – the schedule is simply a set of 0 or 1 values indicating the heat pump state. The `tank` object is passed each time the simulation is called and deep copied to create a clone that can run at a higher time resolution during the simulation.

The simulation then loops through the forecast series, for each timestep performing `tank_timestep`_multiple subtimesteps and then reporting on electricity consumed, and the amount of this electricity that has had to be import. The third parameter returned from the function is the `failure_time` – the timestamp at which the simulation failed to meet the comfort criteria, or a boolean False value representing the absence of failure.

5.7.2 Object Attributes

All parameters passed to the constructor are set as object attributes. The current deepcopy of the tank is also stored as an attribute.

5.8. Third party systems and libraries

The codebase makes extensive use of the Numpy and Pandas mathematical and data analysis libraries. The full list of dependencies is given in Appendix A.

5.8.1. The Dark Sky forecasting API

The Dark Sky API is an HTTP-based 'hyperlocal weather prediction' service, written in Node.JS and C, that generates both historical and forecast weather data. It powers a series of apps available on most platforms as well as their website, darksky.net. The algorithm generates a consistently structured dataset of weather conditions (including temperature, observed temperature, precipitation, wind speed, direction, gust, humidity and cloud coverage) for any location on the globe at any time by synthesising a large number of data sources, primarily doppler radar station data, and analysing it using artificial neutral networks and their own "heuristic clean-up code" (Dark Sky, 2018). Their sources include the UK Met Office's NIMROD system, the German Meteorological Office's ICON model and user-uploaded local condition data. Their forecasts are quality controlled and retrospectively checked against observed data. Use of the API requires a license key which permits up to 1,000 requests per day free of charge. This license key must is not stored within the code but is instead read from a text file 'darksky_api_key.txt' stored in the same location at the Python code.

Up to 71 hours of forecast data can be retrieved in a single call, or 48 hours of historical data. The code will stitch together datasets returned by separate calls if necessary, and all forecasts are stored locally to prevent the need for repeated API calls within the same hour. The daily temperature average is calculated alongside the calls and stored in this forecast.

5.8.2. pvlib

pvlib is an open-source PV modelling Python library principally developed by Holmgren et al (2018). It is a Python port of the PVLIB MATLAB toolbox developed at Sandia National Laboratories but has been extended to incorporate many other

models and functions. It is hosted on github at github.com/pvlib/pvlib-python. V0.6.3 (release May 2019) was used in the development of this model.

The pvlib library has a database of the characteristics of tens of thousands of PV modules and thousands of inverters, drawn from the CEC Module database, the Sandia database and the ADR Inverter database. It is also possible to define the manually define a set of characteristics for any module to be modelled by the library. pvlib determines the performance of a panel based on far more characteristics than are provided in manufacturer datasheets (including, for example, parameters determining material losses dependent on the angle of incidence), and so to avoid using assumptions based on generic data a PV module was chosen that was present in the pvlib database. The latest series of inverters were not present in the database, so earlier models were chosen as surrogates: the SolarEdge SE3300 and the SE6000 were used.

Pvlib also takes account of local conditions such as air temperature, wind speed (the module temperature is modelled based on temperature, wind speed and irradiance) and surface albedo (characterised for Findhorn using the standard figure for "grass").

5.8.3. windpowerlib

windpowerlib is pvlib's complement for determining likely generation from wind turbines. It is a fork (a specialised development branch) of the earlier feedinlib, and is maintained on GitHub at https://github.com/wind-python/windpowerlib. Many features of the library are still experimental and use of the model will display warnings indicating such. V0.1.1 (released June 2019) was used in the development of this model.

The principal inputs to the model are the characteristics of the wind farm (number of turbines, their power characteristics and location), and the weather data (wind speed, temperature, pressure and roughness length). Roughness length is incorporated as a time varying quantity within the forecast in order to model

changing sea conditions, but in this case it was modelled as a fixed value of 0.15m based on terrain-descriptive values given by Davenport et al (2000).

windpowerlib includes functions to model wake losses and smooth power curves (to account for wind speed variance across a wind farm). With a farm size of three turbines neither of these were used.

5.9 Simulation testbeds

Two simulation testbed scripts were created for the purposes of this study and have been included within the repository. These scripts instantiate the Scheduler class with all the characteristics of Findhorn and trigger the simulations.

The script findhorn_single.py simply instantiates the model for a given date, which triggers the first scheduling operation. The findhorn_multi.py script is a more rigorous test that assesses how the system deals with stochastic demand that varies unpredictably. In this script a demand series is generated based on a normalised random function which introduces a divergence from the expected demand profile. The distribution of this divergence has the same standard deviation as the daily profile. If the divergence would cause the demand to go negative, the demand is set to zero and the negative quantity is carried forward to subsequent timesteps. The result of this, over time, is a demand time sequence that has the same average value as the original profile.

To achieve this simulation of the scheduling algorithm run in 'fast forward' it was necessary to reach into the attributes of the module that would be private under intended operation. Instead of reading the tank node temperatures afresh at each decision timestep, the simulation environment runs the `draw_load()` function on the Scheduler's tank model to simulate the extraction of the stochasticised demand taking place between decision timesteps.

6. Simulation Testing

6.1. Generation

Figure 23 shows the generation forecasts for two dates in August 2019. August 1st-2nd was a period of high pressure with low wind speeds, though mostly overcast. August 14-15th was a period of generally poor weather with a weather front moving over ahead of a low pressure system.

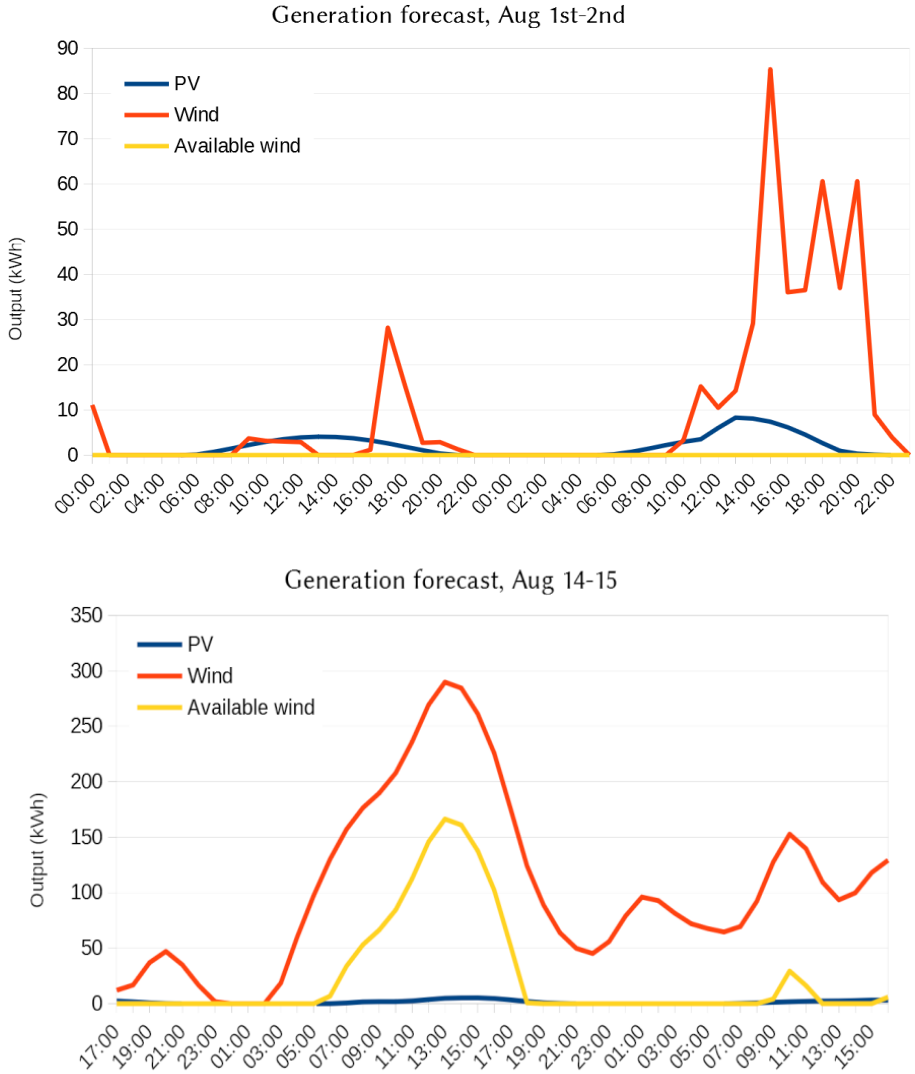


Figure 23: Generation forecasts for two dates in August 2019

A comparison between the forecasted and recorded data was also carried out as a test of the Dark Sky API. The variation of the data for August 2nd is shown in Figure 24.

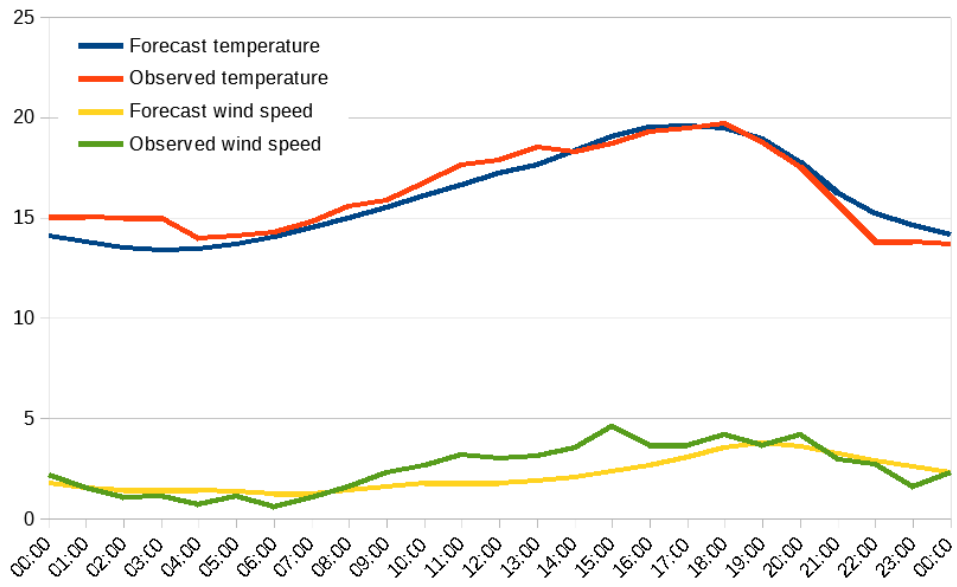


Figure 24: Forecast and observed data from the Dark Sky API for August 2nd 2019

The generation model was also compared with recorded observations available through the UK Met Office's Integrated Data Archive System (MIDAS) (Met Office, 2019), and data gathered on wind generation gathered through the ORIGIN project. Figure 25 shows the result of the comparison for a 48-hour period from November 1st 2014.

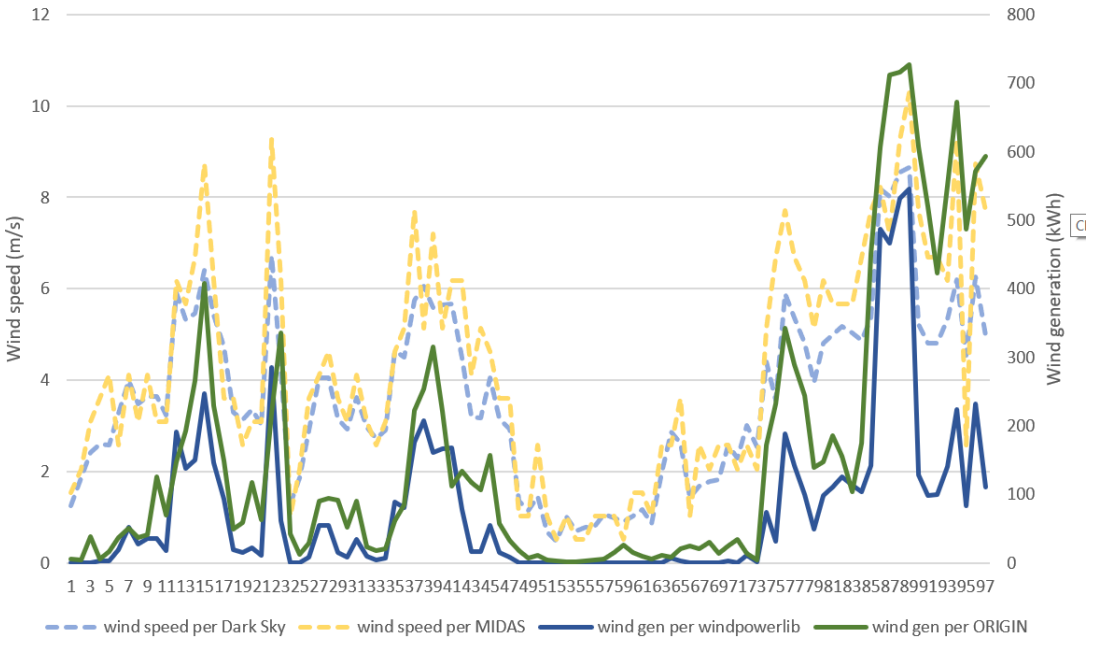


Figure 25: Wind speed from MIDAS and Dark Sky, and generation data from windpowerlib and ORIGIN compared

The MIDAS observation data shown is sourced from the Kinloss airfield, only a few hundred metres from the Three Graces turbines. The wind speeds, shown as dotted lines, display a good degree of correlation, lending confidence to the Dark Sky API wind speed forecasting algorithm. The ORIGIN-derived wind generation figures show a lower degree of correlation with the windpowerlib-sourced generation rates, with the average deviation being 62% of the ORIGIN recorded generation. The ORIGIN data correlates better with the MIDAS observation data. From this cursory examination there is reason to suspect that the combination of Dark Sky forecasts and windpowerlib modelling may underestimate generation at the site. This is as would be expected for low wind speed data, as for hours where the average speed is below cut-in, resulting in a modelled output of zero, there may still be periods where it reaches above cut-in and therefore energy in generated.

6.2. Storage model

Figure 26 shows the modelled behaviour of the TES when subjected to a demand pattern chosen at random from the BSDT for 24 hours, followed by three hours of heating at full rated power (14kW). The meteorological conditions (which in this simulation affect only the heatpump COP and the conduction losses from the TES) were taken for the 36 hours from noon on August 14th. The COP was modelled at 2.6 and the simulation was run at half-hourly intervals.

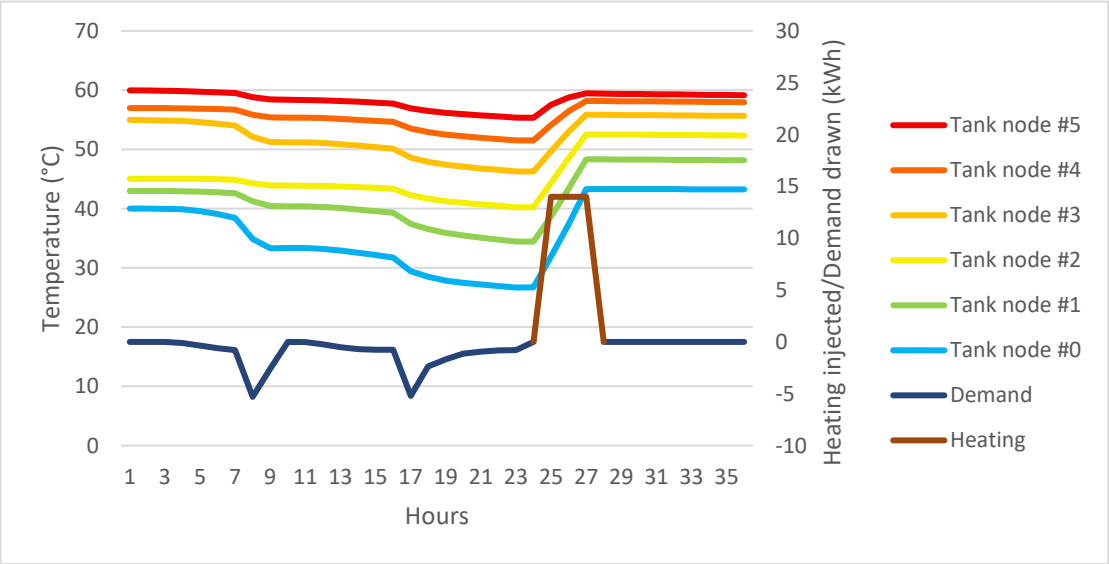


Figure 26: TES Simulation output

A key consideration in running the model is that the finer the graticulation of the tank model the better the result will be. If the heatpump drains an entire node during a timestep, the temperatures of the nodes will become inaccurate during that timestep. The heatpump is capable of producing 60°C at a flow rate of 40l/min, meaning that the volume of a tank node (258l) could be transferred in 6.45 minutes, and the entire 1,550l tank could be heated in 38 minutes. For this reason the simulations were run at 15 minute resolution. In the simulation above the maximum tank draw during any 15-minute simulation timestep was 145l.

As the ORIGIN data uses direct heating (water in the tank is directly output for consumption) it is not possible to use the data to validate the tank model.

6.3. Scheduling model

The scheduling algorithm is the part of the code tasked with selecting the most optimal hours for heating. This can be run independently using a random surplus timeseries and making repeated requests to add hours in for heating. Figure 27 illustrates how hours have been added in to the example surplus timeseries shown, starting at the highest peaks of surplus (shown in red) and working down in order.

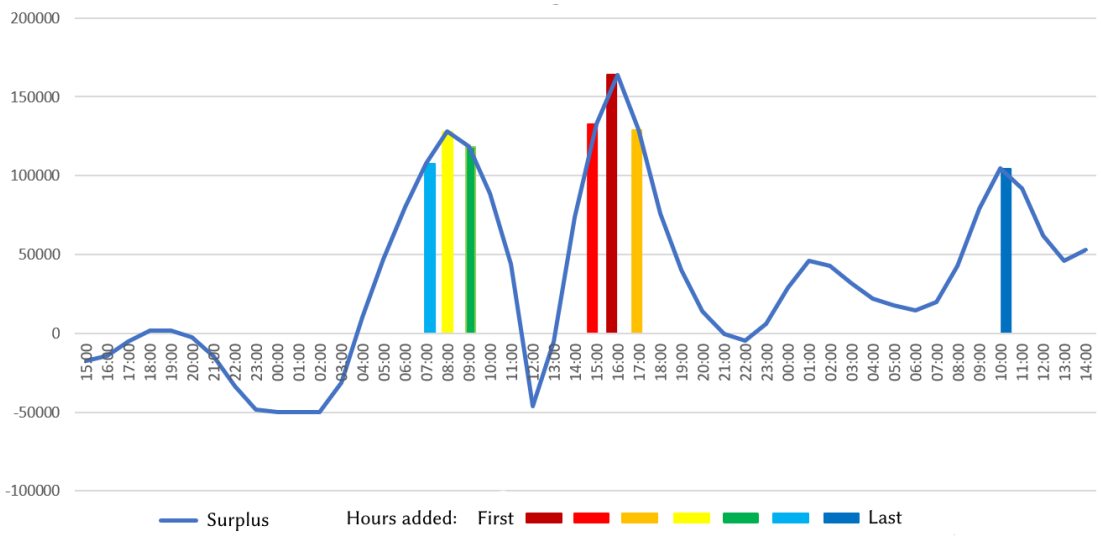


Figure 27: Demonstration output from the scheduling algorithm

6.4 Stochastic demand simulation

An algorithm is used in the simulation testbed to generate stochastic demand based on the standard profile. The output from two runs of this algorithm are shown below in Figure 28.

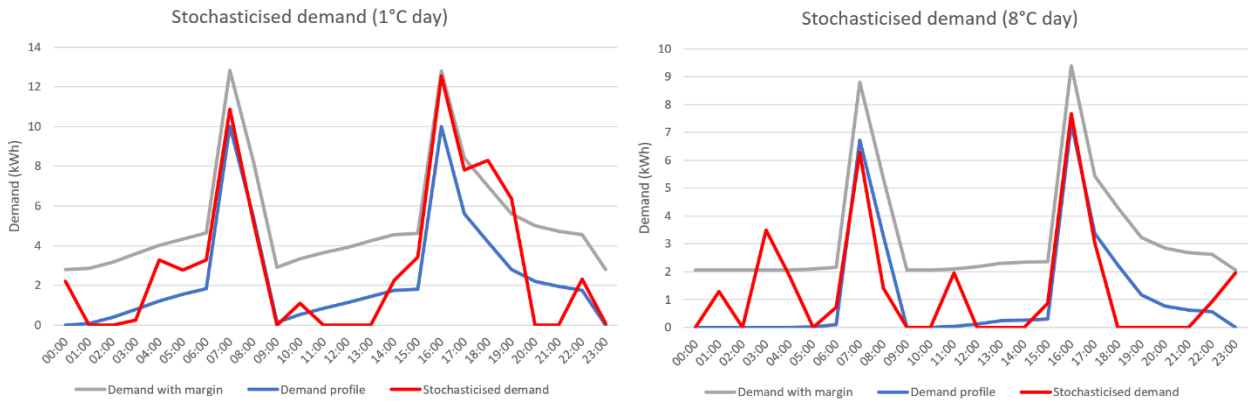


Figure 28: Examples of generated patterns of stochastic demand

The grey lines indicate the demand with margin used in the creation of the pessimistic heating schedules. The blue lines indicate the raw daily profile from the BDST and the red lines are stochasticised demand patterns generated for simulation. Over short periods as shown here the total demand may vary significantly from the average, but generating a 3-day stochasticised demand profile resulted in a change in total demand of only 0.1% - the cumulative deviation (as a percentage) is shown by the grey line in Figure 29.

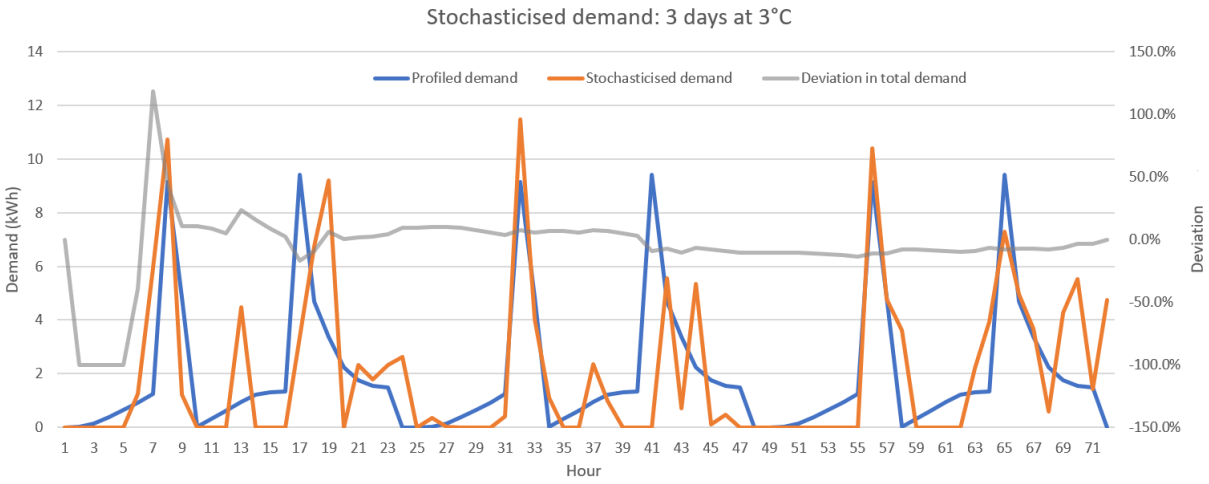


Figure 29: Three day generated demand profile showing deviation in running totals.

7. Results

7.1. Scheduling results

The scheduling simulation was run for a selection of 48 hour periods in 2019. Each time the results are compared with a baseline scenario in which heating of the hot water tank takes place for two hours daily between 3am and 5am. In each case, the total amount of energy input and imported is determined.

The results are presented in Table 2. For all simulations the tank model was working to a resolution of 12 minutes (five timesteps per hour) and the initial conditions of the tank were near fully-charged (top node temperature of 57°C).

Start time	**Scenario**	**Failure at**	**Electricity used**	**Electricity imported**
00:00, Jan 1st	Baseline		24.1kWh	12.3kWh (50.9%)
	Optimised		23.4kWh	0kWh (0%)
00:00, Feb 1st	Baseline	44 hrs	33.9kWh	23.3kWh (68.7%)
	Optimised		37.5kWh	0kWh (0%)
00:00, Mar 1st	Baseline		23.5kWh	23.6kWh (100%)
	Optimised		0kWh	0kWh
00:00, Apr 1st	Baseline		24.7kWh	24.7kWh (100%)
	Optimised		11.8kWh	0kWh (0%)
00:00, May 1st	Baseline		22.7kWh	22.7kWh (100%)
	Optimised		0kWh	0kWh
00:00, Jun 1st	Baseline		22.5kWh	16.9kWh (75.0%)
	Optimised		0kWh	0kWh

Table 2: results from single-decision-step simulations

In one case the baseline scenario was unable to run to the prediction horizon; this is not necessarily a cause for concern as the demand predictions are built to be pessimistic by the inclusion of the margin of one standard deviation (see section 3.1).

The graphs in Figures 30 and 31 show how successive scenarios are run, which the tank outlet temperatures of each scenario coloured from red to indigo. Each run includes more heating hours, until a scenario is found able to complete the 48 hour prediction horizon. Note that the lines showing energy demand and generation surplus are shown on a logarithmic scale, as the demand is usually a few orders of magnitude lower than the wind power generation. The surplus is calculated from the solar generation plus the 'spill wind' over the local consumption threshold.

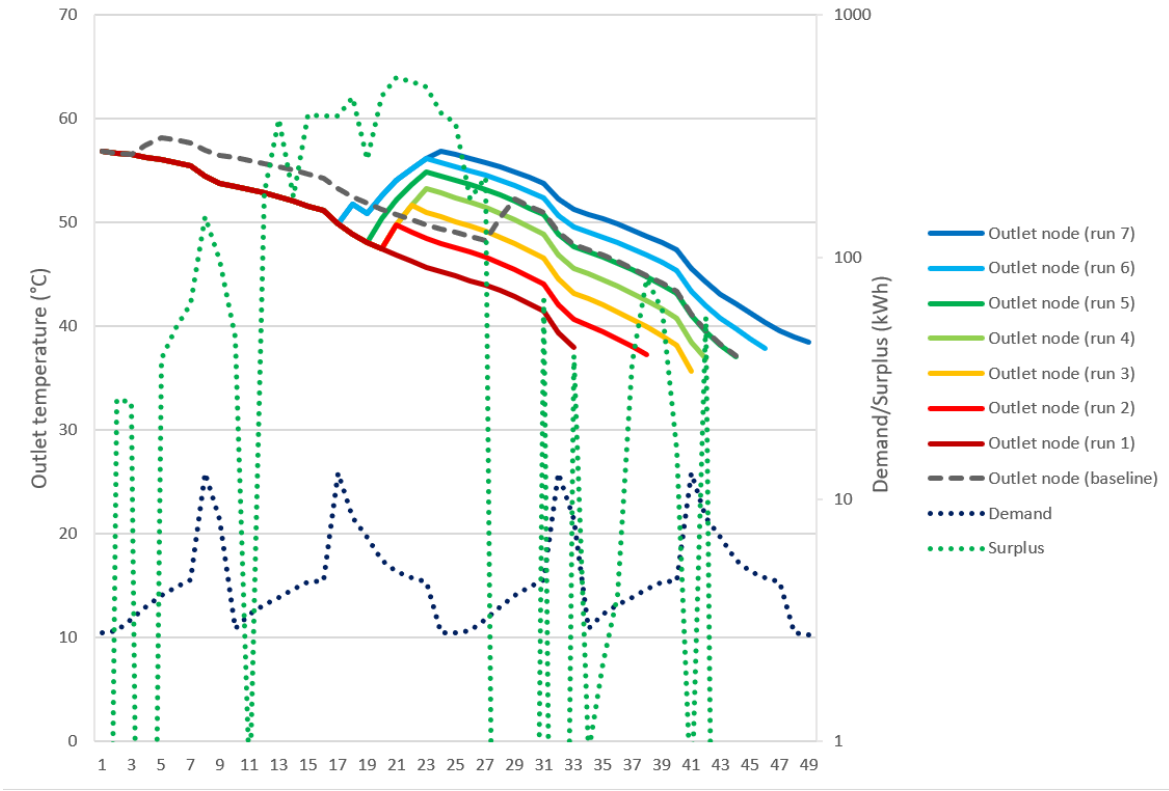


Figure 30: Scheduling simulation output for February 1st-2nd 2019. 7 runs were performed to find a successful scenario with 6 hours of heating.

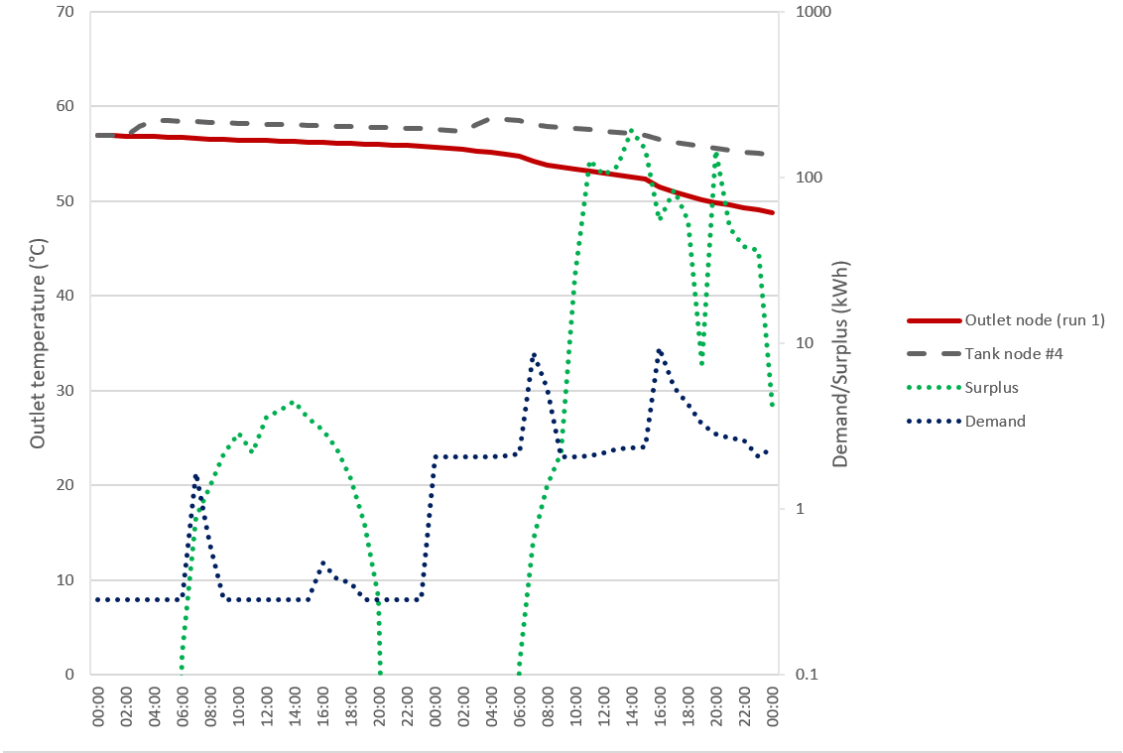


Figure 31: Simulation output for May 1st-2nd 2019. In this case no heating was needed as the system was able to 'coast' to the prediction horizon from a near-full tank.

7.2. Multiple hour simulation results

One of the strengths of the MCP paradigm is its ability to adapt to changing conditions over time. This was explored using a testbed script to simulate multiple hours of activity in which demand changed unpredictably from the expected profiles (see section 5.9 for a fuller description of the testbed).

Figure 32 shows the results of simulating 24 hours from midnight on February 1st 2019 with a stochasticised demand pattern. The successful simulations of tank outlet temperature generated at 4 hour intervals are shown – each one starts with a large dot which indicates the point at which the 'real' tank has been cloned to create the starting point for simulations run at this decision step. The outlet temperatures of the 'real' tank are shown as the dotted black line from which all the coloured simulation lines diverge. From these simulations it can be observed how the heat pump schedule adapts to the divergence of the demand drawn

(shown as a solid grey line) from the expected template demand (profile demand plus one standard deviation) modelled in the simulations (shown as a dotted blue line).

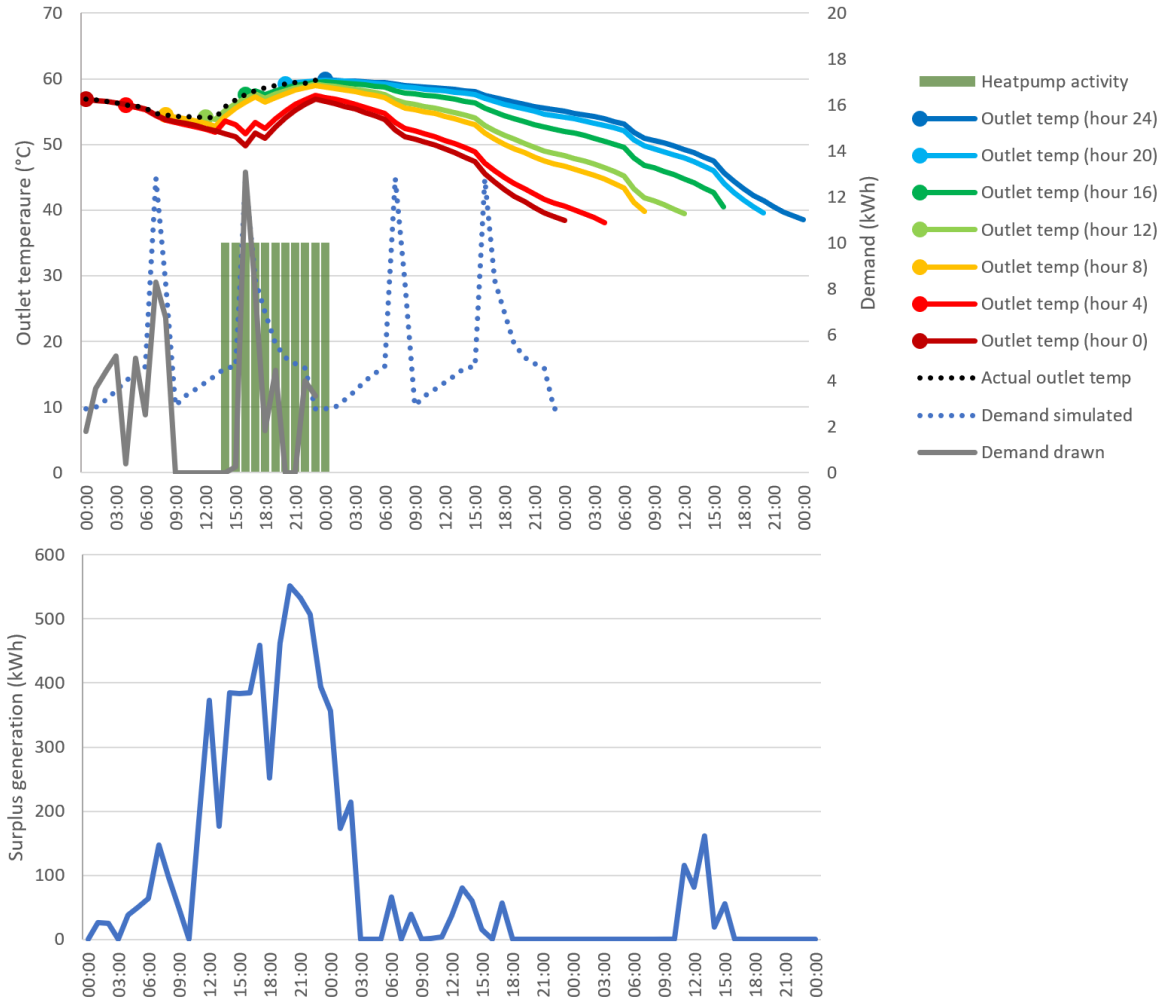


Figure 32: Output from a simulated 24-hour run of the scheduler

The first schedule can be recognised as the successful result of the single timestep modelling for February 1st, the final simulation run shown in Figure 29, which anticipates needing 6 hours of heating.

Already in the second hour, the drawn demand exceeds the simulated demand – the scheduler compensates by adding in an additional hour of heating at 14:00 of February 1st, which can be seen as the uptick at this time in the hour 4 outflow line. As time rolls forwards the MCP is looking further ahead and deciding when to

perform the heating to meet the anticipated demand taking place during the morning peak of February 3rd. There is very little generation surplus anticipated for February 2nd or 3rd, so the scheduler repeatedly adds in additional heating hours during the periods of higher surplus on February 1st in an attempt to coast for as long as possible through the period of low surplus.

8. Discussion

The results have shown that significant increases in self-consumption of local renewable resources are possible using an MPC controller. The Python code framework represents a viable control system but significant further development would be necessary before it could be implemented in a site such as North Whins.

The single-hour simulations demonstrated a clear ability to load shift to match renewable generation, with every run achieving 100% self consumption and a reduction in electricity consumption under the daily timed thermostatically controlled baseline scenario.

The multiple hour simulation showed an excellent ability to cope with variation over time whilst maintaining consumption purely of local renewables. However, it has also highlighted some shortcomings of the basic heuristic approach of the scheduling algorithm.

8.1. The scheduling algorithm

The approach taken to scheduling by this system is a very simple one which takes into account only the highest hours of local generation and makes no effort to reduce total consumption. The result is that hours of less generation – when there is still a sizable margin over the anticipated electrical demand – are often ignored, when losses (and therefore consumption) might be reduced by spacing out hours of heating more evenly.

In the 24-hour simulation performed for February, the relatively low surplus projections for the hours after 4am on February 2nd still peak at an order of magnitude higher than the anticipated electrical demand. By seizing on the periods of higher surplus on February 1st and performing as much heating as possible here, the tank is brought up to an unnecessarily high temperature. The surplus at 6am, 2pm and 3pm on February 2nd is easily sufficient to cover the ASHP consumption

without needing to import any electricity, and the tank could be allowed to coast until then, resulting in fewer losses.

A classic optimisation algorithm to address this would use a 'cost function', potentially based on a fluctuating price for grid electricity and a fixed price for local generation, to drive the scheduling algorithm. The results of this optimisation process would likely be extremely different from the approach taken, not least as instantaneous (half-hourly) energy prices can in some conditions be negative. Under these circumstances such an optimisation might also artificially increase consumption and result in higher tank losses.

The argument in favour of prioritising the highest hours is that they are the most likely periods of significant local generation. With the inherent variability of wind a significant margin of error would have to be built in to any cost function – possibly even an increasing discount on predictions further into the future to reflect their uncertainty.

8.2. Model approximations

The most significant shortcoming of the system presented here lies in the simplifications applied in the models, many of which could be addressed in future work.

8.2.1. The DHS model

The 'perfect' model of the DHS obscures many significant realities. The assumption of perfectly variable flow implies that pumps work with 100% accuracy to send only the correct amount of water to meet the demand whilst maintaining a 20°C return flow. This is impractical to achieve in the real world; not least as it would make pipes prone to freezing during periods of low demand in cold weather. Slower flows mean that water is spending longer in pipes and therefore losing more heat. The approach used here - considering system losses as a proportion of demand – does not give an accurate picture of the time-varying pattern of losses and was only ever intended in the SAP to apply as a figure aggregated on monthly or annual demand.

In real DHS systems the flow and return temperatures usually vary by season, and secondary pumps are sometimes employed to provide the range of flow rates required in different seasons (CIBSE, 2015).

8.2.2. The TES model

The perfectly uniformly insulated tank modelled in the code is an impossibility, and experimental experience (not least from the ORIGIN project) has shown it to be a poor template for real-world situations. The losses associated with the connections to the tank are more significant than those through the fabric of the tank.

As part of the ORIGIN work, the TES at West Whins was modelled with 6 nodes and losses from each node were estimated to fit with measurements. Table 3 shows these along with an approximate U-value for each node, based on the tank at West Whins being approximately 0.9m in diameter and 0.86m height, equally insulated on all sides, save for the underside which is perfectly insulated.

Node	Estimated losses from ORIGIN	Derived approximate U value
Top	0.95W/K	0.91W/m^{2}K
0.8	0.95W/K	2.33W/m^{2}K
0.6	0.9W/K	2.21 W/m^{2}K
0.4	0.45W/K	1.10W/m^{2}K
0.2	0.45W/K	1.10W/m^{2}K
Bottom	0.45W/K	1.10W/m^{2}K

Table 3: Estimated node losses for the HWT studied at West Whins as part of the ORIGIN project.

The high U value of the 0.8 node may be explained by the input from the solar collector connecting at this point.

The TES for North Whins would be significantly different in connections, size, geometry and construction, so it is difficult to extrapolate any quantitative lessons from the West Whins data.

Another omission from the multi-node model employed here is the destratification effect of conduction through the tank walls. This was found to be a significant effect by Armstrong (2015), especially in low-flow situations. The use of a stainless steel or even polystyrene tank in place of the traditional copper tank can mitigate this, without any significant effect on the colonisation of the system by Legionella and other bacteria (interestingly the study finds the biocidal effects of copper on Legionella bacteria very much overstated).

Finally, the assumption of zero turbulence or mixing caused by the reinjection of water must be highlighted as a significant shortcut.

8.2.3. The demand model

The shortcomings of the demand model with regard to standing losses have been discussed above. There is also a clear deficiency in basing heating profile demands solely on extrapolation from archetypical template profiles. Scaling these demands is overly simplistic, particularly when based on a value recommended in SAP for the purpose of estimating losses in the assessment of a wide range of house types.

In particular, when considering well-insulated buildings such as those at West and North Whins solar gain becomes as significant a factor as external temperature in predicting demand for space heating. As we are generating illuminance data for use in predicting PV generation this could readily be built into the model.

The MPC paradigm includes the ability to learn from the comparison between modelled behaviour and measured behaviour at subsequent timesteps – this is discussed more fully in section 8.4.2 below, and may be the most practical way of dealing with the unpredictable demand patterns of a particular building undergoing a particular pattern of occupation and use.

8.2.4. The heat pump model

Although less crucial to the success or failure of the model, the heat pump model provides the measure of the degree of success. Taking the nominal performance figures measured by the manufacturer to the EN14511-2013 standard is not necessarily a robust approach. The Mitsubishi databook includes 'medium performance' data (80% of maximum) and also a set of 'minimum performance' data, and notes that actual performance may vary depending on operating conditions. The assumption that the ASHP is stateless is also a simplification of the reality, which is that heat pumps, like all heating plant, do not instantaneously switch from zero operation to perfect modelled operation. This was examined by Murphy et al (2012), whose 'optimum start' algorithm for heat pump control could be built into the control model.

8.3. Implications of widespread use

It is worth considering the implications of the more widespread use of MCP-based systems. As with any kind of DSR the group behaviour of many systems is a key consideration. If independent systems designed to respond to price signals become a significant portion of demand, the system has created a feedback loop which may behave unpredictably. Kelly et al (2014) found that in responding to the price signals from the Economy 10 tariff, a population of heat pumps synchronised to create a new peak demand. Strbac (2008) and Moreau (2011) also highlight ways in which the use of price signals may lead to a reduction in the 'natural' diversity of demand.

It is usually expected that such issues will be dealt with by market intermediaries – aggregators who will develop their own control systems for managing significant amounts of DSR. However alternative approaches, such as the use of a prioritised random function have been described and assessed (for instanced by Moreau, 2011 and Ayodele et al, 2017). This is a way of artificially recreating system diversity based on the urgency of meeting a need.

Such a strategy could also be employed to tackle the issue of overcommitment of local renewables: if the dispatchable load at Findhorn were to become of the same

order of magnitude as the peak surplus energy generated by FWP, there would be a decision to be made on to whom the surplus should be made available.

8.4. Recommendations for future work

Significant improvements in the output of the code could be made with minimal additional work –more thought given to the structure and readability of both the terminal output and logging would greatly increase its usefulness to other developers. Although the system is capable of operating in the absence of a successful response from the Dark Sky API (by relying on earlier forecasts with a shortened prediction horizon), the API calling code could be made more robust with a series of retries.

Many of the shortcomings and approximations in the system could be addressed by improvements to the modelling. Some are caused by guesses made in the absence of a detailed system design and could be better addressed once more details on a proposed DHS become fixed. Others would take a significant amount of further work to address.

8.4.1. Improvements to system modelling

Some aspects of the shortcomings identified in section 8.2 above could be readily addressed in future code development. The effects of conductance of a tank wall could easily be incorporated in the TES model. There is a standard process of estimating pipe losses which could be incorporated. The efficiencies of heat exchangers coupling individual properties from a ground loop could be built in.

Modelling more realistic tank losses would require detailed further research or experimentation. The TES model developed here allows the modelling of any number of nodes and allows the heat pump and heating outlet to be connected at any node (height) in the system. Return connections are not considered as the reinjection is treated as a perfect low velocity merge into a stratified tank. To create an abstract model of tank losses responsive to different connection points is a significant undertaking. To incorporate turbulence and mixing into such a flexible

model would be a huge undertaking. A more pragmatic approach would be that adopted in the ORIGIN project, where losses are modelled on a tank-specific basis.

The ASHP model is likely to require some recoding for use with a WWSHP. Whilst the COP regression model provides a viable structure – and simplified without the need for the 'defrost breakpoint' – the source temperature will be harder to model from weather forecast data. The length of the pipe run to the site (over 500m) is such that the water temperature will be approaching that of the surrounding soil. A model will have to be constructed for this. The easiest method would be through regression from observational data.

8.4.2. Learning

Some of the shortcomings of the system models may more easily and pragmatically be addressed by harnessing the possibility for an MPC system to learn from the continual comparison of its model-based prediction with subsequent measurements. This approach could be applied to every part of the system, but the greatest benefits would be from its application to demand prediction, tank loss modelling, and generation forecasts. To achieve the first two would only require temperature sensing from tank nodes, whereas significant additional remote electric monitoring would be required to measure generation.

The ORIGIN project used a 5-week learning period to generate a dataset of demand profiles for the houses studied. The result was a model characterising the demands against time (of day), outdoor temperature and solar radiation (Tuohy et al, 2015). In the ORIGIN project there was significant scope for detecting changing circumstances (as each property was modelled individually) and interacting with occupants via an app to make decisions on observation data outside of predicted bounds. This approach could be replicated in a future development of the PyREmatcher system. The interactivity provided by the app may not be required where the diversity of a larger demand population lessens the impact of individual behaviour. A system similar to the Shewhart control rules (Yasui et al, 2006) could be employed to detect changes in behaviour based on deviation from an expected mean triggering an update in a stored demand pattern model.

Modelling the tank losses is less complex, as these would be considered time-invariant. A learning period could determine a fixed set of values from the changes of tank temperatures as compared with demand extracted and external temperature. This learning period could be re-triggered periodically to absorb any changes taking place in the DHS or TES – and a significant change would be worth detecting and flagging for investigation.

Creating a learned PV model in order to improve the generation prediction in a similar way is likely to be impractical due to the unmeasurable effects of shading. It would be unfeasibly complex to infer shading from output data and solar geometry. However, the prediction of local wind generation could be significantly improved by a learning model. One aspect that is not taken into account at all in the windpowerlib library is the wind direction of forecast wind, yet this is likely to have a significant effect on wind generation at any real location, due to landscape and obstacles present around the turbines. As wind direction data is provided with the Dark Sky API, the combined power curve for the wind farm could be artificially altered to generate a series of different power curves each for a different direction. The resultant characteristic of the wind farm would be a polar power curve response to a forecast wind – a power cone. Figure 33 illustrates the concept – the windpowerlib expects generation to be equal whichever direction the wind is coming from. In reality obstacles and terrain will result in an uneven power curve response to wind direction. By building a directional profile, the wind generation could more accurately be predicted. This would be relatively easy to achieve programmatically, but would require additional monitoring to be installed with the system to capture hourly or half hourly wind generation data.

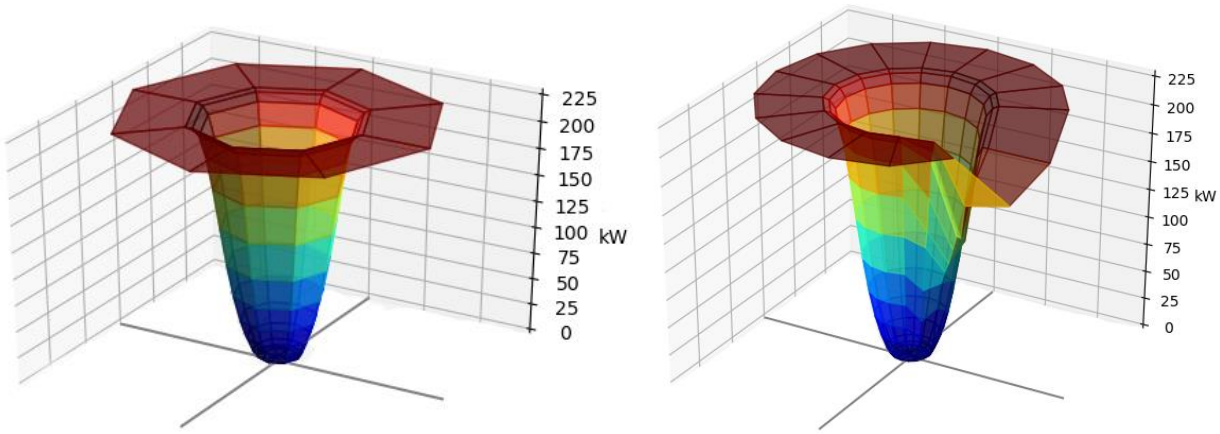


Figure 33: Wind farm "power cones"– left: the theoretical, rotationally symmetric cone; right: a hypothetical power cone showing an obstacle reducing power output from southeasterly wind.

8.4.3. Implementation of the system

The open-source nature of the Python language means that it is readily available in a range of different environments. One particular environment of interest to energy systems research is the OpenEnergyMonitor (OEM) platform: a suite of sensors, actuators and controllers based around the low-cost Raspberry Pi and Arduino systems (OEM, 2019). The central processor part of the OEM is the Raspberry Pi-based emonPi device, which runs custom made open-source software to interface with the other devices in the family and the open-source system emonCMS web app. Much of the software on the emonPi is itself Python code.

Significant development would be required to change the Python codebase from a simulator to a controller. Means of interfacing, both with actuators using the MQTT message broker service and with sensors using the device's 433MHz radio signal, would be to be incorporated and would need extensive testing. To allow monitoring of the system's performance data would have to be either routinely pushed into a cloud server or retrieved via SSH.

The codebase would need to be adapted such that its internal state is accessible for examination during runtime. In simulation this is achieved easily by running the system from an interactive prompt, but when running as an scheduled process, SSH users will not normally be able to interrogate the state of a process initiated by another user. One way to achieve this would be to use the RPyC (Remote Python Call) library[5], which provides a framework within which locally running code and locally stored data can be selectively exposed to remote users.

The local storage of the retrieved Dark Sky forecast would have limited benefit in the case of a system running a decision step on every hour, as forecasts would not then be reused. This feature could either be removed or a 'garbage collector' routine established to limit their accumulation. A logging system recording system decisions rather than full simulation run outputs would also be advisable.

[5] https://pypi.org/project/rpyc/

9. Conclusions

The model created and simulated in Python for this study has demonstrated a clear and dramatic improvement in consumption of local renewables over a timed schedule baseline. The results of the multi-hour simulation raise questions over the energy efficiency of the solution and its effect on total consumption, but insofar as this study aimed to use a MPC system to increase the consumption of local renewables it has clearly succeeded.

Whether this makes the system appropriate for use in North Whins is a deeper question which depends on the details of local circumstances. This highlights a perversity in energy systems management: whilst drives toward energy efficiency are being attempted in all industries, high levels of renewables penetration may mean that dispatchable inefficiency is entirely acceptable, or even a virtue, especially in the context of negative energy prices. It is a persistent problem of engineering: determining the objective for an approach is often more difficult that meeting it.

The principles of MPC and its efficacy in load shifting are well understood, and whilst this study has described and assessed one application in a specific context, the Python code created has the potential for wider use and has been made available on an open-source basis for further development, examination and exploration.

The incorporation of learning algorithms into the code have the potential to make it more robust, but as a 'grey-box' method it is inherently linked to a model of the system in which it is operating. Whilst efforts were made during coding to make the models as flexible as possible, this has inherent limitations. For instance, it would take significant additional development to characterise and simulate the WWSHP proposed for North Whins, or a system involving two heat pumps or a tank immersion heater.

That said, it is to be hoped that the housing developments at Findhorn are a success and are replicated far more widely within the UK as part of the ongoing efforts toward the decarbonisation of our energy sector. Such systems as low temperature heating, 4GDH networks (of all sizes), heat exchangers, high solar gain design and Passivhaus-standard insulation should become more widespread, and the opportunities for intelligent control in these systems is a hot topic. An open-source approach presents a considerable advantage in terms of interoperability and transparency.

10. References

Afram, A., Janabi-Sharifi, F. (2014). *Theory and applications of HVAC control systems – A review of model predictive control (MPC).* Building and Environment **72**, 343–355. https://doi.org/10.1016/j.buildenv.2013.11.016

Armstrong, P (2015). *Enhancing the energy storage capability of electric domestic hot water tanks.* D.Phil thesis, Oxford University.

Arteconi, A., Hewitt, N.J., Polonara, F. (2013). *Domestic demand-side management (DSM): Role of heat pumps and thermal energy storage (TES) systems.* Applied Thermal Engineering **51**, 155–165. https://doi.org/10.1016/j.applthermaleng.2012.09.023

Ayodele, T.R., Ogunjuyigbe, A.S.O., Akpeji, K.O., Akinola, O.O. (2017). *Prioritized rule based load management technique for residential building powered by PV/battery system.* Engineering Science and Technology, an International Journal **20**, 859–873. https://doi.org/10.1016/j.jestch.2017.04.003

Bañuelos-Ruedas, F., Camacho, C.A., Rios-Marcuello, S. (2011). 'Methodologies Used in the Extrapolation of Wind Speed Data at Different Heights and Its Impact in the Wind Energy Resource Assessment in a Region' in Suvire , G.O. (ed) *Wind Farm Technical Regulations, Potential Estimation and Siting Assessment.* Rijeka: InTech. Available at https://www.intechopen.com/books/wind-farm-technical-regulations-potential-estimation-and-siting-assessment (Accessed July 23rd 2019)

BRE (2016). *Standard Assessment Procedure (Draft SAP 2016).* Available at https://www.bregroup.com/sap/standard-assessment-procedure-sap-2016/ (Accessed 4th August 2019)

Buckley, R.C (2012), *Development of an energy storage tank model*, MEng thesis, University of Tennessee, Chattanooga

The Carbon Trust (2017). Biomass decision support tool. Online resource. Available at https://www.carbontrust.com/resources/tools/biomass-decision-support-tool/ (Accessed 27th June 2019)

The Chartered Institution of Building Services Engineers (CIBSE) (2015). *Heat networks: Code of Practice for the UK.* London.

Chua, K.J., Chou, S.K., Yang, W.M. (2010). *Advances in heat pump systems: A review*. Applied Energy **87**, 3611–3624. https://doi.org/10.1016/j.apenergy.2010.06.014

Committee on Climate Change (2016). *Next Steps for UK Heat Policy.* London. Available at https://www.theccc.org.uk/publication/next-steps-for-uk-heat-policy/ (Accessed: 18th July 2019)

Davenport, A.G., Grimmond, C.S.B., Oke, T.R., Wieringa, J. (2000). 'Estimating the roughness of cities and sheltered country' in *12th Applied Climatology Conference*. American Meteorological Society, Asheville, NC, 96-99

The Dark Sky Company LLC (Dark Sky) (2018). How Dark Sky Works. Blog entry, available at https://blog.darksky.net/how-dark-sky-works/ (Accessed 15th August 2019)

De Césaro Oliveski, R., Krenzinger, A., Vielmo, H.A. (2003). *Comparison between models for the simulation of hot water storage tanks.* Solar Energy **75**, 121–134. https://doi.org/10.1016/j.solener.2003.07.009

Duffie, J.A., Beckman, W.A. (2013). *Solar Engineering of Thermal Processes.* 4th ed. Hoboken, NJ: Wiley

EMD International S/A (2019), *Electric heat pumps in energyPRO*. How to Guide for Energy PRO. Available at https://www.emd.dk/files/energypro/HowToGuides/Electric%20heat%20pumps%20in%20energyPRO.pdf (Accessed 8th August 2019)

The European Guidelines Working Group (2017). *European technical guidelines for the prevention, control and investigation of infections caused by Legionella species.* Available at http://ecdc.europa.eu/en/publications-data/european-technical-guidelines-prevention-control-and-investigation-infections (Accessed 7th August 2019)

Ferguson, A., Kelly, N.J., Weber, A., Griffith, B. (2009) *Modelling residential-scale combustion-based cogeneration in building simulation.* Journal of Building Performance Simulation, 2 **1** 1-14

Findhorn Wind Park (n.d.) Technical Information. Web page http://findhornwind.co.uk/technical-blog/ (Accessed 27th July 2019)

Fischer, D., Madani, H. (2017). *On heat pumps in smart grids: A review*. Renewable and Sustainable Energy Reviews **70**, 342–357. https://doi.org/10.1016/j.rser.2016.11.182

Ghaddar, N.K., Al-Marafie, A.M., Al-Kandari, A. (1989). *Numerical simulation of stratification behaviour in thermal storage tanks.* Applied Energy **32**, 225–239. https://doi.org/10.1016/0306-2619(89)90031-7

Ghaddar, N.K. (1994). *Stratified storage tank influence on performance of solar water heating system tested in Beirut.* Renew Energy 4 **8**, 911-925

Haas, S., Schachler, B.; Krien, U.; Bosch, S. (2019). *windpowerlib v0.1.1.* Python library. Available at https://github.com/wind-python/windpowerlib (Accessed 29th May 2019)

Haslett, A (2016). *The Journey to Smarter Heat* (ETI Insights Report). Available at https://www.eti.co.uk/insights/the-journey-to-smarter-heat

Holmgren, W.F., Hansen, C. W., Mikofski, M.A. (2018). *pvlib python: a python package for modeling solar energy systems.* Journal of Open Source Software, 3 (29), 884. https://doi.org/10.21105/joss.00884

HM Treasury (2019). *Spring Statement 2019: Philip Hammond's speech.* Available at https://www.gov.uk/government/speeches/spring-statement-2019-philip-hammonds-speech (Accessed: 18th July 2019)

Ineichen, P., Perez, R. (2002) *A new airmass independent formulation for the Linke turbidity coefficient.* Solar Energy, 2002, **73** 3, 151

Kelly, N.J., Cockroft, J. (2011). *Analysis of retrofit air source heat pump performance: Results from detailed simulations and comparison to field trial data.* Energy and Buildings **43**, 239–245. https://doi.org/10.1016/j.enbuild.2010.09.018

Kelly, N.J., Tuohy, P.G., Hawkes, A.D. (2014). *Performance assessment of tariff-based air source heat pump load shifting in a UK detached dwelling featuring phase change-enhanced buffering.* Applied Thermal Engineering, Special Issue: MICROGEN III: Promoting the transition to high efficiency distributed energy systems **71**, 809–820. https://doi.org/10.1016/j.applthermaleng.2013.12.019

Lavan, Z., Thompson, J. (1977). *Experimental study of thermally stratified hot water storage tanks.* Solar Energy **19**, 519–524. https://doi.org/10.1016/0038-092X(77)90108-6

Lund, H., Werner, S., Wiltshire, R., Svendsen, S., Thorsen, J.E., Hvelplund, F., Mathiesen, B.V. (2014). *4th Generation District Heating (4GDH): Integrating smart thermal grids into future sustainable energy systems.* Energy **68**, 1–11. https://doi.org/10.1016/j.energy.2014.02.089

Lund, H., Østergaard, P.A., Connolly, D., Ridjan, I., Mathiesen, B.V., Hvelplund, F., Thellufsen, J.Z., Sorknæs, P. (2016). *Energy Storage and Smart Energy Systems.* 1 **11**, 3–14. https://doi.org/10.5278/ijsepm.2016.11.2

Millar, M.-A., Burnside, N.M., Yu, Z. (2019). *District Heating Challenges for the UK. Energies* **12**, 310. https://doi.org/10.3390/en12020310

Met Office (2019). *MIDAS Open: UK hourly weather observation data, v201901.* Centre for Environmental Data Analysis. doi:10.5285/c58c1af69b9745fda4cdf487a9547185.

Met Office (n.d.). *UK Climate Averages Data: Kinloss.* Available at https://www.metoffice.gov.uk/research/climate/maps-and-data/uk-climate-averages/gfjryyz20 (Accessed: 19th July 2019)

Met Office (2019). *MIDAS: UK Hourly Weather Observation Data. NCAS British Atmospheric Data* Available at https://catalogue.ceda.ac.uk/uuid/916ac4bbc46f7685ae9a5e10451bae7c

Mitsubishi Electric Corporation (2015). *Ecodan Renewable Heating Technology Data Book.* Available at http://www.mitsubishitech.co.uk/Data/Ecodan/Controls/PAR-WT50R-E_FTC5/FTC5_Databook.pdf (Accessed 27th July 2019)

Moreau, A. (2011). *Control Strategy for Domestic Water Heaters during Peak Periods and its Impact on the Demand for Electricity.* Energy Procedia, The Proceedings of International Conference on Smart Grid and Clean Energy Technologies (ICSGCE) 2011 **12**, 1074–1082. https://doi.org/10.1016/j.egypro.2011.10.140

Murphy, G., Counsell, J., Baster, E., Allison, J., Counsell, S. (2013). *Symbolic modelling and predictive assessment of air source heat pumps.* Building Services Engineering Research and Technology **34**, 23–39. https://doi.org/10.1177/0143624412462592

Open Energy Monitor (OEM) (2019). *OpenEnergyMonitor: Open source monitoring for understanding energy* (web site). https://openenergymonitor.org/ (Accessed August 20th 2019)

Owens, Edward. (2015). *Orchestration of Renewable Integrated Generation in Neighbourhoods: Final Report.* 10.13140/RG.2.1.2659.0320.

Sarbu, I., Sebarchievici, C. (2018). *A Comprehensive Review of Thermal Energy Storage.* Sustainability **10**, 191. https://doi.org/10.3390/su10010191

Sharma, A., Tyagi, V.V., Chen, C.R., Buddhi, D. (2009). *Review on thermal energy storage with phase change materials and applications.* Renewable and Sustainable Energy Reviews **13**, 318–345. https://doi.org/10.1016/j.rser.2007.10.005

Staffell, I., Brett, D., Brandon, N., Hawkes, A. (2012). *A review of domestic heat pumps.* Energy & Environmental Science **5**, 9291. https://doi.org/10.1039/c2ee22653g

Strbac, G. (2008). *Demand side management: Benefits and challenges.* Energy Policy, Foresight Sustainable Energy Management and the Built Environment Project 36, 4419–4426. https://doi.org/10.1016/j.enpol.2008.09.030

SolarEdge (2016). Oversizing of SolarEdge Inverters, Technical Note. Online resource, available at https://www.solaredge.com/sites/default/files/inverter_dc_oversizing_guide.pdf (Accessed August 4th 2019)

Thieblemont, H., Haghighat, F., Ooka, R., Moreau, A. (2017). *Predictive control strategies based on weather forecast in buildings with energy storage system: A review of the state-of-the art.* Energy and Buildings **153**, 485–500. https://doi.org/10.1016/j.enbuild.2017.08.010

Thorsen, J.E., Lund, H., Mathiesen, B.V. (2018). *"Progression of District Heating – 1st to 4th generation"*. Figure, available online at https://vbn.aau.dk/ws/portalfiles/portal/280710833/1_4GDH_progression_revised_May2018.pdf (Accessed: 7th August 2019)

Tuohy, P., Kim, J.-M., Samuel, A., Peacock, A., Owens, E., Dissanayake, M., Corne, D., Chaney, J., Bryden, L., Galloway, S., Stephen, B., Santonja, S., Todoli, D., (2015). *Orchestration of Renewable Generation in Low Energy Buildings and Districts Using Energy Storage and Load Shaping.* Energy Procedia, 6th International Building Physics Conference, IBPC 2015 **78**, 2172–2177. https://doi.org/10.1016/j.egypro.2015.11.311

Underwood, C.P., Royapoor, M., Sturm, B. (2017). *Parametric modelling of domestic air-source heat pumps.* Energy and Buildings **139**, 578–589. https://doi.org/10.1016/j.enbuild.2017.01.026

Vesterlund, M., Sandberg, J., Lindblom, B., Dahl, J. (2013). *Evaluation of losses in district heating system, a case study*, in: ECOS 2013: Proceedings of International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems; 2013 July 16-19; Guilin, China.

Yasui, S., Ojima, Y., Suzuki, T., 2006. Generalization of the Run Rules for the Shewhart Control Charts, in: Lenz, H.-J., Wilrich, P.-T. (Eds.), *Frontiers in Statistical Quality Control 8*. Physica-Verlag HD, Heidelberg, pp. 207–219. https://doi.org/10.1007/3-7908-1687-6_13

Yu, Z., Huang, G., Haghighat, F., Li, H., Zhang, G. (2015). *Control strategies for integration of thermal energy storage into buildings: State-of-the-art review.* Energy and Buildings, SI: IEA-ECES Annex 31 Special Issue on Thermal Energy Storage **106**, 203–215. https://doi.org/10.1016/j.enbuild.2015.05.038

Appendix A: Python codebase dependencies

The follow is not an exhaustive list as many core Python libraries are installed automatically (for instance, datetime, sys, warnings, typing and math) and most do not present any compatibility problems. However this should list all the modules that might conceivably require installation using pip or conda (or equivalents).

Module/package (& minimum version)	**Description**	**Required by**
Pandas (0.16.0)	Python Data Analysis library	All model code
NumPy (1.10.1)	Numerical Python library	Tank modelling code, Pandas, pvlib,
pvlib (0.6.3)	Open source PV modelling library	Generation.py
windpowerlib (0.1.1)	Open source wind power modelling library	Generation.py
requests (2.7.0)	Processing remote HTTP requests	Forecast.py, pvlib, windpowerlib
os	Local operating system access	Forecast.py
pytz (2017.2)	Python timezone library	pvlib, windpowerlib
six (1.5)	Python 2 and 3 compatibility utils	pvlib, tables
SciPy	Python science and engineering library	pvlib
PyTables*	Python packaging for managing hierarchical datasets	pvlib (used to access Linke turbidity database)
h5py*	Library for managing HDF5 binary data encoding format	pvlib (used to access Linke turbidity database)
mock* (2.0)	Testing library	PyTables

* indicates modules which are not required as their availability is detected before attempting to use them. This was a workaround for the inability to install some dependencies on university administered computers.

Appendix B: Example output

B.1 Simulator logfile

If a log file name is provided, the scheduler will write the outputs of all attempted scenarios to a CSV file. This CSV file has the following headings:

- "Tank node #0", "Tank node #1" ... (repeated for all nodes) – in °C
- "timestamp" – in the form "2019-04-01 00:00:00+01:00"
- "temperature" – external air temperature from the forecast
- "demand (kWh)" – the demand profile plus margin
- "energy stored (kWh)" – the nominal energy stored in the tank
- "tank draw to load (kg)" – the mass drawn from the tank to the DHS in this timestep
- "heat injected (kWh)" – the heat injected into the tank by the ASHP in this timestep
- "electricity used (kWh)" – the electricity consumed by the ASHP
- "energy surplus (kWh)" – the solar PV generation plus available wind
- "tank draw to heatpump (kg)" – the mass circulated through the heatpump and back into the tank in this timestep
- "heatpump active" – whether the heat pump is active in this hour in the current simulation

Before each simulation two lines are written giving the time at which the simulation was started followed by the first time to be simulated:

```
Scheduler instantiated at 2019-08-20 11:12:02
Simulation starting 2019-05-01 00:00:00+01:00
```

'for current hour' is substituted is the simulation is running on latest forecast data as would be the case if the code was being called by a scheduled process (cron job) in a real implementation.

After each simulation is run a report is added below the tabular data: if the run has been unsuccessful this will note "`COMFORT CONDITION BREACHED`". The successful scenario is always the final scheduling run recorded in the log. After this, the consumption is reported ("`At time 2019-01-01 00:00:00+00:00 the optimal scenario has 4 hours of heating, requiring 23.427543775744542kWh of electricity of which 0.0kWh (0.0%) was imported`"). If a baseline comparison has been requested this is then shown as a single run (headed with "`--- Baseline scenario ---`") with the consumption reported similarly afterward.

B.2 Single hour simulations

For the single hour simulations, the system was called from a Findhorn.py script which set the system characteristics and performed a single run of the scheduling algorithm. The output from running this script in an interactive prompt is shown below for a simulation starting at midnight on February 1st 2019:

```
>>> import Findhorn

Warning (from warnings module):
  File "C:\Python37\lib\site-packages\pvlib\forecast.py", line 20
    'The forecast module algorithms and features are highly
experimental. '
UserWarning: The forecast module algorithms and features are highly
experimental. The API may change, the functionality may be
consolidated into an io module, or the module may be separated into
its own package.

Warning (from warnings module):
  File "C:\Python37\lib\site-packages\windpowerlib\wake_losses.py",
line 124
    labels=[[], []]))
FutureWarning: the 'labels' keyword is deprecated, use 'codes'
instead
Running scenario: 0hours of heating
Scenario has failed at  2019-01-02 16:00:00+00:00
Running scenario: 1hours of heating
Scenario has failed at  2019-01-02 17:00:00+00:00
Running scenario: 2hours of heating
Scenario has failed at  2019-01-02 20:00:00+00:00
Running scenario: 3hours of heating
Scenario has failed at  2019-01-02 23:00:00+00:00
Running scenario: 4hours of heating
At time 2019-01-01 00:00:00+00:00 the optimal scenario has 4 hours
of heating, requiring 23.427543775744542kWh of electricity of which
0.0kWh (0.0%) was imported
```

```
At time 2019-01-01 00:00:00+00:00 the baseline scenario has 4 hours
of heating, requiring 24.12794672436572 kWh of electricity of which
12.283675338267306kWh (50.910570545410636%) was imported
At time 2019-01-01 00:00:00+00:00 the heatpump is ON
```

The two initial warnings are generated by the third party generation libraries. The successful schedule is also output inline (displayed as 'squeezed text' in the IDLE shell). In this case one of the hours of heating identified in the successful schedule includes the first hour in the simulation, which has resulted in the ON signal being sent to the heatpump.

B.3 Multiple hour simulations

The multiple hour simulations are performed by another test scaffold script which calls the scheduler's run_model() repeatedly, augmenting the log file each time and resulting in the same output to the terminal as shown above for each decision timestep.

Once completed, the test script makes its own addition to the log file to report the stochastised demands that were drawn during each decision timestep.